

RELATÓRIO TÉCNICO-CIENTÍFICO DE PROJETO DE PESQUISA

Processo 474831/2007-8

EDITAL MCT/CNPq 15/2007 – Universal

Algoritmos eficientes para resolução de problemas combinatórios

Área: Engenharia de Produção

Local de realização: Universidade Federal de Ouro Preto

Coordenador:

Prof. Dr. Marcone Jamilson Freitas Souza

Equipe:

Prof. MSc. Alexandre Xavier Martins

Prof. MSc. Euler Horta Marinho

Profa. MSc. Geiza Cristina da Silva

Prof. Dr. Luiz Henrique de Campos Merschmann

Prof. Dr. Luiz Satoru Ochi

Prof. Dr. Marcone Jamilson Freitas Souza

Profa. MSc. Tatiana Alves Costa

MSc. Túlio Ângelo Machado Toffolo

Janeiro de 2010

U F O P
UNIVERSIDADE FEDERAL DE OURO PRETO

Índice

ÍNDICE.....	2
1 INTRODUÇÃO	3
2 PRODUÇÃO CIENTÍFICA	4
2.1 ARTIGOS EM EVENTOS NACIONAIS	4
2.2 ARTIGOS EM EVENTOS INTERNACIONAIS.....	6
2.3 ARTIGOS SUBMETIDOS A PERIÓDICOS INTERNACIONAIS.....	6
2.4 DISSERTAÇÕES DE MESTRADO	7
2.5 RELATÓRIOS TÉCNICOS.....	8
3 SISTEMAS DESENVOLVIDOS	9
4 ANEXOS	10

1 Introdução

Este relatório apresenta os resultados do desenvolvimento de técnicas eficientes para resolução de problemas combinatórios. Para avaliar as técnicas desenvolvidas foram considerados os seguintes problemas:

- 1) Alocação Dinâmica de Espaços (ADE);
- 2) Diversidade Máxima (DM);
- 3) Planejamento Operacional de Lavra (POLAD);
- 4) Programação Integrada de Veículos e Tripulações no Sistema de Transporte Público (PPVT);
- 5) Seqüenciamento em Uma Máquina com penalidades por Antecipação e Atraso da produção (PSUMAA);
- 6) Roteamento de veículos com coleta e entrega simultânea (PRVCES);
- 7) Recobrimento de Rotas com Coleta de Prêmios (PRRCP)
- 8) Árvore Geradora Mínima Capacitada em Níveis (MLCMST)
- 9) Fluxo de Produção em Mineração (FPM)

O trabalho está organizado como segue. Na seção 2 é listada a produção científica gerada. Na seção 3 são enumerados os sistemas computacionais desenvolvidos. Na última seção são anexados todos os trabalhos produzidos.

2 Produção científica

Com relação à produção científica, todos os objetivos traçados foram alcançados, tendo-se, inclusive, superado a produção originalmente proposta.

2.1 Artigos em eventos nacionais

Propostos: 04

Apresentados e publicados: 11

1) [ADE-SBPO2009.pdf](#): Silva, G. C.; Boaventura, P. O; Bahiense, L.; Ochi, L. S. O problema de alocação dinâmica de espaços: aplicação das metaheurísticas GRASP e Busca Tabu. Anais do XLI Simposio Brasileiro de Pesquisa Operacional (XLI SBPO) (CD-ROM) - Porto Seguro/BA - 2009.

2) [DM-SBPO2008.pdf](#): Duarte, I. L.; Silva, G. C.; Costa, T. A. Algoritmos Heurísticos para o Problema da Diversidade Máxima. In: XL Simpósio de Pesquisa Operacional, 2008, João Pessoa. Anais do XL SBPO, 2008.

3) [POLAD-SBPO2009.pdf](#): Ribas, S.; Coelho, I. M.; Souza, M. J. F.; Menotti, D. Parallel Iterated Local Search aplicado ao planejamento operacional de lavra In: XLI Simpósio Brasileiro de Pesquisa Operacional, 2009, Porto Seguro. Anais do XLI SBPO. Rio de Janeiro: SOBRAPO, 2009. v.1. p. 2037-2047.

4) [POLAD-CBRN2009.pdf](#): Ribas, S.; Coelho, I. M., Souza, M. J. F., Coelho, V. N. Um algoritmo híbrido, baseado em GRASP, VND e Iterated Local Search para o planejamento operacional de lavra In: IX Congresso Nacional de Redes Neurais e Inteligência Computacional, 2009, Ouro Preto. Anais do IX CBRN. SBRN, 2009. v.1. 5 p.

5) [POLAD-IBRAM2008.pdf](#): Araújo, F. C. R.; Souza, M. J. F. Planejamento operacional de lavra com alocação dinâmica de caminhos: abordagens exata e heurística In: V Congresso Brasileiro de Mina a Céu Aberto, 2008, Belo Horizonte. Anais do V CBMCA. Belo Horizonte: IBRAM, 2008. v.1. 14 p.

6) [POLAD-SIMPEP2008.pdf](#): Coelho, I. M.; Ribas, S.; Souza, M. J. F. Um algoritmo baseado

em GRASP, VND e Iterated Local Search para a resolução do Problema de Planejamento Operacional de Lavra In: XV Simpósio de Engenharia de Produção, 2008, Bauru. Anais do XV SIMPEP. Bauru (SP): UNESP, 2008. v.1. 12 p.

7) [PPVT-SBPO2008.pdf](#): Souza, M. J. F., Ribas, S.; Coelho, I. M. Um algoritmo heurístico híbrido para resolução do problema de programação integrada de veículos e tripulações In: XL Simpósio Brasileiro de Pesquisa Operacional, 2008, João Pessoa (PB). Anais do XL SBPO. Rio de Janeiro: Instituto de Lógica, Filosofia e Teoria da Ciência, 2008. v.1. p.1871 – 1882.

Além da produção anterior, também foram geradas duas dissertações de mestrado tratando do tema Roteirização, com co-orientação do coordenador do presente projeto. Desses dois trabalhos foram gerados dois artigos, um dos quais ganhou o prêmio de produção acadêmica 2009, concedido pela Confederação Nacional do Transporte (CNT) e Associação Nacional de Pesquisa e Ensino em Transportes (ANPET):

8) [PRRCP-CBRN2009.pdf](#): SILVA, M. S. A.; Mine, M. T.; Ochi, L. S.; Souza, M. J. F. Um algoritmo evolutivo híbrido para o problema de recobrimento de rotas com coleta de prêmios In: IX Congresso Nacional de Redes Neurais e Inteligência Computacional, 2009, Ouro Preto. Anais do IX CBRN. SBRN, 2009. v.1. 5 p.

9) [PRVCES-ANPET2009.pdf](#): Mine, M. T.; Silva, M. S. A.; Ochi, L. S.; Souza, M. J. F. O Problema de Roteamento de Veículos com Coleta e Entrega Simultânea: Uma Abordagem Via Iterated Local Search e GENIUS. Capítulo de livro a ser publicado na série Transporte em transformação IX: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2009.

Vinculado às dissertações de mestrado orientadas ao tema Seqüenciamento em uma máquina foram publicados vários trabalhos completos, dos quais xx são listados a seguir.

10) [PSUMAA-SBPO2009.pdf](#): Rosa, B. F.; Souza, M. J. F.; Souza, S. R. Uma nova formulação de programação matemática indexada no tempo para uma classe de problemas de seqüenciamento em uma máquina. In XLI Simpósio Brasileiro de Pesquisa Operacional, 2009, Porto Seguro. Anais do XLI SBPO, SOBRAPO, p. 2898-2909.

11) [PSUMAA-CNMAC2009.pdf](#): Rosa, B. F.; Souza, S. R.; Souza, M. J. F. Formulações de Programação Matemática para o Problema de Sequenciamento em uma Máquina com Janelas

de Entrega Distintas e Tempo de Preparação Dependente da Sequência de Produção. In XXXII Congresso Nacional de Matemática Aplicada e Computacional, 2009, Cuiabá. Anais do XXXII CNMAC, SBMAC, 7 p.

2.2 Artigos em eventos internacionais

Propostos: 00 Apresentados e publicados: 03

1) [POLAD-CILAMCE2009.pdf](#): Coelho, I. M.; Ribas, S.; Souza, M. J. F., Coelho, V. N., Ochi, L. S. A hybrid heuristic algorithm based on GRASP, VND and Path Relinking for the open-pit-mining problem In: XXX Iberian Latin America Congress on Computational Methods in Engineering, 2009, Búzios (RJ). Proceedings of the XXX CILAMCE. Rio de Janeiro: UFRJ, 2009. v.1. 14p.

2) [PPVT-EngOpt2008.pdf](#): Souza, M. J. F.; Silva, G. P.; Ribas, S.; Coelho, I. M. An algorithm based on Iterated Local Search, Variable Neighborhood Descent and Tabu Search for the Integrated Vehicle and Crew Scheduling Problem In: International Conference on Engineering Optimization, 2008, Rio de Janeiro. Proceedings of the EngOpt 2008. Rio de Janeiro: UFRJ, 2008. v.1. 9 p..

3) [ADE-EngOpt2008.pdf](#): Silva, G. C.; Ferreira, T. G.; Costa, T. A.; Boaventura, P. O., and O, L. S. A Tabu Search Heuristic for the Dynamic Space Allocation Problem. Proc. of the International Conference on Engineering Optimization (EngOpt2008). Sponsoring Societies: Mathematical Programming Society (MPS), ISSMO, EUROPT, ABCM.

2.3 Artigos submetidos a periódicos internacionais

Proposto: 02 Realizado: 02

1) [POLAD-EJOR-S-09-00886](#): Souza, M. J. F.; Coelho, I. M.; Ribas, S. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. Submetido a *European Journal of Operational Research*, 2009. Em processo de revisão.

2) [MLCMST-JournalOfHeuristics2009.pdf](#): Martins, A. X.; de Souza, M. C.; Souza, M. J. F.; Toffolo, T. A. M. GRASP with hybrid heuristic-subproblem optimization for the multi-level capacitated minimum spanning tree problem. *Journal of Heuristics*, v. 15, p. 133-151, DOI: 10.1007/s10732-008-9079-x.

Encontra-se em fase final de revisão o artigo A hybrid metaheuristic algorithm for the Integrated Vehicle and Crew Scheduling Problem para submissão a periódico de circulação internacional (arquivo [PPVT-COR-2010.pdf](#))

2.4 Dissertações de mestrado

Proposta: 01

Realizada: 05

1) [POLAD--Dissertacao-Araujo-2009.pdf](#): Araújo, F. C. Planejamento operacional de lavra com alocação dinâmica de caminhões: abordagens exata e heurística. Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, 2008. Orientador: Marcone Jamilson Freitas Souza.

2) [PPVT--Dissertacao-Simoes-2009.pdf](#): Simões, E. M. L. Algoritmo para programação integrada de veículos e tripulações no sistema de transporte público por ônibus. Dissertação de mestrado, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Minas Gerais, 2009. Orientador: Geraldo Robson Mateus, Co-orientador: Marcone Jamilson Freitas Souza.

3) [FluxoProdutos--Dissertacao-Toffolo-2009.pdf](#): Toffolo, T. A. M. Otimização do fluxo de produtos em uma empresa mineradora. Dissertação de mestrado, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Minas Gerais, 2009. Orientador: Geraldo Robson Mateus, Co-orientador: Marcone Jamilson Freitas Souza.

4) [PSUMAA--Dissertacao-Penna-2009.pdf](#): Penna, P. H. V. Um algoritmo heurístico híbrido para minimizar os custos com a antecipação e o atraso da produção em ambientes com janela de entrega e tempos de preparação dependentes da sequência de produção. Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, 2009. Orientador: Marcone Jamilson Freitas Souza.

5) [PSUMAA--Dissertacao-Rosa-2009.pdf](#): Rosa, B. F. Heurísticas para o problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. Programa de Pós-Graduação em Modelagem Matemática e Computacional, Centro Federal de Educação Tecnológica – CEFET-MG, 2009. Orientador: Marcone Jamilson Freitas Souza.

2.5 Relatórios técnicos

Foram disponibilizados os seguintes relatórios técnico-científicos na página do coordenador (endereço <http://www.iceb.ufop.br/decom/prof/marcone/Publicacoes/Publicacoes.htm>):

[PPVT-RelatorioTecnico-2009.pdf](#): Souza, M. J. F. Programação integrada de veículos e tripulações de ônibus urbano. Relatório técnico-científico CNPq, processo 474831/2007-8, 2009.

[POLAD-RelatorioTecnico-2009.pdf](#): Souza, M. J. F. Planejamento operacional de lavra. Relatório técnico-científico CNPq, processo 474831/2007-8, 2009.

3 Sistemas desenvolvidos

Foram desenvolvidos os seguintes produtos computacionais, relativos aos problemas tratados:

- 1) Algoritmo GVILS, combinando as técnicas GRASP, *Variable Neighborhood Descent* e *Iterated Local Search* para resolver o Problema de Planejamento Operacional de Lavra (POLAD);
- 2) Algoritmo H-GVILS, combinando módulo de programação matemático (acionando otimizador GLPK) com as técnicas GRASP, *Variable Neighborhood Descent* e *Iterated Local Search* para resolver o Problema de Planejamento Operacional de Lavra (POLAD);
- 3) Algoritmo ILS-VND-BTRA, combinando as técnicas heurísticas *Iterated Local Search*, *Variable Neighborhood Descent* e Busca Tabu com Relaxação Adaptativa, para resolver o Problema de Programação Integrada de Veículos e Tripulações (PPVT);
- 4) *Framework* OptFrame, de código aberto, sob licença de uso GNU LGPLv3, para desenvolvimento de algoritmos de otimização (disponível em <http://sourceforge.net/projects/optframe>);
- 5) Algoritmos GRASP e TABU para resolução do problema de Alocação Dinâmica de Espaços (ADE);
- 6) Algoritmos ILS1, ILS2 e ILS3, baseados em *Iterated Local Search*, para resolução do problema da Diversidade Máxima;
- 7) Algoritmo heurístico GRASP integrado com otimizador baseado em programação matemática para resolver o problema da árvore geradora mínima capacitada em níveis (MLCMST)
- 8) Algoritmo BT-VND-PR, combinando as técnicas Busca Tabu, *Variable Neighborhood Descent* e *Path Relinking* para resolver o problema de seqüenciamento em uma máquina com penalidades por antecipação e atraso da produção.
- 9) Algoritmo GPV, combinando as técnicas GRASP, Princípio da Otimalidade Próxima e *Variable Neighborhood Descent* para resolver o problema de seqüenciamento em uma máquina com penalidades por antecipação e atraso da produção.

4 Anexos

Seguem os relatórios técnico-científicos, artigos e dissertações produzidas durante a execução do projeto.

UMA NOVA FORMULAÇÃO DE PROGRAMAÇÃO MATEMÁTICA INDEXADA NO TEMPO PARA UMA CLASSE DE PROBLEMAS DE SEQUENCIAMENTO EM UMA MÁQUINA

Bruno Ferreira Rosa

Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Programa de Pós-Graduação em Modelagem Matemática e Computacional
Av. Amazonas, 7675 - CEP 30510-000, Belo Horizonte - MG
brunofazmat@dppg.cefetmg.br

Marcone Jamilson Freitas Souza

Universidade Federal de Ouro Preto (UFOP)
ICEB, Programa de Pós-Graduação em Engenharia Mineral
CEP 35400-000, Ouro Preto - MG
marcone@iceb.ufop.br

RESUMO

Este trabalho trata do problema de sequenciamento de tarefas em uma máquina. Considera-se o tempo de preparação da máquina dependente da sequência de produção e que cada tarefa está associada a um tempo de processamento e uma janela de tempo, dentro da qual ela deve ser preferencialmente concluída. O objetivo é minimizar a soma ponderada dos atrasos e das antecipações na execução de tais tarefas. Propõe-se um modelo de programação linear inteira mista indexado no tempo para representar o problema. A estimativa do horizonte de planejamento, a qual faz parte dos dados de entrada do modelo, é obtida pela aplicação de um algoritmo heurístico baseado nos procedimentos GRASP, Princípio da Otimalidade Próxima e Descida em Vizinhança Variável. Experimentos computacionais mostram que o modelo utilizado, associado ao algoritmo heurístico proposto, permite resolver problemas maiores e com maior eficiência, quando comparado com outra formulação de programação matemática da literatura.

PALAVRAS CHAVE. Sequenciamento em uma máquina. Programação linear inteira mista. Formulação indexada no tempo. Otimização Combinatória.

ABSTRACT

This work deals with the problem of job scheduling in a single-machine. In the problem considered, the machine's setup time is sequence-dependent and each job has a processing time and a due window in which it should preferably be completed. The objective is to minimize the weighted sum of the tardiness and earliness in the execution of such jobs. A time-indexed mixed integer linear programming model is proposed to represent the problem. The estimate of the planning horizon, which is part of the model's inputs, it is obtained by an algorithm based on GRASP, Proximate Optimality Principle and Variable Neighborhood Descent. Computational experiments show that the new formulation, associated with the proposed algorithm, solves problems of larger dimension and with better efficiency, when compared with a model of the literature.

KEYWORDS. Single-machine sequencing. Mixed Integer Linear Programming. Time-indexed Formulation. Combinatorial Optimization.

1. Introdução

O surgimento do sistema de administração *Just in Time* (JIT), que ocorreu nos meados da década de 70, evidenciou a importância de um planejamento criterioso das atividades produtivas. Visando a redução dos custos provenientes de processos produtivos, a filosofia JIT desencoraja, além dos atrasos, também as antecipações das tarefas. Há antecipação de uma tarefa quando ela é concluída antes da data desejada para sua entrega e há atraso quando ela é concluída após tal data. Deste modo, ambas as situações acarretam penalidades.

Segundo Liaw (1999), concluir uma tarefa com atraso pode resultar em multas contratuais, perda de credibilidade da empresa ou redução de vendas. Do mesmo modo, concluir uma tarefa antecipadamente pode resultar em custos financeiros extras pela necessidade de disponibilização antecipada de capital, necessidade de espaço para armazenamento ou necessidade de outros recursos para manter e gerenciar o estoque (FRANÇA FILHO, 2007).

O problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção, de agora em diante denotado por PSUMAA, consiste em sequenciar e determinar o momento em que as tarefas devem ser executadas em uma máquina, com o objetivo de minimizar a soma ponderada das antecipações e dos atrasos na produção de tais tarefas. Tal problema, segundo Baker e Scudder (1990), reflete melhor ambientes de produção administrados de acordo com a filosofia JIT.

Com relação às datas de entrega das tarefas, o PSUMAA pode ser dividido em três variantes:

- i) datas de entrega comuns (*common due date*): todas as tarefas devem ser preferencialmente concluídas em uma única data pré-determinada;
- ii) datas de entrega distintas (*distinct due dates*): existe uma data de entrega específica associada a cada tarefa, na qual a tarefa deve ser preferencialmente concluída;
- iii) janelas de entrega distintas (*distinct due windows*): há um determinado período de tempo associado a cada tarefa, dentro do qual a tarefa deve ser preferencialmente concluída.

O caso de janelas de entrega distintas ocorre quando existem tolerâncias em torno das datas desejadas para a entrega de cada tarefa (KOULAMAS, 1996). Estas tolerâncias estão relacionadas às características individuais das tarefas e influenciam nos tamanhos das janelas de entrega. As tarefas concluídas dentro de suas respectivas janelas de entrega não incorrem nenhum custo. Já aquelas concluídas fora de suas janelas de entrega, são penalizadas. A produção de bens perecíveis é um exemplo relacionado. Assuma que um fabricante de produtos químicos combina certa substância A, que deteriora rapidamente, com uma segunda substância B para produzir um produto C. Se A for produzida suficientemente antes de B, ela se deteriorará. Por outro lado, se A for produzida muito depois de B, o custo da produção de C será maior.

Nas indústrias em que são produzidos diferentes tipos de produtos e existe uma troca frequente do tipo de tarefa executada em uma máquina, após a conclusão de uma tarefa, geralmente é necessário preparar a máquina antes do início da execução da tarefa seguinte. Este tempo de preparação, chamado tempo de *setup*, inclui os tempos gastos para trocar as ferramentas, preparar o material, limpar a máquina, etc. A maioria dos trabalhos em problemas de sequenciamento assume que os tempos de *setup* são independentes da sequência de produção, isto é, que eles são desprezíveis ou podem ser acrescentados aos tempos de processamento das tarefas (GUPTA e SMITH, 2006). No entanto, de acordo com Panwalkar *et al.* (1973), *apud* Gupta e Smith (2006), em grande parte das situações práticas, tais tempos são dependentes da sequência de produção. Christoforetti (2002) cita o exemplo de uma fábrica de papel que produz diversos tipos de folhas com diferentes cores, espessuras e texturas e que realiza frequentemente a preparação das máquinas para obter os diferentes tipos de produtos. Em problemas como este, os tempos de *setup* variam de acordo com sequência de produção e representam uma parcela de tempo considerável em relação ao tempo total de processamento. Portanto, eles não podem ser desconsiderados.

Considerações sobre a continuidade do funcionamento da máquina também podem ser impostas ao PSUMAA. Parte dos trabalhos da literatura não permite a inserção de tempos ociosos na sequência de produção (CHANG, 1999). Conforme Li (1997), existem casos em que o custo

por manter a máquina inativa é maior que o preço pago pela antecipação de uma tarefa e, neste caso, vale a pena antecipar a produção. Outra situação em que isto ocorre é quando a capacidade da máquina é inferior à demanda. No entanto, há casos em que vale a pena manter a máquina parada, mesmo que exista uma tarefa disponível para ser processada (SOUZA *et al.*, 2008b).

O PSUMAA possui muitas aplicações práticas em indústrias metalúrgicas, têxteis, de tintas, entre outras. O caso real de uma indústria siderúrgica é tratado por Bustamante (2006). Em tal trabalho, uma máquina é considerada como sendo uma sequência de laminadores e cada tarefa representa a produção de um determinado produto (barra chata, cantoneira, vergalhão etc.). Antes da fabricação de cada produto é necessário realizar um conjunto de ajustes de mesma natureza na sequência de laminadores. Estes ajustes dependem do produto a ser fabricado e do produto fabricado anteriormente. Como cada ajuste exige um tempo de execução, a soma destes tempos configura o tempo de *setup*.

Este trabalho trata o PSUMAA com janelas de entrega distintas, tempo de preparação dependente da sequência de produção e são permitidos tempos ociosos entre as execuções de tarefas consecutivas. Para evitar dúvidas, o PSUMAA com tais características será doravante denotado por PSUMAA-JE-TP. Dentre os trabalhos que tratam este problema, a maioria tem seu foco em procedimentos heurísticos para resolvê-lo. Apenas nos trabalhos de Bustamante (2006) e Gomes Jr. *et al.* (2007) são propostos métodos exatos para resolvê-lo.

Neste trabalho, um modelo de programação linear inteira mista indexado no tempo é utilizado para representar o PSUMAA-JE-TP. Para mensurar o horizonte de planejamento de tal modelo, propõe-se um algoritmo heurístico baseado nos métodos GRASP (FEO e RESENDE, 1995), Princípio da Otimalidade Próxima - POP (GLOVER e LAGUNA, 1997) e *Variable Neighborhood Descent* - VND (MLADENOVIC e HANSEN, 1997). O referido modelo é resolvido pelo otimizador CPLEX e comparado com outra formulação da literatura.

O restante deste trabalho está organizado como segue. Na Seção 2 são apresentados trabalhos relacionados ao PSUMAA-JE-TP. Na Seção 3 faz-se uma descrição detalhada do problema tratado. A formulação de programação matemática indexada no tempo é descrita na Seção 4 e na seguinte detalha-se o algoritmo proposto para determinar o horizonte de planejamento de tal formulação. Na Seção 6 são apresentados e discutidos os resultados computacionais, enquanto a Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

Além de possuir um grande número de aplicações práticas, problemas de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção são difíceis de ser resolvidos na otimalidade (ALLAHVERDI *et al.*, 1999). Tal dificuldade cresce explosivamente na medida em que se aumenta a quantidade de tarefas a serem sequenciadas. Esta união entre aplicabilidade e dificuldade de resolução motiva a pesquisa de algoritmos eficientes para a resolução desta classe de problemas. Deste modo, muitos trabalhos são encontrados na literatura com o objetivo de resolver casos particulares do PSUMAA-JE-TP tratado neste trabalho. Uma breve descrição de trabalhos correlatos é apresentada a seguir.

Alidaee e Dragan (1997) tratam do caso com datas de entrega comuns. Estes autores consideram as penalidades por antecipação e por atraso da produção proporcional ao tempo de processamento da tarefa e iguais para a mesma tarefa. Os autores propõem um algoritmo de complexidade $O(n \log n)$ que resolve este problema na otimalidade.

Ying (2008) e Rabadi *et al.* (2004) também trataram do PSUMAA com datas de entrega comuns. O primeiro autor resolve o problema com tempos de *setup* independentes da sequência de produção por meio de um algoritmo *branch-and-bound* que faz uso de um procedimento *Recovering Beam Search* para caminhar na árvore de busca. No segundo trabalho é proposto um algoritmo *branch-and-bound* para obter soluções do caso com tempos de *setup* dependentes da sequência de produção e com soma dos tempos de processamento das tarefas não maior que a data de entrega. Os autores resolveram problemas-teste com até 25 tarefas em tempo

computacional aceitável, o que representou um avanço para a época, já que os algoritmos exatos de então resolviam, na otimalidade, apenas problemas desta classe com até 8 tarefas.

Li (1997) estudou o PSUMAA com datas de entrega distintas, tempos de *setup* independentes da sequência de produção e sem a permissão de tempos ociosos na máquina. O problema é decomposto em dois subproblemas com estruturas mais simples. O limite inferior do problema é, então, dado pela soma dos limites inferiores dos dois subproblemas, cada qual obtido por relaxação lagrangeana. Também são desenvolvidos dois procedimentos de ajuste de multiplicadores, com complexidade $O(n \log n)$, para resolver os duais dos subproblemas. Um algoritmo *branch-and-bound* baseado nesses procedimentos é apresentado e utilizado para resolver problemas-teste com até 50 tarefas, dobrando a dimensão dos problemas resolvidos na otimalidade por algoritmos exatos até aquela data. O autor também propôs uma heurística baseada em busca local para resolver problemas-teste com até 3000 tarefas. Valente e Alves (2005) propuseram duas novas heurísticas, sendo uma regra de despacho e um procedimento guloso, para o mesmo problema. Também são utilizadas regras de dominância, visto que elas melhoram os resultados das heurísticas e demandam pouco tempo adicional.

Lee e Choi (1995) abordaram o PSUMAA com datas de entrega distintas, tempos de *setup* independentes da sequência de produção e com permissão de tempos ociosos. É apresentado um algoritmo de complexidade polinomial para determinar a data ótima de início de uma tarefa em uma dada sequência de produção. Um Algoritmo Genético que faz uso deste algoritmo de data ótima também é apresentado. O mesmo problema é estudado por Mazzini e Armentano (2001). Uma heurística construtiva que determina a sequência de produção e, simultaneamente, insere tempos ociosos é proposta por estes autores.

No trabalho de Bustamante (2006) é estudado o PSUMAA com tempos de *setup* dependentes da sequência de produção, datas de entrega distintas e é permitida a ociosidade de máquina. São desenvolvidos dois modelos de programação linear inteira mista, sendo que tais modelos exigem que os tempos de *setup* satisfaçam a desigualdade triangular. O autor também sugere alterações nos modelos que permitem resolver o caso com janelas de entrega distintas, mas os testes computacionais foram realizados apenas em problemas-teste com datas de entrega distintas. Foram resolvidos na otimalidade problemas com até 10 tarefas, por meio do *software* de otimização GLPK, versão 4.8.

Wan e Yen (2002) trataram o PSUMAA com janelas de entrega distintas, tempos de *setup* independentes da sequência de produção, permitindo tempos ociosos de máquina. Primeiramente, foi apresentada uma formulação matemática do problema junto com várias propriedades importantes para sua resolução. Em seguida, foi proposto um procedimento de complexidade polinomial para determinar a data de conclusão ótima de processamento de cada tarefa em uma dada sequência, sendo este procedimento uma extensão do algoritmo proposto por Lee e Choi (1995). Por fim, uma Busca Tabu (*Tabu Search*, TS), que faz uso do procedimento de datas ótimas, é proposta para resolver o problema. Foram realizados testes em problemas envolvendo até 80 tarefas.

Koulamas (1996) focou o problema com janelas de entrega distintas, porém com uma sensível distinção dos demais trabalhos da literatura em relação à antecipação. Em seu trabalho, há antecipação de uma tarefa quando seu processamento é iniciado antes do início de sua janela de entrega. O restante da literatura considera que há antecipação quando uma tarefa é concluída antes do início de tal janela. O autor também considerou tempos de *setup* independentes da sequência de produção, sendo permitida a ociosidade de máquina. O autor adaptou heurísticas já utilizadas em outros casos do PSUMAA e as aplicou nessa versão. Também foi proposto um algoritmo que insere tempos ociosos, de modo ótimo, em uma dada sequência de produção. Foram realizados testes em problemas com até 200 tarefas.

Gomes Jr. *et al.* (2007), Souza *et al.* (2008a) e Souza *et al.* (2008b) foram os únicos trabalhos encontrados para resolver o PSUMAA-JE-TP. No primeiro trabalho é proposto um modelo de programação matemática para representar o problema, no qual deixa de ser necessário que o problema satisfaça a desigualdade triangular. Também é proposto um método heurístico baseado em GRASP, *Iterated Local Search* e Descida em Vizinhaça Variável (*Variable Neighborhood*

Descent, VND). Para cada sequência de tarefas gerada pela heurística proposta, é utilizado um algoritmo de complexidade polinomial que determina a data ótima de início de processamento das tarefas na sequência dada, sendo este algoritmo uma extensão daquele proposto por Wan e Yen (2002). São realizados experimentos computacionais em problemas-teste com os tempos de *setup* simétricos e com até 75 tarefas, utilizando a heurística, e até 12 tarefas, utilizando o modelo matemático. O segundo trabalho apresenta uma heurística baseada em GRASP, VND, TS e Reconexão por Caminhos (*Path Relinking*, PR); enquanto no terceiro são apresentados os resultados detalhados do algoritmo proposto. Nos dois últimos trabalhos são realizados experimentos nos mesmos problemas-teste utilizados por Gomes Jr. *et al.* (2007), sendo mostrada a superioridade do algoritmo proposto em relação a este último.

3. Descrição do Problema

O PSUMAA-JE-TP abordado neste trabalho possui as seguintes características:

- i) Uma máquina deve processar um conjunto I de n de tarefas;
- ii) Associado a cada tarefa $i \in I$ está:
 - a) Um tempo de processamento P_i ;
 - b) Uma janela de entrega $[E_i, T_i]$, dentro da qual esta tarefa deve ser preferencialmente concluída;
 - c) Um custo α_i por unidade de tempo de antecipação;
 - d) Um custo β_i por unidade de tempo de atraso;
- iii) Há antecipação de uma tarefa $i \in I$ quando seu processamento é concluído antes de E_i ;
- iv) Há atraso de uma tarefa $i \in I$ quando seu processamento é concluído depois de T_i ;
- v) As tarefas que forem concluídas dentro de suas respectivas janelas de entrega não geram penalidades;
- vi) A máquina executa no máximo uma tarefa por vez e, uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção;
- vii) Todas as tarefas estão disponíveis para processamento na data 0;
- viii) Entre duas tarefas i e $j \in I$ consecutivas, é necessário um tempo S_{ij} de preparação da máquina, chamado tempo de *setup*. Assume-se que o tempo de preparação da máquina para o processamento da primeira tarefa na sequência é igual a zero;
- ix) É permitido tempo ocioso entre a execução de duas tarefas consecutivas.

O objetivo é determinar uma sequência de produção e as datas de início de produção das tarefas de sorte a minimizar a soma ponderada das antecipações e atrasos.

4. Formulação Matemática

O modelo de programação matemática utilizado para representar o PSUMAA-JE-TP é baseado na formulação indexada no tempo proposta em de Paula (2008). Este modelo, doravante denotado por MPMDT, faz uma discretização do tempo e pode ser descrito como segue.

Considere que o conjunto $H = \{h_0, h_1, h_2, \dots, h_L\}$ representa o horizonte de planejamento para o processamento das tarefas e sejam x_{ih} variáveis binárias que determinam a sequência de produção, sendo

$$x_{ih} = \begin{cases} 1, & \text{se a tarefa } i \text{ é programada pra iniciar na data } h; \\ 0, & \text{caso contrário.} \end{cases}$$

para todo $i \in I$ e para todo $h \in H$.

Se e_i e t_i representam, respectivamente, as unidades de tempo de antecipação e de tempo de atraso da tarefa $i \in I$, o MPMDT pode ser formulado pelas equações (1) a (8).

$$\min \quad z = \sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (1)$$

$$\text{s. a.} \quad x_{ih} + \sum_{u=h}^{\min(h+P_i+S_{ij}-1, H_L)} x_{ju} \leq 1, \quad \forall i, j \in I, \quad \forall h \in H \text{ e } i \neq j \quad (2)$$

$$\sum_{h=H_0}^{H_L} x_{ih} = 1, \quad \forall i \in I \quad (3)$$

$$\sum_{h=H_0}^{H_L} h x_{ih} + P_i + e_i \geq E_i, \quad \forall i \in I \quad (4)$$

$$\sum_{h=H_0}^{H_L} h x_{ih} + P_i - t_i \leq T_i, \quad \forall i \in I \quad (5)$$

$$e_i \geq 0, \quad \forall i \in I \quad (6)$$

$$t_i \geq 0, \quad \forall i \in I \quad (7)$$

$$x_{ih} \in \{0,1\}, \quad \forall i \in I \text{ e } \forall h \in H \quad (8)$$

A função objetivo, representada pela equação (1), tem como critério de otimização a minimização da soma ponderada das antecipações e dos atrasos. As restrições (2) garantem que existe tempo suficiente para executar uma tarefa i e preparar a máquina antes do início do processamento da tarefa seguinte j . As restrições (3) garantem que cada tarefa seja executada uma, e somente uma vez. As restrições (4) e (5) definem as antecipações e os atrasos de acordo com as respectivas janelas de entrega de cada tarefa. As restrições (6), (7) e (8) estão associadas aos domínios das variáveis do problema.

É importante observar que esta formulação somente é válida se o problema satisfizer a desigualdade triangular, ou seja, se as condições (9) a seguir forem satisfeitas:

$$S_{ik} \leq S_{ij} + S_{jk} + P_j, \quad \forall i, j, k \in I, \quad i \neq j, \quad i \neq k \text{ e } j \neq k \quad (9)$$

No PSUMAA-JE-TP abordado, a execução de qualquer tarefa pode ser iniciada em qualquer momento futuro. Portanto, caso se tenha a garantia de que na sequência ótima o início do processamento da primeira tarefa não ocorre antes de h_{inf} e que o início do processamento da última tarefa é no máximo h_{sup} , é conveniente adotar $H = \{h_{inf}, h_{inf+1}, h_{inf+2}, \dots, h_{sup}\}$. Deste modo, como o MPMDT é fortemente sensível à cardinalidade do conjunto H , quanto menor for o intervalo $[h_{inf}, h_{sup}]$, menor será o número de variáveis do modelo.

5. Determinação do Conjunto H

O MPMDT é fortemente dependente do conjunto $H = [h_{inf}, h_{sup}]$, visto que ele representa o horizonte de planejamento e, conseqüentemente, determina o número de variáveis e restrições deste modelo. Sendo assim, é de fundamental importância escolher tal conjunto de modo a facilitar a obtenção da solução ótima. Para tanto, a amplitude desse intervalo não deve ser muito grande, para que não se tenha muitas restrições, variáveis e um modelo mais difícil de ser resolvido na otimalidade.

Para determinar os limites do conjunto H da formulação MPMDT, desenvolveu-se o algoritmo heurístico GPV, descrito pelo pseudocódigo da Figura 1.

Este algoritmo combina os procedimentos GRASP (FEO e RESENDE, 1995), Princípio da Otimalidade Próxima - POP (GLOVER e LAGUNA, 1997) e VND (MLADENOVIC e HANSEN, 1997), e é composto de duas fases. Na primeira (linhas 1 a 12 da Figura 1), gera-se uma solução com base na metaheurística GRASP (*GRASPMax* é um parâmetro do método). Na segunda (linha 13), faz-se o pós-refinamento da solução proveniente da fase anterior. Os valores de h_{inf} e de h_{sup} (linha 14) são dados, respectivamente, pela data de início da primeira tarefa e pela

data de conclusão da última tarefa na sequência v^* . O detalhamento deste algoritmo é apresentado nas subseções seguintes.

```

Algoritmo GPV()
1   $f^* \leftarrow +\infty$ ;
2   $Iter \leftarrow 0$ ;
3  enquanto ( $Iter < GRASPM_{ax}$ ) faça
4       $Iter \leftarrow Iter + 1$ ;
5       $v_0 \leftarrow ConstruaSolucao()$ ;
6       $v \leftarrow VND_1(v_0)$ ;
7      se ( $f(v) < f^*$ ) então
8           $v^* \leftarrow v$ ;
9           $f^* \leftarrow f(v)$ ;
10          $Iter \leftarrow 0$ ;
11     fim-se;
12 fim-enquanto;
13  $v^* \leftarrow VND_2(v^*)$ ;
14 Retorne  $v^*, h_{inf}, h_{sup}$ ;
Fim GPV;

```

Figura 1: Algoritmo GPV

5.1. Representação de uma Solução

Uma solução (sequência) para o PSUMAA-JE-TP com n tarefas é representada por um vetor v de n posições, onde cada posição $i = 1, 2, \dots, n$ indica a ordem de produção da tarefa v_i . Por exemplo, dada a sequência $v = \{4, 6, 1, 5, 3, 2\}$, para o PSUMAA-JE-TP com 6 tarefas, a tarefa 4 é a primeira a ser realizada e a tarefa 2, a última.

5.2. Vizinhaça de uma Solução

Para explorar o espaço de soluções, são usados três tipos de movimentos: troca da ordem de processamento de duas tarefas da sequência de produção, realocação de uma tarefa para outra posição da sequência e realocação de um bloco com duas ou mais tarefas. Esses movimentos definem, respectivamente, as vizinhanças N^T , N^R e N^{RB} . Sendo assim, para uma solução do PSUMAA-JE-TP com n tarefas, há $n(n-1)/2$ vizinhos na vizinhança N^T , $(n-1)^2$ vizinhos na vizinhança N^R e $(n-1)(n-2)(n-3)/6$ vizinhos na vizinhança N^{RB} .

5.3. Função de Avaliação

Como os movimentos utilizados não geram soluções inviáveis, uma sequência é avaliada pela própria função objetivo, dada pela expressão (1) do MPMDT. Para determinar os valores de e_i e t_i é utilizado o algoritmo de determinação das datas ótimas de início de processamento (ADDOIP) proposto por Gomes Jr. *et al.* (2007).

5.4. Construção de uma Solução

Nesta etapa da primeira fase do algoritmo (linha 5 da Figura 1), uma solução é formada, tarefa por tarefa, de forma parcialmente gulosa, seguindo as ideias da fase de construção do algoritmo GRASP. A cada iteração, as tarefas que ainda estão fora da solução são avaliadas por uma função g , que estima o benefício associado à sua inclusão na solução parcial. As tarefas j que possuírem $g(j)$ menor ou igual a $g_{min} + \gamma \times (g_{max} - g_{min})$, com $\gamma \in [0, 1]$, são inseridas em uma Lista Restrita de Candidatos (LRC). Desta lista é escolhida aleatoriamente uma, a qual é adicionada à solução parcial.

Para estimar o benefício da inserção de cada tarefa j ainda não sequenciada, em cada iteração i , utiliza-se uma das quatro seguintes funções como critério de seleção:

- (1) $g_1(t) = 2E_j / \max\{E_k \mid k \in C\} + 2T_j / \max\{T_k \mid k \in C\} + \alpha_j / \max\{\alpha_k \mid k \in C\} - \beta_j / \max\{\beta_k \mid k \in C\} + P_j / \max\{P_k \mid k \in C\} + S_{i-1,j} / \max\{S_{i-1,k} \mid k \in C\}$, onde C representa o conjunto das tarefas ainda não sequenciadas até a i -ésima iteração (para $i = 1$, é utilizado $-m\u00e9dia\{S_{kj} \mid k \in I \text{ e } k \neq j\} / \max\{S_{kl} \mid k, l \in I \text{ e } k \neq l\}$ no lugar de $S_{i-1,j} / \max\{S_{i-1,k} \mid k \in C\}$);
- (2) $g_2(j) = T_j$;
- (3) $g_3(j) = (\alpha_j E_j + \beta_j T_j) / (\alpha_j + \beta_j)$;
- (4) $g_4(j) = E_j$.

As funções g são utilizadas conforme a ordem anterior. Nas quatro primeiras iterações, ou seja, na primeira vez em que cada função g é utilizada, faz-se $\gamma = 0$. Nas demais iterações, γ é selecionado aleatoriamente dentro de um conjunto Γ , sendo Γ um parâmetro.

Ainda na etapa de construção, é aplicado o Princípio da Otimalidade Próxima (POP). Este princípio é baseado na idéia de que boas soluções em um nível estão próximas de boas soluções em um nível adjacente (GLOVER e LAGUNA, 1997). Assim, sempre que uma nova tarefa é inserida na solução em formação, esta é submetida a uma busca local.

A busca local utilizada consiste em uma Descida Randômica em relação à vizinhança N^R . Dada uma solução, aleatoriamente escolhe-se uma tarefa na sequência e uma nova posição para ela. Se a nova sequência produzir uma solução com um valor menor para a função objetivo, a nova sequência é aceita e passa a ser a solução corrente; caso contrário, é testada outra realocação. A busca é interrompida após $MRDMax$ realocações consecutivas sem melhora na função objetivo, sendo $MRDMax$ um parâmetro do procedimento.

O pseudocódigo da fase de construção de uma solução é apresentado na Figura 2.

```

Procedimento ConstruaSolucao();
1   $v \leftarrow \emptyset$ ;
2  Inicialize o conjunto  $C$  de tarefas candidatas;
3  enquanto  $C \neq \emptyset$  faça
4       $g_{min} \leftarrow \min\{g(t) \mid t \in C\}$ ;
5       $g_{max} \leftarrow \max\{g(t) \mid t \in C\}$ ;
6       $LRC = \{t \in C \mid g(t) \leq g_{min} + \gamma(g_{max} - g_{min})\}$ ;
7      Selecione, aleatoriamente, uma tarefa  $t \in LRC$ ;
8       $v \leftarrow v \cup \{t\}$ ;
9      Atualize o conjunto  $C$  de tarefas candidatas;
10      $v \leftarrow MRD(v, N^R(\cdot), MRDMax)$ ;
11 fim-enquanto;
12 Retorne  $v$ ;
Fim ConstruaSolucao;

```

Figura 2: Procedimento para Construir uma Solução

5.5. VND₁

Para refinar as soluções geradas na primeira fase do algoritmo (linha 6 da Figura 1), utiliza-se a Busca em Vizinhança Variável (*Variable Neighborhood Descent* - VND). Proposto por Mladenovic e Hansen (1997), o VND é um método de busca local que consiste em explorar o espaço de busca por meio de trocas sistemáticas de estruturas de vizinhanças. Basicamente, o procedimento utilizado consiste em três passos:

- (1) Método Randômico de Descida (MRD) com a vizinhança N^T ;
- (2) MRD com a vizinhança N^R ;
- (3) MRD com a vizinhança N^{RB} .

Cada passo é constituído de iterações que geram vizinhos aleatórios, com relação à respectiva estrutura de vizinhança. Sempre que uma solução de melhora é encontrada, volta-se ao primeiro passo. Quando um dos MRD's atinge MRD_{max} iterações sem melhora, passa-se para o passo seguinte. Estando-se no último passo, então o procedimento é interrompido e a melhor solução encontrada é retornada.

5.6. VND_2

Como a solução proveniente da primeira fase do algoritmo GPV (Figura 1) não é necessariamente um ótimo local em relação às vizinhanças adotadas, ela é submetida a uma busca local mais efetiva, no caso, também baseada na Busca em Vizinhança Variável (VND_2). Nesta, a exploração do espaço de soluções é realizada de acordo com os seguintes passos:

- (1) Descida Completa (DC) com relação à vizinhança N^T ;
- (2) DC com relação à vizinhança N^R ;
- (3) DC na vizinhança N^{RB} .

No passo (3), em que são realizadas realocações de blocos de tarefas, inicialmente testam-se todas as realocações com blocos de duas tarefas possíveis e quando não for mais possível melhorar a solução com um determinado tamanho de bloco, passa-se a explorar movimentos com blocos de tamanho imediatamente maior. Sempre que uma solução de melhora é encontrada, volta-se ao passo (1). Se em um determinado passo, todos os vizinhos com relação à respectiva estrutura de vizinhança não são de melhora, passa-se para o passo seguinte. O procedimento é interrompido quando um ótimo local com relação às três vizinhanças é encontrado.

6. Resultados Computacionais

Para testar o modelo proposto, foram gerados problemas-teste baseados nos trabalhos de Wan e Yen (2002) e Rabadi *et al.* (2004), conforme a seguir se descreve. Dada uma tarefa i , o tempo de processamento (P_i), o custo por unidade de atraso (β_i) e o custo por unidade de antecipação (α_i) são números inteiros selecionados aleatoriamente dentro dos intervalos $[1, 40]$, $[1, 10]$ e $[1, \beta_i]$, respectivamente. O centro da janela de entrega da tarefa i é um valor inteiro aleatório no intervalo $[(1 - FA - VRJ / 2) \times TTP, (1 - FA + VRJ / 2) \times TTP]$, sendo TTP o tempo total de processamento de todas as tarefas, FA o fator de atraso e VRJ a variação relativa da janela de entrega. O tamanho da janela de entrega é um valor inteiro selecionado aleatoriamente no intervalo $[0, TTP/n]$, sendo n o número de tarefas do problema. Para toda tarefa $j \neq i$, o tempo de *setup* (S_{ij}) é um número inteiro aleatório dentro do intervalo $[5, 15]$. Desta forma, os tempos de *setup* não são necessariamente simétricos, como em Gomes Jr. *et al.* (2007).

Foram gerados conjuntos de problemas-teste com 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 e 16 tarefas, respectivamente, sendo utilizados os valores 0,1; 0,3; 0,5 e 0,8 para FA e 0,4; 0,7; 1,0 e 1,3 para VRJ . Sendo assim, há 16 problemas-teste em cada conjunto.

Se um problema gerado não satisfazia a desigualdade triangular, ele era descartado e outro com os mesmos valores para FA e VRJ era gerado. Assim, todos os problemas dessa base de dados satisfazem a desigualdade triangular.

O AMPL foi utilizado para implementar as formulações MPMDT e a de Gomes Jr. *et al.* (2007). Estes modelos de programação matemática foram resolvidos pelo otimizador CPLEX, versão 10.1, da ILOG. O algoritmo para determinação do conjunto H utilizado no MPMDT (GPV) foi implementado na linguagem C, usando-se o compilador Dev-C++, versão 2.9.9.2. Os testes foram realizados em um computador AMD Turion(tm) 64 X2 TL-58 1900 MHz, com 2 GB de RAM, sob sistema operacional Windows XP. Os parâmetros do algoritmo GPV foram calibrados empiricamente, sendo $GRASPM_{max} = 16$ e $MRDM_{max} = 7n$. O conjunto Γ utilizado é o mesmo de Gomes Jr. *et al.* (2007), ou seja, $\Gamma = \{0,02; 0,04; 0,12; 0,14\}$.

Considerou-se, ainda, o limite de 3600 segundos para o CPLEX obter cada solução. Para os

problemas em que este limite de tempo foi atingido, a solução retornada não é necessariamente ótima; porém, é retornado, pelo CPLEX, um limite inferior para o ótimo de tal problema. Este limite foi utilizado para mensurar a qualidade da solução retornada, sendo o *gap* calculado como:

$$gap = \frac{(f^{CPLEX} - L)}{L} \times 100\% \quad (10)$$

em que f^{CPLEX} e L representam, respectivamente, o valor da solução e o limite inferior encontrados pelo otimizador. Se a solução ótima é encontrada, tem-se $gap = 0$.

Os resultados encontrados para o MPMDT e para o modelo de Gomes Jr. *et al.* (2007) são resumidos na Tabela 1. Na primeira coluna indica-se o número de tarefas do grupo de problemas-teste e na segunda, o tempo médio despendido pelo algoritmo GPV para mensurar o conjunto H do MPMDT. Para cada um dos modelos são apresentadas as médias dos *gaps* encontrados e as médias dos tempos demandados pelo CPLEX.

Tabela 1: Comparação modelo proposto por Gomes Jr. *et al.* (2007) $\times$ MPMDT

Número de Tarefas	GPV	MPMDT		Gomes Jr. <i>et al.</i> (2007)	
	Tempo Médio (s)	Média dos <i>gaps</i> (%)	Tempo Médio (s)	Média dos <i>gaps</i> (%)	Tempo Médio (s)
06	0,13	0,00	5,40	0,00	0,18
07	0,11	0,00	15,74	0,00	1,69
08	0,21	0,00	34,64	0,00	16,51
09	0,30	0,00	56,30	0,00	226,25
10	0,47	0,00	149,00	14,99	1809,11
11	0,63	0,00	412,47	41,59	2628,39
12	1,03	1,45	862,03	66,88	3378,70
13	1,53	1,77	1296,35	68,42	3075,20
14	2,23	1,56	2052,93	71,57	3267,77

Pela Tabela 1, observa-se que, por meio do modelo de programação matemática proposto por Gomes Jr. *et al.* (2007), o CPLEX somente conseguiu encontrar a solução ótima de todos os problemas-teste com até 9 tarefas. Para os problemas-teste com 10 tarefas, apesar da média dos *gaps* ser 14,99%, a solução ótima foi encontrada apenas em 62,5% dos problemas. Para os problemas-teste com mais de 10 tarefas, a média dos *gaps* foi sempre superior a 41% e a solução ótima foi encontrada em, no máximo, 31,25% dos problemas. Por outro lado, usando-se a formulação MPMDT proposta, combinada com o algoritmo GPV, o CPLEX conseguiu encontrar todas as soluções ótimas dos problemas-teste com até 11 tarefas. Além disso, o tempo computacional demandado foi menor para a resolução de problemas com 9 ou mais tarefas, mesmo adicionando-se o tempo gasto pelo algoritmo para determinar o horizonte de planejamento. Finalmente, para os casos em que a formulação proposta não foi capaz de gerar a solução ótima no tempo limite estabelecido, a média dos *gaps* foi bem menor; no caso, menor ou igual a 1,77%, contra 71,57% da formulação anterior.

Os resultados dos problemas-teste envolvendo 15 e 16 tarefas não foram apresentados porque o CPLEX aplicado à formulação MPMDT associada ao algoritmo GPV sequer foi capaz de encontrar uma solução viável em 12,5% dos problemas-teste com 15 tarefas, bem como em 31,25% dos problemas-teste com 16 tarefas. Contudo, para os casos em que essa formulação conseguiu gerar uma solução viável, os *gaps* finais médios foram bem baixos (2,66% em problemas com 15 tarefas e 4,39% nos de 16 tarefas), se comparados com aqueles produzidos pela formulação de Gomes Jr. *et al.* (2007), os quais foram sempre superiores a 70%.

7. Conclusões

Este trabalho tratou o problema de sequenciamento em uma máquina com penalidades por

antecipação e atraso da produção (PSUMAA-JE-TP), considerando janelas de entrega distintas e tempo de preparação da máquina dependente da sequência de produção. Foi utilizado um modelo de programação matemática (MPMDT), o qual faz uma discretização do tempo. Para mensurar o horizonte de planejamento H a ser utilizado na formulação MPMDT, foi proposto um algoritmo heurístico, denominado GPV, baseado nos procedimentos GRASP, Princípio da Otimalidade Próxima e Busca em Vizinhança Variável.

O otimizador CPLEX 10.1 foi utilizado para resolver o MPMDT e também o modelo de programação inteira mista proposto por Gomes Jr. *et al.* (2007) e foi aplicado em problemas-teste com até 16 tarefas e com tempos de *setup* não necessariamente simétricos, o que torna a base de dados mais genérica que a utilizada por Gomes Jr. *et al.* (2007). Ambos os modelos permitiram ao CPLEX encontrar a solução ótima em todos os problemas-teste com até 9 tarefas. O MPMDT associado ao algoritmo GPV se mostrou mais eficiente que o modelo de Gomes Jr. *et al.* (2007) na resolução de problemas com até 14 tarefas, visto que, para os problemas-teste com mais de 9 tarefas, o MPMDT com o algoritmo GPV proporcionou ao CPLEX encontrar soluções de melhor qualidade e em menor tempo computacional que o modelo anterior da literatura. Isto permite concluir, também, que o GPV é capaz de encontrar, em pouco tempo computacional, bons limites para o conjunto H , apesar de não garantir que a solução ótima está contida dentro dos limites retornados. Para os problemas com mais de 14 tarefas, os dois modelos não permitiram ao otimizador encontrar bons resultados sempre, sendo que a formulação MPMDT não foi capaz em alguns casos de encontrar uma solução viável para o problema. Apesar disso, para os problemas-teste em que uma solução viável foi encontrada, o procedimento proposto proporcionou *gaps* médios bem baixos (no máximo 1,77%), quando comparados com a formulação de Gomes Jr. *et al.* (2007), que chegou a encontrar *gap* médio de 71,57%.

Agradecimentos

O primeiro autor agradece ao CEFET-MG pela bolsa de pesquisa e o terceiro, ao CNPq (processo 474831/2007-8) e à FAPERJ (processo E-26/101.023/2007), pelo apoio ao desenvolvimento da presente pesquisa.

Referências

- Alidaee, B. e Dragan, I.** (1997) A note on minimizing the weighted sum of tardy and early completion penalties in a single machine: A case of small common due date, *European Journal of Operational Research*, 96, 559-563.
- Allahverdi, A., Gupta, J. N. D. e Aldowaisan, T.** (1999), A review of scheduling research involving setup considerations, *Omega: The International Journal of Management Science*, 27, 219-239.
- Baker, K. R. e Scudder, G. D.** (1990), Sequencing with earliness and tardiness penalties: A review, *Operations Research*, 38, 22–36.
- Bustamante, L. M.**, Minimização do Custo de Antecipação e Atraso para o Problema de Sequenciamento de uma Máquina com Tempo de Preparação Dependente da Sequência: Aplicação em uma Usina Siderúrgica, *Dissertação de mestrado*, Programa de Pós-Graduação em Engenharia de Produção, UFMG, Belo Horizonte, 2006.
- Chang, P. C.** (1999), A Branch and Bound Approach for Single Machine Scheduling with Earliness and Tardiness Penalties, *Computers & Mathematics with Applications*, 37, 133-144.
- Christoforetti, L. M.**, Métodos de Reinício Aplicados ao Sequenciamento em Uma Máquina com Tempos de Preparação e Datas de Entrega, *Dissertação de mestrado*, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Campinas, 2002.
- de Paula, M. R.**, Heurísticas para a Minimização dos Atrasos em Sequenciamento de Máquinas Paralelas com Tempos de Preparação Dependentes da Sequência, *Dissertação de Mestrado*, Programa de Pós-Graduação em Ciências da Computação, UFMG, Belo Horizonte, 2008.
- Feo, T. A. e Resende, M. G. C.** (1995), Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6, 109-133.

França Filho, M. F., GRASP e Busca Tabu aplicados a problemas de programação e tarefas em máquinas paralelas, *Tese de doutorado*, Departamento de Engenharia de Sistemas, UNICAMP, Campinas, 2007.

Glover, F. e Laguna, M. (1997), *Tabu Search*, Kluwer Academic Publishers.

Gomes Jr., A. C., Carvalho, C. R. V., Munhoz, P. L. A. e Souza, M. J. F. (2007), Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção, *In Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional*, SBPO, 649-1660.

Gupta, S. R. e Smith, J. S. (2006), Algorithms for single machine total tardiness scheduling with sequence dependent setups, *European Journal of Operational Research*, 175, 722-739.

Koulamas, C. (1996), Single-machine scheduling with time windows and earliness/tardiness penalties, *European Journal of Operational Research*, 91, 190-202.

Lee, C. Y. e Choi, J. Y. (1995), A Genetic Algorithm for Job Sequencing Problems with Distinct Due Dates and General Early-Tardy Penalty Weights, *Computers & Operations Research*, 22, 857-869.

Li, G. (1997), Single Machine Earliness and Tardiness Scheduling, *European Journal of Operational Research*, 96, 546-558.

Liaw, C.-F. (1999), A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem, *Computers & Operations Research*, 26, 679-693.

Mazzini, R. e Armentano, V. A. (2001), A Heuristic for Single Machine Scheduling with Early and Tardy Costs, *European Journal of Operational Research*, 128, 129-146.

Mladenovic, N. e Hansen, P. (1997), Variable Neighborhood Search, *Computers and Operations Research*, 24, 1097-1100.

Panwalkar, S. S., Dudek, R. A. e Smith, M. L. (1973), Sequencing research and the industrial scheduling problem, *In Elmaghraby SE (Eds), Symposium on the Theory of Scheduling and its Applications*. Springer: Berlin, 29-38.

Rabadi, G., Mollaghasemi, M. e Anagnostopoulos, G. C. (2004), A Branch-and-Bound Algorithm for the Early/Tardy Machine Scheduling Problem with a Common due-date and Sequence-Dependent Setup Time, *Computers & Operations Research*, 31, 1727-1751.

Souza, M. J. F., Penna, P. H. V. e Gonçalves, F. A. C. A. (2008a), GRASP, VND, Busca Tabu e Reconexão por Caminhos para o problema de sequenciamento em uma máquina com tempos de preparação dependentes da sequência da produção, janelas de entrega distintas e penalidades por antecipação e atraso da produção, *In Anais do XL Simpósio Brasileiro de Pesquisa Operacional*, SBPO, 1320-1331.

Souza, M. J. F., Penna, P. H. V., Gonçalves, F. A. C. A. e Ochi, L. S. (2008b), Uma heurística híbrida para minimizar custos com antecipação e atraso em sistemas de produção com janelas de entrega e tempos de preparação dependentes da sequência, *In Anais do XI Simpósio de Pesquisa Operacional e Logística da Marinha, SPOLM, CD-ROM*, 16 p.

Valente, J. M.S. e Alves, R. A. F. S. (2005), Improved heuristics for the early/tardy scheduling problem with no idle time, *Computers & Operations Research*, 32, 557-569.

Wan, G. e Yen, B. P. C. (2002), Tabu Search for Single Machine Scheduling with Distinct Due Windows and Weighted Earliness/Tardiness Penalties, *European Journal of Operational Research*, 142, 271-281.

Ying, K. C. (2008), Minimizing earliness-tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm, *Computers & Industrial Engineering*, 55, 494-502.

Formulações de Programação Matemática para o Problema de Seqüenciamento em uma Máquina com Janelas de Entrega Distintas e Tempo de Preparação Dependente da Seqüência de Produção

Bruno F. Rosa, **Sérgio R. de Souza,**

Programa de Pós-graduação em Modelagem Matemática e Computacional, CEFET-MG,
30510-000, Belo Horizonte, MG

E-mail: brunofazmat@dppg.cefetmg.br, sergio@dppg.cefetmg.br

Marccone J. F. Souza

UFOP - Departamento de Computação

Campus Universitário

35.400-000, Ouro Preto, MG

E-mail: marcone@iceb.ufop.br.

Resumo: *Este trabalho trata do problema de seqüenciamento em uma máquina em que o tempo de preparação depende da seqüência de produção. No problema abordado, cada tarefa possui uma janela de tempo, dentro da qual deve ser preferencialmente concluída. O objetivo a ser satisfeito é minimizar a soma ponderada dos atrasos e antecipações na produção de tais tarefas. Duas novas formulações de programação matemática são propostas para representar o problema, sendo a primeira delas um aperfeiçoamento de uma formulação da literatura, e, a outra, uma formulação indexada no tempo. Experimentos computacionais mostram que a formulação indexada no tempo possibilita encontrar soluções de melhor qualidade em menor tempo que as outras duas formulações.*

Palavras-chave: *Seqüenciamento em Uma Máquina, Penalidades por Antecipação e Atraso, Modelagem Matemática Indexada no Tempo.*

1 Introdução

O problema de seqüenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA) consiste em seqüenciar e determinar o momento em que as tarefas devem ser executadas em uma máquina, com o objetivo de minimizar a soma ponderada das antecipações e dos atrasos na produção de tais tarefas. Neste problema, cada tarefa está associada a um tempo de processamento e a uma data desejada para sua entrega. Além disso, após a conclusão de uma tarefa, é necessário preparar a máquina para a execução da tarefa seguinte.

Segundo [1], o tempo de preparação da máquina, também conhecido como tempo de *setup*, aparece em um grande número de aplicações do PSUMAA. Este tempo de preparação ocorre principalmente em empresas que produzem diversos tipos de produtos, sendo, assim, necessária a troca do tipo de tarefa executada na máquina. Nele são incluídos os tempos gastos para trocar as ferramentas, preparar os materiais, limpar a máquina, dentre outras questões. A maioria dos trabalhos em problemas de seqüenciamento assume que os tempos de *setup* são independentes da seqüência de produção, ou seja, que esses tempos são ou desprezíveis ou podem ser acrescentados aos tempos de processamento das tarefas. Apesar disso, em grande parte das situações práticas, os tempos de *setup* são dependentes da seqüência de produção [6].

As datas de entrega podem ser comuns (*common due date*) ou distintas (*distinct due dates*). Uma abordagem genérica do PSUMAA considera a existência de janelas de entrega distintas,

em que há um determinado período de tempo associado a cada tarefa, dentro do qual a tarefa deve ser preferencialmente concluída. De acordo com [9], este último caso ocorre quando existem tolerâncias em torno das datas desejadas para a entrega de cada tarefa. Deste modo, as tarefas concluídas dentro de suas respectivas janelas de entrega não ocasionam penalidades, ao contrário daquelas concluídas fora de suas janelas de entrega.

Existem trabalhos sobre o PSUMAA nos quais a ociosidade da máquina não é permitida. Conforme [7], há situações em que manter a máquina parada é mais custoso que a antecipação de uma tarefa, ou outras, em que a demanda é superior à capacidade da máquina. Por outro lado, [4] afirma que há casos em que manter a máquina inativa é vantajoso, ainda que exista uma tarefa disponível para processamento.

Na literatura, são encontrados muitos trabalhos com o objetivo de resolver o PSUMAA. Isto se deve ao fato de ser um problema com muitas aplicações industriais, de um lado, e, de outro, difícil de ser resolvido na otimalidade, dado que se trata de um problema NP-difícil [1].

Neste trabalho, é estudado o PSUMAA com janelas de entrega distintas, tempo de preparação da máquina dependente da seqüência de produção, sendo permitidos tempos ociosos entre as execuções de tarefas consecutivas. Apesar de ser uma generalização do PSUMAA, esta versão tem recebido pouca atenção. Dentre os trabalhos que tratam desta versão generalizada do problema, a maioria tem seu foco em procedimentos heurísticos para resolvê-lo. Apenas no trabalho de [2] e no trabalho de [5] são propostos métodos exatos para resolver tal problema.

Em [2] é estudado o caso real de uma indústria siderúrgica, em que uma máquina é considerada como sendo uma seqüência de laminadores e cada tarefa representa a produção de um determinado produto (barra chata, cantoneira, vergalhão, etc.). São desenvolvidos dois modelos de programação linear inteira mista (PLIM), os quais, no entanto, exigem que os tempos de *setup* satisfaçam à desigualdade triangular. Apesar de ser considerado o caso com janelas de entrega, os testes computacionais foram realizados em problemas-teste com datas de entrega distintas. Foram resolvidos na otimalidade problemas com até 10 tarefas por meio do *software* de otimização GLPK, versão 4.8.

Já em [5] é proposto um modelo de PLIM baseado em uma das formulações de programação matemática apresentadas por [2]. Porém, este modelo não exige que os tempos de *setup* satisfaçam à desigualdade triangular. Por meio do otimizador CPLEX, versão 9.1, foram encontrados os ótimos em problemas-teste com até 12 tarefas e com tempos de *setup* simétricos. Os autores propõem também um algoritmo heurístico, baseado em GRASP, *Iterated Local Search* e *Variable Neighborhood Descent*. Para cada seqüência de tarefas gerada por esse algoritmo, é acionado um procedimento de complexidade polinomial para determinar a data ótima de início de processamento das tarefas na seqüência dada.

No presente trabalho, é proposta uma formulação matemática indexada no tempo para representar o problema em foco. É apresentado também um novo modelo de PLIM, baseado naquele proposto por [5], com menor número de variáveis e restrições. Estes dois modelos, bem como o modelo proposto por [5], são utilizados para resolver problemas-teste por meio do otimizador CPLEX, versão 10.1, sendo realizada uma comparação entre os resultados encontrados.

Este trabalho está organizado do seguinte modo. Na Seção 2 é feita uma descrição detalhada do problema estudado. O modelo de PLIM proposto por [5], um modelo de PLIM baseado neste último e uma formulação de programação matemática indexada no tempo são apresentados nas Seções 3, 4 e 5, respectivamente. Na Seção 6 são apresentados e discutidos os resultados encontrados, enquanto a Seção 7 conclui o trabalho.

2 Descrição do problema

O PSUMAA abordado neste trabalho possui as seguintes características: (i) Uma máquina deve processar um conjunto I de n tarefas; (ii) A cada tarefa $i \in I$ está associado um tempo de processamento P_i ; uma janela de entrega $[E_i, T_i]$, na qual a tarefa i deve ser preferencialmente concluída; um custo α_i por unidade de tempo de antecipação; e um custo β_i por unidade de tempo de atraso. Há antecipação de uma tarefa $i \in I$ quando seu processamento é concluído

antes de E_i e há atraso quando seu processamento é concluído depois de T_i ; (iii) As tarefas que forem concluídas dentro de suas respectivas janelas de entrega não geram custo; (iv) A máquina executa no máximo uma tarefa por vez e, uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção; (v) Todas as tarefas estão disponíveis para processamento na data 0; (vi) Entre duas tarefas i e j consecutivas é necessário um tempo S_{ij} de preparação da máquina, chamado tempo de *setup*; (vii) Assume-se que o tempo de preparação da máquina para o processamento da primeira tarefa na seqüência é igual a 0; (viii) É permitido tempo ocioso entre a execução de duas tarefas consecutivas.

O objetivo é determinar a seqüência de produção e as datas de início de produção das tarefas, de forma a minimizar a soma ponderada das antecipações e dos atrasos.

3 Modelo MPLIM-G

Nesta seção, apresenta-se o modelo de programação linear inteira mista (PLIM) proposto por [5], doravante denotado por MPLIM-G. Para auxiliar na modelagem, são utilizadas duas tarefas fictícias, 0 (zero) e $n+1$, que devem ser seqüenciadas necessariamente na primeira e na última posição, respectivamente. Admite-se que P_0 e P_{n+1} são iguais a zero e que $S_{0i} = S_{i0} = 0$ e $S_{i,n+1} = S_{n+1,i} = 0, \forall i \in I$. A data de início do processamento da tarefa $j \in I \cup \{0, n+1\}$ e os tempos de antecipação e atraso da tarefa $i \in I$ são representados por s_i , e_i e t_i , respectivamente.

Sejam y_{ij} variáveis que determinam a seqüência de produção, sendo $y_{ij} = 1$ se a tarefa j for seqüenciada imediatamente após a tarefa i e $y_{ij} = 0$, caso contrário, $\forall i, j \in I \cup \{0, n+1\}$. Considere, ainda, uma constante M de valor suficientemente grande. O modelo MPLIM-G é representado pelas equações (1)-(10):

$$\min \quad Z = \sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (1)$$

$$\text{s.a. } s_i + P_i + y_{ij}(M + S_{ij}) - M \leq s_j \quad \forall i \in I \cup \{0\}, \forall j \in I \cup \{n+1\} \text{ e } i \neq j \quad (2)$$

$$\sum_{j=1, j \neq i}^{n+1} y_{ij} = 1 \quad \forall i \in I \cup \{0\} \quad (3)$$

$$\sum_{i=0, i \neq j}^n y_{ij} = 1 \quad \forall j \in I \cup \{n+1\} \quad (4)$$

$$s_i + P_i + e_i \geq E_i \quad \forall i \in I \quad (5)$$

$$s_i + P_i - t_i \leq T_i \quad \forall i \in I \quad (6)$$

$$s_i \geq 0 \quad \forall i \in I \cup \{0, n+1\} \quad (7)$$

$$e_i \geq 0 \quad \forall i \in I \quad (8)$$

$$t_i \geq 0 \quad \forall i \in I \quad (9)$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j \in I \cup \{0, n+1\} \text{ e } i \neq j \quad (10)$$

A função objetivo, representada pela equação (1), busca a minimização da soma ponderada das antecipações e atrasos. As restrições (2) garantem que existe tempo suficiente para executar uma tarefa i e preparar a máquina antes do início do processamento da tarefa seguinte j . As restrições (3) e (4) garantem que cada tarefa terá apenas uma tarefa imediatamente sucessora e uma tarefa imediatamente antecessora, respectivamente. As restrições (5) e (6) definem as antecipações e os atrasos de acordo com as respectivas janelas de entrega de cada tarefa. As restrições (7), (8), (9) e (10) dizem respeito ao tipo de variáveis.

4 Modelo MPLIM-BG

O modelo de programação linear inteira mista (PLIM) apresentado a seguir é baseado no proposto por [5] (ver Seção 3) e será denotado por MPLIM-BG. Diferentemente de tais autores, é utilizada apenas uma tarefa fictícia, denominada 0 (zero), para auxiliar na modelagem. Esta tarefa é seqüenciada duas vezes, sendo uma na primeira posição e outra na última. Considera-se que $P_0 = 0$ e que $S_{0i} = S_{i0} = 0, \forall i \in I$.

Considerando as definições de s_i , e_i , t_i e y_{ij} tal como no MPLIM-G, tem-se que as variáveis $y_{n+1,i}$ e $y_{i,n+1}$, $\forall i \in I \cup \{0\}$, e s_{n+1} deixam de ser necessárias no MPLIM-BG. Sendo assim, se comparado ao MPLIM-G, no MPLIM-BG há redução de $2n + 3$ variáveis e $n + 1$ restrições, com n representando o número de tarefas a serem sequenciadas.

Finalmente, o MPLIM-BG é obtido do MPLIM-G substituindo-se as restrições (2), (3), (4), (7) e (10) pelas restrições (11), (12), (13), (14) e (15), respectivamente, apresentadas a seguir:

$$s_i + P_i + y_{ij}(M + S_{ij}) - M \leq s_j \quad \forall i \in I \cup \{0\}, \quad \forall j \in I \text{ e } i \neq j \quad (11)$$

$$\sum_{j=0, j \neq i}^n y_{ij} = 1 \quad \forall i \in I \cup \{0\} \quad (12)$$

$$\sum_{i=0, i \neq j}^n y_{ij} = 1 \quad \forall j \in I \cup \{0\} \quad (13)$$

$$s_i \geq 0 \quad \forall i \in I \cup \{0\} \quad (14)$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j \in I \cup \{0\} \text{ e } i \neq j \quad (15)$$

5 Modelo MPLIM-IT

Propõe-se, a seguir, um modelo de programação linear inteira mista (PLIM) indexado no tempo para representar o PSUMAA abordado. Tal modelo, doravante denotado por MPLIM-IT, é baseado no trabalho de [3], que utilizou a discretização do tempo para modelar o problema de minimização dos atrasos em seqüenciamento de máquinas paralelas com tempos de preparação dependentes da seqüência. O MPLIM-IT é descrito a seguir.

Seja $H = \{h_0, h_1, h_2, \dots, h_L\}$ o conjunto com as possíveis datas de início de processamento das tarefas e considere x_{ih} variáveis binárias, sendo $x_{ih} = 1$ se a tarefa i é programada para iniciar na data h e $x_{ih} = 0$, caso contrário, $\forall i \in I$ e $\forall h \in H$.

Se e_i e t_i representam, respectivamente, as unidades de tempo de antecipação e tempo de atraso da tarefa $i \in I$, a formulação MPLIM-IT pode ser escrita como:

$$\min \quad Z = \sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (16)$$

$$\text{s.a. } x_{ih} + \sum_{u=h}^{\min(h+P_i+S_{ij}-1, H_L)} x_{ju} \leq 1 \quad \forall i, j \in I, \quad \forall h \in H \text{ e } i \neq j \quad (17)$$

$$\sum_{h=H_0}^{H_L} x_{ih} = 1 \quad \forall i \in I \quad (18)$$

$$\sum_{h=H_0}^{H_L} x_{ih} h + P_i + e_i \geq E_i \quad \forall i \in I \quad (19)$$

$$\sum_{h=H_0}^{H_L} x_{ih} h + P_i - t_i \leq T_i \quad \forall i \in I \quad (20)$$

$$e_i \geq 0 \quad \forall i \in I \quad (21)$$

$$t_i \geq 0 \quad \forall i \in I \quad (22)$$

$$x_{ih} \in \{0, 1\} \quad \forall i \in I \text{ e } \forall h \in H \quad (23)$$

A função objetivo representada pela equação (16) tem como critério de otimização a minimização da soma ponderada das antecipações e dos atrasos. As restrições (17) garantem que existe tempo suficiente para executar uma tarefa i e preparar a máquina antes do início do processamento da tarefa seguinte j . As restrições (18) garantem que cada tarefa seja executada uma única vez. As restrições (19) e (20) definem as antecipações e os atrasos de acordo com as respectivas janelas de entrega de cada tarefa. As restrições (21), (22) e (23) determinam os domínios das variáveis do problema.

É importante observar que, diferentemente do MPLIM-G e do MPLIM-BG, o MPLIM-IT somente é válido se a desigualdade triangular for satisfeita pelos dados do problema, ou seja, se as condições (24) forem satisfeitas:

$$S_{ik} \leq S_{ij} + S_{jk} + P_j \quad \forall i, j, k \in I, i \neq j, i \neq k \text{ e } j \neq k \quad (24)$$

No PSUMAA abordado neste trabalho, a execução de qualquer tarefa pode ser iniciada em qualquer momento futuro e, portanto, caso tenha-se a garantia de que na seqüência ótima de um determinado problema o início do processamento da primeira tarefa não ocorre antes de h_{inf} e que o início do processamento da última tarefa é no máximo h_{sup} , é conveniente tomar $H = \{h_{inf}, h_{inf} + 1, h_{inf} + 2, \dots, h_{sup}\}$. Assim, como o MPLIM-IT é fortemente sensível à cardinalidade de H , quanto menor o intervalo $[h_{inf}, h_{sup}]$, menor o número de variáveis terá o modelo.

6 Experimentos computacionais

Os modelos matemáticos apresentados nas seções 3, 4 e 5 foram implementados usando o modelador AMPL e resolvidos pelo otimizador CPLEX, versão 10.1, da ILOG. Os testes foram realizados em um computador AMD Turion(tm) 64 X2 TL-58 1900 MHz, com 2 GB de RAM, sob plataforma Windows XP.

Foram gerados problemas-teste baseados nos trabalhos de [8] e [9], conforme a seguir se descreve. Dada uma tarefa i , o tempo de processamento (P_i), o custo por unidade de atraso (β_i) e o custo por unidade de antecipação (α_i) são números inteiros selecionados aleatoriamente dentro dos intervalos $[1, 40]$, $[1, 10]$ e $[1, \beta_i]$, respectivamente. O centro da janela de entrega de i é um valor inteiro aleatório dentro do intervalo $[(1 - FA - VRJ/2) \times TTP, (1 - FA + VRJ/2) \times TTP]$, em que TTP é o tempo total de processamento de todas as tarefas, FA é o fator de atraso e VRJ é a variação relativa da janela de entrega. O tamanho da janela de entrega é um valor inteiro selecionado aleatoriamente no intervalo $[0, TTP/n]$, sendo n o número de tarefas. Para toda tarefa $j \neq i$, o tempo de *setup* (S_{ij}) é um número inteiro aleatório pertencente ao intervalo $[5, 15]$. Assim, diferentemente de [5], os tempos de *setup* não são necessariamente simétricos.

Foram gerados conjuntos de problemas-teste com 6, 7, 8, 9, 10, 11, 12, 13 e 14 tarefas, respectivamente, sendo utilizados os valores 0,1; 0,3; 0,5 e 0,8 para FA e 0,4; 0,7; 1,0 e 1,3 para VRJ . Sendo assim, há 16 problemas-teste em cada conjunto. Durante a geração dos problemas-teste, sempre que um problema gerado não satisfazia a desigualdade triangular, ele era descartado e outro problema com os mesmos valores para FA e VRJ era gerado. Desta forma, todos os problemas da base de dados satisfazem à desigualdade triangular.

Para os modelos MPLIM-G e MPLIM-BG, foi utilizado $M = 1000$. Para cada problema-teste, o conjunto H do MPLIM-IT foi determinado pelas expressões (25) e (26):

$$h_{inf} = \max \left(0, \min \left\{ T_i \mid i \in I \right\} + \min \left\{ S_{ij} \mid i, j \in I \text{ e } i \neq j \right\} - \sum_{i \in I} P_i - \sum_{i \in I} \max \left\{ S_{ij} \mid j \in I \text{ e } i \neq j \right\} \right) \quad (25)$$

$$h_{sup} = \max \left\{ E_i \mid i \in I \right\} + \sum_{i \in I} P_i + \sum_{i \in I} \max \left\{ S_{ij} \mid j \in I \text{ e } i \neq j \right\} - \min \left\{ S_{ij} \mid i, j \in I \text{ e } i \neq j \right\} - \min \left\{ P_i \mid i \in I \right\} - \min \left(\min \left\{ P_i \mid i \in I \right\}, \max \left\{ E_i \mid i \in I \right\} \right) \quad (26)$$

em que h_{inf} garante que todas as tarefas podem ser concluídas sem atraso (ou o processamento da primeira tarefa é iniciado na data 0) e h_{sup} assegura que todas as tarefas podem ser executadas sem antecipação. Portanto, tem-se a garantia de que as datas de início de processamento das tarefas na seqüência ótima pertencem ao conjunto $H = \{h_{inf}, h_{inf} + 1, h_{inf} + 2, \dots, h_{sup}\}$.

Considerou-se, ainda, o limite de 3600 segundos para resolução de cada problema pelo CPLEX. Para os problemas em que este limite de tempo é atingido, a solução retornada não é

necessariamente ótima; porém, é retornado, pelo CPLEX, um limite inferior para o ótimo do mesmo. Este limite é utilizado para medir a qualidade da solução retornada, dada pela equação:

$$gap = \frac{(f^{CPLEX} - L)}{L} \times 100\% \quad (27)$$

em que f^{CPLEX} e L representam o valor da solução e o limite inferior encontrados, respectivamente. Se uma solução ótima é encontrada, tem-se $gap = 0$.

Os resultados encontrados para o MPLIM-G, para o MPLIM-BG e para o MPLIM-IT são resumidos na Tabela 1, na qual a primeira coluna indica o número de tarefas existentes em cada conjunto de problemas-teste. Para cada conjunto de problemas, as colunas “Ótimos Encontrados” mostram (em porcentagem) em quantos problemas o CPLEX encontrou uma solução ótima para os respectivos modelos, as colunas “ $\overline{gap}$ ” mostram as médias dos $gaps$ para os respectivos modelos e as colunas “Tempo” mostram as médias dos tempos (em segundos) demandados por tal otimizador na resolução dos problemas-teste com os respectivos modelos.

Tabela 1: Comparação MPLIM-G $\times$ MPLIM-BG $\times$ MPLIM-IT.

# Tarefas	MPLIM-G			MPLIM-BG			MPLIM-IT		
	Ótimos Encontrados (%)	$\overline{gap}$ (%)	Tempo (s)	Ótimos Encontrados (%)	$\overline{gap}$ (%)	Tempo (s)	Ótimos Encontrados (%)	$\overline{gap}$ (%)	Tempo (s)
06	100,00	0,00	0,18	100,00	0,00	0,16	100,00	0,00	8,77
07	100,00	0,00	1,69	100,00	0,00	1,78	100,00	0,00	20,96
08	100,00	0,00	16,51	100,00	0,00	19,00	100,00	0,00	34,67
09	100,00	0,00	226,25	100,00	0,00	249,67	100,00	0,00	79,24
10	62,50	14,99	1809,11	68,75	14,84	1796,76	100,00	0,00	196,17
11	31,25	41,59	2628,39	31,25	40,61	2658,07	100,00	0,00	453,88
12	6,25	66,88	3378,70	6,25	67,03	3380,08	93,75	0,38	845,25

Pela Tabela 1, observa-se que o MPLIM-G e o MPLIM-BG apresentaram comportamentos bem semelhantes. Usando-se esses dois modelos, o CPLEX foi capaz de resolver na otimalidade somente problemas-teste com até 9 tarefas. Por outro lado, com o modelo MPLIM-IT, o CPLEX conseguiu resolver na otimalidade todos os problemas-teste de até 11 tarefas. Além disso, enquanto as duas primeiras formulações permitiam ao CPLEX resolver na otimalidade apenas 6,25% dos problemas-teste envolvendo 12 tarefas, utilizando-se a formulação MPLIM-IT foi possível solucionar na otimalidade 93,75% do mesmo conjunto de problemas. Esta última formulação também exigiu, na maioria dos casos, um menor tempo computacional para se alcançar a otimalidade. De fato, à exceção dos três primeiros conjuntos de problemas-teste, a formulação MPLIM-IT demandou aproximadamente de 3 a 9 vezes menos tempo que as demais formulações. Ainda na Tabela 1, nota-se que a formulação MPLIM-IT permitiu determinar, para os problemas-teste com 12 tarefas, dentro do limite de tempo adotado, soluções tendo $\overline{gap}$ de 0,38%, em contraste ao valor obtido pelas duas outras formulações avaliadas (MPLIM-G e MPLIM-BG), que obtiveram $\overline{gap}$ de 66,88% e 67,03%, respectivamente.

Apesar de não ser mostrado na Tabela 1, o CPLEX com o MPLIM-IT não conseguiu sequer encontrar uma solução viável, dentro do limite de tempo adotado, em 6,25% dos problemas-teste com 13 tarefas e em 56,25% dos problemas-teste com 14 tarefas. Com os outros dois modelos, o $\overline{gap}$ retornado para estes problemas foi superior a 68%.

7 Conclusões

Este trabalho tratou o problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA), considerando janelas de entrega distintas e tempo de preparação da máquina dependente da seqüência. Foi apresentada uma nova formulação de programação matemática para o problema (MPLIM-BG), a qual exige um número menor de

variáveis e restrições em relação àquela proposta por [5] (MPLIM-G). Propôs-se, também, uma formulação matemática indexada no tempo (MPLIM-IT) para representar o PSUMAA.

O otimizador CPLEX 10.1 foi utilizado para resolver as três formulações matemáticas em tela, aplicadas em problemas-teste com até 14 tarefas. Os resultados computacionais mostram que as formulações MPLIM-G e MPLIM-BG proporcionam resultados semelhantes e que a formulação MPLIM-IT é mais eficiente que estas duas na resolução de problemas com até 12 tarefas, visto que esta última proporcionou ao CPLEX encontrar soluções com garantia de melhor qualidade e com menor tempo médio de processamento que as demais. Para os problemas-teste com mais de 12 tarefas, devido ao limite de tempo estabelecido para o CPLEX retornar uma solução, os resultados obtidos não permitem uma comparação efetiva entre os três modelos. Como trabalhos futuros, sugere-se o estudo de propriedades do PSUMAA que permitam reduzir a cardinalidade do conjunto H da formulação MPLIM-IT, dado que a mesma é fortemente dependente de tal conjunto.

Agradecimentos

O primeiro autor agradece ao CEFET-MG pela bolsa de pesquisa e o terceiro, ao CNPq (processo 474831/2007-8) e à FAPERJ (processo E-26/101.023/2007), pelo apoio ao desenvolvimento da presente pesquisa.

Referências

- [1] A. Allahverdi, J. N.D. Gupta e T. Aldowaisan, A review of scheduling research involving setup considerations, *Omega: International Journal of Management Science*, 27 (1999) 219-239.
- [2] L. M. Bustamante, “Minimização do custo de antecipação e atraso para o problema de sequenciamento de uma máquina com tempo de preparação dependente da sequência: aplicação em uma usina siderúrgica”, Dissertação de Mestrado, DEP-UFMG, 2007.
- [3] M. R. de Paula, “Heurísticas para a minimização dos atrasos em sequenciamento de máquinas paralelas com tempos de preparação dependentes da sequência”, Dissertação de Mestrado, DCC-UFMG, 2008.
- [4] M. F. França Filho, “GRASP e Busca Tabu aplicados a problemas de programação de tarefas em máquinas paralelas”, Tese de Doutorado, FEEC-Unicamp, 2007.
- [5] A. C. Gomes Júnior, C. R. V. Carvalho, P. L. A. Munhoz e M. J. F. Souza, Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção, em “Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional”, pp. 1649-1660, SOBRAPO, Fortaleza, 2007.
- [6] S. R. Gupta e J. S. Smith, Algorithms for single machine total tardiness scheduling with sequence dependent setups, *European Journal of Operational Research*, 175 (2006) 722-739.
- [7] G. Li, Single machine earliness and tardiness scheduling, *European Journal of Operational Research*, 96 (1997) 546-558.
- [8] G. Rabadi, M. Mollaghasemi e G. C. Anagnostopoulos, A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time, *Computers & Operations Research*, 31 (2004) 1727-1751.
- [9] G. Wan e B. P.-C. Yen, Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties, *European Journal of Operational Research*, 142 (2002) 271-281.

A hybrid heuristic algorithm based on GRASP, VND, ILS and Path Relinking for the open-pit-mining operational planning problem

Igor Machado Coelho

Sabir Ribas

imcoelho@ic.uff.br

sribas@ic.uff.br

Fluminense Federal University, Computer Science Institute, Niterói, 24210-240, RJ, Brazil

Marcone Jamilson Freitas Souza

Vitor Nazário Coelho

marcone@iceb.ufop.br

vncoelho@gmail.com

Federal University of Ouro Preto, Department of Computer Science, Ouro Preto, 35400-000, MG, Brazil

Luiz Satoru Ochi

satoru@ic.uff.br

Fluminense Federal University, Computer Science Institute, Niterói, 24210-240, RJ, Brazil

Abstract. *This work deals with dynamic truck allocation in open-pit-mining operational planning. The objective is to optimize mineral extraction in the mines by minimizing the number of mining trucks used to meet production goals and quality requirements. According to literature, this problem is NP-hard, therefore heuristic strategies are justified. We present a hybrid algorithm that combines the power of three metaheuristics, Greedy Randomized Adaptive Search Procedure, Variable Neighborhood Descent and Iterated Local Search, as well as a mathematical programming module that is triggered periodically to solve smaller parts of the problem in optimality, and a post optimization module based on Path Relinking approach. The proposed algorithm was tested in a set of instances based on real-life problems and the results were compared with other metaheuristics-based strategies available in the literature and with an optimization solver. The computational experiments show that the developed algorithm is competitive, since it can obtain good quality solutions in an acceptable execution time.*

Keywords: *Open-pit-mining, Metaheuristics, Iterated Local Search, GRASP, Path Relinking*

1. INTRODUCTION

This work deals with dynamic truck allocation in Open-Pit-Mining Operational Planning (OPMOP). The situation involves mineral extraction (ore and waste rocks), as well as the equipment (shovels and mining trucks) used in the operation. Also taken into consideration are several ore control parameters (e.g., % iron, % silica, and water). The objective is to determine the extraction rate at each pit in a way that production and quality goals are satisfied, and to minimize the number of trucks needed for the process.

The problem considers that only mining trucks are used to transport the material from the loading points (pits) to the unloading points (either the crusher or the waste rock deposit). It is considered, too, that there are shovels of different productivities and their set is smaller than the number of pits they can be allocated to. Given the high cost of a shovel, a minimum quantity of extrac mineral is required to justify its use.

In the dynamic allocation system, the trucks are not fixed to a specific shovel/or pit. They can be directed to different pits, which increases the fleet productivity, reducing the amount of equipment needed to maintain a certain level of production. In this system it is also possible to decrease the time of the queue, since the truck can be allocated to different loading points. The disadvantages of dynamic vehicle allocation are: the demand for a greater number of operations; and a computerized dispatching system for the mining trucks.

The problem in focus has the Multiple Knapsack Problem (MKP) as a subproblem. In fact, the analogy can be made by considering each shovel like a knapsack and the loads (ore or waste rock) of the trucks as the items. In this analogy, the goal is to determine which loads are the most attractive to allocate to each knapsack, respecting its capacity (productivity). Thus, as MKP belongs to the NP-hard class (Papadimitriou and Steiglitz, 1998), OPMOP does too. Since in real cases the decision must be fast and it is unlikely that optimum solutions would be obtained by exact techniques in a short space of time, it is proposed to find sub-optimal solutions for the problem by means of heuristic techniques. The proposed heuristic algorithm is based on the procedures Greedy Randomized Adaptive Search Procedures - GRASP (Feo and Resende, 1995), Variable Neighborhood Descent - VND (Mladenovic and Hansen, 1997) and Iterated Local Search - ILS (Lourenço et al., 2003). To test its efficiency, the results from this algorithm were compared to those achieved by a market optimization software applied to a mathematical programming model also proposed in this work.

The contribution of this work is the presentation of a more complete mathematical programming model of OPMOP than those found in literature. This model seeks to more faithfully depict a real operational mining industry environment. Moreover, it presents a new heuristic model not yet found in literature in order to solve the problem in focus.

This paper is organized as follows. Section 2 shows the related work. Section 3 presents a mathematical programming formulation to OPMOP, while Section 4 presents a heuristic approach to the problem in focus. The testing scenarios are described in Section 5, while in the following section, the computational experiments are presented and analyzed. Section 7 concludes the work.

2. RELATED WORKS

White and Olson (1986) proposed an algorithm that is the basis for the *DISPATCH* System, which operates in many mines around the world. A solution is obtained in two steps. The first, based on linear programming, handles the problem of ore mixture optimizing by minimizing costs considering the mining rate, the quality of the mixture, the ore feed rate to the plant for beneficiation, and the material handling. The restrictions of the model are related to the pro-

duction capacity of the shovels, the quality of the mixture and the minimum feeding rate to the processing plant. The second stage of the algorithm, which is solved by dynamic programming, uses a model similar to White et al. (1982), differing from this by using a decision variable for the volume of material transported per hour on a given route, instead of the truck working rate per hour. Also considered is the presence of storage piles. In this second stage of the algorithm, the objective is to minimize material transportation in the mine.

Sgurev et al. (1989) describe an automated system for real-time control of truck haulage in open-pit mines. This system is called TRASY and it is designed towards the improvement of the technical-economical indices of the loading-unloading process in open-pit mines where trucks are used as vehicles. The authors describe the two ways of organizing the trucks work: on a closed-circuit system and on an open-circuit system, so called dynamic allocation system in the present paper. The benefits of the open-circuit system are shown and the authors describe the four modules of the TRASY system: configuration, control, monitoring and report. The authors conclude that the increase of the operation productivity in open-pit mines may be achieved by improving the effectiveness of the loading-haulage process control, so the introduction of automated systems for haulage vehicles control is one way to accomplish this goal. However, this system does not take into account the quality goals of the ore control parameters.

Chanda and Dagdelen (1995) developed a linear programming model that solves the problem of mixed minerals in the short-term planning of a coal mine. The objective function of this model is the weighted sum of three distinct objectives: to maximize an economic criterion, to minimize production deviations, and to minimize quality deviations from the desired values of the control parameters. No allocation for the loading and transport equipment was considered in this model.

Ezawa and Silva (1995) developed a system for dynamic truck allocation with the objective of reducing variability in the levels of the ore and increasing transport productivity. The system uses a heuristic to sequence the trucks in order to minimize changes in the levels. To validate it, the authors used a simulation and the theory of graphs for the mathematical modeling of the mine. Deploying this system transport productivity increased by 8% and management obtained more accurate data in real time.

Alvarenga (1997) developed a program for the optimal dispatch of trucks in the iron mining of an open pit mine, with the objectives of minimizing the queue time of the trucks in the fleet, increasing productivity and improving the quality of the extracted ore. In the work, which is the basis of the SMART MINE system widely used in various Brazilian mines, a technique of stochastic optimization was applied, using the genetic algorithm with parallel processing. Basically, the problem is to indicate the best point of tipping or loading and the trajectory for the movement, when there is a situation of choice to be made. The author pointed to productivity gains of 5% to 15%, proving the validity of the proposal.

Merschmann (2002) developed an optimization system and simulation for analyzing the production scenario in open pit mines. The system, called OTISIMIN (Simulator and Optimizer for Mining), was developed in two modules. The first is the optimization module where a linear programming model is constructed and solved, while the second is a simulation module that allows the user to use the results obtained by solving the linear programming model as input for the simulation. The optimization module was developed with the aim of optimizing the process of mixing the ores from the mining of several pits in order to meet the quality specifications imposed by the treatment plant and allocating the equipment (trucks, shovels and / or excavators) to the pits, considering both static and dynamic truck allocation. The developed model does not consider production optimization and quality targets, or reduction of the number of trucks required by the production system.

Godoy and Dimitrakopoulos (2004) deals with the open pit mine design and production scheduling problem, with a view to find the most profitable mining sequence over the life of a mine. According to the authors the dynamics of mining ore and waste and the spatial grade uncertainty make predictions of the optimal mining sequence a challenging task. The authors show a risk-based approach to life-of-mine production scheduling, including the determination of optimum mining rates for the life of mine, whilst considering ore production, stripping ratios, investment in equipment purchase and operational costs; and the generation of a detailed mining sequence from the previously determined mining rates, focusing on spatial evolution of mining sequences and equipment utilization. The production scheduling stage uses a specially-developed combinatorial optimization algorithm based on the Simulated Annealing metaheuristic. A new risk-based, multistage optimization process for long-term production scheduling is presented, and the results show the potential to considerably improve the valuation and forecasts for life-of-mine schedule.

Guimaraes et al. (2007) presents a computational simulation model to validate the results obtained by applying a mathematical programming model to determine the mining rate in open pit mines. LINGO solver, version 7.0, is used for optimizing the problem and ARENA, version 7.0, simulate the solver's solution. Contrary to belief, the modeling demonstrate that by increasing the number of vehicles, the production goal is not met and is further deterred due to increased queue time. Thus, increasing the number of vehicles does not necessarily optimize mining operations.

Boland et al. (2009) deals with the open pit mining production scheduling problem. The treated problem consists of finding the sequence in which the blocks should be removed from the pit, over the lifetime of the mine, such that the net present value (NPV) of the operation is maximized. Due to the large number of blocks and precedence constraints linking them, blocks are aggregated to form larger scheduling units. The authors investigated the characteristics of the problem and showed how the aggregates can be systematically divided into bins (groups of blocks) so that the solution of the linear programming (LP) relaxation with all processing decision variables fully disaggregated to block level (D-LP) can be recovered from the solution of our compactly disaggregated LP relaxation (B-LP) with processing decisions made at the level of bins. As the number of bins is much smaller than the number of blocks, using their binning approach, D-LP can be solved to optimality for much larger data instances than by a direct disaggregation approach. They showed that their approach can lead to significant improvements in NPV.

Coelho et al. (2008) presents a metaheuristics based algorithm to solve the OPMOP. It is also presented a more complete formulation of the mathematical programming model to solve the problem. The proposed model is an extension of the model of Costa (2005). Computational experiments compare the solutions of the metaheuristics approach and the mathematical programming model using the CPLEX solver. Results show that the metaheuristics approach can find good solutions with low variability quickly, what did not occur with CPLEX in the computational experiments. Since the decision maker needs fast answers the proposed algorithm is validated to real applications.

Ribas et al. (2009) proposes a parallel version of the Iterated Local Search metaheuristic to solve the OPMOP. The authors compare the sequential and parallel versions of the proposed algorithm, as well as with a metaheuristic-based algorithm from literature (Coelho et al., 2008). Also, a comparison was performed between the solutions found by the parallel algorithm and those obtained by a mathematical programming model. The results show the efficiency of the parallel implementation since it solves the OPMOP with gains in the solutions quality of up to 242,13%.

3. MATHEMATICAL FORMULATION

This section presents a mixed integer programming (MIP) model based on goal programming (Romero, 2004) to solve OPMOP. This model was proposed by Coelho et al. (2008) and it is an extension of the model of Costa (2005). The model refers to production planning for an hour, replicated while there isn't any exhausted pit and operational conditions of the mine remain the same. The objective is to minimize the deviations of the production and quality goals and to reduce the number of vehicles required for the operation.

Let the parameters be:

- O : Set of ore pits;
- W : Set of waste rock pits;
- F : Set of ore and waste rock pits, ie, $F = O \cup W$;
- P : Set of control parameters analysed in the ore (% Fe, SiO₂, etc);
- S : Set of shovels;
- T : Set of mining trucks;
- O_r : Recommended rate of mining for ore (ton/h);
- O_l : Minimum rate of mining for ore (ton/h);
- O_u : Maximum rate of mining for ore (ton/h);
- W_r : Recommended rate of mining for waste rocks (ton/h);
- W_l : Minimum rate of mining for waste rocks (ton/h);
- W_u : Maximum rate of mining for waste rocks (ton/h);
- α^- : Penalty for negative deviation from the production of ore;
- α^+ : Penalty for positive deviation from the production of ore;
- β^- : Penalty for negative deviation from the production of waste rocks;
- β^+ : Penalty for positive deviation from the production of waste rocks;
- p_{ij} : Percentage of the control parameter j in pit i (%);
- pr_j : Recommended percentage for the control parameter j in the mixture (%);
- pl_j : Minimum allowable percentage for the control parameter j in the mixture (%);
- pu_j : Maximum allowable percentage for the control parameter j in the mixture (%);
- λ_j^- : Penalty for a negative deviation of the control parameter j in the mixture;
- λ_j^+ : Penalty for a positive deviation of the control parameter j in the mixture;
- ω_l : Penalty for use of the l -th truck;
- Qu_i : Maximum rate of mining for the pit i (ton/h);
- Tx_l : Maximum rate of use for the truck l (%);
- Sl_k : Minimum productivity for the shovel k (ton/h);
- Su_k : Maximum productivity for the shovel k (ton/h);
- cap_l : Capacity of truck l (ton);
- ct_{il} : Total cycle time of the truck l in the pit i (min);
- g_{lk} : 1, if the truck l is compatible with the shovel k ; and 0, otherwise.

Consider also the following variables of decision:

- x_i : Mining rate of the pit i (ton/h);
- y_{ik} : 1, if the shovel k operates in the pit i ; e 0, otherwise.
- n_{il} : Number of trips that a truck l performs to the pit i ;
- D_o^- : Negative deviation from the recommended ore production (ton/h);
- D_o^+ : Positive deviation from the recommended ore production (ton/h);
- D_w^- : Negative deviation from the recommended waste rock production (ton/h);
- D_w^+ : Positive deviation from the recommended waste rock production (ton/h);

- d_j^- : Negative deviation of the control parameter j in the mixture (ton/h);
 d_j^+ : Positive deviation of the control parameter j in the mixture (ton/h);
 U_l : 1, if the truck l is being used; and 0, otherwise.

Next, the equations (1)-(26) present the MIP model for the problem in focus.

$$\min \sum_{j \in P} \lambda_j^- d_j^- + \sum_{j \in P} \lambda_j^+ d_j^+ + \alpha^- D_o^- + \alpha^+ D_o^+ + \beta^- D_w^- + \beta^+ D_w^+ + \sum_{l \in T} \omega_l U_l \quad (1)$$

$$\sum_{i \in O} (p_{ij} - pu_j) x_i \leq 0 \quad \forall j \in P \quad (2)$$

$$\sum_{i \in O} (p_{ij} - pl_j) x_i \geq 0 \quad \forall j \in P \quad (3)$$

$$\sum_{i \in O} (p_{ij} - pr_j) x_i + d_j^- - d_j^+ = 0 \quad \forall j \in P \quad (4)$$

$$\sum_{i \in O} x_i \leq O_u \quad (5)$$

$$\sum_{i \in O} x_i \geq O_l \quad (6)$$

$$\sum_{i \in O} x_i + D_o^- - D_o^+ = O_r \quad (7)$$

$$\sum_{i \in W} x_i \leq W_u \quad (8)$$

$$\sum_{i \in W} x_i \geq W_l \quad (9)$$

$$\sum_{i \in W} x_i + D_w^- - D_w^+ = W_r \quad (10)$$

$$x_i \leq Qu_i \quad \forall i \in F \quad (11)$$

$$x_i \geq 0 \quad \forall i \in F \quad (12)$$

$$d_j^-, d_j^+ \geq 0 \quad \forall j \in P \quad (13)$$

$$D_o^-, D_o^+ \geq 0 \quad (14)$$

$$D_w^-, D_w^+ \geq 0 \quad (15)$$

$$\sum_{k \in S} y_{ik} \leq 1 \quad \forall i \in F \quad (16)$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in S \quad (17)$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in F, \forall k \in S \quad (18)$$

$$x_i - \sum_{k \in S} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad (19)$$

$$x_i - \sum_{k \in S} Cl_k y_{ik} \geq 0 \quad \forall i \in F \quad (20)$$

$$n_{il} ct_{il} - 60 \sum_{k \in S, g_{lk}=1} y_{ik} \leq 0 \quad \forall i \in F, \forall l \in T \quad (21)$$

$$x_i - \sum_{l \in T} n_{il} cap_l = 0 \quad \forall i \in F \quad (22)$$

$$\frac{1}{60} \sum_{l \in T} n_{il} ct_{il} \leq Tx_l \quad \forall i \in F \quad (23)$$

$$U_l - \frac{1}{60} \sum_{i \in F} n_{il} ct_{il} \geq 0 \quad \forall i \in F \quad (24)$$

$$n_{il} \in \mathbb{Z}^+ \quad \forall i \in F, \forall l \in T \quad (25)$$

$$U_l \in \{0, 1\} \quad \forall l \in T \quad (26)$$

The objective function (1) seeks to minimize the differences with regard to production goals of ore and waste rock, quality targets of the mixture, as well as to reduce the number of trucks used. The constraints (2)-(15) model the classic problem of blending with goals. In this group, the constraints (7) and (10) relate respectively to the care of the production targets of ore and waste rock, while the constraints (11) limit the maximum mining rate defined by the user for each pit.

The other constraints which complement the model can be divided into two groups. The first concerns the allocation of shovels and productivity range in order to justify the equipment use. The second is related to the allocation of trucks for material transport in the mine.

For the first group, the constraints (16) define that at most one shovel can be allocated to each pit, while the constraints (17) define that each shovel can be allocated to one pit at most. The constraints (18) define that the variables y_{ik} are binary. The constraints (19) and (20) limit respectively the maximum and minimum mining rate defined by the shovel allocated to the pit.

In the second group of constraints, each constraint (21) forces the truck to only perform trips where there is compatible shovel allocated. The constraints (22) are such that the mining rate of a pit is equal to the total production of the trucks allocated to the pit. The constraints (23) ensure that each truck l is in operation for at most $Tx_l\%$ in an hour. The constraints (24), together with the objective function, force the number of trucks used to be penalized. The constraints (25) force the number of trips that a truck performs to a pit to be a positive integer value. The constraints (26) indicate that the variables U_l are binary.

4. HEURISTIC MODEL

4.1 Representation of a Solution

A solution is represented by a matrix $R = [Y \mid N]$, where Y is a matrix $|F| \times 1$ and N is a matrix $|F| \times |T|$.

Each cell y_i of the matrix $Y_{|F| \times 1}$ represents a shovel k allocated to the pit i . A value -1 means that there isn't any shovel allocated to that pit. If there aren't any trips made to a pit i , the shovel k associated to that pit is considered *inactive* and it isn't penalized for a production below the minimum for a shovel (restrictions (18) of the mathematical programming model).

In the matrix $N_{|F| \times |T|}$, each cell n_{il} represents the number of trips that each truck $l \in T$ performs to a pit $i \in F$. A value 0 (zero) means that there aren't any trips allocated to that truck, while a value -1 indicates that the truck and the shovel allocated to that pit aren't compatible.

From Y , N and the cycle times from the matrix CT ($|F| \times |T|$ dimensional) the extraction rate at each pit is determined, as well as the sum of the cycle times for each truck.

4.2 Neighborhood

To explore the solution space of the problem, eight movements were developed and are presented below. Each movement is associated to a neighborhood defined by the function $N(\cdot)$.

Movement Number of Trips - $N^{NT}(s)$. This movement increases or decreases the number of trips of a truck l to a pit i where there's an allocated compatible shovel. Thus, in this movement, a cell n_{il} of the matrix N has its value increased or decreased by one trip.

Movement Load - $N^L(s)$. Consists of changing two separate cells y_i and y_k of the matrix Y , i.e. exchanging the shovels that operate in the pits i and k , if both pits have allocated shovels. If only one of the two pits has an allocated shovel and the other is available, this movement will relocate the shovel to the available pit. In order to maintain compatibility between shovels and trucks, the trips made to the pit are relocated along with the shovel.

Movement Relocate Trip from a Truck - $N^{TT}(s)$. Consists of choosing two cells n_{il} and n_{kl} from the matrix N and passing a trip from n_{il} to n_{kl} . Thus, in this movement, a truck l ceases to do a trip to a pit i and does it at another pit k . Compatibility restrictions between equipment are respected in this movement, so the trip relocation is only done when there's compatibility between them.

Movement Relocate Trip from a Pit - $N^{TP}(s)$. Two cells n_{il} and n_{ik} from the matrix N are chosen and a unit of n_{il} is relocated to n_{ik} . So this movement consists of relocating a trip from a truck l to a truck k that are both working at the pit i . Compatibility restrictions between equipment are respected in this movement, so the trip relocation is only done when there's compatibility between them.

Movement Pit Operation - $N^{PO}(s)$. Consists of removing from operation the shovel that is allocated to the pit i . The movement removes all the trips made to this pit, leaving this shovel *inactive*. The shovel is again put in operation as soon as a new trip is associated to it.

Movement Truck Operation - $N^{TO}(s)$. Consists of selecting a cell n_{il} from the matrix N and zero-fill its content, meaning that the truck l doesn't operate in the pit i anymore.

Movement Swap Trips - $N^{ST}(s)$. Two cells of the matrix N are selected and a trip is relocated between them. This movement can occur in any cell of the matrix N if compatibility restrictions between equipment are respected.

Movement Swap Shovels - $N^{SS}(s)$. Consists of swapping two separate cells y_i and y_k from the matrix Y , i.e. changing the shovels that operate in the pits i and k . This movement is similar to the movement L , because the shovels are also exchanged, but the trips made to the pits are not exchanged. To maintain compatibility between the shovels and trucks, the trips made to incompatible equipment are removed.

4.3 Evaluation of a Solution

As the developed movement can generate infeasible solutions, a solution is evaluated by a single-objective function $f : S \leftarrow \mathbb{R}$, where S represents the set of all possible s solutions generated from the movements developed in the previous section. This function f , defined by Eq. (27), to be minimized, consists of two parts: first, the objective function itself (Eq. (1) from the mathematical programming model) and second, a group of functions that penalize the occurrence of infeasibility in the current solution.

$$f(s) = f^{MP}(s) + f^P(s) + \sum_{j \in P} f_j^q(s) + \sum_{l \in T} f_l^u(s) + \sum_{k \in S} f_k^c(s) \quad (27)$$

In Eq. (27), $f^{MP}(s)$ is the objective function from the mathematical programming model given by Eq. (1), i.e. $f^{MP}(s)$ evaluates $s \in S$ considering the production and quality goals, as well as the number of trucks used; $f^P(s)$ evaluates s for disrespect of the production limits

established for the ore and waste rock; $f_j^q(s)$ evaluates s considering the infeasibility of the j -th control parameter; $f_l^u(s)$ evaluates s regarding disrespect of the maximum use rate of the l -th truck; and $f_k^c(s)$ evaluates s for disrespect of the productivity limits of the shovel k .

4.4 Initial Solution Generation

An initial solution to the problem is obtained by a partially greedy constructive procedure, similar to the construction phase of GRASP (Feo and Resende, 1995). The construction is done in two steps. The allocation of the shovels and the distribution of trips to the pits are done in the first step for the waste rock pits, and secondly, for the ore pits.

This strategy is adopted because in the waste rock pits, it is important to meet production and not necessary to observe the quality of the control parameters. The classification of the candidate elements to be inserted in the solution is made considering that for the waste rock pits, the best is the pit that has the greatest mass, the best is the shovel that provides the greatest production and the best is the truck that is the largest. For the ore pits, it is considered that the best pit is the one that has the least deviation of the control parameter levels in relation to the targets, the best is the shovel that provides the greatest production and the best is the truck that has the smallest capacity.

The choice of the ore pit in the second stage of the construction is done by a guide function, as in Bresina (1996). First, all pit candidates are sorted with respect to the deviation values of the quality goals. To the r -th better classified pit, that pit is associated to the guide function $bias(r) = 1/(2r)$. Each candidate pit is chosen with probability $p(r) = bias(r) / \sum bias(r)$. In this strategy, it is more likely to choose the pit that best helps to minimize the deviations from the quality targets.

4.5 A Hybrid Algorithm

We present a hybrid algorithm, called H-GVILS, that combines the power of the heuristic procedures GRASP (Feo and Resende, 1995), *Variable Neighborhood Descent* - VND (Mladenovic and Hansen, 1997) and *Iterated Local Search* - ILS (Lourenço et al., 2003), with a mathematical programming based module. When ILS reaches a certain level of perturbation, this module is triggered to solve smaller parts of the problem in optimality.

Figure 1 shows the H-GVILS algorithm.

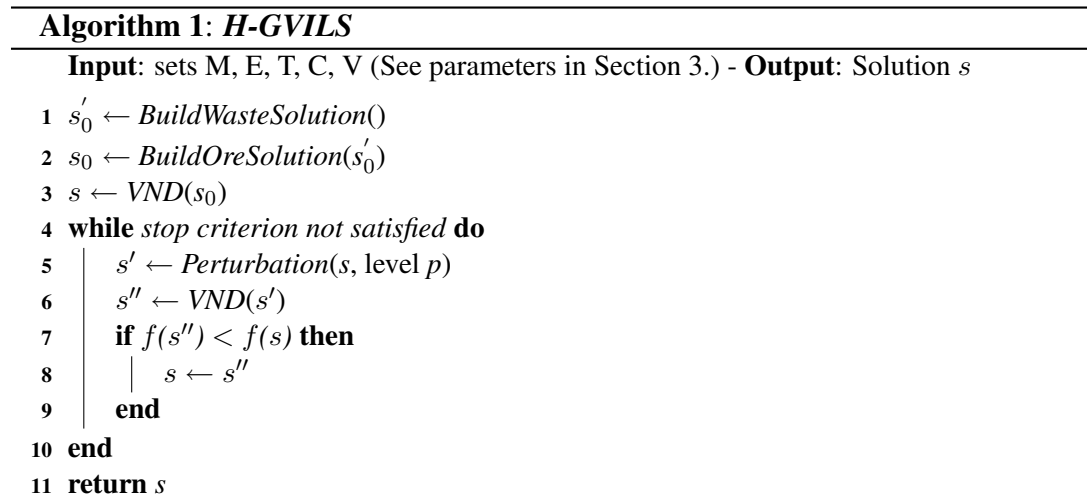


Figure 1: H-GVILS Algorithm

Building an initial solution s_0 (lines 1 and 2 of the Algorithm in Fig. 1) is made by the procedure described in Subsection 4.4. The local search (line 3 of the Algorithm in Fig. 1), in turn, uses the VND procedure with the movements described in Subsection 4.2. Strategically, the local search operates in the neighborhoods in a pre-defined order, beginning from the ones requiring less computational effort to those that require more effort. As in the preliminary tests some neighborhoods did not produce great local optimums or spend too much processing time to achieve a local optimum; only a small group of neighborhoods has been used in the local search. Thus, the VND uses the following order for the exploration of the neighborhoods: N^L , N^{NT} , N^{TT} and N^{TP} . The movements on neighborhoods N^{SS} , N^{TO} , N^{PO} and N^{ST} were used only as a disturbance, or perturbation, of the current solution.

The objective of the perturbation is to diversify the search, generating a different solution, which becomes increasingly *distant* from the current location in the search space. To fulfill this mission, different levels of disturbance are set. For a given level p of perturbation, $p + 2$ movements are applied to the current solution, each movement randomly chosen among 6 of those described in Subsection 4.2. Each movement is chosen according to a certain probability, given by: 30% for N^{NT} , 20% for N^{ST} , 20% for N^L , 10% for N^{SS} , 10% for N^{PO} and 10% for N^{TO} . This probability varies due to the greater influence of changing the number of trips, exchanging the trips and exchanging the equipment (mining trucks and shovels) in the quality of the final solution. The local search applied on the disrupted solution is based on the procedure VND (line 6 of the Algorithm in Fig. 1). After *IterMax* iterations without improvement in a given level, this level increases by one unit. If the local search method finds a better solution, the perturbation returns to the lowest level, in this case, $p = 0$.

When the perturbation level p reaches a limit H , the Mathematical Programming Module (MPM) is triggered to solve smaller parts of the problem in optimality. Initially, 90% of the pits are locked and the mathematical programming module finds the optimal solution of this reduced solutions space. At each step it considers 10% more pits until p reaches $H + 9$ and the complete mathematical programming model is solved. The MPM stops in two situations: (i) after finding the optimal solution or (ii) after τ seconds.

4.6 Post Optimization: Path Relinking

The Path Relinking technique was proposed by Glover (1996) as an intensification strategy to explore links between elite solutions generated during Tabu Search or Scatter Search (Glover and Laguna, 1997) metaheuristics. This search consists in generating and exploring paths in the solution space, starting from one or more solutions (called *base solutions*) in direction to other elite solutions (called *guide solutions*).

In our case, the *base solution* is the final solution generated by the H-GVILS procedure (Section 4.5) and the *guide solution* is the initial solution generated by GRASP procedure (lines 1 and 2 of the Algorithm in Fig. 1), as presented in Section 4.4. Our Path Relinking approach is based on *attributes locking* and the considered attribute was the *shovel position* in the solution. At each step, the procedure selects a pit in the *guide solution* and moves the shovel k (the one assigned to the selected pit) to the same pit in the *base solution*. The trips made to that pit are also exchanged to keep compatibility between shovels and trucks. Finally, a new local search is made in the *base solution* having both shovel k and its trips *locked* during the search procedure. The process is repeated until the *base solution* has all *guide solution* attributes, i.e., all the shovels and trips of the *base solution* are assigned to the same pits and trucks of the *guide solution*.

The local search is also based on the VND procedure of the H-GVILS algorithm (line 6 of the Algorithm in Fig. 1) and the same order of exploration of neighborhoods is used. However,

in this case these neighborhoods have an *extra* ability of preserving the locked attributes in the solution during the search.

5. SCENARIOS DESCRIPTION

The scenarios utilized for the tests refer to an iron mining company located in the state of Minas Gerais, Brazil, and it was considered an 85% maximum utilization rate for the trucks. Instances are available at <http://www.decom.ufop.br/prof/marcone/projects/mining.html>.

Considered were four scenarios, giving origin to the instances OPM01, OPM02, OPM03 e OPM04. On the two first instances, there are 17 pits, 8 shovels and 30 mining trucks available. The trucks from 1 to 15 have 50-ton capacity and are compatible with shovels Car01 to Car04, while the other trucks have 80-ton capacity and are compatible with shovels Car05 to Car08. For the instances OPM03 and OPM04, there are 32 pits, 7 shovels and 30 mining trucks, with all trucks having a 50-ton capacity and compatibility with all the shovels. Ten control parameters (chemical and/or granulometric) are considered in all instances.

6. COMPUTATIONAL RESULTS AND ANALYSIS

The proposed algorithm, so-called H-GVILS-PR, is composed by the H-GVILS algorithm (Section 4.5) and the Post Optimization Module based on the Path Relinking procedure (Section 4.6). H-GVILS-PR was coded in C++ programming language using the g++ 4.0 compiler and the IDE Eclipse 3.1. The mathematical programming model was done in AMPL and solved by CPLEX solver, version 11. The hybrid Mathematical Programming Module used the GLPK solver, version 4.29. Both heuristic and exact models were tested in a PC Intel Core 2 Quad (Q9550), 2.83 GHz, with 8 GB of RAM, running Ubuntu 9.04 kernel 2.6.28-15.

The H-GVILS-PR algorithm was applied 30 times for each instance and it was interrupted after 2 and 15 minutes of processing. The CPLEX solver also was interrupted within this time limit. This strategy was adopted because the time for the decision maker is small and 2-15 minutes is a typical value for the maximum tolerance in a real case.

The following weights were adopted in the evaluation function: $\alpha^- = \alpha^+ = \beta^- = \beta^+ = 100$, $\lambda_j^- = \lambda_j^+ = 1 \quad \forall j \in T$, $\omega_l = 1 \quad \forall l \in V$, $Tx_l = 85\% \quad \forall l \in V$. After a preliminary sequence of tests the three parameters of the proposed algorithm, *IterMax*, *H* and τ were set to: 60, 8 and 30, respectively.

Results were also compared to a pure metaheuristic version of the algorithm found in the literature, so-called GVILS (Coelho et al., 2008). This version of the algorithm is also based on the GRASP, VND and ILS metaheuristics, without the Post Optimization Module and the Mathematical Programming Module. The only parameter of the algorithm *IterMax* was also set to 60 after a preliminary set of tests.

Tables 1 and 2 compare the results found by the CPLEX, GVILS and H-GVILS-PR algorithms. In these tables, “Instance” column means the analyzed instance; “Best Known” represents the best known value found in literature for the instance; “CPLEX” indicates the value (upper bound) found by CPLEX; “Best” and “Average” indicate the best and average values, respectively, in thirty executions of both GVILS and H-GVILS-PR algorithms; “Gap” for CPLEX, GVILS and H-GVILS-PR are calculated as equations (28), (29) and (30), and indicate the deviation between the CPLEX, GVILS and H-GVILS-PR solutions, respectively, in relation to the best known results. Column “Imp” measures the improvement of the GVILS (or H-GVILS-PR), in terms of average value, over the CPLEX value.

$$gap_i^{CPLEX} = \frac{f_i^{CPLEX} - f_i^*}{f_i^*} \quad (28)$$

$$gap_i^{GVILS} = \frac{\bar{f}_i^{GVILS} - f_i^*}{f_i^*} \quad (29)$$

$$gap_i^{H-GVILS-PR} = \frac{\bar{f}_i^{H-GVILS-PR} - f_i^*}{f_i^*} \quad (30)$$

In Eq. (29) and Eq. (30), for each instance i , $\bar{f}_i^{GVILS}$ is the average value found in thirty executions by the GVILS algorithm; and $\bar{f}_i^{H-GVILS-PR}$ is the average value for the thirty H-GVILS-PR executions; and f_i^* is the best known value. In Eq. (28), f_i^{CPLEX} represents the upper bound found in instance i within the time limit.

Table 1: Results comparison between CPLEX and GVILS

Instance	Time (min)	Best Known	CPLEX		GVILS			Imp. (%)
			value	Gap (%)	Best	Average	Gap (%)	
OPM01	2	227.12	227.90	0.34	227.12	227.44	0.14	0.00
OPM02	2	252.41	254.06	0.66	253.37	259.68	2.88	-0.38
OPM03	2	164029.15	164029.76	0.00	164044.41	164099.43	0.04	-0.01
OPM04	2	164054.04	164054.04	0.00	164106.14	164204.60	0.09	-0.03
OPM01	15	227.12	227.90	0.34	227.12	227.16	0.02	0.00
OPM02	15	252.41	254.06	0.66	252.41	253.32	0.36	0.00
OPM03	15	164029.15	164029.76	0.00	164040.91	164059.98	0.02	-0.01
OPM04	15	164054.04	164054.04	0.00	164100.21	164121.80	0.04	-0.03

Table 2: Results comparison between CPLEX and H-GVILS-PR

Instance	Time (min)	Best Known	CPLEX		H-GVILS-PR			Imp. (%)
			value	Gap (%)	Best	Average	Gap (%)	
OPM01	2	227.12	227.90	0.34	227.12	227.15	0.01	0.00
OPM02	2	252.41	254.06	0.66	225.28	225.39	-10.71	10.75
OPM03	2	164029.15	164029.76	0.00	164017.23	164017.28	-0.01	0.01
OPM04	2	164054.04	164054.04	0.00	164017.66	164018.22	-0.02	0.02
OPM01	15	227.12	227.90	0.34	227.12	227.14	0.01	0.00
OPM02	15	252.41	254.06	0.66	225.27	225.28	-10.75	10.75
OPM03	15	164029.15	164029.76	0.00	164017.18	164017.29	-0.01	0.01
OPM04	15	164054.04	164054.04	0.00	164017.68	164017.88	-0.02	0.02

As can be observed in Tab. 1, the H-GVILS-PR algorithm was able to generate better solutions than the best known solutions available in the literature for the instances OPM02, OPM03 and OPM04, with the best improvement of 10.75% (OPM02). Both heuristic algorithms GVILS and H-GVILS-PR had obtained gaps up to 2.88%. In addition, H-GVILS-PR was found capable to obtain better solutions when compared to CPLEX in all instances. Finally, H-GVILS-PR produced better results than the GVILS algorithm, with respect to the quality of the final solutions and with smaller gaps.

7. CONCLUSIONS

This work dealt with the operational planning of mines considering the dynamic allocation of trucks. Because of the complexity of this combinatorial problem, we proposed a hybrid heuristic algorithm, called H-GVILS-PR, which combines the heuristic procedures GRASP, Variable Neighborhood Descent and Iterated Local Search, with a Mathematical Programming Module and a Post Optimization Module based on Path Relinking to solve it.

Using instances from literature, the proposed heuristic algorithm was compared to the optimizer CPLEX 11.0 applied to a mathematical programming model, also developed in this work. It was found that the H-GVILS-PR algorithm is competitive with CPLEX 11.0 solver, since H-GVILS-PR is able to find good quality solutions quickly with low variability. The proposed algorithm is also compared to its pure heuristic version (GVILS) and shows better results than this one. Since staff decisions have to be made quickly, the results validate the use of the proposed algorithm, as a tool for decision support.

Acknowledgements

The authors acknowledge CNPq (process 474831/2007-8) and FAPEMIG (process CEX 2991-06.1/07) for supporting the development of this research.

REFERENCES

- Alvarenga, G. B., 1997. Optimal dispatch of trucks in an iron mine using genetic algorithms with parallel processing (in portuguese). Master's thesis, Programa de Pós-Graduação em Engenharia Elétrica, Escola de Engenharia, UFMG, Belo Horizonte, Minas Gerais, Brazil.
- Boland, N., Dumitrescu, I., Froyland, G., & Gleixner, A. M., 2009. Lp-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. *Computers and Operations Research*, vol. 36, pp. 1064–1089.
- Bresina, J. L., 1996. Heuristic-biased stochastic sampling. In *Proceedings of the 13th National Conference on Artificial Intelligence*, AAAI Press, pp. 271–278, Portland.
- Chanda, E. K. C. & Dagdelen, K., 1995. Optimal blending of mine production using goal programming and interactive graphics systems. *International Journal of Surface Mining, Reclamation and Environment*, vol. 9, pp. 203–208.
- Coelho, I. M., Ribas, S., & Souza, M. J. F., 2008. Um algoritmo baseado em grasp, iterated local search para a otimização do planejamento operacional de lavra. In *Proceedings of the XI Encontro de Modelagem Computacional*, pp. 8 p., Volta Redonda (RJ).
- Costa, F. P., 2005. Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto. Master's thesis, Programa de Pós-Graduação em Engenharia Mineral, Escola de Minas, UFOP, Ouro Preto.
- Ezawa, L. & Silva, K. S., 1995. Dynamic allocation of trucks aiming quality (in portuguese). In *Proceedings of the VI Congresso Brasileiro de Mineração*, pp. 15–19, Salvador, Bahia, Brazil.
- Feo, T. A. & Resende, M. G. C., 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, vol. 6, pp. 109–133.

- Glover, F., 1996. Tabu Search and adaptive memory programming - advances, applications and challenges. In Barr, R., Helgason, R., & Kennington, J., eds, *Interfaces in Computer Sciences and Operations Research*, pp. 1–75. Kluwer Academic Publishers.
- Glover, F. & Laguna, M., 1997. *Tabu Search*. Kluwer Academic Publishers, Boston.
- Godoy, M. & Dimitrakopoulos, R., 2004. Managing risk and waste mining in long-term production scheduling of open-pit mines. *SME Transactions*, vol. 316, pp. 43–50.
- Guimaraes, I. F., Pantuza, G., & Souza, M. J. F., 2007. A computational simulation model to validate results by dynamic allocation of trucks in open-pit mines (in portuguese). In *Proceedings of the XIV Simpósio de Engenharia de Produção (SIMPEP)*, pp. 11, Bauru, São Paulo, Brazil.
- Lourenço, H. R., Martin, O. C., & Stützle, T., 2003. Iterated local search. In Glover, F. & Kochenberger, G., eds, *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston.
- Merschmann, L. H. C., 2002. Development of an optimization and simulation system for the analysis of production scenarios in open-pit mines (in portuguese). Master's thesis, Programa de Engenharia de Produção/COPPE, UFRJ, Rio de Janeiro, Brazil.
- Mladenovic, N. & Hansen, P., 1997. A variable neighborhood search. *Computers and Operations Research*, vol. 24, pp. 1097–1100.
- Papadimitriou, C. H. & Steiglitz, K., 1998. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., New York.
- Ribas, S., Coelho, I. M., Souza, M. J. F., & Menotti, D., 2009. Parallel iterated local search aplicado ao planejamento operacional de lavra. In *Proceedings of the XLI SBPO*, pp. 12 p., Porto Seguro (BA).
- Romero, C., 2004. A general structure of achievement function for a goal programming model. *European Journal of Operational Research*, vol. 153, pp. 675–686.
- Sgurev, V., Vassilev, V., Dokev, N., Genova, K., Drangajov, S., Korsemov, C., & Atanassov, A., 1989. Trasy - an automated system for real-time control of the industrial truck haulage in open-pit mines. *European Journal of Operational Research*, vol. 43, pp. 44–52.
- White, J. W., Arnold, M. J., & Clevenger, J. G., 1982. Automated open-pit truck dispatching at Tyrone. *Engineering and Mining Journal*, vol. 183, n. 6, pp. 76–84.
- White, J. W. & Olson, J. P., 1986. Computer-based dispatching in mines with concurrent operating objectives. *Mining Engineering*, vol. 38, n. 11, pp. 1045–1054.

An algorithm based on Iterated Local Search, Variable Neighborhood Descent and Tabu Search for the Integrated Vehicle and Crew Scheduling Problem

Marcone Jamilson Freitas Souza, Gustavo Peixoto Silva, Sabir Ribas, Igor Machado Coelho

Programa de Pós-Graduação em Engenharia Mineral, Departamento de Computação
Universidade Federal de Ouro Preto

Campus Universitário, 35.400-000 Ouro Preto, Minas Gerais, Brazil
{marcone, gustavo}@iceb.ufop.br, {sabarribas, igor.machado}@yahoo.com.br

Abstract

This work deals with the Integrated Vehicle and Crew Scheduling Problem (VCSP) in urban mass transit. The Vehicle Scheduling Problem (VSP) consists in creating a daily routine of operation for a fleet of vehicles of a company so that all timetabled trips are covered and the operational costs of such activity reduced, making the best use of the fleet. In the Crew Scheduling Problem (CSP) each crew corresponds to a duty, and the goal is the generation of its schedule, satisfying a set of labor agreement and operational rules of the company, as well as the workforce optimization. Traditionally, the VSP and the CSP are solved independently, e.g., by vehicles-first duties-second approach, but these methodologies do not explore the link between both problems. The difficulty of solving VCSP is greater than the isolated problems since it contains all the degrees of freedom of vehicle scheduling and all the degrees of freedom of crew scheduling. Since VCSP is NP-hard, an algorithm based on metaheuristics approaches is proposed here. This algorithm combines Iterated Local Search (ILS), Variable Neighborhood Descent (VND) and Tabu Search (TS). In order to explore the solution space, some moves based on reallocation and swap of both, trips of the vehicles schedule and tasks of the duties schedule are used. The algorithm was tested with real data provided by BHTRANS, the manager of the transit mass system of the city of Belo Horizonte, Brazil. The results show the effectiveness of the proposed approach.

Keywords: Vehicle Scheduling, Crew Scheduling, Integrated Vehicle and Crew Scheduling, Public Transport, Metaheuristics

1. Introduction

The public transportation system planning is a very complex activity. Thus it is generally divided into five main stages, which are: route planning, timetables defining, vehicle scheduling, crew scheduling and crew rostering. In general, the output of one stage is the input for the subsequent one. This work deals with the integration of two stages, the vehicle and crew scheduling problems.

According to Freling *et al.* (2003), there are three different approaches to solve the vehicle and crew scheduling: Sequential, Independent and Integrated.

In the sequential approach, the vehicle and crew scheduling problems are solved separately and the solution of one is used as input for the other. The sequential approach, in turn, can be classified as Traditional or Reverse.

The traditional sequential approach was the first one to be studied and it is widely adopted by the companies. It consists in solving the *vehicle scheduling problem* (VSP) and then, considering this solution as input, the *crew scheduling problem* (CSP) is solved. This approach has the inconvenience that the vehicle scheduling solution guides the crew scheduling solution (FREELING *et al.*, 2003), which has the most relevant cost among the ones involved in the system (BOUZADA, 2003). An alternative to solve it is to apply the traditional sequential approach, but considering the crew features during the vehicle scheduling solution. Even though the vehicle scheduling costs increase, it becomes easier to solve the CSP and much probably the total cost will be lower than the traditional approach. Scott (1985), Darby-Dowman (1988) and Reis (2006, 2007) use this strategy for solving the problem involving vehicle and crew scheduling.

In the reverse sequential approach, the CSP is solved first and then, considering this solution as input, the VSP is solved. This strategy is justified for two main motives: 1) the crew costs are higher than vehicle costs and 2) the set of vehicle restrictions is much smaller, which suggests solving CSP first, because it has fewer alternatives for solution. Bassi *et al.* (2007) propose a GRASP heuristic to solve the independent crew scheduling problem (ICSP), which builds the crew duties directly from the timetable, independently of the vehicle scheduling. In a second stage, the vehicle schedule problem is solved having as input the driver shifts already defined. A path-relinking mechanism was also implemented in order to improve the ICSP solution. The heuristic was implemented for several transport companies and the results were compared with the traditional approach. According to the authors, it can be concluded that this strategy is able to produce solutions with lower costs than the traditional process.

In the independent approach the vehicle schedule and the crew schedule are built directly from trip timetables. Therefore, in this approach the vehicle schedule is built without considering any crew characteristics and the crew schedule is obtained without considering vehicle constraints. Generally, the results obtained by this approach are not feasible in practice, because some vehicle trips will not have a driver, and some drivers will operate virtual vehicles that are not in the vehicle schedule. This approach is used to determine lower bounds for VSP and CSP.

In the integrated approach both problems are solved together in the same model. As both problems are NP-hard (GAREY and JOHNSON, 1979), then Vehicle and Crew Scheduling Problem (VCSP) is NP-hard. It is obvious that the difficulty of solving VCSP is bigger because the solution space is huge, since VCSP contains all the degrees of freedom of vehicle scheduling and all the degrees of freedom of crew scheduling. However, with the improvement in the CPU speed of the computers, combined with algorithm progress, the research has taken up this topic recently. Freling *et al.* (1995), Haase and Friberg (1999) and Freling *et al.* (1999) use this type of approach. See Borndröfer *et al.* (2006) and Freeling *et al.* (2003) for a survey and experience about this topic.

This paper proposes an algorithm which combines Iterated Local Search, Variable Neighborhood Descent and Tabu Search heuristics for solving VCSP. This heuristic strategy is justified because VCSP is NP-hard and according to the literature these approaches are able to produce good solutions for each problem, considered individually (See Souza *et al.*, 2007 and Marinho *et al.*, 2004).

The rest of the paper is organized as follows. Section 2 describes the problem. Section 3 presents the proposed methodology, including the representation of the problem, the procedure employed for generating an initial solution, the neighborhood structure, the evaluation function and finally the adaptation of the Iterated Local Search, VND and Tabu Search heuristics to the problem. Section 4 shows the results achieved in this study and discusses the improvement obtained. Section 5 brings the concluding remarks of this work.

2. Problem definition

According to Freeling *et al.* (2003), the Vehicle and Crew Scheduling Problem (VCSP) is defined as follows. Given a set of trips within a fixed planning horizon, the objective is to find a minimum cost schedule for the vehicles and the crews, such that both the vehicle and the crew schedules are feasible and mutually compatible. Each trip has fixed starting and ending times, and the traveling times between all pairs of location are known. A vehicle schedule is feasible if (1) each trip is assigned to a vehicle, and (2) each vehicle performs a feasible sequence of trips, where a sequence of trips is feasible if a vehicle can execute each pair of consecutive trips in the sequence. Some trips do not belong to any route and these are called the dead trips. A dead trip time is the time a vehicle takes to travel between two locations (terminals or depots) that do not belong to any route. This is necessary in order to move the vehicle from an ending point to the next starting point, so that the vehicle can start the next trip. A vehicle schedule defines which trips have to be performed by the same vehicle and this defines the so-called blocks. If there is enough time between one trip and the next, a change of driver may occur, and these are called relief points, defined by location and time. The blocks are divided by relief points, and each part is the so-called task, that is defined by two consecutive relief points and represents the minimum piece of work that can be assigned to a crew. These tasks have to be assigned to crew members, and the tasks of a crew member define a crew duty. All the duties constitute a crew schedule. Such schedule is feasible if (1) each task is assigned to one duty, and (2) each duty is a sequence of tasks that can be performed by a single crew, both from a physical and a legal point of view. In particular, each duty must satisfy several complicated constraints corresponding to work load regulations for the crews. Typical examples of such constraints are the maximum working time without a break, minimum break duration and maximum total working time. The cost of a duty is usually a combination of fixed costs such as wages, and variable costs such as overtime payment. The subsection 3.4 details the considered requirements in this work.

It is assumed that there is only one depot and all vehicles are always available.

3. Proposed methodology

In this section is presented a hybrid heuristic, which combines Iterated Local Search, Variable Neighborhood Descent and Tabu Search for reaching suboptimal solutions to the VCSP.

3.1. A solution representation to the problem

A solution s to the VCSP consists in a (s_v, s_c) pair, where s_v represents a solution to the VSP and s_c represents a solution to the CSP. The s_v solution consists in a set of vehicles, where each one has an associated list of trips to be realized during a day. The s_c solution consists in a set of crew duties, where each one is associated to a list of tasks to be carried out daily.

3.2. Construction of an initial solution

An initial solution to the VCSP is made by a sequential mechanism. To begin with, an initial solution for the vehicle scheduling is built. Later, from this vehicle solution, a crew scheduling solution is generated. Both solutions are built by a greedy heuristic.

An initial solution to the VSP is obtained by applying a constructive heuristic where at each iteration, a new trip not yet assigned is added to the current schedule of the best vehicle, in accordance to the value of the evaluation function relative to each vehicle scheduling (see eq. (2) in subsection 3.4). The choice of the trip to be added to the vehicle schedule is totally greedy, which means that the trip with lower cost will be chosen. The approach ends when all the trips are allocated. The trips belonging to each vehicle are known as the blocks of the vehicle.

From the blocks of each vehicle, a solution for the CSP is built. Initially, for each block, the trips are grouped in a task until a relief point is found. A relief point is defined by location and time, where and when a change of driver may occur.

Next, the tasks are ordered by their starting time. At each step, there is a set of non-empty duties and a duty without tasks. The constructive procedure works as follows. The first task is assigned to the first duty, the empty one. Next, for each task of the sequence, we evaluate its insertion for all the duties and also for a new duty without tasks. The evaluation of each crew duty is carried out according to the eq. (8), relative to the crew scheduling (see subsection 3.4). The task is assigned to the duty with the

lowest cost. The crew schedule stops when all the tasks are allocated.

3.3. Neighborhood Structure

With a view to defining the neighborhoods of a given solution s six moves were applied. The N_V^R , N_V^S , N_C^R and N_C^S neighborhoods use moves based on reallocation and swap of both, trips of the vehicles schedule and tasks of the crew duties schedule, respectively. The N_V^{RP} and N_V^{SP} neighborhoods consist in reassigning and swapping the trips of vehicles followed by the reconstruction of the duties affected by the moves according to the previous subsection, respectively.

3.4. Evaluation function

The cost function of VCSP is computed by the eq. (1):

$$f(s) = \alpha \times f_v(s) + \beta \times f_c(s) \quad (1)$$

where $f_v(s)$ represents the component of f that evaluates s with respect to the vehicle scheduling (see eq. (2)), $f_c(s)$ evaluates s with respect to the duty scheduling (see eq. (8)) and α and β are weights.

$$f_v(s) = \sum_{k \in F} f_v^k(s) + \omega_{CR} \times \text{ChangeRoutes} + \omega_{ESB} \times \text{ExcSplitBlocks} \quad (2)$$

In the eq. (2), $f_v^k(s)$ is the cost of the vehicle k of the fleet, obtained through the eq. (3), *ChangeRoutes* is the total number of changes of routes made by all the vehicles, ω_{CR} is the penalty associated to it, *ExcSplitBlocks* represents the excess of times that each vehicle returns to the depot and stays there more than 120 minutes and, finally, ω_{ESB} is the respective penalty. By operational restrictions, we observe that the maximum number of times that it can occur is 60% of the fleet size.

$$f_v^k(s) = \sum_{(i,j) \in T} C_{ij} + CD_{d1} + CD_{nd} \quad (3)$$

In the eq. (3), T is the set of n trips of the vehicle k , C_{ij} is obtained by the eq. (4) and represents the cost to perform the consecutive trips i and j ; CD_{d1} is the transportation cost of the vehicle k from the depot to the its first trip and CD_{nd} is the transportation cost of the vehicle k between its last trip and the depot.

$$C_{ij} = \begin{cases} CD_{ij} + CC_{ij} & \text{if } t_{ij} < 0 \\ CT_{ij} + CD_{ij} & \text{if } 0 \leq t_{ij} \leq \text{MaxTimeTerm} \\ CD_{id} + CD_{dj} & \text{if } t_{ij} > \text{MaxTimeTerm} \end{cases} \quad (4)$$

In the eq. (4), CT_{ij} , CD_{ij} and CC_{ij} represent, respectively, the time that the vehicle stays at the terminal between trips i and j , the time of a dead trip and the time in minutes that a trip i coincides with a trip j . In the eq. (4), *MaxTimeTerm* is the maximum time that a vehicle can stay at the terminal, established according to the operational politics of the Public Transport System and t_{ij} is the time, in minutes, that the vehicle is idle (i.e., without productive activity) between the trips i and j . Therefore t_{ij} is the time between the beginning of the trip j and the ending of the trip i minus the time of the dead trip if this one is necessary to reposition the vehicle.

The CD_{ij} and CT_{ij} costs are obtained as follows:

$$CD_{ij} = \begin{cases} (\alpha_1 \times dt_{ij})^{\beta_1} & \text{if } dt_{ij} \leq \text{TimeLimitDeadTrip} \\ dt_{ij} \times \omega_{TD} & \text{if } dt_{ij} > \text{TimeLimitDeadTrip} \end{cases} \quad (5)$$

$$CT_{ij} = \begin{cases} (\alpha_2 \times t_{ij})^{\beta_2} & \text{if } t_{ij} \leq \text{TimeLimitTerm} \\ t_{ij} \times \omega_{TT} & \text{if } t_{ij} > \text{TimeLimitTerm} \end{cases} \quad (6)$$

where: dt_{ij} is the time of dead trip between the trips i and j ; *TimeLimitDeadTrip* means the maximum time of the company schedule in which a dead trip can be done, ω_{TD} is the weight admitted if the dead trip time is greater than *TimeLimitDeadTrip*, *TimeLimitTerm* is the maximum time of the company schedule that a vehicle stays at the terminal; ω_{TT} is the weight considered if the vehicle stays at the terminal more than *TimeLimitTerm* minutes, $\alpha_1 \in [0, 1]$ and $\beta_1 \in [0, 1]$ are weights utilized to adjust the costs in eqs. (5) and (6), respectively.

The conflict CC_{ij} between trips i and j is evaluated by the eq. (7), as follows:

$$CC_{ij} = -(t_{ij}) \times \omega_{TC} \quad (7)$$

where ω_{TC} is the weight associated to this infeasibility.

Let $R = \{r_1, r_2, \dots, r_{|R|}\}$ be the set of routes of the company and $S_1, S_2, \dots, S_{|S|}$ subsets of R , called groups of routes, with $R = S_1 \cup S_2 \cup \dots \cup S_{|S|}$ and $S_1 \cap S_2 \cap \dots \cap S_{|S|} = \emptyset$.

The evaluation of the duty scheduling is given by the eq. (8), which must be minimized:

$$f_c(s) = \sum_{i \in \text{Duties}} (\text{PenaltyIdleTime}_i + \text{PenaltyOvertime}_i + \mu_3 \times \text{ChangeVehicles}_i + \mu_4 \times \text{ChangeRoutes}_i + \mu_5 \times \text{TerminalChange}_i + \lambda_1 \times \text{ConflictTime}_i + \lambda_2 \times \text{ExcTime}_i + \lambda_3 \times \text{DutyTimeInsuf}_i + \lambda_4 \times \text{ChangeVehiclesFor}_i + \lambda_5 \times \text{ChangeRoutesFor}_i + \lambda_6 \times \text{TerminalChangeFor}_i) + \lambda_7 \times \text{ExcessSplitDuty} + \delta \times n\text{Duties} \quad (8)$$

where:

- (i) Duties represents the set of duties;
- (ii) μ_i and λ_i are the penalties applied to the constraints and objectives, respectively;
- (iii) PenaltyIdleTime_i assumes the value μ_1 if the idleness of the crew on the duty i is greater than two hours and $0.347 \times \text{IdleTime}_i^{0.8}$, otherwise, where IdleTime_i corresponds to the time that each crew is idle on the duty i ;
- (iv) PenaltyOverTime_i assumes μ_2 when the overtime is greater than 120 minutes and $1.357 \times \text{Overtime}_i^{0.6}$, otherwise, where Overtime_i indicates the time, in minutes, that the crew i is in overtime payment;
- (v) ChangeVehicles_i indicates the number of times that the crew is reassigned to a different vehicle during the duty i and there is enough time to do the changes;
- (vi) ChangeRoutes_i indicates the number of times that the crew assigned to a duty i in the route r_j is reassigned to another route r_k , and both routes must belong to the same group, i.e., $r_j \in S_m \subset R$ and $r_k \in S_m$;
- (vii) TerminalChange_i indicates the number of times that the crew is reassigned to a different terminal during the duty i . This crew reassignment is allowed in the following situations: (1) if the terminal involved belongs to the same group of terminals; (2) if the involved tasks belong to the same vehicle and (3) if there is at least 120 minutes of idleness between two consecutive tasks of the duty i , that is, in a split duty, a crew reassignment to a different terminals is always allowed. On the other hand, a crew reassignment is forbidden if the reallocation does not satisfy none of the three items previously described;
- (viii) ConflictTime_i represents the total time, in minutes, with respect to the duty i , that there are tasks in conflict, for example, two consecutive tasks of the same crew, when the second starts before the first ends;
- (ix) ExcTime_i represents the time, in minutes, with respect to the duty i , that surpasses nine hours of daily work. The worked time in each duty i is computed as follows: $\text{WorkedTime}_i = (\text{final time of the last task of the duty } i) - (\text{the initial time of the first task}) - (\text{the duration of the split, if the duty is classified as split duty})$;
- (x) $\text{DutyTimeInsuf}_i = \max \{0, 11 \times 60 - (1440 - \text{time in which duty } i \text{ ends} + \text{time in which duty } i \text{ begins})\}$. It occurs because between two consecutive labor days must be 11 hours of resting. Therefore, this factor represents the amount of time, in minutes, in respect to the duty i , in which this labor law is not verified;
- (xi) $\text{ChangeVehiclesFor}_i$ indicates the number of times that the crew is reassigned to a different vehicle during the duty i and there are not enough time to do the changes;
- (xii) ChangeRoutesFor_i indicates the number of times that a crew allocated to a route r_j in a duty i is also allocated to another route r_k not belonging to the same group, i.e., $r_j \in S_m \subset R$ and $r_k \notin S_m$;
- (xiii) $\text{TerminalChangeFor}_i$ represents the number of times during the duty i that there are forbidden reassignments in relation with terminals (See item (vii));
- (xiv) ExcSplitDuty represents the number of split duties in the solution that is greater than $1.25 \times (\text{fleet size})$. It represents an estimative of the company that assumes that at most 50% of the crews are necessary to realize split duties.
- (xv) $n\text{Duties}$ represents the number of duties and δ is the penalty applied to minimize the number of duties in the schedule;

3.5. The ILS-VND-TSAR algorithm

With an aim to introduce the Variable Neighborhood Descent – VND (Hansen and Mladenovic, 2003), let N^k ($k=1, \dots, k_{\max}$) be a finite set of pre-selected neighborhood structures, and $N^k(s)$ the set of solutions in the k th neighborhood of s . Neighborhoods N^k may be induced from one or more metric (or quasi-metric) functions introduced into a solution space S . An *optimal solution* is a feasible solution where a minimum of the evaluated function f is reached. There is a *local minimum* s' of f with respect to $N^k(s)$, if there is no solution $s \in N^k(s')$ such that $f(s) < f(s')$. Metaheuristics based on local search procedures try to continue the search by other means after finding the first local minimum. VND is based on three simple facts: (1) A local minimum with respect to one neighborhood structure is not necessary so with another; (2) A global minimum is a local minimum with respect to all possible neighborhood structures; and (3) For many problems local minimum with respect to one or several are relatively close to each other.

According to these authors, this last observation, which is empirical, implies that a local optimum often provides some information about the global one. This may for instance be several variables with the same value in both. However, it is usually not known which ones are such. An organized study of the neighborhood of this local optimum is therefore in order, until a better one is found.

The Variable Neighborhood Descent (VND) approach is obtained if a change of neighborhoods is performed in a deterministic way

and its main steps are presented in Figure 1.

Procedure VND	
1:	Input: $s, f(\cdot)$, the set of neighborhood structures N^k , for $k = 1, \dots, k_{max}$ that will be used in the search
2:	$k \leftarrow 1$
3:	while $k \leq k_{max}$ do
4:	Find the best neighbor $s' \in N^k(s)$
5:	if $f(s') < f(s)$ then
6:	$s \leftarrow s'$
7:	$k \leftarrow 1$
8:	else
9:	$k \leftarrow k + 1$
10:	end if
11:	end while
12:	return s

Figure 1 – Basic VND procedure

For large instances, finding the best neighbor at each iteration (line 4 of Figure 1) is quite costly. In these cases, a random approach is normally used and this is what is done here. A classical Random Descent (RD) procedure works as follows. A point s' is generated at random from the k th neighborhood of s (line 4 of Figure 1). If s' is better than the incumbent, then s is updated to s' ($s \leftarrow s'$) and the search continues from s . If there are not any improvements in *DescentMax* iterations, then the search stops. Therefore, the procedure can return not necessarily a local optimum. In case of improvement, the search starts from the first neighborhood ($k \leftarrow 1$). This mechanism is justified because the neighborhoods are nested such that N^{k+1} is more computationally complex than N^k .

The proposed procedure to do a local search in the solution space of the VCSP, called VND_2_LEVELS, is described in the Figure 2. This procedure is based on the ideas of the VND heuristic in the sense that not only the neighborhoods are changed, but also the local search procedures. In this procedure, s_0 is a resulting solution of the perturbation in a local optimum. The RDVVP (s_0, N_V^{RP}) is the Random Descent Procedure that makes use of reallocation movements belonging to the N_V^{RP} neighborhood structure, and the tasks of the crews are allocated in a greedy way, ignoring the weights of the crew scheduling, that is, $\beta = 0$ in equation (1). The RDVVP (s_1, N_V^{SP}) is the Random Descent Algorithm that makes swapping movements of the N_V^{SP} neighborhood, ignoring the weights of the crew scheduling function. The RDVVI (s_4, N_V^{RP}) and RDVVI (s_5, N_V^{SP}) are similar to the RDVVP (s_0, N_V^{RP}) and RDVVP (s_1, N_V^{SP}), respectively, but in these cases the weights of the crew scheduling function are computed. These random approaches are interrupted if no improvements are found in *DescentMax* iterations. The TSAR_R and TSAR_S are algorithms based on Tabu Search metaheuristic with Adaptive Relaxation (Glover, 1996) that makes use of the neighborhoods N_C^R and N_C^S , respectively. These algorithms consist in changing the weights of the infeasibilities periodically during the search, either promoting the generation of infeasible solutions or feasible solutions in order to explore other regions not yet visited in the solution space. These algorithms were used by Marinho *et al.* (2004) to solve the Crew Scheduling Problem (CSP) and a detailed description can be found in their work.

Procedure	VND_2_LEVELS
Initialization	Let s_0 be a initial solution
Step 1	$s_1 \leftarrow \text{RDVVP}(s_0, N_V^{RP})$ $s_2 \leftarrow \text{RDVVP}(s_1, N_V^{SP})$ If the value of s_2 is lower than s_0 , do $s_0 \leftarrow s_2$ and return to the beginning of the Step 1
Step 2	$s_3 \leftarrow \text{TSAR\_R}(s_2)$ $s_4 \leftarrow \text{TSAR\_S}(s_3)$ If s_4 is better than s_2 , do $s_2 \leftarrow s_4$ and run the Step 2 again
Step 3	$s_5 \leftarrow \text{RDVVI}(s_4, N_V^{RP})$ $s_6 \leftarrow \text{RDVVI}(s_5, N_V^{SP})$ If s_6 is better than s_4 , do $s_4 \leftarrow s_6$ and executes again the Step 3 If there was some improvement in this step, return to the Step 2
Step 4	Return s_6

Figure 2 – VND_2_LEVELS procedure applied to the VCSP

The Iterated Local Search – ILS procedure (Lourengo, Martin and Stützle, 2003) is a metaheuristic with four basic components: *GenerateInitialSolution*, *LocalSearch*, *Perturbation* and *AcceptanceCriterion*. The *GenerateInitialSolution* is a module that consists in building a good solution to the problem. *LocalSearch* is the module that starts the search from a solution and returns a local optimum. *Perturbation* consists in realizing modifications on the local optimum in order to escape from it. Finally, the procedure *AcceptanceCriterion* consists in determining whether the new solution is accepted or not as the new current solution. According to the authors, a reasonable first guess for the acceptance criterion is to force the cost to decrease. *AcceptanceCriterion* has a strong influence on the nature and effectiveness of the walk in the space of solutions. Roughly, it can be used to control the balance between intensification and diversification of that search.

The Figure 3 presents the pseudo-code of ILS-VND-TSAR, an ILS procedure combined with VND and Tabu Search, applied to the VCSP.

Algorithm ILS-VND-TSAR

```

1:  Input:  $s, f(\cdot), LevelMax, ILSmax, TimeMax$ 
2:   $s^* \leftarrow s$ 
3:   $level \leftarrow 0$ 
4:  while  $level < LevelMax$  do
5:     $iter \leftarrow 0$ 
6:    while  $iter < ILSmax$  and  $time < TimeMax$  do
7:       $s \leftarrow Perturbation(s, level)$ 
8:       $s \leftarrow VND\_2\_LEVELS(s)$ 
9:      if  $f(s) < f(s^*)$  then
10:         $s^* \leftarrow s$ 
11:         $level \leftarrow 0$ 
12:         $iter \leftarrow 0$ 
13:      else
14:         $s \leftarrow s^*$ 
15:         $iter \leftarrow iter + 1$ 
16:      end if
17:    end while
18:     $level \leftarrow level + 1$ 
19:  end while
20:  return  $s^*$ 

```

Figure 3 – ILS-VND-TSAR algorithm applied to the VCSP

Six levels of perturbations have been considered. A perturbation of level i consists in realizing $i + 2$ random movements in the N_V^{RP} neighborhood.

4. Computational results

The proposed algorithm, so-called ILS-VND-TSAR, was implemented in C++ language using the Borland C++ Builder environment, version 5.0 and tested in a PC Intel Pentium IV, with 3.0 GHz and 1 GB of RAM running Windows XP Professional Edition.

Preliminary tests were performed to calibrate the parameters of the ILS-VND-TSAR algorithm. The number of iterations without improvement of the ILS ($ILSmax$) was fixed at 50. The number of iterations without improvement of the TSAR ($BTmax$) was fixed at 500. The $LevelMax$ value was fixed at 6 and the number of iterations without improvement of the Random Descent approaches ($DescentMax$) was fixed as number of trips $\times$ number of vehicles. The total processing time was limited to half an hour. The first phase of the traditional sequential approach uses ILS algorithm described in Souza *et al.* (2007) and second one uses the TSAR algorithm described in Marinho *et al.* (2004). Both algorithms stop after 15 minutes of execution.

In order to test the algorithms, we examined scheduling problems for one public transportation company that operates in the city of Belo Horizonte, Brazil.

Tables (1) and (2) show the value of the parameters used in the evaluation function, described in section 3.4. In the Table (1), the values of the parameters $TimeLimDeadTrip$ and $TimeLimTerm$ are linked to each instance and represent the maximum time of a dead trip and the maximum waiting time of the vehicles at the terminal, respectively, in minutes, found in the solution utilized by the company.

Table 1 – Instances and their respective time limits with respect to the VSP

Instances	$TimeLimDeadTrip$	$TimeLimTerm$
G02_SUN_90V	18	83
GO2_MON_260V	18	82
G02_FRI_260V	18	77
G02_SAT_172V	18	75

Table 2 – Parameters of the VCSP evaluation function

Parameter	Value	Parameter	Value
$MaxTimeTerm$	119 minutes	μ_1	18
α_1	2	μ_2	40
β_1	$\ln(16) / \ln(\alpha_1 \times TimeLimDeadTrip)$	μ_3	19
α_2	1	μ_4	5
β_2	$\ln(16) / \ln(\alpha_2 \times TimeLimTerm)$	μ_5	5
ω_{TD}	50	λ_1	40
ω_{TT}	40	λ_2	40
ω_{TC}	80	λ_3	40
ω_{ESB}	80	λ_4	45
ω_{CR}	5	λ_5	35
δ	100	λ_6	35
α	1	λ_7	45
β	2.5		

Table 3 contains the evaluation function values of the traditional sequential and integrated approaches. It can be observed that both best and average values were reduced in all cases analyzed and the gap between these values in each instance is also small, ratifying that the proposed algorithm is robust. In addition, there was an improvement of up to 4.19% if compared to the best solution found in the sequential approach.

Table 3: Evaluation function values from the traditional sequential and integrated approaches

Instance	Sequential Approach			Integrated Approach			
	Best Value	Average Value	Gap	Best Value	Average Value	Gap	Improvement
G02_SUN_90V	7036	7160,35	6,47%	6725,5	6841,5	1,72%	4,41%
G02_MON_260V	22507,5	22828,55	8,26%	21087,5	21297,7	1,00%	6,31%
G02_FRI_260V	23045,5	23354,1	7,50%	21724,5	21910,1	0,85%	5,73%
G02_SAT_172V	12938,5	13260,5	8,51%	12220	12468,05	2,03%	5,55%

Table 4 presents the details from the best solutions found on each instance, in respect of the sequential and integrated approaches.

Table 4: Schedules from the sequential and integrated approaches

G02 Company	Sequential Approach				Integrated Approach			
	SUN	MON	FRI	SAT	SUN	MON	FRI	SAT
Number of trips	90	260	260	172	90	260	260	172
Number of vehicles	11	40	40	23	11	40	39	23
Vehicle waiting time (hh:mm)	16:13	22:26	5:56	21:02	13:44	8:12	22:14	21:02
Dead trip time (hh:mm)	8:24	12:50	12:14	16:48	9:50	15:06	14:58	18:36
Number of split blocks	3	21	20	5	5	24	24	8
Number of split duties	12	50	50	16	7	49	43	20
Number of crews	21	62	63	37	19	62	63	34
Crew idleness (hh:mm)	15:13	15:04	9:23	15:44	11:07	7:52	6:28	15:12
Overtime payment (hh:mm)	5:04	6:44	7:34	20:08	14:58	8:28	12:31	19:44
Change of routes	1	5	7	1	4	4	6	2
Change of terminals	2	18	19	9	3	11	8	10
Change of vehicles	6	62	66	11	2	34	34	15
f_v	841	2510	2553	1491	893	2740	2887	1585
f_c	2478	7999	8197	4579	2333	7339	7535	4254
$f = f_v + 2.5 \times f_c$	7036	22507.5	23045.5	12938.5	6725.5	21087.5	21724.5	12220

From Table 4 it can be verified that the proposed algorithm was able to reduce the number of crews (From 21 to 19 on Sunday, from 37 to 35 on Saturday), the waiting time of the vehicles at the terminals (From 16:13 to 15:31 on Sunday, from 22:26 to 21:12 on Monday, from 5:56 to 2:24 on Friday and from 21:02 to 20:23 on Saturday), the crew idleness on Sunday (From 15:13 to 10:47) and the number of changes of terminals on Monday and Friday (From 18 to 13 on Monday and from 19 to 14 on Friday). It can be

observed that the number of crews grew up on Friday, but on the other hand there was a reduction on the overtime payment and on the number of changes of routes and terminals. Soares *et al.* (2006) has shown that there is a trade-off between, on the one hand, the number of trips plus crew idleness and, on the other hand, the overtime payment. This trade-off can be handled by manipulating the weights of the evaluation function. Also, it is important to observe that the integrated approach produces worst vehicle schedule, but its crew schedule are better than the ones produced by the sequential approach.

7. Conclusions

This paper dealt with the Integrated Vehicle and Crew Scheduling Problem. In order to solve it, an algorithm based on Iterated Local Search, Variable Neighborhood Descent and Tabu Search with Adaptive Relaxation, so-called ILS-VND-TSAR, is proposed. It is an adaptation of the ILS algorithm described in Souza *et al.* (2007) for solving the vehicle scheduling problem and the TSAR algorithm developed by Marinho *et al.* (2004) for the crew scheduling problem. Two new neighborhoods are introduced, which consist in reassigning and swapping the trips of vehicles followed by the reconstruction of the duties affected by the moves.

A comparison between the integrated approach and the traditional sequential one is performed. The first considers both problems simultaneously, while the later first solves the vehicle scheduling and then solves the crew scheduling. The results showed that the integrated approach was able to produce better results, with global improvement of up to 4.19%. In addition, it can be verified that the proposed algorithm is robust since the maximum variation of the final solutions is quite small, namely, 1.96%.

It is important to point out that the integrated approach produced a worst vehicle schedule, but its crew schedule was better than the traditional sequential one. This fact clearly illustrates that producing a good quality solution to the vehicle scheduling may not necessarily lead to a good quality solution to the crew scheduling.

8. Acknowledgements

The authors thank FAPEMIG (grant TEC 679/06), CNPq (grant 474831/2007-8) and the Federal University of Ouro Preto for the support given to the development of this study, as well as to BHTRANS for providing the data.

9. References

1. Bassi, H. V., Silva, G. P., Marinho, E. H., Souza, M. J. F. (2007) GRASP and Path-relinking Applied to the Independent Crew Scheduling Problem, In: International Conference on Computers in Urban Planning and Urban Management, 2007, Foz do Iguaçu, Brazil, *Reviewed papers in the Conference Proceedings of CUPUM 2007*, v. 1, p.1 – 11.
2. Borndröfer, R.; Grötschel, M. and Pfetsch, M. (2006) Public transport to the fORE!, *ORMS Today*, v. 33(2), p. 30-40.
3. Bouzada, C.F. (2003) *Costs of the Public Transport System by Bus* (in portuguese), Editora C/Arte, Belo Horizonte, Brazil.
4. Darby-Dowman, K., Jachnik, J. K.; Lewis, R. L. and Mitra, G. (1988) Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience, in J. R. Daduna and A. Wren (eds.), *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop*, Springer Verlag, Berlin, p. 226-239.
5. Feo, T. A., Resende, M. G. C. (1995) Greedy randomized adaptive search procedures, *Journal of Global Optimization*, v. 6, p. 109-133.
6. Freeling, R. C.; Boender, G. E. and Paixão, J. M. P. (1995) An integrated approach to vehicle and crew scheduling, Technical Report 9503/A, Econometric Institute, Erasmus University Rotterdam, Rotterdam.
7. Freling, R., Huisman, D., Wagelmans, A. (2003) Models and Algorithms for Integration of Vehicle and Crew Scheduling, *Journal of Scheduling*, v. 6(1), p. 63–85.
8. Freeling, R.; Wagelmans, A. P. M. and Paixão, J. M. P. (1999) An overview of models and techniques for integrating vehicle and crew scheduling, in N. H. M. Wilson (ed.), *Computer-Aided Transit Scheduling*, Springer Verlag, Berlin, p. 441-460.
9. Garey, M. R., Johnson, D. S. (1979) *Computers and Intractability*, In W. H. Freeman (ed.), *A Guide to the Theory of NP-Completeness*, New York.
10. Glover, F. (1986) Future paths for integer programming and artificial intelligence, *Computers and Operations Research*, v. 13(5), p. 533-549.
11. Glover, F. (1996) Tabu search and adaptive memory programming – advances, applications and challenges, In R. Barr, R. Helgason, J. Kennington (Eds.), *Interfaces in Computer Science and Operations Research*, p. 1-75.
12. Glover, F., Kochenberger, G. (2003) *Handbook of Metaheuristics*, Kluwer Academic Publishers.
13. Glover, F., Laguna, M. (1997) *Tabu Search*, Kluwer Academic Publishers.
14. Haase, K. and Friberg, C. (1999) An exact branch and cut algorithm for the vehicle and crew scheduling problem, in N. H. M. Wilson (ed.), *Computer-Aided Transit Scheduling*, Springer Verlag, Berlin, p. 63-80.
15. Hansen, P.; Mladenovic, N. (2003) Variable neighborhood search: Principles and applications, *European Journal of Operations Research*, v. 130, p. 449-467.
16. Lourenço, H. R.; Martin, O.; Stützle, T. (2003) Iterated Local Search, In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers.
17. Marinho, E. H., Ochi, L. S., Drummond, L. M. A., Souza, M. J. F. and Silva, G. P. (2004) Tabu Search Applied to the Bus Driver Scheduling Problem (in portuguese), In Simpósio Brasileiro de Pesquisa Operacional, 2004, São João Del Rey, Brazil, *Proceedings of the XXXVI SBPO*, Rio de Janeiro: SOBRAPO, 2004, v. 1. p. 1471 – 1482.
18. Reis, J. v. A. (2007) *Integrated Vehicle and Bus Driver Scheduling* (in portuguese) Exame de qualificação de mestrado, Programa de pós-graduação em Engenharia de Transportes, São Paulo: Escola Politécnica da Universidade de São Paulo, Brazil.
19. Reis, J. v. A. (2006) *Optimization techniques applied to the daily vehicle and crew scheduling* (in portuguese) Monografia de graduação em Engenharia de Produção, Departamento de Computação, Ouro Preto, Brazil: Universidade Federal de Ouro Preto, 70 p.

20. Soares, G. F., Silva, G. P. Marinho, E. H., Simões, E. L., Souza, M. J. F. (2006) Optimization in the Public Transport System (in portuguese). *Proceedings of the 16th Simpósio de Pesquisa Operacional e Logística da Marinha – SPOLM*, Escola de Guerra e Mar, Brazil.
21. Scott, D. (1985) A large linear programming approach to the public transport scheduling and cost model, in J. M. Rousseau (ed.), *Computer Scheduling of Public Transport 2*, North Holland, Amsterdam, p. 473-491.
22. Souza, M. J. F., Silva, G. P., Simões, E. M. L. (2007) Vehicle Scheduling: an heuristic approach (in portuguese) In: *Transporte em transformação XI: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2006*. Brasília: Positiva, 2007, v. 1, p. 39-57.

An Efficient Algorithm for the Dynamic Space Allocation Problem

Geiza Cristina da Silva
Tiago Geraldo Ferreira
Tatiana Alves Costa

Universidade Federal de Ouro Preto
Departamento de Ciências Exatas e Aplicadas
Rua 37, 115 – Loanda
35930-970 João Monlevade – MG – Brasil
{geiza, tatiana}@decea.ufop.br
tiago.g.f@gmail.com

Paulo Oswaldo Boaventura Netto

Universidade Federal do Rio de Janeiro
Programa de Engenharia de Produção
Centro de Tecnologia, Sala F113
21949-900 Rio de Janeiro – RJ- Brasil
boaventu@pep.ufrj.br

Luiz Satoru Ochi

Universidade Federal Fluminense
Departamento de Ciência da Computação
Rua Passo da Pátria 156 – Bloco E – Sala 328
24210-240 Niterói – RJ – Brasil
satoru@ic.uff.br

Abstract

The problem of dynamic space allocation (DSAP) was recently formalized in the literature. It was inspired by the need of optimizing the distances traveled by the resources associated to the activities of a project. Owing to its highly combinatorial nature it is difficult to solve exactly and even a feasible solution is not straightforward. In this work we propose new heuristic algorithms combining construction and refinement techniques. The results of these heuristics show them as being competitive for the available instances from the literature.

Keywords: Problem of Dynamic Space Allocation, Tabu Search, Metaheuristics.

1. Introduction

Heuristic techniques are used to solve problems of artificial intelligence when one of the following situations arises (Luger 2004): either the problem may not have an exact solution owing to inherent ambiguities in its formulation or because there are missing or unavailable data; or the problem may have an exact solution, but the computing cost of determining it is prohibitively high. The Dynamic Space Allocation Problem (DSAP) is a combinatorially explosive problem of the second type, where the number of possible states grows exponentially with search depth. In these cases, brute-force exhaustive search techniques, both in depth or in breadth, may not arrive at a solution within an acceptable time. The use of heuristic techniques to deal with this complexity by guiding searches through the solution space along promising paths is thus justified.

2. The dynamic space allocation problem (DSAP)

DSAP can be described as follows: given a project divided into periods, a set of activities has to be completed in each period. Each activity demands a set of necessary *resources* in order to be accomplished, with the resources not used in a period by any of its activities being considered to be *idle*. The problem is how to associate resources with appropriate locations, that is, associate resources used by activities with *workspaces* and idle resources with *storage spaces* or *depots*, in such a manner that the total distance traveled by the resources between locations is kept to a minimum.

The problem was first conceived for resource management during the construction of electrical power distribution grids, it was later found to have potential applications in projects where moving resources is difficult or where resource congestion is undesirable, such as bridge construction and mining activities.

In every case, space is a scarce resource which makes the reduction of materials handling a key factor, through the efficient allocation of activities and active or idle equipment and building materials, with efficient use of space being fundamental to reducing

congestion.

Despite the fact that the DSAP is a recently formulated problem, its importance can be justified by applications such as the above. From a theoretical point of view, the relevance of DSAP derives from its association with two widely-studied combinatorial optimization problems, the dynamic facility layout problem (DFLP) and the quadratic assignment problem (QAP).

The QAP is one of the most difficult among NP-hard problems [6]. It can be defined as the allocation of n facilities to n locations. The distances between pairs of locations and the flows of some type of demand being known, the objective is to find a minimum cost allocation.

The DFLP aims to minimize the cost of materials flow and the reallocation cost of each facility to a specific location at multiple periods. The problem can then be seen as a succession of QAP's, that is, a QAP has to be solved for each period.

A quick bibliography search produced little work related to DSAP. In [9] the problem was defined and a mathematical programming formulation was used together with a solver to obtain exact solutions for some of the proposed cases. Two instances based on simulated annealing were also proposed in order to obtain approximate solutions.

In [8] construction algorithms and a tabu search were proposed and the results thus obtained were compared with those in the literature. The article reported that tabu search outperformed the latter ones both in solution quality and computing time.

Problem data are:

- The *project agenda*, where the *total number* of activities, which activities have to be completed within each period and the required resources for each activity are specified;
- The *locations* (workspaces and depots), the *distances* between them and the capacity of each location, that is, the quantity of resources each one can host.

The problem definition assumes that a single activity can be associated with each workspace in a given period and that the capacity of this workspace is sufficient to accommodate its associated resources. In addition, each activity requires only one workspace and at least one resource.

In order to avoid unnecessary resource movement it is assumed that, if an activity is carried out during two or more consecutive periods, it should stay at the same workspace. In a similar manner, a resource which remains idle during multiple consecutive periods should be associated with the same depot during all of them.

A DSAP solution is then composed of two matrices: an *activity matrix* that associates periods, workspaces and activities, and an *idle resource matrix* that describes periods, depots, idle resources and the total distances to be traveled by resources.

In the following example, Table 1 presents the agenda for a small instance case with 3 periods, 4 activities and 8 resources.

Periods	Activities	Resources	Idle Resources
1	A1	4,5,8	6,7
	A2	1,23	
2	A2	1,23	4,5,8
	A3	6,7	
3	A3	6,7	1,2,3,5
	A4	4,8	

Table 2 shows the distances among the locations, where locations 1 to 3 are workspaces and locations 4 to 6 are depots. We take as a given in this instance that each location can support up to 3 resources.

Table 2. Distance matrix among locations

	1	2	3	4	5	6
1	0	1	2	1	2	3
2	1	0	1	2	1	2
3	2	1	0	3	2	1
4	1	2	3	0	1	2
5	2	1	2	1	0	1
6	3	2	1	2	1	0

Tables 3 and 4 present an instance solution for one resource's total distance traveled of 10 units. Table 3 shows the allocation of activities and their resources to workspaces (ws_i) and Table 4 shows the association between idle resources and depots (d_i).

Periods	ws_1	ws_2	ws_3

Periods	ws ₁	ws ₂	ws ₃
1	A1(4,5,8)	A2(1,23)	
2		A2(1,23)	A3(6,7)
3	A4(4,8)		A3(6,7)

Table 4. Idle resources matrix			
Periods	d ₁	d ₂	d ₃
1			6,7
2	4,5,8		
3	5	1,2,3	

3. Mathematical Formulation

The DSAP can be written as quadratic integer programming by using the following notation:

Index sets:

- J = total number of activities ($j = 1, 2, \dots, J$);
- T = total number of resources ($t = 1, 2, \dots, T$);
- P = total number of periods ($p = 1, 2, \dots, P$);
- N = total number of locations (workspaces and depots);
- L = location set, $|L| = N$ ($L = 1, 2, \dots, N$);
- W = workspace set, where $W \in L$ ($w \in W$);
- S = depot set, onde $S \in L$ ($s \in S$) e $W \in S = L$;
- R_j = resource set for activity j to be accomplished;
- I_p = idle resource set during period p ;
- A_p = activity set during period p .

Start parameters:

- d_{kl} = distance between locations k and l ;
- C_s = capacity of depot s ;
- NR_j = number of resources associated with activity j .

Formulation:

Minimize:

$$\sum_{t=1}^T \sum_{k=1}^N \sum_{l=1}^N \sum_{p=1}^{P-1} d_{kl} x_{ptk} x_{p+1tl} \quad (1)$$

Sujeito a:

$$\sum_{\forall s \in S} x_{pts} = 1, \forall p, \forall t \in I_p, \quad (2)$$

$$\sum_{\forall s \in S} x_{pts} \leq C_s, \forall s \in S, \forall p, \quad (3)$$

$$\sum_{\forall w \in W} y_{jw} = 1, \forall j, \quad (4)$$

$$\sum_{\forall j \in A_p} y_{jw} \leq 1, \forall w \in W, \forall p, \quad (5)$$

$$\sum_{t \in R_j} x_{ptk} = NR_j y_{jw}, \forall p, \forall j \in A_p, \forall w \in W, \quad (6)$$

$$x_{ptk} = 0 \text{ ou } 1, \forall p, \forall t, \forall k \in L, \quad (7)$$

$$x_{jw} = 0 \text{ ou } 1, \forall j, \forall w \in W. \quad (8)$$

The objective function (1) minimizes the resources' transportation cost between workspaces and depots during the periods when the activities will be carried out. Constraints (2) guarantee that every idle resource will be allocated to a depot during a period and

constraints (3) guarantee that that depot's capacity will be respected. Constraints (4) reserve a workspace for each activity and constraints (5) guarantee that an activity will occupy one workspace. Equation (6) associate the resources used by a given activity to the same workspace to which it was allocated. Constraints (7) and (8) indicate that the problem variables are binary.

4. Construction heuristic for the DSAP

The construction heuristic presented in this section can be divided into two parts. The first is an adaptation of the constructive randomized heuristic (CRH) proposed in [10]. This heuristic, hereafter denoted as ACRH, aims to generate an initial solution for the DSAP, corresponding to an activity association. The second part, called *randomized storage policy* (RSP), was proposed in [8] and deals with the association of idle resources with depots.

ACRH is based on cluster generation: the number of clusters is the number of workspaces and each one contains a set of activities. Two activities can belong to the same cluster (workspace) if their periods do not coincide.

The first pass of ACRH is used to calculate and store the three less similar pairs of activities: among them, one is randomly chosen to compose the first two clusters. The similarity measure used between two activities i and j is the number of resources they share.

Since the activities are analyzed according to the value of their labels, in order to avoid having only smaller label activities be chosen, the algorithm implements a probabilistic function which allows an activity i to be replaced by an activity j with a probability of $p\%$. After each replacement, this probability decreases by $\mu\%$.

To build the following clusters, a similarity measure is calculated for each activity not yet allocated with respect to each cluster already built. Here, the measure will be the number of common resources between the activity and each cluster. The least similar activity is allocated to the cluster. If a tie occurs, the activity with the smaller label is chosen.

Finally the ARCH begins to make allocations of the remaining activities to the clusters. For each activity, a similarity measure is calculated with respect to each cluster and the activity is allocated to the most similar cluster, provided that there is no period conflict with some activity already in that cluster.

The RSP heuristic allocates the idle resources to those depots that are nearest the workspace to which an activity that will use them has been allocated.

5. A refinement heuristic for the DSAP

The refinement heuristic presented in this section applies the following activity movements defined in [8]. Movement 1 is the exchange of workspaces of two or more activities during one or more periods. Movement 2 is the removal of an activity from a workspace and its allocation to another available workspace during one or more periods. Movement 3 is the association of Movements 1 and 2.

Starting with a solution built by ARCH, the following steps are executed until a given number of iterations without improvement is attained:

- at the beginning, the algorithm evaluates every possible activity movement that can be made in the current solution;
- for each one, the RSP heuristic is executed to reallocate the idle resources in accordance with the new activity matrix;
- the best movement is executed and the new solution becomes the current one.

The proposed refinement heuristic has the possibility of being stuck to local optima. A list of the most recent movements is compiled to avoid their repetition. This strategy is based on the tabu search concept [1]; [5]; [2]; [3].

The movement list implemented in this work can be considered dynamic since its size varies between given upper and lower limits, changing at each γ iteration without improvement. This limitation was adopted because an excessively long list could bring the search to a premature end. On the other hand, too short a list could permit the search to repeat recent movements which would allow for inefficient use of space.

6. Computational results

In order to test the efficiency of our proposal, two test batteries were executed, both using the 96 instances (P01-P96) proposed in [8], kindly furnished by the authors. The heuristics were implemented on C language using Version 4.0.3 of the GNU gcc compiler. All experiments were run on Dual Intel Pentium 4 3192 Mhz computers with 1010 Mb RAM, under Ubuntu Linux 6.0.4.

The purpose of the first test battery was to test the efficiency of the proposed construction algorithm, since a good performance of a refinement algorithm is directly dependent on the initial solution it receives. The results were compared with those from the RC (*Randomized Clustering*) heuristic also described in [8]. This comparison is justified by the similar nature of RC and ARCH, that is, both are based on algorithms for manufacturing cell problems.

Table 5 presents the sum of costs obtained by each construction algorithm for instances with 6, 12, 20 and 30 locations. The graph in Figure 2 illustrates the same results.

Table 5. Comparison between costs obtained by ARCH and RC algorithms

Problems	ACRH	RC
6-locations	1,010	1,057
12-locations	2,548	1,931
20-locations	3,482	3,752
32-locations	5,734	7,238
Total	11,764	13,978

From these results, we can see that ARCH was able to find a total of 1204 fewer distance units than RC did. Only for 12-location

cases was the ARCH sum of costs greater than that of the algorithm from the literature. We can see that ARCH performs satisfactorily with the larger instances.

The literature reports the optimal cost for 25 of the 96 instances (P01 to P24 and P27). The ARCH heuristic is able to find the optimum value for 5 of these instances, surpassing the clustering algorithm from the literature, which obtained the optimum for only 4 of these instances.

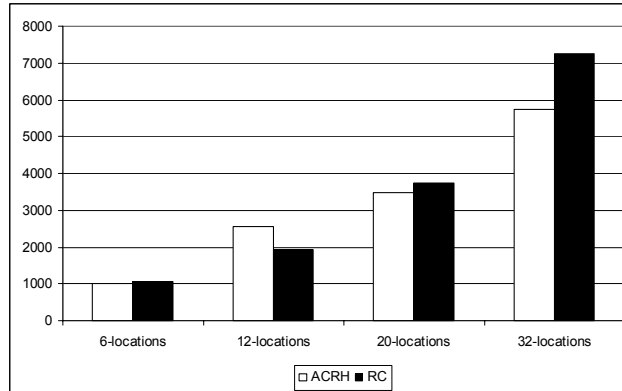


Figure 1. AHRC versus RC results

The next test battery aimed to evaluate ARCH results when combined with the proposed refinement heuristic, which thus constitute the PA (Proposed Algorithm). The configurations used by the PA algorithm are described in Figure 2.

Lower bound of the list of recent moves	$0.7 \times n$
Upper bound of the list of recent moves	$1.1 \times n$
Number of iterations without improvement	100
Parameter ρ	25%
Parameter μ	1%
Parameter γ	10

Figure 2. Parameters used in tests, n =number of activities

For comparison, we used the results obtained with the RC heuristic combined with the tabu search (TS) algorithm from the same authors.

Table 6 indicates a total cost reduction of 242 distance units, that is, about 2.5 units per instance. This reduction is obtained for the larger instances of 20 and 32 locations (Figure 3).

Table 6. Comparison between costs obtained by PA and RC+TS algorithms

Problems	PA	RC+TS
6-locations	977	969
12-locations	2,203	1,898
20-locations	2,918	2,989
32-locations	4,593	5,247
Total	9,987	10,134

Table 6 indicates a total cost reduction of 242 distance units, that is, about 2.5 units per instance. This reduction is obtained for the larger instances of 20 and 32 locations. We can thus ascertain that PA obtained better results for 35 among the 96 instances, when compared with RC+TS.

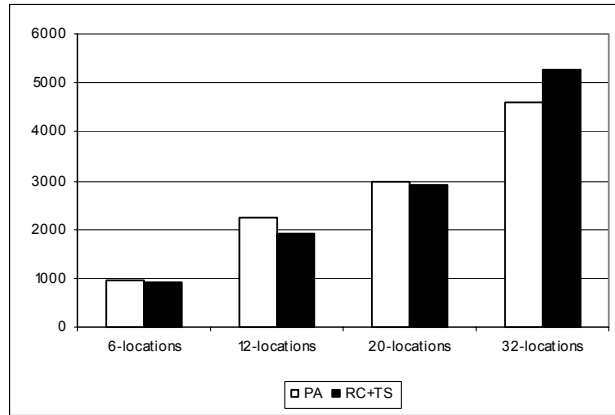


Figure 3. Results of PA compared with RC+TS

Figures 4 to 7 show the results of each algorithm with each test problem, for 6, 12, 20 and 32 locations, respectively. Through them, we can see that PA obtained equal or better results for most of the instances. Only for 12-location problems do the literature algorithms show better results. These results show that, the better the initial solution, the better the results obtained by the refining algorithm: we recall that Table 5 showed that, for the 12-location series, the ARCH algorithm was not able to surpass the algorithm from the literature, on average.

There was no comparison between computing time for the heuristics, since different machines were used in the tests. Nevertheless, it was found that for every instance the time used by PA was less than 140 seconds.

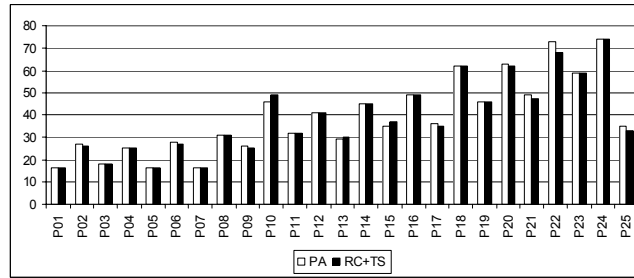


Figure 4. Results for 6-location problems

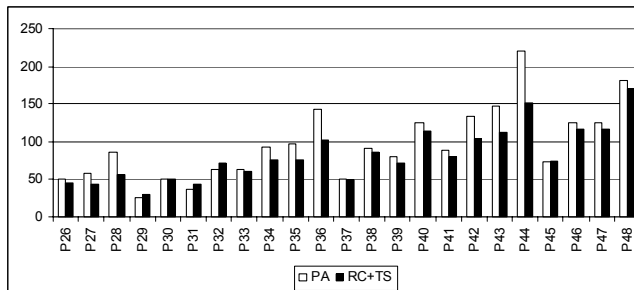


Figure 5. Results for 12-location problems

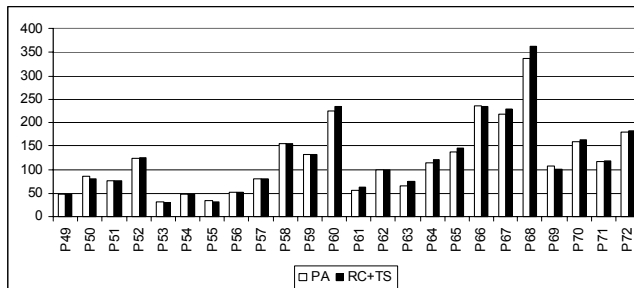


Figure 6. Results for 20-location problems

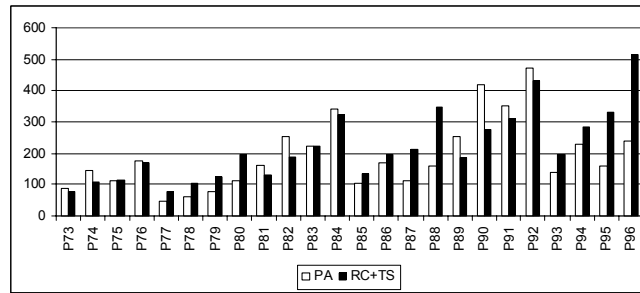


Figure 7. Results for 32-location problems

7. Conclusions

The first objective of this work was to adapt to DSAP definition a heuristic already proven to be efficient for manufacturing cell problems. We expected that a good construction heuristic could lead to better global solutions.

To verify this, we implemented the proposed construction and refinement algorithms and we used a set of instances from the literature. It was verified through extensive testing that the proposed heuristics are competitive with respect to the problem, since a number of better results were obtained in satisfactorily low computing time.

Acknowledgement

This research has been partially supported by the CT-Info/MCT/CNPq Universal 15/2007. The first author is grateful to CAPES and FAPEMIG for financial support.

References

- [1] Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research* 5: 553-549.
- [2] Glover, F. 1990. Tabu Search: a Tutorial. *Interfaces* 20: 74-94.
- [3] Glover, F. e Laguna, M. 1998. *Tabu Search*. Kluwer Academic Publishers, The Netherlands.
- [4] Glover, F. e Laguna, M. (2000). Fundamentals of Scatter Search and Path Relinking, *Control and Cybernetics* 29: 653-684.
- [5] Hansen, P. 1986. The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. In: *Proceedings of Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.
- [6] Loiola, E.M.; Abreu, N.M.M.; Netto, P.O.B., Hahn, P.M. e Querido, T. 2007. A Survey for the Quadratic assignment problem, *European Journal of Operational Research* 176: 657-690.
- [7] Luger, G.F. 2004. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley.
- [8] Mckendall, A.R. & Jaramillo, J.R. 2006. A Tabu Search Heuristic for the Dynamic Space Allocation Problem. *Computers & Operations Research* 33: 768-789.
- [9] Mckendall, A.R.; Noble, J.S. & Klein, C.M. 2006. Simulated Annealing Heuristics for Maninging Resources During Planned Outages at Eletric Power Plants. *Computers & Operations Research* 32: 107-125.
- [10] Trindade, A.R. e Ochi, L.S. 2006. Um Algoritmo Evolutivo Híbrido para a Formação de Células de Manufatura em Sistemas de Produção. *Pesquisa Operacional* 26: 255-294 (*Int. Abstracts in Operations Research; e Statistical Theory and Methods Abstracts*).

GRASP with hybrid heuristic-subproblem optimization for the multi-level capacitated minimum spanning tree problem

Alexandre X. Martins · Mauricio C. de Souza ·
Marcone J.F. Souza · Túlio A.M. Toffolo

Received: 16 February 2007 / Revised: 6 February 2008 / Accepted: 24 April 2008 /
Published online: 8 May 2008
© Springer Science+Business Media, LLC 2008

Abstract We propose a GRASP using an hybrid heuristic-subproblem optimization approach for the Multi-Level Capacitated Minimum Spanning Tree (MLCMST) problem. The motivation behind such approach is that to evaluate moves rearranging the configuration of a subset of nodes may require to solve a smaller-sized MLCMST instance. We thus use heuristic rules to define, in both the construction and the local search phases, subproblems which are in turn solved exactly by employing an integer programming model. We report numerical results obtained on benchmark instances from the literature, showing the approach to be competitive in terms of solution quality. The proposed GRASP have in fact improved the best known upper bounds for almost all of the considered instances.

Keywords Multi-level capacitated minimum spanning tree problem · Network design · GRASP · Subproblem optimization · Local search

A.X. Martins
Departamento de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto, João Monlevade,
MG, Brazil
e-mail: xmartins@decea.ufop.br

M.C. de Souza (✉)
Departamento de Engenharia de Produção, Universidade Federal de Minas Gerais,
Caixa Postal 4006, cep: 31250-970, Belo Horizonte, MG, Brazil
e-mail: mauricio.souza@pq.cnpq.br

M.J.F. Souza
Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil
e-mail: marcone@iceb.ufop.br

T.A.M. Toffolo
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte,
MG, Brazil
e-mail: tuliotoffolo@yahoo.com.br

1 Introduction

Capacitated trees arise in the context of centralized network design problems when constraints limit the traffic transmission supported by a link. These problems consist of finding a minimal cost spanning tree rooted at a central node such that the amount of traffic to be transferred from the central to the other nodes is bounded by edge capacities. Applications can be found, for instance, in communication networks, e.g., Esau and Williams (1966), Gavish (1982, 1991), Rothfarb and Goldstein (1971).

Esau and Williams (1966) were the first, to our knowledge, to introduce the Capacitated Minimum Spanning Tree (CMST) problem. Let $G = (V, E)$ be a connected undirected graph, where V is the set of nodes and E is the set of edges. Non-negative costs c_{ij} and weights b_i are associated respectively to each edge $(i, j) \in E$ and to each node $i \in V$. Given an integer Q and a central node $r \in V$, the CMST problem consists of finding a minimum spanning tree T of G such that the sum of the node weights in each connected component of the subgraph induced in T by $V - \{r\}$ is less than or equal to Q . The CMST problem is NP-Hard for $3 \leq Q \leq \frac{|V|}{2}$, Papadimitriou (1978). Both exact and heuristic methods have been developed for the CMST problem, see for instance Ahuja et al. (2001), Amberg et al. (1996), Gavish (1982), Gouveia (1995), Gouveia and Martins (2000, 2005), Hall (1996), Martins (2007), Sharaiha et al. (1997), Souza et al. (2003) and Uchoa et al. (2008).

In the CMST problem, a capacity Q is installed on every edge (i, j) composing a feasible tree T , paying thus its full cost c_{ij} no matter the traffic to be transferred between nodes i and j . A natural extension to the CMST problem is to consider that between each pair of nodes a feasible set of capacities with different costs is available. This means that instead of installing a capacity Q on every edge of T , choice can be made among different values of capacities and respective costs. This problem has been recently treated by Gamvros et al. (2003, 2006) as the Multi-Level Capacitated Minimum Spanning Tree (MLCMST) problem.

Other problems having characteristics in common with the MLCMST problem have been treated in the literature. Rothfarb and Goldstein (1971) have treated in the early seventies a similar problem, denoted one-terminal telpak problem. That work concerns a network design problem in which requirements from locations to a common facility must be satisfied minimizing a nonlinear neither convex nor concave but piecewise linear cost function. Network design problems related to the MLCMST can be found in telecommunications, see Berger et al. (2000) and Gavish (1991), and in gas and oil pipeline systems, see Brimberg et al. (2003).

In this paper we propose a GRASP for the MLCMST problem that exploits an hybrid heuristic-subproblem optimization scheme. Let us consider a collection $\Gamma = \{S_i\}$ of nonempty sets forming a partition of $V - \{r\}$ such that $\sum_{u \in S_i} b_u \leq Q$ for each S_i belonging to Γ , and let G_i be the subgraph of G induced by $S_i \cup \{r\}$. The motivation to employ subproblem optimization is that while in the CMST problem an instance defined by G_i reduces to the MST problem, this is not the case for the MLCMST problem. Indeed, for the MLCMST problem, to find a minimum cost capacitated spanning tree of a subgraph induced in G by $S_i \cup \{r\}$ may be itself a MLCMST instance. Gamvros et al. (2002, 2006), in their genetic algorithm, exploited this fact by partitioning $V - \{r\}$ and then trying to locally optimize each subproblem. We exploit this same search structure, but in our GRASP, heuristics are used to construct

and rearrange a partition of $V - \{r\}$ generating smaller-sized MLCMST subproblems which are independently solved to optimality.

The paper is organized as follows. In the next section we define the MLCMST problem and present recent advances and an integer programming formulation as well. In Sects. 3 and 4, respectively, we describe the proposed GRASP and report computational experiments on benchmark instances. We make concluding remarks in the last section.

2 Problem definition

The MLCMST problem can be stated as follows. Let $G = (V, E)$ be a connected undirected graph, where V denotes the set of nodes and E denotes the set of edges. Let us consider L different capacities of value z^l , $l = 1, \dots, L$, such that $0 < z^1 < z^2 < \dots < z^L$, which are available to be installed on each edge $(i, j) \in E$ with a cost c_{ij}^l . We consider that only one capacity can be installed on an edge.

Given a spanning tree $T = (V, \hat{E})$ of G , $z_{(i,j)}^{\hat{l}}$ denotes the capacity installed on edge $(i, j) \in \hat{E}$. The cost of T is given by $\sum_{(i,j) \in \hat{E}} c_{ij}^{\hat{l}}$. A non-negative weight b_i is associated to each node $i \in V$. The tree T being rooted at a central node $r \in V$, the predecessor $p(i)$ of a node $i \in V - \{r\}$ is the first node in the path from i to r in T . We denote by T_i the connected component containing node i in the forest obtained by removing edge $(p(i), i)$ of T . The MLCMST problem consists of finding a minimum cost spanning tree T of G such that the sum of the node weights in each T_i , $i \in V - \{r\}$, is less than or equal to the capacity $z_{(p(i),i)}^{\hat{l}}$ installed on edge $(p(i), i)$. This problem clearly reduces to the CMST if only one capacity of value Q is available.

2.1 Recent advances

The MLCMST have been treated considering a cost structure that reflects the economies of scale effect, i.e., $\frac{c_{ij}^l}{z^l} > \frac{c_{ij}^{l+1}}{z^{l+1}}$, for $l = 1, \dots, L - 1$, and for all $(i, j) \in E$. Gamvros et al. (2002) proposed a mixed-integer programming model for the MLCMST problem in which each node $i \in V - \{r\}$ is the source of a commodity with unitary demand, and the central node is the destination of all the $|V| - 1$ commodities. They also developed a genetic algorithm that split the MLCMST into two separate subproblems: a grouping problem and a network design problem. The grouping problem consists of assigning nodes to components to be connected to the central node such that the component's node weights sum does not exceed the highest capacity z^L . The network design problem is in the worst-case a MLCMST instance in the subgraph induced by the nodes of a group together with the central node. As proposed by Falkenauer (1996), the chromosome represents a solution by breaking it up into two parts: a group part and an item part which are associated with the grouping and network design subproblems respectively. A crossover operator similar to that of Falkenauer (1996) and a mutation operator are applied.

Martins et al. (2005) conducted computational experiments on solving with a commercial package the MLMST problem. In that work, three integer programming models for MLMST were compared on instances adapted from benchmark ones introduced by Gouveia (1995) for the CMST. Two models are mixed integer flow formulations: the single commodity formulation due to Gavish (1982), and the multicommodity formulation proposed by Gamvros et al. (2002). The remaining model is a pure integer programming formulation: the $2n$ constraint formulation due to Gouveia (1995), i.e., $|V| = n$. The models due to Gavish (1982) and to Gouveia (1995) were originally proposed to the CMST problem, and they are straightforwardly adapted to the MLMST. The $2n$ constraint model, in the CMST context, uses binary variables that limit to l , $l = 1, \dots, Q - b_i$ (b_i is assumed to be 0), the flow on edge (i, j) . This leads to what we call a capacity-indexed model for the MLMST problem. The capacity-indexed model has shown to be the most effective for the purpose of solving MLMST instances to optimality, though the multicommodity flow model proposed by Gamvros et al. (2002) seems to provide tighter linear relaxation lower bounds. The capacity-indexed model was the only one able to solve instances up to 30 nodes in less than one hour (quite often in few seconds) and obtaining less than 10% of optimality gap for instances up to 40 nodes, running LINGO version 7.0 on a Pentium IV 1.8 GHz with 256 MB of RAM memory (Martins et al. 2005).

Gamvros et al. (2006) performed recently a comprehensive study on the MLMST problem. The authors compared in terms of lower bounds two flow-based mixed integer programming formulations: a strengthened formulation of the single commodity flow model by Gavish (1982), denoted by ESCF, and a strengthened formulation of the multicommodity flow model (Gamvros et al. 2002), denoted by MCF. Computational experiments with medium-sized instances (up to 30 nodes) showed that for bounding purposes ESCF is most useful than MCF. The linear relaxation of ESCF is rapidly solved, while the relaxation of MCF generates more difficult linear programs providing only slightly better bounds.

Gamvros et al. (2006) also developed heuristic procedures. A saving construction heuristic starts with a star network in which each node $i \in V - \{r\}$ is connected to r with capacity $z_{(i,r)}^1$. Then, based upon savings in joining nodes in a same component, some edge capacities are upgraded from z^1 to z^L . When no more savings with capacity z^L are possible, the procedure searches for savings in upgrading from z^1 to z^{L-1} , and so on. As observed in the CMST problem with the classical Esau and Williams heuristic (Esau and Williams 1966), the proposed saving heuristic obtain good upper bounds for the MLMST in reduced computational times. Gamvros et al. (2006) exploited the node-based, multi-exchange neighborhood structures developed by Ahuja et al. (2001) in two local search procedures for MLMST problem. The cyclic and path exchanges define neighborhoods of exponential size which are explored by constructing an improvement graph. In the first variant, local search is directly applied to a solution obtained with the saving construction heuristic. In the second variant, called Randomized Start Local Search, different starting solutions are obtained by running the saving construction heuristic on perturbed cost versions of the original graph. Local search is then independently applied to each starting solution. The authors improved the genetic algorithm proposed in Gamvros et al. (2002) by applying a mutation operator based upon the cyclic and path exchange neighborhoods. Computational experiments were conducted on graphs with up to 150 nodes.

Three capacities with costs proportional to the Euclidean distances by factors reflecting economies of scale were considered. Numerical results obtained with the saving heuristic, the two local search procedures, and the genetic algorithm showed average gaps from lower bounds ranging from 6.09% to 9.91%, see Gamvros et al. (2006).

2.2 Capacity-indexed model

We now present the capacity-indexed formulation for the MLCMST problem. This model was originally proposed by Gouveia (1995), and has recently been used with success by Uchoa et al. (2008) in a branch-and-cut-and-price algorithm for the CMST. Based on our previous computational experience reported in Martins et al. (2005), we have chosen this model to solve to optimality the subproblems generated by the proposed GRASP.

The capacity-indexed model requires the following assumptions: (i) $z^1 = 1$ and (ii) capacities increase from 1 to z^L by unitary increments. This however may not be found in ordinary practice. In case conditions (i) and (ii) do not hold, we create artificial capacities $\bar{z}$. The number of different capacities available is set to $P = z^L$, and then $\bar{z}^1 = 1, \bar{z}^2 = 2, \dots, \bar{z}^P = z^L$. Given an edge (i, j) , the cost $\bar{c}_{ij}^p$ associated to each artificial capacity $\bar{z}^p, p = 1, \dots, P$, is then set to

$$\bar{c}_{ij}^p = \begin{cases} c_{ij}^1, & p = 1, \dots, z^1, \\ c_{ij}^l, & p = z^{l-1} + 1, \dots, z^l, l = 2, \dots, L. \end{cases}$$

Although the MLCMST problem is defined on an undirected graph, we describe a model that works on a directed graph generated by replacing an edge (i, j) in the original graph by two arcs (i, j) and (j, i) keeping the same cost structure, e.g., (Gamvros et al. 2006; Martins et al. 2005; Uchoa et al. 2008). A binary variable $x_{ij}^l, i \in V, j \in V - \{r\}$, and $l = 1, \dots, L$ is defined such that

$$x_{ij}^l = \begin{cases} 1, & \text{if capacity } l \text{ is installed on arc } (i, j), \\ 0, & \text{otherwise.} \end{cases}$$

The capacity-indexed formulation is then written as:

$$\min \sum_{p=1}^P \sum_{i \in V} \sum_{j \in V - \{r\}} \bar{c}_{ij}^p x_{ij}^p \quad (1)$$

$$\text{s.t.} \quad \sum_{p=1}^P \sum_{i \in V} x_{ij}^p = 1 \quad \forall j \in V - \{r\}, \quad (2)$$

$$\sum_{p=1}^P \sum_{i \in V} \bar{z}_{ij}^p x_{ij}^p - \sum_{p=1}^P \sum_{i \in V - \{r\}} \bar{z}_{ji}^p x_{ji}^p = b_i \quad \forall j \in V - \{r\}, \quad (3)$$

$$x_{ij}^p \in \{0, 1\} \quad \forall i \in V, \forall j \in V - \{r\}, p = 1, \dots, P. \quad (4)$$

Constraints Eq. 2 and Eq. 3 are respectively the in-degree constraint for an arborescence rooted at r and the capacity-balance constraint. These constraints together guarantee an arborescence feasible to the MLCMST problem, since Eqs. 2 and 3 prevent both cycles and sub-arborescences rooted at a node $i \in V - \{r\}$ with node weights exceeding arc $(p(i), i)$ capacity.

3 GRASP with hybrid heuristic-subproblem optimization

The proposed GRASP employs an hybrid heuristic-subproblem optimization scheme. Heuristic rules are used both to do a partition of $V - \{r\}$ and to decide whether to call an exact method within the construction and the local search phases. GRASP is a multi-start metaheuristic proposed by Feo and Resende (1995) which has been widely used to obtain good quality solutions for many combinatorial problems. The reader is referred to Festa and Resende (2001, 2004) and Resende and Ribeiro (2003) for in-depth surveys covering GRASP from basic scheme to recent enhancements, implementation strategies and successful applications. A GRASP iteration consists basically of two subsequent phases: construction phase and local search phase. The construction phase builds a feasible solution. The local search starts off with the solution built in the former phase and tries, by investigating neighborhoods, to achieve improvements until a local minima. The procedure returns the best solution found after Max_It iterations.

3.1 Construction phase

The construction phase builds in two steps a feasible solution to the MLCMST problem. We first use a greedy randomized heuristic to do a partition of $V - \{r\}$ in R_k , $k = 1, \dots, K$ subsets. This step has an input parameter $w \geq z^L$ that limits the cardinality of each subset in the partition, thus $K = \lceil \frac{|V - \{r\}|}{w} \rceil$ in the unitary node weights case. We build one subset of the partition at a time. Let S denote the candidate nodes to be inserted in subset R_k being built, initially we set $S = V - \{r\}$ and $k = 1$. The procedure starts an iteration by removing a node i at random from S and inserting it in R_k . While $|R_k| < w$ and $S \neq \emptyset$ the procedure moves one node after the other from S to R_k . Thus, all nodes remaining in S are candidates to be inserted in R_k . We create then a restricted candidate list (RCL) formed by the best elements given by a greedy evaluation function. The RCL is formed by those nodes whose incorporation to R_k results in the smallest incremental cost according to Prim's algorithm to compute a MST, see for instance (Ahuja et al. 1993). Let d_j , for a node $j \in S$, be a label defined as $d_j = \min\{c_{ij}^1 : i \in R_k\}$. We then set $d_{\min}$ and $d_{\max}$ respectively to the minimum and maximum values of d_j , $j \in S$. Given a parameter $\alpha \in [0, 1]$, RCL is defined as

$$\text{RCL} = \{j \in S : d_j \leq d_{\min} + \alpha(d_{\max} - d_{\min})\}.$$

The element to be moved from S to R_k is randomly selected from those in the RCL, and labels d_j are updated for the nodes remaining in S following Prim's algorithmic logic (Ahuja et al. 1993). When w nodes are inserted in R_k , we increment k and proceed until a partition of $V - \{r\}$ is done.

At this point, there are K subproblems consisting each of an independent MLCMST instance on the subgraph induced in G by $R_k \cup \{r\}$, $k = 1, \dots, K$. In the second step, we solve independently each of the K subproblems to optimality, and we have thus a feasible solution to the original MLCMST instance. For this purpose we employ the capacity-indexed model, cf., Sect. 2.2, by making calls to an optimization package. Note that by solving exactly the subproblems, nodes grouped in one subset R_k of the partition can actually form two or more components connected to r .

3.2 Local search phase

The local search phase tries to improve a feasible solution by re-arranging nodes of different components connected to r . Let us consider, in the sequel of this section, a spanning tree $T = (V, \hat{E})$ feasible to the MLCMST problem. Let us also designate a connected component of the forest obtained by the elimination of r and its incident edges from T as a component of T . A move is then defined by combining two or more components. The value of applying a move to T in order to generate a neighbor $T' = (V, \hat{E}')$ is given by $\Delta = \sum_{(i,j) \in \hat{E}'} \hat{c}_{ij}' - \sum_{(i,j) \in \hat{E}} \hat{c}_{ij}$, where $\hat{c}_{ij}$ denotes the cost of capacity $\hat{z}_{(i,j)}$ installed on edge $(i, j) \in \hat{E}$, cf., Sect. 2. The value of Δ is computed by obtaining the best way to connect to r the nodes of the components involved. As mentioned before, evaluate this kind of move means solving a smaller-sized MLCMST instance in the worst-case. Thus, we limit the size of the subproblem induced in G by the components defining a move. We make within the local search calls to an optimization package to solve exactly each MLCMST instance modeled with the capacity-indexed formulation. We propose thus heuristic rules to form the subproblems.

The rationale of the heuristic is to identify a local gain in connecting two nodes that are in different components of T . Given a node i , let us define by V_i the set of nodes in the component containing i , and let us also denote by $c(T_i)$ the cost of the subgraph induced in T by $V_i \cup \{r\}$ with the respectively installed capacities. Figure 1 shows in details the local search procedure. We consider the set S of non-leaf nodes in T as candidates to be “reference” nodes to form a subproblem. At each iteration of the local search, a node i is chosen at random from S . The procedure then starts to build a subset of nodes P which may induce a subproblem in G . Initially P is set to $V_i \cup \{r\}$, and the gain γ is set to $c(T_i)$. The set $\bar{S}$ contains the nodes that connect the other components to r . The parameter h , which limits the cardinality of P , is chosen at random from the interval $[\underline{h}, \bar{h}]$, where $\underline{h}$ and $\bar{h}$ are positive integers. We try then to add the nodes of one or more components to P so as to build a subproblem. A node j is chosen at random from $\bar{S}$, and the procedure looks for a node u belonging to V_j that could be connected to i with the capacity $\hat{l}$ installed on edge $(p(u), u)$ at a smaller cost. The component V_j is included in the subproblem being built if both there exists such a node and the cardinality of $V_j \cup P$ does not exceed h . In this case the gain is increased by adding $c(T_j)$ to γ . Node j is removed from $\bar{S}$ and the procedure continues trying to enlarge the subproblem until either the cardinality of P is h or $\bar{S}$ is empty.

At this point, a subproblem has been built if P contains also nodes other than the ones in $V_i \cup \{r\}$. We then solve to optimality the subproblem induced in G by P . That is, we try to re-arrange in an optimal manner the components of T selected to be included in P . Let us suppose that the optimal solution of the subproblem has a value of ϕ , note that ϕ cannot be greater than γ . If ϕ is smaller than γ , an improving move with value $\Delta = \phi - \gamma$ has been found, and the current solution T is updated. The set S of candidates to be “reference” nodes is also reconfigured. We include in (resp. remove from) S the non-leaf (resp. leaf) nodes of the optimal solution of the subproblem that are not (reps. are) already in it. The choice of a node i from S at the beginning of an iteration may not lead to an improving move for two reasons: the move value Δ is zero or none of the components has been added to P . In the first case the optimal solution of the subproblem has the same configuration the components involved already had in T , and we remove from S all the nodes of component V_i . In the second case the exact method has not been called, and we remove from S only the node i . The procedure iterates while S is not empty.

4 Computational results

In order to measure the potential of the proposed GRASP in finding good quality solutions, we conducted computational experiments on benchmark instances for the MLCMST problem. We report in this section numerical results on instances introduced in the literature by Gamvros et al. (2006). The authors introduced small instances with 20 and 30 nodes plus the central node, and larger ones with 50, 100 and 150 nodes plus the central node. These are unitary demand instances in which the nodes, except the central, are randomly distributed in a 40×40 square grid. There are three problem types for each size, according to the location of the central node: in the center, at the edge, and randomly located. For each problem type and size, 50 instances were generated. In all instances, three different capacities are available and their values are the same for every edge: $z^1 = 1$, $z^2 = 3$, and $z^3 = 10$. For each edge (i, j) , the cost c_{ij}^1 of the first facility is equal to Euclidean distance between their extremities (not rounded), and the cost of the second and third facilities are respectively $c_{ij}^2 = 2c_{ij}^1$ and $c_{ij}^3 = 3c_{ij}^1$.

Regarding the larger instances, Gamvros et al. (2006) have reported, for each combination number of nodes—location of the central node, aggregated percentage gaps (average, min and max) with respect to the lower bounds obtained by the LP relaxation of ESCF. For the 50 and 100 nodes instances, the authors have reported results for all the methods developed by them: the construction heuristic, the two local search procedures, and the genetic algorithm (Gamvros et al. 2006), cf., Sect. 2.1. The genetic algorithm seems to perform better, since it obtained the smallest average gaps for all the six problem type size pairs of the instances of 50 and 100 nodes. For the 150 nodes instances, Gamvros et al. (2006) have reported results for the construction heuristic and the first variant of the local search, cf., Sect. 2.1.

We considered in this study the three problem type instances with 50 nodes, and the central node located in the center instances with 100 and 150 nodes, resulting in a total of 250 instances. This option is due to the large computational times needed

procedure Local_Search

```

1  Set  $S$  as the set of non-leaf nodes in  $T$ ;
2  while ( $S \neq \emptyset$ ) do
3      Select at random a node  $i$  from  $S$ .
4       $\gamma \leftarrow c(T_i)$ 
5       $P \leftarrow V_i \cup \{r\}$ 
6       $\bar{S} \leftarrow \{j \in V - P \mid (j, r) \in \hat{E}\}$ 
7      Chose at random  $h \in [\underline{h}, \bar{h}]$ 
8      while ( $|P| < h$ ) and ( $\bar{S} \neq \emptyset$ ) do
9          Select at random a node  $j$  from  $\bar{S}$ .
          For each node  $u \in V_j$  let  $\hat{l}$  be the capacity installed on edge  $(p(u), u)$ 
10         if (there exists a node  $u \in V_j$  such that  $c_{iu}^{\hat{l}} < c_{p(u),u}^{\hat{l}}$ )
            and ( $|V_j| + |P| \leq h$ ) then
11              $P \leftarrow P \cup V_j$ 
12              $\gamma \leftarrow \gamma + c(T_j)$ 
13         end-if
14          $\bar{S} \leftarrow \bar{S} - \{j\}$ 
15     end-while
16     if ( $|P| > |V_i| + 1$ ) then
17         Solve to optimality the subproblem induced in  $G$  by  $P$ , and let  $\phi$ 
            be the value of its optimal solution.
18         if ( $\phi < \gamma$ ) then
19             Update the current solution  $T$ .
            Let  $I$  and  $\bar{I}$  be respectively the set of non-leaf and leaf nodes in
            the optimal solution of the subproblem.
20              $S \leftarrow (S - \bar{I}) \cup I$ ;
21         end-if
22     else
23          $S \leftarrow S - V_i$ ;
24     end-else
25 end-if
26 else
27      $S \leftarrow S - \{i\}$ ;
28 end-else
29 end-while
end-procedure

```

Fig. 1 Pseudo-code of the local search

to run our approach on the central node located at the edge and randomly. To run our GRASP approach on the 100 and 150 nodes instances with the central node located at the edge and randomly we shall develop extensions such as elimination tests.

The heuristic procedures were coded in C++, using compiler mingw32. CPLEX version 9.1 was used to solve all the subproblems to optimality. The 50 and 150 nodes instances were ran on a Pentium 4, 2.5 GHz, with 1 GB of RAM memory. The

Table 1 Results for the instances with 50 nodes, central node located in the center

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g(%)	UB	Time (s)	it	g (%)	Improved?
0	572.286	533.417	7.29	568.476	2073.375	1	6.57	yes
1	541.608	500.122	8.30	540.621	1811.016	2	8.10	yes
2	563.120	525.770	7.10	558.659	1717.453	1	6.26	yes
3	566.703	525.203	7.90	564.283	2087.125	2	7.44	yes
4	542.397	511.112	6.12	541.677	1182.516	2	5.98	yes
5	609.805	570.393	6.91	608.158	1270.328	1	6.62	yes
6	574.594	524.273	9.60	571.427	5721.984	2	8.99	yes
7	587.016	549.233	6.88	580.527	1163.516	1	5.70	yes
8	622.000	587.959	5.79	616.956	1075.734	8	4.93	yes
9	639.954	599.071	6.82	635.477	999.641	1	6.08	yes
10	560.428	528.558	6.03	557.311	1987.359	3	5.44	yes
11	597.520	563.720	6.00	592.567	1132.719	1	5.12	yes
12	633.220	593.333	6.72	630.495	1727.406	1	6.26	yes
13	545.470	501.707	8.72	541.065	1865.031	3	7.84	yes
14	570.354	533.994	6.81	565.960	2296.625	2	5.99	yes
15	561.199	531.141	5.66	560.875	1933.500	1	5.60	yes
16	581.043	541.948	7.21	577.772	1901.578	1	6.61	yes
17	576.984	539.608	6.93	570.136	1138.922	2	5.66	yes
18	612.073	566.167	8.11	605.112	1288.203	3	6.88	yes
19	609.880	572.305	6.57	608.206	1647.266	1	6.27	yes
20	564.933	515.615	9.57	561.301	2946.375	9	8.86	yes
21	595.280	555.307	7.20	593.014	2327.422	1	6.79	yes
22	570.357	533.678	6.87	565.352	2047.984	1	5.93	yes
23	575.103	527.074	9.11	573.494	2200.922	2	8.81	yes
24	629.089	584.053	7.71	623.627	1498.594	1	6.78	yes
25	559.847	539.019	3.86	554.543	2212.188	3	2.88	yes
26	575.675	532.307	8.15	570.778	899.422	4	7.23	yes
27	568.571	531.626	6.95	564.583	1076.750	1	6.20	yes
28	583.516	549.753	6.14	581.974	2180.656	1	5.86	yes
29	575.400	535.022	7.55	572.327	2128.484	1	6.97	yes
30	605.376	570.623	6.09	601.977	1419.344	1	5.49	yes
31	587.130	545.171	7.70	582.820	2425.828	12	6.91	yes
32	595.326	558.370	6.62	591.413	2155.375	1	5.92	yes
33	565.632	531.286	6.46	562.659	903.203	2	5.91	yes
34	600.291	553.032	8.55	596.988	1728.234	8	7.95	yes
35	576.653	532.293	8.33	574.187	2422.344	2	7.87	yes
36	582.251	535.656	8.70	579.590	1301.953	3	8.20	yes
37	529.314	498.100	6.27	525.880	1266.328	2	5.58	yes
38	598.707	558.825	7.14	595.329	3557.641	1	6.53	yes
39	549.914	509.313	7.97	549.045	5646.625	1	7.80	yes

Table 1 (Continued)

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g(%)	UB	Time (s)	it	g (%)	Improved?
40	559.654	521.103	7.40	554.145	1436.578	2	6.34	yes
41	582.251	544.404	6.95	578.868	2082.656	1	6.33	yes
42	588.473	541.122	8.75	581.187	1047.781	1	7.40	yes
43	585.324	547.437	6.92	581.138	1813.766	1	6.16	yes
44	585.415	558.671	4.79	584.498	1257.313	3	4.62	yes
45	585.161	543.586	7.65	581.792	1911.250	2	7.03	yes
46	581.248	546.615	6.34	575.617	1447.938	1	5.31	yes
47	626.617	574.898	9.00	622.094	3655.563	1	8.21	yes
48	586.974	542.572	8.18	579.057	1429.844	2	6.72	yes
49	559.967	532.136	5.23	559.609	1129.094	1	5.16	yes
Average	580.910	542.075	7.13	577.265	1891.729		6.45	

100 nodes instances were ran on a Athlon 64 3200+, 2 GHz, with 512 Mb of RAM memory. We ran GRASP for a number of `Max_It` equal to 15 iterations and 10 iterations for respectively the 50 nodes instances and the 100 and 150 nodes instances, since for the larger instances a larger number of subproblems are created increasing the time spent in each iteration. In the construction phase, α was chosen at random in the interval $[0.1, 0.4]$, and the parameter w , which limits the cardinality of each set in the partition, was set to 10. In the local search phase, the value of h was chosen at random from the interval $[16, 31]$.

We report in Tables 1 to 5 numerical results for each of the 250 instances considered. Tables 1, 2 and 3 correspond to the 50 nodes instances with central node located respectively in the center, at the edge and randomly. Tables 4 and 5 correspond respectively to the 100 and 150 nodes instances. In each table, the first column presents the instance identification. We then show the best known upper bound so far, the lower bound given by the linear relaxation of ESCF, and the gap with respect to the lower bound, i.e., $\frac{ub-lb}{lb}$ %. The best known upper bounds and the lower bounds were informed to us by Raghavan (2007). we report in the next columns results obtained by the proposed GRASP. We show the upper bound, the time in seconds to perform `Max_It` iterations, the iteration the best solution was found, and the gap with respect to the lower bound given by the linear relaxation of ESCF. In the last column we inform whether GRASP improved (case “yes”) or not (case “no”) the value of the best upper bound known so far. We display average results on the last line of each table.

Numerical results have shown the proposed hybrid strategy to be very efficient in finding good quality solutions for the MLCMST problem. GRASP using hybrid heuristic-subproblem optimization was able to improve the best known upper bounds for 247 out of 250 instances. The only instances for which the solution obtained by GRASP did not improved the best known value are the indexed by 13 and 41 with 50 nodes and central node located at the edge, and the one indexed by 34 with 100 nodes. Computational times are relatively large since the approach has to solve smaller-sized

Table 2 Results for the instances with 50 nodes, central node located at the edge

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
0	1117.320	1068.715	4.55	1108.674	4939.016	1	3.74	yes
1	1152.570	1106.826	4.13	1147.926	45926.094	1	3.71	yes
2	1011.130	969.158	4.33	1007.268	17089.469	1	3.93	yes
3	1089.590	1042.383	4.53	1084.108	8939.922	3	4.00	yes
4	1124.930	1080.000	4.16	1123.234	13108.781	10	4.00	yes
5	1097.488	1052.866	4.24	1096.208	12027.750	3	4.12	yes
6	1006.158	960.042	4.80	1002.304	19471.266	1	4.40	yes
7	1046.880	994.715	5.24	1038.709	11485.844	4	4.42	yes
8	1079.360	1036.033	4.18	1077.996	15925.734	11	4.05	yes
9	1123.570	1066.439	5.36	1118.622	17105.969	1	4.89	yes
10	1122.596	1065.632	5.35	1115.708	50987.828	1	4.70	yes
11	1093.840	1042.840	4.89	1093.425	54976.344	1	4.85	yes
12	1020.390	969.542	5.24	1018.228	2256.859	2	5.02	yes
13	1158.830	1120.440	3.43	1158.830	6380.422	3	3.43	no
14	1097.010	1047.044	4.77	1093.222	8038.391	3	4.41	yes
15	1088.038	1037.810	4.84	1081.795	13872.750	1	4.24	yes
16	1048.104	1007.032	4.08	1044.068	10578.953	2	3.68	yes
17	1065.522	1014.985	4.98	1058.846	17435.656	3	4.32	yes
18	1076.144	1025.321	4.96	1073.228	9574.797	5	4.67	yes
19	1048.350	998.147	5.03	1039.356	8240.891	9	4.13	yes
20	1043.620	1008.909	3.44	1042.514	5688.469	2	3.33	yes
21	1141.432	1093.608	4.37	1138.702	7677.422	1	4.12	yes
22	1111.630	1064.362	4.44	1107.963	3579.891	14	4.10	yes
23	1024.092	976.259	4.90	1020.009	6712.328	4	4.48	yes
24	1067.950	1029.884	3.70	1064.896	20781.875	1	3.40	yes
25	1034.200	985.120	4.98	1030.405	3755.047	1	4.60	yes
26	1126.770	1082.106	4.13	1121.626	3691.234	1	3.65	yes
27	945.774	902.381	4.81	941.540	5145.250	7	4.34	yes
28	1064.890	1016.481	4.76	1059.916	13199.000	1	4.27	yes
29	1130.400	1087.191	3.97	1128.082	6261.453	12	3.76	yes
30	1047.546	1006.175	4.11	1044.712	8275.469	2	3.83	yes
31	1111.790	1071.455	3.76	1109.926	3436.719	9	3.59	yes
32	1138.346	1090.908	4.35	1135.531	34444.250	4	4.09	yes
33	1072.710	1020.903	5.07	1068.535	4951.891	15	4.67	yes
34	1030.482	983.065	4.82	1024.628	4875.438	2	4.23	yes
35	1045.550	1001.190	4.43	1039.296	6268.766	6	3.81	yes
36	1067.570	1014.803	5.20	1057.711	29760.641	5	4.23	yes
37	1006.294	959.416	4.89	1001.467	6612.141	3	4.38	yes
38	1081.320	1030.120	4.97	1080.916	4030.422	4	4.93	yes
39	1041.820	993.939	4.82	1038.405	9547.359	7	4.47	yes

Table 2 (Continued)

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
40	1051.422	1001.262	5.01	1046.796	9388.578	10	4.55	yes
41	1051.030	1000.015	5.10	1051.428	6526.375	15	5.14	no
42	1058.582	1010.345	4.77	1052.056	6826.547	1	4.13	yes
43	1052.962	1005.704	4.70	1048.190	16615.703	3	4.22	yes
44	1018.830	974.004	4.60	1017.236	4810.297	8	4.44	yes
45	1064.190	1017.711	4.57	1062.960	6587.734	3	4.45	yes
46	1112.650	1059.193	5.05	1106.692	5963.859	10	4.48	yes
47	1119.244	1065.072	5.09	1113.767	10856.734	2	4.57	yes
48	1034.750	988.710	4.66	1028.043	9018.047	4	3.98	yes
49	1101.820	1051.821	4.75	1098.593	54296.188	9	4.45	yes
Average	1070.848	1023.538	4.60	1066.781	13141.269		4.21	

instances of an NP-Hard problem to evaluate a move. Indeed, more than 95% of the computational time effort on each instance is spent to solve exactly the subproblems. This inhibit running the method for more iterations, which could perhaps lead to further improvements in the best known upper bounds. Nevertheless, it is worth noting that, apart the instances with central node located at the edge, GRASP was able to improve the best upper bound for 187 out of 200 instances in less than one hour of CPU time. As already noted by other authors working with the CMST problem, the instances where the central node is located at the edge are in fact harder ones. This seems to also be true in the MLCMST context, as GRASP spent much more time in this type of instances because CPLEX had much more difficult in solving the subproblems to optimality.

5 Concluding remarks and extensions

The MLCMST problem possesses a structure that even simple moves defined as a re-configuration of a subset of nodes may lead to a smaller-sized instance of the problem itself. This fact makes MLCMST a quite challenging problem for local search based algorithms. In this paper we proposed so an hybrid heuristic-subproblem optimization strategy for the MLCMST problem. Heuristics are used to build subproblems, which are in turn solved to optimality by an exact method. We embedded this scheme in a GRASP framework. The proposed GRASP has shown to be quite competitive. Considering 250 benchmark instances, it was able to improve almost all the best known upper bounds obtained by the powerful methods developed by Gamvros et al. (2006).

The main direction of further research is related to heuristic rules to speed up the search. Elimination tests and move value estimations shall be developed to reduce the number of times the local search make calls to the exact method. These extensions, if successfully managed, can enable GRASP to perform much more iterations in acceptable computational time.

Table 3 Results for the instances with 50 nodes, central node randomly located

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
0	596.700	557.138	7.10	591.991	2763.495	1	6.26	yes
1	737.906	693.221	6.45	737.046	1698.13	10	6.32	yes
2	705.761	667.657	5.71	701.633	2855.06	6	5.09	yes
3	683.067	645.708	5.79	676.355	3432.59	1	4.75	yes
4	866.084	822.798	5.26	859.786	2034.48	4	4.50	yes
5	961.915	910.483	5.65	958.692	9046.31	2	5.29	yes
6	768.232	730.783	5.12	767.158	2464.86	1	4.98	yes
7	744.171	702.911	5.87	742.123	2763.63	3	5.58	yes
8	688.182	649.021	6.03	684.667	1161.94	1	5.49	yes
9	831.260	794.448	4.63	829.176	2782.61	1	4.37	yes
10	856.208	813.103	5.30	850.330	2763.50	4	4.58	yes
11	675.777	638.062	5.91	670.775	637.77	15	5.13	yes
12	718.659	672.276	6.90	715.212	4266.78	2	6.39	yes
13	795.780	752.459	5.76	790.461	1455.67	14	5.05	yes
14	791.095	745.260	6.15	786.882	3103.08	2	5.58	yes
15	874.707	828.337	5.60	872.357	1910.34	3	5.31	yes
16	704.393	663.789	6.12	702.672	969.58	2	5.86	yes
17	591.825	550.665	7.47	585.497	530.19	2	6.33	yes
18	692.904	660.101	4.97	690.706	1050.86	1	4.64	yes
19	816.991	768.450	6.32	811.374	3220.55	2	5.59	yes
20	881.706	842.081	4.71	879.342	4848.55	3	4.42	yes
21	801.536	764.734	4.81	797.794	1154.78	3	4.32	yes
22	691.108	657.499	5.11	688.662	876.70	1	4.74	yes
23	724.777	683.482	6.04	720.743	1240.91	1	5.45	yes
24	813.879	769.352	5.79	809.756	2016.31	3	5.25	yes
25	625.558	590.817	5.88	625.111	1005.27	5	5.80	yes
26	853.640	818.897	4.24	853.640	9867.516	2	4.24	yes
27	702.418	667.857	5.17	694.711	1212.938	1	4.02	yes
28	774.331	731.137	5.91	773.762	2037.203	3	5.83	yes
29	1065.210	1019.436	4.49	1062.234	6316.703	1	4.20	yes
30	717.690	679.062	5.69	712.625	2188.734	5	4.94	yes
31	736.650	689.199	6.88	729.420	1765.141	9	5.84	yes
32	977.977	933.054	4.81	973.967	4454.375	7	4.38	yes
33	976.020	929.517	5.00	970.959	2810.750	2	4.46	yes
34	796.694	754.480	5.60	792.202	2566.891	1	5.00	yes
35	552.085	522.575	5.65	551.251	977.688	2	5.49	yes
36	700.238	664.150	5.43	698.521	1407.859	1	5.18	yes
37	770.738	730.834	5.46	769.785	4535.516	4	5.33	yes
38	663.472	622.798	6.53	657.630	2455.500	1	5.59	yes
39	689.381	650.206	6.03	687.322	1205.813	5	5.71	yes

Table 3 (Continued)

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
40	650.031	611.928	6.23	649.141	1117.766	3	6.08	yes
41	790.728	747.924	5.72	788.533	1207.609	14	5.43	yes
42	536.435	505.561	6.11	535.245	613.656	1	5.87	yes
43	720.548	687.324	4.83	719.962	1258.297	1	4.75	yes
44	627.707	590.234	6.35	621.563	605.031	1	5.31	yes
45	800.358	758.279	5.55	800.283	3250.391	5	5.54	yes
46	550.632	514.568	7.01	547.649	705.734	4	6.43	yes
47	767.506	732.944	4.72	766.440	3746.797	8	4.57	yes
48	686.307	644.523	6.48	684.083	909.844	2	6.14	yes
49	629.679	598.730	5.17	628.967	729.688	1	5.05	yes
Average	743.394	703.636	5.68	740.224	2358.306		5.22	

Table 4 Results for the instances with 100 nodes, central node located in the center

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
0	1083.450	1024.191	5.79	1076.434	1801.177	7	5.10	yes
1	1117.510	1048.661	6.57	1104.727	1828.250	2	5.35	yes
2	1118.070	1051.477	6.33	1110.312	1245.278	9	5.60	yes
3	1105.440	1047.969	5.48	1096.077	1658.452	1	4.59	yes
4	1079.160	1017.385	6.07	1074.979	1995.801	10	5.66	yes
5	1117.620	1054.653	5.97	1106.464	1648.595	6	4.91	yes
6	1051.260	985.254	6.70	1043.467	1070.935	4	5.91	yes
7	1138.590	1078.650	5.56	1136.636	1594.024	9	5.38	yes
8	1112.700	1050.654	5.91	1106.025	2118.244	6	5.27	yes
9	1155.010	1077.064	7.24	1140.360	888.275	3	5.88	yes
10	1047.700	992.060	5.61	1044.391	1427.077	4	5.28	yes
11	1061.080	993.088	6.85	1048.174	3658.813	3	5.55	yes
12	1111.580	1042.317	6.65	1101.045	1479.012	3	5.63	yes
13	1084.010	1015.896	6.70	1078.415	2595.106	7	6.15	yes
14	1057.480	999.349	5.82	1053.560	1422.297	6	5.42	yes
15	1100.650	1036.539	6.19	1098.935	2158.163	6	6.02	yes
16	1092.330	1032.529	5.79	1083.318	2190.453	8	4.92	yes
17	1206.120	1134.773	6.29	1196.739	1527.039	3	5.46	yes

Table 4 (Continued)

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
18	1111.210	1055.557	5.27	1107.420	1751.125	8	4.91	yes
19	1071.840	1010.115	6.11	1066.128	5288.727	2	5.55	yes
20	1112.400	1053.188	5.62	1103.919	2050.216	6	4.82	yes
21	1092.440	1029.058	6.16	1083.063	1186.734	10	5.25	yes
22	1036.480	974.752	6.33	1027.713	2435.408	8	5.43	yes
23	1101.340	1031.378	6.78	1092.071	1190.802	4	5.88	yes
24	1104.400	1035.416	6.66	1097.585	2037.423	3	6.00	yes
25	1095.780	1030.565	6.33	1090.636	2278.575	10	5.83	yes
26	1062.130	999.691	6.25	1059.917	2116.016	1	6.02	yes
27	1075.840	1011.056	6.41	1069.501	1594.648	3	5.78	yes
28	1077.270	1011.218	6.53	1073.943	1588.139	2	6.20	yes
29	1150.040	1076.069	6.87	1137.530	1765.426	1	5.71	yes
30	1041.360	988.261	5.37	1036.628	1693.930	8	4.89	yes
31	1072.380	1010.986	6.07	1064.187	1704.863	5	5.26	yes
32	1069.780	998.009	7.19	1060.309	1864.445	3	6.24	yes
33	1118.910	1050.361	6.53	1111.018	1654.515	3	5.77	yes
34	1072.150	1020.362	5.08	1074.551	3057.399	3	5.31	no
35	1142.252	1073.319	6.42	1128.925	1265.307	1	5.18	yes
36	1130.150	1063.698	6.25	1127.004	2542.987	4	5.95	yes
37	1126.950	1058.023	6.51	1114.458	1235.497	10	5.33	yes
38	1132.680	1071.240	5.74	1124.892	1722.432	1	5.01	yes
39	1102.900	1041.975	5.85	1097.777	2023.678	4	5.36	yes
40	1098.310	1030.333	6.60	1087.756	1189.118	7	5.57	yes
41	1081.720	1016.515	6.41	1072.136	1318.062	5	5.47	yes
42	1009.950	951.936	6.09	1008.490	1724.516	8	5.94	yes
43	1069.070	1005.234	6.35	1061.292	1722.712	5	5.58	yes
44	1105.940	1045.801	5.75	1098.635	2845.270	1	5.05	yes
45	1149.150	1073.378	7.06	1137.514	1442.134	3	5.98	yes
46	1124.670	1061.748	5.93	1122.407	2106.192	1	5.71	yes
47	1142.940	1075.421	6.28	1133.873	1241.770	7	5.44	yes
48	1075.840	1013.634	6.14	1062.411	1265.299	1	4.81	yes
49	1039.850	977.940	6.33	1034.727	1230.213	1	5.81	yes
Average	1095.056	1030.994	6.19	1087.784	1829.977		5.48	

Table 5 Results for the instances with 150 nodes, central node located in the center

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
0	1580.940	1483.970	6.53	1555.086	4088.422	4	4.79	yes
1	1656.710	1569.578	5.55	1639.308	8907.453	7	4.44	yes
2	1644.040	1550.347	6.04	1624.750	7602.703	8	4.80	yes
3	1613.520	1509.925	6.86	1586.540	7786.656	6	5.07	yes
4	1659.450	1555.021	6.72	1633.248	4725.172	5	5.03	yes
5	1681.760	1583.078	6.23	1658.822	6100.688	3	4.78	yes
6	1580.190	1486.176	6.33	1560.927	5875.156	1	5.03	yes
7	1614.120	1520.713	6.14	1601.457	6128.094	6	5.31	yes
8	1616.600	1513.958	6.78	1584.621	6723.203	8	4.67	yes
9	1575.550	1477.302	6.65	1551.760	5727.875	4	5.04	yes
10	1692.590	1589.691	6.47	1670.655	7285.422	3	5.09	yes
11	1560.320	1468.010	6.29	1536.382	4121.359	7	4.66	yes
12	1630.830	1526.610	6.83	1600.248	4777.875	1	4.82	yes
13	1640.320	1548.118	5.96	1620.416	6793.125	4	4.67	yes
14	1623.600	1520.679	6.77	1596.448	6998.672	3	4.98	yes
15	1612.230	1522.013	5.93	1591.070	5340.594	8	4.54	yes
16	1575.520	1466.952	7.40	1546.311	4744.141	7	5.41	yes
17	1662.970	1566.313	6.17	1642.191	7605.047	4	4.84	yes
18	1656.130	1548.836	6.93	1627.517	6613.781	6	5.08	yes
19	1690.210	1600.140	5.63	1674.517	8642.484	8	4.65	yes
20	1540.850	1452.732	6.07	1524.373	22588.672	1	4.93	yes
21	1627.950	1542.231	5.56	1614.360	7018.938	4	4.68	yes
22	1633.970	1539.500	6.14	1615.515	4026.125	5	4.94	yes
23	1643.040	1538.718	6.78	1611.305	6657.594	2	4.72	yes
24	1664.360	1552.271	7.22	1626.808	6818.172	3	4.80	yes
25	1616.310	1512.606	6.86	1588.242	5791.656	8	5.00	yes
26	1587.260	1485.253	6.87	1564.279	11945.141	3	5.32	yes
27	1691.870	1581.592	6.97	1660.209	6437.563	6	4.97	yes
28	1599.100	1506.521	6.15	1573.735	7968.938	2	4.46	yes
29	1607.750	1502.692	6.99	1580.154	4740.234	2	5.15	yes
30	1608.570	1511.768	6.40	1585.971	7222.000	6	4.91	yes
31	1629.970	1529.406	6.58	1608.105	4494.375	1	5.15	yes
32	1640.740	1519.517	7.98	1599.565	6046.578	5	5.27	yes
33	1667.140	1572.180	6.04	1646.669	7204.375	8	4.74	yes
34	1548.240	1451.053	6.70	1523.056	6650.594	3	4.96	yes
35	1648.000	1548.540	6.42	1626.485	8050.063	7	5.03	yes
36	1551.870	1456.257	6.57	1525.583	4232.906	10	4.76	yes
37	1598.370	1509.759	5.87	1576.402	7346.703	6	4.41	yes
38	1546.380	1441.415	7.28	1519.990	12916.313	8	5.45	yes
39	1604.670	1520.043	5.57	1584.219	4259.750	3	4.22	yes

Table 5 (Continued)

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
40	1701.160	1595.441	6.63	1673.146	11445.047	10	4.87	yes
41	1625.390	1530.982	6.17	1608.741	7715.984	9	5.08	yes
42	1654.450	1554.185	6.45	1626.899	6533.875	9	4.68	yes
43	1634.980	1539.881	6.18	1610.013	4597.219	6	4.55	yes
44	1643.670	1546.491	6.28	1613.592	5924.797	5	4.34	yes
45	1653.450	1554.087	6.39	1630.234	5386.359	9	4.90	yes
46	1669.640	1567.323	6.53	1641.708	4399.047	2	4.75	yes
47	1637.750	1534.329	6.74	1610.863	8037.734	5	4.99	yes
48	1613.150	1523.084	5.91	1597.074	6040.188	3	4.86	yes
49	1687.700	1588.236	6.26	1668.591	5103.953	10	5.06	yes
Average	1625.220	1526.607	6.44	1601.140	6827.744		4.86	

Acknowledgements We are in debt with S. Raghu Raghavan and Ioannis Gamvros for kindly making available their instances and bound values. We also wish to thank Eduardo Uchoa and Haroldo Santos for useful discussions. The second author was supported by CNPq grant 301603/2006-5, Brazil.

References

- Ahuja, R.V., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice-Hall, Englewood Cliffs (1993)
- Ahuja, R.K., Orlin, J.B., Sharma, D.: Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Math. Program.* **91**, 71–97 (2001)
- Amberg, A., Domschke, W., Voss, S.: Capacitated minimum spanning tree: Algorithms using intelligent search. *Comb. Optim.: Theory Pract.* **1**, 9–40 (1996)
- Berger, D., Gendron, B., Potvin, J.Y., Raghavan, S., Soriano, P.: Tabu search for a network loading problem with multiple facilities. *J. Heuristics* **6**, 253–267 (2000)
- Brimberg, J., Hansen, P., Lih, K.-W., Mladenović, N., Breton, M.: An oil pipeline design problem. *Oper. Res.* **51**, 228–239 (2003)
- de Souza, M.C., Duhamel, C., Ribeiro, C.C.: A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. In: Resende, M.G.C., De Sousa, J.P. (eds.) *Metaheuristics: Computer Decision-Making*, pp. 627–657. Kluwer Academic, Dordrecht (2003)
- Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. *J. Heuristics* **2**, 5–30 (1996)
- Esau, L.R., Williams, K.C.: On teleprocessing system design. *IBM Syst. J.* **5**, 142–147 (1966)
- Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6**, 109–133 (1995)
- Festa, P., Resende, M.G.C.: GRASP: An annotated bibliography. In: Ribeiro, C.C., Hansen, P. (eds.) *Essays and Surveys in Metaheuristics*, pp. 325–367. Kluwer Academic, Dordrecht (2001)
- Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP. AT&T Labs Research Technical Report, TD-5WYSEW (2004)
- Gamvros, I., Raghavan, S., Golden, B.L.: An evolutionary approach for the multi-level capacitated minimum spanning tree. Technical Research Report TR 2002-18, Institute for Systems Research, University of Maryland (2002)
- Gamvros, I., Raghavan, S., Golden, B.L.: An evolutionary approach for the multi-level capacitated minimum spanning tree. In: Anandalingam, G., Raghavan, S. (eds.) *Telecommunications Network Design and Management*, pp. 99–124. Kluwer Academic, Dordrecht (2003)

- Gamvros, I., Golden, B.L., Raghavan, S.: The multi-level capacitated minimum spanning tree. *INFORMS J. Comput.* **18**, 348–365 (2006)
- Gavish, B.: Topological design of centralized computer networks: Formulations and algorithms. *Networks* **12**, 355–377 (1982)
- Gavish, B.: Topological design of telecommunication networks-local access design methods. *Ann. Oper. Res.* **33**, 17–71 (1991)
- Gouveia, L.: A $2n$ formulation for the capacitated minimal spanning tree problem. *Oper. Res.* **43**, 130–141 (1995)
- Gouveia, L., Martins, P.: A hierarchy of hop-indexed models for the capacitated minimum spanning tree problem. *Networks* **35**, 1–16 (2000)
- Gouveia, L., Martins, P.: The capacitated minimum spanning tree problem: revisiting hop-indexed formulations. *Comput. Oper. Res.* **32**, 2435–2452 (2005)
- Hall, L.: Experience with a cutting plane approach for the capacitated spanning tree problem. *INFORMS J. Comput.* **8**, 219–234 (1996)
- Martins, P.: Enhanced second order algorithm applied to the capacitated minimum spanning tree problem. *Comput. Oper. Res.* **34**, 2495–2519 (2007)
- Martins, A.X., Souza, M.J.F., de Souza, M.C.: Modelos matemáticos para o problema da árvore geradora mínima capacitada em níveis. In: *Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional*, pp. 1971–1982. Gramado, Brazil (2005) (in Portuguese)
- Papadimitriou, C.: The complexity of the capacitated tree problem. *Networks* **8**, 217–230 (1978)
- Raghavan, S.: Personal communication, 2007
- Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) *Handbooks of Metaheuristics*, pp. 219–249. Kluwer Academic, Dordrecht (2003)
- Rothfarb, B., Goldstein, M.: The one-terminal telpak problem. *Oper. Res.* **19**, 156–169 (1971)
- Sharaiha, Y., Gendreau, M., Laporte, G., Osman, I.: A tabu search algorithm for the capacitated shortest spanning tree problem. *Networks* **29**, 161–171 (1997)
- Uchoa, E., Fukasawa, R., Lysgaard, J., Pessoa, A., Poggi de Aragão, M., Andrade, D.: Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation. *Math. Program.* **112**, 443–472 (2008)

A hybrid metaheuristic algorithm for the Integrated Vehicle and Crew Scheduling Problem

M.J.F.Souza^{*,1}, S.Ribas^{**,2}, I.M.Coelho^{**,3}

Federal University of Ouro Preto, Department of Computer Science, Ouro Preto, Minas Gerais, Brazil 35400-000

Abstract

This work deals with the Integrated Vehicle and Crew Scheduling Problem (VCSP) in urban mass transit. The Vehicle Scheduling Problem (VSP) consists in creating a daily routine of operation for a fleet of vehicles of a company so that all timetabled trips are covered and the operational costs of such activity reduced, making the best use of the fleet. In the Crew Scheduling Problem (CSP) each crew corresponds to a duty, and the goal is the generation of its schedule, satisfying a set of labor agreement and operational rules of the company, as well as the workforce optimization. Traditionally, the VSP and the CSP are solved sequentially by the vehicles-first duties-second approach, but these methodologies do not explore the link between both problems. The difficulty of solving VCSP is greater than the individual problems since it contains all the degrees of freedom of vehicle scheduling and all the degrees of freedom of crew scheduling. Since VCSP is NP-hard, an algorithm based on metaheuristics approaches is proposed here. This algorithm combines Iterated Local Search (ILS), Variable Neighborhood Descent (VND) and Tabu Search with Adaptive Relaxation (TSAR). In order to explore the solution space, we defined six movements based on relocation and swap of both, trips of the vehicles schedule and tasks of the duties schedule. The algorithm was tested with real data of a Brazilian city. The results show the effectiveness of the proposed approach.

Key words: Vehicle Scheduling, Crew Scheduling, Integrated Vehicle and Crew Scheduling, Public Transport, Metaheuristics

1. INTRODUCTION

The public transportation system planning is a very complex activity. Thus it is generally divided into five main stages, which are: route planning, timetables defining, vehicle scheduling, crew scheduling and crew rostering.

Figure 1 shows the traditional stages of public transportation system planning, where the output of one stage is the input for the subsequent one.

^{*}Principal corresponding author

^{**}Corresponding author

¹marcone@iceb.ufop.br

²imcoelho@iceb.ufop.br

³sabir@iceb.ufop.br

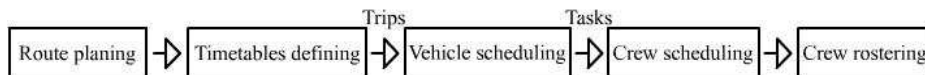


Figure 1: Stages of public transportation system planning

This work deals with the third and fourth stages, the vehicle and crew scheduling problems.

According to Freling et al. (2003), there are three different approaches to solve the vehicle and crew scheduling: Sequential, Independent and Integrated.

In the sequential approach, the vehicle and crew scheduling problems are solved separately and the solution of one is used as input for the other. The sequential approach, in turn, can be classified as Traditional or Reverse.

The traditional sequential approach was the first one to be studied and it is widely adopted by the companies. It consists in solving the vehicle scheduling problem (VSP) and then, considering this solution as input, the crew scheduling problem (CSP) is solved. This approach has the inconvenience that the vehicle scheduling solution guides the crew scheduling solution (Freling et al., 2003), which has the most relevant cost among the ones involved in the system (Bouzada, 2003). An alternative to solve it is to apply the traditional sequential approach, but considering the crew features during the vehicle scheduling solution. Even though the vehicle scheduling costs increase, it becomes easier to solve the CSP and much probably the total cost will be lower than the traditional approach. Scott (1985), Darby-Dowman et al. (1988), Reis (2008) use this strategy for solving the problem involving vehicle and crew scheduling.

In the reverse sequential approach, the CSP is solved first and then, considering this solution as input, the VSP is solved. This strategy is justified for two main motives: 1) the crew costs are higher than vehicle costs and 2) the set of vehicle restrictions is much smaller, which suggests solving CSP first, because it has fewer alternatives for solution.

In the independent approach the vehicle schedule and the crew schedule are built directly from trip timetables. Therefore, in this approach the vehicle schedule is built without considering any crew characteristics and the crew schedule is obtained without considering vehicle constraints. Generally, the results obtained by this approach are not feasible in practice, because some vehicle trips will not have a driver, and some drivers will operate virtual vehicles that are not in the vehicle schedule. This approach is used to determine lower bounds for VSP and CSP.

In the integrated approach both problems are solved together in the same model. As both problems are NP-hard (Garey and Johnson, 1979), then Vehicle and Crew Scheduling Problem (VCSP) is NP-hard. It is obvious than the difficulty of solving VCSP is bigger because the solution space is huge, since VCSP contains all the degrees of freedom of vehicle scheduling and all the degrees of freedom of crew scheduling. However, with the improvement in the CPU speed of the computers, combined with algorithm progress, the research has taken up this topic recently. Freling et al. (1995), Haase and Friberg (1999) and Freling et al. (1999) use this type of approach. See Borndröfer et al. (2006) and Freling et al. (2003) for a survey and experience about this topic.

This paper proposes an algorithm, named ILS-VND-TSAR, which combines

Iterated Local Search, Variable Neighborhood Descent and Tabu Search heuristics for solving VCSP. This heuristic strategy is justified because VCSP is NP-hard and according to the literature these approaches are able to produce good solutions for each problem, considered individually (See Souza et al. (2007) and Marinho et al. (2004)).

The rest of the paper is organized as follows. Section 2 describes the problem. Section 3 presents the proposed methodology, including the representation of the problem, the procedure employed for generating an initial solution, the neighborhood structures, the evaluation function and finally the adaptation of the Iterated Local Search, Variable Neighborhood Descent and Tabu Search heuristics to the problem. Section 4 shows the results achieved in this study and discusses the improvement obtained. Section 5 brings the concluding remarks of this work.

2. Problem definition

According to Freling et al. (2003), the Vehicle and Crew Scheduling Problem (VCSP) is defined as follows. Given a set of trips within a fixed planning horizon, the objective is to find a minimum cost schedule for the vehicles and the crews, such that both the vehicle and the crew schedules are feasible and mutually compatible. Each trip has fixed starting and ending times, and the traveling times between all pairs of location are known. A vehicle schedule is feasible if (1) each trip is assigned to a vehicle, and (2) each vehicle performs a feasible sequence of trips, where a sequence of trips is feasible if a vehicle can execute each pair of consecutive trips in the sequence. Some trips do not belong to any route and these are called the **dead trips**. A dead trip time is the time a vehicle takes to travel between two locations (terminals or depots) that do not belong to any route. This is necessary in order to move the vehicle from an ending point to the next starting point, so that the vehicle can start the next trip. A vehicle schedule defines which trips have to be performed by the same vehicle and this defines the so-called blocks. If there is enough time between one trip and the next, a change of driver may occur, and these are called **relief points**, defined by location and time. The blocks are divided by relief points, and each part is the so-called **task**, that is defined by two consecutive relief points and represents the minimum piece of work that can be assigned to a crew. These tasks have to be assigned to crew members, and the tasks of a crew member define a crew duty. All the duties constitute a crew schedule. Such schedule is feasible if (1) each task is assigned to one duty, and (2) each duty is a sequence of tasks that can be performed by a single crew, both from a physical and a legal point of view. In particular, each duty must satisfy several complicated constraints corresponding to work load regulations for the crews. Typical examples of such constraints are the maximum working time without a break, minimum break duration and maximum total working time. The cost of a duty is usually a combination of fixed costs such as wages, and variable costs such as overtime payment. The subsection 3.4 details the considered requirements in this work.

It is assumed that there is only one depot and all vehicles are always available. We assumed too that from 05:00 to 08:00 and from 16:00 to 19:00 two minutes are necessary for a change of driver, while in the other periods, only one minute is necessary.

3. Proposed methodology

A solution s to the VCSP consists in a (s_v, s_c) pair, where s_v represents a solution to the VSP and s_c represents a solution to the CSP. The s_v solution consists in a set of vehicles, where each one has an associated list of trips to be done during a day. The s_c solution consists in a set of crew duties, where each one is associated to a list of tasks to be carried out daily. Figure 2 shows an example.

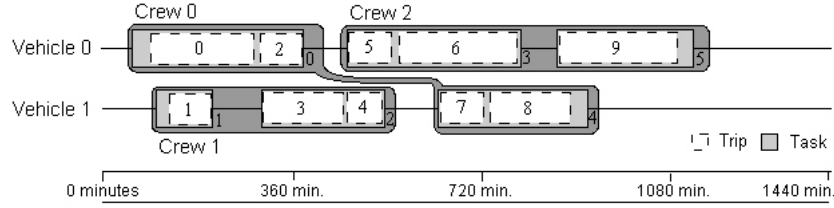


Figure 2: Solution representation

In the Figure 2 there are two vehicles and three crew duties. The Vehicle 1 performs the trips 1, 3, 4, 7 and 8. The crew 0 performs the tasks 0, makes a vehicle change and performs the task 4. Just remembering, some trips are grouped in the same task because there is not enough time to make a vehicle change.

We observe that the solution representation adopted allows a vehicle to make more than one trip at the same moment, and a crew to make more than one task at the same time. To eliminate these practical inconsistencies, our evaluation function penalize these occurrences.

3.1. Evaluation function

The cost of VCSP is calculated by equation (1), that we try to minimize. In this equation, $f_v(s)$ represents the component of f that evaluates the solution s with respect to the vehicle scheduling and $f_c(s)$ evaluates the solution s with respect to the duty scheduling.

$$f(s) = f_v(s) + f_c(s) \quad (1)$$

3.1.1. Vehicle schedule evaluation

In this work we try to minimize the follow items about the vehicle schedule:

- Number of vehicles
- Down time, i.e., the time that a vehicle is idle
- Dead trip time, i.e., the time that a vehicle performs trips without passengers

According to the companies that we contacted, a solution can not be used if more than 60% of the vehicles comes back to the depot during its duties. Thus, a vehicle scheduling is feasible if:

- There are no conflict time in the vehicle schedule

- The number of split vehicle duties is less than 60% of the number of vehicle duties

The vehicle schedule cost is given by equation (2):

$$f_v(s) = f(s_v) = \sum_{i \in Q_v} \alpha_i q_i^v(s_v) + \sum_{i \in O_v} \gamma_i o_i^v(s_v) \quad (2)$$

where Q_v and O_v are the quality and the operational requirements of the vehicle schedule, respectively; α_i and γ_i are the penalties applied to the quality and the operational requirements of a vehicle solution s_v , respectively; and q_i^v and o_i^v are the values of the i th quality and operational vehicle requirement, respectively.

3.1.2. Crew evaluation

In relation to the crew schedule, we try to minimize the follow items:

- Number of crew duties
- Overtime payment
- Crew idleness time

According to the Brazilian labor laws, the crews can not work more than nine hours in one day. Moreover, there can be no less than eleven hours between the end of a workday and the beginning of the next day journey. We also consider the following operational restriction: a solution can not be used if more than 25% of the crew comes back to the depot during its duties. Thus, a crew scheduling is feasible if:

- There is no conflict time in the crew schedule
- There is no crew working more than 9 hours per day
- There is no duty having less than 11 hours between its ending time and its next starting time
- The number of split crew duties is less than 25% of the number of crew duties

The crew schedule cost is given by equation (3):

$$f_c(s) = f(s_c) = \sum_{i \in Q_c} \beta_i q_i^c(s_c) + \sum_{i \in O_c} \theta_i o_i^c(s_c) \quad (3)$$

where Q_c and O_c are the quality and the operational requirements of the crew schedule, respectively; β_i and θ_i are the penalties applied to the quality and the operational requirements of a crew solution s_c , respectively; and q_i^c and o_i^c are the values of the i th quality and operational crew requirement, respectively.

Table 1: Used data on the weights calculation

Description	Value
Driver payment (month)	R\$ 1000,99
Ticket collector payment	R\$ 600,59
Crew payment	R\$ 1601,58
Crew payment + taxes (38%)	R\$ 2210,18
Crew workdays in a month	24
Work time during a day (minutes)	430
Payment about one work minute	R\$ 0,21
Payment of one overtime minute	R\$ 0,32
Payment of one workday	R\$ 92,09
Fixed cost of one vehicle	R\$ 13415,58
Vehicle workdays in a month	30
Vehicle cost daily	R\$ 447,19
Average velocity (km/h)	20
Cost for kilometer	R\$ 2,64
Cost of one minute of dead trip	R\$ 0,88
Variable cost of one kilometer	R\$ 0,78
Cost of one minute of downtime	R\$ 0,26

3.1.3. Defining weights to the evaluation function

The objective of the evaluation function is to evaluate the quality of a solution. As we saw in previous sections, this function considers requirements that must to be minimized and some of these requirements are relatively more important than others. To contemplate the relative importance of the requirements, our the weight used are based on real costs of a Brazilian city company, as used in Reis (2008). The Table 1 presents the data used to calculate the evaluation function weights, where R\$ means Brazilian currency.

The Table 2 presents the weights of the VCSP evaluation function. The quality weights are based on the real costs presented in the Table 1, while the operational weights are empirical and its values are greater than the quality ones. The last column of this table give us the characteristics of an example solution.

The equations (4), (5), (6), (7), (8) and (9) show how is made the evaluation the solution of the Table 2.

$$f(s) = f_v(s) + f_c(s) = \sum_{i \in Q_v} \alpha_i q_i^v(s_v) + \sum_{i \in O_v} \gamma_i o_i^v(s_v) + \sum_{i \in Q_c} \beta_i q_i^c(s_c) + \sum_{i \in O_c} \theta_i o_i^c(s_c) \quad (4)$$

$$\sum_{i \in Q_v} \alpha_i q_i^v(s_v) = 447.19 \times 5 + 0.26 \times 1700 + 0.88 \times 161 = 2819.63 \quad (5)$$

$$\sum_{i \in O_v} \gamma_i o_i^v(s_v) = 4000 \times 0 + 1000 \times 0 = 0 \quad (6)$$

$$\sum_{i \in Q_c} \beta_i q_i^c(s_c) = 92.09 \times 11 + 0.21 \times 1702 + 0.32 \times 815 = 1631.21 \quad (7)$$

Table 2: Weights of the VCSP evaluation function

Type	Refers to	Requirement	Weight	Example of s
Quality	Vehicles	Number of vehicles	447.19	5
Quality	Vehicles	Down time	0.26	1700
Quality	Vehicles	Dead trip time	0.88	161
Quality	Crews	Number of crews duties	92.09	11
Quality	Crews	Crew idleness time	0.21	1702
Quality	Crews	Overtime payment	0.32	815
Operational	Vehicles	Conflict time	4000	0
Operational	Vehicles	Excess of split duties	1000	0
Operational	Crews	Conflict time	4000	0
Operational	Crews	Excess of split duties	1000	0
Operational	Crews	Insufficient time between duties	4000	0
Operational	Crews	Excess time worked	4000	0

$$\sum_{i \in O_c} \theta_i o_i^c(s_c) = 4000 \times 0 + 1000 \times 0 + 4000 \times 0 + 4000 \times 0 = 0 \quad (8)$$

$$f(s) = 2819.63 + 1631.21 = 4450.84 \quad (9)$$

3.2. Constructing an initial solution

An initial solution to the VCSP is made by a sequential mechanism. To begin with, an initial solution for the vehicle scheduling is built. Later, from this vehicle solution, a crew scheduling solution is generated. Both solutions are built by a greedy heuristic.

An initial solution to the VSP is obtained by applying a constructive heuristic where at each iteration, a new trip not yet assigned is added to the current schedule of the best vehicle, in accordance to the value of the evaluation function relative to each vehicle scheduling (see eq. (2) in subsection 3.1.1). The choice of the trip to be added to the vehicle schedule is totally greedy, which means that the trip with lower cost will be chosen. The approach ends when all the trips are allocated. The trips belonging to each vehicle are known as the blocks of the vehicle.

From the blocks of each vehicle, a solution for the CSP is built. Initially, for each block, the trips are grouped in a task until a relief point is found. A relief point is defined by location and time, where and when a change of driver may occur.

Next, the tasks are ordered by their starting time. At each step, there is a set of non-empty duties and a duty without tasks. The constructive procedure works as follows. The first task is assigned to the first duty, the empty one. Next, for each task of the sequence, we evaluate its insertion for all the duties and also for a new duty without tasks. The evaluation of each crew duty is carried out according to the eq. (3), relative to the crew scheduling (see subsection 3.1.2). The task is assigned to the duty with the lowest cost. The crew schedule is interrupted when all the tasks are allocated.

3.3. Neighborhood Structure

The heuristic methods used in this work are based on making modifications in a given solution to improve the quality. These modifications are so-called movements and we can apply different types of movements in a solution. A neighborhood structure is defined by a type of movement.

A neighbor of a given solution s is defined by applying a movement in the solution s . Applying different movements the heuristic methods are able to explore different regions of the solution space.

With a view to explore the solutions space we define six neighborhood structures, which are based on relocation (R) and swap (S) of both, trips of the vehicles (v) schedule and tasks of the duties (c) schedule are used.

Two of them (N_v^R and N_v^S) do modifications only on the vehicles, ignoring the repercussions of these movements in the crew scheduling, which creates, possibly infeasible solutions.

Two other structures (N_c^R and N_c^S) only modify the crew scheduling, preserving the compatibility between vehicles and crews.

The third pair of neighborhood structures (N_v^{RP} and N_v^{SP}) is integrated and modifies both items, vehicles and crews. These last two structures are to relocate or to change the vehicles trips followed by a reconstruction of the duties to make the schedules of vehicles and crews compatible. For each vehicle involved in the modification, all tasks that belong to these vehicles are removed from the crew scheduling. Then, from the new configuration of the blocks of the involved vehicles, the new tasks are generated. These tasks, in turn, are added to any crew member, in accordance with the greedy procedure described in the previous section. For these two structures, N_v^{RP} and N_v^{SP} , the change made is "spread" for the crews.

3.4. The Proposed Algorithm

To solve the VCSP we propose the ILS-VND-TSAR algorithm, which is described in section 3.4.5. The following sections show the components of this algorithm.

3.4.1. The Vehicle Random Descent Procedure

For large instances, finding the best neighbor at each iteration is a costly operation. In these cases, a random approach is normally used. A classical Random Descent (RD) procedure works as follows. A point s' is generated at random from the k th neighborhood of s . If s' is better than the incumbent, then s is updated to s' ($s \leftarrow s'$) and the search continues from s . If there are not any improvements in $RDmax$ iterations, then the search stops. Therefore, the procedure can return not necessarily a local optimum.

When it comes to integrated problems, finding the best neighbor at each iteration is a much more costly operation. This is due to the fact that in integrated neighborhoods considered (N_v^{RP} and N_v^{SP}), each modification in the vehicle schedule is reflected in the crew scheduling. To reduce the number of spreadings of these modifications in the crew scheduling, the Vehicle Random Descent (VRD) procedure is proposed as described in Algorithm 1.

The working principle of this algorithm is quite simple and it is similar to the classic random descent. The main difference is that in this new algorithm, only

the changes made in vehicles (relocation or swap) that improve the evaluation function f_v are propagated in the crew scheduling.

In the Algorithm 1, s is a solution formed by the pair of solutions s_v and s_c , $VRDmax$ is the maximum number of iterations without improvement, N_v is a specific neighborhood structure (relocation or swap) for the vehicles, m is the movement made in the considered structure, N_v^P is the neighborhood structure (relocation or swap) with the spreading in accordance to the movement m and f_v is the function that evaluates the part s_v of the solution s .

It is observed that the final solution of this algorithm can, not necessarily be better than the initial solution in relation to the integrated function f since the vehicles evaluation function f_v is the only one used.

Algorithm 1: RDP

Input: Solution s , Integer $VRDmax$, Neighborhood $N_v(\cdot)$,
Neighborhood $N_v^P(\cdot)$, Function $f_v(\cdot)$
Output: Solution s^* better than or equal to s according to the function f_v

```

 $iter \leftarrow 1$ ;
while ( $iter \leq VRDmax$ ) do
     $m \leftarrow \text{randomMovement}(s, N_v)$ ;
     $s' \leftarrow \text{neighbor}(s, m)$ ;
    if  $s'$  better than  $s$  according to the function  $f_v$  then
         $s \leftarrow \text{neighbor}(s, N_v^P, m)$ ;
         $iter \leftarrow 0$ ;
    end
     $iter \leftarrow iter + 1$ ;
end
 $s^* \leftarrow s$ ;
return  $s^*$ ;

```

3.4.2. The Integrated Vehicle and Crew Random Descent Procedure

This strategy is similar to the one presented in the previous section. There are two basic differences. The first one is that only the movements that produce better solutions with respect to vehicles are candidates (better than f_v) or worse than f_v in at most $(\lambda - 1) \times 100\%$.

The second difference is that it considers both the costs of s_v and the s_c , which is not the case of the Vehicle Random Descent that only considers the cost of s_v when applied to solving the VCSP. The objective of this algorithm is refining a solution s using integrated movements without unduly burdening the processing, as may happen if the refinement was done by classical Random Descent algorithm using only integrated movements.

The Algorithm 2 presents the Integrated Vehicle and Crew Random Descent Procedure (IRD). In this algorithm, s is a solution formed by the pair of the mutually compatible solutions s_v and s_c , $IRDmax$ is the number of iterations without improvement in f , m is a movement made in the vehicles in accordance with the neighborhood N_v (relocation or swap), N_v^P is the neighborhood structure (relocation or swap), with spreading in accordance with the modification m , f_v is the function that evaluates s_v and λ , with $\lambda \geq 1$, is a value used to

determine if it worth processing the most expensive part of the refinement, the construction and the spreading of a integrated movement, in the absence of improvement. For example, $\lambda = 1.01$ implies that if a movement generates a solution s'_v which costs more than 1% of the cost of s_v , the method does not test the movement of spreading. Otherwise, the test is performed and, if the resulting solution s' of the movement of spreading is better than s , then the solution s is updated, that is, $s \leftarrow s'$. Tests have shown that for the instances examined, 1.01 is a good value for λ and this is the value used in this work.

Algorithm 2: IRD

Input: Solution s , Integer $IRDmax$, Neighborhood $N_v(\cdot)$, Neighborhood $N_v^P(\cdot)$, Function $f(\cdot)$, Real λ
Output: Solution s^* better than or equal to s according to the function f

```

 $iter \leftarrow 1;$ 
while ( $iter \leq IRDmax$ ) do
     $m \leftarrow \text{randomMovement}(s, N_v);$ 
     $s' \leftarrow \text{neighbor}(s, m);$ 
    if  $f_v(s') \leq \lambda \times f_v(s)$  then
         $s'' \leftarrow \text{neighbor}(s, N_v^P, m);$ 
        if  $s''$  better than  $s$  according to the function  $f$  then
             $s \leftarrow s'';$ 
             $iter \leftarrow 0;$ 
        end
    end
     $iter \leftarrow iter + 1;$ 
end
 $s^* \leftarrow s;$ 
return  $s^*;$ 

```

3.4.3. The Tabu Search with Adaptative Relaxation Procedure

The Tabu Search methods (Glover and Kockenberger, 2003) is an iterative procedure for solving combinatorial optimization problems, which accepts non-improvement movements to avoid be trapped in optima local.

Starting from an initial solution s_0 , the method explores, at each iteration, a subset V of the neighborhood $N(s)$, of current solution s . The member $s' \in V$ with the best evaluation according to the function $f(s')$ became the current solution, even if s' is worst than s . This strategy, however, can led the search process to visit solutions already reached before. In order to prevent cycles, short term memory is employed, usually in the form of a Tabu List, which forbids undoing the last movements. Movements remain in the tabu list for a given number of iterations (tabu tenure). Since this strategy can be too restrictive, in order to not disregard high quality solutions, movements with tabu status can be accepted if the produced solution satisfies an aspiration criterion, which usually considers the cost of the best solution found so far, that is, if the produced solution is better than the best solution found so far, the movement can be accepted. With the aim of reducing the risk of revisiting the same solutions, we use a dynamic tabu list, in our case, the tabu tenure value is selected from an interval, which

defines a set of possible values $\{minTabuTenure, \dots, maxTabuTenure\}$.

The procedure TSAR (Tabu Search with Adaptive Relaxation), proposed in Marinho et al. (2004), is applied for solving the Crew Scheduling Problem. Its pseudo-code is presented at Algorithm 3.

In the TSAR procedure we seek for the First Improvement in a subset of the neighborhoods N_c^R or N_c^S , or if there is not a improvement, it returns the best neighbor in this subset. This procedure also incorporates an Adaptive Relaxation (AR) mechanism. This mechanism consists in changing the weights of the infeasibilities periodically during the search, either promoting the generation of infeasible solutions or feasible solutions in order to explore other regions not yet visited in the solution space.

Our AR implementation is based on the proposal of Schaefer (1996), where weights of each source of infeasibility are dynamically updated, as originally proposed by Gendreau et al. (1994). For each source of infeasibility i , the corresponding weight θ_i in the objective function f_c is multiplied by a factor δ_i , which is updated as follows:

1. In the beginning of the search set $\delta_i \leftarrow 1$.
2. At every k movements:
 - if all k visited solutions are feasible considering constraint i then $\delta_i \leftarrow \delta_i / \gamma$;
 - if all k are infeasible considering constraint i then $\delta_i \leftarrow \delta_i \times \gamma$;
 - if part of k solutions are feasible, and another part is infeasible considering constraint i , δ_i keeps unchanged.

The parameter μ is randomly selected in the interval $[1.8, 2.2]$. As observed in Schaefer (1996), this strategy tends to avoid deterministic ratios which could bias the search.

Each value of δ_i is bounded by two constants $\delta_{i,min}$ and $\delta_{i,max}$, which prevents the adaptive relaxation from indefinitely increasing/decreasing weights for constraints which are always unsatisfied/satisfied.

In the Algorithm 3, *UpdateIteration()* indicates when the dynamic weights must be updated (In our case, every $k = 10$ iterations). In our implementation, the subset V consists in 50% of the number of crews, selected at random.

3.4.4. The Two Levels Variable Neighborhood Descent

With an aim to introduce the Variable Neighborhood Descent - VND (Hansen and Mladenovic, 2001; Hansen et al., 2008), let $N^k (k = 1, \dots, k_{max})$ be a finite set of pre-selected neighborhood structures, and $N^k(s)$ the set of solutions in the k th neighborhood of s . Neighborhoods N^k may be induced from one or more metric (or quasi-metric) functions introduced into a solution space S . An optimal solution is a feasible solution where a minimum of the evaluated function f is reached. There is a local minimum s' of f with respect to $N^k(s)$, if there is no solution $s \in N^k(s')$ such that $f(s) < f(s')$. Metaheuristics based on local search procedures try to continue the search by other means after finding the first local minimum. VND is based on three simple facts: (1) A local minimum with respect to one neighborhood structure is not necessary so with another; (2) A global minimum is a local minimum with respect to all possible neighborhood structures; and (3) For many problems local minimum with respect to one or several are relatively close to each other.

Algorithm 3: TSAR

Input: Solution s , Integer $TStime$, Neighborhood N_c (N_c^R or N_c^S),
Function $f_c(\cdot)$
Output: Solution s^* better than or equal to s according to the function
 f_c

$s^* \leftarrow s$;
 $TabuList \leftarrow \emptyset$;
repeat
 Define subset $V \subseteq N_c(s)$
 $bestMovement \leftarrow RandomMovement(V)$
 $bestCost \leftarrow \infty$
 for all Movement $m \in V$ **do**
 if $f_c(s \oplus m) < f_c(s^*)$ **then**
 $bestMovement \leftarrow m$
 break
 end
 else
 if $m \notin TabuList$ **then**
 if $f_c(s \oplus m) < f_c(s)$ **then**
 $bestMovement \leftarrow m$
 break
 end
 else
 if $DynamicWeightsCost(s \oplus m) < bestCost$ **then**
 $bestMovement \leftarrow m$
 $bestCost \leftarrow DynamicWeightsCost(s \oplus m)$
 end
 end
 end
 end
 end
 $s \leftarrow s \oplus bestMovement$
 if $f_c(s) < f_c(s^*)$ **then**
 $s^* \leftarrow s$
 end
 $UpdateTabuList()$
 if $UpdateIteration()$ **then**
 $UpdateDynamicWeights()$
 end
until $TStime$ reached ;
return s^*

According to these authors, this last observation, which is empirical, implies that a local optimum often provides some information about the global one. This may for instance be several variables with the same value in both. However, it is usually not known which ones are such. An organized study of the neighborhood of this local optimum is therefore in order, until a better one is found.

The Variable Neighborhood Descent (VND) approach is obtained if a change of neighborhoods is performed in a deterministic way.

The proposed procedure to do a local search in the solution space of the VCSP, called Two Levels Variable Neighborhood Descent (VND2L), is described in the Algorithm 4.

Algorithm 4: VND2L

Input: Solution s , Function $f(\cdot)$, Function $f_v(\cdot)$, Function $f_c(\cdot)$
Output: Solution s^* better than or equal to s according to the function f

```

 $s_0 \leftarrow s$ ;
 $s_1 \leftarrow \text{VRD}(s_0, \text{VRDmax}, N_v^R, N_v^{RP}, f_v)$ ;
[Step1]
 $s_2 \leftarrow \text{VRD}(s_1, \text{VRSmax}, N_v^S, N_v^{SP}, f_v)$ ;
if  $s_2$  better than  $s_0$  according to the  $f_v$  then
  |  $s_0 \leftarrow s_2$  and go back to Step1;
end
 $s_3 \leftarrow \text{TSAR}(s_2, \text{TStime}, N_c^R, f_c)$ ;
[Step2]
 $s_4 \leftarrow \text{TSAR}(s_3, \text{TStime}, N_c^S, f_c)$ ;
if  $s_4$  better than  $s_2$  according to the function  $f_c$  then
  |  $s_2 \leftarrow s_4$  and go back to Step2;
end
 $s_5 \leftarrow \text{IRD}(s_4, \text{IRDmax}, N_v^R, N_v^{RP}, f, \lambda)$ ;
[Step3]
 $s_6 \leftarrow \text{IRD}(s_5, \text{IRDmax}, N_v^S, N_v^{SP}, f, \lambda)$ ;
if  $s_6$  better than  $s_4$  according to the  $f$  then
  |  $s_4 \leftarrow s_6$  and go back to Step3;
end
if there is an improvement in the Step3 then
  | Go back to Step2;
end
 $s^* \leftarrow s_6$ ;
return  $s^*$ ;

```

The algorithm 4 is based on the ideas of the VND heuristic in the sense that not only the neighborhoods are changed, but also the local search procedures. There are two levels because this procedure has intern levels and an extern one.

The intern levels consist in three VND applications. The first one has the objective of improve the vehicles; the second one, to improve the crews and, the last one, to improve both, vehicles and crews.

The extern level is an application of VND considering that each intern level is an iteration of the external part of the VND2L.

In this algorithm, s , as we will see in the section 3.4.5, is a resulting solution

of the perturbation in a local optimum. The VRD is the Vehicle Random Descent procedure described in the section 3.4.1. The IRD is the Integrated Vehicle and Crew Random Descent Procedure presented in the section 3.4.2. These random approaches, VRD and IRD, are interrupted if no improvements are found in *VRDmax* and *IRDmax* iterations, respectively. The TSAR is the Tabu Search with Adaptive Relaxation procedure, described in the section 3.4.3.

3.4.5. The proposed algorithm for solving VCSP

The proposed algorithm for solving VCSP is based on Iterated Local Search (ILS) metaheuristic. ILS (Lourenço et al., 2003; Stützle, 2006) is a metaheuristic with four basic components: *GenerateInitialSolution*, *LocalSearch*, *Perturbation* and *AcceptanceCriterion*. The *GenerateInitialSolution* is a module that consists in building a good solution to the problem. *LocalSearch* is the module that starts the search from a solution and returns a local optimum. *Perturbation* consists in realizing modifications on the local optimum in order to escape from it. Finally, the procedure *AcceptanceCriterion* consists in determining whether the new solution is accepted or not as the new current solution. According to these authors, a reasonable first guess for the acceptance criterion is to force the cost to decrease. *AcceptanceCriterion* has a strong influence on the nature and effectiveness of the walk in the space of solutions. Roughly, it can be used to control the balance between intensification and diversification of that search. In our case, a perturbation of level i consists in making $i + 2$ random movements in a given neighborhood structure.

The Algorithm 5 presents the ILS-VND-TSAR algorithm applied to solve the VCSP. In this algorithm, s is the initial solution; *ILStime* defines the time limit; *ILSmax* is the maximum number of iterations without improvement in the same level and f is the function to be minimized. We observe that whenever there are some improvement in the current solution, the parameter *level* returns to its first value.

4. Computational results

The proposed algorithm, so-called ILS-VND-TSAR, was implemented in C++ language using the Borland C++ Builder environment, version 5.0 and tested in a PC Intel Pentium IV, with 3.0 GHz and 1 GB of RAM running Windows XP Professional Edition.

Preliminary tests were performed to calibrate the parameters of the proposed algorithm. The number of iterations without improvement of the ILS (ILSmax) was fixed at 50, as well as the number of iterations without improvement of IRD (IRDmax). The number of iterations without improvement of VRD (VRDmax) was fixed at 100. The time limit to apply TSAR was fixed at 20 seconds. The total processing time was limited to 20 minutes.

The first phase of the traditional sequential approach uses ILS algorithm described in Souza et al. (2007) and second one uses the TSAR algorithm described in Marinho et al. (2004). Both algorithms stop after 10 minutes of execution. In this approach, *ILSmax* was fixed at 50, and one perturbation of level i is made applying $i + 2$ moves at the neighborhood N_v^R , the local search of ILS uses two nested random descents, the first using the neighborhood N_v^R and the second, N_v^S .

Algorithm 5: ILS-VND-TSAR

Input: Solution s , Integer $ILStime$, Integer $ILSmax$, Function $f(\cdot)$

Output: Solution s^* better than or equal to s according to the function f

```
 $s^* \leftarrow \text{VND2L}(s, f, f_v, f_c);$ 
 $level \leftarrow 0;$ 
while  $ILStime$  not satisfied do
     $iter \leftarrow 0;$ 
    while  $iter < ILSmax$  and  $ILStime$  not satisfied do
         $s' \leftarrow \text{perturbation}(s^*, level);$ 
         $s'' \leftarrow \text{VND2L}(s', f, f_v, f_c);$ 
        if  $s''$  better than  $s^*$  according to the function  $f$  then
             $s^* \leftarrow s'';$ 
             $level \leftarrow 0;$ 
             $iter \leftarrow 0;$ 
        end
        else
             $iter \leftarrow iter + 1;$ 
        end
    end
     $level \leftarrow level + 1;$ 
end
return  $s^*;$ 
```

In order to test the algorithms, we examined twelve scheduling problems for nine public transportation company that operates in a Brazilian city. The data are available at <http://www.decom.ufop.br/prof/marcone/pt/instances/VC.zip>.

Tables 1 and 2 show the value of the parameters used in the evaluation function, described in section 3.1.3.

Table 3 contains the evaluation function values of the traditional sequential and integrated approaches. The two first columns “Best” show the best value for f found in 20 runs of the algorithms, while the two first columns “Avg.” indicate the average value in these runs. Column “Gap” means the gap between the average value and the best known value to the respective instance. The last columns “Best” and “Avg.” indicate the improvement of the integrated approach over the sequential one with respect to the best and average solution found for each algorithm, respectively. It can be observed that the proposed algorithm was able to reduce both best and average values, as well the gap between the average value and the best known value to each instance. In addition, the gap is also small, ratifying that the proposed algorithm is robust, and there was an improvement of up to 1.64% if compared to the best solution found in the sequential approach and up to 2.39% if compared to the average one.

Tables 4 to 7 present some characteristics of the best solutions found on each instance, in respect of the sequential (Seq.) and integrated (Int.) approaches.

From these tables it can be verified that the proposed algorithm was able to reduce the number of crews in one unit in the instances VC52, VC260a and VC498. In the instances VC90 and VC639a, the number of crews increased in one unit; however, there was a significative reduction on the overtime payment

Table 3: Evaluation function values from the traditional sequential and integrated approaches

Instance	Sequential			Integrated			Improvement	
	Best	Avg.	Gap	Best	Avg.	Gap	Best	Avg
VC52	3936.96	3988.46	2.97	3873.31	3895.43	0.57	1.64	2.39
VC90	7519.20	7561.96	1.14	7476.57	7518.14	0.56	0.57	0.59
VC98a	9638.79	9675.39	1.16	9564.95	9594.67	0.31	0.77	0.84
VC98b	9500.16	9514.50	1.47	9376.67	9398.28	0.23	1.32	1.24
VC260a	28479.80	28714.00	1.43	28310.50	28329.11	0.07	0.60	1.36
VC260b	28064.80	28215.82	0.54	27921.40	27950.78	0.11	0.51	0.95
VC498	32673.50	32756.81	0.26	32500.10	32612.90	0.35	0.53	0.44
VC538	18917.00	18964.83	0.25	18887.20	18939.01	0.27	0.16	0.14
VC639a	39478.10	39586.00	0.27	39301.30	39398.46	0.25	0.45	0.48
VC639b	42557.60	42608.22	0.12	42336.20	42436.66	0.24	0.52	0.40
VC1036	48764.60	48840.86	0.16	48669.10	48778.18	0.22	0.20	0.13
VC1038	49223.90	49335.35	0.23	49085.40	49140.27	0.11	0.28	0.40

Table 4: Best schedules to instances VC52, VC90 and VC98a

Instance	VC52		VC90		VC98a	
	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	5	5	10	10	13	13
Vehicle waiting time (hh:mm)	12:57	12:57	16:13	16:13	11:58	11:58
Dead trip time (hh:mm)	03:50	03:50	08:24	08:24	15:20	15:20
Number of split blocks	0	0	4	4	7	7
Number of crews	12	11	20	21	27	27
Crew idleness (hh:mm)	11:31	08:31	18:44	14:28	17:05	13:29
Overtime payment (hh:mm)	02:25	05:52	14:13	10:00	06:38	05:09
Number of split duties	2	2	1	3	7	7
f_v	2640.4	2640.4	5168.4	5168.4	6809.8	6809.8
f_c	1296.6	1232.9	2350.8	2308.2	2829.0	2755.2
$f_v + f_c$	3937.0	3873.3	7519.2	7476.6	9638.8	9565.0

and on the crew idleness. For the other instances, o number of crews was the same in both algorithms. An interesting observation is that the integrated approach produced worse solutions to instances VC498 and VC538, but on the other hand the crew solution was better, resulting in a schedule with smaller cost. Also, the final solution produced by the integrated approach was better than the sequential one in all the instances.

5. Conclusions

This paper dealt with the Integrated Vehicle and Crew Scheduling Problem. In order to solve it, an algorithm based on Iterated Local Search, Variable Neighborhood Descent and Tabu Search with Adaptive Relaxation, so-called ILS-VND-TSAR, is proposed. It is an adaptation of the ILS algorithm described in Souza et al. (2007) for solving the vehicle scheduling problem and the TSAR algorithm developed by Marinho et al. (2004) for the crew scheduling problem. Two new neighborhoods are introduced, which consist in reassigning and swapping the trips of vehicles followed by the reconstruction of the duties affected by the moves.

Table 5: Best schedules to instances VC98b, VC260a and VC260b

Instance	VC98b		VC260a		VC260b	
Algorithm	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	13	13	40	40	39	39
Vehicle waiting time (hh:mm)	10:21	10:21	19:15	19:15	13:22	22:02
Dead trip time (hh:mm)	14:34	14:34	37:54	37:54	37:54	37:54
Number of split blocks	6	6	22	22	23	23
Number of crews	25	25	76	75	76	76
Crew idleness (hh:mm)	12:50	09:29	91:31	83:03	84:10	79:07
Overtime payment (hh:mm)	15:13	10:59	07:14	08:46	11:28	07:19
Number of split duties	5	5	17	17	17	17
f_v	6744.1	6744.1	20189.0	20189.0	19785.3	19785.3
f_c	2756.1	2632.6	8290.8	8121.5	8279.5	8136.19
$f_v + f_c$	9500.2	9376.7	28479.8	28310.5	28064.8	27921.4

Table 6: Best schedules to instances VC498, VC538 and VC639a

Instance	VC498		VC538		VC639a	
Algorithm	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	47	47	24	24	55	55
Vehicle waiting time (hh:mm)	37:38	39:27	65:26	65:28	50:36	51:45
Dead trip time (hh:mm)	23:36	24:48	18:59	20:23	52:03	52:37
Number of split blocks	25	26	3	3	25	25
Number of crews	94	93	53	53	109	110
Crew idleness (hh:mm)	55:03	44:51	50:55	47:57	57:47	51:29
Overtime payment (hh:mm)	24:36	22:17	33:17	29:48	30:10	17:48
Number of split duties	19	19	2	2	22	22
f_v	22851.1	22942.8	12755.6	12830.1	28133.0	28180.9
f_c	9822.4	9557.3	6161.4	6057.1	11345.1	11120.3
$f_v + f_c$	32673.5	32500.1	18917.0	18887.2	39478.1	39301.3

A comparison between the integrated approach and the traditional sequential one was performed. The first considers both problems simultaneously, while the later first solves the vehicle scheduling and then solves the crew scheduling. The results showed that the integrated approach was able to produce better results, with global improvement of up to 1.64% in relation to the best solution and up to 2.39% in relation to the average solution. In addition, it can be verified that the proposed algorithm is robust since the maximum variation of the final solutions is quite small, namely, 0.57%.

6. ACKNOWLEDGEMENTS

The authors thank CNPq (process 474831/2007-8), FAPEMIG (grant CEX 00357/09) and the Federal University of Ouro Preto for the support given to the development of this study.

References

Borndröfer, R., Grötschel, M., Pfetsch, M., 2006. Public transport to the fore! ORMS Today 33 (2), 30–40.

Table 7: Best schedules to instances VC639b, VC1036 and VC1038

Instance	VC639b		VC1036		VC1038	
Algorithm	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	60	60	69	69	70	70
Vehicle waiting time (hh:mm)	50:42	52:16	66:07	69:13	67:41	69:10
Dead trip time (hh:mm)	32:59	56:06	50:28	51:41	50:32	51:46
Number of split blocks	25	27	35	35	32	33
Number of crews	116	116	136	136	137	137
Crew idleness (hh:mm)	59:44	49:23	67:57	60:41	78:26	67:10
Overtime payment (hh:mm)	31:06	22:01	43:20	37:16	30:50	26:25
Number of split duties	24	24	28	27	28	28
f_v	30525.4	30608.8	34552.2	34664.8	35027.3	35115.6
f_c	12032.2	11727.4	14212.4	14004.4	14196.6	13969.8
$f_v + f_c$	42557.6	42336.2	48764.6	48669.1	49223.9	49085.4

- Bouzada, C. F., 2003. Costs of the Public Transport System by Bus (in portuguese). FUMEC, Brazil.
- Darby-Dowman, K., Jachnik, J. K., Lewis, R. L., Mitra, G., 1988. Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience. In: Daduna, J. R., Wren, A. (Eds.), Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop. Springer Verlag, Berlin, pp. 226–239.
- Freling, R., Huisman, D., Wagelmans, A., 2003. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling* 6 (1), 63–85.
- Freling, R., Wagelmans, A. P. M., ao, J. M. P. P., 1999. An overview of models and techniques for integrating vehicle and crew scheduling. In: Wilson, N. H. M. (Ed.), Computer-Aided Transit Scheduling. Springer Verlag, Berlin, pp. 441–460.
- Freling, R. C., Boender, G. E., ao, J. M. P. P., 1995. An integrated approach to vehicle and crew scheduling. Tech. Rep. 9503/A, Econometric Institute, Erasmus University Rotterdam, Rotterdam.
- Garey, M. R., Johnson, D. S., 1979. A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York.
- Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. *Management science* 40 (10), 1276–1290, linthicum.
- Glover, F., Kockenberger, G., 2003. Handbook of Metaheuristics. Kluwer Academic Publishers.
- Haase, K., Friberg, C., 1999. An exact branch and cut algorithm for the vehicle and crew scheduling problem. In: Wilson, N. H. M. (Ed.), Computer-Aided Transit Scheduling. Springer Verlag, Berlin, pp. 63–80.
- Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: Principles and applications. *European Journal of Operational Research* 130, 449–467.

- Hansen, P., Mladenovic, N., Pérez, J. A. M., 2008. Variable neighborhood search. *European Journal of Operational Research* (191), 593–595.
- Lourenço, H. R., Martin, O. C., Stützle, T., 2003. Iterated local search. In: Glover, F., Kochenberger, G. (Eds.), *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston.
- Marinho, E. H., Ochi, L. S., Drummond, L. M. A., Souza, M. J. F., Silva, G. P., 2004. Tabu search applied to the bus driver scheduling problem (in portuguese). In: *Proceedings of the XXXVI SBPO*. São João Del Rey, Brazil, pp. 1471–1482.
- Reis, J. V. A., 2008. Heuristics based on variable neighborhood search for the bus vehicle crew scheduling problem (in portuguese). Master’s thesis, Escola Politécnica da USP, São Paulo, Brazil.
- Schaerf, A., 1996. Tabu search techniques for large high-school timetabling problems. In: *IEEE Transactions on Systems, Man, and Cybernetics*. pp. 363–368.
- Scott, D., 1985. A large linear programming approach to the public transport scheduling and cost model. In: Rousseau, J. M. (Ed.), *Computer Scheduling of Public Transport 2*. North Holland, Amsterdam, pp. 473–491.
- Souza, M. J. F., Silva, G. P., oes, E. M. L. S., 2007. Vehicle scheduling: an heuristic approach (in portuguese). In: *Transporte em transformação XI: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2006*. Positiva, Brasília, Ch. 2, pp. 39–57.
- Stützle, T., 2006. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research* 174, 1519–1539.



PLANEJAMENTO OPERACIONAL DE LAVRA COM ALOCAÇÃO DINÂMICA DE CAMINHÕES: ABORDAGENS EXATA E HEURÍSTICA

Autor: Francisco César Rodrigues de Araújo

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Mineral do Departamento de Engenharia de Minas da Escola de Minas da Universidade Federal de Ouro Preto, como parte integrante dos requisitos para obtenção do título de Mestre em Engenharia Mineral, área de concentração: Lavra de Minas.

Ouro Preto, 27 de junho de 2008

A827u Araújo, Francisco César Rodrigues de.
Um método heurístico baseado em Iterated Local Search aplicado ao planejamento operacional de lavra com alocação dinâmica de caminhões [manuscrito]. / Francisco César Rodrigues de Araújo. – 2008.
xi, 128f. : il. color., graf., tabs.

Área de concentração: Lavra de Minas.
Orientador: Prof. Dr. Marcone Jamilson Freitas Souza.
Dissertação (Mestrado) – Universidade Federal de Ouro Preto. Escola de Minas. Departamento de Engenharia de Minas. Programa de pós-graduação em Engenharia Mineral.

1.. Engenharia de Minas - Teses. 2. Lavra a céu aberto - Teses. 3. Programação linear. – Teses. 4. Minérios de Ferro - Misturas - Teses. 5. Minérios de Ferro – Planejamento de lavra – Teses. I. Universidade Federal de Ouro Preto. Escola de Minas. Departamento de Engenharia de Minas. Programa de Pós-graduação em Engenharia Mineral. II. Título.

CDU: 622.27

Catálogo: sisbin@sisbin.ufop.br

**PLANEJAMENTO OPERACIONAL DE LAVRA COM ALOCAÇÃO
DINÂMICA DE CAMINHÕES: ABORDAGENS EXATA E
HEURÍSTICA**

Francisco César Rodrigues de Araújo

Esta dissertação foi apresentada em sessão pública e aprovada em 27 de junho de 2008, pela Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Marcone Jamilson Freitas Souza (Orientador/UFOP)

Prof. Dr. Luiz Henrique de Campos Merschmann (UFOP)

Prof. Dr. Luiz Satoru Ochi (UFF)

Agradecimentos

A Deus, por tudo.

A meu orientador, Professor Marcone Jamilson Freitas Souza, minha referência não só pela capacidade intelectual incomparável, mas principalmente como pessoa, pela amizade e apoio demonstrados ao longo dos anos.

A meus pais pelo exemplo de vida.

Aos filhos, Marcella e Giuliano, pela compreensão e carinho que nunca faltaram.

À LÍlian, pelo incentivo.

Aos colegas professores Ronaldo Trindade e Adolfo Baudson, pela colaboração sempre que solicitados.

Às empresas mineradoras da região do quadrilátero ferrífero pelo suporte.

À FAPEMIG e ao CNPq pelo apoio ao desenvolvimento deste trabalho.

À UFOP e ao CEFET Ouro Preto pela oportunidade e por fazerem parte da minha vida.

A todos que, de alguma maneira, ajudaram-me a concluir esse trabalho.

Resumo

Este trabalho trata do planejamento operacional de lavra com alocação dinâmica de caminhões. Este problema consiste em determinar o número de viagens que cada caminhão deve fazer a cada frente de lavra, bem como decidir em quais frentes de lavra alocar as carregadeiras, de sorte a atender às metas de produção e qualidade requeridas para o minério a ser produzido, fazendo o melhor aproveitamento da frota de veículos disponível. Dada sua complexidade combinatória, o problema é abordado por um procedimento heurístico baseado na metaheurística *Iterated Local Search*. Também é formulado um modelo de programação matemática, o qual serve para validar o método heurístico proposto. Os modelos propostos são testados por meio da utilização de dados reais em um *software* desenvolvido para este fim.

Palavras-Chave: Planejamento de lavra em minas a céu aberto, Alocação Dinâmica de Caminhões, Planejamento da produção, *Iterated Local Search*, Programação Linear por Metas, Metaheurísticas.

Abstract

This work deals with the Operational Mining Planning in open pit mines with equipment dynamic allocation. This problem consists in determining the number of trips that each truck would do in each mining site as well as deciding in which mining site to allocate the loaders machines, in order to attend the production and quality goals required to the ore production, doing the best utilization of the available vehicle fleet. Due to the combinatorial complexity of the problem, it is solved using a heuristic procedure based on the Iterated Local Search metaheuristic. Also it is proposed a model of mathematical programming, which serves to validate the proposed metaheuristic method. The proposed models are validated by the utilization of real data in a software developed for this purpose.

Keywords: Operational Mining Planning in open pit mines, Equipment Dynamic Allocation, Production planning, Iterated Local Search, Goal Programming, Metaheuristics.

Sumário

AGRADECIMENTOS	III
RESUMO	IV
ABSTRACT.....	V
SUMÁRIO	VI
LISTA DE FIGURAS	VIII
LISTA DE TABELAS	X
LISTA DE SIGLAS E ABREVIATURAS.....	XI
1. PRELIMINARES	1
1.1 INTRODUÇÃO	1
1.2 JUSTIFICATIVA	2
1.3 ESTRUTURA DO TRABALHO	3
2. PROBLEMA DE PLANEJAMENTO DE LAVRA COM ALOCAÇÃO DINÂMICA DE CAMINHÕES	4
2.1 PLANEJAMENTO DE LAVRA.....	4
2.1.1 <i>Introdução</i>	4
2.1.2 <i>Descrição do Planejamento</i>	5
2.2 ALOCAÇÃO DE CAMINHÕES.....	12
2.2.1 <i>Alocação Estática de Caminhões</i>	14
2.2.2 <i>Alocação Dinâmica de Caminhões</i>	15
2.3 CARACTERÍSTICAS DO PROBLEMA DE ALOCAÇÃO ABORDADO.....	16
3. REVISÃO BIBLIOGRÁFICA	18
3.1 INTRODUÇÃO	18
3.2 PROGRAMAÇÃO POR METAS	22
3.3 O MODELO DE ALOCAÇÃO DINÂMICA DE COSTA <i>ET AL.</i> (2004)	25
3.4 HEURÍSTICAS	29
3.4.1 <i>Introdução</i>	29
3.4.2 <i>Metaheurísticas</i>	31
4. MODELAGEM DO PROBLEMA ABORDADO	35
4.1 INTRODUÇÃO	35
4.2 MODELO DE PROGRAMAÇÃO MATEMÁTICA	35
4.3 MODELO HEURÍSTICO	40
4.3.1 <i>Dados de Entrada</i>	40
4.3.2 <i>Representação de uma Solução</i>	40

4.3.3	<i>Geração de uma Solução Inicial</i>	42
4.3.4	<i>Avaliação de uma Solução.....</i>	48
4.3.5	<i>Movimentos e Estruturas de Vizinhanças.....</i>	52
4.3.6	<i>Métodos de Busca Local Aplicados ao Planejamento de Lavra</i>	59
4.3.7	<i>Perturbações.....</i>	67
4.3.8	<i>Iterated Local Search (ILS)</i>	81
4.4	IMPLEMENTAÇÃO COMPUTACIONAL	88
4.4.1	<i>Tela de abertura.....</i>	88
4.4.2	<i>Tela principal do sistema.....</i>	89
4.4.3	<i>Tela de relatório das soluções inicial e final.....</i>	90
4.4.4	<i>Tela de relatório do número de viagens de cada caminhão.....</i>	91
4.4.5	<i>Tela do Tempo de Ciclo dos Caminhões</i>	92
4.4.6	<i>Tela de Compatibilidade entre Caminhões e Carregadeiras</i>	92
4.4.7	<i>Tela com a Capacidade Máxima de cada Caminhão.</i>	93
4.4.8	<i>Tela com a Capacidade Mínima de cada Carregadeira.</i>	94
4.4.9	<i>Tela com a Capacidade Máxima de cada Carregadeira.....</i>	94
5.	RESULTADOS COMPUTACIONAIS.....	96
5.1	INTRODUÇÃO	96
5.2	RESULTADOS OBTIDOS.....	99
5.3	DISCUSSÃO DOS RESULTADOS.....	106
6.	CONCLUSÕES E TRABALHOS FUTUROS.....	109
	REFERÊNCIAS BIBLIOGRÁFICAS.....	112
	ANEXO I: PROBLEMAS-TESTE.....	115
	ANEXO II: MODELO DE PROGRAMAÇÃO MATEMÁTICA	119
	ANEXO III: PUBLICAÇÕES	123

Lista de Figuras

Figura 2.1 – Modelo de Carregadeira (L1850)	6
Figura 2.2 – Modelo de Caminhão	7
Figura 2.3 – Pilha de Estoque	8
Figura 2.4 – Britador	9
Figura 2.5 – Correia transportadora	10
Figura 2.6 – Modelo de Caminhão Pipa	11
Figura 2.7 – Modelo de Motoniveladora	11
Figura 2.8 – Processo Produtivo de uma mineradora do Quadrilátero Ferrífero.....	12
Figura 2.9 – Equipamentos de Transporte e de Carga	13
Figura 2.10 – Alocação Estática de Caminhões	14
Figura 2.11 – Alocação Dinâmica de Caminhões.....	15
Figura 3.1 – Método da Descida	30
Figura 3.2 – Método Randômico de Descida	31
Figura 3.3 – Pseudocódigo para o procedimento <i>ILS</i>	33
Figura 3.4 – Pseudocódigo do algoritmo VND	34
Figura 4.1 – Representação de uma Solução	41
Figura 4.2 – Alocação Aleatória das Carregadeiras	43
Figura 4.3 – Número de Viagens de cada Caminhão às diversas Frentes	44
Figura 4.4 – Massa Lavrável em cada Frente	44
Figura 4.5 – Somatório dos Tempos de Ciclo por Caminhão.....	45
Figura 4.6 – Reparador de Inviabilidade do Número de Viagens	46
Figura 4.7 – Reparador de Inviabilidade de Massa Lavrável	47
Figura 4.8 – Movimento Retirar_uma_viagem.....	53
Figura 4.9 – Movimento Retirar_duas_viagens.....	53
Figura 4.10 – Movimento Realocar uma Viagem mantendo o caminhão	54
Figura 4.11 – Movimento Realocar uma Viagem mantendo a frente.....	55
Figura 4.12 – Movimento Realocar Carregadeira	55
Figura 4.13 – Movimento Ativar Carregadeira	56
Figura 4.14 – Movimento Desativar Carregadeira	57
Figura 4.15 – Movimento Inserir uma viagem	58
Figura 4.16 – Movimento Retirar Viagens	58
Figura 4.17 – Movimento Trocar Carregadeira	59
Figura 4.18 – Descida_tirar_uma_viagem.....	60
Figura 4.19 – Descida_tirar_duas_viagens.....	61
Figura 4.20 – Descida_realocação_de_viagem_mantendo_caminhão	62
Figura 4.21 – Descida_realocação_de_viagem_mantendo_frente	63
Figura 4.22 – Descida_inserir_uma_viagem	64
Figura 4.23 – Descida_tirar_viagens	65
Figura 4.24 – Descida_desativar_carregadeira	66
Figura 4.25 – Descida_realocar_carregadeira	67
Figura 4.26 – Perturbação_Retirar_uma_viagem	68
Figura 4.27 – Perturbação Retirar_duas_viagens	69

Figura 4.28 – Perturbação Realocar uma Viagem mantendo o caminhão.....	71
Figura 4.29 – Perturbação Realocar uma Viagem mantendo a frente	72
Figura 4.30 – Perturbação Realocar Carregadeira	74
Figura 4.31 – Perturbação Desativar Carregadeira.....	75
Figura 4.32 – Perturbação Ativar Carregadeira.....	76
Figura 4.33 – Perturbação Trocar Carregadeira.....	78
Figura 4.34 – Perturbação_tirar_viagens	79
Figura 4.35 – Perturbação_Retirar_uma_viagem	80
Figura 4.36 – <i>Iterated Local Search</i>	81
Figura 4.37 – Níveis de perturbação do ILS	82
Figura 4.38 – Perturbação desativar uma carregadeira.....	83
Figura 4.39 – Perturbação_n1	83
Figura 4.40 – Perturbação_n2	84
Figura 4.41 – Perturbação tirar todas as viagens	84
Figura 4.42 – Perturbação_n3.....	85
Figura 4.43 – Perturbação ativar uma carregadeira	86
Figura 4.44 – Perturbação trocar carregadeira ativa/inativa	86
Figura 4.45 – Algoritmo VND.....	87
Figura 4.46 – Tela de abertura do sistema	88
Figura 4.47 – Tela principal.....	89
Figura 4.48 – Tela das soluções inicial e final.....	90
Figura 4.49 – Tela do número de viagens de cada caminhão	91
Figura 4.50 – Tela do tempo de ciclo dos caminhões.....	92
Figura 4.51 – Tela de compatibilidade entre caminhões e carregadeiras	93
Figura 4.52 – Tela com a capacidade máxima de cada caminhão	93
Figura 4.53 – Tela com a capacidade mínima de cada carregadeira	94
Figura 4.54 – Tela com a capacidade máxima de cada carregadeira.....	95

Lista de Tabelas

Tabela 5.1 – Problemas-teste	96
Tabela 5.2 – Metas de produção e qualidade, número de equipamentos disponíveis e prioridades	97
Tabela 5.3 – Parâmetros utilizados	98
Tabela 5.4 – Características dos melhores resultados obtidos.....	99
Tabela 5.5 – Desvios percentuais dos parâmetros de controle da melhor solução em relação às metas estabelecidas	100
Tabela 5.6 – Valores médios obtidos.....	100
Tabela 5.7 – Desvios percentuais encontrados para algumas características das soluções.	101
Tabela 5.8 – Características das soluções do otimizador no tempo da melhor solução do ILS	102
Tabela 5.9 – Características das soluções do otimizador no tempo médio do ILS.....	102
Tabela 5.10 – Percentual de melhora do ILS em relação ao LINGO no tempo da melhor solução do ILS	103
Tabela 5.11 – Percentual de melhora do ILS em relação ao LINGO no tempo médio de processamento do ILS.....	104
Tabela 5.12 – Características das soluções obtidas pelo otimizador em uma hora de processamento.....	105
Tabela 5.13 – Características das soluções obtidas pelo otimizador em 20 horas de processamento.....	105
Tabela 5.14 – Comparação entre o LINGO e o melhor desempenho do ILS.....	106

Lista de Siglas e Abreviaturas

AG	Algoritmo Genético
ARENA	Software de Simulação
BT	Busca Tabu
CVRD	Companhia Vale do Rio Doce
CSN	Companhia Siderúrgica Nacional
GRASP	<i>Greedy Randomized Adaptive Search Procedures</i>
ILS	<i>Iterated Local Search</i>
LINGO	Modelador e Otimizador de Programação Linear e Inteira
NP	Não Polinomial
OTISIMIN	Otimizador e Simulador para Mineração
PCV	Problema do Caixeiro Viajante
PO	Pesquisa Operacional
SA	<i>Simulated Annealing</i>
VNS	<i>Variable Neighborhood Search</i>
VND	<i>Variable Neighborhood Descent</i>

1. Preliminares

1.1 Introdução

A Pesquisa Operacional (PO) é uma ciência aplicada voltada para a resolução de problemas reais envolvendo situações de tomada de decisão, através de modelos matemáticos habitualmente processados computacionalmente. Ela aplica conceitos e métodos de outras disciplinas científicas na concepção, no planejamento ou na operação de sistemas para atingir seus objetivos. Procura, assim, introduzir elementos de objetividade e racionalidade nos processos de tomada de decisão, sem descuidar, no entanto, dos elementos subjetivos e de enquadramento organizacional que caracterizam os problemas.

O termo Pesquisa Operacional, do inglês *Operations Research*, foi empregado pela primeira vez em 1939 como uma tentativa de englobar, sob uma única denominação, todas as técnicas existentes ou que viriam a ser desenvolvidas e que tinham o mesmo objetivo citado.

De uma maneira geral, todas as disciplinas que constituem a PO se apóiam em quatro ciências fundamentais: Economia, Matemática, Estatística e Informática.

O objetivo principal da Pesquisa Operacional é encontrar a melhor utilização de recursos limitados procurando determinar o uso otimizado de atividades ou recursos, fornecendo um conjunto de procedimentos e métodos quantitativos para tratar de forma sistêmica problemas que envolvam a utilização de recursos escassos.

Dentre as aplicações da PO, inclui-se sua utilização em planejamento de lavra a céu aberto, em especial, na alocação de máquinas e caminhões às frentes de lavra.

Estudos como os de Chanda e Dagdelen (1995), Alvarenga (1997), Pinto e Merschmann (2001), Merschmann (2002), Pinto *et al.* (2003), Costa *et al.* (2004) e Costa (2005) têm utilizado a Pesquisa Operacional com o objetivo de aplicar metodologias para a solução de problemas inerentes ao planejamento de lavra. Tais esforços se concentraram na aplicação de técnicas de otimização (programação linear e métodos heurísticos).

Nesta investigação trata-se o problema de planejamento operacional de lavra com alocação dinâmica de caminhões. O objetivo é aperfeiçoar o modelo de programação matemática de Costa *et al.* (2004), incluindo uma nova restrição relativa à utilização dos caminhões, bem como apresentar um novo modelo heurístico.

A utilização de técnicas heurísticas para resolver este problema se justifica face à sua natureza combinatória. Este fato torna proibitiva sua resolução por técnicas exatas quando o número de variáveis é elevado e o tempo para a tomada de decisão é baixo. O modelo heurístico proposto é baseado na metaheurística *Iterated Local Search* (LOURENÇO, MARTIN e STÜTZLE, 2003). Apesar de estas técnicas heurísticas não garantirem a otimalidade da solução, elas conseguem, em geral, produzir soluções de boa qualidade rapidamente. Dentre as heurísticas, destacam-se as chamadas metaheurísticas, as quais, ao contrário das heurísticas convencionais, têm caráter geral e são providas de mecanismos para tentar escapar de ótimos locais, ainda distantes de um ótimo global (SOUZA, 2007). O modelo heurístico proposto será validado pelo modelo de programação matemática.

1.2 Justificativa

As soluções para o problema de planejamento de lavra com alocação dinâmica de caminhões podem ser encontradas utilizando diversos métodos. O primeiro deles, ainda utilizado em algumas empresas de mineração, é o manual. É uma tarefa árdua, dado o elevado número de combinações a testar e o pouco tempo que se tem para a tomada de decisão. Outra forma de se encontrar uma solução para este problema é formulá-lo como um modelo de programação matemática e resolvê-lo por um software de otimização, como por exemplo, o LINGO, o CPLEX e o XPRESS. O inconveniente neste caso é o alto custo associado à utilização desta metodologia. Além disso, o problema é da classe NP-difícil, o que significa que esta metodologia pode não conseguir resolver em tempo hábil casos reais deste problema.

O método proposto e modelado no capítulo 4 é uma alternativa aos métodos atualmente utilizados na alocação dinâmica de caminhões em lavra de mina a céu aberto. Em vista do potencial das metaheurísticas e a facilidade de sua implementação, a

expectativa é que soluções de melhor qualidade sejam produzidas a um custo de desenvolvimento mais baixo.

1.3 Estrutura do Trabalho

Este trabalho está organizado como segue. No presente capítulo foi contextualizada a pesquisa operacional nas aplicações em empresas de mineração e apresentada a justificativa do presente trabalho.

No capítulo 2 é descrito em detalhes o problema abordado.

No capítulo 3 é feita uma revisão bibliográfica de aplicações de pesquisa operacional na mineração, bem como de técnicas de programação matemática e de técnicas heurísticas.

No capítulo 4 é apresentada uma modelagem de programação matemática para o problema abordado e, ainda, uma modelagem heurística e sua implementação computacional.

No capítulo 5 são apresentados, discutidos e analisados os resultados obtidos pela aplicação do modelo proposto a um conjunto de diferentes cenários.

O capítulo 6 conclui o trabalho e apresenta propostas para trabalhos futuros.

2. Problema de Planejamento de Lavra com Alocação Dinâmica de Caminhões

2.1 Planejamento de Lavra

2.1.1 Introdução

Em praticamente todas as empresas de mineração, principalmente as de médio e grande porte, há a necessidade de se fazer um planejamento estratégico de retirada (exploração) de minério para atender às necessidades do mercado. Isto se deve ao fato de que nessas situações o ser humano não consegue analisar todos os cenários possíveis e detectar as melhores opções em tempos compatíveis.

Para executar essa tarefa, conhecida como Planejamento de Lavra, é necessário se conhecer com riqueza de detalhes a área a ser explorada e explotada.

Normalmente, um planejamento é elaborado considerando-se três fases: longo, médio e curto prazo. Na primeira fase, a de longo prazo, faz-se um planejamento mais amplo. Por exemplo, pensa-se na mina para alguns anos. Na segunda, a de médio prazo, faz-se um planejamento para até um ano e na terceira fase, a de curto prazo, trabalha-se num planejamento diário, mensal e até trimestral. É importante salientar que esses intervalos de tempo não são rígidos e, portanto, podem variar de uma empresa para outra.

O planejamento de curto prazo é normalmente realizado por profissionais de Geologia, Engenharia de Minas, Qualidade, Produção e Administração. Esta equipe se interage com o objetivo final de atender ao cliente e é a mais exigida tendo em vista a necessidade de se cumprir tudo aquilo que fora pré-estabelecido pelas equipes responsáveis pelo longo e médio prazo.

Mostra-se, a seguir, o processo produtivo de uma empresa mineradora de ferro, situada no Quadrilátero Ferrífero, na região central do Estado de Minas Gerais.

2.1.2 Descrição do Planejamento

De posse dos dados relativos à quantidade e qualidade do minério disponível na mina, pode-se falar em vida útil desta. Essa é uma tarefa das equipes de médio e longo prazo, que objetivam fazer o planejamento para atender ao mercado por um dado período. Evidentemente que, em função do avanço da tecnologia, esse tempo pode variar para mais ou para menos. Material antes considerado estéril poderá, no futuro, ser economicamente aproveitado como minério. Em contra-partida, equipamentos de melhor qualidade e *softwares* mais eficazes poderão fazer com que a mina seja exaurida mais rapidamente.

De posse dessas informações, a equipe de curto prazo da empresa analisada planeja como atender às necessidades por um período de até 3 meses. O planejamento é feito para o dia, o mês e o trimestre.

Com a ajuda de *softwares* elaborados para este fim, a equipe do planejamento determina aos diversos setores envolvidos (Qualidade, Sistema de Despacho e Produção) quais são as frentes a serem lavradas bem como quais e quantos equipamentos serão envolvidos (operação de alocação de máquinas e caminhões).

O objetivo é elaborar um plano diário de tal forma que ao final do mês e também do trimestre, as metas sejam atingidas tanto em qualidade quanto em quantidade.

Após se escolher as frentes de lavra, os equipamentos de carga e de transporte, inicia-se todo o processo que começa na mina com a extração do material (minério e estéril), passando pelo(s) britador(es) e terminando nas pilhas de homogeneização.

O processo produtivo dessa empresa mineradora se desenvolve a partir da execução das ações elaboradas por uma equipe formada por profissionais de diversas áreas.

Os operadores, com a ajuda de *softwares*, repassam as informações à equipe da produção (técnicos, operadores das máquinas, operadores dos caminhões). Esses, por sua vez, fazem o minério chegar ao(s) britador(es) ou às pilhas de estoque.

Os equipamentos de carga e descarga trabalham sob orientação da equipe responsável pelo sistema de despacho.

A Figura 2.1 mostra um tipo de equipamento utilizado para o carregamento de minério e estéril da mina.



Figura 2.1 – Modelo de Carregadeira (L1850)

A partir do carregamento, o condutor do veículo responsável pelo transporte do material recebe instruções da equipe de despacho para fazer o descarregamento.

A Figura 2.2 mostra um tipo de equipamento utilizado para transporte de minério e estéril.



Figura 2.2 – Modelo de Caminhão

Esse material pode ser transportado diretamente até ao britador ou depositado em pilhas denominadas de pilhas de estoque.

A Figura 2.3 mostra uma pilha de estoque sendo formada.



Figura 2.3 – Pilha de Estoque

A Figura 2.4 mostra o minério sendo depositado diretamente no britador.



Figura 2.4 – Britador

Após passar pelo(s) britador(es), o minério é transportado por caminhões ou correias transportadoras até as pilhas de homogeneização.

A Figura 2.5 mostra o minério sendo conduzido por correias transportadoras até às pilhas de homogeneização.



Figura 2.5 – Correia transportadora

A todo momento, equipamentos responsáveis pela manutenção das vias (estradas) de acesso ao britador e às pilhas de homogeneização (caminhões pipa e motoniveladora, por exemplo) entram em ação para minimizar os problemas causados pela poeira e pelos “borrachudos”, denominação dada às saliências no piso ocasionadas pelo intenso tráfego de máquinas de grande porte nas estradas.

A Figura 2.6 mostra um equipamento responsável pela minimização da poeira e a Figura 2.7 mostra uma motoniveladora, responsável pela correção do solo.



Figura 2.6 – Modelo de Caminhão Pipa



Figura 2.7 – Modelo de Motoniveladora



Figura 2.9 – Equipamentos de Transporte e de Carga

Para satisfazer as especificações exigidas pelos clientes, é necessário selecionar as frentes a serem lavradas e determinar seu ritmo de lavra. Para a operação de minério e estéril, a mina conta com uma frota limitada de equipamentos de carga, os quais devem ser alocados às frentes de lavra e operarem em uma faixa de produtividade que torne viável sua utilização (COSTA, 2005).

O transporte do material retirado da frente de lavra é realizado por uma frota de caminhões com capacidades de carga diferentes. O ritmo de lavra é determinado pelas capacidades de operação dos equipamentos de carga e transporte alocados às diversas frentes.

Em minas a céu aberto, são utilizados dois critérios para a alocação de caminhões: alocação estática e alocação dinâmica.

2.2.1 Alocação Estática de Caminhões

O problema da mistura de minério com alocação estática de caminhões é geralmente adotado devido à simplificação das operações e ao custo relativo à implantação de um sistema de despacho computadorizado de caminhões. Na alocação estática (Figura 2.10), um caminhão é alocado a uma única rota, ou seja, permanece se deslocando entre dois pontos fixos, um de basculamento e outro de carga, onde esteja alocado um equipamento de carga compatível. Desta forma, o ritmo de lavra de uma frente dependerá da capacidade de produção dos caminhões e do equipamento de carga alocados à frente.

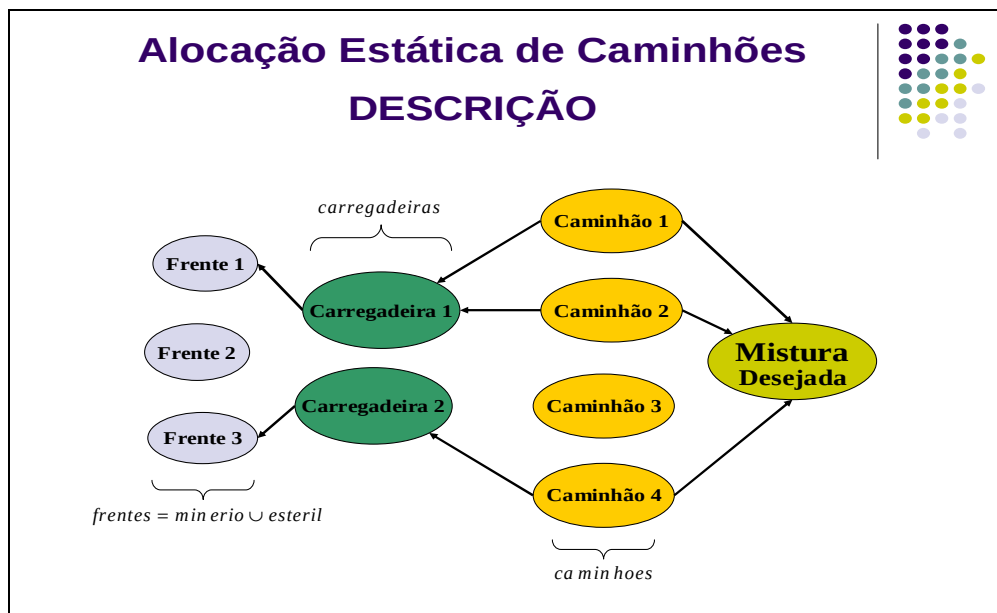


Figura 2.10 – Alocação Estática de Caminhões

2.2.2 Alocação Dinâmica de Caminhões

No sistema de alocação dinâmica (Figura 2.11), os caminhões não ficam fixos a uma determinada frente de lavra, como no modelo estático. Eles podem ser direcionados a diferentes frentes de lavra compatíveis, aumentando a produtividade da frota e proporcionando, segundo Costa (2005), um aumento na capacidade de produção da mina ou a redução do número de equipamentos necessários para manter um determinado nível de produção.

Como desvantagens da alocação dinâmica, apontam-se exigência de um maior número de operações e de um sistema de despacho de caminhões computadorizado.

Nesse sistema é também possível diminuir o tempo de fila, uma vez que o caminhão pode ser alocado a pontos de carga diferentes.

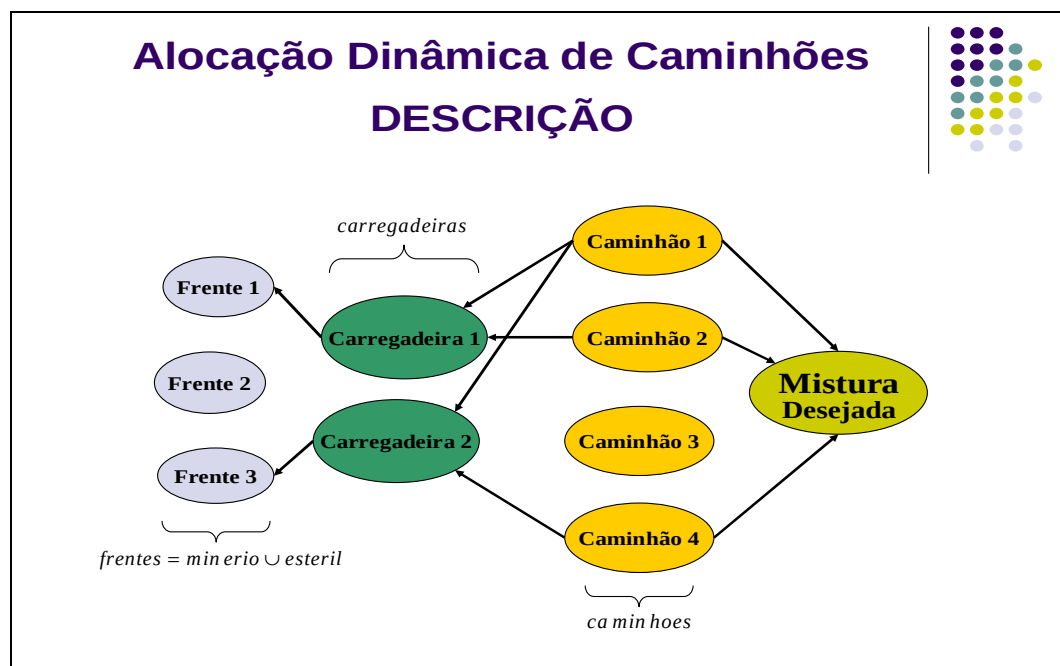


Figura 2.11 – Alocação Dinâmica de Caminhões

2.3 Características do Problema de Alocação Abordado

O problema abordado é o de Planejamento Operacional de Lavra com alocação dinâmica de caminhões, onde há dois pontos de descarga: um para minério e outro para estéril.

Na realidade, apesar de ser considerado um único ponto de descarga para o minério, que normalmente acontece no britador, é possível que esta seja feita em um ponto próximo ao britador. Isto é, considera-se que a distância das frentes ao britador e seus pontos próximos é a mesma. Este procedimento é normalmente admitido nas minerações para evitar fila na descarga junto ao britador. O caminhão encontrando fila no britador descarrega em uma pilha em um ponto próximo do britador e, posteriormente, esse material é retomado para descarga no britador.

Considera-se também que o transporte do material às frentes e ao britador é feito somente por caminhões, no caso de capacidades diferentes.

Sendo a alocação dinâmica, ao descarregar o material, seja no britador (ou pilhas de estoque próximas ao britador) ou na pilha de estéril, o caminhão é direcionado a uma frente, não necessariamente a mesma da viagem anterior.

Admite-se, também, que há um conjunto de carregadeiras de diferentes capacidades de carregamento, sendo este conjunto menor que o conjunto das frentes às quais elas serão alocadas.

Considera-se, também, o planejamento para uma hora de produção, o qual é aplicado até uma frente exaurir ou ocorrer uma parada de equipamento, seja por quebra ou manutenção, situação na qual deve ser feito outro planejamento.

Dado o elevado custo de uma carregadeira, normalmente impõe-se uma quantidade mínima a ser extraída para justificar economicamente sua utilização.

É desejável no planejamento de lavra considerar, também, uma taxa máxima de utilização para os caminhões, ou seja, pode-se desejar que cada caminhão utilizado no transporte de minério e estéril tenha uma produtividade máxima. Por exemplo, supondo uma taxa máxima de utilização de 85%, um caminhão (*I*), com capacidade de 80 t, deveria trabalhar 51 ($0,85 \times 60$) minutos, no máximo, em uma hora. Supondo que o

tempo médio de ciclo (TMC) deste caminhão (I) às diversas frentes seja de 7 minutos por viagem, o número de viagens que esse caminhão conseguiria fazer em 51 minutos é 7. Transformando em toneladas, esse caminhão poderia transportar cerca de 560 t em uma hora (7×80), caso atingisse os 85% de desempenho. Com a introdução desta restrição, o número de caminhões utilizados no transporte do material poderá ser, naturalmente, menor do que aquele conseguido sem a imposição desta taxa.

3. Revisão Bibliográfica

3.1 Introdução

A Pesquisa Operacional tem sido utilizada ultimamente de modo eficaz na resolução de problemas de planejamento de lavra a céu aberto utilizando-se técnicas de otimização, dentre elas a programação linear e os métodos heurísticos.

Mutmansky (1979), apud Costa (2005), relatou que a programação linear é a técnica de pesquisa operacional mais aplicada aos problemas de planejamento de produção em mineração, sendo adotada principalmente pelas minerações a céu aberto, devido à maior complexidade de suas operações em relação às de minas subterrâneas.

White *et al.* (1982), apud Rodrigues (2006), propõem um modelo baseado em programação matemática que minimiza o número de caminhões necessários através de restrições relacionadas à continuidade do fluxo de material pelos pontos de carga e descarga e as capacidades de produção dos pontos de carga.

White e Olson (1986), apud Rodrigues (2006), propuseram um algoritmo que é a base para o sistema *Dispatch*, que vem operando em muitas minas. Esse modelo utiliza programação linear e dinâmica. A programação linear é dividida em duas partes. Na primeira parte do modelo, realiza-se a otimização do problema da mistura conseguindo, através disso, a minimização do remanuseio, forçando uma solução próxima das restrições críticas de blendagem (mistura) e satisfazendo a necessidade de alimentação da planta. A segunda parte do modelo otimiza o transporte na mina por meio da minimização da necessidade de transporte, gerando o fluxo ótimo de transporte no caminho. A programação dinâmica usa o fluxo ótimo no caminho obtido na programação linear, uma lista formada pelos caminhões a serem alocados, os tempos de viagem e as distâncias atuais entre os pontos de carga e descarga e é executada a cada necessidade de alocação de caminhão. Mais especificamente, para cada caminhão a ser alocado, determina-se uma medida de tempo para cada caminho existente na mina e aloca-se o caminhão ao caminho que apresentar o menor valor.

Segundo Munirathinam e Yingling (1994), vários modelos propostos na literatura, entre os quais os de White e Olson (1986), ignoram a natureza estocástica das operações de transporte, carregamento e descarregamento, os quais tendem a introduzir aumento no tempo de espera, em virtude do aumento do número de caminhões alocados a um determinado equipamento de carga.

Chanda & Dagdelen (1995) apresentam um modelo de programação linear por metas aplicado à resolução de um problema de blendagem de minérios no planejamento de curto prazo em uma mina de carvão. O objetivo é maximizar um critério econômico e minimizar a soma dos desvios dos parâmetros individuais de qualidade e da tonelagem em relação aos valores desejados. Os modelos são analisados, comparando-os com os pertencentes a um sistema baseado em gráficos interativos. Melhorias na qualidade dos resultados de planejamento podem ser obtidas, segundo alguns estudos, através da combinação dessas duas técnicas.

Merschmann (2002) desenvolveu um sistema de otimização e simulação para análise de cenário de produção em minas a céu aberto. O sistema denominado OTISIMIN (Otimizador e Simulador para Mineração) foi desenvolvido em dois módulos. O primeiro corresponde ao módulo de otimização onde um modelo de programação linear foi construído e resolvido e o segundo a um módulo de simulação que permite ao usuário utilizar os resultados obtidos na resolução do modelo de programação linear como dados de entrada para a simulação. O módulo de otimização foi elaborado com o objetivo de otimizar o processo de mistura de minérios oriundos das várias frentes de lavra de forma a atender as especificações de qualidade impostas pela usina de tratamento e realizar a alocação de equipamentos (caminhões, carregadeiras e/ou escavadeiras) às frentes de lavra, considerando tanto alocação dinâmica quanto estática dos caminhões.

Os modelos de otimização de Merschmann (2002) foram aperfeiçoados por Costa *et al.* (2004, 2005). Nesses trabalhos, são apresentadas modelagens de programação matemática que reduzem significativamente o número de restrições. Além disso, os modelos incluem o atendimento às metas de produção e qualidade por meio da minimização, na função objetivo, dos desvios de atendimento às restrições de meta.

Moraes (2005) tratou o problema de blendagem de produtos de minério de ferro, estocados nos pátios da mina Cauê, da empresa Companhia Vale do Rio Doce (CVRD), em Minas Gerais, visando a composição de lotes. É proposto um modelo de programação linear por metas que visa determinar os locais de retomada do produto estocado, de tal forma que a mistura atenda aos limites de especificações de qualidade e quantidade preestabelecidos pelo cliente e satisfaça as restrições operacionais do pátio. O modelo de programação matemática desenvolvido foi implementado no modelador e otimizador LINGO 9.0, interfaceando com planilhas do EXCEL 2000, o que possibilitou a utilização e exportação de dados em um ambiente familiar à empresa de mineração. O sistema desenvolvido foi validado comparando-se os resultados obtidos com os produzidos manualmente pela empresa. Os resultados computacionais apresentados comprovaram que é possível prover uma melhora na composição dos lotes através do modelo proposto.

Muitos problemas de mineração, como o de planejamento operacional de lavra com alocação dinâmica de caminhões, em geral, somente podem ser resolvidos na otimalidade, com o uso de métodos de programação matemática, quando o número de variáveis do modelo é pequeno. Isto se deve ao fato de que este problema é de natureza combinatória, pertencente à classe NP-difícil (COSTA, 2005). Assim, a utilização de métodos de programação matemática para resolver problemas deste tipo não é indicada para a maioria dos casos reais, que envolvem um número elevado de variáveis. Além do mais, nestas situações, métodos de programação matemática podem não encontrar soluções viáveis em tempos computacionais aceitáveis. Uma alternativa comumente adotada para resolução de problemas combinatórios é a utilização de heurísticas. Estas técnicas não garantem a otimalidade da solução produzida para o problema; no entanto, produzem resultados muito bons a um custo computacional baixo devido à sua fácil implementação e a uma maior flexibilidade na modelagem das restrições em comparação aos métodos exatos. Assim, vêm conquistando cada vez mais espaço na resolução de problemas de planejamento de produção em mineração.

Ezawa e Silva (1995), apud Rodrigues (2006), desenvolveram um sistema de alocação dinâmica de caminhões às frentes de lavra com o objetivo de reduzir a variabilidade dos teores do minério e aumentar a produtividade do sistema de

transporte. O sistema utiliza uma heurística para gerar as decisões que têm como resultado um seqüenciamento de caminhões que minimiza a variação dos teores. Para validar esse modelo foi utilizado simulação e para o modelamento matemático da mina a teoria dos grafos. Com a implantação do sistema, foi obtido um aumento de 8% na produtividade do sistema de transporte e o gerenciamento passou a ter em mãos dados mais precisos e em tempo real.

Alvarenga (1997) desenvolveu um programa para o despacho ótimo de caminhões em uma mineração de ferro, a céu aberto, com o objetivo de minimizar o tempo de filas para as frotas de caminhões, aumentar a produtividade da frota e melhorar a qualidade do minério lavrado com relação ao atendimento da qualidade requerida. No trabalho desenvolvido, que é base do sistema SMART MINE, foi aplicada uma técnica estocástica de otimização, o algoritmo genético com processamento paralelo, em um caso real da Mineração Casa de Pedra, uma empresa do grupo CSN – Companhia Siderúrgica Nacional. O problema se resume em sugerir qual o ponto de basculamento ou carregamento ideal e a trajetória para o deslocamento, no momento em que há uma situação de escolha a ser realizada. Enquanto os ganhos com a qualidade foram mais difíceis de serem quantificados, os resultados numéricos mostraram ganhos de 5% a 15% de produtividade, comprovando a validade da proposta.

Costa (2005) aborda problemas de planejamento operacional de lavra em minas a céu aberto. São apresentados e modelados problemas relativos à mistura de minérios provenientes de várias frentes de lavra, levando-se em consideração metas de produção e qualidade, restrições operacionais e a alocação dos equipamentos de carga e transporte necessários ao processo. São considerados três problemas: mistura de minérios, mistura de minérios com alocação dinâmica de caminhões e mistura de minérios com alocação estática de caminhões. Cada problema é resolvido por duas metodologias, uma baseada em técnicas de programação matemática e outra em técnicas heurísticas. Observa-se que o modelo de programação matemática, no entanto, não considera taxa de utilização de caminhões. Dada a NP-completude do problema de mistura de minérios com alocação dinâmica e/ou estática de caminhões, foi utilizada a metaheurística *Variable Neighborhood Search* (VNS), a qual permite escapar de ótimos locais ainda distantes do ótimo global. Para explorar o espaço de soluções, o autor utiliza seis tipos de

movimentos: a) realocar equipamentos de carga; b) retomar ou parar a operação em uma frente; c) acrescentar ou diminuir o número de viagens de um caminhão em uma frente; d) realocar o número de viagens de um caminhão em frentes diferentes; e) realocar o número de viagens realizadas em uma frente para caminhões diferentes; f) retomar ou parar a operação de um caminhão em uma frente. Experimentos computacionais realizados com dados reais e instâncias hipotéticas baseadas nesses dados validam os modelos propostos. Pelos experimentos realizados, o modelo heurístico é capaz de gerar soluções de melhor qualidade em um menor tempo de processamento que aquelas obtidas pelo modelo de programação matemática.

Rodrigues (2006) fez uma análise comparativa de várias metodologias utilizadas para o despacho de caminhões em minas a céu aberto. As metodologias testadas, baseadas em Programação Linear, Programação Dinâmica e em Heurísticas, são aquelas consideradas as bases de algoritmos utilizados em sistemas de despacho comercializados no Brasil. As soluções finais geradas por essas metodologias foram simuladas usando-se o ambiente ARENA, com o objetivo de reproduzir o comportamento das operações de lavra. Os resultados obtidos mostraram o desempenho dos algoritmos utilizando as metodologias sob diferentes condições em minas a céu aberto. A autora, porém, afirma que não se pode, através dos resultados obtidos, concluir pela superioridade de uma delas.

A seguir, é feita uma revisão da programação linear por metas, do método de Costa *et al.* (2004), bem como das heurísticas utilizadas neste trabalho.

3.2 Programação por Metas

Programação por metas (*Goal Programming*) é um método para se resolver um problema de otimização multiobjetivo.

Segundo Arenales *et al* (2007), a programação por metas estabelece *a priori* um conjunto de valores (metas) aceitáveis para cada função objetivo, para encontrar uma solução de compromisso entre os objetivos conflitantes.

Considere o seguinte problema de otimização linear multiobjetivo (por simplicidade, serão considerados três objetivos):

$$\text{Maximizar } f_1(x) = c^T x$$

$$\text{Minimizar } f_2(x) = d^T x$$

$$\text{Minimizar } f_3(x) = e^T x$$

$$Ax = b$$

$$x \geq 0,$$

ou seja, buscam-se soluções viáveis que façam com que o valor de $f_1(x)$ seja o maior possível e $f_2(x)$ e $f_3(x)$ sejam pequenos. Supõe-se que o decisor (pessoa responsável pela implementação da solução) fique satisfeito com qualquer solução que atenda:

$$f_1(x) \geq \sigma_1, \text{ sendo } \sigma_1 \text{ o valor mínimo para } f_1(x) \text{ que se deseja maximizar,}$$

$$f_2(x) \leq \sigma_2, \text{ sendo } \sigma_2 \text{ o valor máximo para } f_2(x) \text{ que se deseja minimizar,}$$

$$f_3(x) = \sigma_3, \text{ sendo } \sigma_3 \text{ o valor ideal para } f_3(x),$$

Nas relações acima, σ_i são chamadas *metas*. As desigualdades e a igualdade anteriores são definidas pelo decisor. Desta forma, o problema de otimização linear é reformulado como um problema de encontrar uma solução factível para o sistema:

$$f_1(x) \geq \sigma_1$$

$$f_2(x) \leq \sigma_2$$

$$f_3(x) = \sigma_3$$

$$Ax = b$$

$$x \geq 0.$$

Tal problema, entretanto, pode ser infactível, indicando que o decisor deve abrir mão de uma ou mais metas. Supõe-se que as restrições originais do problema $Ax = b$, $x \geq 0$, gerem soluções factíveis, caso contrário o problema multiobjetivo não tem qualquer solução. Para formalizar essa flexibilização das restrições-meta, são introduzidas novas variáveis nas restrições-meta:

$$f_1(x) + y_1 \geq \sigma_1$$

$$f_2(x) - y_2 \leq \sigma_2$$

$$f_3(x) + y_3^+ - y_3^- = \sigma_3,$$

em que $y_1 \geq 0$, $y_2 \geq 0$, $y_3^+ \geq 0$, $y_3^- \geq 0$. Tais variáveis são chamadas variáveis de desvio porque representam os desvios acima ou abaixo dos valores das metas. Por exemplo, se uma solução factível x é tal que $f_1(x) < \sigma_1$, então, com $y_1 = \sigma_1 - f_1(x) > 0$, a restrição-meta flexibilizada é satisfeita.

Dois métodos são comumente utilizados para otimizar um modelo de programação de metas: (i) o *método dos pesos* e (ii) o *método hierárquico* (em inglês, *preemptive*). Ambos baseiam-se na estratégia de converter múltiplos objetivos em um único objetivo. O método dos pesos considera que as soluções de interesse do decisor são aquelas em que as variáveis de desvio ou flexibilização assumem valores o mais próximo possível de zero. Portanto, o seguinte problema deve ser resolvido:

$$\text{Minimizar } \varphi(x,y) = w_1 y_1 + w_2 y_2 + w_3 (y_3^+ + y_3^-)$$

$$f_1(x) + y_1 \geq \sigma_1$$

$$f_2(x) - y_2 \leq \sigma_2$$

$$f_3(x) + y_3^+ - y_3^- = \sigma_3$$

$$Ax = b$$

$$x \geq 0, y_1 \geq 0, y_2 \geq 0, y_3^+ \geq 0, y_3^- \geq 0,$$

em que $w_1 + w_2 + w_3$ são os pesos (positivos) das metas, respectivamente, representando a importância relativa entre elas para o decisor. Por exemplo, se $w_i = 1$, $i = 1, 2, 3$, então todas as metas têm a mesma importância para o decisor, ou se $w_1 > w_2 > w_3$ então a meta 1 é mais importante que a meta 2, e esta é mais importante que meta 3.

No método hierárquico, os objetivos são tomados em ordem de prioridade, um após o outro. Por exemplo, suponha que o objetivo 1 é o mais importante para o decisor, o objetivo 2 é o segundo mais importante, o objetivo 3 é o terceiro mais importante, e assim por diante. Então, primeiro minimiza-se o desvio y_1 do primeiro objetivo. A seguir, minimiza-se o desvio y_2 do segundo objetivo, sujeito à condição de que o desvio

do primeiro objetivo seja mínimo (isto é, o valor y_1 é fixado no valor mínimo obtido anteriormente). Posteriormente, minimiza-se o desvio y_3 do terceiro objetivo, sujeito à condição de que os desvios do primeiro e do segundo objetivos sejam mínimos (isto é, os valores y_1 e y_2 são fixados nos valores mínimos já obtidos anteriormente), e assim por diante.

Os métodos de programação de metas procuram uma solução que satisfaça as metas do problema sem se preocupar com a otimização do problema. Em outras palavras, se houver várias soluções que satisfaçam todas as metas, esses métodos não determinam a melhor delas do ponto de vista da otimização dos objetivos originais do problema, f_1, f_2, f_3 . Isso pode ser contornado adicionando-se às funções objetivos dos modelos de programação de metas, múltiplos positivos e pequenos dos objetivos originais a serem minimizados, ou subtraindo-se múltiplos positivos e pequenos, se originalmente desejava-se maximizar. Por exemplo, no método dos pesos, é conveniente considerar: minimizar $\varphi(x,y) = w_1y_1 + w_2y_2 + w_3(y_3^+ + y_3^-) - \epsilon_1f_1(x) + \epsilon_2f_2(x) + \epsilon_3f_3(x)$. A programação de metas também pode ser aplicada a problemas de otimização discreta e otimização não-linear. No caso de otimização linear, o método Simplex pode ser especializado para resolver os modelos de programação de metas.

3.3 O modelo de Alocação Dinâmica de Costa *et al.* (2004)

Costa *et al.* (2004) utilizou a técnica de pesquisa operacional conhecida como programação por metas (*Goal Programming*), para modelar a alocação dinâmica de caminhões no planejamento de lavra.

No modelo destes autores o objetivo é minimizar os desvios das metas de produção, bem como os desvios das metas de qualidade. O modelo, apresentado pelas equações (3.1)-(3.22), considera os seguintes dados de entrada:

M : Conjunto de frentes de minério;

E : Conjunto de frentes de estéril;

F : Conjunto de frentes formado por $M \cup E$;
 T : Conjunto de parâmetros de qualidade analisados no minério;
 C : Conjunto de equipamentos de carga;
 V : Conjunto de equipamentos de transporte;
 Pr : Ritmo de lavra recomendado (t/h);
 Pl : Ritmo de lavra mínimo (t/h);
 Pu : Ritmo de lavra máximo (t/h);
 α^- : Penalidade por desvio negativo da produção;
 α^+ : Penalidade por desvio positivo da produção;
 t_{ij} : Valor do parâmetro de qualidade j na frente i (%);
 tr_j : Valor recomendado para o parâmetro de qualidade j na mistura (%);
 tl_j : Valor mínimo admissível para o parâmetro de qualidade j na mistura (%);
 tu_j : Valor máximo admissível para o parâmetro de qualidade j na mistura (%);
 β^- : Penalidade por desvio negativo para o parâmetro de qualidade j na mistura;
 β^+ : Penalidade por desvio positivo para o parâmetro de qualidade j na mistura;
 Ql_i : Ritmo de lavra mínimo para a frente i (t/h);
 Qu_i : Ritmo de lavra máximo para a frente i (t/h);
 rem : Relação estéril/minério requerida;
 Cl_k : Produção mínima do equipamento de carga k (t/h);
 Cu_k : Produção máxima do equipamento de carga k (t/h);
 cap_l : Capacidade do caminhão l (t);
 tc_{il} : Tempo total de ciclo do caminhão l na frente i (min);

$$g_{lk} = \begin{cases} 1 & \text{se o caminhão } l \text{ é compatível com o equipamento de carga } k, \\ 0 & \text{caso contrário} \end{cases}$$

e as seguintes variáveis de decisão:

x_i : Ritmo de lavra da frente i (t/h);

$$y_{ik} = \begin{cases} 1 & \text{se o equipamento de carga } k \text{ opera na frente } i; \\ 0 & \text{caso contrário} \end{cases}$$

n_{il} : Número de viagens que um caminhão l realiza na frente i em uma hora;

d_j^- : Desvio negativo do parâmetro j na mistura (t/h);

d_j^+ : Desvio positivo do parâmetro j na mistura (t/h);

P^- : Desvio negativo do ritmo de lavra em relação ao recomendado (t/h);

P^+ : Desvio positivo do ritmo de lavra em relação ao recomendado (t/h).

O modelo de programação matemática relativo à alocação dinâmica de uma frota heterogênea de caminhões e equipamentos de carga, levando-se em consideração metas de produção e qualidade de minério, é apresentado pelas equações (3.1)-(3.22).

$$\min \sum_{j \in T} \beta_j^- d_j^- + \sum_{j \in T} \beta_j^+ d_j^+ + \alpha^- P^- + \alpha^+ P^+ \quad 3.1$$

s.a:

$$\sum_{i \in M} (t_{ij} - tu_j) x_i \leq 0 \quad \forall j \in T \quad 3.2$$

$$\sum_{i \in M} (t_{ij} - tl_j) x_i \geq 0 \quad \forall j \in T \quad 3.3$$

$$\sum_{i \in M} (t_{ij} - tr_j) x_i + d_j^- - d_j^+ = 0 \quad \forall j \in T \quad 3.4$$

$$\sum_{i \in M} x_i \leq Pu \quad 3.5$$

$$\sum_{i \in M} x_i \geq Pl \quad 3.6$$

$$\sum_{i \in M} x_i + P^- - P^+ = Pr \quad 3.7$$

$$x_i \leq Qu_i \quad \forall i \in F \quad 3.8$$

$$x_i \geq Ql_i \quad \forall i \in F \quad 3.9$$

$$x_i \geq 0 \quad \forall i \in F \quad 3.10$$

$$d_j^+, d_j^- \geq 0 \quad \forall j \in T \quad 3.11$$

$$P^+, P^- \geq 0 \quad 3.12$$

$$\sum_{i \in E} x_i - rem \sum_{i \in M} x_i \geq 0 \quad 3.13$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad 3.14$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad 3.15$$

$$y_{ik} \in \{0,1\} \quad \forall i \in F, k \in C \quad 3.16$$

$$x_i - \sum_{k \in C} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad 3.17$$

$$x_i - \sum_{k \in C} Cl_k y_{ik} \geq 0 \quad \forall i \in F \quad 3.18$$

$$n_{il} tc_{il} - 60 \sum_{k \in C, g_{ik} \neq 0} y_{ik} \leq 0 \quad \forall i \in F, l \in V \quad 3.19$$

$$\sum_{i \in F} n_{il} tc_{il} \leq 60 \quad \forall l \in V \quad 3.20$$

$$x_i - \sum_{l \in V} n_{il} cap_l = 0 \quad \forall i \in F \quad 3.21$$

$$n_{il} \in \mathbb{Z}^+ \quad \forall i \in F, l \in V \quad 3.22$$

Observa-se que (3.2)-(3.12) são restrições que juntamente com a função objetivo (3.1) formam o modelo de mistura de minérios com metas. A restrição (3.13) diz respeito ao atendimento da relação estéril/minério mínima requerida. As demais restrições que complementam o modelo podem ser divididas em dois grupos. O primeiro diz respeito à alocação de equipamentos de carga e a faixa de produtividade que torne viável a utilização desses equipamentos. O conjunto de restrições (3.14) define que à cada frente deve ser alocado, no máximo, um único equipamento de carga, enquanto que as restrições (3.15) definem que cada equipamento de carga opera em uma única frente. As restrições (3.16) definem que as variáveis y_{ik} são binárias. As restrições (3.17) e (3.18) limitam, respectivamente, o ritmo de lavra máximo e mínimo.

O segundo grupo de restrições está relacionado ao transporte de material na mina e a alocação dos caminhões. Cada restrição (3.19) faz com que um caminhão somente realize viagens a uma frente onde esteja alocado um equipamento de carga compatível. As restrições (3.20) definem que cada caminhão opere no máximo sessenta minutos. As restrições (3.21) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados à frente. As restrições (3.22) determinam que o número de viagens que um caminhão faz à uma frente é um valor inteiro positivo.

3.4 Heurísticas

3.4.1 Introdução

Muitos problemas práticos são classificados na literatura como NP-completos ou NP-difíceis, isto é, são problemas para os quais ainda não existem algoritmos que os resolvam em tempo polinomial. Um exemplo clássico é o conhecido Problema do Caixeiro Viajante (PCV). O PCV é descrito por um conjunto de n cidades e uma matriz de distância entre elas, tendo o seguinte objetivo: o caixeiro viajante deve sair de uma cidade chamada origem, visitar cada uma das cidades restantes ($n-1$) apenas uma única vez e retornar à cidade origem percorrendo a menor distância possível. Em outras palavras, deve ser encontrado um ciclo hamiltoniano (ciclo fechado) de comprimento mínimo que passe exatamente uma única vez por cada cidade.

No PCV simétrico, situação na qual a distância da cidade i para a cidade j é igual à distância da cidade j para a cidade i , há $(n-1)!/2$ rotas possíveis para serem analisadas. Neste problema, o uso de métodos de programação matemática, dito exatos, se torna bastante restrito, uma vez que demandam tempos proibitivos para encontrar a melhor solução ou mesmo uma solução viável para valores elevados de n . Por outro lado, na prática, em geral, é suficiente encontrar uma “boa” solução para o problema, ao invés do ótimo global, o qual, às vezes, somente pode ser encontrado após um considerável esforço computacional.

Este é o motivo pelo qual os pesquisadores têm concentrado esforços na utilização de outros procedimentos para solucionar problemas desse nível de

complexidade. Dentre esses procedimentos, destacam-se as heurísticas. Uma heurística pode ser definida como uma técnica, normalmente inspirada em processos intuitivos, que procura uma boa solução a um custo computacional aceitável, sem, no entanto, estar capacitada a garantir sua otimalidade, bem como garantir quão próximo se está da solução ótima. O desafio é produzir, em tempo mínimo, soluções tão próximas quanto possíveis da solução ótima.

As heurísticas podem ser construtivas ou de refinamento. No primeiro caso, o objetivo é construir uma solução, elemento por elemento, adicionando-se a cada passo, um único elemento, normalmente aquele que tem o valor mais favorável da função guiadora da construção. Já as heurísticas de refinamento consistem em melhorar uma solução por meio de modificações realizadas nesta. Nas heurísticas clássicas, apenas as modificações que produzirem os melhores resultados são aceitos, sendo os métodos interrompidos quando se alcança o primeiro ótimo local.

Como exemplos clássicos de heurísticas convencionais têm-se o Método da Descida e o Método Randômico de Descida, a seguir apresentados.

3.4.1.1 Método da Descida/Subida (Descent/Uphill Method)

A idéia desta técnica é partir de uma solução inicial qualquer e a cada passo analisar todos os seus possíveis vizinhos, movendo somente para aquele que representar uma melhora no valor atual da função de avaliação. Dessa forma, o método pára quando um ótimo local é encontrado. A Figura 3.1 mostra o pseudocódigo do Método de Descida aplicado à minimização de uma função de avaliação f a partir de uma conhecida solução inicial s e considerando a busca em uma dada vizinhança $N(.)$.

```

procedimento Descida( $f(.), N(.), s$ );
1.       $V = \{s' \in N(s) \mid f(s') < f(s)\};$ 
2.      enquanto ( $|V| > 0$ ) faça
3.          Selecione  $s' \in V$ , onde  $s' = \arg \min \{f(s') \mid s' \in V\};$ 
4.           $s \leftarrow s'$ ;
5.           $V = \{s' \in N(s) \mid f(s') < f(s)\};$ 
6.      fim-enquanto;
7.      Retorne  $s$ ;
fim Descida;
```

Figura 3.1 – Método da Descida

3.4.1.2 Método Randômico de Descida

O Método de Descida/Subida requer a exploração de toda a vizinhança. Outro método alternativo, que evita essa pesquisa exaustiva é o Método Randômico de Descida/Subida (*Random Descent/Uphill Method*). Ele consiste em analisar um vizinho qualquer e o aceitar somente se ele for estritamente melhor que a solução corrente; não o sendo, a solução corrente permanece inalterada e outro vizinho é gerado. O procedimento é interrompido após um número fixo de iterações sem melhora no valor da melhor solução obtida até então. O método Randômico de Descida/Subida também fica preso no primeiro ótimo local encontrado (SOUZA, 2007).

Na Figura 3.2 mostra-se o pseudocódigo do Método Randômico de Descida aplicado ao refinamento de uma solução s em um problema de minimização de uma função $f(\cdot)$, utilizando uma estrutura de vizinhança $N(\cdot)$. Nessa figura, $IterMax$ representa o número máximo de iterações sem melhora no valor da função de avaliação.

```
procedimento RandomicoDescida( $f(\cdot), N(\cdot), IterMax, s$ );  
1.       $Iter = 0$ ; {Contador de iterações sem melhora }  
2.      enquanto ( $Iter < IterMax$ ) faça  
3.           $Iter \leftarrow Iter + 1$ ;  
4.          Selecione aleatoriamente  $s' \in N(s)$ ;  
5.          se ( $f(s') < f(s)$ ) então  
6.               $Iter = 0$ ;  
7.               $s \leftarrow s'$ ;  
8.          fim-se;  
9.      fim-enquanto;  
10.     Retorne  $s$ ;  
fim RandomicoDescida;
```

Figura 3.2 – Método Randômico de Descida

3.4.2 Metaheurísticas

A maioria das heurísticas desenvolvidas é muito específica para um problema particular, não sendo eficientes (ou mesmo aplicáveis) na resolução de uma classe mais ampla de problemas.

Somente a partir da década de 1980 intensificaram-se os estudos no sentido de se desenvolver procedimentos heurísticos com uma certa estrutura teórica e com caráter mais geral, sem prejudicar a principal característica destes, que é a flexibilidade.

Esta meta tornou-se mais realista a partir da reunião de conceitos das áreas de Otimização e Inteligência Artificial, viabilizando a construção das chamadas “melhores estratégias” ou dos métodos “inteligentemente flexíveis”, comumente conhecidos como “metaheurísticas”.

As metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico. Contrariamente às heurísticas convencionais, as metaheurísticas são de caráter geral e têm condições de escapar de ótimos locais (SOUZA, 2007).

Dentre os procedimentos enquadrados como metaheurísticas que surgiram ao longo das últimas décadas, destacam-se: Algoritmos Genéticos (AGs), Redes Neurais, *Simulated Annealing* (SA), Busca Tabu (BT), GRASP, VND, VNS, ILS, Colônia de Formigas etc.

3.4.2.1 *Iterated Local Search*

O método *Iterated Local Search* (ILS) é baseado na idéia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local. A perturbação precisa ser suficientemente forte para permitir que a busca local explore diferentes soluções, mas também fraca o suficiente para evitar um reinício aleatório (LOURENÇO *et al.*, 2003).

O método ILS é, portanto, um método de busca local que procura focar a busca não no espaço completo de soluções, mas em um pequeno subespaço definido por soluções que são ótimas locais de determinado procedimento de otimização (LOURENÇO *et al.*, 2003).

Para aplicar um algoritmo *ILS*, quatro componentes têm que ser especificadas: (a) Procedimento *GeraSolucaoInicial()*, que gera uma solução inicial s' para o problema; (b) Procedimento *BuscaLocal*, que retorna a melhor solução s'' de uma dada vizinhança; (c) Procedimento *Perturbação*, que modifica a solução corrente s guiando a uma solução intermediária s' e (d) Procedimento *CritérioAceitação*, que decide de qual solução a próxima perturbação será aplicada.

A Figura 3.3 ilustra o pseudocódigo do algoritmo *ILS* básico.

<u>Procedimento</u> <i>ILS()</i>	
1.	$s_0 \leftarrow \text{GeraSolucaoInicial}();$
2.	$s \leftarrow \text{BuscaLocal}(s_0);$
3.	$\text{iter} \leftarrow 0;$
4.	<u>enquanto</u> ($\text{iter} < \text{iter}_{\max}$)
5.	$\text{iter} \leftarrow \text{iter} + 1;$
6.	$s' \leftarrow \text{Perturbação}(\text{nível}, s);$
7.	$s'' \leftarrow \text{BuscaLocal}(s');$
8.	$s \leftarrow \text{CritérioAceitação}(s'');$
9.	<u>fim-enquanto;</u>
10.	<u>retorne</u> $s;$
<u>fim</u> <i>ILS</i>;	

Figura 3.3 – Pseudocódigo para o procedimento *ILS*

De acordo com Lourenço *et al.* (2003), o sucesso do *ILS* é centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha do método de busca local, das perturbações e do critério de aceitação. Em princípio, qualquer método de busca local pode ser usado, mas o desempenho do *ILS* com respeito à qualidade da solução final e a velocidade de convergência depende fortemente do método escolhido. Normalmente um método de descida é usado, mas também é possível aplicar algoritmos mais sofisticados, tais como Busca Tabu ou outras metaheurísticas.

O mecanismo de perturbação deve ser forte o suficiente para permitir escapar do ótimo local corrente e permitir explorar diferentes regiões do espaço de soluções. Ao mesmo tempo, a modificação precisa ser fraca o suficiente para guardar características do ótimo local corrente.

O critério de aceitação é usado para decidir de qual solução se continuará a exploração, bem como qual será a perturbação a ser aplicada. Um aspecto importante do

critério de aceitação e da perturbação é que eles induzem aos procedimentos de intensificação e diversificação. A intensificação consiste em permanecer na região do espaço onde a busca se encontra, procurando explorá-la de forma mais efetiva; enquanto a diversificação consiste em se deslocar para outras regiões do espaço de soluções.

A intensificação da busca ao redor da melhor solução encontrada é obtida, por exemplo, pela aplicação de “pequenas” perturbações sobre ela. A diversificação, por sua vez, pode ser realizada aceitando-se quaisquer soluções s e aplicando-se “grandes” perturbações na solução ótima local.

3.4.2.2 Método de Descida em Vizinhança Variável

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), de acordo com Mladenovic e Hansen (1997), é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada. Apresenta-se, a seguir, o pseudocódigo desse algoritmo para um problema de minimização.

- 1) Seja r o número de estruturas de vizinhança distintas
- 2) $s \leftarrow s_0$, onde s_0 é a solução inicial e s é a solução corrente
- 3) $k \leftarrow 1$, onde k é a estrutura de vizinhança corrente
- 4) enquanto $(k \leq r)$ faça
 - a) encontre o melhor vizinho $s' \in N^k(s)$, onde k é a k -ésima estrutura de vizinhança
 - b) se $f(s') < f(s)$, então $s \leftarrow s'$ e $k \leftarrow 1$
 - c) caso contrário, $k \leftarrow k + 1$
- 5) retorne a solução s

Figura 3.4 – Pseudocódigo do algoritmo VND

4. Modelagem do Problema Abordado

4.1 Introdução

Para resolver o problema de planejamento de lavra com alocação dinâmica de caminhões, são apresentados dois modelos: um de programação matemática e outro heurístico. Dada sua complexidade combinatória, uma vez que o mesmo se reduz ao problema da mochila múltipla, o qual é NP-difícil (COSTA, 2005), o primeiro modelo servirá para validar o segundo e será aplicado a instâncias de pequeno porte. Para problemas de porte mais elevado, o primeiro modelo, em geral, não é capaz de resolvê-los na otimalidade, em virtude do tempo proibitivo de execução.

4.2 Modelo de Programação Matemática

Para a modelagem exata desse problema, será utilizada a técnica de pesquisa operacional conhecida como programação por metas (*Goal Programming*), descrita na Seção 3.2.

O modelo proposto de alocação dinâmica de caminhões é uma adaptação daquele de Costa *et al.* (2004). Especificamente, a equação referente à relação estéril/minério é substituída por uma equação de meta de produção de estéril. Esta substituição é feita apenas para clareza de entendimento. Assim são criadas variáveis de desvio relativas ao não atendimento desta meta. Adicionalmente são incluídas restrições relativas às taxas de utilização dos veículos de transporte, bem como uma componente de avaliação do número de veículos usados.

Tal como em Costa *et al.* (2004), o planejamento da produção é feito para uma hora, sendo replicado enquanto não houver frente exaurida e as condições operacionais forem as mesmas.

O objetivo do modelo é minimizar os desvios das metas de produção, bem como os das metas de qualidade.

Para a apresentação do modelo, sejam os seguintes parâmetros de entrada:

M : Conjunto de frentes de minério;

E : Conjunto de frentes de estéril;

F : Conjunto de frentes formado por $M \cup E$;

T : Conjunto de parâmetros de controle¹ analisados no minério;

C : Conjunto de equipamentos de carga;

V : Conjunto de equipamentos de transporte;

Pr : Ritmo de lavra recomendado relativo a minério (t/h);

Pl : Ritmo de lavra mínimo relativo a minério (t/h);

Pu : Ritmo de lavra máximo relativo a minério (t/h);

Er : Ritmo de lavra recomendado relativo a estéril (t/h);

α^- : Penalidade por desvio negativo da produção de minério;

α^+ : Penalidade por desvio positivo da produção de minério;

γ^- : Penalidade por desvio negativo da produção de estéril;

γ^+ : Penalidade por desvio positivo da produção de estéril;

t_{ij} : Valor do parâmetro j na frente i (%);

tr_j : Valor recomendado para o parâmetro de controle j na mistura (%);

tl_j : Valor mínimo admissível para o parâmetro de controle j na mistura (%);

tu_j : Valor máximo admissível para o parâmetro de controle j na mistura (%);

β^- : Penalidade por desvio negativo para o parâmetro de controle j na mistura;

β^+ : Penalidade por desvio positivo para o parâmetro de controle j na mistura;

Qu_i : Ritmo de lavra máximo para a frente i (t/h);

¹ Teores de Fe, Al₂O₃ etc e granulometria. A granulometria ou análise granulométrica dos solos é o processo que visa definir, para determinadas faixas pré-estabelecidas de tamanhos de grãos, a percentagem em peso que cada fração possui em relação à massa total da amostra em análise (<http://wikipedia.org/wiki/granulometria>, acessado em julho de 2008)

Cl_k : Produção mínima do equipamento de carga k (t/h);

Cu_k : Produção máxima do equipamento de carga k (t/h);

cap_l : Capacidade do caminhão l (t);

tc_{il} : Tempo total de ciclo do caminhão l na frente i (min);

ω_l : Peso associado à importância da utilização do caminhão l ;

$TxMax_l$: Taxa máxima de utilização admitida para o caminhão l ;

$$g_{ik} = \begin{cases} 1 & \text{se o caminhão } l \text{ é compatível com o equipamento de carga } k, \\ 0 & \text{caso contrário} \end{cases}$$

e as seguintes variáveis de decisão:

x_i : Ritmo de lavra da frente i (t/h);

$$y_{ik} = \begin{cases} 1 & \text{se o equipamento de carga } k \text{ opera na frente } i; \\ 0 & \text{caso contrário} \end{cases}$$

n_{il} : Número de viagens que um caminhão l realiza na frente i em uma hora;

d_j^- : Desvio negativo do parâmetro de controle j na mistura (t/h);

d_j^+ : Desvio positivo do parâmetro de controle j na mistura (t/h);

P^- : Desvio negativo do ritmo de lavra de minério em relação ao recomendado (t/h);

P^+ : Desvio positivo do ritmo de lavra de minério em relação ao recomendado (t/h).

Pe^- : Desvio negativo do ritmo de lavra de estéril em relação ao recomendado (t/h);

Pe^+ : Desvio positivo do ritmo de lavra de estéril em relação ao recomendado (t/h).

$usou_l$: Variável binária que assume valor 1 se o caminhão l é usado e 0, caso contrário.

A seguir, é apresentado pelas equações (4.1)-(4.24) o modelo de programação matemática relativo à alocação dinâmica de uma frota heterogênea de caminhões e equipamentos de carga, levando-se em consideração metas de produção e qualidade de minério.

$$\min \sum_{j \in T} \beta_j^- d_j^- + \sum_{j \in T} \beta_j^+ d_j^+ + \alpha^- P^- + \alpha^+ P^+ + \gamma^- Pe^- + \gamma^+ Pe^+ + \sum_{l \in V} \omega_l usou_l \quad 4.1$$

s.a:

$$\sum_{i \in M} (t_{ij} - tu_j) x_i \leq 0 \quad \forall j \in T \quad 4.2$$

$$\sum_{i \in M} (t_{ij} - tl_j) x_i \geq 0 \quad \forall j \in T \quad 4.3$$

$$\sum_{i \in M} (t_{ij} - tr_j) x_i + d_j^- - d_j^+ = 0 \quad \forall j \in T \quad 4.4$$

$$\sum_{i \in M} x_i \leq Pu \quad 4.5$$

$$\sum_{i \in M} x_i \geq Pl \quad 4.6$$

$$\sum_{i \in M} x_i + P^- - P^+ = Pr \quad 4.7$$

$$\sum_{i \in E} x_i + Pe^- - Pe^+ = Er \quad 4.8$$

$$x_i \leq Qu_i \quad \forall i \in F \quad 4.9$$

$$x_i \geq 0 \quad \forall i \in F \quad 4.10$$

$$d_j^+, d_j^- \geq 0 \quad \forall j \in T \quad 4.11$$

$$P^+, P^- \geq 0 \quad 4.12$$

$$Pe^+, Pe^- \geq 0 \quad 4.13$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad 4.14$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad 4.15$$

$$y_{ik} \in \{0,1\} \quad \forall i \in F, k \in C \quad 4.16$$

$$x_i - \sum_{k \in C} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad 4.17$$

$$x_i - \sum_{k \in C} Cl_k y_{ik} \geq 0 \quad \forall i \in F \quad 4.18$$

$$n_{il} tc_{il} - 60 \sum_{k \in C, g_{ik} \neq 0} y_{ik} \leq 0 \quad \forall i \in F, l \in V \quad 4.19$$

$$\frac{1}{60} \sum_{i \in F} n_{il} tc_{il} \leq TxMax_l \quad \forall l \in V \quad 4.20$$

$$x_i - \sum_{l \in V} n_{il} cap_l = 0 \quad \forall i \in F \quad 4.21$$

$$usou_l \geq \frac{1}{60} \sum_{i \in F} n_{il} tc_{il} \quad \forall l \in V \quad 4.22$$

$$n_{il} \in \mathbb{Z}^+ \quad \forall i \in F, l \in V \quad 4.23$$

$$usou_l \in \{0,1\} \quad l \in V \quad 4.24$$

O modelo de mistura de minérios com metas é formado pelas restrições (4.2)-(4.13) e pela função objetivo (4.1). As restrições (4.7) e (4.8) dizem respeito, respectivamente, aos atendimentos das metas de produção de minério e estéril. As demais restrições que complementam o modelo podem ser divididas em dois grupos. O primeiro diz respeito à alocação de equipamentos de carga e a faixa de produtividade que torne viável a utilização desses equipamentos. O segundo grupo de restrições está relacionado ao transporte de material na mina e a alocação e utilização dos caminhões.

Em relação ao primeiro grupo de restrições, o conjunto (4.14) define que em cada frente pode ser alocado, no máximo, um único equipamento de carga, enquanto que as restrições (4.15) definem que cada equipamento de carga pode operar, no máximo, em uma única frente. As restrições (4.16) definem que as variáveis y_{ik} são binárias. As restrições (4.17) e (4.18) limitam, respectivamente, o ritmo de lavra máximo e mínimo, definido pela carregadeira alocada à frente. Já as restrições (4.9) limitam o ritmo de lavra máximo definido pelo usuário.

No segundo grupo de restrições, que está relacionado ao transporte de material na mina e a alocação e utilização dos caminhões, cada restrição (4.19) faz com que um caminhão somente realize viagens à uma frente onde esteja alocado um equipamento de carga compatível. As restrições (4.20) definem que cada caminhão opere no máximo $TxMax_l\%$ em uma hora. As restrições (4.21) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados à frente. As restrições (4.22), juntamente com a função objetivo, forçam com que os caminhões usados sejam penalizados. As restrições (4.23) determinam que o número de viagens que um

caminhão faz a uma frente é um valor inteiro positivo. As restrições (4.24) indicam que as variáveis $usou_l$ são binárias.

4.3 Modelo Heurístico

4.3.1 Dados de Entrada

Sejam: F o conjunto de frentes, V o conjunto de veículos, C o conjunto de equipamentos de carga, T o conjunto dos parâmetros de controle (teores de Fe, Al_2O_3 e etc e granulometria) presentes nas frentes, qu_i a quantidade máxima de massa lavrável em cada frente i , tc_{il} o tempo de ciclo de cada veículo l a cada frente i , t_{ij} o percentual do parâmetro de controle j presente na frente i , $Compatib_{|C| \times |V|}$ a matriz que representa a compatibilidade entre os equipamentos de carga k e os caminhões l , cap_l a capacidade de cada caminhão l , Cl_k a capacidade mínima de cada carregadeira k e Cu_k a capacidade máxima de cada carregadeira k e $TxMax_l$ a taxa de utilização máxima permitida para o l -ésimo caminhão.

Descreve-se, a seguir, a representação de uma solução para o problema, a função de avaliação de uma solução, os movimentos e as técnicas heurísticas utilizadas na exploração do espaço de soluções do problema.

4.3.2 Representação de uma Solução

Uma solução é representada por uma matriz $R_{|F| \times (1+V)}$ formada pelas matrizes $Y_{|F| \times 1}$ e $N_{|F| \times |V|}$, em que:

y_{ik} : representa a alocação do equipamento de carga k à frente i , observando-se o estado ativo ou inativo de cada equipamento;

n_{il} : representa o número de viagens do caminhão l à frente i .

$$R_{|F| \times (1+V)} = [Y_{|F| \times 1} \mid N_{|F| \times |V|}] \quad 4.25$$

A partir de Y , N e os tempos de ciclo da matriz $T_{C_{|F| \times |V|}}$ são determinados $S_{|F| \times 1}$ e $T_{cv_{1 \times |V|}}$, os quais representam, respectivamente, a quantidade de massa lavrada em cada frente e o somatório dos tempos de ciclo de cada veículo.

A Figura 4.1 exemplifica uma solução para uma instância do problema, considerando $|F|$ frentes e $|V|$ caminhões. As linhas representam as frentes de lavra (F). A coluna *Carreg* ($Y_{|F| \times 1}$) representa a alocação das carregadeiras às frentes. As outras colunas ($N_{|F| \times |V|}$) representam as viagens realizadas pelos caminhões (*Cam*) em cada frente.

	Y	N			
	<i>Carreg</i>	<i>Cam₁</i>	<i>Cam₂</i>	...	<i>Cam_V</i>
F_1	(5 , 1)	5	0	...	2
F_2	(3 , 0)	-1	-1	...	0
F_3	(-1 , 0)	0	0	...	-1
F_4	(6 , 1)	7	-1	...	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
$F_{ F }$	(2 , 1)	4	0	...	1

Figura 4.1 – Representação de uma Solução

Cada célula da matriz $Y_{|F| \times 1}$ é composta de duas informações: a primeira diz respeito ao número da carregadeira k alocada à frente i . O valor -1 significa que não existe carregadeira alocada. A segunda informação diz se a carregadeira está ativa (1) ou não (0).

Na matriz $N_{|F| \times |V|}$, cada célula é composta do número de viagens de um caminhão l a uma frente i . O valor 0 (zero) significa que não há nenhuma viagem para aquele

caminhão. O valor -1 informa a incompatibilidade entre o caminhão e a carregadeira alocados àquela frente.

Exemplificando, a linha 1 diz respeito à frente F_1 onde está alocada a carregadeira de número 5 que se encontra ativa (5 , 1). Já na frente F_2 , a carregadeira 3 está alocada, mas não se encontra em operação no momento (3 , 0). Entretanto, na frente F_3 não existe carregadeira alocada no momento (-1 , 0). Em relação à matriz $N_{|F| \times |V|}$, que se refere ao número de viagens de um caminhão l a uma determinada frente i , a linha 4 mostra que o caminhão Cam_1 realizará 7 viagens à frente F_4 , o caminhão Cam_2 é incompatível com a carregadeira de número 6 alocada à frente e o caminhão $Cam_{|V|}$ não fará nenhuma viagem à esta frente.

4.3.3 Geração de uma Solução Inicial

Para se gerar a solução inicial é necessário se construir as matrizes $Y_{|F| \times 1}$, $N_{|F| \times |V|}$, $S_{|F| \times 1}$ e $Tcv_{1 \times |V|}$.

A Figura 4.2 mostra o pseudocódigo da geração da matriz Y (alocação das carregadeiras). Inicialmente, atribui-se o valor -1 para todas as carregadeiras, ou seja, considera-se que não existe carregadeira alocada a nenhuma frente. Atribui-se, também, o valor 0 (zero) para o *status* de cada carregadeira informando a inatividade de cada uma delas em cada frente. Em seguida, as carregadeiras são alocadas aleatoriamente a cada uma das frentes disponíveis. Para estas carregadeiras, atribui-se o *status* ‘ativo’ (1), inicialmente.

```

Procedimento Gera_Carreg( Y );
  Variáveis  $i, k$ : inteiro;
  Início
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncarreg  $\leftarrow$  n° total de carregadeiras;
    Para  $i \leftarrow 1$  até maxFr
       $y[i].n\_carr \leftarrow -1$ ; {número da carregadeira}
       $y[i].n\_ativo \leftarrow 0$ ; {status da carregadeira}
    Fim para
    Para  $k \leftarrow 1$  até Ncarreg
      Repita
         $i \leftarrow \text{aleatório}(\text{maxFr})$ ;
        se  $y[i].n\_carr = -1$  {não existe carregadeira alocada}
           $y[i].n\_carr \leftarrow k$ ;
           $y[i].n\_ativo \leftarrow 1$ ;
        até  $y[i].n\_car \neq -1$ 
      Fim para
  Fim{ Gera_Carreg}

```

Figura 4.2 – Alocação Aleatória das Carregadeiras

A Figura 4.3 mostra o pseudocódigo da geração da matriz $N_{|F| \times |V|}$ (número de viagens de cada caminhão l às diversas frentes i). O número de viagens em cada célula da matriz $N_{|F| \times |V|}$ é gerado aleatoriamente. Para cada célula n_{il} é verificado inicialmente o número máximo (nv_max) de viagens, por hora, em função do tempo de ciclo (T_c) de cada caminhão l às diversas frentes i . É gerado, então, de forma aleatória, um número compreendido entre 0 e o número máximo de viagens (nv_max). Posteriormente, é verificado se existe carregadeira alocada a essa frente. Se existir, verifica-se se há compatibilidade entre a carregadeira k e o caminhão alocado l . Caso afirmativo, verifica-se se a carregadeira k alocada à frente i encontra-se ativa ou não. Estando ativa, considera-se o número aleatório calculado anteriormente como sendo o número de viagens do caminhão l à frente i . Não estando ativa ou não havendo compatibilidade entre a carregadeira e o caminhão ou, ainda, não havendo carregadeira alocada à frente, não é atribuído a esse caminhão nenhuma viagem nesta frente. O procedimento é repetido para todas as células da matriz $N_{|F| \times |V|}$.

```

Procedimento Gera_Viagens( N );
  Variáveis  $i, l, nv, nv\_max$ : inteiro;
  Início
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncam  $\leftarrow$  n° total de caminhões;
    Para  $i \leftarrow 1$  até MaxFr
      Para  $l \leftarrow 1$  até Ncam
         $nv\_max \leftarrow \text{int}(60/Tc[i, l]);$ 
         $nv \leftarrow \text{aleatório}(nv\_max);$ 
        se carregadeira_alocada
          se houver_compatib_Carreg_Cam
            se carregadeira_ativa
               $n[i, l] \leftarrow nv;$ 
      Fim para
    Fim para
  Fim{ Gera_Viagens}

```

Figura 4.3 – Número de Viagens de cada Caminhão às diversas Frentes

A Figura 4.4 mostra o pseudocódigo da geração da matriz $S_{|F| \times 1}$ (quantidade de massa lavrável em cada frente). A quantidade de massa lavrável (s_i) em cada frente é gerada da seguinte forma: para cada frente i , é gerado s_i como sendo o somatório dos produtos das matrizes $N_{|F| \times |V|}$ e $Cap_{|V|}$, ou seja, entre o número de viagens (n_{il}) e a capacidade de cada caminhão (cap_l).

```

Procedimento Gera_Massa_Lavrável( S, N, Cap );
  Variáveis  $i, l$ : inteiro;
  Início
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncam  $\leftarrow$  n° total de caminhões;
    Para  $i \leftarrow 1$  até MaxFr
      Para  $l \leftarrow 1$  até Ncam
        Se  $n[i, l] > 0$ 
           $s[i] \leftarrow \sum (n[i, l] * cap_l);$ 
      Fim para
    Fim para
  Fim{ Gera_Massa_Lavrável}

```

Figura 4.4 – Massa Lavrável em cada Frente

A Figura 4.5 mostra o pseudocódigo da geração da matriz $Tcv_{1 \times |V|}$ (somatório dos tempos de ciclo por caminhão). Para cada caminhão, é calculado o somatório do

produto entre o número de viagens (n_{il}) e o tempo de ciclo Tc_{il} nas diversas frentes, gerando a matriz $Tcv_{1 \times |V|}$.

```

Procedimento Gera_Tempo_Cam( Tcv, N, Tc );
  Variáveis  $i, l$ : inteiro;
  Início
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncam  $\leftarrow$  n° total de caminhões;
    Para  $l \leftarrow 1$  até Ncam
      Para  $i \leftarrow 1$  até MaxFr
        Se  $n[i, l] > 0$ 
           $tcv[l] \leftarrow \sum (n[i, l] * tc[i, l]);$ 
        Fim para
      Fim para
  Fim{Gera_Tempo_Cam}

```

Figura 4.5 – Somatório dos Tempos de Ciclo por Caminhão

Mesmo com a geração de todas essas matrizes, é conveniente verificar se existem inviabilidades na solução inicial gerada. A correção das mesmas propicia uma solução inicial de melhor qualidade diminuindo o esforço computacional exigido no refinamento dessa solução.

O procedimento verifica a existência de duas inviabilidades. A primeira diz respeito ao somatório dos tempos de ciclo por caminhão. O tempo de ciclo acumulado de cada caminhão não poderá exceder a 60 minutos. Como o objetivo é fornecer uma solução em uma hora, há a necessidade de se fazer uma correção na solução.

A Figura 4.6 mostra o pseudocódigo referente ao “Reparador de Inviabilidade do Número de Viagens” que tem como objetivo refinar o número de viagens do caminhão l à todas as frentes, fazendo com que o somatório dos tempos de ciclo seja inferior ou, no máximo, igual a 60 minutos.

O pseudocódigo da Figura 4.6 mostra que, para cada caminhão l , é verificado se o tempo de ciclo acumulado (tcv_l) é superior a 60 minutos. Caso seja, é escolhida uma frente i , aleatoriamente, e retirada uma viagem do caminhão l , caso exista. Em seguida, subtrai-se o tempo de ciclo ($tc_{i,l}$) correspondente. O procedimento é repetido, para cada caminhão, até que o tempo acumulado seja inferior ou igual a 60 minutos.

```

Procedimento Refina_N(N,Tc,Tcv);
  Variáveis  $i,l$ :inteiro;
  Início
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncam  $\leftarrow$  n° total de caminhões;
    Para  $l \leftarrow 1$  até Ncam
      enquanto Tcv[l]>60
         $i \leftarrow$  aleatório(maxFr);
        Se  $n[i,l] > 1$ 
           $n[i,l] \leftarrow n[i,l]-1$ ;
           $tcv[l] \leftarrow tcv[l] - tc[i,l]$ ;
        Fim enquanto
      Fim para
  Fim{ Refina_N }

```

Figura 4.6 – Reparador de Inviabilidade do Número de Viagens

Após aplicar este reparador de inviabilidade, é necessário reconstruir a matriz $S_{|F| \times 1}$ (massa lavrável em cada frente), como mostrado na Figura 4.4.

A segunda inviabilidade diz respeito à quantidade de massa lavrável, por frente. Mesmo com a correção da primeira inviabilidade, não há garantias de que a massa lavrável, calculada através da aplicação do pseudocódigo mostrado na fig. 4.4, seja inferior à quantidade máxima lavrável (Q_u) existente em cada frente. Sendo superior, há a necessidade de se fazer a correção como explicitado no pseudocódigo mostrado na Figura 4.7. Essa correção também deverá ser realizada caso a quantidade de massa lavrável calculada seja maior que a capacidade máxima de produção da carregadeira em uma hora.

Procedimento Refina_S(S, N, Qu, Tc, Tcv, Cu, Y, Cap);

Variáveis i, l : inteiro;

Início

MaxFr $\leftarrow$ n° máximo de frentes;

Ncam $\leftarrow$ n° total de caminhões;

Para $i \leftarrow 1$ até MaxFr

Enquanto ($s[i] > qu[i]$) ou ($s[i] > cu[Y[i].N\_carr]$)

$l \leftarrow \text{aleatório}(Ncam)$;

Se $n[i, l] > 0$

$n[i, l] \leftarrow n[i, l] - 1$;

$tcv[l] \leftarrow tcv[l] - tc[i, l]$;

$s[i] \leftarrow s[i] - cap[l]$;

se minério

$t \leftarrow t - cap[l]$

senão est $\leftarrow est - cap[l]$;

Fim enquanto

Fim para

Fim{ Refina_S }

Figura 4.7 – Reparador de Inviabilidade de Massa Lavrável

O pseudocódigo mostrado na Figura 4.7 mostra que, para cada frente, é verificado se a quantidade de massa lavrável calculada inicialmente é superior à quantidade máxima (Qu) existente na frente i e superior à capacidade máxima de produção da carregadeira alocada à frente i , por hora. Se pelo menos uma das duas situações acontecer, é escolhido, aleatoriamente, um caminhão l . Caso haja viagens desse caminhão l à frente i , é retirada uma viagem e recalculados o tempo total (tcv) do caminhão l e a quantidade de massa lavrável s_i . É atualizada também a variável que corresponde à somatória de todas as massas lavráveis, seja de minério (t), seja de estéril (est). Esse procedimento é repetido para todas as frentes até que não haja mais nenhuma inviabilidade.

Esse processo de geração de uma solução inicial é repetido *IterConstr* vezes e, em cada vez, aplica-se o método da descida em vizinhança variável (VND), descrito na Figura 4.45 da seção 4.3.8, e a melhor solução obtida é utilizada como solução inicial para o ILS. Este procedimento tem a vantagem de produzir uma solução inicial de boa qualidade rapidamente, já que a busca local exige pouco esforço computacional.

4.3.4 Avaliação de uma Solução

Uma solução s é avaliada com relação aos desvios das metas de produção e qualidade. Considerando que o modelo heurístico pode gerar inviabilidades, além da função objetivo do modelo de programação matemática, descrito na equação (4.1), são acrescentadas outras parcelas para penalizar tais ocorrências. Adicionalmente, são utilizados, também, componentes na função de avaliação para guiar a busca no espaço de soluções.

Assim, uma solução s é avaliada em função da produção, da qualidade do produto final e da alocação dos equipamentos utilizados (carga e transporte), de acordo com a equação a seguir:

$$f(s) = f^p(s) + \sum_{j \in T} f_j^q(s) + f^l(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \quad 4.26$$

em que:

$f^p(s)$: função que avalia s quanto à produção;

$f_j^q(s)$: função que avalia s quanto à qualidade do j -ésimo parâmetro de controle;

$f_l^u(s)$: função que avalia s quanto ao atendimento da taxa de utilização máxima do l -ésimo caminhão.

$f^l(s)$: função que avalia s quanto ao número de caminhões utilizados.

$f_k^c(s)$: função que avalia s quanto à produtividade da carregadeira k .

As duas últimas parcelas da função de avaliação são usadas apenas para guiar a busca no espaço de soluções e são desconsideradas na apresentação dos resultados de forma a permitir uma comparação com a função objetivo do modelo de programação matemática.

A seguir, é mostrada a avaliação de cada um dos componentes da função $f(s)$:

- **Produção de Minério e Estéril**

A Produção de minério e estéril é avaliada pela equação 4.27.

$$f^p(s) = \beta \times (|Pr - P| + \gamma \times |Pre - Pe|) \quad 4.27$$

onde:

P : Produção de minério (t);

Pr : Meta de Produção de Minério (t);

Pe : Produção de Estéril (t);

Pre : Meta de Produção de Estéril (t);

β : Penalidade por não atendimento à meta de minério;

γ : Penalidade por não atendimento à meta de estéril.

A Produção P é calculada segundo a equação 4.28, ou seja, é obtida pela multiplicação do número de viagens dos caminhões às frentes, pela capacidade de carga de cada caminhão.

$$P = \sum_{i \in M} \sum_{l \in V} n_{il} cap_l \quad 4.28$$

onde:

M : conjunto de frentes de minério;

V : conjunto de equipamentos de transporte;

n_{il} : número de viagens de um caminhão l à frente i ;

cap_l : capacidade do caminhão l (t);

Para o cálculo de Pe , substitui-se o conjunto das frentes de minério M pelo conjunto E das frentes de estéril e aplica-se a fórmula 4.28.

▪ Qualidade da Mistura

A qualidade da mistura é avaliada segundo a equação 4.29.

$$f_j^q(s) = \alpha_j^q \times P \times \frac{|tr_j - tm_j|}{100} \quad \forall j \in T \quad 4.29$$

onde :

tm_j : Valor percentual encontrado para o parâmetro j (%) na mistura;

tr_j : Meta de qualidade para o parâmetro j (%);

P : Produção de minério (t);

α_j^q : Penalidade por não atendimento à meta de qualidade do parâmetro de controle j ;

O valor tm_j é calculado através da média ponderada entre t_{ij} e a produção (em uma hora) de cada frente i de minério, conforme mostrado na equação 4.30.

$$tm_j = \frac{\sum_{i \in M} t_{ij} x_i}{\sum_{i \in M} x_i} \quad \forall j \in T \quad 4.30$$

onde o ritmo de lavra em cada frente i é calculado com base na equação a seguir:

$$x_i = \sum_{l \in V} n_{il} cap_l \quad 4.31$$

▪ Taxa de utilização dos caminhões

A utilização dos caminhões é avaliada pela equação 4.32.

$$f_l^u(s) = \mu \times |Tx_l - TxMax_l| \quad \forall l \in V \quad 4.32$$

onde:

μ : Penalidade por se ultrapassar a taxa máxima de utilização dos caminhões;

$TxMax_l$: Taxa máxima permitida de utilização do caminhão l ;

Tx_l : Taxa de utilização do caminhão l , dada por:

$$Tx_l = \sum_{i \in F} n_{il} tc_{il} \quad \forall l \in V \quad 4.33$$

sendo n_{il} o número de viagens realizadas pelo caminhão l à frente i e tc_{il} o tempo de ciclo do caminhão l à frente i .

São utilizados dois valores para o peso μ na expressão (4.32): um valor alto (μ_A) caso a quantidade transportada esteja acima da máxima permitida e outro valor, mais baixo (μ_B), caso a quantidade transportada esteja abaixo da máxima permitida.

▪ **Número de caminhões utilizados**

A avaliação do número de caminhões utilizados é feita com base na equação 4.34.

$$f^l(s) = \omega \times \sum_{l \in V} \lceil Tx_l \rceil \quad 4.34$$

onde:

ω : Penalidade pela utilização de caminhões;

$\lceil Tx_l \rceil$: Função que assume valor igual a 1 se o caminhão l for utilizado;

▪ **Avaliação da produtividade das carregadeiras**

A produtividade das carregadeiras é avaliada pela equação 4.35.

$$f_k^c(s) = \begin{cases} \varphi \times (Cu_k - prod_k) & \text{se } Cl_k \leq prod_k \leq Cu_k \\ \lambda \times (Cl_k - prod_k) & \text{se } prod_k < Cl_k \end{cases} \quad \forall k \in C \quad 4.35$$

onde:

φ : Penalidade por não se atingir a produtividade máxima da carregadeira k ;

λ : Penalidade por utilizar uma carregadeira k com uma produtividade inferior à mínima;

4.3.5 Movimentos e Estruturas de Vizinhanças

Seja s' uma solução vizinha de uma dada solução s , e seja *movimento* alguma alteração que proporcione mudança na solução corrente. A construção de s' é determinada após se aplicar um ou mais movimentos.

Foram desenvolvidos vários tipos de movimentos, aplicáveis tanto para as frentes de minério quanto para as de estéril:

- Retirar uma viagem de um caminhão em uma frente qualquer (Retirar_Viagem);
- Retirar duas viagens de um caminhão em uma frente qualquer (Retirar_Duas_Viagens);
- Realocar uma viagem a uma frente diferente, mantendo o caminhão (Realocar_Viagem_mc);
- Realocar uma viagem a um caminhão diferente, mantendo a frente (Realocar_Viagem_mf);
- Realocar um equipamento de carga, uma única vez (Realocar_Carregadeira);
- Ativar um equipamento de carga (Ativar_Carregadeira);
- Desativar um equipamento de carga (Desativar_Carregadeira);
- Inserir uma viagem de um caminhão em uma frente (Inserir_Viagem);
- Retirar todas as viagens de um caminhão em todas as frentes (Retirar_Viagens);
- Trocar carregadeira inativa por uma ativa (Trocar_carregadeira);

Esses movimentos definem as estruturas de vizinhança e são detalhados a seguir.

4.3.5.1 Movimento Retirar_Viagem

O movimento “Retirar Viagem” consiste em retirar uma viagem de um determinado caminhão, ou seja, seleciona-se uma célula n_{il} da matriz N e subtrai-se uma viagem do caminhão l na frente i .

A Figura 4.8 ilustra o movimento relatado acima.

Antes do Movimento								Depois do Movimento							
	1	2	3	4	5	...	V		1	2	3	4	5	...	V
1	2	0	1	4	1	...	1	1	2	0	1	4	1	...	1
2	0	3	5	1	0	...	4	2	0	3	4	1	0	...	4
3	-1	-1	-1	-1	-1	...	-1	3	-1	-1	-1	-1	-1	...	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F	1	0	2	5	2	...	0	F	1	0	2	5	2	...	0

Figura 4.8 – Movimento Retirar_uma_viagem

Nessa figura o caminhão de número 3 realiza 5 viagens à frente 2, inicialmente. Após a implementação desse movimento, este caminhão passa a realizar 4 viagens à frente 2.

4.3.5.2 Movimento Retirar_Duas_Viagens

Nesse movimento, o procedimento é análogo ao mostrado no item 4.3.5.1. Consiste em retirar duas viagens de um determinado caminhão, ou seja, seleciona-se uma célula n_{il} da matriz N e subtraem-se duas viagens do caminhão l na frente i .

A Figura 4.9 ilustra o movimento relatado acima.

Antes do Movimento								Depois do Movimento							
	1	2	3	4	5	...	V		1	2	3	4	5	...	V
1	2	0	1	4	1	...	1		2	0	1	2	1	...	1
2	0	3	5	1	0	...	4		0	3	5	1	0	...	4
3	-1	-1	-1	-1	-1	...	-1		-1	-1	-1	-1	-1	...	-1
...	...	...	...	...	...	...	...		...	...	...	...	...	...	...
F	1	0	2	5	2	...	0		1	0	2	5	2	...	0

Figura 4.9 – Movimento Retirar_duas_viagens

Nessa figura o caminhão de número 4 realiza 4 viagens à frente 1, inicialmente. Após a implementação desse movimento, este caminhão passa a realizar 2 viagens à frente 1.

4.3.5.3 Movimento Realocar_Viagem_mc

O movimento “Realocar_Viagem_mc” consiste em transferir uma viagem de uma frente i para uma outra frente i' , mantendo o mesmo caminhão l , isto é, seleciona-se uma célula n_{il} da matriz N , retira-se uma viagem, seleciona-se outra célula $n_{i'l}$ da matriz N e faz-se a transferência da viagem.

A Figura 4.10 ilustra este movimento.

Antes do Movimento								Depois do Movimento							
	1	2	3	4	5	...	V		1	2	3	4	5	...	V
1	2	0	1	4	1	...	1	1	2	0	1	4	1	...	1
2	0	3	5	1	0	...	4	0	3	5	1	1	1	...	4
3	-1	-1	-1	-1	-1	...	-1	-1	-1	-1	-1	-1	-1	...	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F	1	0	2	5	2	...	0	1	0	2	5	1	1	...	0

Figura 4.10 – Movimento Realocar uma Viagem mantendo o caminhão

Na Figura 4.10 o caminhão de número 5 realiza 2 viagens à frente F e nenhuma viagem à frente 2, inicialmente. Após a implementação deste movimento, este caminhão passa a realizar 1 viagem à frente F e 1 viagem frente 2.

4.3.5.4 Movimento Realocar_Viagem_mf

O movimento “Realocar_Viagem_mf” consiste em transferir uma viagem de um caminhão l para um outro caminhão l' , mantendo a mesma frente i , isto é, seleciona-se uma célula n_{il} da matriz N , retira-se uma viagem, seleciona-se outra célula $n_{i'l'}$ da matriz N e faz-se a transferência da viagem.

A Figura 4.11 ilustra este movimento.

Antes do Movimento								Depois do Movimento							
	1	2	3	4	5	...	V		1	2	3	4	5	...	V
1	2	0	1	4	1	...	1	2	0	1	4	1	1	...	1
2	0	3	5	1	0	...	4	0	2	5	2	2	0	...	4
3	-1	-1	-1	-1	-1	...	-1	-1	-1	-1	-1	-1	-1	...	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F	1	0	2	5	2	...	0	1	0	2	5	2	2	...	0

Figura 4.11 – Movimento Realocar uma Viagem mantendo a frente

Nessa figura o caminhão de número 2 realiza 3 viagens à frente 2 e o caminhão 4 realiza 1 viagem à frente 2, inicialmente. Após a implementação desse movimento, o caminhão 2 passa a realizar 2 viagens e o caminhão 4 realiza também 2 viagens à frente 2.

4.3.5.5 Movimento Realocar_Carregadeira

No movimento “Realocar_Carregadeira”, procede-se da seguinte forma: a carregadeira k e todas as viagens alocadas à frente i são transferidas para uma outra frente i' . Da mesma forma, a carregadeira k' e todas as viagens alocadas à frente i' são transferidas para a frente i (se houver carregadeira alocada à frente i').

A Figura 4.12 ilustra este movimento.

Antes do Movimento									Depois do Movimento								
	Carreg	1	2	3	4	5	...	V		Carreg	1	2	3	4	5	...	V
1	(3, 1)	2	0	1	4	1	...	1	(-1,0)	-1	-1	-1	-1	-1	-1	...	-1
2	(6, 1)	0	3	5	1	0	...	4	(6, 1)	0	3	5	1	0	0	...	4
3	(-1,0)	-1	-1	-1	-1	-1	...	-1	(3, 1)	2	0	1	4	1	1	...	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F	(1, 1)	1	0	2	5	2	...	0	(1, 1)	1	0	2	5	2	2	...	0

Figura 4.12 – Movimento Realocar Carregadeira

A Figura 4.12 mostra o movimento “Realocar_Carregadeira” efetuado nas frentes 1 e 3. A carregadeira de número 3 alocada à frente 1 é transferida para a frente 3 bem como todas as viagens alocadas à essa frente. Mesmo não havendo carregadeira alocada à frente 3, inicialmente, todos os dados relativos a essa frente são transferidos à frente 1.

4.3.5.6 Movimento Ativar_Carregadeira

Esse movimento consiste em ativar uma carregadeira, isto é, seleciona-se uma carregadeira, verifica-se se ela encontra-se desativada e, caso esteja, muda-se o seu estado para ativada. Se a carregadeira selecionada não estiver desativada, repete-se o procedimento um certo número de vezes até que se encontre uma desativada ou até que se ultrapasse esse número de vezes, no caso, dado pelo produto do número de frentes pelo número de caminhões.

A Figura 4.13 mostra o movimento “Ativar_Carregadeira” efetuado na frente 2. A carregadeira de número 6 alocada na frente 2 que estava desativada passa a estar ativa. Isso significa que, a partir daquele momento, essa frente estará em atividade podendo, portanto, ter caminhões alocados a ela.

Antes do Movimento								Depois do Movimento									
	Carreg	1	2	3	4	5	...	V		Carreg	1	2	3	4	5	...	V
1	(3, 1)	2	0	1	4	1	...	1	(3, 1)	2	0	1	4	1	...	1	
2	(6, 0)	0	0	0	0	0	...	0	(6, 1)	0	0	0	0	0	...	0	
3	(-1,0)	-1	-1	-1	-1	-1	...	-1	(-1,0)	-1	-1	-1	-1	-1	...	-1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
F	(1, 1)	1	0	2	5	2	...	0	(1, 1)	1	0	2	5	2	...	0	

Figura 4.13 – Movimento Ativar Carregadeira

4.3.5.7 Movimento Desativar_Carregadeira

O movimento “Desativar_Carregadeira” consiste em desativar uma carregadeira, isto é, seleciona-se uma carregadeira, verifica-se se ela encontra-se ativada e, caso

esteja, muda-se o seu estado para desativada. Caso não esteja, repete-se o procedimento até encontrar uma carregadeira ativa. Encontrando, retiram-se todas as viagens alocadas aos caminhões que trabalham naquela frente.

A Figura 4.14 ilustra este movimento.

Antes do Movimento									Depois do Movimento								
	Carreg	1	2	3	4	5	...	V		Carreg	1	2	3	4	5	...	V
1	(3, 1)	2	0	1	4	1	...	1	(3, 1)	2	0	1	4	1	1	...	1
2	(6, 1)	3	0	1	0	1	...	0	(6, 0)	0	0	0	0	0	0	...	0
3	(-1,0)	-1	-1	-1	-1	-1	...	-1	(-1,0)	-1	-1	-1	-1	-1	-1	...	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F	(1, 1)	1	0	2	5	2	...	0	(1, 1)	1	0	2	5	2	...	...	0

Figura 4.14 – Movimento Desativar Carregadeira

A Figura 4.14 mostra o movimento “Desativar_Carregadeira” efetuado na frente 2. A carregadeira de número 6 alocada na frente 2 que estava ativa passa a estar desativada. Observa-se que todas as viagens foram zeradas a partir do momento em que a carregadeira foi desativada.

4.3.5.8 Movimento Inserir_Viagem

O movimento “Inserir_Viagem” consiste em inserir uma viagem para um determinado caminhão, ou seja, seleciona-se uma célula n_{il} da matriz N e adiciona-se uma viagem do caminhão l na frente i .

A Figura 4.15 ilustra este movimento.

Antes do Movimento								Depois do Movimento							
	1	2	3	4	5	...	V		1	2	3	4	5	...	V
1	2	0	1	4	1	...	1	1	2	0	1	5	1	...	1
2	0	3	5	1	0	...	4	2	0	3	5	1	0	...	4
3	-1	-1	-1	-1	-1	...	-1	3	-1	-1	-1	-1	-1	...	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F	1	0	2	5	2	...	0	F	1	0	2	5	2	...	0

Figura 4.15 – Movimento Inserir uma viagem

Nessa figura o caminhão de número 4 realiza 4 viagens à frente 1, inicialmente. Após a implementação desse movimento, este caminhão passa a realizar 5 viagens à frente 1.

4.3.5.9 Movimento Retirar_Viagens

O movimento “Retirar_Viagens” consiste em selecionar um caminhão e retirar todas as viagens alocadas a esse caminhão em todas as frentes.

A Figura 4.16 mostra o movimento “Retirar Viagens” efetuado na frente 5. Observa-se que todas as viagens foram zeradas nessa frente.

Antes do Movimento								Depois do Movimento									
	Carreg	1	2	3	4	5	...	V		Carreg	1	2	3	4	5	...	V
1	(3, 1)	2	0	1	4	1	...	1		(3, 1)	2	0	1	4	0	...	1
2	(6, 1)	3	0	1	0	1	...	0		(6, 1)	3	0	1	0	0	...	0
3	(-1,0)	-1	-1	-1	-1	-1	...	-1		(-1,0)	-1	-1	-1	-1	-1	...	-1
...	...	...	...	...	...	...	...	...		...	...	...	...	...	...	...	...
F	(1, 1)	1	0	2	5	2	...	0		(1, 1)	1	0	2	5	0	...	0

Figura 4.16 – Movimento Retirar Viagens

4.3.5.10 Movimento Trocar_carregadeira

Esse movimento consiste em trocar de carregadeira em uma determinada frente. Inicialmente, procura-se uma carregadeira inativa. Esse procedimento é repetido até que se encontre tal carregadeira ou se atinja um certo número de vezes, dado por pelo produto do número de frentes pelo número de caminhões. Caso exista uma carregadeira inativa, procura-se, agora, uma que esteja ativa e que possua capacidade de produção inferior à inativa. Se encontrar, faz-se a troca das carregadeiras sem trocar as respectivas viagens. Caso não se encontre uma carregadeira inativa, o movimento não é realizado.

A Figura 4.17 ilustra este movimento.

Antes do Movimento								Depois do Movimento							
	Carreg	1	2	3	4	5	... V		Carreg	1	2	3	4	5	... V
1	(3, 1)	2	0	1	4	1	... 1	(4, 1)	2	0	1	4	1	... 1	
2	(4, 0)	0	0	0	0	0	... 0	(3, 0)	0	0	0	0	0	... 0	
3	(-1,0)	-1	-1	-1	-1	-1	... -1	(-1,0)	-1	-1	-1	-1	-1	... -1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
F	(1, 1)	1	0	2	5	2	... 0	(1, 1)	1	0	2	5	2	... 0	

Figura 4.17 – Movimento Trocar Carregadeira

A Figura 4.17 mostra o movimento “Trocar Carregadeira” efetuado nas frentes 1 e 2. A carregadeira de número 4, inativa, alocada à frente 2 é transferida para a frente 1 e a carregadeira 3, alocada à frente 1 e de capacidade inferior à carregadeira 4 é transferida para a frente 2. Com isso, após a troca, a carregadeira 4 passa a ter o *status* de ativa, ao contrário da carregadeira 3 que torna-se inativa.

4.3.6 Métodos de Busca Local Aplicados ao Planejamento de Lavra

As técnicas de busca local, também conhecidas como heurísticas de refinamento em problemas de otimização, são baseadas na noção de vizinhança.

Essa classe de heurística parte de uma solução inicial qualquer e caminha, a cada iteração, de vizinho para vizinho de acordo com a definição de vizinhança adotada.

Para explorar o espaço de soluções do Problema de Alocação Dinâmica de Caminhões em questão, foram utilizados os métodos de busca local mostrados a seguir.

4.3.6.1 Descida com retirada de uma viagem

Este método consiste em aplicar o método da Descida utilizando o movimento “retirar uma viagem”, descrito na seção 4.3.5.1.

A Figura 4.18 ilustra o pseudocódigo desta busca local.

Neste procedimento, para cada caminhão com alguma viagem é retirada uma viagem e analisada sua função de avaliação. A retirada que produzir a melhor função de avaliação é realizada, desde que represente uma melhora global.

Procedimento Descida_tirar1_Viagem(N, Tcv, Tc, S, fo, Cap);
 Variáveis i, l, fo' : inteiro;
 Melhorou: lógico;

Início
 Melhorou $\leftarrow$ verdadeiro;
 Repita
 $fo' \leftarrow fo$; {guardar a solução corrente}
 escolher_melhor_vizinho(fo', i, l);
 se $fo' < fo$ {se melhorou}
 $fo \leftarrow fo'$;
 $n[i, l] \leftarrow n[i, l] - 1$; {tirar uma viagem}
 $tcv[l] \leftarrow tcv[l] - tc[i, l]$; {atualizar tempo caminhão}
 $s[i] \leftarrow s[i] - cap[l]$; {atualizar massa lavrada na frente}
 se minério {atualizar massa lavrada geral de minério ou estéril}
 $t \leftarrow t - Cap[l]$
 senão est $\leftarrow est - Cap[l]$;
 senão
 melhorou $\leftarrow$ falso;
 fim se
 até **não** melhorou;
Fim{ Descida_tirar1_viagem }

Figura 4.18 – Descida_tirar_uma_viagem

4.3.6.2 Descida com retirada de duas viagens

Este método consiste na aplicação do método da Descida utilizando o movimento “retirar duas viagens”, descrito na seção 4.3.5.2.

A Figura 4.19 ilustra o pseudocódigo desta busca local.

Neste procedimento, para cada caminhão com duas ou mais viagens são retiradas duas viagens e analisada sua função de avaliação. A retirada que produzir a melhor função de avaliação é realizada, desde que represente uma melhora global.

```
Procedimento Descida_tirar2_Viagens(N, Tcv, Tc, S, fo, Cap);  
  Variáveis  $i, l, fo'$ : inteiro;  
           Melhorou: lógico;  
Início  
  Melhorou  $\leftarrow$  verdadeiro;  
  Repita  
     $fo' \leftarrow fo$ ; { guardar a solução corrente}  
    escolher_melhor_vizinho( $fo', i, l$ );  
    se  $fo' < fo$  { se melhorou}  
       $fo \leftarrow fo'$ ;  
       $n[i, l] \leftarrow n[i, l] - 2$ ; { tirar uma viagem}  
       $tcv[l] \leftarrow tcv[l] - 2*(tc[i, l])$ ; { atualizar tempo caminhão}  
       $s[i] \leftarrow s[i] - 2*(cap[l])$ ; { atualizar massa lavrada na frente}  
      se minério { atualizar massa lavrada geral de minério ou estéril}  
         $t \leftarrow t - Cap[l]$   
         $senão\ est \leftarrow est - 2*Cap[l]$ ;  
       $senão$   
        melhorou  $\leftarrow$  falso;  
      fim se  
    até não melhorou;  
Fim{ Descida_tirar2_viagens }
```

Figura 4.19 – Descida_tirar_duas_viagens

4.3.6.3 Descida com realocação de uma viagem mantendo o caminhão

Esse método consiste em aplicar o método da Descida utilizando o movimento “realocar uma viagem_mc”, descrito na seção 4.3.5.3.

A Figura 4.20 ilustra o pseudocódigo desta busca local.

```

Procedimento Descida_Realocar_uma_Viagem_mc(N, Tcv, Tc, S, fo, Cap);
  Variáveis  $i, l, i', fo'$ : inteiro;
           Melhorou: lógico;

Início
  Melhorou  $\leftarrow$  verdadeiro;
  Repita
     $fo' \leftarrow fo$ ;           {guardar a solução corrente}
    escolher_melhor_vizinho( $fo', i, l, i'$ );
    se  $fo' < fo$                  {se melhorou}
       $fo \leftarrow fo'$ ;
       $n[i, l] \leftarrow n[i, l] - 1$ ;    {tirar uma viagem}
       $tcv[l] \leftarrow tcv[l] - tc[i, l]$ ; {atualizar tempo caminhão}
       $s[i] \leftarrow s[i] - Cap[l]$ ; {atualizar massa lavrada na frente  $i$ }
      se minério                 {atualizar massa lavrada geral de minério ou estéril}
         $t \leftarrow t - Cap[l]$ 
        senão  $est \leftarrow est - Cap[l]$ ;
       $n[i', l] \leftarrow n[i', l] + 1$ ;    {realocar uma viagem}
       $tcv[l] \leftarrow tcv[l] + tc[i', l]$ ; {atualizar tempo caminhão}
       $s[i'] \leftarrow s[i'] + Cap[l]$ ; {atualizar massa lavrada na frente  $i'$ }
      se minério                 {atualizar massa lavrada geral de minério ou estéril}
         $t \leftarrow t + Cap[l]$ 
        senão  $est \leftarrow est + Cap[l]$ ;
      senão
        melhorou  $\leftarrow$  falso;
      fim se
    até não melhorou;
Fim { Descida_Realocar_uma_Viagem_mc }

```

Figura 4.20 – Descida_realocação_de_viagem_mantendo_caminhão

Neste procedimento, para cada caminhão com alguma viagem é realocada uma viagem para outra frente e analisada sua função de avaliação. Aquela que produzir a melhor função de avaliação é feita, desde que represente uma melhora global.

4.3.6.4 Descida com realocação de uma viagem mantendo a frente

Este método consiste na aplicação do método da Descida utilizando o movimento “realocar uma viagem_mf”, descrito na seção 4.3.5.4.

A Figura 4.21 ilustra o pseudocódigo desta busca local.

```

Procedimento Descida_Realocar_uma_Viagem_mf(N, Tcv, Tc, S, fo, Cap);
    Variáveis  $i, l, l', fo'$ : inteiro;
    Melhorou: lógico;

Início
    Melhorou  $\leftarrow$  verdadeiro;
    Repita
         $fo' \leftarrow fo$ ;           {guardar a solução corrente}
        escolher_melhor_vizinho( $fo', i, l, l'$ );
        se  $fo' < fo$                 {se melhorou}
             $fo \leftarrow fo'$ ;
             $n[i, l] \leftarrow n[i, l] - 1$ ;      {tirar uma viagem}
             $tcv[l] \leftarrow tcv[l] - tc[i, l]$ ; {atualizar tempo caminhão}
             $s[i] \leftarrow s[i] - cap[l]$ ;    {atualizar massa lavrada na frente  $i$ }
            se minério                {atualizar massa lavrada geral de minério ou estéril}
                 $t \leftarrow t - cap[l]$ 
            senão est  $\leftarrow est - cap[l]$ ;
             $n[i, l'] \leftarrow n[i, l'] + 1$ ;    {realocar uma viagem}
             $tcv[l'] \leftarrow tcv[l'] + tc[i, l']$ ; {atualizar tempo caminhão}
             $s[i] \leftarrow s[i] + cap[l']$ ;    {atualizar massa lavrada na frente  $i$ }
            se minério                {atualizar massa lavrada geral de minério ou estéril}
                 $t \leftarrow t - cap[l]$ 
            senão est  $\leftarrow est - cap[l]$ ;
        senão
            melhorou  $\leftarrow$  falso;
    fim se
    até não melhorou;
Fim{ Descida_Realocar_uma_Viagem_mf }

```

Figura 4.21 – Descida_realocação_de_viagem_mantendo_frente

Neste procedimento, para cada frente, é realocada uma viagem de um caminhão para outro, caso exista alguma viagem e analisada sua função de avaliação. Aquela que produzir a melhor função de avaliação é realizada, desde que represente uma melhora global.

4.3.6.5 Descida inserir uma viagem

Este método consiste em aplicar o método da Descida utilizando o movimento “inserir uma viagem”, descrito na seção 4.3.5.8.

A Figura 4.22 ilustra o pseudocódigo desta busca local.


```

Procedimento Descida_Inserir1_Viagem(N, Tcv, Tc, S, fo, Cap);
  Variáveis  $i, l, fo'$ : inteiro;
           Melhorou: lógico;

Início
  Melhorou  $\leftarrow$  verdadeiro;
  Repita
     $fo' \leftarrow fo$ ;           {guardar a solução corrente}
    escolher_melhor_vizinho( $fo', i, l$ );
    se  $fo' < fo$                  {se melhorou}
       $fo \leftarrow fo'$ ;
       $n[i, l] \leftarrow n[i, l] + 1$ ;   {inserir uma viagem}
       $tcv[l] \leftarrow tcv[l] + tc[i, l]$ ; {atualizar tempo caminhão}
       $s[i] \leftarrow s[i] + cap[l]$ ; {atualizar massa lavrada na frente}
      se minério                 {atualizar massa lavrada geral de minério ou estéril}
         $t \leftarrow t + cap[l]$ 
         $senão\ est \leftarrow est + cap[l]$ ;
       $senão$ 
         $melhorou \leftarrow falso$ ;
      fim se
    até não melhorou;
Fim{ Descida_inserir1_viagem }

```

Figura 4.22 – Descida_inserir_uma_viagem

Neste procedimento, para cada caminhão é inserida uma viagem e analisada sua função de avaliação. Aquela que produzir a melhor função de avaliação é realizada, desde que represente uma melhora global.

4.3.6.6 Descida tirar viagens

Este método consiste na aplicação do método da Descida utilizando uma adaptação do movimento “retirar_viagens”, descrito na seção 4.3.5.9.

A Figura 4.23 ilustra o pseudocódigo desta busca local.

```

Procedimento Descida_Tirar_Viagens(N, Tcv, Tc, S, fo, Cap);
  Variáveis  $i, l, fo'$ : inteiro;
           Melhorou: lógico;

Início
  Melhorou  $\leftarrow$  verdadeiro;
  Repita
     $fo' \leftarrow fo$ ;           {guardar a solução corrente}
    escolher_melhor_vizinho( $fo', i, l$ );
    se  $fo' < fo$                  {se melhorou}
       $fo \leftarrow fo'$ ;
      se  $n[i, l] > 0$ 
         $tcv[l] \leftarrow tcv[l] - (n[i, l] * tc[i, l]);$  {atualizar tempo caminhão}
         $s[i] \leftarrow s[i] - (n[i, l] * cap[l]);$  {atualizar massa lavrada na frente}
        se minério               {atualizar massa lavrada geral de minério ou estéril}
           $t \leftarrow t - (n[i, l] * cap[l])$ 
           $senão\ est \leftarrow est - (n[i, l] * cap[l]);$ 
           $n[i, l] := 0;$  {atualizar (zerar) viagens}
         $senão$ 
          melhorou  $\leftarrow$  falso;
      fim se
    até não melhorou;
Fim{ Descida_tirar_viagens }

```

Figura 4.23 – Descida_tirar_viagens

Neste procedimento, para cada caminhão com uma ou mais viagens em uma determinada frente, são retiradas todas as viagens e analisada sua função de avaliação. A retirada que produzir a melhor função de avaliação é realizada, desde que represente uma melhora global.

4.3.6.7 Descida desativar carregadeira

Este método consiste na aplicação do método da Descida utilizando o movimento “desativar_carregadeira”, descrito na seção 4.3.5.7.

A Figura 4.24 ilustra o pseudocódigo desta busca local.

```

Procedimento Descida_Desativar_Carregadeira(N, Tcv, Tc, S, fo, Cap);
  Variáveis  $i, l, fo'$ : inteiro;
             Melhorou: lógico;
Início
  Ncam  $\leftarrow$  n° total de caminhões;
  Melhorou  $\leftarrow$  verdadeiro;
  Repita
    fo'  $\leftarrow$  fo;           {guardar a solução corrente}
    escolher_melhor_vizinho(fo', i);
    se fo' < fo               {se melhorou}
      fo  $\leftarrow$  fo';
      para l de 1 até Ncam
        se n[i, l] > 0
          tcv[l]  $\leftarrow$  tcv[l] - (n[i, l]*tc[i, l]); {atualizar tempo caminhão}
          s[i]  $\leftarrow$  s[i] - (n[i, l]*cap[l]); {atualizar massa lavrada na frente}
          se minério          {atualizar massa lavrada geral de minério ou estéril}
            t  $\leftarrow$  t - (n[i, l]*cap[l])
            senão est  $\leftarrow$  est - (n[i, l]*cap[l]);
            n[i, l]  $\leftarrow$  0; {atualizar (zerar) viagens}
          senão
            melhorou  $\leftarrow$  falso;
        fim se
      até não melhorou;
  Fim{ Descida_Desativar_Carregadeira }

```

Figura 4.24 – Descida_desativar_carregadeira

Neste procedimento, para cada frente onde exista uma carregadeira ativa, são retiradas todas as viagens de todos os caminhões e analisada sua função de avaliação. A retirada que produzir a melhor função de avaliação é realizada, desde que represente uma melhora global.

4.3.6.8 Descida realocar carregadeira

Este método consiste na aplicação do método da Descida utilizando o movimento “realocar_carregadeira”, descrito na seção 4.3.5.5.

A Figura 4.25 ilustra o pseudocódigo desta busca local.

```

Procedimento Descida_Realocar_Carregadeira(N, Tcv, Tc, S, fo, Y, Cap);
  Variáveis  $i, l, fo', aux$ : inteiro;
             Melhorou: lógico;

Início
  Ncam  $\leftarrow$  n° total de caminhões;
  Melhorou  $\leftarrow$  verdadeiro;
  Repita
     $fo' \leftarrow fo$ ;           {guardar a solução corrente}
    escolher_melhor_vizinho( $fo', i, k$ );
    se  $fo' < fo$                 {se melhorou}
       $fo \leftarrow fo'$ ;
    // troca viagens
    para  $l$  de 1 até Ncam
       $aux \leftarrow n[i, l]$ ;
       $n[i, l] \leftarrow n[k, l]$ ;
       $n[k, l] \leftarrow aux$ ;
    // troca carregadeiras
     $aux \leftarrow y[i].N\_carr$  ;
     $y[i].N\_carr \leftarrow y[k].N\_carr$ ;
     $y[k].N\_carr \leftarrow aux$ ;
     $aux \leftarrow Y[i].N\_ativo$  ;
     $y[i].N\_ativo \leftarrow y[k].N\_ativo$ ;
     $y[k].N\_ativo \leftarrow aux$ ;
    Gera_nova_massa_lavrável_por_frente;
    Gera_novo_tempo_por_caminhão;
    Gera_total_de_massa_lavrável;
    Gera_novas_percentagens;
  senão
    melhorou  $\leftarrow$  falso;
  fim se
  até não melhorou;
Fim{ Descida_Realocar_Carregadeira }

```

Figura 4.25 – Descida_realocar_carregadeira

Neste procedimento, para cada frente onde exista uma carregadeira ativa, é feita a inversão dessa carregadeira com as seguintes trocando também as viagens de cada caminhão e analisada sua função de avaliação. A inversão que produzir a melhor função de avaliação é realizada, desde que represente uma melhora global.

4.3.7 Perturbações

Para tentar escapar de ótimos locais, ainda distante do ótimo global, o método *Iterated Local Search*, que será descrito na seção 4.3.8, utiliza-se de mecanismos de

perturbações. Foram desenvolvidas as seguintes perturbações para o método ILS aplicado ao planejamento de lavra:

4.3.7.1 Perturbação “Retirar uma viagem”

Nessa perturbação, é utilizado o movimento “Retirar uma Viagem”, descrito na seção 4.3.5.1. São gerados aleatoriamente dois valores (i,l) . O primeiro se refere à frente i e o segundo, ao caminhão l . Se houver viagens alocadas para o caminhão l , é retirada uma viagem da célula n_{il} da matriz N ; caso contrário, o procedimento é repetido até que se encontre um caminhão l com pelo menos uma viagem à frente i .

A Figura 4.26 mostra o pseudocódigo referente à perturbação “Retirar uma Viagem” no qual as frentes e os caminhões são escolhidos de forma aleatória até que se encontre viagens alocadas.

Procedimento Perturbação_Retirar_uma_viagem(N);
 Variáveis i,l :inteiro;
Início
 MaxFr $\leftarrow$ n° máximo de frentes;
 Ncam $\leftarrow$ n° total de caminhões;
 Repita
 $i \leftarrow$ aleatório(MaxFr);
 $l \leftarrow$ aleatório(Ncam);
 até $n[i,l] > 0$;
 $n[i,l] \leftarrow n[i,l] - 1$;
 se minério {atualizar massa lavrada geral de minério ou estéril}
 $t \leftarrow t - \text{cap}[l]$
 senão est $\leftarrow$ est – cap[l];
Fim{ Perturbação_Retirar_uma_viagem }

Figura 4.26 – Perturbação_Retirar_uma_viagem

4.3.7.2 Perturbação “Retirar duas viagens”

Esta perturbação utiliza-se do movimento “Retirar duas Viagens” descrito em 4.3.5.2, análogo ao mostrado no item 4.3.7.1. São gerados dois valores (i,l) , de forma aleatória, representando a frente i e o caminhão l . Se houver viagens alocadas para esse

caminhão, são retiradas duas viagens da célula n_{il} da matriz N. Caso contrário, são gerados outros valores (i,l) até que se encontre uma célula n_{il} que possua pelo menos duas viagens alocadas.

A Figura 4.27 mostra o pseudocódigo referente à perturbação “Retirar duas Viagens”.

```

Procedimento Perturbação_Retirar_duas_viagens(N);
  Variáveis  $i,l$ :inteiro;
  Início
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncam  $\leftarrow$  n° total de caminhões;
    Repita
       $i \leftarrow$  aleatório(MaxFr);
       $l \leftarrow$  aleatório(Ncam);
    até  $n[i,l] \geq 2$ ;
     $n[i,l] \leftarrow n[i,l] - 2$ ;
    se minério      {atualizar massa lavrada geral de minério ou estéril}
       $t \leftarrow t - \text{cap}[l]$ 
    senão est  $\leftarrow$  est  $- 2 * \text{cap}[l]$ ;
  Fim {Perturbação_Retirar_duas_viagens}

```

Figura 4.27 – Perturbação Retirar_duas_viagens

4.3.7.3 Perturbação “Realocar_Viagem_mc”

Na perturbação “Realocar uma Viagem mantendo caminhão” utiliza-se o movimento “Realocar_Viagem_mc”, descrito em 4.3.5.3. Geram-se inicialmente dois valores (i,l) , aleatoriamente. O primeiro diz respeito à frente i e o segundo, ao caminhão l . Para o primeiro valor i , verifica-se se existe um equipamento de carga alocado à frente i e, também, se este está ativo. Na inexistência de um equipamento satisfazendo essas condições, o procedimento é repetido até que a condição seja satisfeita. De posse do segundo valor l , verifica-se se existe viagem alocada à célula n_{il} da matriz N. Inexistindo um veículo com pelo menos uma viagem, procura-se aleatoriamente um novo veículo. Este procedimento é repetido até que seja encontrado um veículo com pelo menos uma viagem, interrompendo-se a busca caso ela não seja bem sucedida após um determinado número de iterações. Nesta última situação, nova frente é gerada

aleatoriamente. Este procedimento se encerra quando é gerada uma frente i e um caminhão l tal que $n_{il} > 0$. Em seguida, gera-se um outro valor, aleatoriamente, correspondente a uma outra frente i' , diferente da primeira i e que possua equipamento de transporte compatível com o equipamento de carga. Caso contrário, a frente i' é gerada novamente até que as condições acima sejam satisfeitas.

De posse de i , l e i' , ou seja, das frentes i e i' e do caminhão l , realoca-se uma viagem da célula n_{il} para a célula $n_{i'l}$, pertencentes à matriz N , isto é, o caminhão l deixa de fazer uma viagem à frente i para realizá-la na frente i' . Esse movimento poderá ser desfeito caso a quantidade total de massa lavrada da frente i' ($s_{i'}$) seja superior à quantidade máxima de produção da carregadeira ou, então, o tempo total utilizado pelo caminhão l para realizar todas as viagens seja superior a uma hora.

A Figura 4.28 mostra o pseudocódigo referente à perturbação “Realocar uma Viagem mantendo o caminhão”.

```

Procedimento Perturbação_Realocar_uma_Viagem_mc(N, S, Cu, Cap, Tcv, Y );
    Variáveis  $i, l, i'$ , cont: inteiro;
    Achou: lógico;

Início
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncam  $\leftarrow$  n° total de caminhos;
    Cont  $\leftarrow$  0;
    Achou  $\leftarrow$  falso;
Repita
    Repita
         $i \leftarrow$  aleatório(MaxFr);
    até encontrar carregadeira alocada e ativa;
Repita
     $l \leftarrow$  aleatório(Ncam);
    cont  $\leftarrow$  cont + 1;
    se  $n[i, l] > 0$  então Achou  $\leftarrow$  verdadeiro;
    até Achou ou cont > MaxFr * Ncam;
até Achou;
Repita
     $i' \leftarrow$  aleatório(MaxFr);
    até ( $n[i', l] < -1$ ) e ( $i' < i$ );
     $n[i, l] \leftarrow n[i, l] - 1$ ;
    se minério {atualizar massa lavrada geral de minério ou estéril}
         $t \leftarrow t - \text{cap}[l]$ 
        senão est  $\leftarrow$  est - cap[l];
     $n[i', l] \leftarrow n[i', l] + 1$ ;
    se minério {atualizar massa lavrada geral de minério ou estéril}
         $t \leftarrow t + \text{cap}[l]$ 
        senão est  $\leftarrow$  est + cap[l];
    se ( $s[i'] > \text{cu}[Y[i']]$ ) ou ( $\text{tcv}[l] > 60$ )
        {desfaz movimento}
         $n[i, l] \leftarrow n[i, l] + 1$ ;
        se minério {atualizar massa lavrada geral de minério ou estéril}
             $t \leftarrow t + \text{cap}[l]$ 
            senão est  $\leftarrow$  est + cap[l];
         $n[i', l] \leftarrow n[i', l] - 1$ ;
        se minério {atualizar massa lavrada geral de minério ou estéril}
             $t \leftarrow t - \text{cap}[l]$ 
            senão est  $\leftarrow$  est - cap[l];
Fim { Perturbação_Realocar_uma_Viagem_mc }

```

Figura 4.28 – Perturbação Realocar uma Viagem mantendo o caminho

4.3.7.4 Perturbação “Realocar_Viagem_mf”

A Figura 4.29 mostra o pseudocódigo referente à perturbação “Realocar uma Viagem mantendo a frente”.

```
Procedimento Perturbação_Realocar_uma_Viagem_mf(N, S, Cu, Cap, Tcv, Y);  
  Variáveis i, l, l', cont: inteiro;  
  Achou: lógico;  
  
  Início  
    MaxFr ← n° máximo de frentes;  
    Ncam ← n° total de caminhões;  
    Cont ← 0;  
    Achou ← falso;  
  Repita  
    Repita  
      i ← aleatório(MaxFr);  
    até encontrar carregadeira alocada e ativa;  
  Repita  
    l ← aleatório(Ncam);  
    cont ← cont + 1;  
    se n[i, l] > 0 então Achou ← verdadeiro;  
  até (Achou) ou (cont > (MaxFr * Ncam));  
  até Achou;  
  Repita  
    l' ← aleatório(Ncam);  
  até (n[i, l'] <> -1) e (l <> l');  
  n[i, l] ← n[i, l] - 1;  
  se minério {atualizar massa lavrada geral de minério ou estéril}  
    t ← t - cap[l]  
    senão est ← est - cap[l];  
  n[i, l'] ← n[i, l'] + 1;  
  se minério {atualizar massa lavrada geral de minério ou estéril}  
    t ← t + cap[l]  
    senão est ← est + cap[l];  
  se (s[i] > cu[y[i].N_carr]) ou (Tcv[ll] > 60)  
    {desfaz movimento}  
    n[i, l] ← n[i, l] + 1;  
    se minério {atualizar massa lavrada geral de minério ou estéril}  
      t ← t + cap[l]  
      senão est ← est + cap[l];  
    n[i, l'] ← n[i, l'] - 1;  
    se minério {atualizar massa lavrada geral de minério ou estéril}  
      t ← t - cap[l]  
      senão est ← est - cap[l];  
Fim{ Perturbação_Realocar_uma_Viagem_mf }
```

Figura 4.29 – Perturbação Realocar uma Viagem mantendo a frente

Esta perturbação utiliza o movimento “Realocar_Viagem_mf” descrito em 4.3.5.4. De posse de uma frente i com uma carregadeira alocada e ativa e de dois caminhões l e l' , é realocada uma viagem da célula n_{il} para a célula $n_{il'}$, pertencentes à matriz N , isto é, uma viagem é transferida do caminhão l para o caminhão l' , mantendo-se a frente i .

4.3.7.5 Perturbação “Retirar_Viagem_2x”

Na perturbação “Retirar uma Viagem, duas vezes”, todo o procedimento explicitado no item 4.3.7.1 é executado duas vezes.

4.3.7.6 Perturbação “Retirar_Duas_Viagens_2x”

Essa perturbação é similar ao descrito no item 4.3.7.2, bastando repetir todo o procedimento duas vezes.

4.3.7.7 Perturbação “Realocar_Viagem_mc_2x”

Para se construir esta perturbação, basta repetir todo o procedimento descrito no item 4.3.7.3, duas vezes.

4.3.7.8 Perturbação “Realocar_Viagem_mf_2x”

Na perturbação “Realocar uma Viagem mantendo a frente, duas vezes”, é necessário repetir o procedimento descrito no item 4.3.7.4, duas vezes.

4.3.7.9 Perturbação “Realocar_Carregadeira”

Na perturbação “Realocar_Carregadeira” utiliza-se o movimento descrito na seção 4.3.5.5. Gera-se um valor referente à frente i , verificando-se se há carregadeira alocada e se a mesma encontra-se ativa. Caso isso não ocorra, gera-se novamente um

novo valor de i até que as condições prévias sejam satisfeitas. Em seguida, gera-se um segundo valor i' referente à outra frente, diferente do primeiro.

De posse dos valores de i e i' , ou seja, das frentes i e i' , efetua-se a troca entre as carregadeiras, isto é, a carregadeira que se encontra na frente i é realocada à frente i' e, caso haja carregadeira alocada à frente i' , esta é transferida à frente i . Também efetuam-se as trocas referentes aos números de viagens de cada caminhão alocados às duas frentes.

A Figura 4.30 mostra o pseudocódigo referente à perturbação Realocar_Carregadeira.

```
Procedimento Perturbação_Realocar_Carregadeira(N, S, Cu, Tcv, Y);
  Variáveis  $i, l, ii, aux$ : inteiro;
Início
  MaxFr  $\leftarrow$  n° máximo de frentes;
  Ncam  $\leftarrow$  n° total de caminhões;
  Repita
     $i \leftarrow$  aleatório(MaxFr);
    até encontrar carregadeira alocada e ativa;
     $ii \leftarrow$  aleatório(MaxFr);
    {troca viagens}
    Para  $l \leftarrow 1$  até Ncam faça
       $aux \leftarrow n[i, l]$ ;
       $n[i, l] \leftarrow n[ii, l]$ ;
       $n[ii, l] \leftarrow aux$ ;
    Fim para;
    {troca carregadeiras}
     $aux \leftarrow y[i].n\_carr$ ;
     $y[i].n\_carr \leftarrow y[ii].n\_carr$ ;
     $y[ii].n\_carr \leftarrow aux$ ;
Fim{ Perturbação_Realocar_Carregadeira }
```

Figura 4.30 – Perturbação Realocar Carregadeira

4.3.7.10 Perturbação “Desativar_Carregadeira”

Na perturbação “Realocar_Carregadeira” utiliza-se o movimento descrito na seção 4.3.5.7. Gera-se um valor referente à carregadeira k , verificando-se se a mesma encontra-se ativa. Caso isso não ocorra, gera-se novamente um novo valor de k até que

as condições acima sejam satisfeitas. Em seguida, localiza-se em qual frente i a carregadeira k está operando. De posse desses valores, a carregadeira k é desativada e são zeradas todas as viagens de cada caminhão j na frente i onde opera a carregadeira k .

A Figura 4.31 mostra o pseudocódigo referente à perturbação Desativar_Carregadeira.

```

Procedimento Perturbação_Desativar_Carregadeira(N, S, Cu, Cap, Tcv, Y);
  Variáveis  $i, l, ii, k$ : inteiro;
            achou: lógico;

  Início
    Ncarreg  $\leftarrow$  n° total de carregadeiras;
    MaxFr  $\leftarrow$  n° máximo de frentes;
    Ncam  $\leftarrow$  n° total de caminhões;
    achou  $\leftarrow$  falso;
    Repita
       $k \leftarrow$  aleatório(Ncarreg);
       $i \leftarrow 1$ ;
      Enquanto  $i \leq \text{MaxFr}$  faça
        se  $y[i].N_{\text{carr}} = k$ 
          se  $y[i].N_{\text{ativo}} = 1$            { encontrou carregadeira ativa }
             $y[i].N_{\text{ativo}} = 0$ ;       { muda status para inativa }
             $ii \leftarrow i$ ;
            achou  $\leftarrow$  verdadeiro
          fim se
         $i \leftarrow i + 1$ ;
      fim se
    Fim enquanto
    até achou = verdadeiro;
     $i \leftarrow ii$ ;
    { retirar viagens }
    Para  $l \leftarrow 1$  até Ncam faça
      se  $n[i, l] > 0$ 
        se minério
           $t \leftarrow t - (n[i, l] * \text{cap}[l])$ 
           $\text{senão est} \leftarrow \text{est} - (n[i, l] * \text{cap}[l]);$ 
        fim se
       $n[i, l] \leftarrow 0$ ;
    fim se
  Fim para
Fim { Perturbação_Desativar_Carregadeira }

```

Figura 4.31 – Perturbação Desativar Carregadeira

4.3.7.11 Perturbação “Ativar_Carregadeira”

Na perturbação “Ativar_Carregadeira” utiliza-se o movimento descrito na seção 4.3.5.6. Gera-se um valor referente à carregadeira k , verificando-se se a mesma encontra-se inativa. Caso isso não ocorra, gera-se novamente um novo valor de k até que se encontre uma carregadeira inativa ou, um certo número de vezes, caso não exista nenhuma carregadeira inativa.

De posse desses valores, a carregadeira k é ativada e os caminhões já podem ser deslocados para esta frente de trabalho.

A Figura 4.32 mostra o pseudocódigo referente à perturbação Ativar_Carregadeira.

```
Procedimento Perturbação_Ativar_Carregadeira(N, S, Cu, Tcv, Y);  
  Variáveis  $i, l, k, nv, x$ : inteiro;  
           achou: lógico;  
Início  
  Ncarreg  $\leftarrow$  n° total de carregadeiras;  
  MaxFr  $\leftarrow$  n° máximo de frentes;  
  Ncam  $\leftarrow$  n° total de caminhões;  
  nv  $\leftarrow$  MaxFr * Ncam;  
  x  $\leftarrow$  0;  
  achou  $\leftarrow$  falso;  
  Repita  
    x  $\leftarrow$  x + 1;  
    k  $\leftarrow$  aleatório(Ncarreg);  
    i  $\leftarrow$  1;  
    Enquanto i  $\leq$  MaxFr faça  
      se y[i].N_carr = k  
        se y[i].N_ativo = 0      {encontrou carregadeira inativa }  
          y[i].N_ativo = 1;      {muda status para ativa}  
          achou  $\leftarrow$  verdadeiro;  
        fim se  
      i  $\leftarrow$  MaxFr;  
    fim se  
    i  $\leftarrow$  i + 1;  
  Fim enquanto  
  até (achou = verdadeiro) ou (x > nv);  
Fim{ Perturbação_Ativar_Carregadeira }
```

Figura 4.32 – Perturbação Ativar Carregadeira

4.3.7.12 Perturbação “Trocar_Carregadeira”

Na perturbação “Trocar_Carregadeira” utiliza-se o movimento descrito na seção 4.3.5.10. Gera-se um valor referente à carregadeira k , verificando-se se a mesma encontra-se inativa. Caso isso não ocorra, gera-se novamente um novo valor de k até que se encontre uma carregadeira inativa ou que se atinja um certo número de vezes, dado pelo produto do número de frentes pelo número de caminhões. Nesta última situação, a perturbação não é realizada.

Encontrada a carregadeira inativa, armazena-se a frente onde exista uma carregadeira ativa com capacidade inferior à inativa. De posse dessa frente e havendo compatibilidade entre a carregadeira e os caminhões alocados à ela, faz-se a troca entre as duas carregadeiras sem trocar as viagens dos caminhões.

A Figura 4.33 mostra o pseudocódigo referente à perturbação Ativar_Carregadeira.

```

Procedimento Perturbação_Trocar_Carregadeira(N, S, Cu, Y, Compatib);
  Variáveis  $i, i', i'', l, k, nv, x, aux$ : inteiro;
           ok, achou: lógico;

Início
  Ncarreg  $\leftarrow$  n° total de carregadeiras;
  MaxFr  $\leftarrow$  n° máximo de frentes;
  Ncam  $\leftarrow$  n° total de caminhões;
  nv  $\leftarrow$  MaxFr * Ncam;
  x  $\leftarrow$  0;
  achou  $\leftarrow$  falso;
  Repita
    x  $\leftarrow$  x + 1;
    k  $\leftarrow$  aleatório(Ncarreg);
    i  $\leftarrow$  1;
    Enquanto i  $\leq$  MaxFr faça
      se y[i].N_carr = k
        se y[i].N_ativo = 0      {encontrou carregadeira inativa }
          i'  $\leftarrow$  i;
          achou  $\leftarrow$  verdadeiro;
        fim se
        i  $\leftarrow$  MaxFr;
      fim se
      i  $\leftarrow$  i + 1;
    Fim enquanto
  até (achou = verdadeiro) ou (x > nv);
  se x > nv abandone;
  nv  $\leftarrow$  1;
  Enquanto nv  $\leq$  Ncarreg faça
    i  $\leftarrow$  aleatório(MaxFr);
    se (y[i].N_carr  $\neq$  -1) e (Cu[y[i].N_carr] < Cu[y[i']].N_carr)
      se y[i].N_Ativo = 1
        i''  $\leftarrow$  i;
        ok  $\leftarrow$  falso;
        verifica_compatibilidade(ok, i', i'', Y, N, Compatib);
        se ok então
          // troca as carregadeiras
          aux  $\leftarrow$  y[i'].N_carr;
          y[i'].N_carr  $\leftarrow$  Y[i''].N_carr;
          y[i''].N_carr  $\leftarrow$  aux;
        fim se;
        nv  $\leftarrow$  Ncarreg;
      fim se
    fim se
    nv  $\leftarrow$  nv + 1;
  Fim enquanto;
Fim { Perturbação_Trocar_Carregadeira }

```

Figura 4.33 – Perturbação Trocar Carregadeira

4.3.7.13 Perturbação “Tirar_Viagens”

Nesta perturbação, é utilizado o movimento “Retirar_Viagens”, descrito na seção 4.3.5.9. É gerado aleatoriamente um valor l que se refere a um caminhão l . São retiradas todas as viagens alocadas para o caminhão l em todas as frentes.

A Figura 4.34 mostra o pseudocódigo referente à perturbação “Tirar_Viagens”.

```
Procedimento Perturbação_Tirar_Viagens(N);  
  Variáveis  $i, l$ : inteiro;  
Início  
  MaxFr  $\leftarrow$  n° máximo de frentes;  
  Ncam  $\leftarrow$  n° total de caminhões;  
   $l \leftarrow$  aleatório(Ncam);  
  Para  $i \leftarrow 1$  até MaxFr faça  
    se ( $n[i, l] > 0$ ) e ( $y[i].N\_ativo = 1$ )  
      então  
        se minério  
          então  $t \leftarrow t - (n[i, l] * cap[l])$   
          senão  $est \leftarrow est - (n[i, l] * cap[l])$   
        fim se  
         $n[i, l] \leftarrow 0$ ;  
      fim se  
    Fim para  
Fim{ Perturbação_Tirar_Viagens }
```

Figura 4.34 – Perturbação_tirar_viagens

4.3.7.14 Perturbação “Inserir_viagem”

Nesta perturbação, é utilizado o movimento “Inserir_Viagem”, descrito na seção 4.3.5.8. São gerados aleatoriamente dois valores (i, l) . O primeiro se refere à frente i e o segundo, ao caminhão l . Se houver carregadeira ativa nesta frente i e esta for compatível com o caminhão l , é inserida uma viagem da célula n_{il} da matriz N ; caso contrário, o procedimento é repetido até que se encontre uma frente onde haja uma carregadeira trabalhando, ou se atinja um número de vezes, dado pelo produto do número de frentes pelo número de caminhões. Neste último caso, a perturbação não é realizada.

A Figura 4.35 mostra o pseudocódigo referente à perturbação “Inserir uma Viagem”.


```

Procedimento Perturbação_Inserir_uma_viagem(N,Y);
  Variáveis  $i,l$ :inteiro;
Início
  MaxFr  $\leftarrow$  n° máximo de frentes;
  Ncam  $\leftarrow$  n° total de caminhões;
  vz  $\leftarrow$  MaxFr * Ncam;
  x  $\leftarrow$  0;
  Repita
    x  $\leftarrow$  x+1;
     $i \leftarrow$  aleatório(MaxFr);
     $l \leftarrow$  aleatório(Ncam);
  até (( $n[i,l] \geq 0$ ) e ( $y[i].N\_Ativo=1$ )) ou ( $x = vz$ );
  se  $x=vz$ 
    // incompatibilidade entre caminhão e carregadeira
    Abandone;
  fim se;
   $n[i,l] \leftarrow n[i,l] + 1$ ;
  se minério {atualizar massa lavrada geral de minério ou estéril}
     $t \leftarrow t + cap[l]$ 
  senão est  $\leftarrow$  est +  $cap[l]$ ;
  fim se
Fim{ Perturbação_Inserir_uma_viagem }

```

Figura 4.35 – Perturbação_Retirar_uma_viagem

4.3.8 Iterated Local Search (ILS)

A Figura 4.36 ilustra o pseudocódigo referente ao método *Iterated Local Search* adaptado ao Problema de Alocação Dinâmica de Caminhões.

```
Procedimento ILS(fo,N,S,Qu,Cu,Cap,Tc,Tcv,Y,itermax, tempolimito);  
  Variáveis iter,nível, internivelsemmelhora:inteiro;  
           fo':real; {função objetivo}  
           s':arranjo inteiro;  
  
  Início  
    Iter←0; nível←1;  
    MaxFr←nº máximo de frentes;  
    Ncam← nº total de caminhões;  
    Enquanto (iter < itermax e tempo de processamento < tempolimito) faça  
      Internivelsemmelhora←0;  
      Repita  
        s' ← s;  
        fo' ← fo;  
        perturbação(s',nível,Qu,N,Y,Cap,Cu);  
        buscalocal_descidas(fo',s',Qu,N,Y,Cap,Cu);  
        se fo' < fo  
          então fo ← fo';  
              nível ← 1;  
              internivelsemmelhora ← 0;  
              iter ← 0;  
              s ← s';  
        senão internivelsemmelhora ← internivelsemmelhora + 1;  
      fim-se;  
      até (internivelsemmelhora > (10/100 * MaxFr* Ncam) );  
      nível ← nível + 1;  
      iter ← iter + 1;  
    fim-enquanto;  
Fim{ ILS}
```

Figura 4.36 – *Iterated Local Search*

Na adaptação proposta para o ILS, o método parte de uma solução inicial gerada conforme seção 4.3.3 e pára quando um tempo limite de processamento é alcançado ou quando o número de iterações atingir um valor limiar (*Itermax*). Para cada iteração, faz-se uma “perturbação” na solução corrente e, logo após, uma “busca local”. Em seguida, verifica-se se houve melhora no valor da função de avaliação (critério de aceitação). Em havendo, guarda-se esta solução como a melhor até o presente momento e volta-se ao

menor nível de perturbação. Não havendo, a solução não é armazenada e, neste caso, aplica-se a perturbação no nível corrente à solução atual até que não haja melhora no valor da função de avaliação (no caso, 10% do resultado de “nº de frentes $\times$ nº de caminhões”). Decorrido esse número de iterações sem melhora, o nível de perturbação é incrementado.

A Figura 4.37 ilustra o pseudocódigo referente aos níveis de perturbação aplicados ao método ILS descrito anteriormente.

Procedimento Perturbação(S',nível,Qu,N,Y,Cu,Cap);

Início

Caso nível seja

- 1: perturbação_cr(s',Qu,N,Y, Cu,Cap);
- 2: perturbação_n1(s',Qu,N,Y, Cu,Cap);
- 3: perturbação_n2(s',Qu,N,Y, Cu,Cap);
- 4: perturbação_nv(s',Qu,N,Y, Cu,Cap);
- 5: perturbação_n3(s',Qu,N,Y, Cu,Cap);
- 6: perturbação_acr(s',Qu,N,Y, Cu,Cap);
- 7: perturbação_ad(s',Qu,N,Y, Cu,Cap);
- 8: perturbação_ad(s',Qu,N,Y, Cu,Cap);
- 9: perturbação_ad(s',Qu,N,Y, Cu,Cap);
- 10: perturbação_ad(s',Qu,N,Y, Cu,Cap);

senão

- perturbação_n2(s',Qu,N,Y, Cu,Cap);
- perturbação_n3(s',Qu,N,Y, Cu,Cap);

fim-caso;

Fim{ Perturbação }

Figura 4.37 – Níveis de perturbação do ILS

O procedimento “Perturbação” desenvolvido no método ILS e descrito anteriormente consiste na aplicação de vários níveis de perturbação, as quais são dispostas de forma que as de menores índices são as mais fracas no sentido de complexidade computacional. Após a aplicação da perturbação de nível 10 são aplicadas consecutivamente as perturbações perturbação_n2 e perturbação_n3, as quais estabelecem um mecanismo de diversificação para o algoritmo.

As figuras 4.38 a 4.44 ilustram os pseudocódigos referentes à cada nível de perturbação.

A Figura 4.38 mostra o pseudocódigo referente ao nível 1 de perturbação.

Procedimento Perturbação_cr(S',Qu,N,Y,Cu,Cap); Início {Perturbação: desativar uma carregadeira} movimento_desativar1_carregadeira(s',Qu,N,Y, Cu,Cap); Fim{ Perturbação_cr }

Figura 4.38 – Perturbação desativar uma carregadeira

A perturbação de nível 1 consiste em aplicar o movimento desativar uma carregadeira. A perturbação gerada é resultante da aplicação deste movimento.

A Figura 4.39 mostra o pseudocódigo referente ao nível 2 de perturbação.

Procedimento Perturbação_n1(S',Qu,N,Y, Cu,Cap); Var x:inteiro; Início {Perturbações: 1- tirar uma viagem; 2- tirar duas viagens; 3- realocar uma viagem mantendo a frente; 4- realocar uma viagem mantendo o caminhão; } x ← aleatório(4); Caso x seja 1: movimento_tirar1_viagem(s',Qu,N,Y, Cu,Cap); 2: movimento_tirar2_viagens(s',Qu,N,Y, Cu,Cap); 3: movimento_realocar1_viagem_mf(s',Qu,N,Y, Cu,Cap); 4: movimento_realocar1_viagem_mc(s',Qu,N,Y, Cu,Cap); fim-caso; Fim{ Perturbação_n1 }

Figura 4.39 – Perturbação_n1

A perturbação de nível 2 consiste em aplicar um dos 4 movimentos a seguir: (a) tirar uma viagem; (b) tirar duas viagens; (c) realocar uma viagem mantendo a frente e (d) realocar uma viagem mantendo o caminhão. Em seguida, é gerado um número (entre 1 e 4) aleatoriamente referente a um dos movimentos citados acima. A perturbação gerada é resultante da aplicação do movimento escolhido.

A Figura 4.40 mostra o pseudocódigo referente ao nível 3 de perturbação.

```
Procedimento Perturbação_n2(S',Qu,N,Y, Cu,Cap);  
  Var x:inteiro;  
  Início  
    {Perturbações:  
      1- tirar uma viagem, duas vezes;  
      2- tirar duas viagens, duas vezes;  
      3- realocar uma viagem mantendo a frente, duas vezes;  
      4- realocar uma viagem mantendo o caminhão, duas vezes; }  
    x ← aleatório(4);  
    Caso x seja  
      1: movimento_tirar1_viagem_2x(s',Qu,N,Y, Cu,Cap);  
      2: movimento_tirar2_viagens_2x (s',Qu,N,Y, Cu,Cap);  
      3: movimento_realocar1_viagem_mf_2x (s',Qu,N,Y, Cu,Cap);  
      4: movimento_realocar1_viagem_mc_2x (s',Qu,N,Y, Cu,Cap);  
    fim-caso;  
  Fim{ Perturbação_n2 }
```

Figura 4.40 – Perturbação_n2

A perturbação de nível 3 consiste em se aplicar um dos 4 movimentos a seguir: (a) tirar uma viagem (duas vezes); (b) tirar duas viagens (duas vezes); (c) realocar uma viagem mantendo a frente (duas vezes) e (d) realocar uma viagem mantendo o caminhão (duas vezes). Em seguida, é gerado aleatoriamente um número entre 1 e 4 referente a um dos movimentos citados acima. A perturbação gerada é resultante da aplicação do movimento escolhido.

A Figura 4.41 mostra o pseudocódigo referente ao nível 4 de perturbação.

```
Procedimento Perturbação_nv(S',Qu,N,Y, Cu,Cap);  
  Início  
    {Perturbação: tirar viagens}  
    movimento_tirar_viagens(s',Qu,N,Y, Cu,Cap);  
  Fim{ Perturbação_nv }
```

Figura 4.41 – Perturbação tirar todas as viagens

A perturbação de nível 4 consiste em aplicar o movimento tirar todas as viagens. A perturbação gerada é resultante da aplicação deste movimento.

A Figura 4.42 mostra o pseudocódigo referente ao nível 5 de perturbação.

```
Procedimento Perturbação_n3(S',Qu,N,Y, Cu,Cap);  
  Var  $k,k',l$ ,aux:inteiro;  
Início  
  {movimento utilizado: realocar uma carregadeira, mudando a frente}  
  repita  
     $k \leftarrow \text{aleatório}(\text{maxFr})$ ;  
  até  $y[k] <> -1$ ; {existir carregadeira alocada e ativa}  
  repita  
     $k' \leftarrow \text{aleatório}(\text{maxFr})$ ;  
  até  $k <> k'$ ;  
  {troca viagens}  
  para  $l \leftarrow 1$  até Ncam faça  
    aux  $\leftarrow n[k,l]$ ;  
     $n[k,l] \leftarrow n[k',l]$ ;  
     $n[k',l] \leftarrow \text{aux}$ ;  
  fim para  
  {troca carregadeiras}  
  aux  $\leftarrow y[k]$ ;  
   $y[k] \leftarrow y[k']$ ;  
   $y[k'] \leftarrow \text{aux}$ ;  
  Gera_nova_massa_lavrável;  
  Gera_novo_tempo_por_caminhão;  
  Gera_total_massa_lavrável;  
  Calcula_novas_percentagens;  
Fim{ Perturbação_n3 }
```

Figura 4.42 – Perturbação_n3

A perturbação de nível 5 consiste em se aplicar o movimento “realocar uma carregadeira, mudando a frente”.

Nessa perturbação, geram-se inicialmente duas frentes k e k' , sendo k diferente de k' . É necessário que exista carregadeira alocada e ativa pelo menos na frente k . Em seguida, faz-se a realocação das viagens e das carregadeiras.

A Figura 4.43 mostra o pseudocódigo referente ao nível 6 de perturbação.

Procedimento Perturbação_acr(S',Qu,N,Y, Cu,Cap); Início {Perturbação: ativar uma carregadeira} movimento_ativar1_carregadeira(s',Qu,N,Y, Cu,Cap); Fim{ Perturbação_acr }

Figura 4.43 – Perturbação ativar uma carregadeira

A perturbação de nível 6 consiste em aplicar o movimento ativar uma carregadeira. A perturbação gerada é resultante da aplicação deste movimento.

A Figura 4.44 mostra o pseudocódigo referente aos nível 7, 8, 9 e 10 de perturbação.

Procedimento Perturbação_ad(S',Qu,N,Y, Cu,Cap); Início {Perturbação: trocar uma carregadeira ativa por uma inativa invertendo as funções} movimento_trocar_carregadeira(s',Qu,N,Y, Cu,Cap); Fim{ Perturbação_ad }

Figura 4.44 – Perturbação trocar carregadeira ativa/inativa

As perturbações de nível 7, 8, 9 e 10 consistem em aplicar o movimento trocar uma carregadeira ativa por uma inativa de capacidade de produção maior. A perturbação gerada é resultante da aplicação deste movimento.

A Figura 4.45 ilustra o pseudocódigo referente ao método de busca local (VND) usado para refinar uma solução do método ILS.

```

Procedimento VND(fo',S',Qu,N,Y, Cu,Cap);
  Var k, nv:inteiro;
      fo'':real;
Início
  k ← 1;
  nv ← 8; {quantidade de descidas}
  fo'' ← fo';
  enquanto k <= nv faça
    caso k seja
      1:Descida_tirar1_viagem(fo',S',Qu,N,Y, Cu,Cap);
      2:Descida_tirar_viagens(fo',S',Qu,N,Y, Cu,Cap);
      3:Descida_inserir1_viagem(fo',S',Qu,N,Y, Cu,Cap);
      4:Descida_realocar1_viagem_mf(fo',S',Qu,N,Y, Cu,Cap);
      5:Descida_realocar1_viagem_mc(fo',S',Qu,N,Y, Cu,Cap);
      6:Descida_realocar1_carregadeira(fo',S',Qu,N,Y, Cu,Cap);
      7:Descida_desativar1_carregadeira(fo',S',Qu,N,Y, Cu,Cap);
      8:Descida_tirar2_viagens(fo',S',Qu,N,Y, Cu,Cap);
    fim caso;
  // VND
  se fo' < fo''
    então k ← 1;
        fo'' ← fo';
    senão k ← k + 1;
  fim se;
  fim enquanto;
Fim{ VND }

```

Figura 4.45 – Algoritmo VND

Nesse procedimento são executados oito tipos diferentes de busca local, todos já descritos no item 4.3.6:

- (a) descida_tirar1_viagem;
- (b) descida_tirar_viagens;
- (c) descida_inserir1_viagem;
- (d) descida_realocar1_viagem_mf;
- (e) descida_realocar1_viagem_mc;
- (f) descida_realocar1_carregadeira;
- (g) descida_desativar1_carregadeira;
- (h) descida_tirar2_viagens.

4.4 Implementação Computacional

O algoritmo proposto foi implementado em Delphi, versão 7.0, interfaceando com planilhas do Excel. Mostram-se, a seguir, as principais telas do sistema desenvolvido.

4.4.1 Tela de abertura

Problema da Mistura de Minério

Teores

Elem	Fe (%)	Al2O3 (%)	P (%)	PPC (%)	He (%)
Frente1	52,64	0,52	0,084	4,48	45
Frente2	39,92	0,18	0,029	0,65	97
Frente3	47,19	0,5	0,05	2,52	52
Frente4	49,36	0,22	0,039	1,74	78
Frente5	43,94	0,46	0,032	2,36	41
Frente6	48,97	0,54	0,057	4,34	90
Frente7	47,46	0,2	0,047	5,07	9
Frente8	46,52	0,32	0,039	3,51	4
Frente9	56,09	0,95	0,059	4,1	80
Frente10	46	0,26	0,031	2,51	21
Frente11	49,09	0,22	0,04	4,2	12
Frente12	49,77	0,2	0,047	4,81	12
Frente13	53,03	0,24	0,047	4,17	1
Frente14	52,96	0,29	0,052	4,81	1
Frente15	42,09	0,17	0,031	1,38	47

Elementos

Fe Al2O3 P PPC He

Cálculos **Ver tabelas** **Sair**

Figura 4.46 – Tela de abertura do sistema

A Figura 4.46 mostra a tela de abertura do sistema. No exemplo considerado, há 15 frentes de lavra, sendo 11 de minério e 4 de estéril. Esses dados são digitados diretamente numa planilha no software Excel que interage com o Delphi.

Nesta Figura são mostrados os teores de cada parâmetro nas diversas frentes de lavra. Por ela, observa-se, por exemplo, que na frente 2 o teor de ferro (Fe) é de 39,92% e na frente 9 é de 56,09%.

4.4.2 Tela principal do sistema

A Figura 4.47 mostra a tela com os parâmetros do método e alguns resultados, como o cálculo da produção de minério e também de estéril, o cálculo das percentagens de cada parâmetro de controle e as diferenças, em toneladas, em relação à meta.

Meta(t): 4000 **Minério: soma =** 4000 **4000**

Meta(t): 1200 **Estéril: soma =** 1200 **1200**

Taxa de utilização (%): 0 **Diferença =** 0 **0**

sem taxa de utilização de caminhões

Elem - teores

01 - 52,64	0,52	0,084	4,48	45
02 - 39,92	0,18	0,029	0,65	97
03 - 47,19	0,5	0,05	2,52	52
04 - 49,36	0,22	0,039	1,74	78
05 - 43,94	0,46	0,032	2,36	41
06 - 48,97	0,54	0,057	4,34	90

	Meta (%)	Peso Critério	Peso Comparação	% Calculada	Diferença em relação à meta (t)	Desvio * Difer.(t)
Fe	47	5	1	47,9825	-39,3000000000001	196,5
Al2O3	0,32	1	100	0,3175	0,1	10
P	0,040	100	1000	0,04195	-0,078	7800
PPC	2,35	10	100	2,91375	-22,55	22550
He	40	1	1	37,95	81,9999999999999	81,9999999999999

Quantidade (Qu) máxima (t) existente em cada frente (1-minério; 0-estéril):

Frente1	1500	1	Frente5	1250	0	Frente9	1990	1	Frente13	1320	1
Frente2	2000	0	Frente6	1890	1	Frente10	900	0	Frente14	1245	1
Frente3	1700	1	Frente7	1640	1	Frente11	1540	1	Frente15	1859	0
Frente4	1450	1	Frente8	1124	1	Frente12	1630	1			

Total: 30638,5

obs.: Peso Critério :

Irrelevante:	0	1	10	100
Importante:	1			
Muito Importante:	5	1000	10000	
Crítico:	10			
Muito Crítico:	100			

Obs.:

Desvio=Critério * Comparação

Calcula **fo** **Limpar** **Sair** **Volta**

Figura 4.47 – Tela principal

Em relação aos parâmetros do método, destacam-se as metas de minério e de estéril que, no exemplo considerado, são de 4000 t e 1200 t, respectivamente, e as metas de qualidade (Fe=47%, Al_2O_3 =0,32%, P=0,040%, PPC=2,35%, He=40%).

São apresentadas também, nesta tela, as quantidades iniciais de minério ou estéril existente em cada frente, bem como o tipo de frente (1=minério; 0=estéril) e os pesos adotados para se calcular as penalidades no caso do não atendimento às metas de qualidade.

4.4.3 Tela de relatório das soluções inicial e final

A Figura 4.48 mostra a tela com a alocação das carregadeiras nas diversas frentes, bem como a tonelagem máxima a ser retirada de cada uma delas tanto para a solução inicial e final da heurística.

Relatório					
Solução inicial (So)					
Frente	Carreg		Carreg		Carreg
1: 0	0	6: 0	4	11: 900	1
2: 250	3	7: 0	0	12: 0	2
3: 1000	6	8: 0	0	13: 1000	7
4: 1100	8	9: 0	0	14: 0	0
5: 0	0	10: 0	0	15: 950	5
fo = 49542,85					
Solução final (s)					
Frente	Carreg		Carreg		Carreg
1: 0	0	6: 0	4	11: 800	1
2: 300	3	7: 0	0	12: 0	2
3: 1000	6	8: 1100	7	13: 0	0
4: 1100	8	9: 0	0	14: 0	0
5: 0	0	10: 0	0	15: 900	5
fo = 30638,5					
					

Figura 4.48 – Tela das soluções inicial e final

É mostrado na Figura 4.48 que, na frente 11, deveriam ser retiradas 900 t de minério pela carregadeira 1, inicialmente. Já na solução final, o método propõe que sejam retiradas 800 t de minério pela carregadeira 1.

A melhoria da solução pode ser constatada pelo valor da função objetivo (fo), que na solução inicial era de 49.542,85 e na solução final é de 30.638,5, nesse exemplo.

4.4.4 Tela de relatório do número de viagens de cada caminhão

A Figura 4.49 mostra a tela com o número de viagens de cada caminhão, por frente, o número total de viagens de cada caminhão, o total de viagens considerando todos os caminhões, o tempo acumulado de viagem para cada caminhão e a taxa de utilização de cada caminhão.

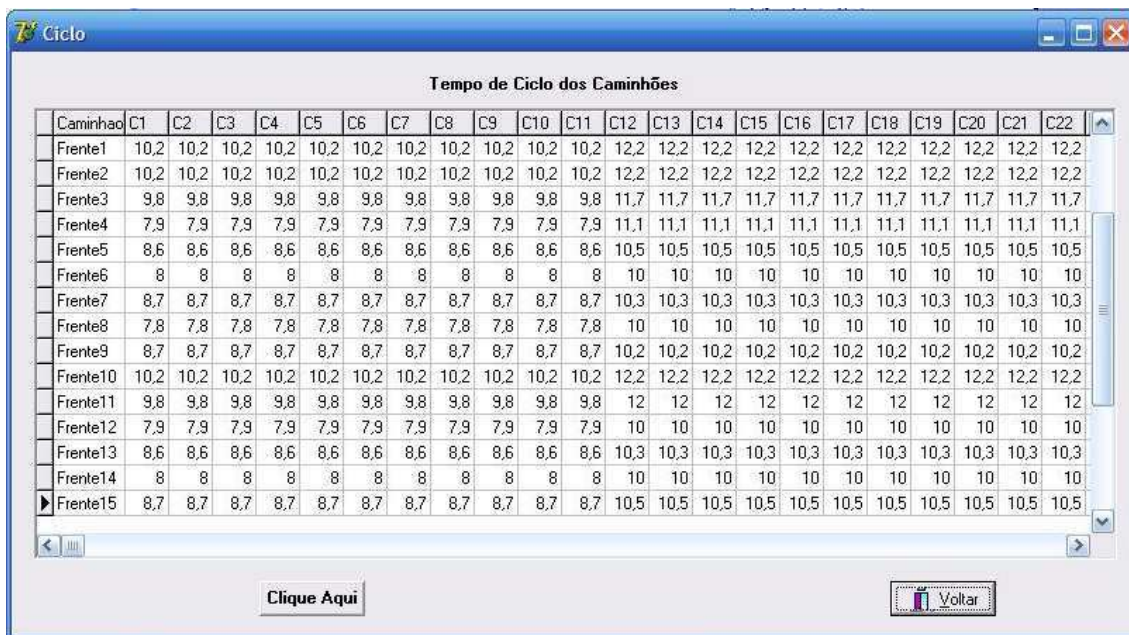
A Figura 4.49 mostra o caminhão 29 fazendo 3 viagens à frente 3, totalizando 35,10 minutos de percurso com 58,50% de taxa de utilização.

CAMINHÕES																N.Viag	Tempo	Util(%)
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15			
CAM1			3	1						1						5	47,50	79,17
CAM2		1		3												4	33,90	56,50
CAM3				4				2								6	47,20	78,67
CAM4		3		1						1						5	48,70	81,17
CAM5		4	1													5	50,60	84,33
CAM6		3								2						5	51,00	85,00
CAM7				1				2		1						4	33,70	56,17
CAM8		2		1						2						5	48,70	81,17
CAM9		1		4				1								6	49,60	82,67
CAM10		3		1												4	38,50	64,17
CAM11			2					3								5	43,00	71,67
CAM12								5								5	50,00	83,33
CAM13				2				2								4	42,20	70,33
CAM14								5								5	50,00	83,33
CAM15				4												4	44,40	74,00
CAM16																0	0,00	0,00
CAM17																0	0,00	0,00
CAM18																0	0,00	0,00
CAM19																0	0,00	0,00
CAM20			2								2					4	47,40	79,00
CAM21																0	0,00	0,00
CAM22			1								3					4	47,70	79,50
CAM23			3								1					4	47,10	78,50
CAM24											3					3	36,00	60,00
CAM25																0	0,00	0,00
CAM26			1								1					2	23,70	39,50
CAM27																0	0,00	0,00
CAM28																0	0,00	0,00
CAM29			3													3	35,10	58,50
CAM30																0	0,00	0,00
Total																92		72,70
																Caminhões utilizados: 21		

Figura 4.49 – Tela do número de viagens de cada caminhão

4.4.5 Tela do Tempo de Ciclo dos Caminhões

A Figura 4.50 mostra a tela com o tempo de ciclo dos caminhões em cada frente.



The screenshot shows a software window titled 'Ciclo'. Inside, there is a table titled 'Tempo de Ciclo dos Caminhões'. The table has 23 columns: 'Caminhao' (Truck) and 'C1' through 'C22' (Fronts). The rows represent 15 fronts, from 'Frente1' to 'Frente15'. The data is as follows:

Caminhao	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22
Frente1	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2
Frente2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2
Frente3	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	11,7	11,7	11,7	11,7	11,7	11,7	11,7	11,7	11,7	11,7	11,7
Frente4	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	11,1	11,1	11,1	11,1	11,1	11,1	11,1	11,1	11,1	11,1	11,1
Frente5	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5
Frente6	8	8	8	8	8	8	8	8	8	8	8	10	10	10	10	10	10	10	10	10	10	10
Frente7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3
Frente8	7,8	7,8	7,8	7,8	7,8	7,8	7,8	7,8	7,8	7,8	7,8	10	10	10	10	10	10	10	10	10	10	10
Frente9	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2
Frente10	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	10,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2	12,2
Frente11	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	9,8	12	12	12	12	12	12	12	12	12	12	12
Frente12	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	10	10	10	10	10	10	10	10	10	10	10
Frente13	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	8,6	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3	10,3
Frente14	8	8	8	8	8	8	8	8	8	8	8	10	10	10	10	10	10	10	10	10	10	10
Frente15	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	8,7	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5	10,5

At the bottom of the window, there are two buttons: 'Clique Aqui' and 'Voltar'.

Figura 4.50 – Tela do tempo de ciclo dos caminhões

A Figura 4.50 mostra que na frente 7, o tempo de ciclo do caminhão 11 é 8,7 minutos. Os dados dessa tabela são lidos diretamente de uma planilha em Excel.

4.4.6 Tela de Compatibilidade entre Caminhões e Carregadeiras

A Figura 4.51 mostra a tela com a compatibilidade entre caminhões e carregadeiras.

Compatibilidade

Compat	Cam1	Cam2	Cam3	Cam4	Cam5	Cam6	Cam7	Cam8	Cam9	Cam10	Cam11	Cam12	Cam13	Cam14	Cam15	Cam16	Cam17
Car1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
Car2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
Car3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
Car4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
Car5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Car6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Car7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Car8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Clique Aqui Voltar

Figura 4.51 – Tela de compatibilidade entre caminhões e carregadeiras

Na Figura 4.51, o número 1 significa que a carregadeira e o caminhão são compatíveis e 0, caso contrário. Assim, mostra-se nesta Figura que a carregadeira 4 é compatível com o caminhão 1.

4.4.7 Tela com a Capacidade Máxima de cada Caminhão.

A Figura 4.52 mostra a tela com a capacidade máxima de cada caminhão.

Capacidades

Caminhões

Capac	Cam1	Cam2	Cam3	Cam4	Cam5	Cam6	Cam7	Cam8	Cam9	Cam10	Cam11	Cam12	Cam13	Cam14	Cam15	Cam16
(t)	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	80

Clique Aqui Voltar

Figura 4.52 – Tela com a capacidade máxima de cada caminhão

A Figura 4.52 mostra que o caminhão 7 tem capacidade de 50 toneladas. Esses dados são disponibilizados por uma planilha em Excel.

4.4.8 Tela com a Capacidade Mínima de cada Carregadeira.

A Figura 4.53 mostra a tela com a capacidade mínima de cada carregadeira.



The screenshot shows a window titled "Capacidades Mínimas" with a table of minimum capacities for 8 loaders. The table has columns for "Cgmi", "Car1", "Car2", "Car3", "Car4", "Car5", "Car6", "Car7", and "Car8". The row "Min" shows values: 250, 300, 250, 300, 350, 300, 400, and 400. Below the table are two buttons: "Clique Aqui" and "Voltar".

Cgmi	Car1	Car2	Car3	Car4	Car5	Car6	Car7	Car8
Min	250	300	250	300	350	300	400	400

Figura 4.53 – Tela com a capacidade mínima de cada carregadeira

A Figura 4.53 mostra que a carregadeira 1 para ser colocada em operação deve atingir um mínimo de 250 toneladas por hora.

4.4.9 Tela com a Capacidade Máxima de cada Carregadeira.

A Figura 4.54 mostra a tela com a capacidade máxima de cada carregadeira.



Figura 4.54 – Tela com a capacidade máxima de cada carregadeira

A Figura 4.54 mostra que a carregadeira 7 trabalhará com um máximo de 1100 toneladas por hora.

5. Resultados Computacionais

5.1 Introdução

O algoritmo heurístico proposto foi desenvolvido na linguagem visual Delphi, versão 7 e o modelo de programação matemática, adaptado de Costa *et al.* (2004), foi implementado no modelador e otimizador LINGO, versão 10.0. Ambos foram testados em um PC com processador AMD Turion64 Mobile, Technology MK-36, 2,01 GHz, com 1 GB de RAM.

Foram considerados 8 problemas-teste, todos envolvendo 5 parâmetros de controle (Fe, Al_2O_3 , P, PPC, He), 15 frentes de lavra, sendo 11 de minério e 4 de estéril, com 30 caminhões e 8 carregadeiras disponíveis. Cada problema-teste representa um cenário diferente de uma mineração de ferro da região de Ouro Preto e Mariana, em Minas Gerais, conforme Tabela 5.1 a seguir.

Tabela 5.1 – Problemas-teste

	Problema-teste	Cenário
1	F15_M11_E4_V30_C8.1	Base
2	F15_M11_E4_V30_C8.2	Base, impondo taxa de utilização de caminhões de até 85%
3	F15_M11_E4_V30_C8.3	Base, sem limite mínimo para carregadeiras
4	F15_M11_E4_V30_C8.4	Cenário 3 com taxa de utilização de caminhões de até 85%
5	F15_M11_E4_V30_C8.5	Base, com compatibilidade total entre caminhões e carregadeiras
6	F15_M11_E4_V30_C8.6	Cenário 5, com taxa de utilização de caminhões de até 85%
7	F15_M11_E4_V30_C8.7	Base, com compatibilidade total entre caminhões e carregadeiras e sem limite mínimo para carregadeiras
8	F15_M11_E4_V30_C8.8	Cenário 7, com taxa de utilização de até 85%

Na Tabela 5.1, cada problema-teste é da forma FX_MY_EZ_VT_CK.W, onde X representa o número de frentes de lavra, Y o de frentes de minério, Z o de frentes de estéril, T o número de caminhões, K o número de carregadeiras e W o cenário.

Como referência para os demais, o cenário 1 (base) leva em consideração a compatibilidade entre caminhões e carregadeiras, o limite mínimo para as carregadeiras e a não observância da taxa de utilização de caminhões.

Na Tabela 5.2 são apresentadas as metas de produção e qualidade, bem como o número de equipamentos disponíveis e as prioridades de cada parâmetro de controle de qualidade. Nesta tabela, %Fe, %Al₂O₃, %P, %PPC e %He representam, respectivamente, os percentuais de ferro, alumina, fósforo, perda por calcinação e hematita especulada, os quais representam os parâmetros de controle. Minério (t) e Estéril (t) representam, respectivamente, as metas de produção de minério e estéril, em toneladas. Cam representa o número de caminhões e Carreg, o número de carregadeiras. Cada parâmetro de controle é classificado em cinco níveis de importância: IR (irrelevante), I (importante), MI (muito importante), C (crítico) e MC (muito crítico).

Tabela 5.2 – Metas de produção e qualidade, número de equipamentos disponíveis e prioridades

Item	% Fe	% Al ₂ O ₃	% P	% PPC	% He	Minério (t)	Estéril (t)	Cam.	Carreg.
Meta / # Equip.	47,000	0,320	0,040	2,350	40,000	4000	1200	30	8
Prioridade	MI	I	MC	C	I	-	-	-	-

Na Tabela 5.3 são apresentados os parâmetros usados nas execuções do algoritmo desenvolvido, os quais foram calibrados em uma bateria inicial de testes. Os pesos adotados consideram que o fósforo (P) é o parâmetro de controle mais crítico.

Tabela 5.3 – Parâmetros utilizados

Parâmetro	Significado	Valor
μ_A	Penalidade por ultrapassar a taxa máxima de utilização de um caminhão	1.000.000
μ_B	Penalidade por não atingir a taxa máxima de utilização de um caminhão	100
ω	Penalidade pela utilização de um caminhão	1
λ	Penalidade pela produtividade da carregadeira ficar abaixo da mínima exigida	10
ϕ	Penalidade pela produtividade da carregadeira ter ficado acima da máxima imposta	1
α_j^q	Penalidade por não atendimento à meta de qualidade do j -ésimo parâmetro de controle	$\alpha_{Fe}^q = 5$ $\alpha_{Al_2O_3}^q = 100$ $\alpha_P^q = 100.000$ $\alpha_{PPC}^q = 1.000$ $\alpha_{He}^q = 1$
β, γ	Penalidade por não atendimento à meta de produção de minério e estéril, respectivamente	2.000.000
<i>IterConstr</i>	Número de aplicações do método de construção da solução inicial	100
<i>IterMax</i>	Número máximo de iterações sem melhora do ILS	50
<i>NivelMax</i>	Número máximo de níveis de perturbação do ILS	10
<i>NumMaxVezes</i>	Número máximo de vizinhos avaliados em um dado nível de perturbação	10% do número de frentes $\times$ número de caminhões

5.2 Resultados obtidos

Os dados de cada cenário foram submetidos a dez execuções do algoritmo proposto. Para as tabelas apresentadas a seguir, foram adotadas as seguintes notações: Fe (ferro), Al₂O₃ (alumina), P (fósforo), PPC (perda por calcinação), He (Hematita especulada), minério (quantidade em toneladas de minério), estéril (quantidade em toneladas de estéril), fo (função de avaliação, calculada conforme equação 4.26, seção 4.3.4), carregadeiras (número total de carregadeiras), caminhões (número total de caminhões), viagens (número total de viagens), tempo (tempo de execução em segundos), TxUtil (taxa de utilização média de caminhões, sendo a taxa de utilização de cada caminhão calculada conforme equação 4.33).

As características da melhor solução obtida para cada cenário são apresentadas na Tabela 5.4.

Tabela 5.4 – Características dos melhores resultados obtidos

	Cenários							
	1	2	3	4	5	6	7	8
% Fe	47,9825	47,9825	47,9825	47,9825	47,9889	47,9825	47,9825	47,9825
% Al ₂ O ₃	0,3175	0,3175	0,3175	0,3175	0,3172	0,3175	0,3175	0,3175
% P	0,0419	0,0419	0,0419	0,0419	0,0419	0,0419	0,0419	0,0419
% PPC	2,9137	2,9137	2,9137	2,9137	2,9154	2,9137	2,9137	2,9137
% He	37,95	37,95	37,95	37,95	37,97	37,95	37,95	37,95
minério(t)	4000	4000	4000	4000	4000	4000	4000	4000
estéril (t)	1200	1200	1200	1200	1200	1200	1200	1200
Fo	30.630,50	30.583,83	30.626,50	30.577,83	30.710,98	30.615,16	30.632,10	30.594,76
carregadeiras	6	6	6	6	7	6	6	6
caminhões	27	20	26	21	28	21	24	21
viagens	89	92	92	89	86	89	89	89
Tempo(s)	62	161	67	216	143	171	121	177
% TxUtil	54,19	76,98	58,78	73,13	52,73	71,81	59,56	72,18

A Tabela 5.5 mostra, em cada célula, o desvio percentual encontrado para cada parâmetro de controle referente à melhor solução obtida, tendo em vista as metas estabelecidas. O desvio é calculado com base na expressão:

$$DesvioMeta = \frac{Encontrado - Meta}{Meta}$$

em que *Encontrado* representa o valor encontrado para o parâmetro de controle e *Meta* representa a meta estabelecida para o respectivo parâmetro de controle.

Tabela 5.5 – Desvios percentuais dos parâmetros de controle da melhor solução em relação às metas estabelecidas

Cenários								
	1	2	3	4	5	6	7	8
% Fe	2,09	2,09	2,09	2,09	2,10	2,09	2,09	2,09
% Al ₂ O ₃	-0,78	-0,78	-0,78	-0,78	-0,88	-0,78	-0,78	-0,78
% P	4,75	4,75	4,75	4,75	4,75	4,75	4,75	4,75
% PPC	23,99	23,99	23,99	23,99	24,06	23,99	23,99	23,99
% He	-5,12	-5,12	-5,12	-5,12	-5,08	-5,12	-5,12	-5,12
minério(t)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
estéril (t)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Na Tabela 5.6 são apresentados os valores médios obtidos nas dez execuções do algoritmo para cada cenário.

Tabela 5.6 – Valores médios obtidos

Cenários								
	1	2	3	4	5	6	7	8
% Fe	48,0294	48,0418	47,9974	48,0044	48,0026	47,9942	47,9891	48,0008
% Al ₂ O ₃	0,3164	0,3165	0,3169	0,3179	0,3165	0,3163	0,3169	0,3168
% P	0,0419	0,0420	0,0419	0,0420	0,0419	0,0419	0,0419	0,0419
% PPC	2,9241	2,9227	2,9205	2,9132	2,9248	2,9226	2,9179	2,9212
% He	37,8053	38,0143	37,7995	38,2775	37,9240	37,7885	37,8855	37,9420
minério(t)	4000	4000	4000	4000	4000	4000	4000	4000
estéril (t)	1200	1200	1200	1200	1200	1200	1200	1200
Fo	31.207,84	31.234,70	31.057,22	30.862,02	31.074,52	30.850,68	30.752,01	30.929,69
carregadeiras	6,4	6,3	6,3	6	6,5	6,5	6,4	6,2
caminhões	26,6	20,9	25	20,6	26,8	21,5	26,5	21,1
Viagens	89,6	90,2	91,1	90,5	87,2	89	88,4	88,4
tempo(s)	78,9	148,3	70,3	151,2	109,9	185,9	110,2	211,7
% TxUtil	55,951	72,909	60,361	74,381	54,342	69,911	55,249	71,126

A Tabela 5.7 mostra, em cada célula, o desvio percentual encontrado para cada elemento de avaliação da Tabela 5.6. O desvio é calculado com base na expressão:

$$Desvio = \frac{Média - Melhor}{Melhor}$$

em que *Média* representa o valor médio encontrado nas dez execuções do método e *Melhor* representa o melhor valor encontrado nessas execuções.

Tabela 5.7 – Desvios percentuais encontrados para algumas características das soluções.

Cenários								
	1	2	3	4	5	6	7	8
fo	1,88	2,13	1,41	0,93	1,18	0,77	0,39	1,09
carregadeiras	6,67	5,00	5,00	0,00	-7,14	8,33	6,67	3,33
caminhões	-1,48	4,50	-3,85	-1,90	-4,29	2,38	10,42	0,48
viagens	0,67	-1,96	-0,98	1,69	1,40	0,00	-0,67	-0,67
% TxUtil	3,25	-5,29	2,69	1,71	3,06	-2,64	-7,24	-1,46

Na Tabela 5.7, um valor negativo para um elemento de avaliação indica que na melhor solução, esse elemento de avaliação é pior que o valor da solução média. Por exemplo, para o cenário 1, a média de caminhões utilizados foi de 26,6 e o número de caminhões utilizados considerando a melhor solução obtida, foi de 27, o que proporcionou um desvio de -1,48%. Neste mesmo cenário, no entanto, necessitou-se de 6 carregadeiras na melhor solução, enquanto que na solução média foram necessárias 6,4 carregadeiras, mostrando que com relação a este item de avaliação, a melhor solução é cerca de 6,67% melhor que a da solução média.

De forma a comparar o desempenho do método heurístico proposto com o método de programação matemática, para cada cenário, executou-se o otimizador LINGO, versão 10.0, sobre o modelo de programação matemática da seção 4.2, interrompendo-se a execução em duas situações: (1) decorrido o tempo gasto pelo ILS para gerar a melhor solução em cada cenário, tempo esse dado pela Tabela 5.4; (2) decorrido o tempo médio de processamento requerido pelo método heurístico, dado pela Tabela 5.6. Esta comparação se justifica uma vez que o tempo para a tomada de decisão

é relativamente curto, da ordem de minutos, compatíveis com os tempos demandados pelo ILS.

As características da solução obtida pelo otimizador para cada cenário na primeira opção, isto é, decorrido o tempo gasto pelo ILS para gerar a melhor solução, são apresentadas na Tabela 5.8. Já na segunda opção, são apresentadas na Tabela 5.9.

Tabela 5.8 – Características das soluções do otimizador no tempo da melhor solução do ILS

Cenários								
	1	2	3	4	5	6	7	8
% Fe	47,9825	47,9889	47,9825	47,9825	47,9825	47,9937	47,9825	47,9883
% Al ₂ O ₃	0,3175	0,3173	0,3175	0,3175	0,3175	0,3166	0,3175	0,3173
% P	0,04195	0,04195	0,04195	0,04195	0,04195	0,04193	0,04195	0,04196
% PPC	2,91375	2,91548	2,91375	2,91375	2,91375	2,91968	2,91375	2,92163
% He	37,95	37,97	37,95	37,95	37,95	37,87	37,95	37,81
minério(t)	4000	4000	4000	4000	4000	4000	4000	4000
Estéril (t)	1200	1200	1200	1200	1200	1200	1200	1200
Fo	30.638,50	30.812,70	30.638,50	30.655,50	30.638,50	30.811,74	30.638,50	31.002,45
carregadeiras	6	6	7	6	8	6	7	6
Caminhões	19	17	17	17	18	17	16	20
Viagens	86	71	86	71	89	71	86	71
Tempo(s)	62	161	67	216	143	171	121	177
% TxUtil	76,44	73,50	82,46	72,44	78,87	72,89	85,67	62,54

Tabela 5.9 – Características das soluções do otimizador no tempo médio do ILS

Cenários								
	1	2	3	4	5	6	7	8
% Fe	47,9825	47,9937	47,9825	47,9825	47,9825	47,9825	47,9825	47,9825
% Al ₂ O ₃	0,3175	0,3166	0,3175	0,3175	0,3175	0,3175	0,3175	0,3175
% P	0,04195	0,04193	0,04195	0,04195	0,04195	0,04195	0,04195	0,04195
% PPC	2,91375	2,91968	2,91375	2,91375	2,91375	2,91375	2,91375	2,91375
% He	37,95	37,87	37,95	37,95	37,95	37,95	37,95	37,95
minério(t)	4000	4000	4000	4000	4000	4000	4000	4000
Estéril (t)	1200	1200	1200	1200	1200	1200	1200	1200
Fo	30.638,5	30.737,99	30.638,50	30.659,5	30.638,5	30.659,50	30.638,50	30.655,5
carregadeiras	8	6	7	6	6	6	7	8
Caminhões	16	19	17	20	16	20	16	17
Viagens	89	74	86	74	89	74	86	71
Tempo(s)	78,9	148,3	70,3	151,2	109,9	185,9	110,2	211,7
% TxUtil	88,54	67,19	82,46	62,96	89,38	63,11	85,67	73,11

A Tabela 5.10 mostra, para cada cenário, o percentual de melhora proporcionado pelo método heurístico ILS, em seu melhor comportamento (Tabela 5.4), comparado com a solução produzida pelo otimizador LINGO no mesmo tempo de processamento do ILS. O percentual de melhora é calculado com base na expressão:

$$GanhoMelhor = \frac{LINGO - MelhorILS}{LINGO}$$

em que *MelhorILS* representa o melhor valor da função de avaliação (fo) encontrado nas dez execuções do método ILS e *LINGO* representa o resultado encontrado pelo otimizador LINGO no mesmo tempo de processamento do ILS (vide Tabela 5.4).

Tabela 5.10 – Percentual de melhora do ILS em relação ao LINGO no tempo da melhor solução do ILS

Item	Cenários							
	1	2	3	4	5	6	7	8
<i>GanhoMelhor</i> (%)	0,03	0,74	0,04	0,25	-0,24	0,64	0,02	1,32
Tempo de execução (s)	62	161	67	216	143	171	121	177

Na Tabela 5.10, um valor positivo em uma célula indica que o ILS produziu uma solução de melhor qualidade que o otimizador LINGO. Por exemplo, para o cenário 2, o ILS foi 0,74% melhor que o LINGO. Já no cenário 5, a solução final do ILS foi 0,24% pior que aquela produzida pelo LINGO.

A Tabela 5.11 mostra, para cada cenário, o percentual de melhora proporcionado pelo método heurístico ILS, em seu comportamento médio (Tabela 5.6), comparado com a solução produzida pelo otimizador LINGO nesse mesmo tempo de processamento. O percentual de melhora é calculado com base na expressão:

$$GanhoMédio = \frac{LINGO^* - MédiaILS}{LINGO^*}$$

em que *MédiaILS* representa o valor médio da função de avaliação (fo) encontrado nas dez execuções do método ILS e *LINGO** representa o resultado encontrado pelo otimizador LINGO no tempo médio de processamento do ILS (vide tempos médios de processamento do ILS na Tabela 5.6).

Tabela 5.11 – Percentual de melhora do ILS em relação ao LINGO no tempo médio de processamento do ILS

Item	Cenários							
	1	2	3	4	5	6	7	8
<i>GanhoMédio</i> (%)	-1,86	-1,62	-1,37	-0,66	-1,42	-0,62	-0,37	-0,89
Tempo de execução (s)	78,9	148,3	70,3	151,2	109,9	185,9	110,2	211,7

Na Tabela 5.11, um valor negativo em uma célula indica que o otimizador LINGO produziu uma solução de melhor qualidade que o ILS. Como se observa, o LINGO foi capaz de produzir soluções de melhor qualidade que aquelas geradas pelo algoritmo ILS, sendo que esta melhora variou entre 0,37 e 1,86%.

Para mostrar a eficiência do método ILS, foram executados testes com o otimizador de programação matemática, dando-se a este um maior tempo de processamento. A Tabela 5.12 mostra as características da solução obtida pelo otimizador de programação matemática em uma hora de processamento, enquanto a Tabela 5.13 mostra os resultados decorridos 20 horas de processamento.

Tabela 5.12 – Características das soluções obtidas pelo otimizador em uma hora de processamento

	1	2	3	4	5	6	7	8
% Fe	47,9825	47,9889	47,9825	47,9825	47,9825	47,9825	47,9825	47,9825
% Al ₂ O ₃	0,3175	0,3173	0,3175	0,3175	0,3175	0,3175	0,3175	0,3175
% P	0,04195	0,04195	0,04195	0,04195	0,04195	0,04195	0,04195	0,04195
% PPC	2,91375	2,91548	2,91375	2,91375	2,91375	2,91375	2,91375	2,91375
% He	37,95	37,97	37,95	37,95	37,95	37,95	37,95	37,95
minério(t)	4000	4000	4000	4000	4000	4000	4000	4000
estéril (t)	1200	1200	1200	1200	1200	1200	1200	1200
Fo	30.638,50	30.736,99	30.638,50	30.657,50	30.638,50	30.655,50	30.638,50	30.656,50
carregadeiras	6	6	7	7	6	6	7	6
caminhões	19	18	17	19	16	17	16	18
Viagens	86	71	86	74	89	71	86	71
tempo(s)	3600	3600	3600	3600	3600	3600	3600	3600
% TxUtil	74,18	69,42	82,46	66,95	89,38	73,11	85,67	69,05

Tabela 5.13 – Características das soluções obtidas pelo otimizador em 20 horas de processamento

	1	2	3	4	5	6	7	8
% Fe	47,9825	47,9889	47,9825	47,9825	47,9825	47,9825	47,9825	47,9825
% Al ₂ O ₃	0,3175	0,3173	0,3175	0,3175	0,3175	0,3175	0,3175	0,3175
% P	0,04195	0,04195	0,04195	0,04195	0,04195	0,04195	0,04195	0,04195
% PPC	2,91375	2,91548	2,91375	2,91375	2,91375	2,91375	2,91375	2,91375
% He	37,95	37,97	37,95	37,95	37,95	37,95	37,95	37,95
minério(t)	4000	4000	4000	4000	4000	4000	4000	4000
estéril (t)	1200	1200	1200	1200	1200	1200	1200	1200
Fo	30.638,50	30.735,99	30.638,50	30.655,50	30.638,50	30.655,50	30.638,50	30.655,50
carregadeiras	6	6	7	7	6	6	7	6
caminhões	19	17	17	17	16	17	16	17
Viagens	86	71	86	71	89	71	86	71
tempo(h)	20	20	20	20	20	20	20	20
% TxUtil	74,18	73,50	82,46	72,61	89,38	72,44	85,67	72,44

A Tabela 5.14 mostra, para cada cenário, a distância entre a solução do LINGO em uma hora de processamento (respectivamente, 20 horas) e a gerada pelo método heurístico ILS, em seu melhor desempenho (Tabela 5.4). Esta métrica é calculada com base na expressão:

$$Distancia = \frac{LINGO^{**}}{MelhorILS} - 1$$

em que *MelhorILS* representa o melhor valor para a função de avaliação (fo) encontrado nas dez execuções do método ILS no tempo especificado na Tabela 5.4 e *LINGO^{**}* representa o resultado encontrado pelo otimizador LINGO em uma hora de processamento (respectivamente, 20 horas).

Tabela 5.14 – Comparação entre o LINGO e o melhor desempenho do ILS

Descrição	Cenários							
	1	2	3	4	5	6	7	8
<i>Distancia (%)</i> em uma hora	0,0261	0,501	0,039	0,261	-0,236	0,132	0,021	0,201
<i>Distancia (%)</i> em 20 horas	0,0261	0,498	0,039	0,254	-0,236	0,132	0,021	0,199
Tempo ILS (s)	62	161	67	216	143	171	121	177

Na Tabela 5.14, um valor positivo em uma célula indica que o ILS produziu uma solução de melhor qualidade que o otimizador LINGO. Por exemplo, para o cenário 4, a solução do LINGO em uma hora distou da solução do ILS em 0,261% e esta distância diminuiu para 0,254% quando o LINGO foi processado em 20 horas. Como se observa, não houve melhora do LINGO nos cenários ímpares, onde não se considera a taxa de utilização máxima para os caminhões. Apenas no cenário 5, a solução do LINGO foi melhor que a do ILS.

5.3 Discussão dos resultados

Pela Tabela 5.4, verifica-se que as soluções produzidas pelo ILS são de valores muito próximos, independentemente de se considerar incompatibilidade entre

carregadeiras e caminhões e de se impor um limite mínimo de carregamento para as carregadeiras.

Impondo-se uma taxa de utilização para os caminhões (cenários 2, 4, 6 e 8), há redução significativa no número de veículos necessários para atender a produção requerida. Por exemplo, para o cenário 2 (vide Tabela 5.6), reduz-se, em relação ao cenário 1, o número de veículos necessários de 26,6 para 20,9, isto é, uma redução de cerca de 21,4% na média. Comparando-se os cenários 7 e 8, verifica-se que no cenário 8 há uma redução de 20,4% no número médio de caminhões necessários, no caso, de 26,5 caminhões para 21,1 caminhões.

Verifica-se, também, que impondo-se uma taxa máxima de 85% para a utilização dos caminhões, a utilização média de caminhões é significativamente aumentada, fazendo com que a frota seja melhor aproveitada, com baixa ociosidade. Para exemplificar, no cenário 1 a taxa de utilização média dos caminhões é de 55,951%, enquanto que no cenário 2 essa taxa aumenta para 72,909%. Deve ser ressaltado que tal taxa de utilização não é a efetiva, uma vez que não se leva em consideração tempo de fila nas frentes.

Nos cenários 2, 4, 6 e 8, onde se aplica a taxa máxima de utilização para os caminhões, percebe-se que o tempo computacional requerido para a execução do algoritmo é superior àquele encontrado nos outros cenários onde tal taxa não é aplicada. Isso é devido ao maior esforço computacional existente na execução do algoritmo quando se aplica a taxa de utilização de caminhões.

Na Tabela 5.5, observa-se que o algoritmo proposto é capaz de gerar soluções bastante próximas das metas estabelecidas. A exceção ocorre com os parâmetros de controle PPC, He e P. Por exemplo, no cenário 1, apesar de o PPC ser considerado um parâmetro crítico (com peso igual a 10), é o parâmetro de controle com maior desvio. Isso é devido, no entanto, ao fato de existir apenas uma frente de minério com menos de 2,35% (Figura 4.46: frente 4 com 1,74%), que é o valor estabelecido como meta para este parâmetro, sendo, portanto, impossível seu atendimento. Já o parâmetro de controle He com -5,12% de desvio em relação à meta estabelecida foi outro cujo desvio foi alto. Observa-se, no entanto, que tal parâmetro tem baixa prioridade, no caso, ele é

considerado com prioridade mínima (Importante, com valor 1). No caso do fósforo, P, que é o parâmetro que possui a maior prioridade, considerado Muito Crítico, com peso igual a 100, o desvio não ultrapassa 5% mas, pelos mesmos motivos elencados para o parâmetro PPC, não é possível atender a meta com os recursos minerais disponíveis. No caso do fósforo, mesmo nas melhores soluções obtidas em cada cenário (vide Tabela 5.4), não foi possível atingir a meta.

Apesar de os cenários mais favoráveis serem aqueles nos quais não há restrições de compatibilidade entre caminhões e carregadeiras e nem imposição de limite mínimo de carregamento para as carregadeiras, o método ILS proposto não explora tal característica. Isto ocorre porque a função de avaliação usada para guiar a busca procura fazer sempre o melhor aproveitamento tanto da frota de caminhões quanto das carregadeiras, penalizando uma baixa taxa de utilização de caminhões e o não atendimento do carregamento mínimo. Nas soluções produzidas pelo modelo LINGO, relativas a essas situações, aparecem taxas de utilização de caminhões de até 100% e carregamento inferior ao mínimo da carregadeira. Mesmo assim, o método ILS consegue gerar soluções ainda melhores que as produzidas pelo LINGO.

A robustez do método é confirmada pelas tabelas 5.7, 5.11 e 5.14. Pela Tabela 5.7, observa-se que o desvio na função de avaliação é baixo, inferior a 2,13%, significando que partindo de uma solução inicial qualquer, o método em geral produz uma solução final inferior a 2,13% do melhor resultado, na média. Pela Tabela 5.11 observa-se que, na média, o ILS produz soluções piores que o LINGO, quando é dado ao LINGO o tempo médio de processamento do ILS. Entretanto, é importante ressaltar que as soluções do LINGO em geral não são factíveis, como apontado no parágrafo anterior. Por outro lado, pela Tabela 5.14, que compara a distância das soluções produzidas pelo LINGO, em uma hora e 20 horas de processamento, com a melhor solução do ILS, verifica-se que o ILS gera soluções melhores, à exceção do cenário 5, apesar do pouco tempo de processamento (máximo de 216 segundos). As referidas distâncias variam de 0,021% a 0,501% quando consideradas as soluções do LINGO em uma hora de processamento. Quando o tempo dado ao LINGO é aumentado para 20 horas de processamento, tais distâncias são levemente reduzidas, variando de 0,021% a 0,498%.

6. Conclusões e trabalhos futuros

Este trabalho tratou o problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões. Dada sua dificuldade de resolução na otimalidade, foi desenvolvido um método heurístico baseado na metaheurística *Iterated Local Search* (ILS).

Para gerar uma solução inicial distribuem-se aleatoriamente carregadeiras às frentes de lavra, seguida da atribuição também aleatória de viagens aos diversos caminhões, respeitando-se a compatibilidade entre caminhões e carregadeiras. A seguir, utilizam-se dois reparadores para as inviabilidades que esta estratégia pode trazer. O primeiro reparador de inviabilidade procura solucionar o problema de a soma dos tempos de ciclo de cada caminhão superar 60 minutos, enquanto o segundo visa a respeitar a capacidade máxima da carregadeira em uma hora de produção. A essa solução gerada aplica-se um refinamento por meio do método da descida em vizinhança variável (VND), usando-se diferentes tipos de movimento para explorar o espaço de busca. Este método de construção e refinamento é aplicado certo número de vezes e a melhor solução construída é utilizada como solução inicial para o método ILS. Essa solução inicial representa um ótimo local com respeito a todas as vizinhanças consideradas, uma vez que é feita a descida completa com relação a cada vizinhança. Para escapar do ótimo local produzido pelo método VND, utilizou-se o método *Iterated Local Search* (ILS). Nesse método, cada perturbação consiste em promover modificações na solução ótima local corrente. A solução perturbada é, então, refinada pelo método VND. Havendo melhora no ótimo local corrente, move-se para a melhor solução; do contrário, aumenta-se o nível de perturbação. O nível de perturbação mais baixo indica pequenas modificações na solução ótima local corrente, enquanto níveis mais elevados consistem em promover modificações mais acentuadas na solução. O método encerra-se após um número máximo de iterações sem melhora.

Para testar o método foram utilizados 8 cenários relativo a um problema real de uma mineração de ferro. Para cada cenário, foram comparadas as soluções produzidas

pelo método ILS proposto com aquelas produzidas por um otimizador aplicado a um modelo de programação matemática desenvolvido.

Os resultados obtidos mostraram que as melhores soluções produzidas pelo método heurístico proposto (ILS) são de melhor qualidade que aquelas produzidas pelo otimizador LINGO aplicado ao modelo de programação matemática, mesmo considerando as dimensões relativamente modestas dos problemas-teste tratados.

Adicionalmente, verificou-se que as soluções de qualidade produzidas pelo ILS foram obtidas sem muito esforço computacional, ao contrário daquelas geradas pelo otimizador. Este último demandou tempos proibitivos para gerar soluções próximas àsquelas geradas pelo ILS em poucos minutos, fato que valida a utilização do método proposto, enquanto ferramenta de apoio à decisão.

Com relação às características das soluções produzidas pelo método proposto nos diversos cenários analisados, pode-se verificar que os valores dos parâmetros de controle ficaram bem próximos das metas estabelecidas. As exceções foram para o PPC e o fósforo (P), que só não obtiveram resultados melhores devido à baixa qualidade dos recursos minerais existentes para esses parâmetros. Em relação à frota de veículos de transporte e de equipamentos de carga ficou constatado o bom aproveitamento dos mesmos. No caso dos equipamentos de carga conseguiu-se reduzir a quantidade desses, bem como utilizá-los próximos de suas capacidades máximas de produção. Já no caso dos veículos de transporte, também houve redução da quantidade de veículos necessários, bem como um melhor aproveitamento desses, indicado pelas taxas de utilização próximas à meta.

Esses resultados comprovam, portanto, a eficiência do método heurístico proposto para a otimização do planejamento operacional de lavra. Destaca-se, finalmente, que o método ILS é flexível, no sentido de que outras restrições podem ser facilmente incorporadas ao método.

Em relação a Costa (2005), este trabalho representa uma evolução, no sentido de considerar número de caminhões e taxas de utilização desses no modelo de programação matemática, bem como um conjunto maior de movimentos para explorar o espaço de soluções no método heurístico. Adicionalmente, apesar de a comparação

entre os métodos não ser possível devido às características diferentes dos problemas considerados, o procedimento de geração da solução inicial aqui proposto, com os reparadores de inviabilidade, é mais eficiente, uma vez que no método de Costa (2005) a solução inicial é totalmente aleatória. Experimentos relatados na literatura mostram que os métodos heurísticos, em especial *Iterated Local Search*, são altamente dependentes da qualidade da solução inicial (Glover e Kochenberger, 2003).

Como trabalhos futuros sugere-se a incorporação da técnica Reconexão por Caminhos (*Path Relinking*), vide Glover e Kochenberger (2003), para melhorar ainda mais a qualidade das soluções produzidas. Outra proposição é a inserção do otimizador para resolver partes menores do problema. Nesse caso, de cada solução obtida pela heurística, seria escolhido certo número de frentes, carregadeiras e caminhões e submetido ao otimizador o planejamento operacional dessa parte. Caso a solução parcial obtida fosse melhor que aquela gerada pela heurística, essa solução parcial substituiria a da heurística. Para a eficiência dessa estratégia é necessário que o tempo de resolução da parte menor do problema pelo otimizador seja rápido.

Referências Bibliográficas

- ALVARENGA, G. B. (1997). *Despacho ótimo de caminhões numa mineração de ferro utilizando algoritmo genético com processamento paralelo*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais.
- ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. (2007). *Pesquisa Operacional para cursos de engenharia*, Editora Campus, Rio de Janeiro.
- CHANDA, E. K. C.; DAGDELEN, K. (1995). Optimal blending of mine production using goal programming and interactive graphics systems, *International Journal of Surface Mining, Reclamation and Enviroment*, v. 9, p. 203-208.
- COSTA, F. P. (2005). *Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, 141 p.
- COSTA, F. P.; SOUZA, M. J. F. e PINTO, L. R. (2004). Um modelo de alocação dinâmica de caminhões, *Revista Brasil Mineral*, v. 231, p. 26-31.
- COSTA, F. P.; SOUZA, M. J. F. e PINTO, L. R. (2005). Um modelo de programação matemática para alocação estática de caminhões visando ao atendimento de metas de produção e qualidade, *Revista da Escola de Minas*, v. 58, p. 77-81.
- EZAWA, L. e SILVA, K. S. (1995). Alocação dinâmica de caminhões visando qualidade. *VI Congresso Brasileiro de Mineração*, p. 15–19, Salvador, Bahia.
- GLOVER, F.; KOCHENBERGER, G. (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston.
- LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. (2003). Iterated Local Search. *In Handbook of Metaheuristics*, Glover, F. and Kochenberger, G. (Eds). Kluwer Academic Publishers, Boston.

- MERSCHMANN, L. H. C. (2002). *Desenvolvimento de um Sistema de Otimização e Simulação para Cenários de Produção em Minas a Céu Aberto*, Dissertação de mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.
- MLADENOVIC, N. & HANSEN, P. (1997). "A Variable Neighborhood Search". *Computers and Operations Research*, v. 24, p. 1097-1100.
- MORAES, E. F. (2005). *Um modelo de programação matemática para otimizar a composição de lotes de minério de ferro na mina Cauê da CVRD*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, 78 p.
- MUNIRATHINAM, M. e YINGLING, J. C. (1994). A review of computer-based truck dispatching strategies of surface mining operations, *International Journal of Surface Mining, Reclamation and Environment*, v. 8, p. 1-15.
- MUTMANSKY, J. M. (1979). Computing and operations research techniques for production scheduling. *Computer methods for the 80's in the mineral industry*, p. 615-625, New York.
- PHPMETAR (2006). Pesquisa Operacional. Disponível em phpmetar.incubadora.fapesp.br/portal/Faculdade/Pesquisa%20Operacional. Acesso em 04/07/2006.
- PINTO, L.R.; MERSCHMANN, L. H. C. (2001). Planejamento operacional da lavra de mina usando modelos matemáticos. *Revista Escola de Minas*, v. 54, n. 3, p. 211-214.
- PINTO, L. R.; BIAJOLI, F. L. e MINE, O. M. (2003). *Uso de otimizador em planilhas eletrônicas para auxílio ao planejamento de lavra*. Relatório técnico, Programa de Pós-graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, Ouro Preto, Minas Gerais.
- RODRIGUES, L. F. (2006). *Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Minas Gerais, 86 p.

- SOUZA, M. J. F. (2006). Notas de aula do curso Pesquisa Operacional aplicada à Mineração. Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto.
- SOUZA, M. J. F. (2007). Notas de aula da disciplina Inteligência Computacional para Otimização. Departamento de Computação, Universidade Federal de Ouro Preto, disponível em <http://www.decom.ufop.br/prof/marcone/disciplinas/InteligênciaComputacional/InteligênciaComputacional.pdf>. Acesso em 08/05/2007.
- WHITE, J. W., ARNOLD, M. J. e CLEVINGER, J. G. (1982). Automatated open-pit truck dispatching at Tyrone, *Engineering and Mining Journal*, v. 183, n. 6, p. 76-84.
- WHITE, J. W. e OLSON, J. P. (1986). Computer-based dispatching in mines with concurrent operating objectives. *Mining Engineering*, v. 38, n. 11, p. 1045–1054.

Anexo I: PROBLEMAS-TESTE

São caracterizados, a seguir, os dados dos problemas-teste utilizados.

- Parâmetros de Controle

	Tipo Frente	Fe (%)	Al₂O₃ (%)	P (%)	PPC (%)	He (%)	Massa (t)
Frente1	M	52,64	0,52	0,084	4,48	45,0	1500
Frente2	E	39,92	0,18	0,029	0,65	97,0	2000
Frente3	M	47,19	0,50	0,050	2,52	52,0	1700
Frente4	M	49,36	0,22	0,039	1,74	78,0	1450
Frente5	E	43,94	0,46	0,032	2,36	41,0	1250
Frente6	M	48,97	0,54	0,057	4,34	90,0	1890
Frente7	M	47,46	0,20	0,047	5,07	9,0	1640
Frente8	M	46,52	0,32	0,039	3,51	4,0	1124
Frente9	M	56,09	0,95	0,059	4,10	80,0	1990
Frente10	E	46,00	0,26	0,031	2,51	21,0	900
Frente11	M	49,09	0,22	0,040	4,20	12,0	1540
Frente12	M	49,77	0,20	0,047	4,81	12,0	1630
Frente13	M	53,03	0,24	0,047	4,17	1,0	1320
Frente14	M	52,96	0,29	0,052	4,81	1,0	1245
Frente15	E	42,09	0,17	0,031	1,38	47,0	1859

Legenda: M = minério; E = estéril; Fe = ferro; Al₂O₃ = alumina, P = fósforo, PPC = perda por calcinação; He = hematita especulada.

- Tempo de Ciclo dos Caminhões, em minutos

Fr Cam	Cam1 a Cam11	Cam12 a Cam30
Frente1	10,2	12,2
Frente2	10,2	12,2
Frente3	9,8	11,7
Frente4	7,9	11,1
Frente5	8,6	10,5
Frente6	8	10
Frente7	8,7	10,3
Frente8	7,8	10
Frente9	8,7	10,2
Frente10	10,2	12,2
Frente11	9,8	12
Frente12	7,9	10
Frente13	8,6	10,3
Frente14	8	10
Frente15	8,7	10,5

Legenda: Fr = frente; Cam = caminhão

- Capacidade dos caminhões, em toneladas

1	2	3	4	5	...	11	12	13	14	15	16	17	18	19	20	...	29	30
50	50	50	50	50	...	50	50	50	50	50	80	80	80	80	80	...	80	80

- Capacidade máxima e mínima das carregadeiras, em toneladas

	C1	C2	C3	C4	C5	C6	C7	C8
Max	900	900	900	900	1000	1000	1100	1100
Min	250	300	250	300	350	300	400	400

Legenda: Max = capacidade máxima; Min = capacidade mínima; c1 a c8 = carregadeiras

- Matriz de compatibilidade entre caminhões e carregadeiras

	C1	C2	C3	C4	C5	C6	C7	C8
CAM1	1	1	1	1	1	1	1	1
CAM2	1	1	1	1	1	1	1	1
CAM3	1	1	1	1	1	1	1	1
CAM4	1	1	1	1	1	1	1	1
CAM5	1	1	1	1	1	1	1	1
CAM6	1	1	1	1	1	1	1	1
CAM7	1	1	1	1	1	1	1	1
CAM8	1	1	1	1	1	1	1	1
CAM9	1	1	1	1	1	1	1	1
CAM10	1	1	1	1	1	1	1	1
CAM11	1	1	1	1	1	1	1	1
CAM12	1	1	1	1	1	1	1	1
CAM13	1	1	1	1	1	1	1	1
CAM14	1	1	1	1	1	1	1	1
CAM15	1	1	1	1	1	1	1	1
CAM16	0	0	0	0	1	1	1	1
CAM17	0	0	0	0	1	1	1	1
CAM18	0	0	0	0	1	1	1	1
CAM19	0	0	0	0	1	1	1	1
CAM20	0	0	0	0	1	1	1	1
CAM21	0	0	0	0	1	1	1	1
CAM22	0	0	0	0	1	1	1	1
CAM23	0	0	0	0	1	1	1	1
CAM24	0	0	0	0	1	1	1	1
CAM25	0	0	0	0	1	1	1	1

CAM26	0	0	0	0	1	1	1	1
CAM27	0	0	0	0	1	1	1	1
CAM28	0	0	0	0	1	1	1	1
CAM29	0	0	0	0	1	1	1	1
CAM30	0	0	0	0	1	1	1	1

Legenda: c1 a c8 = carregadeiras; CAM1 a CAM30: caminhões; 0 = incompatível; 1 = compatível

Anexo II: Modelo de Programação Matemática

model:

title: Alocação Dinâmica de Caminhões;

sets:

```
parametros/@ole('Pilhas.xls','parametros')/:
    tl, ! Teor mínimo admissível para o parâmetro j (%);
    tu, ! Teor máximo admissível para o parâmetro j (%);
    tr, ! Teor recomendado para o parâmetro j (%);
    wnm, ! Peso por desvio negativo para o parâmetro j;
    wpm, ! Peso por desvio positivo para o parâmetro j;
    dnm, ! Desvio negativo de meta do parâmetro j (t/h);
    dpm, ! Desvio positivo de meta do parâmetro j (t/h);
frentes/@ole('Pilhas.xls','frentes')/:
    Qu, ! Massa disponível na frente i (t);
    estMin, ! Se a frente i é de minério (1) ou estéril (0);
    x, ! Ritmo de lavra para a frente i (t/h);
carregadeiras/@ole('Pilhas.xls','carregadeiras')/:
    Cu, ! Produção máxima da carregadeira k (t/h);
    Cl, ! Produção mínima da carregadeira k (t/h);
caminhoes/@ole('Pilhas.xls','caminhoes')/:
    capCam, ! Capacidade do caminhão l (t);
    txMax, ! Taxa de utilização máxima indicada para o caminhão l (%);
    dnu, ! Desvio negativo de utilização de caminhões;
    dpu, ! Desvio positivo de utilização de caminhões;
    usou, ! Vale 1 se caminhão l for usado;
matriz1(frentes,parametros):
    t, ! Teor do parâmetro j na frente i (%);
matriz2(frentes,carregadeiras):
    y, ! Se a carregadeira k opera na frente i;
matriz3(caminhoes,carregadeiras):
    comp, ! Se o caminhão l é compatível (1) ou não (0) com a carregadeira k;
matriz4(frentes,caminhoes):
    n, ! Número de viagens que o caminhão l realiza a frente i;
    tempCiclo, ! Tempo de ciclo do caminhão l à frente i;
```

endsets

data:

```
! Produção mínima (t/h), Produção Máxima (t/h), Produção recomendada (t/h);
pr, pe = @ole('Pilhas.xls','pr','pe');
```

```
! Penalidade para o desvio negativo de produção, Penalidade para o desvio positivo de produção;
```

```
wnp = @ole('Pilhas.xls','wnp');
```


! Teor mínimo, teor máximo, teor recomendado, teor;
tr, t = @ole('Pilhas.xls','tr','t');

! Penalidade por desvio negativo de meta de qualidade, penalidade por desvio positivo de meta de qualidade;
wnm = @ole('Pilhas.xls','wnm');

! Massa disponível, vetor estéril/minério que vale 1 se minério e 0 se estéril;
Qu, estMin = @ole('Pilhas.xls','Qu','estMin');

! Capacidade máxima carregadeira e Capacidade mínima da carregadeira;
Cu, Cl = @ole('Pilhas.xls','Cu','Cl');

! Capacidade do caminhão, vetor de compatibilidade (vale 1 se caminhão é compatível com carregadeira), tempo de ciclo;
capCam, comp, tempCiclo = @ole('Pilhas.xls','capCam','comp','tempCiclo');

! Taxa de utilização máxima permitida para cada caminhão ;
txMax = @ole('Pilhas.xls','txMax');

! Peso para o desvio negativo de taxa de utilização para os caminhões ;
wnu = @ole('Pilhas.xls','wnu');

enddata

[fo] min = @sum(parametros(j): wnm(j)*dnm(j) + wnm(j)*dpm(j)) +
wnp*dnp + wnp*dpp + wnp*dne + wnp*dpe +
@sum(caminhoes(l): usou(l));

! O teor do parâmetro j na mistura deve ser inferior ou igual ao teor máximo admitido;
@for(parametros(j): @sum(frentes(i) | estMin(i) #eq# 1: (t(i,j) - tu(j))*x(i)) <= 0);

! O teor do parâmetro j na mistura deve ser superior ou igual ao teor mínimo admitido;
@for(parametros(j): @sum(frentes(i) | estMin(i) #eq# 1: (t(i,j) - tl(j))*x(i)) >= 0);

! A meta do parâmetro j na mistura deve ser buscada sempre que possível;
@for(parametros(j): @sum(frentes(i) | estMin(i) #eq# 1: (t(i,j) - tr(j))*x(i)) + dnm(j) - dpm(j) = 0);

! A produção total deve ser inferior ou igual à produção máxima admitida;
@sum(frentes(i) | estMin(i) #eq# 1: x(i)) <= pu;

! A produção total deve ser superior ou igual à produção mínima admitida;
@sum(frentes(i) | estMin(i) #eq# 1: x(i)) >= pl;

! A meta de produção de minério deve ser buscada sempre que possível;
@sum(frentes(i) | estMin(i) #eq# 1: x(i)) + dnp - dpp = pr;

! A meta de produção de estéril deve ser buscada sempre que possível;
 $\text{@sum}(\text{frentes}(i) \mid \text{estMin}(i) \neq 0: x(i)) + \text{dne} - \text{dpe} = \text{pe};$

! Deve haver no máximo uma carregadeira k operando em cada frente i;
 $\text{@for}(\text{frentes}(i): \text{@sum}(\text{carregadeiras}(k): y(i,k)) \leq 1);$

! Cada carregadeira k deve operar no máximo em uma frente i;
 $\text{@for}(\text{carregadeiras}(k): \text{@sum}(\text{frentes}(i): y(i,k)) \leq 1);$

! O ritmo de lavra da frente i deve ser maior do que a produtividade mínima da carregadeira k alocada à frente;
 $\text{@for}(\text{frentes}(i):$
 $\quad x(i) \geq \text{@sum}(\text{carregadeiras}(k): Cl(k)*y(i,k)));$

! O ritmo de lavra da frente i deve ser menor do que a produtividade máxima da carregadeira k alocada à frente;
 $\text{@for}(\text{frentes}(i):$
 $\quad x(i) \leq \text{@sum}(\text{carregadeiras}(k): Cu(k)*y(i,k)));$

! Cada caminhão l deve realizar viagens apenas à uma frente i que esteja alocada uma carregadeira compatível;
 $\text{@for}(\text{frentes}(i):$
 $\quad \text{@for}(\text{caminhoes}(l):$
 $\quad \quad n(i,l)*\text{tempCiclo}(i,l) \leq$
 $\quad \quad 60*\text{@sum}(\text{carregadeiras}(k) \mid \text{comp}(l,k) \neq 1: y(i,k))$
 $\quad)$
 $);$

! Cada caminhão l deve operar no máximo $60*\text{txMax}(l)$ minutos;
 $\text{@for}(\text{caminhoes}(l):$
 $\quad \text{@sum}(\text{frentes}(i): n(i,l)*\text{tempCiclo}(i,l)) \leq \text{txMax}(l)*60);$

! O ritmo de lavra da frente i deve ser igual à produção realizada pelos caminhões alocados à frente;
 $\text{@for}(\text{frentes}(i):$
 $\quad x(i) = \text{@sum}(\text{caminhoes}(l): n(i,l)*\text{capcam}(l)));$

! A taxa de utilização de cada caminhão l deve ser de, no mínimo, $\text{txMax}(l)\%$;
 $\text{@for}(\text{caminhoes}(l):$
 $\quad \text{@sum}(\text{frentes}(i): n(i,l)*\text{tempCiclo}(i,l))/60 + \text{dnu}(l) = \text{txMax}(l));$

! Um caminhão l é usado se fizer alguma viagem à uma frente qualquer;
 $\text{@for}(\text{caminhoes}(l):$
 $\quad \text{usou}(l) \geq \text{@sum}(\text{frentes}(i): n(i,l)*\text{tempCiclo}(i,l))/60;$
 $\quad \text{@bin}(\text{usou}(l)));$

! A variável y deve ser binária;

```
@for(frentes(i): @for(carregadeiras(k): @bin(y(i,k))));
```

! O número de viagens que cada caminhão l faz à frente i (variável n) deve ser inteiro;

```
@for(frentes(i):
```

```
  @for(caminhoes(l): @gin(n(i,l))));
```

! Exporta resultados para a planilha;

```
data:
```

```
  @ole('Pilhas.xls','n')=n;
```

```
  @ole('Pilhas.xls','x')=x;
```

```
  @ole('Pilhas.xls','y')=y;
```

```
  @ole('Pilhas.xls','dnm')=dnm;
```

```
  @ole('Pilhas.xls','dpm')=dpm;
```

```
  @ole('Pilhas.xls','dnp')=dnp;
```

```
  @ole('Pilhas.xls','dpp')=dpp;
```

```
  @ole('Pilhas.xls','dnu')=dnu;
```

```
  @ole('Pilhas.xls','usou')=usou;
```

```
enddata
```

```
end
```

Anexo III: Publicações

Os trabalhos oriundos desta pesquisa que foram submetidos a periódicos de circulação nacional e internacional e/ou apresentados em eventos são listados a seguir.

ARAÚJO, F.C.R.; SOUZA, M.J.F.; TOLEDO, J.P. Um Método Híbrido, Baseado em *Simulated Annealing* e Método da Subida, para Resolver o Problema de Seleção de Projetos Mineiros Concorrentes: IV Congresso Brasileiro de Mina a Céu Aberto e IV Congresso Brasileiro de Mina Subterrânea. Belo Horizonte, 2006.

ARAÚJO, F.C.R.; SOUZA, M.J.F. Planejamento Operacional de Lavra com Alocação Dinâmica de Caminhões: Abordagens Exata e Heurística. V Congresso Brasileiro de Mina a Céu Aberto e V Congresso Brasileiro de Mina Subterrânea. Belo Horizonte, 2008.

ARAÚJO, Francisco César Rodrigues; SOUZA, Marcone Jamilson Freitas. Modelos exato e heurístico para o Planejamento de Lavra com Alocação Dinâmica de Caminhões. Submetido para publicação na Revista Escola de Minas, julho de 2008.

**ALGORITMO PARA PROGRAMAÇÃO
INTEGRADA DE VEÍCULOS E TRIPULAÇÕES
NO SISTEMA DE TRANSPORTE PÚBLICO POR
ÔNIBUS**

EMILIANA MARA LOPES SIMÕES

ORIENTADOR: GERALDO ROBSON MATEUS

CO-ORIENTADOR: MARCONE JAMILSON FREITAS SOUZA

**ALGORITMO PARA PROGRAMAÇÃO
INTEGRADA DE VEÍCULOS E TRIPULAÇÕES
NO SISTEMA DE TRANSPORTE PÚBLICO POR
ÔNIBUS**

Dissertação apresentada ao Programa de
Pós-Graduação em Ciência da Computação
da Universidade Federal de Minas Gerais
como requisito parcial para a obtenção do
grau de Mestre em Ciência da Computação.

Belo Horizonte

Agosto de 2009

© 2009, Emilianara Mara Lopes Simões.
Todos os direitos reservados.

D1234p Lopes Simões, Emilianara Mara
ALGORITMO PARA PROGRAMAÇÃO
INTEGRADA DE VEÍCULOS E TRIPULAÇÕES
NO SISTEMA DE TRANSPORTE PÚBLICO POR
ÔNIBUS / Emilianara Mara Lopes Simões. — Belo
Horizonte, 2009
xviii, 82 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal
de Minas Gerais
Orientador: Geraldo Robson Mateus
Co-orientador: Marcane Jamilson Freitas Souza

1. Programação Integrada de Veículos e
Tripulações. 2. Metaheurísticas. 3. Busca Local
Iterada. I. Título.

CDU 519.6*82.10

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={nome do arquivo}`
ao comando `\ppgccufmg`.

Dedico este trabalho aos meus inesquecíveis avós, Vô Lopes e Vó Mms, dos quais sempre recebi um amor incondicional, acalentador e encorajador.

*“Tudo é do pai!
Toda honra e toda glória,
é dele a vitória alcançada em minha vida.”
(Frederico Cruz)*

Resumo

Este trabalho aborda dois dos problemas envolvidos no processo de planejamento do Sistema de Transporte Público de Ônibus Urbano: o Problema de Programação de Veículos (PPV) e o Problema de Programação de Tripulações (PPT). O PPV consiste em criar uma rotina diária de operação para uma frota de veículos, dado um conjunto de viagens a serem obrigatoriamente realizadas. Na resolução do PPT são definidas as jornadas diárias de trabalho para motoristas e cobradores, de maneira que, todos os deslocamentos dos veículos estejam sob responsabilidade de alguma tripulação. O objetivo é reduzir os custos envolvidos em tais atividades, fazendo o melhor aproveitamento da frota e da mão-de-obra, mas ao mesmo tempo obedecendo à legislação trabalhista e as regras operacionais vigentes.

Normalmente, devido à complexidade destes problemas, eles são resolvidos separadamente e seqüencialmente (abordagem seqüencial tradicional, ou seja, primeiro resolve-se PPV e logo em seguida o PPT). Porém, dada a forte dependência entre PPV e PPT, espera-se que a resolução integrada dos mesmos possa proporcionar considerável redução dos custos. Assim, dadas as expectativas atuais acerca dessa nova metodologia, neste trabalho é desenvolvida uma heurística, baseada na metaheurística Busca Local Iterada, para resolução do Problema de Programação Integrada de Veículos e Tripulações (PPVT). No PPVT as rotinas de operação dos veículos e as jornadas de trabalho das tripulações são definidas simultaneamente. Trata-se de um problema altamente combinatorial e que se enquadra na classe de problemas NP-difícil, não havendo, portanto, algoritmos em tempo polinomial para encontrar soluções ótimas.

Com o intuito de analisar o desempenho da abordagem integrada, PPV e PPT também são resolvidos de forma iterativa, seqüencial tradicional e independente (PPV e PPT são resolvidos separadamente, em qualquer ordem e de forma a não se relacionarem). Para validar os métodos propostos são apresentados experimentos com dados reais de uma empresa de transporte público da cidade de Belo Horizonte. Os resultados encontrados confirmam as expectativas e mostram que as soluções geradas pela heurística integrada são melhores do que as obtidas de forma seqüencial.

Abstract

This work deals with two of the problems involved in the planning process of the public mass transit: the Vehicle Scheduling Problem (VSP) and the Crew Scheduling Problem (CSP). The VSP is a way to create a daily operation routine for a fleet of vehicles, given a set of trips required to be performed. In the resolution of CSP the daily work duty for the drivers and collectors are defined, so that all displacements of the vehicles are in charge of a crew. This paper aimed to investigate how to reduce costs involved in such activities for the best use of the fleet and labor force, but also comply with the public policies of labor laws and operational rules.

These problems has been usually treated separately and sequentially due to their complexity (traditional sequential approach, in other words, first VSP is solved and then CSP). However, since the strong dependence between VSP and CSP, it is expected that the integrated resolution of them could provide considerable costs reduction. Thus, given the current expectations about this new methodology, in this work heuristics are developed, based on the Iterated Local Search metaheuristic to solve the Integrated Vehicle and Crew Scheduling Programing (VCSP). In the VCSP the routines of vehicles operation and the crews work duty are defined simultaneously. This is a highly combinatorial problem, which is inserted in the class of NP-hard problems, not having so, polynomial-time algorithms to find optimal solutions.

To analyze the efficiency of the performance of the integrated approach, both VSP and CSP are solved in the traditional sequential and independent way (VSP and CSP are solved separately, in any order and in an unrelated way). The proposed methods can be validated with real data from a Public Transport Company of Belo Horizonte city. The results obtained signalize that the solutions generated by the integrated heuristics are better than those obtained in sequential order.

Sumário

1	Introdução	1
1.1	Justificativa e Objetivos do Trabalho	3
1.2	Organização do Texto	4
2	Revisão Bibliográfica	7
3	Programações Tradicional e Independente de Veículos e Tripulações	13
3.1	Heurísticas	14
3.1.1	Método de Descida	14
3.1.2	Busca Local Iterada	15
3.2	Descrição do Problema de Programação de Veículos	16
3.3	Método de Solução do PPV	19
3.3.1	Representação	19
3.3.2	Estruturas de Vizinhaça	19
3.3.3	Função de Avaliação	20
3.3.4	Procedimento de Geração da Solução Inicial	22
3.3.5	Algoritmo de Resolução	22
3.4	Descrição do Problema de Programação de Tripulações	25
3.5	Método de Solução do PPT	28
3.5.1	Representação	29
3.5.2	Estruturas de Vizinhaça	29
3.5.3	Função de Avaliação	31
3.5.4	Procedimento de Construção das Tarefas	31
3.5.5	Procedimento de Geração da Solução Inicial	32
3.5.6	Algoritmo de Resolução	33
3.6	Programação Independente de Veículos e Tripulações	35
4	Programação Integrada de Veículos e Tripulações	37
4.1	Descrição do Problema de Programação Integrada de Veículos e Tripulações	38

4.2	Método de Solução Iterativa do PPV e PPT	40
4.2.1	Representação	40
4.2.2	Função de Avaliação	41
4.2.3	Algoritmo de Resolução	42
4.3	Método de Solução do PPVT	44
4.3.1	Representação	45
4.3.2	Estruturas de Vizinhaça	49
4.3.3	Função de Avaliação	53
4.3.4	Procedimento de Geração da Solução Inicial	54
4.3.5	Algoritmo de Resolução	54
5	Experimentos Computacionais	59
5.1	Descrição das Instâncias	59
5.2	Resultados	62
6	Conclusões e Trabalhos Futuros	75
	Referências Bibliográficas	79

Lista de Figuras

1.1	Planejamento do sistema de transporte público.	1
3.1	Exemplo de uma solução s^v para o PPV.	20
3.2	Estruturas de vizinhança do PPV.	21
3.3	Bloco de veículo particionado em tarefas.	27
3.4	Exemplo de uma solução s^t para o PPT.	29
3.5	Estruturas de vizinhança do PPT.	30
4.1	Bloco de veículo particionado em tarefas.	41
4.2	Representação de um bloco de veículo.	41
4.3	Bloco de Veículo - 1 e o seu particionamento em tarefas.	46
4.4	Viagens expandidas obtidas do Bloco de Veículo - 1 da Figura 4.3.	47
4.5	Representação de uma solução para o PPVT com definição das viagens expandidas.	49
4.6	Solução para o PPVT utilizada para exemplificar as estruturas de vizinhança.	51
4.7	Exemplo do movimento realocação PPV.	51
4.8	Exemplo do movimento realocação PPT.	51
4.9	Exemplo do movimento super realocação.	52
4.10	Exemplo do movimento troca PPV.	52
4.11	Exemplo do movimento troca PPT.	52
4.12	Exemplo do movimento super troca.	53

Lista de Tabelas

3.1	Tabela de horários.	17
5.1	Descrição das instâncias utilizadas nos experimentos.	60
5.2	Penalizações utilizadas no PPV e seus respectivos valores.	61
5.3	Penalizações utilizadas no PPT e seus respectivos valores.	61
5.4	Resultados considerando os custos das soluções obtidas pelas abordagens independente, seqüencial, iterativa e integrada.	62
5.5	Detalhamento das melhores soluções obtidas pelas abordagens independente, seqüencial, iterativa e integrada.	67

Capítulo 1

Introdução

Devido à sua complexidade o processo de planejamento do sistema de transporte público é normalmente decomposto em subproblemas menores, conforme mostrado na Figura 1.1.

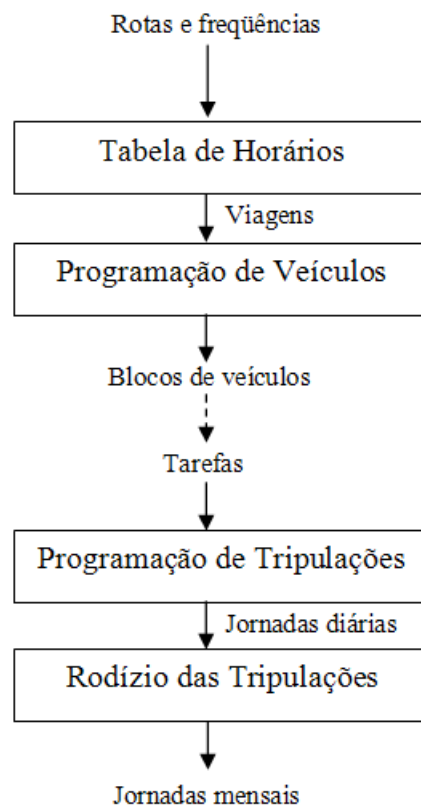


Figura 1.1. Planejamento do sistema de transporte público.

O conjunto de viagens, que compõe a tabela de horários, é definido a partir das rotas a serem percorridas e a frequência com que as mesmas serão atendidas. Estas decisões devem ser tomadas com base na disponibilidade de infra-estrutura, nos serviços

requeridos pelos usuários e em aspectos de demanda.

Dada a tabela de horários passa-se à resolução do Problema de Programação de Veículos (PPV), que consiste em determinar uma rotina diária de operação para uma frota de veículos, tendo por objetivo fazer o seu melhor aproveitamento. Como resultado, obtém-se os chamados “blocos de veículos”. Cada bloco corresponde ao conjunto de viagens realizadas por um mesmo veículo ao longo do dia e mantém informações relativas à seqüência de execução das viagens, seus horários e locais de início e término, os tempos destinados para embarque e desembarque de passageiros e os tempos de espera e de deslocamento fora de operação do veículo.

Uma vez que deve haver uma tripulação (motorista e cobrador) responsável por cada deslocamento realizado por um veículo, o Problema de Programação de Tripulações (PPT) trata da alocação das atividades a serem realizadas por cada tripulação no período de um dia, definindo assim as chamadas jornadas de trabalho das tripulações. Para resolução do PPT, inicialmente os blocos de veículos provenientes da solução do PPV são simplificados em tarefas. Cada tarefa engloba viagens consecutivas de um mesmo veículo e entre as quais não pode ocorrer a troca de tripulações devido a limitações de tempo e espaço. Portanto, a solução do PPT é um conjunto de jornadas diárias de trabalho, em que cada jornada consiste em uma seqüência de tarefas a serem realizadas por uma mesma tripulação e que satisfaz a legislação trabalhista e as regras operacionais da empresa.

Resolvido o PPT para cada dia, o problema seguinte é o da Rotação de Tripulações (PRT). Esse problema consiste em atribuir uma jornada de trabalho mensal para cada tripulação, considerando certas regras trabalhistas relativas a longos períodos de tempo e que, portanto, não foram contempladas na programação diária.

Este trabalho trata especificamente da definição das programações de veículos e tripulações, que juntas, concentram a maior parte dos gastos das empresas do setor de transporte público por ônibus. Comumente PPV e PPT são resolvidos de forma seqüencial tradicional, isto é, primeiro é definido o modo de operação da frota e, em seguida, as jornadas de trabalho das tripulações. Nesta metodologia a resolução do PPT torna-se totalmente dependente da programação de veículos a partir da qual é obtida e isto pode ser visto como uma desvantagem, pois em muitas situações práticas os gastos das empresas com tripulantes superam aqueles relativos aos veículos.

Outras abordagens para tratar PPV e PPT encontradas na literatura são:

1. Seqüencial Inversa: primeiro é resolvido o PPT e em seguida, utilizando os resultados obtidos, é solucionado o PPV. Nesta abordagem é priorizada a melhora da qualidade da programação de tripulações e restringe-se a definição da programação dos veículos. Os resultados encontrados normalmente não são bons, pois

o número de veículos necessários costuma aumentar drasticamente;

2. Independente: PPV e PPT são resolvidos separadamente, em qualquer ordem e de forma a não se relacionarem. Geralmente os resultados obtidos são inviáveis do ponto de vista prático, pois dificilmente as soluções para PPV e PPT são compatíveis. A principal finalidade desta abordagem é a determinação de uma melhor programação para as tripulações desconsiderando os veículos e de uma melhor programação para os veículos desconsiderando as tripulações;
3. Integrada: PPV e PPT são resolvidos simultaneamente. Trata-se de uma abordagem promissora, uma vez que a forte dependência entre estes problemas sugere que uma resolução integrada possa resultar em consideráveis reduções de custo.

1.1 Justificativa e Objetivos do Trabalho

Segundo Sampaio et al. [2006] o transporte público é considerado uma prioridade governamental em todo o mundo, pois trata-se de uma necessidade para locomoção das pessoas, predominantemente, mas não exclusivamente, para as de renda mais baixa. Além disso, o transporte coletivo atende a um número mais amplo de passageiros que o transporte individual, sem sobrecarregar a infra-estrutura rodoviária e permite substanciais reduções nas emissões de poluentes ao meio ambiente, dada a menor quantidade de veículos por passageiros.

Mais especificamente, nas cidades brasileiras observa-se que juntamente com o crescimento da população urbana vem aumentando a demanda por um serviço de transporte público barato, eficiente e de qualidade. Conforme a ANTP [1999], para que ocorra a democratização do acesso ao transporte público, deve haver a garantia de tarifas toleráveis para os usuários, o que depende em grande parte do aumento da eficiência e redução de custos por parte das organizações do setor. Justificando assim, qualquer iniciativa que venha a proporcionar melhoras no planejamento do sistema de transporte público.

No que se refere ao Sistema de Transporte Público de Ônibus Urbano, determinar uma rotina diária de operação para uma frota de veículos de uma empresa bem como definir as jornadas de trabalho de suas tripulações (motoristas e cobradores) são tarefas muito complexas. Quando estas são realizadas de forma manual, requerem grande esforço e tempo dos profissionais e nem sempre as programações geradas são de qualidade. As dificuldades inerentes à resolução destes problemas devem-se principalmente a uma infinidade de restrições e objetivos a serem alcançados e que são fortemente de-

pendentes das políticas operacionais e das leis trabalhistas consideradas. Evidenciando assim, a necessidade da utilização de recursos computacionais.

Normalmente, no processo de planejamento do sistema de transporte público, a programação de veículos é realizada de forma independente e em uma etapa anterior à programação de tripulações. Nesta metodologia a resolução do PPT torna-se totalmente condicionada à programação de veículos a partir da qual será obtida e, portanto, este pode ser um dos fatores que limita a melhora da qualidade da programação de tripulações. Esta abordagem foi fortemente criticada por Ball et al. [1983], que ao estudar uma empresa de transporte público da América do Norte verificou que o custo associado à programação das tripulações supera o custo relacionado à programação dos veículos, e em muitos casos chega a corresponder a mais de 80% do custo operacional total.

Portanto, neste trabalho é proposta uma estratégia para resolução do Problema de Programação Integrada de Veículos e Tripulações (PPVT). No PPVT são geradas simultaneamente as programações dos veículos e dos tripulantes possibilitando encontrar soluções com custo operacional menor que aquele obtido pela solução seqüencial do PPV e PPT. Além disso, com o intuito de identificar a efetividade da abordagem integrada, PPV e PPT também são resolvidos de forma seqüencial tradicional, independente e iterativa (em que, ora a busca por melhoras se concentra especificamente no PPV e ora no PPT). O objetivo é avaliar a metodologia proposta, a fim de que seja encontrada uma maneira eficiente de especificar a forma de utilização da frota e da mão-de-obra de uma empresa de transporte público, de modo a minimizar ao máximo os custos envolvidos em tal atividade, respeitando as regras operacionais e as leis trabalhistas intrínsecas ao problema. Para tanto, são resolvidos problemas reais referentes ao Sistema de Transporte Público da cidade de Belo Horizonte, capital do estado de Minas Gerais, permitindo que seja adquirido conhecimento sobre as características e dificuldades da aplicação da metodologia integrada à realidade brasileira.

1.2 Organização do Texto

A seguir, no Capítulo 2, é feita uma revisão da literatura sobre o problema que norteia todo esse trabalho, o Problema de Programação Integrada de Veículos e Tripulações (PPVT). Modelos e algoritmos propostos para tratar o PPVT são apresentados.

O Capítulo 3 trata separadamente dos problemas de Programação de Veículos (PPV) e de Programação de Tripulações (PPT), considerando duas abordagens distintas, a seqüencial tradicional e a independente. O objetivo é descrever detalhadamente os problemas e a metodologia utilizada para resolvê-los.

O Capítulo 4 apresenta o Problema de Programação Integrada de Veículos e Tripulações. Inicialmente é proposto um método que aborda PPV e PPT de forma iterativa e, em um segundo passo, é apresentada uma estratégia de resolução totalmente integrada dos problemas.

No Capítulo 5 são descritos, respectivamente, as instâncias utilizadas e os resultados obtidos nos experimentos realizados, considerando as abordagens apresentadas nos capítulos 3 e 4.

Por fim, no Capítulo 6, são apresentados as conclusões e os trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Embora haja um entendimento de que a resolução integrada das programações de veículos e tripulações seja vantajosa, a maioria dos trabalhos encontrados na literatura segue a abordagem seqüencial na qual a programação de veículos é definida a priori e independentemente da programação de tripulações.

O Problema de Programação de Veículos há muito tempo vem sendo pesquisado e resolvido com o auxílio de computadores. Os primeiros trabalhos tratavam o PPV tanto utilizando métodos exatos [Kirkman, 1968] quanto procedimentos heurísticos [Saha, 1970; Wren, 1972]. O Problema de Programação de Tripulações também tem recebido considerável atenção na literatura e é estudado desde a década de 60, sendo Elias [1964] um dos precursores. Pesquisas mais recentes para resolução do PPV podem ser encontradas em Baita et al. [2000], Silva [2001] e Souza et al. [2007] e para o PPT em Shen e Kwan [2001], Daduna e Voss [2001], Marinho [2005] e em Santos e Mateus [2009].

O Problema de Programação Integrada de Veículos e Tripulações somente começou a ser estudado recentemente, uma vez que devido à sua grande complexidade, o custo computacional desta metodologia sempre foi restritivo. Ball et al. [1983] foram os primeiros a propor um método de resolução para o PPVT. O procedimento apresentado pelos autores envolve a definição de uma rede de programação, consistindo de vértices indicando viagens que devem ser executadas por um veículo e uma tripulação (vértices *d-trip*), e dois outros vértices *s* e *t* representando a garagem. Diversos tipos de arcos podem ser agrupados em duas categorias, aqueles que indicam que uma tripulação e veículo seguem de uma *d-trip* para outra e aqueles que indicam que apenas a tripulação realiza este deslocamento (arcos *crew-only*). O procedimento de resolução é decomposto em três componentes. Inicialmente são criadas partes de jornadas de trabalho, limitadas por uma duração máxima dada por uma constante *T*. É importante relembrar que uma jornada corresponde às atividades a serem realizadas

por uma mesma tripulação em um dia de trabalho. Em seguida, aquelas partes de jornadas obtidas que apresentam pequena duração são colocadas aos pares, formando assim, jornadas parciais. Por fim, longas partes de jornadas e jornadas parciais são combinadas resultando em um conjunto de jornadas de trabalho para as tripulações (solução do PPT). A programação dos veículos é gerada simultaneamente ignorando os arcos exclusivos das tripulações (arcos *crew-only*) presentes na solução. Heurísticas similares são propostas por Tosini e Vercellis [1988] e Patrikalakis e Xerocostas [1992].

A primeira formulação matemática para o PPVT foi proposta em Patrikalakis e Xerocostas [1992]. No entanto, este modelo era apenas para fins ilustrativos sendo computacionalmente intratável [Freling et al., 1999]. Assim, considera-se que Freling et al. [1995] foram os pioneiros em propor uma formulação de programação inteira para o problema e deram origem a uma série de pesquisas desenvolvidas por Freling e co-autores sobre o tema, tais como: Freling [1997], Freling et al. [1999] e Freling et al. [2003]. As formulações apresentadas nestes trabalhos são similares e constituídas basicamente de duas partes:

1. Uma formulação de quase atribuição baseada em uma rede $G(V, A)$ que assegura a viabilidade do Problema de Programação de Veículos. Nesta rede V é o conjunto de todas as viagens a serem realizadas e mais dois nós, s (origem) e t (destino), ambos representando a garagem. O conjunto A corresponde aos arcos que conectam s e t a todos os demais nós em V e aos arcos que ligam viagens compatíveis. Salienta-se que duas viagens são compatíveis se elas podem ser realizadas por um mesmo veículo. Desta forma, uma solução viável para o PPV é um conjunto de caminhos disjuntos de s para t na rede G , de tal forma que cada nó viagem em V seja coberto exatamente uma vez;
2. Restrições de particionamento de conjunto que garantem que cada viagem e cada deslocamento realizado pelo veículo fora de operação, que façam parte da solução do PPV, sejam cobertos pela jornada de trabalho de alguma tripulação.

Em Haase e Friberg [1999] foi apresentada a primeira abordagem de solução exata para o Problema Integrado de Programação de Veículos e Tripulações. O modelo proposto combina as abordagens de Desrochers e Soumis [1989] e Ribeiro e Soumis [1994], em que o PPVT é formulado como um problema de particionamento de conjunto com restrições adicionais que são utilizadas para relacionar as programações de veículos e de tripulações. Na metodologia proposta uma coluna no modelo representa o bloco de um veículo ou a jornada de trabalho de uma tripulação. Estas colunas correspondem a caminhos em dois grafos, um para a programação de veículos e outro para a programação das tripulações. Destaca-se que restrições de recursos foram inseridas

no grafo relacionado às tripulações com o intuito de limitar o tempo de trabalho e o número de paradas em uma jornada. Soluções ótimas são obtidas por um algoritmo *branch-and-price-and-cut* em que a técnica de geração de colunas é utilizada para determinar tanto blocos de veículos, quanto jornadas de trabalho para as tripulações. Com o objetivo de melhorar os limites inferiores obtidos pela relaxação linear do problema, cortes poliédricos como propostos em Hoffman e Padberg [1993] são considerados para os veículos. A modelagem proposta é suficientemente genérica para ser estendida para múltiplas garagens e pode receber restrições adicionais, porém, somente pequenas instâncias envolvendo no máximo 20 viagens são resolvidas na otimalidade em um tempo razoável de CPU.

Em Freling et al. [2003] foram apresentadas três metodologias para resolução dos problemas de Programação de Veículos e Programação de Tripulações: A abordagem seqüencial em que primeiro resolve-se o PPV e a partir da solução obtida o PPT é solucionado; a programação independente das tripulações, isto é, o PPT é resolvido sem considerar a solução obtida no PPV; e a abordagem integrada em que PPV e PPT são resolvidos simultaneamente. Procurou-se neste trabalho identificar os benefícios da programação integrada de veículos e tripulações em comparação com a seqüencial. Para tanto, foram considerados problemas reais de uma empresa de transporte público da Holanda e avaliou-se tanto a situação em que uma tripulação pode realizar troca de veículos durante sua jornada de trabalho, quanto a situação em que esta troca de veículos não é permitida. Modelos matemáticos são apresentados para as três abordagens: Seqüencial, independente e integrada. Para resolução do PPVT foi proposta uma heurística lagrangeana que faz uso da técnica de geração de colunas no processo de resolução do problema relaxado. Nesta abordagem o subproblema lagrangeano considerado é o problema de programação de veículos com uma única garagem, que é resolvido pelo algoritmo proposto em Freling et al. [2001]. Durante o processo de resolução do problema relaxado o conjunto de jornadas de trabalho das tripulações é atualizado, de tal forma que todas as jornadas com custo reduzido negativo sejam consideradas. Como não é possível gerar todas as jornadas das tripulações de imediato (não se tem uma programação de veículos previamente definida), isto é feito dinamicamente a partir de um processo que inicialmente cria *partes de jornadas* e em seguida combina estas partes formando jornadas. Neste contexto, uma *parte de jornada* é entendida como uma seqüência de tarefas provenientes de um mesmo bloco de veículo e que podem ser realizadas por uma mesma tripulação sem interrupções. Destaca-se que cada tarefa continua sendo definida como o agrupamento de viagens de um mesmo veículo que devem ser obrigatoriamente realizadas por uma mesma tripulação. Resultados computacionais mostram a eficiência da abordagem integrada em diferentes situações,

principalmente naquelas em que uma tripulação não pode trocar de veículo.

Laurent e Hao [2007] tratam do problema particular de uma companhia de aluguel de limousines que atua na cidade de Paris (França). Dado um conjunto de viagens diárias, com seus respectivos horários e locais de início e término e número de passageiros, deseja-se alocar simultaneamente, veículos e motoristas, de tal forma que sejam atendidas tantas viagens demandadas quanto possível. Destaca-se que cada viagem, no problema abordado, requer um motorista que tenha certa aptidão e um veículo com determinadas características. Outra particularidade refere-se ao fato da carga de trabalho da companhia ser dinâmica, pois constantemente viagens são agendadas, canceladas e modificadas, conforme necessidades dos clientes. Assim, o objetivo é, ao fim de cada dia, determinar as jornadas de trabalho dos veículos e motoristas visando otimizar a utilização de recursos disponíveis, atender certas restrições e ao mesmo tempo priorizar a qualidade do serviço considerando as viagens até então agendadas para o próximo dia. Dessa forma, posteriores mudanças no quadro de viagens são tratadas manualmente. Para resolver o problema foi utilizado um algoritmo de duas fases. Inicialmente uma solução é obtida utilizando a técnica de programação por restrições [Tsang, 1993], da seguinte forma: um procedimento construtivo guloso primeiramente dispõe as viagens em ordem decrescente de duração e, em seguida, seqüencialmente determina um par motorista-veículo para realizar cada uma delas. Após cada atribuição, um procedimento prevê futuros conflitos e elimina as possibilidades dos mesmos ocorrerem. Esta fase de construção é finalizada quando há um par motorista-veículo alocado a cada viagem ou quando nenhuma atribuição pode mais ser realizada sem que haja violação de alguma restrição. Na segunda fase a solução gerada é melhorada por um procedimento baseado na metaheurística *Simulated Annealing*. Para tanto, foram elaborados cinco mecanismos de exploração da vizinhança sendo o espaço de busca composto apenas por soluções viáveis. A avaliação de uma solução é feita considerando vários objetivos, cada qual com um peso associado. Este peso é definido a partir de um parâmetro de entrada que reflete a importância do objetivo ao qual se refere em conjunto com um método de aprendizagem que, a partir de várias execuções de diferentes instâncias, estima os valores dos atributos da solução baseado no seu número de viagens. Testes foram realizados com 10 quadros de viagens diferentes da companhia de aluguel de limousines. Os resultados obtidos foram comparados com aqueles gerados manualmente e constatou-se que utilizando menos tempo o algoritmo proposto foi capaz de gerar programações de qualidade muito superior.

Outra variação do PPVT aparece quando este é resolvido considerando que a frota de veículos, disponível para realização das viagens, se encontra espalhada em diferentes garagens. Tal problema é conhecido na literatura inglesa como *Multiple-Depot Vehicle*

and Crew Scheduling Problem - MDVCSP.

Em Mesquita e Paias [2008] são propostas duas formulações para o MDVCSP. A primeira (SP-VCSP) é similar a uma apresentada em Huisman et al. [2005], porém com um menor número de restrições e variáveis de decisão. Esta formulação combina dois modelos, um de fluxos em rede multi-produto para a programação de veículos e outro de particionamento de conjunto para a programação de tripulações. A segunda formulação (SPC-VCSP) é uma extensão da anterior em que algumas restrições de particionamento de conjunto são trocadas por restrições de cobertura de conjunto. Esta modificação tornou o modelo mais flexível e viabilizou as situações em que era permitido à tripulação trocar de veículos durante a sua jornada de trabalho sem, contudo, aumentar a complexidade do problema. A metodologia de resolução proposta pelas autoras consiste em quatro etapas:

1. Definição do conjunto de tarefas: Para formação das tarefas (agrupamento de viagens que devem ser realizadas por uma mesma tripulação) primeiramente é necessário definir a programação de veículos. Desconsiderando a necessidade de um veículo retornar sempre para a garagem de origem, o Problema de Programação de Veículos com Múltiplas Garagens conforme formulado no trabalho pôde ser resolvido em tempo polinomial (problema de atribuição). Assim, cada bloco de veículo obtido é analisado separadamente de tal forma que duas viagens consecutivas são agrupadas em uma única tarefa sempre que o tempo compreendido entre elas seja inferior a um dado valor e não se viole nenhuma restrição relativa a uma jornada da tripulação;
2. Obtenção do conjunto de jornadas inicial: Um conjunto inicial de jornadas de trabalho para as tripulações se faz necessário para que seja resolvido o problema de *pricing* no processo de geração de colunas (etapa 3). Estas jornadas também foram obtidas resolvendo o Problema de Programação de Veículos com Múltiplas Garagens sem a restrição de que os veículos retornem à garagem de origem. No entanto, nesta abordagem manipulam-se tarefas ao invés de viagens e se obtém jornadas no lugar de blocos de veículos. Ressalta-se que neste processo jornadas inviáveis podem ser construídas, assim aqueles blocos que excedem a duração máxima permitida para uma jornada são particionados e os blocos que violem qualquer outra restrição são mantidos, porém atribui-se um custo elevado a eles;
3. Resolução da relaxação linear do problema utilizando a técnica de geração de colunas: É definida uma Rede Jornada, tal que, um caminho na rede corresponde a uma jornada viável. A jornada com o menor custo reduzido é um caminho mínimo na Rede Jornada. Assim, como se deseja inserir várias colunas (jorna-

das) no problema mestre a cada iteração, o problema de *pricing* é resolvido por programação dinâmica. Nesta abordagem, com o intuito de agilizar o processo de resolução, durante os estágios da programação dinâmica são eliminados alguns estados, de duas maneiras: Sempre que o custo reduzido da jornada sendo formada exceder um dado valor pré-definido e de forma randômica;

4. Aplicação de um procedimento de *branch and bound* caso a solução obtida na etapa anterior não seja inteira: Basta realizar *branching* sobre as variáveis referentes à programação de veículos ou à programação de tripulações, conforme provado neste trabalho.

Capítulo 3

Programações Tradicional e Independente de Veículos e Tripulações

Conforme mencionado anteriormente existem diferentes maneiras de tratar os problemas de Programação de Veículos e Programação de Tripulações. Nesse capítulo, PPV e PPT, são resolvidos seguindo as estratégias seqüencial tradicional e independente.

Na abordagem seqüencial tradicional inicialmente é definida a forma de operação dos veículos e, em seguida, baseando-se na programação obtida para o PPV, são definidas as jornadas de trabalho dos tripulantes. Essa metodologia de resolução, por si só, já é interessante visto que PPV e PPT são problemas de grande relevância, havendo inclusive, muitos trabalhos na literatura que tratam dos mesmos separadamente. Além disso, as idéias aqui apresentadas se mostram importantes uma vez que são exploradas no decorrer de todo o trabalho e permitem comparar os desempenhos das abordagens, seqüencial e integrada, para definição das programações de veículos e tripulações considerando problemas reais.

Esse capítulo está organizado conforme descrito a seguir. Inicialmente, na seção 3.1, são definidos conceitos básicos sobre o desenvolvimento de heurísticas e é apresentada a metaheurística de Busca Local Iterada, tratam-se de técnicas utilizadas nesse trabalho. Nas seções 3.2 e 3.3 são descritos, respectivamente, o Problema de Programação de Veículos e a metodologia utilizada para resolvê-lo. Na seção 3.4 é apresentado o Problema de Programação de Tripulações e, posteriormente, na seção 3.5, o método de resolução proposto para o PPT é mostrado. Por fim, a seção 3.6 trata do Problema de Programação Independente de Veículos e Tripulações, em que PPV e PPT são resolvidos separadamente e sem se relacionarem.

3.1 Heurísticas

Observa-se que no PPV e PPT o número de combinações, em termos de blocos de veículos e jornadas das tripulações, cresce exponencialmente com o tamanho da instância sendo considerada. Tratam-se de problemas altamente combinatoriais e que se enquadram na classe de problemas NP-difícil, não havendo, portanto, algoritmos em tempo polinomial para encontrar soluções ótimas. Dessa forma, torna-se necessária uma abordagem heurística para solução de problemas maiores.

As heurísticas são algoritmos exploratórios que visam obter uma boa solução a um custo computacional aceitável, sem contudo, garantir a otimalidade ou quão próxima dela a solução se encontra.

As heurísticas são classificadas em construtivas ou de melhoria. Uma heurística construtiva tem por objetivo determinar uma solução, elemento por elemento. A forma de escolha de cada elemento a ser inserido na solução pode variar e depende da estratégia de construção adotada. As heurísticas de melhoria, por sua vez, têm a finalidade de melhorar uma solução construída e normalmente são baseadas no conceito de *vizinhança*. Sendo S o espaço de pesquisa de um problema de otimização, a função N associa a cada elemento $s \in S$ um conjunto $N(s) \subseteq S$. O conjunto $N(s)$ é chamado de *vizinhança* de s e cada solução $s' \in N(s)$ é chamada de *vizinha* de s . Cada solução s' é obtida de s a partir de uma operação chamada de *movimento*, que define a transição entre soluções através de pequenas modificações.

3.1.1 Método de Descida

O Método de Descida é uma técnica iterativa que partindo de uma solução inicial qualquer explora toda a vizinhança de diferentes soluções, até atingir um ótimo local, do qual o método não é capaz de escapar.

Inicialmente, o Método de Descida explora toda a vizinhança de sua solução corrente. Se o melhor vizinho encontrado proporcionar um aprimoramento no valor da função de avaliação (função que quantifica a qualidade de uma solução) até então conhecido, ele é aceito como a nova solução a ser explorada. Caso contrário, um ótimo local foi alcançado e a solução corrente é retornada.

O Algoritmo 1 mostra o pseudo-código do Método de Descida, cujo objetivo é minimizar uma função de avaliação f .

O Método Randômico de Descida é uma variante do Método de Descida e sua finalidade é evitar, a cada iteração, a busca exaustiva em toda a vizinhança da solução corrente.

A idéia básica dessa técnica é obter, a cada iteração, um vizinho qualquer da solução

Algoritmo 1: Método de Descida.

```

1 Seja  $s_0$  uma solução inicial conhecida
2  $s \leftarrow s_0$ , onde  $s$  é a solução corrente
3  $s' \leftarrow \text{melhorVizinho}(s)$ , onde  $s'$  é o melhor dentre todos os vizinhos de uma
  dada solução  $s$ 
4 enquanto  $f(s') < f(s)$  faça
5    $s \leftarrow s'$ 
6    $s' \leftarrow \text{melhorVizinho}(s)$ 
7 Retorna a melhor solução encontrada,  $s$ 

```

corrente, admitindo-o como nova solução corrente caso ele proporcione melhoras no valor da função de avaliação. Caso contrário, a solução corrente é mantida inalterada. Esse procedimento se repete até que uma condição de parada seja satisfeita.

No Algoritmo 2 é apresentado o pseudo-código do Método Randômico de Descida aplicado à minimização de uma função de avaliação f .

Algoritmo 2: Método Randômico de Descida.

```

1 Seja  $s_0$  uma solução inicial conhecida
2  $s \leftarrow s_0$ , onde  $s$  é a solução corrente
3 enquanto condição de parada não for satisfeita faça
4   Seja  $s' \in N(s)$  um vizinho qualquer da solução  $s$ 
5   se  $f(s') < f(s)$  então
6      $s \leftarrow s'$ 
7 Retorna a melhor solução encontrada,  $s$ 

```

3.1.2 Busca Local Iterada

As metaheurísticas podem ser vistas como sendo heurísticas genéricas mais sofisticadas, que visam explorar inteligentemente as instâncias do problema e o seu espaço de soluções. Elas se caracterizam pela habilidade de escapar de ótimos locais.

A metaheurística Busca Local Iterada - *ILS* (*Iterated Local Search*) tem como idéia principal focar a pesquisa por melhoras apenas no subespaço definido pelas soluções que são localmente ótimas segundo um dado procedimento de otimização, e não em todo o espaço de soluções. Para tanto é proposto o uso de um método de busca local combinado com a aplicação de perturbações (modificações realizadas em uma solução).

Assim, a cada iteração do *ILS* é realizada uma perturbação sobre a solução corrente seguida de um procedimento de busca local que melhorará a solução perturbada. Essa nova solução é então avaliada e caso seja satisfeito o critério de aceitação (por exemplo, se houve melhora na função de avaliação), ela será considerada a solução corrente. Caso

contrário, a solução corrente é mantida inalterada. Esse procedimento é repetido até que a condição de parada seja satisfeita.

No Algoritmo 3 é apresentada a metaheurística *ILS* básica [Lourenço et al., 2003] em que se identificam quatro componentes:

1. *geraSolucaoInicial*: que gera uma solução inicial s_0 para o problema;
2. *buscaLocal*: que retorna uma solução melhorada s após geração da solução inicial e outra s'' após perturbação;
3. *perturbacao*: que modifica a solução corrente s guiando a uma solução intermediária s' . Uma perturbação eficiente é aquela que não altera drasticamente a solução corrente, mas, ainda assim, permite que a busca local explore diferentes regiões;
4. *criterioAceitacao*: decisão que se refere tanto à aceitação da solução perturbada como solução corrente, quanto quão intensa será a próxima perturbação a ser aplicada (isto é, determinação do nível de perturbação).

Algoritmo 3: ILS - Busca Local Iterada.

```

1  $s_0 \leftarrow \text{geraSolucaoInicial}()$ 
2  $s \leftarrow \text{buscaLocal}(s_0)$ , onde  $s$  é a solução corrente
3 enquanto condição de parada não for satisfeita faça
4    $s' \leftarrow \text{perturbacao}(\text{historico}, s)$ 
5    $s'' \leftarrow \text{buscaLocal}(s')$ 
6    $s \leftarrow \text{criterioAceitacao}(s, s'', \text{historico})$ 
7 Retorna a solução  $s$ 
```

Destaca-se que algoritmos mais eficientes são obtidos quando tanto a perturbação quanto o critério de aceitação dependem das soluções obtidas em iterações anteriores, ou seja, utiliza-se do recurso de memória (representada pelo *historico* no Algoritmo 3).

3.2 Descrição do Problema de Programação de Veículos

O Problema da Programação de Veículos (PPV) consiste em criar uma rotina diária de operação para uma frota de veículos disponibilizada para uso de uma empresa, de modo que, todas as viagens a ela responsabilizadas sejam realizadas e a programação gerada não viole as regras operacionais e as limitações de tempo e espaço existentes.

A programação dos veículos é feita a partir de uma tabela de horários que é definida em uma etapa anterior a resolução do PPV e na qual se encontram todas as viagens a serem realizadas em um determinado dia por uma empresa. A Tabela 3.1 ilustra um fragmento de uma tabela de horários.

Tabela 3.1. Tabela de horários.

Número da Viagem	Horário de Início	Ponto Inicial	Horário de Término	Ponto Final	Linha	Tempo de Embarque	Tempo de Desembarque
1	5	30	70	30	30.4	1	1
2	5	302	38	302	302.4	1	1
3	20	30	76	30	30.4	1	1
4	240	302	265	302	302.4	1	1
5	255	30	309	30	30.3	1	2
6	260	302	281	302	302.1	1	1
7	270	305	335	305	305.1	1	2
8	270	30	331	30	30.3	1	2
9	270	326	300	326	326.1	1	2
10	280	30	340	30	30.3	1	2

Na tabela de horários, cada viagem é identificada univocamente por meio de um conjunto de características que consistem em informações relevantes no processo de resolução do problema. Organizados em colunas e aparecendo seqüencialmente na tabela, temos: o número da viagem, horário de início da viagem (em minutos), o ponto inicial que corresponde ao local de início da viagem, horário de término da viagem (em minutos), o ponto final que se refere ao local em que a viagem termina, a linha a qual a viagem pertence, e finalmente as duas últimas colunas que se referem respectivamente aos tempos destinados ao embarque e desembarque de passageiros, dados em minutos, e que não necessariamente são iguais, sendo o primeiro dependente do horário de início da viagem e o segundo do horário de fim da mesma.

Para exemplificar, considere a primeira viagem “1” apresentada na Tabela 3.1. Essa viagem começa no ponto “30” aos 5 minutos do dia, ou seja, às 00:05 horas e é finalizada no ponto “30” aos 70 minutos do dia, que corresponde às 01:10 horas. As duas últimas colunas indicam que o veículo que fará a viagem “1” permanecerá um minuto no terminal para embarque e após chegar ao destino mais um minuto de terminal deverá ser cumprido para permitir o desembarque de passageiros.

Além do conjunto de viagens, devem ser fornecidas também informações relativas aos tempos de deslocamento entre os vários pontos de parada dos veículos, incluindo a garagem. De posse dessas informações é possível determinar o tempo que um veículo gasta fora de operação, ou seja, sem passageiros, para sair de um ponto e chegar a outro. A este tipo de deslocamento é dado o nome de viagem morta.

Para que a programação de atividades da frota seja factível é necessário atender a uma série de restrições, as quais podem variar conforme as políticas operacionais que regem o Sistema de Transporte Público Urbano por Ônibus no qual a empresa atua. No problema sendo abordado é considerado que todos os veículos da frota são idênticos e que partem de uma mesma garagem. As restrições que serão levadas em consideração são as seguintes:

- (a) Todas as viagens de responsabilidade de uma empresa, sem exceções, devem ser realizadas por algum veículo da frota;
- (b) Antes de se iniciar e após finalizar uma viagem, um veículo deve obrigatoriamente cumprir um tempo mínimo no terminal, que corresponde ao tempo necessário para o embarque e desembarque dos passageiros;
- (c) Um mesmo veículo somente poderá realizar duas viagens, i e j , consecutivamente, se o intervalo de tempo existente entre elas for suficiente para que o veículo desloque do ponto final da viagem i para o ponto inicial da viagem j , respeite os tempos destinados para embarque e desembarque de passageiros e inicie a viagem j sem atrasos. Caso contrário, ocorrerá uma sobreposição de horários entre as viagens;
- (d) Um veículo não deve ficar mais de duas horas em um terminal esperando para executar a sua próxima viagem. Se o tempo de espera exceder esse limite, o veículo deverá se deslocar para a garagem. Diz-se, nesse último caso, que o veículo está realizando uma dupla pegada;
- (e) O número de duplas pegadas da programação não pode exceder 60% do tamanho da frota;
- (f) Todo veículo deve iniciar e terminar a sua jornada de trabalho na garagem;
- (g) Todo veículo deve permanecer, no mínimo, trinta minutos consecutivos na garagem, por dia, para que ocorra a sua manutenção e limpeza. Este tempo pode ser cumprido quando o veículo pára na garagem durante uma dupla pegada ou mesmo depois de completada toda a sua jornada operacional diária.

Neste trabalho o objetivo é obter uma programação de veículos em que o tamanho da frota utilizada e os deslocamentos dos veículos fora de operação (deslocamentos que não equivalem a nenhuma viagem) sejam minimizados.

A solução para o Problema de Programação de Veículos consiste, portanto, em atribuir um conjunto de viagens para um número de veículos menor ou igual à frota

disponível pela empresa, considerando as restrições apresentadas acima, de tal forma que todas as viagens sejam realizadas com o menor custo operacional possível. Ao se resolver o PPV tem-se como resultado os chamados blocos dos veículos. Cada um destes blocos corresponde a um conjunto de viagens que serão executadas por um mesmo veículo.

3.3 Método de Solução do PPV

Nesta seção é apresentada a metodologia proposta para resolver o PPV. Na subseção 3.3.1 mostra-se como uma solução para o problema é representada. Na subseção 3.3.2 são apresentadas as estruturas de vizinhança desenvolvidas para explorar o espaço de busca. Na subseção 3.3.3 é mostrada a função de avaliação que foi elaborada para quantificar a qualidade de uma solução. Descreve-se na subseção 3.3.4 o procedimento de geração da solução inicial. Por fim, na subseção 3.3.5, é apresentado o algoritmo proposto para resolver o problema.

3.3.1 Representação

Uma solução s^v para o PPV consiste em uma lista de veículos, sendo que a cada veículo está associada uma lista de viagens a serem por ele executadas durante um dia de trabalho. É pertinente ressaltar que as viagens realizadas por cada veículo são mantidas ordenadas por seus horários de início. Esta abordagem é importante, uma vez que através dela será possível avaliar a jornada de trabalho de um veículo sob vários aspectos, tais como: tempo de locomoção fora de operação (viagem morta), tempo de espera no terminal e total de retornos à garagem.

A Figura 3.1 ilustra uma solução s^v para o PPV, na qual uma frota de quatro veículos deve executar doze viagens a ela alocadas.

3.3.2 Estruturas de Vizinhança

Para explorar o espaço de soluções do PPV foram utilizados três tipos de movimentos que definem, respectivamente, as estruturas de vizinhança $VN^{(R)}$, $VN^{(T)}$ e $VN^{(RT)}$, são eles: realocação de viagem, troca de viagens e realocação-troca de viagens. A realocação consiste em transferir uma viagem pertencente a um veículo para outro veículo. O movimento de troca corresponde em permutar uma viagem de um dado veículo com uma viagem pertencente a outro veículo. Já o movimento de realocação-troca de viagens envolve três veículos, v_1 , v_2 e v_3 , o veículo v_1 tem uma de suas viagens transferida para o veículo v_2 e o veículo v_2 , por sua vez, cede uma de suas viagens (desde

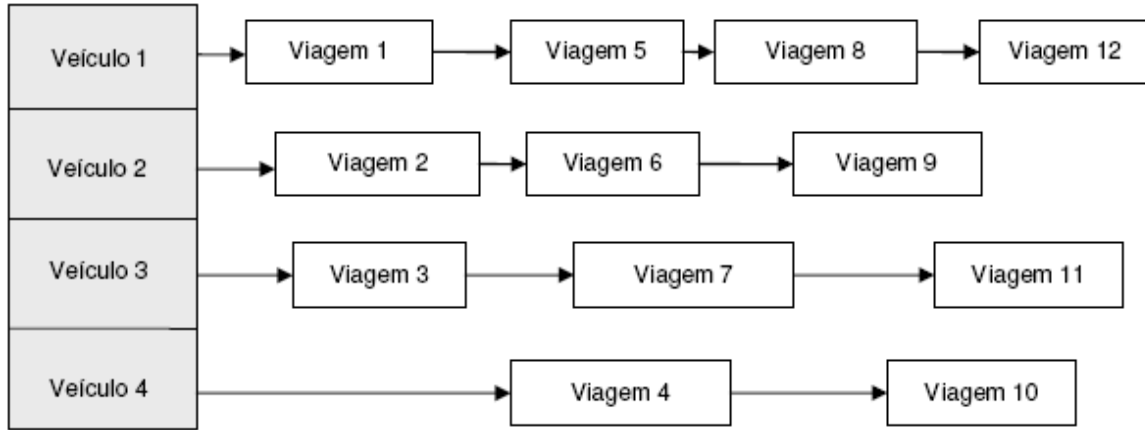


Figura 3.1. Exemplo de uma solução s^v para o PPV.

que não seja a viagem proveniente de v_1) para o veículo v_3 . Portanto, neste movimento o veículo v_2 troca uma de suas viagens a partir de duas realocações sucessivas.

Na Figura 3.2 estão ilustrados estes movimentos.

3.3.3 Função de Avaliação

A função elaborada para quantificar a qualidade de uma solução para o PPV é baseada na penalização dos atributos indesejáveis da programação. Portanto, uma solução s^v para o Problema de Programação de Veículos é avaliada com base na seguinte função, a qual deve ser minimizada:

$$\begin{aligned}
 f^v(s^v) = & \text{totalVeiculos} \times \text{custoUsoVeiculo} \\
 & + \text{tempoViagemMorta} \times \text{custoViagemMorta} \\
 & + \text{totalExcessoDP} \times \text{pesoInviabilidade} \\
 & + \text{tempoSobreposicao} \times \text{pesoInviabilidade}
 \end{aligned} \tag{3.1}$$

em que:

- (a) totalVeiculos é a quantidade de veículos utilizados;
- (b) custoUsoVeiculo é o custo associado ao uso de um veículo;
- (c) tempoViagemMorta é o tempo total, em minutos, que foi despendido com os deslocamentos realizados fora de operação (viagem morta), considerando todos os veículos da frota;

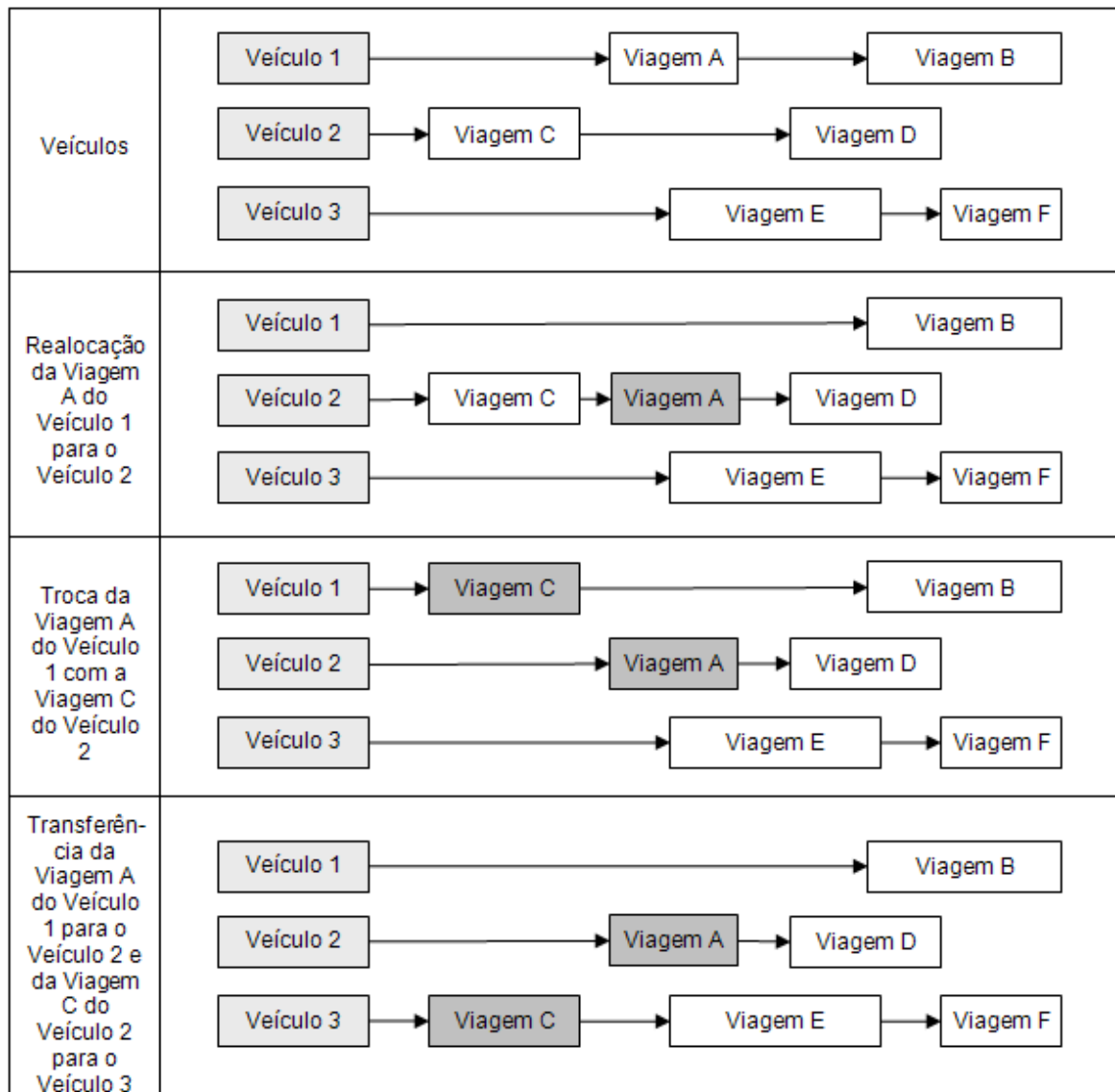


Figura 3.2. Estruturas de vizinhança do PPV.

- (d) *custoViagemMorta* é o custo associado a cada minuto de viagem morta realizada por um veículo;
- (e) *totalExcessoDP* é a quantidade de duplas pegadas da programação que supera um certo número admissível, previamente definido;
- (f) *tempoSobreposicao* é a soma, dada em minutos, de todos os intervalos de tempo em que há sobreposição de horários entre viagens de um mesmo veículo da programação. Também será considerado sobreposição, o tempo faltante para que sejam completados os trinta minutos diários na garagem de cada veículo que não contemple esta restrição. Destaca-se que na sobreposição de horários entre duas

viagens estão embutidos os tempos para embarque e desembarque de passageiros e de deslocamento do veículo que não foram respeitados;

(g) *pesoInviabilidade* é o peso dado a cada inviabilidade presente na solução.

3.3.4 Procedimento de Geração da Solução Inicial

A programação de veículos inicial é gerada por uma heurística construtiva gulosa. Em heurísticas deste tipo uma solução é construída iterativamente, elemento por elemento, tomando-se a cada passo a melhor decisão segundo um dado critério. No algoritmo proposto, inicia-se com uma frota formada por um único veículo com jornada vazia e um conjunto de viagens a serem realizadas, dispostas ordenadas crescentemente pelo horário de início. A cada passo, remove-se a primeira viagem do conjunto e a aloca na jornada de trabalho diária de algum veículo da frota. Esta combinação deve ser a que implique no menor custo possível de operação determinado pela aplicação da função de avaliação descrita anteriormente pela expressão (3.1). À medida que a solução vai sendo gerada novos veículos são criados dinamicamente, de tal maneira que, sempre seja avaliada a atribuição de uma viagem a um veículo com jornada vazia. Este processo é repetido até que não haja mais viagens a serem alocadas.

O Algoritmo 4, a seguir, mostra o pseudo-código da heurística construtiva gulosa utilizada no PPV.

3.3.5 Algoritmo de Resolução

Na resolução do PPV foram utilizados procedimentos de busca local baseados no Método de Descida e no Método Randômico de Descida, descritos na subseção 3.1.1 na página 14, são eles:

1. *DescidaComRealocacao_PPV* e *DescidaRandomicaComRealocacao_PPV* que consideram durante as buscas a estrutura de vizinhança $VN^{(R)}$ (realocação de viagem);
2. *DescidaComTroca_PPV* e *DescidaRandomicaComTroca_PPV* que consideram durante as buscas a estrutura de vizinhança $VN^{(T)}$ (troca de viagens).

Destaca-se que os métodos *DescidaRandomicaComRealocacao_PPV* e *DescidaRandomicaComTroca_PPV* foram utilizados com o intuito de introduzir uma certa aleatoriedade ao processo de busca por melhoras e, ao mesmo tempo, torná-lo menos dispendioso em relação ao tempo de processamento. Para tanto, as escolhas dos veículos e das viagens manipulados nos movimentos são feitas de forma aleatória e a solução

Algoritmo 4: Heurística para geração de uma solução inicial para o PPV.

```

1  Seja  $T$  o conjunto de viagens a serem realizadas, com  $T$  ordenado
   crescentemente por horário de início de suas viagens
2  Seja  $F$  o conjunto de veículos da empresa, inicialmente  $F$  possui um único
   veículo sem qualquer viagem
3  enquanto  $T \neq \emptyset$  faça
4       $t \leftarrow$  primeira viagem de  $T$ 
5       $melhorDelta \leftarrow \infty$ , onde  $melhorDelta$  é a melhor variação ocorrida na
   função de avaliação ao se alocar a viagem  $t$ 
6      para cada veículo  $v \in F$  faça
7           $custoAnterior \leftarrow$  custo do veículo  $v$  antes de receber a viagem  $t$ 
8           $custoPosterior \leftarrow$  novo custo do veículo  $v$  caso a viagem  $t$  seja inserida
   em sua jornada de trabalho diária
9          Calcule a variação de custo  $\Delta$  do veículo  $v$  caso ele receba a viagem  $t$ :
           $\Delta \leftarrow custoPosterior - custoAnterior$ 
10         se  $\Delta < melhorDelta$  então
11              $melhorDelta \leftarrow \Delta$ 
12              $melhorVeiculo \leftarrow info(v)$ , isto é, armazene uma informação que
           permita identificar o veículo da frota que irá contribuir menos para o
           valor da função de avaliação ao receber a viagem  $t$ 
13     se  $melhorVeiculo$  não possui viagens então
14         Crie um novo veículo sem viagens e adicione ao conjunto  $F$ . (criação
           dinâmica de veículo, mantendo sempre um veículo sem viagens em  $F$ )
15     Aloque a viagem  $t$  ao veículo da frota  $F$  indicado por  $melhorVeiculo$ 
16      $fo \leftarrow fo + melhorDelta$ , ou seja, atualize o valor da função de avaliação
17     Remova  $t$  de  $T$ 
18 Retorne, como solução, a programação da frota  $F$  da empresa

```

corrente é atualizada sempre que a solução vizinha tenha reduzido o valor da melhor função de custo até então obtida. Além disso, a condição de parada destes procedimentos é determinada pelo número de iterações sem melhoras na solução corrente, conforme expressão (3.2) a seguir:

$$itMax_{PPV} = (total\_viagens \times (total\_veiculos - 1)) \times 0,6 \quad (3.2)$$

onde:

- (a) $itMax_{PPV}$ é o número máximo de iterações sem melhoras na solução corrente;
- (b) $total\_viagens$ é o número de viagens a serem realizadas;
- (c) $total\_veiculos$ é o número de veículos sendo empregados na programação.

Dessa forma, o número máximo de iterações sem melhoras, $itMax_{PPV}$, será igual a 60% do total de possíveis realocações de viagens considerando a programação de veículos atual. Tal percentual foi definido empiricamente de forma a encontrar uma boa relação entre o tempo de execução da busca e a qualidade da solução obtida. Esta estratégia é interessante, uma vez que, relaciona o valor de $itMax_{PPV}$ com o tamanho do problema sendo considerado.

O procedimento proposto neste trabalho para resolver o Problema de Programação de Veículos, cujo pseudo-código é apresentado pelo Algoritmo 5, é baseado na metaheurística Busca Local Iterada.

Uma solução inicial para o PPV é obtida pela heurística construtiva gulosa descrita anteriormente na subseção 3.3.4. Em seguida, com o objetivo de melhorar a qualidade da programação, foram utilizados os procedimentos de busca local *DescidaRandomicaComRealocacao_PPV*, *DescidaRandomicaComTroca_PPV*, *DescidaComRealocacao_PPV* e *DescidaComTroca_PPV*.

As perturbações correspondem a movimentos realizados sobre a solução corrente e que envolvem veículos e viagens aleatoriamente selecionados na programação. Há cinco níveis de perturbação, são eles: *nível 1* - um movimento de realocação de viagem, *nível 2* - um movimento de troca de viagens, *nível 3* - dois movimentos de realocação de viagem, *nível 4* - dois movimentos de troca de viagens e *nível 5* - um movimento de realocação-troca de viagens.

Os procedimentos de busca local utilizados para refinar a solução perturbada fundamentam-se no Método Randômico de Descida, são eles: *DescidaRandomicaComRealocacao_PPV* e *DescidaRandomicaComTroca_PPV*.

Uma nova solução para o PPV somente será aceita se ela for de qualidade melhor ou igual a da solução corrente (linhas 15 e 18). No primeiro caso, o nível de perturbação é reiniciado de forma a intensificar a busca na nova solução obtida. Nas demais situações o nível de perturbação é aumentado em um ou reiniciado (caso já se esteja no nível de perturbação mais alto).

Algoritmo 5: Algoritmo de resolução do PPV.

```

1  Seja  $T$  o tempo de processamento corrente do algoritmo e  $T_{max}$  o tempo
   máximo admissível de processamento
2  Seja  $nivel$  o nível de perturbação corrente
3   $T \leftarrow 0$ 
4   $nivel \leftarrow 1$ 
5  Gere uma solução inicial  $s^v$  a partir da heurística construtiva gulosa descrita
   pelo Algoritmo 4, na página 23
6   $s^v \leftarrow DescidaRandomicaComRealocacao\_PPV(s^v)$ 
7   $s^v \leftarrow DescidaRandomicaComTroca\_PPV(s^v)$ 
8   $s^v \leftarrow DescidaComRealocacao\_PPV(s^v)$ 
9   $s^v \leftarrow DescidaComTroca\_PPV(s^v)$ 
10  $s_*^v \leftarrow s^v$ , onde  $s_*^v$  é a melhor solução encontrada até então
11 enquanto  $T < T_{max}$  faça
12    $s^v \leftarrow perturbacao(s_*^v, nivel)$ 
13    $s^v \leftarrow DescidaRandomicaComRealocacao\_PPV(s^v)$ 
14    $s^v \leftarrow DescidaRandomicaComTroca\_PPV(s^v)$ 
15   se  $f^v(s^v) < f^v(s_*^v)$  então
16      $s_*^v \leftarrow s^v$ , ou seja, atualize a melhor solução até então encontrada
17      $nivel \leftarrow 1$ 
18   se  $f^v(s^v) = f^v(s_*^v)$  então
19      $s_*^v \leftarrow s^v$ , ou seja, atualize a melhor solução até então encontrada
20      $nivel \leftarrow nivel + 1$ 
21   se  $f^v(s^v) > f^v(s_*^v)$  então
22      $nivel \leftarrow nivel + 1$ 
23   se  $nivel > numNiveis$  então
24      $nivel \leftarrow 1$ , onde  $numNiveis$  é o número máximo de níveis de perturbação
25 Retorne a solução  $s_*^v$ 

```

3.4 Descrição do Problema de Programação de Tripulações

Em um Sistema de Transporte Público de Ônibus Urbano, todo deslocamento realizado por um veículo envolve a mão-de-obra de uma tripulação constituída por dois funci-

onários: um motorista e um cobrador. O Problema de Programação de Tripulações (PPT) consiste em criar jornadas diárias de trabalho para os motoristas e cobradores de tal forma a viabilizar a execução de todas as viagens da empresa. Além disso, esta distribuição de trabalho deve ser realizada de maneira a minimizar os custos com mão-de-obra e ao mesmo tempo obedecer à legislação trabalhista e as regras operacionais sob as quais a empresa atua.

Uma limitação para a resolução do PPT é o fato de que não é entre quaisquer viagens de um veículo que poderá haver a troca de tripulação. Esta só ocorrerá nas chamadas oportunidades de troca, ou seja, em locais e em intervalos de tempo que possibilitem a mudança de tripulação e a revisão do veículo (verificação do estado em que a tripulação anterior deixou o ônibus).

Assim, na abordagem sequencial tradicional, antes de efetivamente se gerar a programação de tripulações, os blocos de veículos são quebrados em tarefas. Uma tarefa engloba viagens consecutivas, de um mesmo veículo, entre as quais não há nenhuma oportunidade de troca e é caracterizada pelos seguintes atributos: veículo ao qual pertence, horário de início, horário de término, ponto inicial, ponto final, folga acumulada (isto é, tempo ocioso da tripulação no decorrer da tarefa) e duração.

Esta estratégia de construir tarefas facilita a resolução do PPT, pois mantém as viagens que obrigatoriamente devem ser realizadas por uma mesma tripulação juntas, evitando dessa forma que as soluções do PPV e PPT sejam incompatíveis.

Supondo que para ocorrer a troca de tripulação é necessário um tempo maior ou igual a seis minutos entre as viagens, um exemplo de um bloco de veículo particionado em tarefas é mostrado na Figura 3.3 a seguir. Além dos atributos normais de uma viagem, no exemplo, as colunas “Viagem Morta” e “Tempo de Terminal” apresentam, respectivamente, os tempos em minutos, de deslocamento do veículo fora de operação e de espera no terminal aguardando o horário de início da próxima viagem programada.

Nesta figura observa-se, por exemplo, que a “Tarefa 5” engloba as viagens “101”, “120” e “149” por não haver um tempo suficiente entre elas para que ocorra a troca de tripulação. Esta tarefa iniciará às 15:15 horas, no ponto “3055”, e será finalizada neste mesmo ponto às 20:04 horas. Assim sendo, a “Tarefa 5” possui uma duração de 4 horas e 49 minutos e sua folga acumulada é de quatro minutos.

Vale ressaltar que as tarefas são construídas de tal forma a cobrir não apenas as viagens operacionais dos veículos, mas como também os deslocamentos sem passageiros dos mesmos, conforme pode ser notado nas estruturas das tarefas “Tarefa 1”, “Tarefa 3” e “Tarefa 6” do exemplo. A “tarefa 1” considera o deslocamento do veículo da garagem para o ponto de início de sua primeira viagem, a “tarefa 3” contempla o reposicionamento do veículo do ponto “3050” (local de fim da viagem “60”) para o

ponto “3055” (local de início da próxima viagem, viagem “87”) e a “Tarefa 6” cobre o retorno do veículo à garagem ao fim de sua jornada de trabalho.

	Número da Viagem	Ponto Inicial	Horário de Início (hh:mm)	Ponto Final	Horário de Término (hh:mm)	Viagem Morta	Tempo de Terminal	Linha
Tarefa 1	-	Garagem	04:12	3050	04:30	18	0	-
	1	3050	04:30	3050	05:50	0	30	3050.1
Tarefa 2	34	3050	06:20	3050	07:55	0	5	3050.1
	47	3050	08:00	3050	09:28	0	32	3050.1
Tarefa 3	60	3050	10:00	3050	11:47	0	0	3050.1
	-	3050	11:47	3055	12:01	14	39	-
Tarefa 4	87	3055	12:40	3055	14:22	0	53	3055.1
	101	3055	15:15	3055	17:03	0	0	3055.1
Tarefa 5	120	3055	17:03	3055	18:20	0	4	3055.1
	149	3055	18:24	3055	20:04	0	8	3055.1
Tarefa 6	183	3055	20:12	3055	21:31	0	0	3055.1
	-	3055	21:31	Garagem	21:49	18	0	-

Figura 3.3. Bloco de veículo particionado em tarefas.

Destaca-se que na resolução do PPV a unidade manipulada é a viagem, já no PPT é a tarefa (a menor porção de uma jornada de trabalho de uma tripulação). As jornadas geradas pela resolução do PPT podem ser de dois tipos: pegada simples ou dupla pegada. Uma jornada do tipo pegada simples é aquela em que todos os intervalos entre suas tarefas consecutivas são inferiores a duas horas, ou seja, a tripulação responsável por essa jornada trabalhará um único turno. Aquelas jornadas que possuem um intervalo de tempo, entre duas de suas tarefas consecutivas, igual ou superior a duas horas, são classificadas como do tipo dupla pegada e são realizadas em dois turnos, sendo esta extensa pausa entre tarefas não remunerada pela empresa. No problema sendo abordado, a duração normal de uma jornada seja ela de pegada simples ou de dupla pegada é de 6:40 horas de efetivo trabalhado. No caso de pegada simples, a tripulação tem direito a um intervalo de 20 minutos para descanso e/ou alimentação. Assim, a duração de sua jornada estende-se a 7:00 horas.

Na formação das jornadas a legislação trabalhista e as regras operacionais devem ser obrigatoriamente satisfeitas para que uma solução para o PPT seja viável. As restrições que serão levadas em consideração são as seguintes:

- (a) Uma tripulação não pode executar duas tarefas simultaneamente, isto é, em nenhuma jornada poderá ocorrer coincidências de horário entre quaisquer de suas tarefas;

- (b) Uma tripulação somente poderá realizar a troca de veículos nas seguintes circunstâncias: nas situações em que os veículos envolvidos se encontram em um mesmo terminal no momento da troca, basta que a tripulação tenha um pequeno intervalo ocioso (cinco minutos no mínimo) para que ela abandone um veículo e desloque rapidamente para assumir o outro. Caso contrário, se os veículos envolvidos estão em terminais distintos, a troca somente poderá ocorrer no intervalo entre os turnos de uma jornada do tipo dupla pegada. Esta última condição se faz necessária devido ao fato de a distância entre os diferentes terminais ser grande. Vale ressaltar que um terminal é o local de disposição dos vários pontos de parada dos veículos e onde ocorre o embarque e desembarque de passageiros;
- (c) Uma tripulação pode extrapolar, no máximo, duas horas diárias de trabalho além da duração normal de sua jornada, sendo este tempo excedente considerado hora extra;
- (d) Toda jornada do tipo pegada simples deve ter uma folga mínima de 20 minutos, podendo esta ser fracionada em dois intervalos, tendo cada um, uma duração igual ou superior a 10 minutos;
- (e) O intervalo mínimo entre o final de uma jornada e o início da mesma no dia seguinte deverá ser de 11:00 horas;
- (f) O número de jornadas de trabalho do tipo dupla pegada não pode exceder a 20% do total de tripulações da programação.

Neste trabalho o objetivo é obter uma programação de tripulações que priorize a diminuição dos custos com mão-de-obra. É importante destacar que isto não implica em apenas reduzir ao máximo o número de tripulantes empregados, mas baseando-se nos gastos envolvidos com a contratação e extensão da jornada (hora extra) de um funcionário, obter a melhor distribuição de trabalho em termos financeiros.

Portanto, uma solução para o PPT é um conjunto de jornadas de trabalho, sendo cada uma delas realizada por uma tripulação pertencente ao quadro de funcionários da empresa. O objetivo é minimizar o custo com a remuneração das tripulações, mas ao mesmo tempo cumprir as regras de caráter operacional e jurídico envolvidos.

3.5 Método de Solução do PPT

A metodologia adotada neste trabalho para resolução do PPT é análoga àquela apresentada anteriormente para abordar o PPV, conforme poderá ser observado no decorrer

desta seção. Na subseção 3.5.1 mostra-se como uma solução para o PPT é representada. Em seguida, na subseção 3.5.2, são descritos os tipos de movimentos utilizados para explorar o espaço de busca. Mostra-se na subseção 3.5.3 como se avalia uma solução do problema. O procedimento de construção das tarefas é detalhado na subseção 3.5.4. Os algoritmos para geração das soluções inicial e final para o PPT são descritos, respectivamente, nas subseções 3.5.5 e 3.5.6.

3.5.1 Representação

Uma solução s^t para o PPT consiste em uma lista de jornadas, sendo que a cada jornada está associada uma lista de tarefas a serem executadas por uma mesma tripulação, durante um dia de trabalho, ordenadas crescentemente por seus horários de início. Apesar de ser computacionalmente custoso manter as tarefas ordenadas, esta disposição se faz necessária, pois através dela é possível determinar características relevantes da jornada, tais como: número de trocas de veículos realizadas, tempo de ociosidade, tipo de jornada (pegada simples ou dupla) e existência de sobreposição entre tarefas.

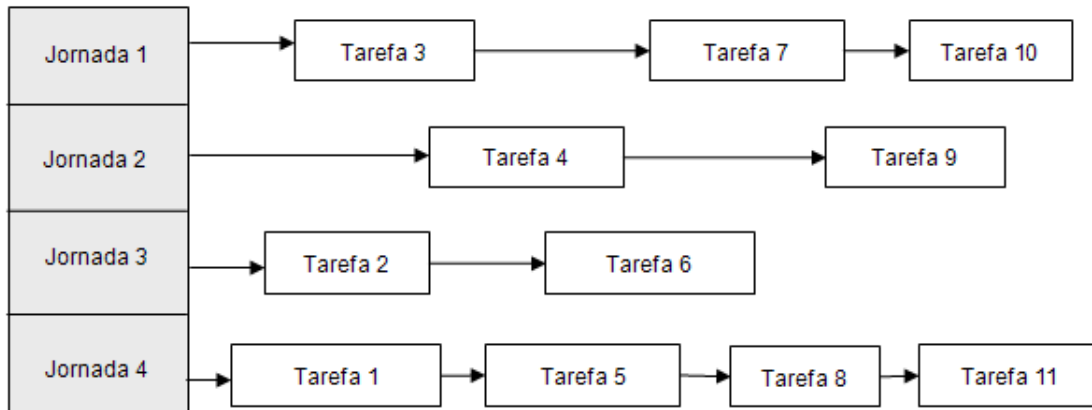


Figura 3.4. Exemplo de uma solução s^t para o PPT.

3.5.2 Estruturas de Vizinhança

Para exploração do espaço de soluções do PPT foram utilizados, assim como no PPV, três tipos de movimentos: realocação de tarefa, troca de tarefas e realocação-troca de tarefas. A realocação de tarefa consiste em transferir uma tarefa pertencente à jornada de trabalho de uma dada tripulação para a jornada de outra tripulação. O movimento de troca de tarefas corresponde a permutar uma tarefa de uma dada jornada com uma tarefa pertencente à outra jornada. Já o movimento de realocação-troca de tarefas envolve três jornadas, j_1 , j_2 e j_3 , a jornada j_1 tem uma de suas tarefas transferida para

a jornada j_2 e a jornada j_2 , por sua vez, cede uma de suas tarefas (desde que não seja a tarefa proveniente de j_1) para a jornada j_3 .

Portanto, foram definidas para o PPT as estruturas de vizinhança $CN^{(R)}$ que se refere ao movimento de realocação de tarefa, $CN^{(T)}$ associada ao movimento de troca de tarefas e $CN^{(RT)}$ relacionada ao movimento de realocação-troca de tarefas.

Os movimentos de realocação, troca e realocação-troca de tarefas estão representados na Figura 3.5.

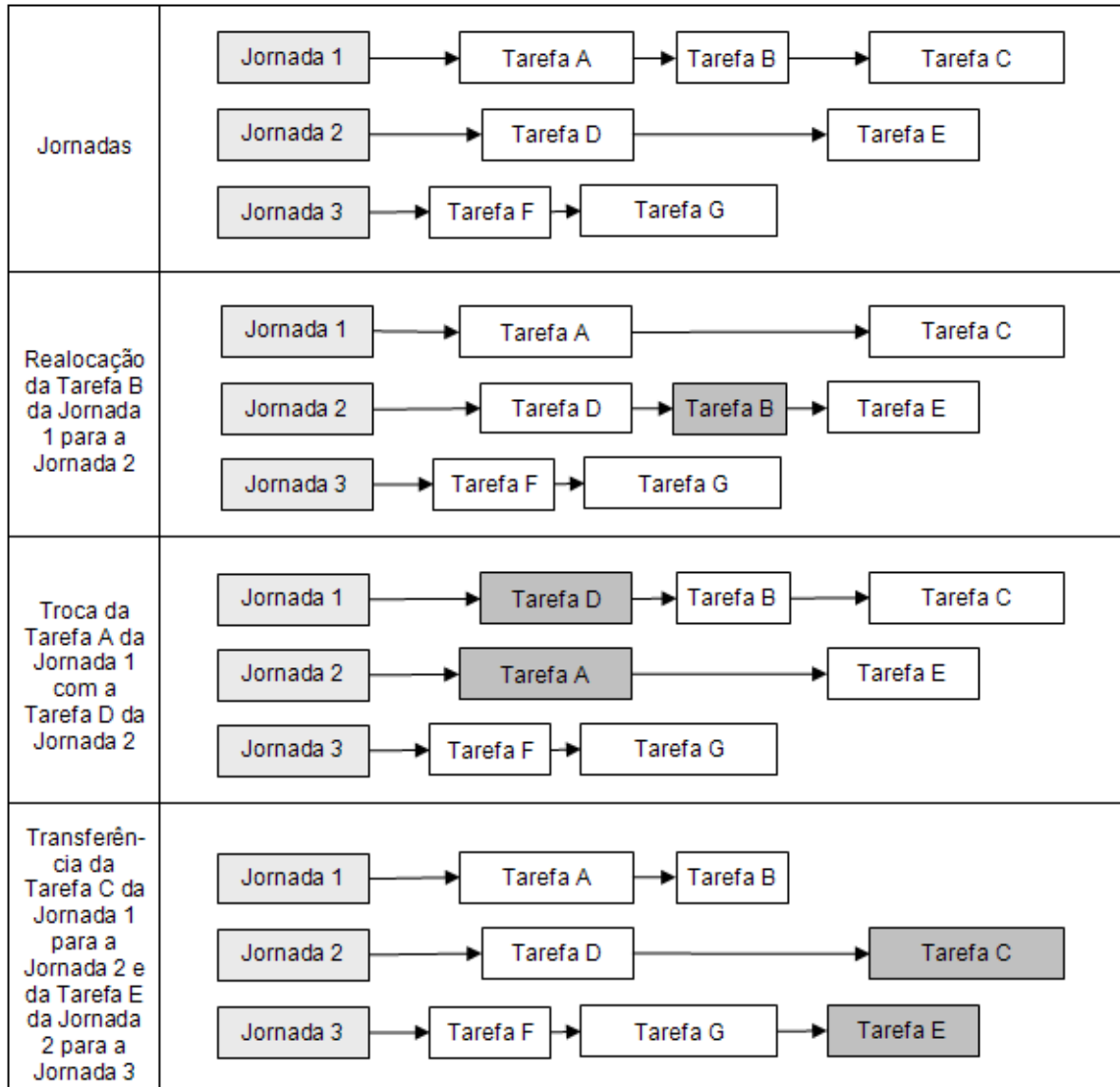


Figura 3.5. Estruturas de vizinhança do PPT.

3.5.3 Função de Avaliação

Uma solução s^t para o Problema de Programação de Tripulações é avaliada penalizando-se os seus atributos indesejáveis. Portanto, o PPT nesta abordagem é tratado como um problema de minimização, cujo custo associado é calculado pela função f^t , conforme apresentado a seguir pela expressão (3.3).

$$\begin{aligned} f^t(s^t) = & \text{totalTripulacoes} \times \text{custoUsoTripulacao} \\ & + \text{tempoHoraExtra} \times \text{custoHoraExtra} \\ & + \text{totalExcessoDP} \times \text{pesoInviabilidade} \\ & + \text{tempoSobreposicao} \times \text{pesoInviabilidade} \end{aligned} \quad (3.3)$$

em que:

- (a) *totalTripulacoes* é o número de tripulações empregadas na programação;
- (b) *custoUsoTripulacao* é o custo associado a cada tripulação sendo utilizada;
- (c) *tempoHoraExtra* é o tempo total de horas extras, em minutos, considerando todas as tripulações da programação;
- (d) *custoHoraExtra* é o custo associado a cada minuto de hora extra realizada por uma tripulação;
- (e) *totalExcessoDP* é a quantidade de duplas pegadas da programação que supera um certo número admissível, previamente definido;
- (f) *tempoSobreposicao* é a soma, dada em minutos, de todos os intervalos de tempo em que há sobreposição de horários entre tarefas de uma mesma tripulação da programação. Destaca-se que, também será considerado sobreposição, o tempo faltante para que sejam completadas as 11:00 horas de intervalo entre o final de uma jornada e o início da mesma no dia seguinte;
- (g) *pesoInviabilidade* é o peso dado a cada inviabilidade presente na solução.

3.5.4 Procedimento de Construção das Tarefas

Para construção das tarefas cada bloco de veículo em uma solução para o PPV é considerado separadamente. Como as viagens em um bloco encontram-se ordenadas crescentemente pelos seus horários de início, o processo começa da primeira viagem do veículo no dia e sistematicamente vai percorrendo toda a sua jornada de operação, viagem por viagem. Neste procedimento, a cada iteração, dadas a viagem i atualmente sendo analisada e a tarefa t em formação, são executados os seguintes passos:

1. Se i é a primeira viagem do veículo, então a tarefa t inicia na garagem, cobrindo assim, o deslocamento fora de operação realizado da garagem para o ponto de início da viagem i ;
2. Se i é a última viagem do veículo, então a tarefa t é finalizada na garagem, após cobrir a viagem i , e não há mais tarefas a serem construídas. Caso contrário, se após a viagem i o veículo realiza a viagem j (que será a viagem a ser analisada na próxima iteração), tem-se que:
 - a) Se entre as viagens i e j não houver uma oportunidade de troca, ambas serão agrupadas na tarefa t e a folga acumulada da tarefa t será acrescida do tempo de ociosidade do veículo entre as viagens i e j . Neste caso, a tarefa t continua em processo de construção;
 - b) Se entre as viagens i e j o veículo retorna à garagem (ou seja, entre estas viagens houver uma dupla pegada), a tarefa t é finalizada na garagem após cobrir a viagem i e uma nova tarefa t' iniciará na garagem. Assim, a nova tarefa em formação será t' e esta cobrirá o deslocamento fora de operação realizado da garagem para o ponto de início da viagem j ;
 - c) Se entre as viagens i e j houver uma oportunidade de troca e, o veículo não realizar dupla pegada entre elas, a tarefa t cobre a viagem i e é finalizada no ponto de início da viagem j . Dessa forma, caso ocorra deslocamento fora de operação entre as viagens i e j , ele será coberto pela tarefa t . Em seguida, inicia-se uma nova tarefa, t' , no ponto de início da viagem j e esta passa a ser a tarefa em construção.

Conforme pode ser observado o procedimento de construção das tarefas contempla todos os deslocamentos realizados pelos veículos e engloba, em uma única tarefa, viagens que devem ser realizadas por uma mesma tripulação.

3.5.5 Procedimento de Geração da Solução Inicial

Antes de dar início ao processo de resolução do PPT, dada a programação de veículos, devem ser definidas as suas tarefas. Para tanto, foi utilizado o procedimento de construção das tarefas descrito anteriormente na subseção 3.5.4.

A programação de tripulações inicial é gerada por uma heurística construtiva gulosa (Algoritmo 6). O algoritmo proposto consiste na alocação, a cada passo, de uma nova tarefa na jornada de trabalho diária de alguma tripulação. Esta combinação (tarefa-jornada) deve ser a que implique no menor custo possível, determinado pela aplicação da função de avaliação apresentada anteriormente pela expressão (3.3).

Destaca-se que, durante a definição da programação, jornadas são criadas dinamicamente conforme demanda por mão-de-obra (linhas 13 e 14, no Algoritmo 6).

Algoritmo 6: Heurística para geração de uma solução inicial para o PPT.

```

1  Seja  $T$  o conjunto de tarefas obtidas pelo procedimento de construção de tarefas
   descrito na subseção 3.5.4, na página 31. Em  $T$ , tarefas de um mesmo veículo se
   encontram dispostas seqüencialmente e ordenadas crescentemente por horário de
   início
2  Seja  $J$  o conjunto de jornadas diárias de trabalho, inicialmente  $J$  possui uma
   única jornada sem qualquer tarefa
3  enquanto  $T \neq \emptyset$  faça
4       $t \leftarrow$  primeira tarefa de  $T$ 
5       $melhorDelta \leftarrow \infty$ , onde  $melhorDelta$  é a melhor variação ocorrida na
       função de avaliação ao se alocar a tarefa  $t$ 
6      para cada jornada  $j \in J$  faça
7           $custoAnterior \leftarrow$  custo da jornada  $j$  antes de receber a tarefa  $t$ 
8           $custoPosterior \leftarrow$  custo da jornada  $j$  caso a tarefa  $t$  passe a pertencer a
               ela
9          Calcule a variação de custo  $\Delta$  da jornada  $j$  após receber a tarefa  $t$ :
               $\Delta \leftarrow custoPosterior - custoAnterior$ 
10         se  $\Delta < melhorDelta$  então
11              $melhorDelta \leftarrow \Delta$ 
12              $melhorJornada \leftarrow info(j)$ , isto é, armazene uma informação que
               permita identificar a jornada que irá contribuir menos para o valor da
               função de avaliação ao receber a tarefa  $t$ 
13     se  $melhorJornada$  não possui tarefas então
14         Crie uma nova jornada sem tarefas e adicione ao conjunto  $J$ . (criação
               dinâmica de jornada, mantendo sempre uma jornada sem tarefas em  $J$ )
15     Aloque a tarefa  $t$  à jornada do conjunto  $J$  indicada por  $melhorJornada$ 
16      $fo \leftarrow fo + melhorDelta$ , ou seja, atualize o valor da função de avaliação
17     Remova  $t$  de  $T$ 
18 Retorne, como solução, o conjunto de jornadas  $J$ 

```

3.5.6 Algoritmo de Resolução

A heurística proposta para resolução do PPT utiliza as mesmas idéias apresentadas anteriormente para tratar o PPV. O seu pseudo-código é apresentado a seguir pelo Algoritmo 7. Trata-se de um procedimento fundamentado na metaheurística Busca Local Iterada em que, primeiramente, uma solução inicial é gerada pela heurística construtiva gulosa descrita na subseção 3.5.5. Com o objetivo de melhorar a qualidade das soluções obtidas, foram utilizados procedimentos de busca local baseados no Método de Descida

e no Método Randômico de Descida, são eles: *DescidaComRealocacao_PPT* e *DescidaRandomicaComRealocacao_PPT* que consideram durante as buscas a estrutura de vizinhança $CN^{(R)}$ (realocação de tarefa), e *DescidaComTroca_PPT* e *DescidaRandomicaComTroca_PPT* que limitam suas pesquisas ao conjunto de soluções dado por $CN^{(T)}$ (troca de tarefas).

Algoritmo 7: Algoritmo de resolução do PPT.

```

1  Seja  $T$  o tempo de processamento corrente do algoritmo e  $Tmax$  o tempo
   máximo admissível de processamento
2  Seja  $nivel$  o nível de perturbação corrente
3   $T \leftarrow 0$ 
4   $nivel \leftarrow 1$ 
5  Gere uma solução inicial  $s^t$  a partir da heurística construtiva gulosa descrita pelo
   Algoritmo 6, na página 33
6   $s^t \leftarrow DescidaRandomicaComRealocacao\_PPT(s^t)$ 
7   $s^t \leftarrow DescidaRandomicaComTroca\_PPT(s^t)$ 
8   $s^t \leftarrow DescidaComRealocacao\_PPT(s^t)$ 
9   $s^t \leftarrow DescidaComTroca\_PPT(s^t)$ 
10  $s_*^t \leftarrow s^t$ , onde  $s_*^t$  é a melhor solução encontrada até então
11 enquanto  $T < Tmax$  faça
12    $s^t \leftarrow perturbacao(s_*^t, nivel)$ 
13    $s^t \leftarrow DescidaRandomicaComRealocacao\_PPT(s^t)$ 
14    $s^t \leftarrow DescidaRandomicaComTroca\_PPT(s^t)$ 
15   se  $f^t(s^t) < f^t(s_*^t)$  então
16      $s_*^t \leftarrow s^t$ , ou seja, atualize a melhor solução até então encontrada
17      $nivel \leftarrow 1$ 
18   se  $f^t(s^t) = f^t(s_*^t)$  então
19      $s_*^t \leftarrow s^t$ , ou seja, atualize a melhor solução até então encontrada
20      $nivel \leftarrow nivel + 1$ 
21   se  $f^t(s^t) > f^t(s_*^t)$  então
22      $nivel \leftarrow nivel + 1$ 
23   se  $nivel > numNiveis$  então
24      $nivel \leftarrow 1$ , onde  $numNiveis$  é o número máximo de níveis de perturbação
25 Retorne a solução  $s_*^t$ 

```

Destaca-se que nos métodos *DescidaRandomicaComRealocacao_PPV* e *DescidaRandomicaComTroca_PPV* a condição de parada é determinada pelo número de iterações sem melhoras na solução corrente, conforme expressão (3.4) a seguir:

$$itMax_{PPT} = (total\_tarefas \times (total\_jornadas - 1)) \times 0,6 \quad (3.4)$$

onde:

- (a) $itMax_{PPT}$ é o número máximo de iterações sem melhoras na solução corrente;
- (b) $total\_tarefas$ é o número de tarefas a serem realizadas;
- (c) $total\_jornadas$ é o número de jornadas da programação.

Dessa forma, o número máximo de iterações sem melhoras, $itMax_{PPT}$, corresponderá a 60% do total de possíveis realocações de tarefas, considerando a programação de tripulações atual.

As perturbações, neste contexto, correspondem a movimentos de realocação, troca e realocação-troca envolvendo tripulações e tarefas aleatoriamente selecionadas na programação. Há cinco níveis de perturbação, são eles: *nível 1* - um movimento de realocação de tarefa, *nível 2* - um movimento de troca de tarefas, *nível 3* - dois movimentos de realocação de tarefas, *nível 4* - dois movimentos de troca de tarefas e *nível 5* - um movimento de realocação-troca de tarefas.

Uma nova solução para o PPT somente será aceita se ela possuir um custo menor ou igual ao custo da solução corrente. No primeiro caso, o nível de perturbação é reiniciado de forma a intensificar a busca na nova solução obtida. Nas demais situações o nível de perturbação é aumentado em um ou reiniciado (caso já se esteja no nível de perturbação mais alto).

3.6 Programação Independente de Veículos e Tripulações

Verifica-se que na abordagem sequencial tradicional a solução do PPV impõe restrições adicionais à resolução do PPT. Este fato é claramente evidenciado na formação das tarefas, pois estas contemplam os deslocamentos fora de operação dos veículos e aglomeram viagens que obrigatoriamente deverão ser realizadas pela mesma tripulação, devido à impossibilidade dos veículos de efetivarem a troca entre algumas de suas viagens.

Na abordagem independente PPV e PPT são resolvidos separadamente, em qualquer ordem e de forma a não se relacionarem. Para tanto, a estratégia de resolução adotada é igual à apresentada anteriormente, na abordagem sequencial, exceto pelo fato de que cada tarefa no PPT passa a corresponder a exatamente uma viagem. Além disso, supõe-se que todas as tarefas são realizadas por um mesmo veículo para que não haja a necessidade de contemplar trocas de veículos na programação de tripulações. Assim, são eliminados todos os vínculos entre as programações de veículos e tripulações e ambas passam a ser definidas a partir do conjunto de viagens.

Difícilmente a programação independente de veículos e tripulações gera soluções para PPV e PPT que sejam compatíveis e factíveis na prática. A principal finalidade desta metodologia é a determinação de uma melhor programação para as tripulações desconsiderando, para tanto, a programação dos veículos. Desta forma, uma comparação entre as soluções obtidas nessa abordagem e na seqüencial tradicional permite identificar os possíveis benefícios de se resolver PPV e PPT de forma integrada.

Segundo Freling et al. [2003], se ao comparar os custos das soluções obtidas nas abordagens seqüencial tradicional e independente houver uma grande diferença entre estes valores (o custo da solução independente é muito menor) espera-se que a qualidade da programação de tripulações seja melhorada significativamente quando esta for definida de forma integrada ao invés de seqüencial. Por outro lado, se os custos das soluções estão próximos, presume-se que a abordagem integrada não será promissora. Não havendo, portanto, a necessidade de aplicá-la.

Capítulo 4

Programação Integrada de Veículos e Tripulações

Apesar dos dois problemas, PPV e PPT, normalmente serem abordados separadamente, há uma relação de dependência entre eles. Os custos das programações podem ser inclusive antagônicos, ou seja, uma característica favorável a uma solução para o PPV nem sempre reflete de forma satisfatória no PPT e vice-versa. Assim, conclui-se que na abordagem seqüencial tradicional é priorizada a resolução do PPV uma vez que este é tratado em uma etapa anterior ao PPT.

Como no contexto do Sistema de Transporte Público de Ônibus Urbano comumente os gastos das empresas com tripulantes superam os gastos com veículos, presume-se que a metodologia de resolução integrada dos problemas poderá produzir resultados melhores do que aqueles gerados pela abordagem seqüencial. Além disso, destaca-se que o PPT é um problema mais restrito do que o PPV, por envolver limitações tanto de caráter operacional quanto trabalhista, o que justifica ainda mais a relevância da integração.

Portanto, a finalidade desse capítulo é apresentar estratégias para resolução do Problema de Programação Integrada de Veículos e Tripulações (PPVT) de maneira que, posteriormente, possa-se verificar a efetividade dessa abordagem comparando-a com a seqüencial tradicional.

Esse capítulo está organizado conforme descrito a seguir. Na seção 4.1 o Problema de Programação Integrada de Veículos e Tripulações é detalhado. Na seção 4.2 é apresentado um método iterativo em que as buscas por melhoras são alternadamente realizadas nos espaços de soluções do PPV e PPT. Por fim, na seção 4.3, um procedimento fundamentado na definição de uma nova estrutura (a viagem expandida) é proposto para resolução do PPVT.

4.1 Descrição do Problema de Programação Integrada de Veículos e Tripulações

O Problema de Programação Integrada de Veículos e Tripulações (PPVT) consiste em definir simultaneamente, as rotinas de operação dos veículos e as jornadas de trabalho das tripulações, de tal forma a minimizar os custos envolvidos e ao mesmo tempo viabilizar a execução de todas as viagens sob responsabilidade da empresa.

Destaca-se que os mesmos conceitos considerados para definir PPV e PPT, separadamente, são válidos no PPVT. Ou seja, “bloco de veículo”, “veículo com dupla pegada”, “viagem morta”, “tarefa”, “jornada com pegada simples”, “jornada com dupla pegada”, dentre outros, são termos que continuarão sendo usados na abordagem integrada e sem alteração de seus significados.

Nesse trabalho, na resolução do PPVT foram consideradas conjuntamente as restrições do PPV e PPT. Estas se encontram exatamente como foram descritas anteriormente no capítulo 3 e são reapresentadas a seguir.

Restrições para a programação de veículos:

- (a) Todas as viagens de responsabilidade de uma empresa, sem exceções, devem ser realizadas por algum veículo da frota;
- (b) Antes de se iniciar e após finalizar uma viagem, um veículo deve obrigatoriamente cumprir um tempo mínimo no terminal, que corresponde ao tempo necessário para o embarque e desembarque dos passageiros;
- (c) Um mesmo veículo somente poderá realizar duas viagens, i e j , consecutivamente, se o intervalo de tempo existente entre elas for suficiente para que o veículo desloque do ponto final da viagem i para o ponto inicial da viagem j , respeite os tempos destinados para embarque e desembarque de passageiros e inicie a viagem j sem atrasos. Caso contrário, ocorrerá uma sobreposição de horários entre as viagens;
- (d) Um veículo não deve ficar mais de duas horas em um terminal esperando para executar a sua próxima viagem. Se o tempo de espera exceder esse limite, o veículo deverá se deslocar para a garagem. Diz-se, nesse último caso, que o veículo está realizando uma dupla pegada;
- (e) O número de duplas pegadas da programação não pode exceder 60% do tamanho da frota;

- (f) Todo veículo deve iniciar e terminar a sua jornada de trabalho na garagem;
- (g) Todo veículo deve permanecer, no mínimo, trinta minutos consecutivos na garagem, por dia, para que ocorra a sua manutenção e limpeza. Este tempo pode ser cumprido quando o veículo pára na garagem durante uma dupla pegada ou mesmo depois de completada toda a sua jornada operacional diária.

Restrições para a programação de tripulações:

- (a) Uma tripulação não pode executar duas tarefas simultaneamente, isto é, em nenhuma jornada poderá ocorrer coincidências de horário entre quaisquer de suas tarefas;
- (b) Uma tripulação somente poderá realizar a troca de veículos nas seguintes circunstâncias: nas situações em que os veículos envolvidos se encontram em um mesmo terminal no momento da troca, basta que a tripulação tenha um pequeno intervalo ocioso (cinco minutos no mínimo) para que ela abandone um veículo e desloque rapidamente para assumir o outro. Caso contrário, se os veículos envolvidos estão em terminais distintos, a troca somente poderá ocorrer no intervalo entre os turnos de uma jornada do tipo dupla pegada. Esta última condição se faz necessária devido ao fato de a distância entre os diferentes terminais ser grande. Vale ressaltar que um terminal é o local de disposição dos vários pontos de parada dos veículos e onde ocorre o embarque e desembarque de passageiros;
- (c) Uma tripulação pode extrapolar, no máximo, duas horas diárias de trabalho além da duração normal de sua jornada, sendo este tempo excedente considerado hora extra;
- (d) Toda jornada do tipo pegada simples deve ter uma folga mínima de 20 minutos, podendo esta ser fracionada em dois intervalos, tendo cada um, uma duração igual ou superior a 10 minutos;
- (e) O intervalo mínimo entre o final de uma jornada e o início da mesma no dia seguinte deverá ser de 11:00 horas;
- (f) O número de jornadas de trabalho do tipo dupla pegada não pode exceder a 20% do total de tripulações da programação.

Atenta-se ao fato de que, além da preocupação em manter as programações de veículos e de tripulações separadamente viáveis, é necessário que elas sejam mutuamente compatíveis para que a resolução do PPVT torna-se aplicável na prática.

O objetivo ao resolver o PPVT é obter programações em que o número de veículos e de tripulações seja minimizado. Além disso, devem-se reduzir os deslocamentos fora de operação realizados pelos veículos e as horas extras das tripulações.

4.2 Método de Solução Iterativa do PPV e PPT

Como primeiro passo para o estudo do PPVT, nessa seção será apresentada uma estratégia iterativa de resolução em que, ora a busca por melhoras se concentra especificamente no PPV e ora no PPT. Porém, para permitir a avaliação das programações dos veículos e das tripulações conjuntamente, durante o processo a compatibilidade entre elas é mantida.

Na subseção 4.2.1 é descrito como as soluções para PPV e PPT são representadas. A função usada para avaliar simultaneamente as programações de veículos e de tripulações é mostrada na subseção 4.2.2. O algoritmo de resolução do PPV e PPT iterativamente é apresentado na subseção 4.2.3.

4.2.1 Representação

Assim como na abordagem sequencial, nesta metodologia, a cada jornada de trabalho está associada uma lista de tarefas.

Conforme mencionado anteriormente, uma tarefa é a associação de viagens de um mesmo veículo entre as quais não há nenhuma oportunidade de troca e que devem ser obrigatoriamente realizadas por uma mesma tripulação. Portanto, conclui-se que a formação das tarefas é totalmente dependente de uma solução para o Problema de Programação de Veículos. Na Figura 4.1 é mostrado o particionamento de um bloco de veículo em três tarefas, supondo que para ocorrer a troca de tripulação é necessário um tempo maior ou igual a seis minutos. Nota-se que remoções e/ou inserções de viagens na jornada de trabalho do veículo provavelmente irão alterar as tarefas obtidas a partir dele. Por exemplo, se a viagem “149” for removida do bloco de veículo, a tarefa “Tarefa 2” será suprimida e surgirá uma nova tarefa que consistirá apenas das viagens “101” e “120”. Por outro lado, se uma viagem com os seguintes atributos (número da viagem: “90”, Ponto inicial: “3055”, Horário de Início: 14:28, Ponto Final: “3055”, Horário de Término: 15:09, Linha: “3055.1”) for inserida neste bloco de veículo, uma nova tarefa será criada entre as tarefas “Tarefa 1” e “Tarefa 2” e compreenderá apenas a viagem “90”.

Dessa forma, com o intuito de preservar a compatibilidade entre as programações de veículos e de tripulações sendo obtidas simultaneamente, foi associado a cada bloco de veículo, além do seu conjunto de viagens, uma lista com as tarefas obtidas a partir dele.

	Número da Viagem	Ponto Inicial	Horário de Início (hh:mm)	Ponto Final	Horário de Término (hh:mm)	Viagem Morta	Tempo de Terminal	Linha
Tarefa 1	-	Garagem	12:22	3055	12:40	18	0	-
	87	3055	12:40	3055	14:22	0	53	3055.1
Tarefa 2	101	3055	15:15	3055	17:03	0	0	3055.1
	120	3055	17:03	3055	18:20	0	4	3055.1
	149	3055	18:24	3055	20:04	0	8	3055.1
Tarefa 3	183	3055	20:12	3055	21:31	0	0	3055.1
	-	3055	21:31	Garagem	21:49	18	0	-

Figura 4.1. Bloco de veículo particionado em tarefas.

Na Figura 4.2 está representado o bloco de veículo dado na Figura 4.1. Com esta nova representação as tarefas que surgem ou são suprimidas após modificações (remoções e/ou inserções de viagens) no bloco de um veículo serão determinadas da seguinte forma: a lista de viagens do veículo é percorrida e suas tarefas são reconstruídas. Em seguida, compara-se o seu conjunto de tarefas anterior com o novo conjunto encontrado, identificando assim, as alterações ocorridas.



Figura 4.2. Representação de um bloco de veículo.

4.2.2 Função de Avaliação

Durante a resolução iterativa as soluções do PPV (s^v) e PPT (s^t) são consideradas conjuntamente. Portanto, para avaliar as programações obtidas foi definida a função f^{vt} dada a seguir pela expressão (4.1).

$$f^{vt}(s^v, s^t) = f^v(s^v) + f^t(s^t) \quad (4.1)$$

Nessa expressão, os valores retornados pelas funções f^v e f^t estão sendo somados e estes são calculados exatamente como descrito anteriormente, na abordagem seqüencial, pelas expressões (3.1) e (3.3).

4.2.3 Algoritmo de Resolução

A seguir no Algoritmo 8 é apresentado o pseudo-código da heurística desenvolvida para resolver iterativamente PPV e PPT. Destaca-se que este algoritmo também é fundamentado na metaheurística Busca Local Iterada. Além disso, as estruturas de vizinhança e os procedimentos de busca local utilizados, são os mesmos que foram descritos anteriormente para tratar PPV e PPT separadamente.

As soluções iniciais para PPV e PPT são geradas pelos mesmos procedimentos descritos na abordagem sequencial (subseções 3.3.4 e 3.5.5, nas páginas 22 e 32). Em seguida, antes de se iniciar as iterações do algoritmo, procura-se melhorar estas soluções a partir de buscas locais realizadas considerando primeiramente as estruturas de vizinhança do PPT, e depois as do PPV (passos de 9 a 12 no Algoritmo 8). Salienta-se que nem sempre se alcança um aprimoramento das soluções após esta etapa, pois como as buscas são realizadas independentemente (procuras por melhoras são realizadas ora no PPT, ora no PPV), pode ocorrer de melhoras na programação de veículos ocasionarem pioras na programação de tripulações e, ao final, ocorrer um aumento na função de custo considerando ambas as programações.

Foram definidos quatro níveis de perturbação, consecutivamente tem-se: *nível 1* - uma realocação de tarefa, *nível 2* - uma realocação de viagem, *nível 3* - uma troca de tarefas e *nível 4* - uma troca de viagens.

É importante ressaltar que sempre que a programação de veículos for modificada é necessário compatibilizar a programação de tripulações com esta nova solução (função $compatibilizar(s^v, s_*^t)$ no Algoritmo 8 e cujo pseudo-código é dado pelo Algoritmo 9), isto é, atualizações serão realizadas de duas maneiras:

1. Remoção na solução do PPT de tarefas que deixaram de existir, devido mudanças no PPV;
2. Inserção na solução do PPT de tarefas que surgiram, devido mudanças no PPV. Neste caso, esta inserção é realizada de forma “gulosa”, isto é, cada tarefa é alocada na jornada da tripulação que ao recebê-la irá contribuir menos com o custo da programação de tripulações corrente.

Assim sendo, sempre que a perturbação for realizada sobre a programação de veículos, executam-se buscas locais nas soluções do PPV e do PPT, pois ambas serão modificadas. Caso contrário, restringem-se as buscas à programação de tripulações.

Uma nova solução (programações de veículos e tripulações) somente será aceita se ela possuir um custo menor ou igual ao custo da solução corrente. No primeiro caso, o nível de perturbação é reiniciado de forma a intensificar a busca na nova solução

Algoritmo 8: Algoritmo de resolução iterativa do PPV e PPT.

```

1  Seja  $T$  o tempo de processamento corrente do algoritmo e  $T_{max}$  o tempo
   máximo admissível de processamento
2  Seja  $nivel$  o nível de perturbação corrente
3   $T \leftarrow 0$ 
4   $nivel \leftarrow 1$ 
5  Gere uma solução inicial  $s^v$ , para o PPV, a partir da heurística construtiva
   gulosa descrita pelo Algoritmo 4, na página 23
6  Dado  $s^v$  gere uma solução  $s^t$  para o PPT. Esta solução inicial é obtida a partir
   do método guloso apresentado pelo Algoritmo 6, na página 33
7   $s_*^v \leftarrow s^v$ , onde  $s_*^v$  é a melhor solução encontrada até então para o PPV
8   $s_*^t \leftarrow s^v$ , onde  $s_*^t$  é a melhor solução encontrada até então para o PPT
9   $s^t \leftarrow DescidaComRealocacao\_PPT(s^t)$ 
10  $s^t \leftarrow DescidaComTroca\_PPT(s^t)$ 
11  $s^v \leftarrow DescidaComRealocacao\_PPV(s^v)$ 
12  $s^v \leftarrow DescidaComTroca\_PPV(s^v)$ 
13  $s^t \leftarrow compatibilizar(s^v, s^t)$ , isto é, atualiza a solução para o PPT a partir das
   modificações realizadas no PPV
14 enquanto  $T < T_{max}$  faça
15   se  $f^{vt}(s^v, s^t) \leq f^{vt}(s_*^v, s_*^t)$  então
16      $s_*^v \leftarrow s^v$ 
17      $s_*^t \leftarrow s^t$ 
18     se  $f^{vt}(s^v, s^t) < f^{vt}(s_*^v, s_*^t)$  então
19        $nivel \leftarrow 1$ 
20     senão
21        $nivel \leftarrow nivel + 1$ 
22   senão
23      $nivel \leftarrow nivel + 1$ 
24   se  $nivel > numNiveis$  então
25      $nivel \leftarrow 1$ , onde  $numNiveis$  é o número máximo de níveis de perturbação
26    $s^v \leftarrow perturbacao(s_*^v, nivel)$  ou  $s^t \leftarrow perturbacao(s_*^t, nivel)$ , conforme o nível
   de perturbação modifique PPV ou PPT
27   se a perturbação realizada envolve apenas realocação ou troca de tarefas no
   PPT então
28      $s^t \leftarrow DescidaRandomicaComRealocacao\_PPT(s^t)$ 
29      $s^t \leftarrow DescidaRandomicaComTroca\_PPT(s^t)$ 
30   senão
31      $s^v \leftarrow DescidaRandomicaComRealocacao\_PPV(s^v)$ 
32      $s^v \leftarrow DescidaRandomicaComTroca\_PPV(s^v)$ 
33      $s^t \leftarrow compatibilizar(s^v, s_*^t)$ , isto é, atualiza a solução para o PPT a
   partir das modificações realizadas no PPV
34      $s^t \leftarrow DescidaRandomicaComRealocacao\_PPT(s^t)$ 
35      $s^t \leftarrow DescidaRandomicaComTroca\_PPT(s^t)$ 
36 Retorne as soluções  $s_*^v$  e  $s_*^t$ 

```

Algoritmo 9: Função compatibilizar(s_2^v, s_1^t).

```

1 Entrada: Uma solução  $s_2^v$  para o PPV obtida a partir de modificações em  $s_1^v$ 
    Uma solução  $s_1^t$  para o PPT que foi definida a partir da solução  $s_1^v$ 
    para o PPV
2 Saída: Uma solução  $s_2^t$  para o PPT compatível com a solução  $s_2^v$  para o PPV
3  $s_2^t \leftarrow s_1^t$ 
4 Seja tarefasCriadas uma lista com todas as tarefas que surgiram a partir das
    mudanças no PPV. Inicialmente esta lista está vazia
5 Seja  $F$  a frota de veículos
6 para cada veículo  $v \in F$  faça
7     se  $v$  foi modificado na transição de  $s_1^v$  para  $s_2^v$  então
8         Percorra a lista de viagens de  $v$  reconstruindo as suas tarefas e atualize
            as listas tarefasSuprimidas e tarefasCriadas. Sendo tarefasSuprimidas o
            conjunto de tarefas que deixaram de existir em  $v$  devido às mudanças
9         para cada tarefa  $t \in \text{tarefasSuprimidas}$  faça
10              $j \leftarrow \text{retornarJornada\_Tarefa}(s_2^t, t)$ , isto é, encontre a jornada  $j$  da
                solução  $s_2^t$  que executa a tarefa  $t$ 
11             Remova a tarefa  $t$  da jornada  $j$  em  $s_2^t$ 
12 para cada tarefa  $t \in \text{tarefasCriadas}$  faça
13      $\text{insercao\_Gulosa}(s_2^t, t)$ , isto é, alocar a tarefa  $t$  à jornada do tripulante que
        ao recebê-la irá contribuir menos com o custo da programação
14 Retorne a solução  $s_2^t$ 

```

obtida. Nas demais situações o nível de perturbação é aumentado em um ou reiniciado (caso já se esteja no nível de perturbação mais alto).

4.3 Método de Solução do PPVT

Nesta seção é proposta uma metodologia de resolução do PPVT com uma idéia de integração mais forte do que a apresentada anteriormente. Em tal estratégia não há mais a formação explícita de tarefas para definição de uma solução para o PPT, porém, é sugerida a criação de uma nova estrutura denominada viagem expandida e, tanto a programação de veículos quanto a programação de tripulações, passam a ser determinadas a partir dela.

Essa seção está organizada como segue. Uma descrição de como o problema é representado é dada na subseção 4.3.1. Na subseção 4.3.2 são apresentadas as estruturas de vizinhança desenvolvidas para explorar o espaço de busca. Descreve-se na subseção 4.3.4 o procedimento de geração da solução inicial. Por fim, na subseção 4.3.5, é apresentado o algoritmo proposto para resolver o problema.

4.3.1 Representação

O elemento básico manipulado nesta estratégia é a viagem expandida. Esta corresponde a uma viagem convencional com propriedades adicionais que definem como ela deve ser considerada no PPT, dada a sua disposição no PPV. Assim, além dos atributos normais de uma viagem (número da viagem, horário de início, ponto inicial, horário de término, ponto final, linha, tempo de embarque e tempo de desembarque) a viagem expandida agrega as seguintes informações: identificador do veículo, horário de início expandido, ponto inicial expandido, horário de término expandido, ponto final expandido, dependência à esquerda e dependência à direita.

Os atributos exclusivos das viagens expandidas consideram os deslocamentos fora de operação dos veículos e mantêm um vínculo entre as viagens que obrigatoriamente deverão ser realizadas pela mesma tripulação (devido à impossibilidade dos veículos de efetivarem a troca entre elas). As regras gerais para definição destes atributos são descritas a seguir:

1. Não é entre quaisquer viagens que uma tripulação poderá realizar a troca de veículos, pois esta operação requer um tempo. Dessa forma, cada viagem expandida deve indicar o veículo que a executa (identificador do veículo);
2. A tripulação responsável pela viagem realizada por um veículo logo após a sua saída da garagem, também deverá conduzi-lo em seu deslocamento fora de operação da garagem para o ponto de início da viagem. Assim, o horário de início expandido, da viagem, será igual ao seu horário de início decrescido do tempo necessário para posicionamento do veículo e, o seu ponto inicial expandido, será a garagem;
3. A tripulação responsável pela viagem realizada por um veículo imediatamente antes de seu retorno à garagem, também deverá conduzi-lo em seu deslocamento fora de operação do ponto de fim da viagem para a garagem. Portanto, o horário de término expandido, da viagem, será igual ao seu horário de término acrescido do tempo necessário para retornar com o veículo para a garagem e, o seu ponto final expandido, será a garagem;
4. Sempre que houver troca de pontos entre duas viagens consecutivas, de um mesmo veículo, a tripulação que executar a primeira viagem deverá posicionar o veículo no ponto de início da segunda. Logo, o horário de término expandido, da primeira viagem, será igual ao seu horário de término acrescido do tempo de deslocamento para reposicionamento do veículo e, o seu ponto final expandido, corresponderá ao ponto inicial da segunda viagem;

5. Sempre que entre duas viagens consecutivas de um veículo não houver tempo suficiente para a troca de tripulação, a dependência à direita da primeira viagem será a segunda e a dependência à esquerda da segunda viagem será a primeira. Isto é, as dependências à esquerda e à direita de uma viagem expandida determinam, respectivamente, quais viagens deverão imediatamente precedê-la e sucedê-la em uma mesma jornada no PPT.

Destaca-se que, a princípio, para uma viagem expandida tem-se que os atributos horário de início expandido, ponto inicial expandido, horário de término expandido e ponto final expandido assumem os mesmos valores, respectivamente, dos seus atributos horário de início, ponto inicial, horário de término e ponto final e, é considerado que não há dependências à esquerda e à direita. Assim, será a aplicação das regras acima que modificará estes valores conforme a disposição da viagem expandida no bloco do veículo que a executa.

A seguir, na Figura 4.3, é mostrado como seria o particionamento, em tarefas, do primeiro bloco de veículo de uma solução para o PPV. Em seguida, na Figura 4.4, são apresentadas as viagens expandidas definidas a partir deste mesmo bloco de veículo. Considera-se, nestes exemplos, que para ocorrer a troca de tripulação é necessário um tempo maior ou igual a seis minutos e que os tempos destinados para embarque e desembarque de passageiros são nulos para todas as viagens.

Bloco de Veículo – 1								
	Número da Viagem	Ponto Inicial	Horário de Início (hh:mm)	Ponto Final	Horário de Término (hh:mm)	Viagem Morta	Tempo de Terminal	Linha
Tarefa 1	-	Garagem	12:22	3055	12:40	18	0	-
	87	3055	12:40	3055	14:22	0	53	3055.1
Tarefa 2	101	3055	15:15	3055	17:03	0	0	3055.1
	120	3055	17:03	3055	18:20	0	4	3055.1
	149	3055	18:24	3055	20:04	0	8	3055.1
	183	3055	20:12	3055	21:31	0	0	3055.1
Tarefa 3	-	3055	21:31	2200	21:36	5	4	-
	210	2200	21:40	2200	22:25	0	0	2200.1
	-	2200	22:25	Garagem	22:45	20	0	-

Figura 4.3. Bloco de Veículo - 1 e o seu particionamento em tarefas.

Observa-se que a “Viagem expandida 87” deverá cobrir a viagem morta realizada pelo veículo ao sair da garagem (regra 2). Logo, o seu ponto inicial expandido é a garagem e o seu horário de início expandido antecede em 18 minutos o seu horário de início. Destaca-se que esta viagem não possui dependências à esquerda e à direita e ela

Viagem expandida 87	
Ponto inicial: 3055	Ponto inicial expandido: Garagem
Horário de início: 12:40 horas	Horário de início expandido: 12:22 horas
Ponto final: 3055	Ponto final expandido: 3055
Horário de término: 14:22 horas	Horário de término expandido: 14:22 horas
Linha: 3055.1	Dependência à esquerda: não há
Tempo de embarque: 0	Dependência à direita: não há
Tempo de desembarque: 0	Identificador do veículo: 1
Viagem expandida 101	
Ponto inicial: 3055	Ponto inicial expandido: 3055
Horário de início: 15:15 horas	Horário de início expandido: 15:15 horas
Ponto final: 3055	Ponto final expandido: 3055
Horário de término: 17:03 horas	Horário de término expandido: 17:03 horas
Linha: 3055.1	Dependência à esquerda: não há
Tempo de embarque: 0	Dependência à direita: Viagem 120
Tempo de desembarque: 0	Identificador do veículo: 1
Viagem expandida 120	
Ponto inicial: 3055	Ponto inicial expandido: 3055
Horário de início: 17:03 horas	Horário de início expandido: 17:03 horas
Ponto final: 3055	Ponto final expandido: 3055
Horário de término: 18:20 horas	Horário de término expandido: 18:20 horas
Linha: 3055.1	Dependência à esquerda: Viagem 101
Tempo de embarque: 0	Dependência à direita: Viagem 149
Tempo de desembarque: 0	Identificador do veículo: 1
Viagem expandida 149	
Ponto inicial: 3055	Ponto inicial expandido: 3055
Horário de início: 18:24 horas	Horário de início expandido: 18:24 horas
Ponto final: 3055	Ponto final expandido: 3055
Horário de término: 20:04 horas	Horário de término expandido: 20:04 horas
Linha: 3055.1	Dependência à esquerda: Viagem 120
Tempo de embarque: 0	Dependência à direita: não há
Tempo de desembarque: 0	Identificador do veículo: 1
Viagem expandida 183	
Ponto inicial: 3055	Ponto inicial expandido: 3055
Horário de início: 20:12 horas	Horário de início expandido: 20:12 horas
Ponto final: 3055	Ponto final expandido: 2200
Horário de término: 21:31 horas	Horário de término expandido: 21:36 horas
Linha: 3055.1	Dependência à esquerda: não há
Tempo de embarque: 0	Dependência à direita: Viagem 210
Tempo de desembarque: 0	Identificador do veículo: 1
Viagem expandida 210	
Ponto inicial: 2200	Ponto inicial expandido: 2200
Horário de início: 21:40 horas	Horário de início expandido: 21:40 horas
Ponto final: 2200	Ponto final expandido: Garagem
Horário de término: 22:25 horas	Horário de término expandido: 22:45 horas
Linha: 2200.1	Dependência à esquerda: Viagem 183
Tempo de embarque: 0	Dependência à direita: não há
Tempo de desembarque: 0	Identificador do veículo: 1

Figura 4.4. Viagens expandidas obtidas do Bloco de Veículo - 1 da Figura 4.3.

é realizada pelo bloco de veículo com identificador igual a 1, exatamente como descrito na Figura 4.4.

Entre o horário de término da viagem “101” e o horário de início da viagem “120” não há tempo ocioso do veículo, logo não será possível ocorrer a troca de tripulação entre elas e a dependência à direita, da viagem “101”, será a viagem “120” e a dependência

à esquerda, da viagem “120”, será a viagem “101” (regra 5). O mesmo ocorrerá entre as viagens “120” e “149”, pois o intervalo ocioso do veículo entre elas é menor que o tempo mínimo necessário para a troca da tripulação. Dessa forma, percebe-se que será criado um vínculo entre as viagens “101”, “120” e “149” que, obrigatoriamente, deverão ser realizadas pela mesma tripulação (definição implícita da tarefa “Tarefa 2” da Figura 4.3).

No bloco de veículo sendo considerado, as viagens “183” e “210” são realizadas consecutivamente. Porém, como o ponto final da primeira viagem (ponto “3055”) é diferente do ponto inicial da segunda (ponto “2200”), existe um deslocamento fora de operação do veículo entre elas (viagem morta de 5 minutos). Assim, o horário de término expandido da viagem “183” será o seu horário de término normal incrementado do tempo de viagem morta do veículo entre os pontos “3055” e “2200” e, o seu ponto final expandido será o ponto “2200” (regra 4). Como o tempo do veículo parado no terminal, entre as viagens “183” e “210”, também é insuficiente para a troca de tripulação, estabelece-se uma relação de dependência entre elas (definição implícita da tarefa “Tarefa 3” da Figura 4.3). Além disso, uma vez que após a viagem “210” o veículo retornará para a garagem, o horário de término expandido da viagem “210” corresponderá ao seu horário de término aumentado do tempo de deslocamento do veículo, do ponto “2200” para a garagem, que será de 20 minutos (regra 3). Dessa forma, o ponto final expandido desta viagem será a garagem.

Pelo exposto, nota-se que os novos atributos das viagens permitem contemplar as mesmas condições asseguradas ao se formar tarefas na resolução do PPT. Portanto, nesta abordagem, tanto os blocos dos veículos quanto as jornadas de trabalho das tripulações serão definidos a partir das viagens expandidas.

A seguir, na Figura 4.5, está ilustrada uma solução para o PPVT com definição das viagens expandidas. Esta solução é composta pela programação dos veículos (s^{vx}) e pela programação das tripulações (s^{tx}), sendo estas mutuamente compatíveis.

Pela Figura 4.5 observa-se que há dez viagens programadas. Estas estão alocadas a três veículos e empregará a mão-de-obra de quatro tripulações. A “Jornada 1” cobrirá as viagens “Viagem Expandida 4” e “Viagem Expandida 7” do “Veículo 3”. Em seguida, a tripulação responsável por esta jornada trocará de veículo e a “Viagem Expandida 10”, do “Veículo 2”, será executada. A “Jornada 2” iniciará suas atividades com a “Viagem Expandida 1”, do “Veículo 3”, posteriormente será realizada a “Viagem Expandida 5”, do “Veículo 1”, e por fim retorna-se ao “Veículo 3” para executar a “Viagem Expandida 9”. A “Jornada 3” será responsável pelas viagens “Viagem Expandida 3” e “Viagem Expandida 6”, ambas do “Veículo 2”, e a “Jornada 4” pelas viagens “Viagem Expandida 2” e “Viagem Expandida 8” do “Veículo 1”.

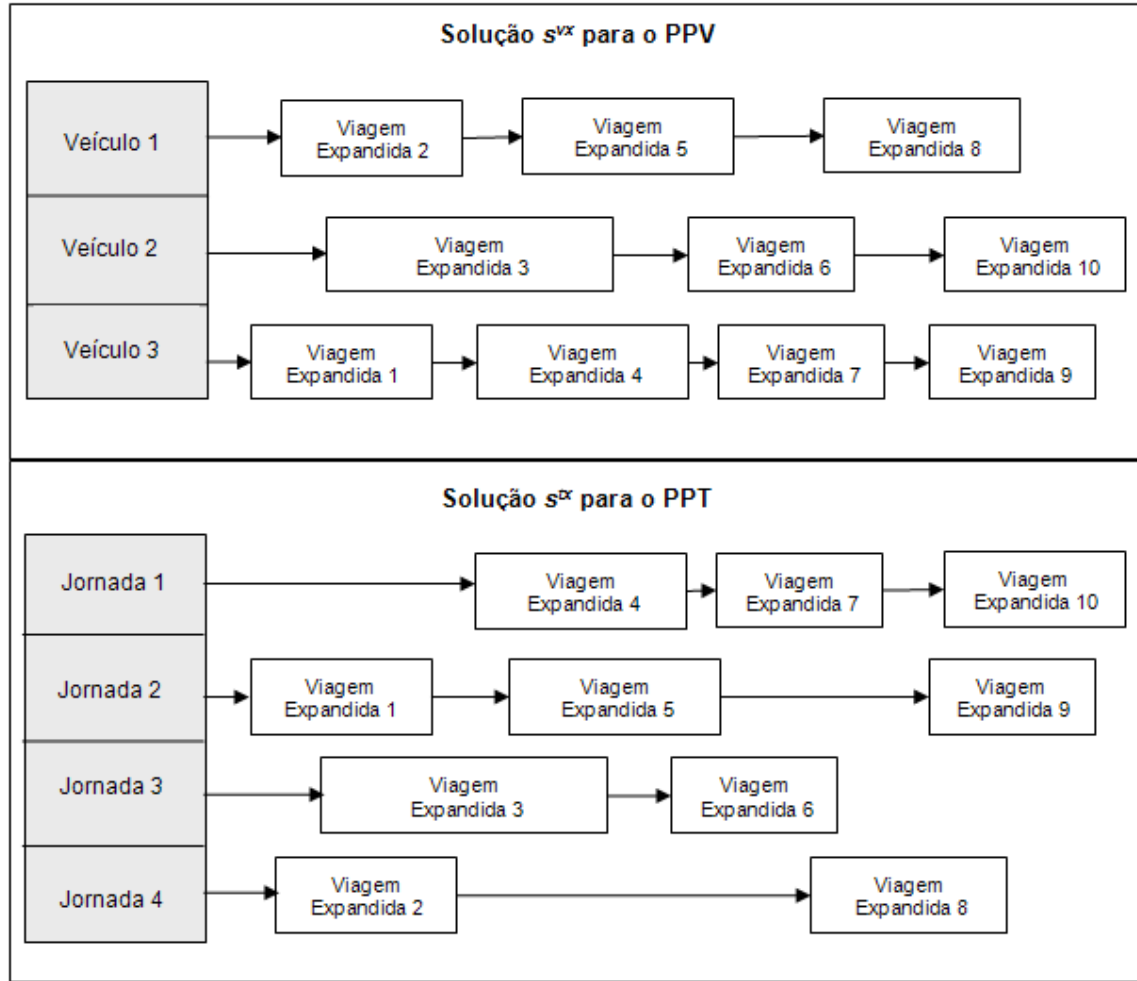


Figura 4.5. Representação de uma solução para o PPVT com definição das viagens expandidas.

4.3.2 Estruturas de Vizinhança

Para exploração do espaço de soluções do PPVT foram utilizados seis tipos de movimentos. Estes e suas respectivas estruturas de vizinhança são: realocação PPV ($VN_x^{(R)}$), realocação PPT ($CN_x^{(R)}$), troca PPV ($VN_x^{(T)}$), troca PPT ($CN_x^{(T)}$), super realocação ($VCN_x^{(R)}$) e super troca ($VCN_x^{(T)}$).

Os movimentos de realocação manipulam uma única viagem expandida modificando o seu veículo e/ou a sua tripulação:

- Realocação PPV: Modifica o veículo que realizará a viagem;

- Realocação PPT: Modifica a tripulação responsável pela viagem;
- Super realocação: Modifica simultaneamente o veículo e a tripulação alocados à viagem.

Os movimentos de troca manipulam duas viagens expandidas permutando os seus veículos e/ou as suas tripulações:

- Troca PPV: Permuta o veículo de uma viagem com o veículo de outra viagem;
- Troca PPT: Permuta a tripulação de uma viagem com a tripulação de outra viagem;
- Super troca: Permuta simultaneamente o veículo e a tripulação alocados a uma viagem com o veículo e a tripulação alocados a outra viagem.

Conforme exposto anteriormente, uma viagem expandida corresponde a uma viagem convencional com atributos adicionais que definem como ela deve ser considerada no PPT, dada a sua disposição no PPV. Portanto, é importante destacar que movimentos que alteram a programação de veículos podem acarretar mudanças nos atributos exclusivos de algumas viagens expandidas. Assim, sempre que uma viagem expandida é transferida de um veículo para outro, os seus atributos exclusivos, bem como os daquelas viagens que imediatamente a precedem e sucedem, no veículo antigo e no novo veículo, poderão estar sendo modificados. Salienta-se que essas mudanças afetarão a solução do PPT e, portanto, é necessário calcular as variações de custo ocorridas nas jornadas das tripulações que realizam as viagens expandidas que foram alteradas.

Nas figuras de 4.7 a 4.12 estão ilustrados os movimentos considerados no PPVT, estando estes aplicados a uma solução dada na Figura 4.6. Nestas ilustrações as viagens que participam diretamente dos movimentos estão destacadas em cinza e as hachuradas são aquelas, cujos atributos, provavelmente foram modificados em decorrência dos movimentos.

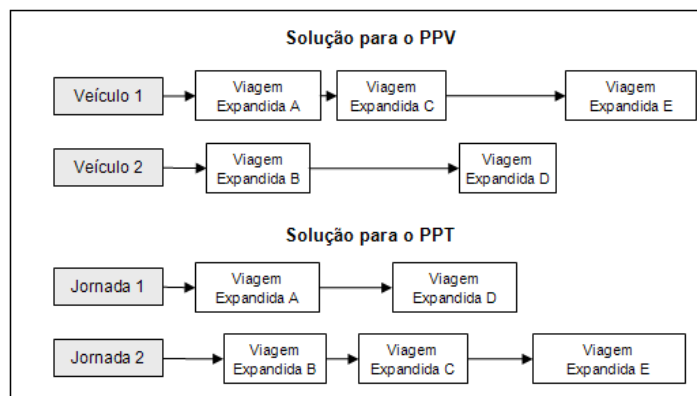


Figura 4.6. Solução para o PPVT utilizada para exemplificar as estruturas de vizinhança.

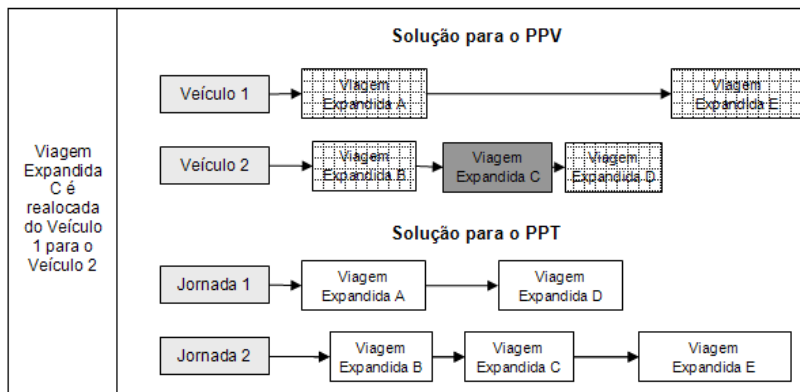


Figura 4.7. Exemplo do movimento realocação PPV.

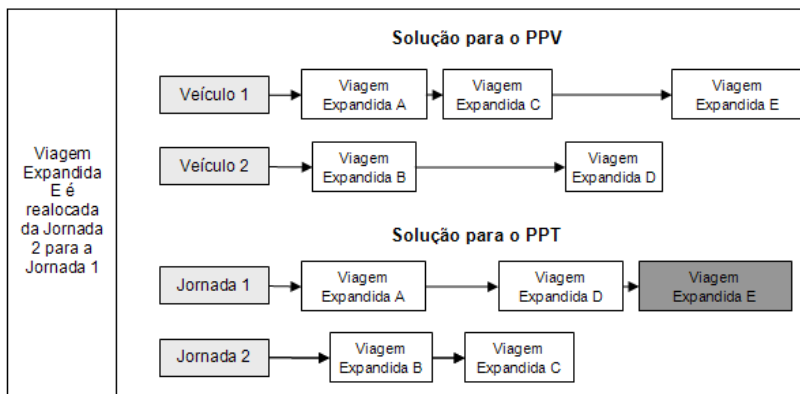


Figura 4.8. Exemplo do movimento realocação PPT.

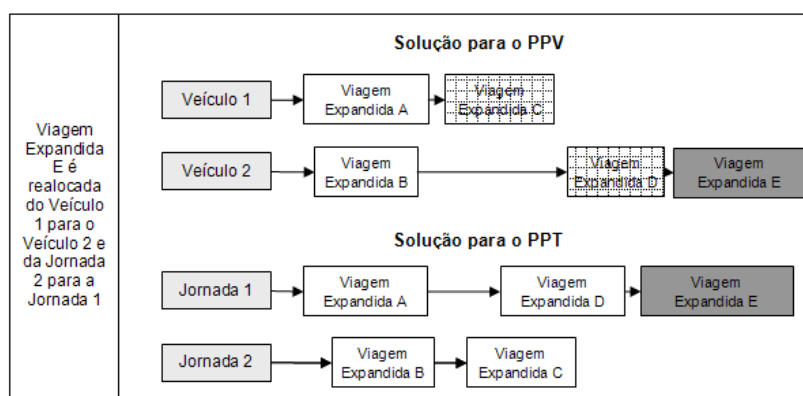


Figura 4.9. Exemplo do movimento super realocação.

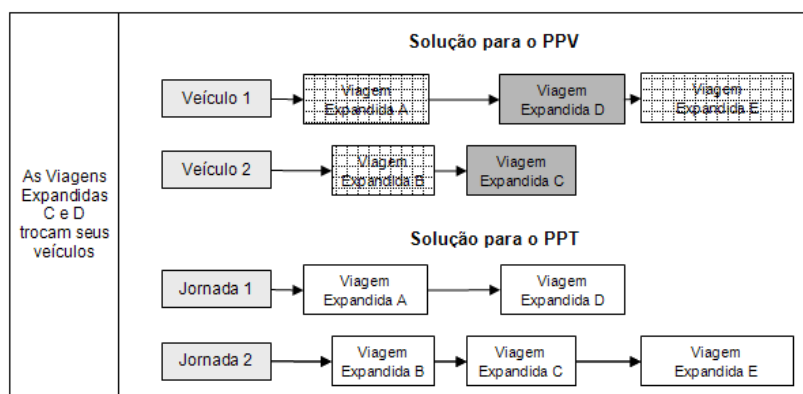


Figura 4.10. Exemplo do movimento troca PPV.

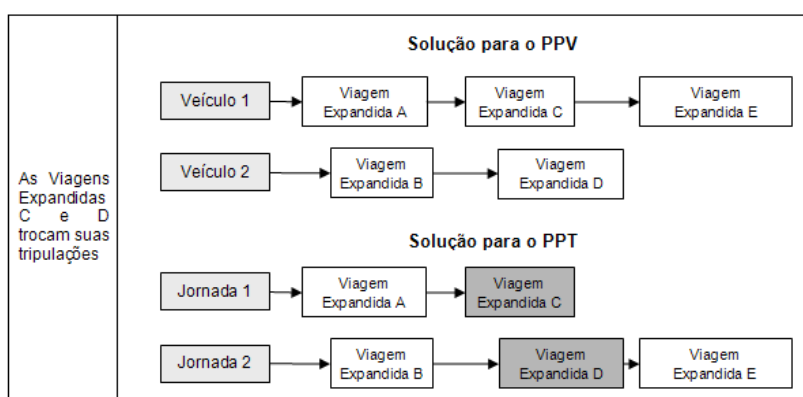


Figura 4.11. Exemplo do movimento troca PPT.

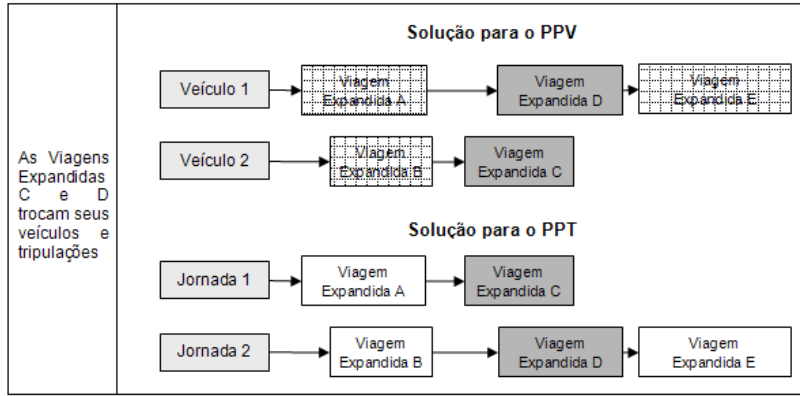


Figura 4.12. Exemplo do movimento super troca.

4.3.3 Função de Avaliação

Como na abordagem integrada a menor porção de trabalho que pode ser atribuída a uma tripulação é a viagem expandida, e não mais a tarefa, surge uma nova possibilidade de construção de jornadas inviáveis no PPT. Trata-se das situações em que ocorre a separação (alocação em diferentes jornadas) das viagens que implicitamente estariam compreendidas em uma mesma tarefa, conforme a especificação de suas dependências à esquerda e à direita.

Portanto, para avaliar as programações obtidas, foi definida a função f^{vtx} dada a seguir pela expressão (4.2). Nessa expressão, os valores retornados pelas funções f^v e f^t estão sendo somados e estes são calculados conforme descrito anteriormente, na abordagem seqüencial. Porém, ao invés de viagens e tarefas, estão sendo consideradas as viagens expandidas, ou seja, as soluções s^{vx} e s^{tx} .

$$f^{vtx}(s^{vx}, s^{tx}) = f^v(s^{vx}) + f^t(s^{tx}) + quebrasTarefas \times pesoInviabilidade \quad (4.2)$$

em que:

- (a) *quebrasTarefas* é o número de vezes que, na solução do PPT, uma viagem expandida encontra-se separada da viagem com a qual mantém uma dependência à esquerda ou à direita;
- (b) *pesoInviabilidade* é o peso dado a cada inviabilidade presente na solução.

4.3.4 Procedimento de Geração da Solução Inicial

A solução inicial para o PPVT é obtida por uma heurística construtiva gulosa, em que as programações dos veículos e das tripulações vão sendo definidas simultaneamente, a cada iteração do processo de construção.

No algoritmo proposto inicia-se com uma tripulação (um motorista e um cobrador), uma frota formada por um único veículo e um conjunto de viagens a serem realizadas (dispostas ordenadas crescentemente pelo horário de início). A cada passo, remove-se a primeira viagem do conjunto e são especificados o veículo e a tripulação que irão realizá-la. Essa combinação deve ser a que implique no menor custo possível determinado pela aplicação da função de avaliação descrita anteriormente pela expressão (4.2). Portanto, a cada iteração são avaliadas todas as possibilidades de alocação da viagem, considerando conjuntamente a mão-de-obra e a frota disponíveis. À medida que a solução vai sendo gerada novos veículos e tripulações são criados dinamicamente, de tal maneira que, sempre haja um veículo e uma tripulação cujas jornadas estão vazias. Esse processo é repetido até que todas as viagens estejam alocadas.

O pseudo-código da heurística construtiva gulosa utilizada no PPVT é descrita a seguir no Algoritmo 10.

4.3.5 Algoritmo de Resolução

A seguir é apresentado o pseudo-código do algoritmo desenvolvido para resolver o PPVT (Algoritmo 11).

Neste algoritmo foram utilizados seis procedimentos de busca local baseados no Método Randômico de Descida, são eles:

1. *DescidaRandomicaComRealocacaoX_PPV*: Que considera durante as buscas a estrutura de vizinhança $VN_x^{(R)}$ (realocação PPV);
2. *DescidaRandomicaComRealocacaoX_PPT*: Que considera durante as buscas a estrutura de vizinhança $CN_x^{(R)}$ (realocação PPT);
3. *DescidaRandomicaComTrocaX_PPV*: Que considera durante as buscas a estrutura de vizinhança $VN_x^{(T)}$ (troca PPV);
4. *DescidaRandomicaComTrocaX_PPT*: Que considera durante as buscas a estrutura de vizinhança $CN_x^{(T)}$ (troca PPT);
5. *DescidaRandomicaComRealocacaoX*: Que considera durante as buscas as estruturas de vizinhança $VN_x^{(R)}$, $CN_x^{(R)}$ e $VCN_x^{(R)}$. Assim, a cada iteração do método

Algoritmo 10: Heurística para geração de uma solução inicial para o PPVT.

-
- 1 Seja T_x o conjunto de viagens a serem realizadas, com T_x ordenado crescentemente por horário de início de suas viagens
 - 2 Seja F o conjunto de veículos da empresa, inicialmente F possui um único veículo sem qualquer viagem
 - 3 Seja J o conjunto de jornadas diárias de trabalho das tripulações, inicialmente J possui uma única jornada sem qualquer viagem
 - 4 **enquanto** $T_x \neq \emptyset$ **faça**
 - 5 $t_x \leftarrow$ primeira viagem de T_x
 - 6 $melhorDelta \leftarrow \infty$, onde $melhorDelta$ é a melhor variação ocorrida na função de avaliação ao se alocar a viagem t_x
 - 7 **para** cada veículo $v \in F$ **faça**
 - 8 $custoAnteriorVeic \leftarrow$ custo do veículo v antes de receber a viagem t_x
 - 9 $custoPosteriorVeic \leftarrow$ novo custo do veículo v caso a viagem t_x seja inserida em sua jornada de trabalho diária
 - 10 **para** cada jornada $j \in J$ **faça**
 - 11 $custoAnteriorJorn \leftarrow$ custo da jornada j antes de receber a viagem t_x
 - 12 $custoPosteriorJorn \leftarrow$ custo da jornada j caso a viagem t_x (supostamente já alocada ao veículo v) passe a pertencer a ela
 - 13 Calcule a variação de custo Δ ocorrida caso a viagem t_x seja alocada ao veículo v e à jornada j :
 - 14 $\Delta = (custoPosteriorVeic + custoPosteriorJorn)$
 - 15 $- (custoAnteriorVeic + custoAnteriorJorn)$
 - 16 **se** $\Delta < melhorDelta$ **então**
 - 17 $melhorDelta \leftarrow \Delta$
 - 18 $melhorVeiculo \leftarrow info(v)$ e $melhorJornada \leftarrow info(j)$, isto é, armazene as informações que permitam identificar o veículo da frota e a jornada que irão contribuir menos para o valor da função de avaliação ao receberem a viagem t_x
 - 19 **se** $melhorVeiculo$ não possui viagens **então**
 - 20 Crie um novo veículo sem viagens e adicione ao conjunto F . (criação dinâmica de veículo, mantendo sempre um veículo sem viagens em F)
 - 21 **se** $melhorJornada$ não possui viagens **então**
 - 22 Crie uma nova jornada sem viagens e adicione ao conjunto J . (criação dinâmica de jornada, mantendo sempre uma jornada sem viagens em J)
 - 23 Aloque a viagem t_x ao veículo da frota F indicado por $melhorVeiculo$ e à jornada do conjunto J indicada por $melhorJornada$
 - 24 $fo \leftarrow fo + melhorDelta$, ou seja, atualize o valor da função de avaliação
 - 25 Remova t_x de T_x

 - 26 Retorne a frota F e o conjunto de jornadas J da empresa

é escolhido aleatoriamente um dos três movimentos: realocação PPV, realocação PPT e super realocação;

6. *DescidaRandomicaComTrocaX*: Que considera durante as buscas as estruturas de vizinhança $VN_x^{(T)}$, $CN_x^{(T)}$ e $VCN_x^{(T)}$. Assim, a cada iteração do método é escolhido aleatoriamente um dos três movimentos: troca PPV, troca PPT e super troca;

As perturbações correspondem a movimentos aleatórios realizados sobre a solução corrente. Foram definidos quatro níveis de perturbação, consecutivamente tem-se: *nível 1* - uma realocação PPT, *nível 2* - uma realocação PPV, *nível 3* - uma troca PPT e *nível 4* - uma troca PPV.

Algoritmo 11: Algoritmo de resolução do PPVT.

```

1  Seja  $T$  o tempo de processamento corrente do algoritmo e  $T_{max}$  o tempo
   máximo admissível de processamento
2  Seja  $nivel$  o nível de perturbação corrente
3   $T \leftarrow 0$ 
4   $nivel \leftarrow 1$ 
5  Gere as soluções iniciais  $s^{vx}$  para o PPV e  $s^{tx}$  para o PPT a partir do método
   guloso descrito pelo Algoritmo 10, na página 55
6   $s_*^{vx} \leftarrow s^{vx}$ , onde  $s_*^{vx}$  é a melhor solução encontrada até então para o PPV
7   $s_*^{tx} \leftarrow s^{tx}$ , onde  $s_*^{tx}$  é a melhor solução encontrada até então para o PPT
8   $s_*^{tx} \leftarrow DescidaRandomicaComRealocacaoX\_PPT(s_*^{tx})$ 
9   $s_*^{tx} \leftarrow DescidaRandomicaComTroca\_PPT(s_*^{tx})$ 
10  $(s_*^{vx}, s_*^{tx}) \leftarrow DescidaRandomicaComRealocacaoX\_PPV(s_*^{vx}, s_*^{tx})$ 
11  $(s_*^{vx}, s_*^{tx}) \leftarrow DescidaRandomicaComTrocaX\_PPV(s_*^{vx}, s_*^{tx})$ 
12  $(s_*^{vx}, s_*^{tx}) \leftarrow DescidaRandomicaComRealocacaoX(s_*^{vx}, s_*^{tx})$ 
13  $(s_*^{vx}, s_*^{tx}) \leftarrow DescidaRandomicaComTrocaX(s_*^{vx}, s_*^{tx})$ 
14 enquanto  $T < T_{max}$  faça
15    $(s^{vx}, s^{tx}) \leftarrow perturbacao(s_*^{vx}, s_*^{tx}, nivel)$  ou  $s^{tx} \leftarrow perturbacao(s_*^{tx}, nivel)$ ,
   conforme o nível de perturbação modifique PPV e/ou PPT
16   se a perturbação realizada envolve apenas realocação ou troca de viagens no
   PPT então
17      $s^{tx} \leftarrow DescidaRandomicaComRealocacaoX\_PPT(s^{tx})$ 
18      $s^{tx} \leftarrow DescidaRandomicaComTrocaX\_PPT(s^{tx})$ 
19   senão
20      $(s^{vx}, s^{tx}) \leftarrow DescidaRandomicaComRealocacaoX\_PPV(s^{vx}, s^{tx})$ 
21      $(s^{vx}, s^{tx}) \leftarrow DescidaRandomicaComTrocaX\_PPV(s^{vx}, s^{tx})$ 
22    $(s^{vx}, s^{tx}) \leftarrow DescidaRandomicaComRealocacaoX(s^{vx}, s^{tx})$ 
23    $(s^{vx}, s^{tx}) \leftarrow DescidaRandomicaComTrocaX(s^{vx}, s^{tx})$ 
24   se  $f^{vt}(s^{vx}, s^{tx}) \leq f^{vt}(s_*^{vx}, s_*^{tx})$  então
25      $s_*^{vx} \leftarrow s^{vx}$ 
26      $s_*^{tx} \leftarrow s^{tx}$ 
27     se  $f^{vt}(s^{vx}, s^{tx}) < f^{vt}(s_*^{vx}, s_*^{tx})$  então
28        $nivel \leftarrow 1$ 
29     senão
30        $nivel \leftarrow nivel + 1$ 
31   senão
32      $nivel \leftarrow nivel + 1$ 
33   se  $nivel > numNiveis$  então
34      $nivel \leftarrow 1$ , onde  $numNiveis$  é o número máximo de níveis de perturbação
35  Retorne as soluções  $s_*^{vx}$  e  $s_*^{tx}$ 

```

Capítulo 5

Experimentos Computacionais

5.1 Descrição das Instâncias

Para validar a metodologia proposta foram disponibilizadas pela empresa responsável pelo gerenciamento do Sistema de Transporte Público da cidade de Belo Horizonte, capital do estado de Minas Gerais, problemas reais de uma dada região da cidade. As tabelas de horários das viagens aqui consideradas são relativas ao mês de maio de 2005.

Há 8 empresas responsáveis pela realização de todas as viagens desta região. Juntamente elas prestam serviços em 40 linhas de ônibus classificadas em dois tipos: troncais e alimentadoras. As linhas troncais operam em áreas onde há alta demanda e têm como objetivo realizar o deslocamento de passageiros entre a estação de ônibus e o centro da cidade. As linhas alimentadoras, que possuem uma abrangência local, têm a função primordial de captar e distribuir a demanda atendida pelas linhas troncais, realizando a ligação bairro-estação-bairro.

A seguir, na Tabela 5.1, é apresentada uma descrição das instâncias utilizadas nos experimentos. Para cada empresa foram considerados separadamente os quatro dias da semana que apresentam quadros de horários distintos: segunda-feira, sexta-feira, sábado e domingo. Assim, cada instância é identificada univocamente através do dia da semana e da empresa aos quais se refere. As empresas serão aqui tratadas pelos seguintes nomes fictícios: G01, G02, G03, G04, G05, G06, G07 e G08. Nesta tabela, após o nome de cada instância estão listados, respectivamente, o seu número de viagens, o tempo total despendido com a execução das mesmas (em horas e minutos) e a duração média de suas viagens (em horas e minutos). Observa-se que as empresas G01, G02 e G08 são responsáveis por poucas viagens, porém sendo estas de longa duração. Já as empresas G03, G04, G05, G06, G07 atendem muitas viagens, no entanto, estas são viagens menores.

Por fim, é bom ressaltar que o tempo total em viagem dado na Tabela 5.1 é sempre

menor do que o tempo real de operação da frota, pois o primeiro não inclui as viagens mortas e as esperas nos terminais realizadas pelos veículos.

Tabela 5.1. Descrição das instâncias utilizadas nos experimentos.

Empresa	Instância	Número de Viagens	Tempo Total em Viagem (hh:mm)	Duração Média das Viagens (hh:mm)
G01	SEG-G01	260	348:10	1:20
	SEX-G01	260	362:17	1:24
	SAB-G01	172	217:16	1:16
	DOM-G01	90	124:27	1:23
G02	SEG-G02	98	141:20	1:27
	SEX-G02	98	147:05	1:30
	SAB-G02	69	91:13	1:19
	DOM-G02	52	64:33	1 14
G03	SEG-G03	505	516:56	1:01
	SEX-G03	498	539:49	0:43
	SAB-G03	372	379:55	1:01
	DOM-G03	286	281:17	0:59
G04	SEG-G04	468	387:05	0:50
	SEX-G04	468	397:35	0:51
	SAB-G04	359	282:50	0:47
	DOM-G04	298	221:04	0:45
G05	SEG-G05	1038	773:58	0:45
	SEX-G05	1036	803:53	0:47
	SAB-G05	769	541:49	0:42
	DOM-G05	538	336:21	0:38
G06	SEG-G06	639	622:30	0:58
	SEX-G06	639	660:44	1:02
	SAB-G06	441	410:20	0:56
	DOM-G06	332	316:15	0:57
G07	SEG-G07	293	253:34	0:52
	SEX-G07	292	267:55	0:55
	SAB-G07	191	147:36	0:46
	DOM-G07	115	60:13	0:31
G08	SEG-G08	206	296:41	1:26
	SEX-G08	203	295:18	1:27
	SAB-G08	130	182:50	1:24
	DOM-G08	108	147:14	1:22

Todos os algoritmos propostos neste trabalho foram desenvolvidos na linguagem C++ e compilados com a versão 4.3.2 do gcc. Os experimentos foram realizados em um microcomputador PC Intel Core 2 Quad, com processador de 2.5 GHz, 4 GB de memória RAM e sob o sistema operacional Linux Ubuntu Hardy Heron 8.04.

Objetivando estabelecer uma proporção entre os gastos da empresa relativos à programação de veículos e à programação de tripulações, foram criados dois custos fundamentais a partir dos quais são definidos todos os demais, são eles:

1. Custo da Unidade de Tempo do Veículo - CUTV: que representa o custo para a empresa do minuto de operação do veículo;
2. Custo da Unidade de Tempo da Tripulação - CUTT: que representa o custo para a empresa do minuto de trabalho de uma tripulação.

Destaca-se que custo nesse contexto, não necessariamente se refere a valores monetários, embora estes possam ser usados. Por exemplo, se em uma dada situação o minuto de trabalho de uma tripulação é duas vezes mais caro que o minuto de operação de um veículo, as atribuições $CUTV = 1$ e $CUTT = 2$ poderiam ser realizadas. Observe que neste caso não foi considerado o valor real gasto pela empresa com uma tripulação ou veículo, mas foi mantida uma proporcionalidade entre estes custos. Nos experimentos realizados considerou-se $CUTT = 5$ e $CUTV = 1$.

Na Tabela 5.2 são apresentados os valores dos pesos utilizados para o cálculo da função de avaliação referente à programação de veículos, dada pela expressão 3.1 apresentada na subseção 3.3.3. Nessa tabela *tempoMaxOperacao* corresponde ao tempo máximo de operação diária do veículo, em minutos. Nos testes, *tempoMaxOperacao* é igual a 1410 minutos, ou seja, 23 horas e 30 minutos.

Tabela 5.2. Penalizações utilizadas no PPV e seus respectivos valores.

Penalização	Valor
<i>custoUsoVeiculo</i>	$tempoMaxOperacao \times CUTV$
<i>custoViagemMorta</i>	$CUTV$
<i>pesoInviabilidade</i>	$1,5 \times CUTV$

Os valores dos pesos utilizados na composição da função de avaliação para a programação de tripulações, apresentada na subseção 3.5.3 e dada pela expressão 3.3, são mostrados na Tabela 5.3. Nessa tabela *duracaoMaxJornada* corresponde ao tempo máximo, sem horas extras, efetivamente trabalhado por uma tripulação (em minutos). Nos testes, *duracaoMaxJornada* é igual a 400 minutos, isto é, 6 horas e 40 minutos.

Tabela 5.3. Penalizações utilizadas no PPT e seus respectivos valores.

Penalização	Valor
<i>custoUsoTripulacao</i>	$duracaoMaxJornada \times CUTT$
<i>custoHoraExtra</i>	$0,75 \times CUTT$
<i>pesoInviabilidade</i>	$1,5 \times CUTT$

O tempo de execução dos algoritmos foi fixado em 10 minutos para todas as instâncias. Nas abordagens independente e seqüencial tradicional foram despendidos tempos iguais para resolver separadamente, PPV e PPT, ou seja, 5 minutos para cada.

5.2 Resultados

A Tabela 5.4 mostra uma comparação entre os custos das soluções obtidas pelas quatro metodologias apresentadas nesse trabalho para definição das programações de veículos e tripulações, são elas: independente, seqüencial tradicional, iterativa e integrada. Nesta tabela, “Melhor FO” e “FO Médio” são, respectivamente, o menor custo e a média dos custos obtidos pelos métodos desenvolvidos em 10 execuções, considerando PPV e PPT. A “Melhoria” representa o aprimoramento, em percentual, do valor da melhor solução encontrada em cada abordagem comparada com o custo da melhor solução obtida na abordagem seqüencial. O “Desvio” corresponde ao desvio médio de cada abordagem em relação ao valor da melhor solução encontrada nos testes, isto é:

$$desvio = \frac{FOMedio - MelhorFO}{MelhorFO} \quad (5.1)$$

Por fim, são apresentadas na Tabela 5.4 as médias para o desvio e a melhoria, considerando todas as instâncias. É importante frisar que os problemas sendo tratados são de minimização, ou seja, o objetivo é reduzir os custos das soluções.

Tabela 5.4: Resultados considerando os custos das soluções obtidas pelas abordagens independente, seqüencial, iterativa e integrada.

Intância	Avaliação	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEG-G01	Melhor FO	188184	206401	198690	199071
	FO Médio	188212	207752	200878	199486
	Desvio (%)	0.02	0.65	1.10	0.21
	Melhoria (%)	8.83	0.00	3.74	3.55

Intância	Avaliação	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEX-G01	Melhor FO	186960	205476	197726	198022
	FO Médio	194633	206860	199879	201015
	Desvio (%)	4.10	0.67	1.09	1.51
	Melhoria (%)	9.01	0.00	3.77	3.63
SAB-G01	Melhor FO	104029	114821	113601	109976
	FO Médio	104181	118892	113991	111540
	Desvio (%)	0.15	3.55	0.34	1.42
	Melhoria (%)	9.40	0.00	1.06	4.22
DOM-G01	Melhor FO	52785	56533	56355	56207
	FO Médio	53622	57822	57119	56368
	Desvio (%)	1.59	2.28	1.35	0.29
	Melhoria (%)	6.63	0.00	0.31	0.58
SEG-G02	Melhor FO	64143	69561	69103	69328
	FO Médio	64245	70335	69207	69782
	Desvio (%)	0.16	1.11	0.15	0.65
	Melhoria (%)	7.79	0.00	0.66	0.33
SEX-G02	Melhor FO	64338	70188	69454	68597
	FO Médio	64999	70562	69663	69510
	Desvio (%)	1.03	0.53	0.30	1.33
	Melhoria (%)	8.33	0.00	1.05	2.27
SAB-G02	Melhor FO	39122	41818	41023	39863
	FO Médio	39216	42668	41123	39920
	Desvio (%)	0.24	2.03	0.24	0.14
	Melhoria (%)	6.45	0.00	1.90	4.68
DOM-G02	Melhor FO	27823	30291	28507	29685
	FO Médio	27863	30857	29321	29833
	Desvio (%)	0.14	1.87	2.85	0.50
	Melhoria (%)	8.15	0.00	5.89	2.00
SEG-G03	Melhor FO	234780	267525	260576	252235
	FO Médio	237778	272265	266441	252888
	Desvio (%)	1.28	1.77	2.25	0.26
	Melhoria (%)	12.24	0.00	2.60	5.72

Intância	Avaliação	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEX-G03	Melhor FO	240457	267965	267924	257405
	FO Médio	241628	272355	271511	257756
	Desvio (%)	0.49	1.64	1.34	0.14
	Melhoria (%)	10.27	0.00	0.02	3.94
SAB-G03	Melhor FO	151853	173211	169502	164866
	FO Médio	152896	177362	171261	165222
	Desvio (%)	0.69	2.40	1.04	0.22
	Melhoria (%)	12.33	0.00	2.14	4.82
DOM-G03	Melhor FO	116330	131449	128258	124673
	FO Médio	117416	133678	130658	126119
	Desvio (%)	0.93	1.70	1.87	1.16
	Melhoria (%)	11.50	0.00	2.43	5.15
SEG-G04	Melhor FO	163948	189628	188030	175311
	FO Médio	165729	193688	193302	175976
	Desvio (%)	1.09	2.14	2.80	0.38
	Melhoria (%)	13.54	0.00	0.84	7.55
SEX-G04	Melhor FO	167986	188238	193405	177275
	FO Médio	168576	193709	194888	177492
	Desvio (%)	0.35	2.91	0.77	0.12
	Melhoria (%)	10.76	0.00	-2.74	5.82
SAB-G04	Melhor FO	116263	133575	130764	126656
	FO Médio	117397	138047	131423	128722
	Desvio (%)	0.98	3.35	0.50	1.63
	Melhoria (%)	12.96	0.00	2.10	5.18
DOM-G04	Melhor FO	92377	104791	104560	98676
	FO Médio	92990	108095	105246	98885
	Desvio (%)	0.66	3.15	0.66	0.21
	Melhoria (%)	11.85	0.00	0.22	5.84
SEG-G05	Melhor FO	346752	400873	404482	377157
	FO Médio	347752	403410	416529	378377
	Desvio (%)	0.29	0.63	2.98	0.32
	Melhoria (%)	13.50	0.00	-0.90	5.92

Intância	Avaliação	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEX-G05	Melhor FO	353728	400175	416129	381173
	FO Médio	354856	408269	420943	381818
	Desvio (%)	0.32	2.02	1.16	0.17
	Melhoria (%)	11.61	0.00	-3.99	4.75
SAB-G05	Melhor FO	223336	256646	255575	241439
	FO Médio	223880	260492	261490	241985
	Desvio (%)	0.24	1.50	2.31	0.23
	Melhoria (%)	12.98	0.00	0.42	5.93
DOM-G05	Melhor FO	137572	158808	157053	149947
	FO Médio	138775	162009	157967	150879
	Desvio (%)	0.87	2.02	0.58	0.62
	Melhoria (%)	13.37	0.00	1.11	5.58
SEG-G06	Melhor FO	280346	313502	312840	303197
	FO Médio	281366	321166	319985	303786
	Desvio (%)	0.36	2.44	2.28	0.19
	Melhoria (%)	10.58	0.00	0.21	3.29
SEX-G06	Melhor FO	295477	334657	339962	322667
	FO Médio	299173	338795	344935	323311
	Desvio (%)	1.25	1.24	1.46	0.20
	Melhoria (%)	11.71	0.00	-1.59	3.58
SAB-G06	Melhor FO	165118	193037	188598	179639
	FO Médio	166562	195165	192456	179997
	Desvio (%)	0.87	1.10	2.05	0.20
	Melhoria (%)	14.46	0.00	2.30	6.94
DOM-G06	Melhor FO	129604	150935	145810	140249
	FO Médio	130094	153483	146001	141200
	Desvio (%)	0.38	1.69	0.13	0.68
	Melhoria (%)	14.13	0.00	3.40	7.08
SEG-G07	Melhor FO	117124	131954	128405	126023
	FO Médio	118978	134641	133693	126240
	Desvio (%)	1.58	2.04	4.12	0.17
	Melhoria (%)	11.24	0.00	2.69	4.49

Intância	Avaliação	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEX-G07	Melhor FO	124389	136460	132810	129068
	FO Médio	125668	138069	135559	129832
	Desvio (%)	1.03	1.18	2.07	0.59
	Melhoria (%)	8.85	0.00	2.67	5.42
SAB-G07	Melhor FO	65074	71026	69286	68426
	FO Médio	65238	72410	70224	68581
	Desvio (%)	0.25	1.95	1.35	0.23
	Melhoria (%)	8.38	0.00	2.45	3.66
DOM-G07	Melhor FO	30875	33450	32084	32014
	FO Médio	31123	34064	33147	32137
	Desvio (%)	0.80	1.84	3.31	0.38
	Melhoria (%)	7.70	0.00	4.08	4.29
SEG-G08	Melhor FO	137326	148608	145909	144207
	FO Médio	138364	150788	150006	144723
	Desvio (%)	0.76	1.47	2.81	0.36
	Melhoria (%)	7.59	0.00	1.82	2.96
SEX-G08	Melhor FO	137141	153945	149802	152010
	FO Médio	141077	155694	152879	153518
	Desvio (%)	2.87	1.14	2.05	0.99
	Melhoria (%)	10.92	0.00	2.69	1.26
SAB-G08	Melhor FO	77555	87444	85303	85896
	FO Médio	77705	90141	86307	86313
	Desvio (%)	0.19	3.08	1.18	0.49
	Melhoria (%)	11.31	0.00	2.45	1.77
DOM-G08	Melhor FO	64171	73763	71370	72581
	FO Médio	65009	75332	71820	72669
	Desvio (%)	1.31	2.13	0.63	0.12
	Melhoria (%)	13.00	0.00	3.24	1.60
Média	Desvio (%)	0.85	1.85	1.54	0.50
	Melhoria (%)	10.67	0.00	1.53	4.18

Com base nos resultados da Tabela 5.4, verifica-se que como era de se esperar as melhores soluções foram obtidas pela abordagem independente, porém, conforme alertado anteriormente, as programações dos veículos e das tripulações encontradas são incompatíveis e, portanto, impraticáveis. A abordagem independente está sendo usada

apenas para indicar o quão promissora é a resolução integrada do PPV e PPT, uma vez que considera isoladamente os dois problemas, quebrando dessa forma as limitações que a resolução do PPV impõe ao PPT e vice-versa. Considerando as demais abordagens, a que obteve melhor resultado foi a integrada, e em seguida a iterativa, evidenciando assim, a relevância da integração. O pior desempenho foi do método de resolução seqüencial. Dentre as 32 instâncias consideradas, em 25 delas o método integrado encontrou os melhores resultados e para as demais 7 instâncias (SEG-G01, SEX-G01, SEG-G02, DOM-G02, SEX-G08, SAB-G08, DOM-G08) o método de resolução iterativa teve melhor desempenho, alcançando os menores custos. Destaca-se que para as instâncias SEX-G04, SEG-G05, SEX-G05 e SEX-G05 a abordagem iterativa não conseguiu melhorar os resultados gerados de forma seqüencial, por outro lado, a abordagem integrada foi capaz de aprimorar o custo de todas as soluções obtidas seqüencialmente.

Ainda na Tabela 5.4, analisando os valores de desvio constata-se a robustez dos métodos desenvolvidos e, dentre os quais, o procedimento de resolução integrada se mostrou o mais estável.

Na Tabela 5.5 as melhores soluções obtidas pelas diferentes abordagens são detalhadas. Nessa tabela, a “Melhoria” representa o aprimoramento, em percentual, do valor da melhor solução encontrada em cada abordagem comparada com o custo da melhor solução obtida na abordagem seqüencial (similar à Tabela 5.4) e, em seguida, estão listados os atributos das programações, são eles: o número de veículos e tripulações utilizados, o tempo total de viagens mortas dos veículos e o tempo total de horas extras das tripulações. Por fim, nessa tabela, também são apresentados os valores totais dos atributos das programações e, os tempos médios de viagens mortas por veículo e de horas extras por tripulação, considerando todas as instâncias. Salienta-se que todos os tempos são dados em horas e minutos (formato hh:mm).

Tabela 5.5: Detalhamento das melhores soluções obtidas pelas abordagens independente, seqüencial, iterativa e integrada.

Intância	Atributos	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEG-G01	Melhoria (%)	8.83	0.00	3.74	3.55
	Veículos	40	40	40	40
	Tripulações	65	74	70	70
	Viagens mortas	29:44	30:12	31:52	36:04
	Horas extras	00:00	01:03	02:06	02:49

Intância	Atributos	Abordagens			
		Independente	Sequencial	Iterativa	Integrada
SEX-G01	Melhoria (%)	9.01	0.00	3.77	3.63
	Veículos	39	39	39	39
	Tripulações	65	74	70	70
	Viagens mortas	30:56	30:20	33:54	37:08
	Horas extras	00:38	03:42	03:54	04:28
SAB-G01	Melhoria (%)	9.40	0.00	1.06	4.22
	Veículos	23	23	23	23
	Tripulações	35	40	40	37
	Viagens mortas	13:48	13:48	14:16	16:36
	Horas extras	04:17	08:41	01:45	14:10
DOM-G01	Melhoria (%)	6.63	0.00	0.31	0.58
	Veículos	10	10	10	10
	Tripulações	18	20	20	20
	Viagens mortas	06:00	06:00	06:56	07:10
	Horas extras	12:55	11:31	10:13	09:19
SEG-G02	Melhoria (%)	7.79	0.00	0.66	0.33
	Veículos	13	13	13	13
	Tripulações	22	25	25	25
	Viagens mortas	09:58	09:58	10:44	10:44
	Horas extras	06:45	03:31	00:43	01:58
SEX-G02	Melhoria (%)	8.33	0.00	1.05	2.27
	Veículos	13	13	13	13
	Tripulações	22	25	25	24
	Viagens mortas	09:58	09:58	10:44	10:44
	Horas extras	07:50	07:00	02:40	09:01
SAB-G02	Melhoria (%)	6.45	0.00	1.90	4.68
	Veículos	7	7	7	7
	Tripulações	14	15	15	14
	Viagens mortas	05:22	05:22	05:22	05:22
	Horas extras	05:10	09:02	04:37	09:17
DOM-G02	Melhoria (%)	8.15	0.00	5.89	2.00
	Veículos	5	5	5	5
	Tripulações	10	11	10	11
	Viagens mortas	03:50	03:50	03:50	03:50
	Horas extras	03:01	05:37	06:49	02:15

Intância	Atributos	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEG-G03	Melhoria (%)	12.24	0.00	2.60	5.72
	Veículos	47	47	47	48
	Tripulações	83	100	96	90
	Viagens mortas	16:35	16:16	19:17	27:58
	Horas extras	08:25	01:33	06:23	15:59
SEX-G03	Melhoria (%)	10.27	0.00	0.02	3.94
	Veículos	47	47	47	48
	Tripulações	86	100	100	93
	Viagens mortas	15:33	15:33	17:40	24:47
	Horas extras	06:58	04:14	03:18	12:26
SAB-G03	Melhoria (%)	12.33	0.00	2.14	4.82
	Veículos	27	27	27	27
	Tripulações	52	65	64	60
	Viagens mortas	08:42	08:08	09:42	18:28
	Horas extras	51:27	25:51	15:50	31:36
DOM-G03	Melhoria (%)	11.50	0.00	2.43	5.15
	Veículos	21	21	21	21
	Tripulações	41	50	48	45
	Viagens mortas	06:28	06:24	07:14	13:05
	Horas extras	24:04	08:05	12:18	23:46
SEG-G04	Melhoria (%)	13.54	0.00	0.84	7.55
	Veículos	31	31	31	31
	Tripulações	57	72	71	62
	Viagens mortas	12:25	12:07	14:10	20:59
	Horas extras	30:31	06:37	08:10	35:14
SEX-G04	Melhoria (%)	10.76	0.00	-2.74	5.82
	Veículos	31	31	31	31
	Tripulações	59	70	74	62
	Viagens mortas	12:30	12:37	14:39	21:25
	Horas extras	30:42	20:57	04:32	46:00
SAB-G04	Melhoria (%)	12.96	0.00	2.10	5.18
	Veículos	20	20	20	20
	Tripulações	41	51	50	46
	Viagens mortas	07:39	07:33	09:20	16:45
	Horas extras	31:08	16:14	11:08	30:17

Intância	Atributos	Abordagens			
		Independente	Sequencial	Iterativa	Integrada
DOM-G04	Melhoria (%)	11.85	0.00	0.22	5.84
	Veículos	16	16	16	16
	Tripulações	32	40	40	35
	Viagens mortas	06:18	06:23	06:47	11:56
	Horas extras	30:13	10:16	08:51	30:00
SEG-G05	Melhoria (%)	13.50	0.00	-0.90	5.92
	Veículos	70	70	70	70
	Tripulações	122	150	151	135
	Viagens mortas	30:05	29:49	34:17	44:33
	Horas extras	12:29	02:08	09:35	32:08
SEX-G05	Melhoria (%)	11.61	0.00	-3.99	4.75
	Veículos	70	70	70	70
	Tripulações	125	148	157	137
	Viagens mortas	30:42	30:21	34:30	44:01
	Horas extras	17:42	20:18	07:33	32:24
SAB-G05	Melhoria (%)	12.98	0.00	0.42	5.93
	Veículos	42	42	42	42
	Tripulações	75	96	96	85
	Viagens mortas	17:01	16:56	18:05	25:57
	Horas extras	72:45	24:30	18:10	59:14
DOM-G05	Melhoria (%)	13.37	0.00	1.11	5.58
	Veículos	24	24	24	24
	Tripulações	46	60	60	53
	Viagens mortas	09:53	10:21	12:03	20:45
	Horas extras	61:53	24:09	13:50	49:14
SEG-G06	Melhoria (%)	10.58	0.00	0.21	3.29
	Veículos	55	55	55	55
	Tripulações	99	116	115	109
	Viagens mortas	36:26	35:55	41:25	47:15
	Horas extras	14:30	09:59	15:35	26:44
SEX-G06	Melhoria (%)	11.71	0.00	-1.59	3.58
	Veículos	60	60	60	60
	Tripulações	103	123	125	115
	Viagens mortas	39:32	38:49	46:55	51:09
	Horas extras	13:55	09:36	14:09	27:46

Intância	Atributos	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SAB-G06	Melhoria (%)	14.46	0.00	2.30	6.94
	Veículos	30	30	30	30
	Tripulações	55	73	71	63
	Viagens mortas	18:17	18:00	20:56	26:41
	Horas extras	65:07	20:19	16:54	54:06
DOM-G06	Melhoria (%)	14.13	0.00	3.40	7.08
	Veículos	23	23	23	23
	Tripulações	44	58	54	50
	Viagens mortas	13:51	13:39	17:40	22:01
	Horas extras	46:21	09:22	24:00	36:06
SEG-G07	Melhoria (%)	11.24	0.00	2.69	4.49
	Veículos	24	24	24	24
	Tripulações	41	48	46	45
	Viagens mortas	11:12	11:11	13:24	15:50
	Horas extras	03:24	08:01	09:47	06:51
SEX-G07	Melhoria (%)	8.85	0.00	2.67	5.42
	Veículos	25	25	25	25
	Tripulações	44	50	48	45
	Viagens mortas	11:08	10:40	13:30	16:32
	Horas extras	02:37	03:10	04:10	15:42
SAB-G07	Melhoria (%)	8.38	0.00	2.45	3.66
	Veículos	12	12	12	12
	Tripulações	23	26	25	24
	Viagens mortas	04:42	04:48	05:50	09:20
	Horas extras	10:24	10:06	11:12	16:22
DOM-G07	Melhoria (%)	7.70	0.00	4.08	4.29
	Veículos	5	5	5	5
	Tripulações	11	13	12	12
	Viagens mortas	01:10	01:10	01:53	01:34
	Horas extras	09:45	01:50	05:07	04:50
SEG-G08	Melhoria (%)	7.59	0.00	1.82	2.96
	Veículos	26	26	26	26
	Tripulações	49	53	52	51
	Viagens mortas	43:20	43:20	43:20	45:00
	Horas extras	00:22	18:36	14:43	15:49

Intância	Atributos	Abordagens			
		Independente	Seqüencial	Iterativa	Integrada
SEX-G08	Melhoria (%)	10.92	0.00	2.69	1.26
	Veículos	28	28	28	28
	Tripulações	47	55	52	54
	Viagens mortas	46:40	46:40	46:40	50:00
	Horas extras	04:47	09:15	19:34	08:30
SAB-G08	Melhoria (%)	11.31	0.00	2.45	1.77
	Veículos	14	14	14	14
	Tripulações	27	32	31	32
	Viagens mortas	23:20	23:20	23:20	23:20
	Horas extras	13:25	12:48	12:01	04:12
DOM-G08	Melhoria (%)	13.00	0.00	3.24	1.60
	Veículos	13	13	13	13
	Tripulações	21	26	25	26
	Viagens mortas	21:40	21:40	21:40	21:40
	Horas extras	14:07	11:51	09:40	05:17
Total	Veículos	911	911	911	913
	Tripulações	1634	1959	1938	1800
	Viagens mortas	554:45	551:08	611:55	748:39
	Horas extras	617:37	339:52	309:17	673:50
Média	Viagens mortas por veículo	00:37	00:36	00:40	00:49
	Horas extras por tripulação	00:22	00:10	00:10	00:22

A partir da Tabela 5.5 observa-se que os tempos totais de viagens mortas obtidos nas abordagens independente e seqüencial foram menores do que os encontrados de forma iterativa ou integrada. Este era um resultado previsível, uma vez que diferentemente do que ocorre na resolução seqüencial, no PPVT limita-se o aprimoramento da programação de veículos ao considerar os efeitos que esta melhoria irá provocar na programação de tripulações. Porém, ainda assim, o número de veículos utilizados manteve-se o mesmo em todas as abordagens, exceto para as instâncias SEG-G03 e SEX-G03 em que a resolução integrada implicou na utilização de um veículo a mais. As melhorias proporcionadas pelas abordagens iterativa e integrada estão concentradas nas soluções para o PPT. Observa-se que o ganho se dá, principalmente, em relação ao número de tripulações empregadas. Nesse sentido, é interessante relembrar que

uma tripulação envolve a mão-de-obra de dois funcionários, um motorista e um cobrador. Portanto, ao analisar na Tabela 5.5 o total de tripulações utilizadas, verifica-se que comparadas com a abordagem seqüencial, a abordagem iterativa proporciona uma redução de 21 tripulações, ou seja, 42 funcionários e a abordagem integrada evita a utilização de 159 tripulações que correspondem a 318 funcionários.

Destaca-se que o objetivo primordial dos testes realizados foi a redução do número de veículos e tripulantes empregados, conforme pode ser observado pelos custos dados aos atributos das programações. Secundariamente, visou-se reduzir as horas extras das tripulações e os deslocamentos fora de operação realizados pelos veículos.

Capítulo 6

Conclusões e Trabalhos Futuros

Esse trabalho abordou o Problema de Programação Integrada de Veículos e Tripulações (PPVT) considerando um cenário real envolvendo empresas do sistema de transporte público por ônibus da cidade de Belo Horizonte. Esse é um problema de grande importância, uma vez que através da sua resolução as empresas do setor podem reduzir custos operacionais, aprimorar processos produtivos e, conseqüentemente, oferecer um serviço de qualidade, eficiente e mais barato aos usuários.

A definição simultânea da rotina de operação da frota de veículos e das jornadas de trabalho das tripulações é uma metodologia ainda pouco explorada na literatura devido à sua complexidade, mas é considerada promissora no que diz respeito à qualidade das soluções geradas. Portanto, a principal contribuição desse trabalho foi propor uma nova estratégia para tratar o PPVT e verificar a sua efetividade na resolução de problemas reais.

Foram propostas heurísticas, baseadas na metaheurística Busca Local Iterada, para definição das programações de veículos e tripulações considerando quatro abordagens distintas:

1. Abordagem Seqüencial Tradicional: Primeiro é resolvido o PPV, cuja solução consiste em um conjunto de blocos de viagens, sendo cada bloco a jornada diária de trabalho de um veículo. Antes de ser iniciada a resolução do PPT, os blocos de viagens são simplificados em tarefas. Cada tarefa consiste em um conjunto de viagens entre as quais não é possível realizar a troca de tripulações e onde são contemplados os deslocamentos do veículo que não correspondem a viagens;
2. Abordagem Independente: PPV e PPT são resolvidos separadamente, em qualquer ordem e de forma a não se relacionarem. Nesta abordagem tanto a programação de veículos quanto a programação de tripulações são determinadas a partir das viagens a serem realizadas, não havendo a formação de tarefas. Assim,

difficilmente as soluções obtidas são compatíveis e factíveis na prática. A principal finalidade desta metodologia independente é a determinação de uma melhor programação para as tripulações desconsiderando os veículos. Desta forma, uma comparação entre as soluções obtidas nesta abordagem e na seqüencial tradicional permite identificar os possíveis benefícios de se resolver PPV e PPT de forma integrada;

3. Abordagem Seqüencial Iterativa: Como primeiro passo para a integração entre PPV e PPT foi desenvolvida uma estratégia iterativa de resolução destes problemas em que, ora a busca por melhoras se concentra especificamente no PPV e ora no PPT. Nesta abordagem a principal preocupação é manter as programações de veículos e tripulações compatíveis entre si;
4. Abordagem Integrada: PPV e PPT são resolvidos de forma simultânea. O desafio encontrado é, durante a resolução, encontrar uma maneira de manter as programações de veículos e tripulações sendo geradas compatíveis. Neste sentido, foi elaborada uma representação baseada na definição de uma nova estrutura denominada viagem expandida. Uma viagem expandida corresponde a uma viagem com atributos adicionais que definem como ela deve ser considerada no PPT, dada a sua disposição no PPV. Os novos atributos das viagens permitem contemplar as mesmas condições asseguradas ao se formar tarefas na abordagem seqüencial. Nesta estratégia tanto a programação de veículos quanto a programação de tripulações são determinadas a partir das viagens expandidas, criando grande flexibilização para obtenção da solução.

Testes foram realizados com instâncias reais relativas aos quatro dias da semana com quadros de horários das viagens distintos (segunda-feira, sexta-feira, sábado e domingo) de 8 empresas do sistema de transporte público por ônibus da cidade de Belo Horizonte. Dentre as 32 instâncias consideradas, em 25 delas o método integrado encontrou os melhores resultados e para as demais 7 instâncias o método de resolução iterativa teve melhor desempenho, alcançando os menores custos, evidenciando assim, a relevância da integração.

Como continuidade deste trabalho pode-se propor, primeiramente, realizar experimentos alterando a função de avaliação de forma a priorizar outras métricas. Lembrando que, nos testes apresentados, o objetivo primordial foi a redução do número de veículos e tripulantes empregados. Outro aprimoramento seria comparar as soluções obtidas pela heurística integrada proposta com soluções ótimas geradas por um método exato. Destaca-se que para as instâncias estudadas, provavelmente não seria possível

obter soluções ótimas em um tempo aceitável, portanto instâncias fictícias menores precisariam ser criadas.

Outra extensão deste trabalho pode considerar a integração entre os problemas de definir a tabela de horários das viagens, a programação dos veículos e a programação das tripulações. Assim, a tabela de horários seria definida de forma a estrategicamente reduzir o custo operacional das programações de veículos e tripulações, além de considerar a disponibilidade de infra-estrutura, os serviços requeridos pelos usuários e os aspectos de demanda.

A Programação Integrada de Veículos e Tripulações com Múltiplas Garagens também é uma proposta a ser considerada para a continuidade da pesquisa. Esse problema é resolvido considerando que a frota de veículos, disponível para realização das viagens, se encontra espalhada em diferentes garagens e os veículos não necessitam iniciar e terminar a sua rotina de operação diária em uma mesma garagem.

Referências Bibliográficas

- ANTP (1999). Associação Nacional dos Transportes Públicos. Disponível na internet. URL: <http://www.antp.org.br>. Acessado em 17/10/2007.
- Baita, F.; Pesenti, R.; Ukovich, W. e Favaretto, D. (2000). A comparison of different solution approaches to the vehicle scheduling problem in a practical case. *Computers and Operations Research*, 27:1249–1269.
- Ball, M. O.; Bodin, L. D. e Dial, R. (1983). A matching based heuristic for scheduling mass transit crew and vehicles. *Transportation Science*, 1(17):4–31.
- Daduna, J. R. e Voss, S. (2001). Lecture notes in economics and mathematical systems. *Computer-Aided Scheduling of Public Transport*, 505. Berlim: Springer Publishers.
- Desrochers, M. e Soumis, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23:1–13.
- Elias, S. E. G. (1964). The use of digital computers in the economic scheduling for both man and machine in public transportation. Manhattan, Kansas.
- Freling, R. (1997). *Models and techniques for integrating vehicle and crew scheduling*. PhD thesis, Erasmus University Rotterdam, Amsterdam.
- Freling, R.; Boender, C. G. E. e Paixão, J. M. P. (1995). An integrated approach to vehicle and crew scheduling. *Technical Report 9503/A*. Econometric Institute, Erasmus University Rotterdam, Rotterdam.
- Freling, R.; Huisman, D. e Wagelmans, A. P. M. (2003). Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6(1):63–85.
- Freling, R.; Paixão, J. M. P. e Wagelmans, A. P. M. (2001). Models and algorithms for vehicle scheduling. *Transportation Science*, 35:165–180.
- Freling, R.; Wagelmans, A. P. M. e Paixão, J. M. P. (1999). An overview of models and techniques for integrating vehicle and crew scheduling. *Computer-Aided Transit Scheduling*, 471:441–460.

- Haase, K. e Friberg, C. (1999). An exact branch and cut algorithm for the vehicle and crew scheduling problem. *Computer-Aided Transit Scheduling*, pp. 63–80.
- Hoffman, K. L. e Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39:657–682.
- Huisman, D.; Freling, R. e Wagelmans, A. P. M. (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, 39(4):491–502.
- Kirkman, F. (1968). Problems of innovation in the transport industry: a bus scheduling program. In *Proceedings of PTRC Public Transport Analysis Seminar, Planning and Transport Research and Computation Co. Ltd.*, volume 1, pp. 1–15.
- Laurent, B. e Hao, J. (2007). Simultaneous vehicle and driver scheduling: A case study in a limousine rental company. *Computers and Industrial Engineering*, 53(3):542–558.
- Lourenço, H. R.; Martin, O. C. e Stutzle, T. (2003). Iterated local search. In Glover, F. e Kochenberger, G., editores, *Handbook of Metaheuristics*. Springer.
- Marinho, E. H. (2005). Heurísticas busca tabu para o problema de programação de tripulações de ônibus urbano. Dissertação de mestrado. Universidade Federal Fluminense, Niterói - RJ.
- Mesquita, M. e Paias, A. (2008). Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. *Computers and Operations Research*, 35(5):1562–1575.
- Patrikalakis, I. e Xerocostas, D. (1992). A new decomposition scheme of the urban public transport scheduling problem. In *Computer-Aided Transit Scheduling: Proceedings of the Fifth International Workshop*, pp. 407–425.
- Ribeiro, C. C. e Soumis, F. (1994). A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, 42:41–52.
- Saha, J. L. (1970). An algorithm for bus scheduling problems. *Operational Research Quarterly*, 21:463–474.
- Sampaio, R. B.; Sampaio, Y. e Sampaio, L. M. B. (2006). Eficiência de sistemas de transporte público no nordeste com análise envoltória de dados (DEA). *Revista Econômica do Nordeste*, 37(2):261–275.

- Santos, A. G. e Mateus, G. R. (2009). General hybrid column generation algorithm for crew scheduling problems using genetic algorithm. In *2009 IEEE Congress on Evolutionary Computation*, pp. 1799–1806.
- Shen, Y. e Kwan, R. S. K. (2001). Tabu search for driver scheduling. *Computer-Aided Scheduling of Public Transpor*, pp. 121–135.
- Silva, G. P. (2001). Uma metodologia baseada na técnica de geração de arcos para o problema de programação de veículos. Tese de doutorado. Escola Politécnica da USP, São Paulo - SP.
- Souza, M. J. F.; Silva, G. P. e Simões, E. M. L. (2007). Programação de ônibus urbano: Uma abordagem heurística. In *Transporte em transformação XI: Trabalhos vencedores do prêmio CNT de Produção Acadêmica 2006*, volume 1, pp. 39–57. Positiva.
- Tosini, E. e Vercellis, C. (1988). An interactive system for extra-urban vehicle and crew scheduling problems. In *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop*, pp. 41–53.
- Tsang, E. (1993). Foundations of constraint satisfaction. Academic press.
- Wren, A. (1972). Bus scheduling: an interactive computer method. *Transportation Planning and Technology*, 1:115–122.

O PROBLEMA DA ALOCAÇÃO DINÂMICA DE ESPAÇOS: APLICAÇÃO DAS METAHEURÍSTICAS GRASP E BUSCA TABU

Geiza Cristina da Silva¹

Departamento de Ciências Exatas e Aplicadas
Universidade Federal de Ouro Preto
e-mail: geiza@decea.ufop.br

Paulo Oswaldo Boaventura-Netto

Laura Bahiense

Programa de Engenharia de Produção, COPPE
Universidade Federal do Rio de Janeiro
e-mails: boaventu@pep.ufrj.br, laura@pep.ufrj.br

Luiz Satoru Ochi

Instituto de Computação
Universidade Federal Fluminense
e-mail: satoru@ic.uff.br

Resumo

Neste trabalho é abordado o Problema da Alocação Dinâmica de Espaços onde um projeto é dividido por um número de períodos consecutivos e, em cada um deles, uma quantidade de atividades é realizada. Os recursos necessários para as atividades devem estar disponíveis em espaços de trabalho e os recursos ociosos no período devem ser guardados em espaços de armazenamento. Este problema contém o Problema Quadrático de Alocação (PQA) como um caso particular e, portanto, pertence a classe NP-difícil. Neste contexto a dificuldade em encontrar soluções ótimas, mesmo para instâncias de média dimensão, justifica o uso de métodos heurísticos. Este trabalho propõe algoritmos de construção e heurísticas baseados no GRASP e Busca Tabu para a obtenção de soluções aproximadas. Os resultados computacionais obtidos mostram que os métodos propostos são competitivos na resolução das instâncias do problema existentes na literatura.

Palavras-chave: Metaheurísticas, Otimização Combinatória, Problema Quadrático de Alocação.

Abstract

This work is dedicated to the Dynamic Space Allocation Problem, where a project is divided in a number of consecutive periods, each of them associated to a number of activities. The resources required by the activities have to be available in the corresponding workspaces and those staying idle during a period have to be stored in depots. This problem contains the Quadratic Assignment Problem (QAP) as a particular case, which puts it in the NP-hard class. In this context, the difficulty to identify optimal solutions, even for instances of medium size, justify the use of heuristics. This work proposes construction algorithms and heuristics based on GRASP and Tabu Search for finding approximate solutions. Computational results show the proposed methods as being competitive with relation to the literature instances and the existing techniques.

Keywords: Metaheuristics, Combinatorial Optimization, Quadratic Assignment Problem.

¹ Agradecemos ao CNPq/(CT-Info e Universal), CAPES, FAPEMIG e FAPERJ pelo apoio a este trabalho.

1 Introdução

O Problema da Alocação Dinâmica de Espaços (PADE) é relativamente novo na literatura e foi inspirado na necessidade de se otimizar os custos de realocação dos recursos necessários para a realização de atividades em projetos, em princípio relacionados com o manejo de equipamentos e de outros recursos. Como custos de realocação são considerados, além da distância percorrida pelos recursos, o custo de preparação dos recursos para serem movimentados.

Um projeto é dividido em um número de períodos de tempo consecutivos e, em cada um deles, um determinado número de atividades deve ser realizado. Um recurso é chamado de *recurso necessário* quando necessário para a realização de uma atividade em um dado período e *ocioso*, quando não for usado por uma das atividades do período. O local de realização do projeto é dividido em *espaços de trabalho* e *espaços de armazenamento*, ou *depósitos*. As atividades e seus recursos são associados a espaços de trabalho e os recursos ociosos, a espaços de armazenamento. O conjunto de períodos, com suas atividades e os respectivos recursos necessários e ociosos, constitui a *agenda* do projeto. O objetivo do problema é alocar os recursos de maneira que a distância total percorrida por estes seja minimizada durante os períodos de tempo associados ao projeto.

A Figura 1 apresenta um pequeno problema-exemplo do PADE. A agenda, consta de 4 períodos, 5 atividades e 9 recursos. Na segunda tabela da Figura 1, é apresentado o *layout* dos espaços: na primeira linha, são localizados 3 espaços de trabalho (E_1 a E_3) e, na segunda, 3 espaços de armazenamento (E_4 a E_6). A métrica de Manhattan é utilizada na determinação da distância entre os espaços. Por exemplo, a distância entre os espaços E_1 e E_2 é de 1 unidade, enquanto entre E_1 e E_6 ela é de 3 unidades. A capacidade de cada espaço é de 3 recursos.

Períodos	Atividades (rec. necessários)	Recursos ociosos
1	$A_1(6,7)$ $A_2(1,5)$	2,3,4,8,9
2	$A_2(1,5)$ $A_3(3,4)$	2,6,7,8,9
3	$A_4(2,8)$	1,3,4,5,6,7,9
4	$A_4(2,8)$ $A_5(6,9)$	1,3,4,5,7

E_1	E_2	E_3
E_4	E_5	E_6

Figura 1: Uma pequena instância do PADE: agenda e layout dos espaços

Para que uma solução seja considerada viável, as seguintes condições devem ser consideradas:

- A cada período, somente uma atividade pode ser realizada em um espaço de trabalho;
- uma atividade deve ser associada a um único espaço de trabalho, isto é, se uma atividade for realizada em períodos consecutivos, ela deve permanecer no espaço de trabalho onde se encontra;
- a capacidade do espaço de trabalho ao qual uma atividade é associada deve ser suficiente para guardar seus respectivos recursos;
- a capacidade dos depósitos deve ser respeitada.

A Figura 2 ilustra uma solução ótima para o problema-exemplo da Figura 1, onde a distância total percorrida pelos recursos é de 12 unidades.

$A_2(1,5)$		$A_1(6,7)$	$A_2(1,5)$	$A_3(3,4)$	
2,8	3,4	9	2,8		6,7,9
Período 1			Período 2		
$A_4(2,8)$			$A_4(2,8)$		$A_5(6,9)$
1,5	3,4	6,7,9	1,5	3,4	7
Período 3			Período 4		

Figura 2: Uma solução para a instância exemplo

O presente trabalho utiliza uma formulação matemática do problema para obter soluções ótimas, ou limites inferiores, para algumas instâncias da literatura. Em termos de algoritmos, são propostas heurísticas baseados nos conceitos do GRASP e da Busca Tabu. A seção 2 apresenta uma revisão bibliográfica. A seção 3 é dedicada à formulação matemática do PADE. A seção 4 descreve os algoritmos propostos. Os resultados computacionais obtidos são mostrados na seção 5, após a qual se apresentam as conclusões.

2 Literatura

O PADE, embora seja aplicável em muitas situações práticas, ainda não apresenta muitas contribuições na literatura. Ele foi originalmente proposto por McKendall Jr. *et al.* (2005) que o associaram ao problema de minimizar o custo do transporte de recursos dados – equipamentos, ferramentas e peças de reposição, entre outros – na ocorrência de desligamentos planejados em centrais de energia elétrica. Na literatura, esses custos têm sido diretamente associados às distâncias percorridas pelos recursos. Os autores também apresentam uma formulação matemática (apresentada na Seção 3) e duas heurísticas do tipo *simulated annealing*.

Em McKendall Jr. e Jaramillo (2006) foram apresentadas cinco heurísticas de construção e uma busca tabu (Tabu I). Neste trabalho, uma nova restrição é incorporada ao problema: recursos que se apresentam ociosos em períodos de tempo consecutivos devem permanecer em um mesmo depósito, o que é razoável em termos de custo e diminui o espaço de busca da solução. Tabu I mantém duas listas tabu: uma para os movimentos de atividades e outra, para os movimentos de recursos ociosos. A cada iteração, o melhor movimento é escolhido e a solução resultante torna-se corrente para a próxima iteração. Através de testes comparativos dos métodos anteriores com os propostos, os autores concluíram que a combinação das heurísticas de construção com a busca tabu proporcionou melhor performance, tanto em termos de qualidade das soluções como em tempo de computação.

Em McKendall Jr. (2008), um modelo matemático modificado e três diferentes heurísticas baseadas em busca tabu são propostos. A primeira heurística (Tabu II) é uma busca tabu simples que se diferencia de Tabu I somente na maneira como os recursos ociosos são explorados. Em Tabu II os recursos ociosos são movimentados individualmente. Na segunda heurística deste trabalho (Tabu III) as listas tabu têm seus tamanhos recalculados a cada iteração e consideram ainda estratégias de intensificação e diversificação. A terceira, Tabu IV, é semelhante a Tabu II diferindo em como o movimento a ser executado na solução corrente é escolhido: todos os possíveis movimentos são avaliados e ordenados; os melhores M movimentos são considerados e selecionados para fazer parte de uma *lista de movimentos candidatos* (LMC). A partir daí é considerado o primeiro movimento de LMC. Este é aceito com uma probabilidade p . Se o movimento for aceito, a solução resultante passa a ser corrente para a próxima iteração. Caso contrário, o próximo movimento é considerado e aceito com uma probabilidade p , sendo este passo repetido até que um movimento seja selecionado. Se nenhum dos movimento de LMC for aceito, o melhor movimento da lista é escolhido. A metodologia apresentada é comparada com os algoritmos da literatura e os resultados reportados apontam a segunda heurística como o melhor método dentre os que figuraram na comparação.

Apesar de sua definição recente, o PADE pode ser correlacionado com outros problemas muito estudados na área de Pesquisa Operacional. A tarefa de associar atividades a espaços de trabalho corresponde ao Problema Quadrático de Alocação (PQA) (Koopmans, 1957; Loiola. *et al.*, 2007). No entanto, o PADE propõe que a alocação de atividades deva ocorrer em múltiplos períodos de tempo, recaindo exatamente na definição do Problema Dinâmico de *Layout* de Facilidades (PDFL) (Rosenblatt, 1986). Já o sub-problema do PADE, de associação dos recursos ociosos a depósitos, é associado ao Problema Quadrático de Alocação Generalizado (PQAG) (Lee e Ma, 2005). Este sub-problema é de complexidade não menor que a do PQAG uma vez que a associação dos recursos ociosos deve ocorrer em mais de um período.

3 Modelo Matemático

Na definição formal do PADE, os seguintes índices e parâmetros necessários devem ser considerados:

Índices:

- J = número total de atividades ($j = 1, 2, \dots, J$);
- R = número total de recursos ($r = 1, 2, \dots, R$);
- P = número total de períodos ($p = 1, 2, \dots, P$);
- N = número total de locações (espaços de trabalho e de armazenamento);
- L = conjunto de locações, $|L| = N$ ($l = 1, 2, \dots, N$);
- W = conjunto de espaços de trabalho, $W \subset L$ ($w \in W$);
- S = conjunto de depósitos, $S \subset L$ ($s \in S$) e $W \cup S = L$;
- R_j = conjunto de recursos necessários para realizar a atividade j ;
- I_p = conjunto de recursos ociosos no período p ;
- A_p = conjunto de atividades no período p .

Parâmetros:

- d_{kl} = distância entre as locações k e l ;
- C_s = capacidade do depósito s .

Deve-se considerar ainda a variável de decisão x_{prk} que se torna 1 se no período p , o recurso r é associado ao depósito k e 0 caso contrário; e a variável de decisão y_{jw} que assume valor 1 se a atividade j é realizada no espaço de trabalho w e 0 caso contrário. O modelo matemático proposto em McKendall *et al.* (2005) é definido como:

Minimizar:

$$(1) \quad \sum_{r=1}^R \sum_{k=1}^N \sum_{l=1}^N \sum_{p=1}^P d_{kl} x_{prk} x_{p+1rl}$$

Sujeito a:

$$(2) \quad \sum_{s \in S_j} x_{prs} = 1, \forall p \in P, \forall r \in I_p,$$

$$(3) \quad \sum_{r \in I_p} x_{prs} \leq C_s, \forall s \in S, \forall p \in P,$$

$$(4) \quad \sum_{w \in W} y_{jw} = 1, \forall j \in J,$$

$$(5) \quad \sum_{j \in A_p} y_{jw} \leq 1, \forall w \in W, \forall p \in P,$$

$$(6) \quad \sum_{r \in R_j} x_{prk} = |R_j| y_{jw}, \forall p \in P, \forall j \in A_p, \forall w \in W,$$

$$(7) \quad x_{prk} = 0 \text{ ou } 1, \forall p \in P, \forall r \in R, \forall k \in L,$$

$$(8) \quad y_{jw} = 0 \text{ ou } 1, \forall j \in J, \forall w \in W.$$

A função objetivo (1) minimiza a distância percorrida pelos recursos ao longo dos períodos em que o projeto é realizado. As restrições (2) garantem que todos os recursos ociosos em cada período sejam associados a um único depósito e as (3) asseguram que a capacidade dos depósitos seja respeitada em cada período de tempo. As restrições (4) e (5) garantem, respectivamente, que toda atividade seja associada a um único espaço de trabalho e que cada espaço de trabalho tenha no máximo uma atividade associada. As restrições (6) garantem que todos os recursos necessários à realização de uma atividade sejam associados ao mesmo espaço de trabalho ao qual a atividade tenha sido associada. Finalmente, as restrições (7) e (8) determinam que as variáveis de decisão são binárias.

Esta formulação claramente quadrática, tem sua forma linearizada substituindo-se a função objetivo (1) pelo termo (1'), onde $w_{p+1prkl}$ é uma variável de decisão binária. Além disso, as restrições 9 – 11 devem ser acrescentadas.

$$\begin{aligned}
 (1') \quad & \sum_{r=1}^R \sum_{k=1}^N \sum_{l=1}^N \sum_{p=1}^{P-1} d_{kl} w_{p+1prkl} \\
 (9) \quad & x_{prk} + x_{p+1rl} - 1 \leq w_{p+1prkl}, \forall p \in P, \forall r \in R, \forall k, l \in L, l \neq k, \\
 (10) \quad & x_{prk} + x_{p+1rl} \geq 2w_{p+1prkl}, \forall p \in P, \forall r \in R, \forall k, l \in L, l \neq k, \\
 (11) \quad & w_{p+1prkl} = 0 \text{ ou } 1, \forall p \in P, \forall r \in R, \forall k, l \in L, l \neq k.
 \end{aligned}$$

4 Algoritmos Propostos

Neste trabalho são propostos algoritmos construtivos e heurísticas usando o conceito de GRASP e Busca Tabu. A escolha destas metaheurísticas, em parte se deve aos bons resultados que as mesmas têm mostrado em diferentes problemas de otimização combinatória (Santos *et al.* 2005; Silva *et al.* 2007).

4.1 Algoritmos de Construção

São propostos dois algoritmos de construção, ambos obtendo uma solução a partir da execução de duas etapas: na primeira delas, ocorre a associação das atividades a espaços de trabalho e, em seguida, a associação dos recursos ociosos a depósitos. Os dois algoritmos propostos diferem um do outro na segunda etapa e são doravantes denotados por: Construção Proposta 1 (C1) e Construção Proposta 2 (C2).

Na primeira fase de C1 e C2, consideremos J , o número de atividades e W , o conjunto de espaços de trabalho.

Inicialmente, os espaços de trabalho são ordenados, a partir do espaço mais bem localizado, formando uma *lista inicial de candidatos* (LIC). O tamanho de LIC é, portanto, $|W|$. A LIC tem, então, os espaços de trabalho ordenados do mais bem localizado para o pior e para cada atividade os elementos de LIC que estão disponíveis são selecionados para compor a LRC.

A solução é então construída iterativamente, a cada iteração uma atividade a sendo associada a um espaço de trabalho. Para isto, é criada uma *lista restrita de candidatos* (LRC), um espaço de trabalho é escolhido aleatoriamente da LRC e associado a a . Na construção da LRC, a LIC é percorrida para que se verifique se, em todos os períodos onde a ocorre, o espaço de trabalho escolhido não é ocupado por alguma outra atividade. Se essa ocupação não ocorrer, ele será incluído na LRC. O tamanho da LRC é determinado por: $\min(n_{disp}, \alpha \times |W|)$, onde n_{disp} é o número de espaços de trabalho disponíveis e α é um parâmetro.

Na segunda etapa, C1 é a implementação da *Randomized Storage Policy* (RSP) (McKendall Jr. *et al.*, 2006) que associa cada recurso ocioso ao depósito mais próximo do espaço de trabalho onde a atividade que o requer estiver alocada, dando maior prioridade para os recursos que são usados

mais cedo e C2 é uma adaptação de RSP por considerar a associação de recursos feita no período anterior.

4.2 Estruturas de Vizinhança

Dada uma solução s , são utilizadas as estruturas de vizinhanças $NA(s)$, definida sobre a parte da solução que se refere à associação de atividades e $NR(s)$, que corresponde à parte dos recursos ociosos. Um movimento de troca em $NA(s)$ consiste em permutar os espaços de trabalho de duas atividades e um movimento de realocação, consiste em retirar uma atividade de um espaço de trabalho ao qual está associada e reassociá-la a outro espaço de trabalho até então vazio. Os possíveis movimentos de recursos ociosos em $NR(s)$ consistem em trocar os depósitos de dois (ou mais) recurso(s) em todos os períodos consecutivos nos quais este(s) permaneça(m) ocioso(s) e em remover um (ou mais) recurso(s) de um depósito em todos os períodos consecutivos no qual este(s) permaneça(m) ocioso(s) e realocá-lo(s) a um depósito até então vazio.

4.3 GRASP Proposto (GRASP)

Na fase de construção, GRASP obtém uma solução inicial por meio de C1.

Na fase de busca local, a cada iteração, são realizados todos os movimentos de atividades e de recursos ociosos na solução corrente, descritos em 4.2. O melhor movimento que represente uma melhora é executado e a solução obtida torna-se corrente para a próxima iteração. A busca termina quando a solução corrente não puder mais ser melhorada.

O algoritmo GRASP retorna a melhor solução obtida em um número dado de iterações.

4.4 Busca Tabu Proposta (Tabu)

Tabu executa os seguintes passos até um dado número de iterações sem melhora: no início, o algoritmo avalia todos os possíveis movimentos de atividades. Para cada um deles, a heurística RSP é executada para realocar os recursos ociosos de acordo com a nova matriz de atividades. O movimento não tabu que, quando executado seguido de RSP, gerar a melhor solução é feito e a solução correspondente torna-se corrente para a próxima iteração. O critério de aspiração permite que um movimento tabu possa ser realizado se a solução gerada por sua execução seguida de RSP tiver menor custo que a melhor solução. Uma lista tabu de tamanho dinâmico é utilizada para armazenar os movimentos recentes de atividades. O tempo que um movimento é considerado tabu é determinado pelo tamanho da lista tabu. Inicialmente e a cada η iterações sem melhora, o tamanho da lista é escolhido aleatoriamente dentro de um limite inferior (li) e limite superior (ls).

A maior diferença entre a Tabu aqui proposta e as heurísticas apresentadas na literatura está nos movimentos usados (somente de atividades) enquanto as buscas tabu propostas em McKendall e Jaramillo (2006) e McKendall (2008) executam o melhor movimento (de atividade ou de recursos ociosos) e no uso da heurística RSP para a geração de uma solução parcial de recursos ociosos. Deve-se notar ainda que tanto as buscas tabu da literatura, como a aqui apresentada, podem ser consideradas implementações simples. Entretanto, conforme mostrado na próxima seção, o método proposto é capaz de um desempenho na média superior para as instâncias do problema testadas. Este mesmo resultado, a partir de uma busca tabu simples, foi obtido no estudo de Santos *et al.* (2005), mostrando com isso, que para obter bons resultados a heurística não precisa necessariamente ser muito sofisticada.

5 Resultados Computacionais

Nos testes computacionais foram utilizadas as instâncias da literatura propostas em McKendall *et al.* (2005). Todos os parâmetros envolvidos nos métodos propostos foram empiricamente definidos através de testes preliminares. Os testes foram realizados em um Computador Intel

Artigo apresentado e publicado nos Anais do XLI Simpósio Brasileiro de Pesquisa Operacional (XLI SBPO) - 2009

Quad Core 2 com 2394 Mhz e 3278 Mb de memória RAM usando o sistema operacional Linux Ubuntu na versão 2.6.24-19.

A Tabela 1 apresenta os resultados para as instâncias com 6 e 12 locações e a Tabela 2, com 20 e 32 locações, totalizando 96 instâncias testadas. Foram computadas para cada instância o melhor valor de função objetivo obtido de um total de 10 execuções dos algoritmos Tabu e GRASP (Colunas Tabu e GRASP), o valor médio das execuções (arredondado para inteiro superior) (M_Tabu e M_GRASP), o tempo de processamento, em segundos, da melhor solução (Tempo(s)) e o tempo médio das execuções (M_Tempo(s)). Reportamos também, para cada método proposto, a diferença percentual entre o valor da melhor solução obtida pelo método e o valor da melhor solução conhecida (Dif.) e a diferença percentual entre a média de valores de soluções obtidas pelo método e o valor da melhor solução conhecida (M_Dif.).

Inst.	Lit.	Tabu	Dif.	Tempo(s)	M_Tabu	M_Dif.	M_Tempo(s)	GRASP	Dif.	Tempo(s)	M_GRASP	M_Dif.	M_Tempo(s)
P1	16	16	0,0	-	16	0,0	-	16	0,0	-	16	0,0	-
P2	25	26	3,8	-	26	3,8	-	26	3,8	-	26	3,8	-
P3	18	18	0,0	-	18	0,0	-	18	0,0	-	18	0,0	-
P4	25	26	3,8	-	26	3,8	-	26	3,8	-	26	3,8	-
P5	16	16	0,0	-	16	0,0	-	16	0,0	-	16	0,0	-
P6	27	27	0,0	-	27	0,0	-	27	0,0	-	27	0,0	-
P7	16	16	0,0	-	16	0,0	-	16	0,0	-	16	0,0	-
P8	31	31	0,0	-	31	0,0	-	31	0,0	-	32	3,1	-
P9	25	25	0,0	-	25	0,0	-	25	0,0	0,1	25	0,0	0,1
P10	46	46	0,0	-	46	0,0	-	46	0,0	0,1	47	2,1	0,1
P11	32	32	0,0	-	32	0,0	-	32	0,0	0,1	32	0,0	0,1
P12	41	43	4,7	-	43	4,7	-	43	4,7	0,1	44	6,8	0,1
P13	28	28	0,0	-	28	0,0	-	28	0,0	0,1	28	0,0	0,1
P14	45	45	0,0	-	45	0,0	-	45	0,0	0,1	46	2,2	0,1
P15	35	35	0,0	-	35	0,0	-	35	0,0	0,1	36	2,8	0,1
P16	49	49	0,0	-	49	0,0	-	49	0,0	0,1	49	0,0	0,1
P17	35	35	0,0	-	35	0,0	0,1	35	0,0	0,2	36	2,8	0,2
P18	60	62	3,2	0,1	62	3,2	-	62	3,2	0,2	64	6,3	0,2
P19	46	46	0,0	0,1	46	0,0	0,1	46	0,0	0,2	46	0,0	0,2
P20	60	63	4,8	-	63	4,8	-	63	4,8	0,1	64	6,3	0,1
P21	46	48	4,2	0,1	48	4,2	0,1	48	4,2	0,2	49	6,1	0,2
P22	67	67	0,0	0,0	67	0,0	-	67	0,0	0,2	68	1,5	0,2
P23	55	56	1,8	0,1	56	1,8	0,1	56	1,8	0,2	57	3,5	0,2
P24	74	74	0,0	-	74	0,0	-	74	0,0	0,2	74	0,0	0,2
P25	*30	31	0,0	0,1	31	3,2	0,2	31	3,2	0,5	31	0,0	0,5
P26	*42	43	0,0	0,1	43	2,3	0,2	43	2,3	0,6	45	4,4	0,6
P27	43	43	0,0	0,1	43	0,0	0,1	43	0,0	0,5	43	0,0	0,6
P28	*54	55	0,0	0,1	55	0,0	0,1	55	1,8	0,5	56	1,8	0,5
P29	29	29	0,0	0,1	29	0,0	0,1	29	0,0	0,4	29	0,0	0,4
P30	50	49	-2,0	0,1	49	-2,0	0,1	49	-2,0	0,6	50	0,0	0,5
P31	42	42	0,0	0,1	42	0,0	0,2	43	2,3	0,5	43	2,3	0,6
P32	*66	69	0,0	0,1	69	0,0	0,1	69	4,3	0,4	70	1,4	0,4
P33	53	52	-1,9	0,7	53	0,0	0,4	54	1,9	1,8	56	5,4	1,8
P34	72	72	0,0	0,4	72	0,0	0,3	74	2,7	1,9	77	6,5	2,0
P35	68	73	6,8	0,5	73	6,8	0,4	73	6,8	1,9	75	9,3	1,9
P36	90	95	5,2	0,3	95	5,3	0,3	95	5,2	1,4	97	7,2	1,4
P37	47	48	2,1	0,4	48	2,1	0,4	48	2,1	1,9	51	7,8	1,9
P38	77	83	7,2	0,5	83	7,2	0,4	83	7,2	1,8	88	12,5	1,8
P39	67	69	2,9	0,4	69	2,9	0,4	69	2,9	1,9	73	8,2	2,0
P40	104	108	3,7	0,2	108	3,7	0,3	108	3,7	1,4	111	6,3	1,4
P41	78	78	0,0	1,3	78	0,0	0,9	80	2,5	4,3	84	7,1	4,3
P42	104	102	-2,0	0,7	103	-1,0	0,8	102	-2,0	4,2	111	6,3	4,3
P43	110	110	0,0	1,1	110	0,0	0,9	110	0,0	4,2	113	2,7	4,3
P44	137	140	2,1	0,4	140	2,1	0,5	140	2,1	3,0	145	5,5	3,0
P45	66	66	0,0	0,7	66	0,0	0,9	68	2,9	4,8	76	13,2	4,7
P46	111	116	4,3	0,6	116	4,3	0,7	118	5,9	4,1	122	9,0	4,1
P47	111	115	3,5	1,1	115	3,5	1,0	115	3,5	4,2	120	7,5	4,1
P48	169	171	1,2	0,5	171	1,2	0,5	172	1,7	3,1	175	3,4	3,1

Tabela 1: Resultados para as instâncias de 6 e 12 locações

Artigo apresentado e publicado nos Anais do XLI Simpósio Brasileiro de Pesquisa Operacional (XLI SBPO) - 2009

Inst.	Mck	Tabu	Dif.	Tempo(s)	M_Tabu	M_Dif.	M_Tempo(s)	GRASP	Dif	Tempo(s)	M_GRASP	M_Dif.	Tempo(s)
P49	45	44	-2,3	1,5	44	-2,3	1,3	44	-2,3	6,3	44	-2,3	6,4
P50	63	60	-5,0	1,8	60	-5,0	1,3	60	-5,0	7,0	64	1,6	7,1
P51	55	54	-1,9	1,1	54	-1,9	1,1	56	1,8	7,0	58	5,2	6,9
P52	98	89	-10,1	0,7	89	-10,1	0,9	92	-6,5	4,5	94	-4,3	4,6
P53	49	47	-4,3	2,2	47	-4,3	1,4	47	-4,3	6,5	48	-2,1	6,5
P54	67	62	-8,1	1,6	63	-6,3	1,4	64	-4,7	7,3	68	1,5	7,4
P55	63	60	-5,0	2,2	60	-5,0	1,4	62	-1,6	7,8	66	4,5	7,5
P56	97	89	-9,0	0,8	90	-7,8	1,1	89	-9,0	6,4	97	0,0	6,4
P57	67	66	-1,5	5,2	67	0,0	4,0	68	1,5	23,4	70	4,3	23,3
P58	106	101	-5,0	3,1	103	-2,9	3,2	106	0,0	21,4	111	4,5	21,7
P59	101	93	-8,6	4,1	95	-6,3	3,6	100	-1,0	20,2	107	5,6	20,6
P60	159	153	-3,9	1,4	154	-3,2	1,8	156	-1,9	13,3	163	2,5	13,2
P61	82	74	-10,8	6,0	76	-7,9	4,6	79	-3,8	28,3	82	0,0	28,0
P62	129	112	-15,2	5,2	114	-13,2	5,0	119	-8,4	28,2	129	0,0	27,6
P63	121	118	-2,5	6,0	120	-0,8	4,2	127	4,7	25,7	132	8,3	25,9
P64	190	175	-8,6	2,0	177	-7,3	3,0	182	-4,4	20,2	191	0,5	20,6
P65	105	101	-4,0	10,7	103	-1,9	10,1	109	3,7	51,4	116	9,5	51,6
P66	156	150	-4,0	7,3	154	-1,3	6,1	163	4,3	47,9	169	7,7	48,0
P67	157	145	-8,3	6,5	148	-6,1	7,3	150	-4,7	45,1	161	2,5	45,8
P68	234	218	-7,3	3,3	222	-5,4	3,7	231	-1,3	30,4	238	1,7	29,9
P69	112	114	1,8	12,1	118	5,1	10,7	121	7,4	65,7	129	13,2	64,8
P70	178	166	-7,2	9,1	170	-4,7	9,7	183	2,7	65,0	189	5,8	64,5
P71	170	162	-4,9	2,2	167	-1,8	13,2	182	6,6	62,8	192	11,5	63,7
P72	265	251	-5,6	6,3	255	-3,9	6,5	259	-2,3	47,6	280	5,4	47,9
P73	74	71	-4,2	8,0	71	-4,2	7,0	71	-4,2	50,8	71	-4,2	50,7
P74	97	90	-7,8	7,0	93	-4,3	8,4	95	-2,1	61,7	99	2,0	61,3
P75	110	102	-7,8	12,7	104	-5,8	9,6	109	-0,9	6,0	114	3,5	67,3
P76	155	144	-7,6	5,2	148	-4,7	6,7	155	0,0	48,7	164	5,5	48,9
P77	73	70	-4,3	9,1	70	-4,3	7,9	70	-4,3	51,7	72	-1,4	51,3
P78	101	92	-9,8	23,9	96	-5,2	10,6	102	1,0	65,8	105	3,8	66,4
P79	110	99	-11,1	8,4	101	-8,9	10,1	108	-1,9	85,9	115	4,3	84,3
P80	175	163	-7,4	7,0	166	-5,4	8,3	174	-0,6	67,3	181	3,3	65,5
P81	119	109	-9,2	44,9	113	-5,3	29,4	124	4,0	229,8	129	7,8	234,1
P82	176	161	-9,3	38,9	167	-5,4	24,5	179	1,7	215,8	188	6,4	221,8
P83	192	185	-3,8	30,1	193	0,5	31,8	206	6,8	235,4	220	12,7	234,4
P84	282	268	-5,2	21,2	278	-1,4	18,5	285	1,1	168,7	304	7,2	167,0
P85	125	117	-6,8	39,8	121	-3,3	34,2	135	7,4	234,9	141	11,3	237,3
P86	192	181	-6,1	30,1	187	-2,7	28,6	202	4,6	228,4	215	10,7	230,1
P87	193	180	-7,2	41,5	186	-3,8	32,2	201	4,0	262,0	214	9,8	263,9
P88	302	275	-9,8	44,8	282	-7,1	30,5	297	-1,7	202,2	315	4,1	201,3
P89	171	162	-5,6	58,9	167	-2,4	67,7	188	9,0	612,7	198	13,6	618,2
P90	262	251	-4,4	81,9	259	-1,2	57,7	273	4,0	553,6	290	9,7	556,5
P91	284	271	-4,8	68,1	283	-0,4	54,2	303	6,3	568,7	319	11,0	572,2
P92	395	388	-1,8	20,5	399	1,0	25,4	409	3,4	402,5	421	6,2	381,0
P93	189	178	-6,2	81,3	186	-1,6	73,6	196	3,6	652,5	209	9,6	636,6
P94	281	270	-4,1	87,6	281	0,0	74,4	301	6,6	613,0	317	11,4	603,9
P95	318	298	-6,7	69,7	312	-1,9	76,8	344	7,6	663,8	362	12,2	668,8
P96	464	437	-6,2	43,1	458	-1,3	41,769	478	2,9	456,7	502	7,6	471,3

Tabela 2: Resultados para as instâncias de 20 e 32 locações

Procurou-se resolver exatamente instâncias de pequeno e médio porte, através do modelo discutido na Seção 3. Dos 96 problemas-testes disponíveis, a literatura reporta os limites inferiores P1 até P24 e P27, num total de 25 problemas. A formulação matemática foi processada no CPLEX 11 e além dos ótimos já conhecidos, outras instâncias foram resolvidas de forma exata, a saber: P25, P26, P28, P29, P31 e P32. Os valores destacados em *itálico*, na Tabela 1, indicam o valor ótimo. O uso da técnica exata foi abandonado a partir do problema P32 visto que neste o tempo computacional já era demasiadamente elevado. Surpreendentemente, nas instâncias P25, P26, P28 e P32, os melhores valores reportados na literatura (McKendall, 2008) são menores que o ótimo, o que mostra equívocos destes resultados da literatura. Estes valores equivocados estão destacados com um asterisco (*). Em relação ao desempenho dos algoritmos propostos, Tabu chegou ao ótimo em 24 e a heurística GRASP, em 23 instâncias diferentes. Verificamos ainda que nas instâncias onde o ótimo não foi alcançado, os métodos estiveram no máximo a 4,8% do ótimo.

Para as 64 instâncias cujo ótimo é desconhecido (P30 e P33 - P96), as heurísticas propostas foram comparadas com os melhores resultados da literatura. Tabu obteve melhores soluções em 50 instâncias mostrando uma redução média de -2,5% e a maior redução (- 15,2%) foi obtida na instância P62. O algoritmo foi capaz de manter o resultado da literatura em 5 instâncias e, nas demais, a maior diferença percentual entre Tabu e o melhor da literatura foi de 7,2%. GRASP foi

capaz de melhorar o resultado conhecido 24 vezes, empatou 3 vezes com a literatura e chegou a reduzir o melhor custo da instância P56 em -9,0%. No entanto, forneceu, em média, soluções de custo 3,0% maior do que os da literatura.

As Figuras 3 e 4 mostram as diferenças percentuais entre o melhor resultado da literatura e a média dos resultados obtidos pelos algoritmos propostos para as instâncias das Tabelas 1 e 2.

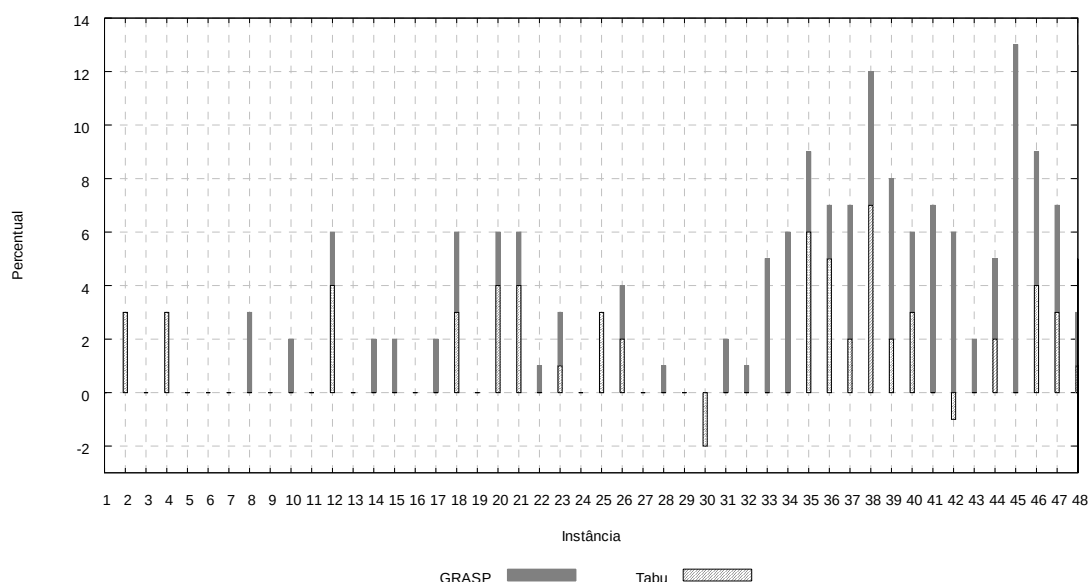


Figura 3: Gráfico de diferenças percentuais para as instâncias de 6 e 12 locações

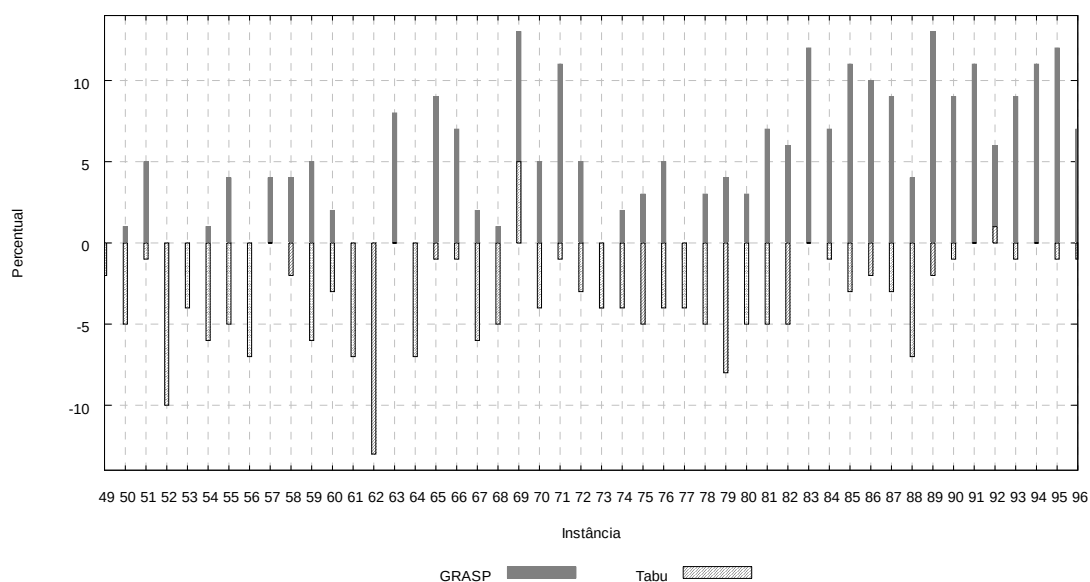


Figura 4: Gráfico de diferenças percentuais para as instâncias de 20 e 32 locações

Em tempo computacional, o algoritmo Tabu se mostrou mais eficiente que o GRASP para os parâmetros utilizados. Não foi feita a comparação de tempo entre os algoritmos da literatura e propostos, visto que os algoritmos foram executados em máquinas de poder computacional diferente.

Através dos resultados empíricos apresentados nota-se que Tabu mostra eficiência maior em instâncias com 20 e 32 locações, o que, na prática, mostra que o algoritmo proposto é menos míope que a versão da literatura. Em relação ao método GRASP, pela maneira simples que foi

implementado, acredita-se que este pode ser melhorado a partir de um estudo mais amplo do fator de determinação de gulosidade/aleatoriedade do algoritmo de construção (valor de α) cuja escolha é determinante para o bom desempenho da metaheurística, como pode ser visto em (Resende, 2008). Neste mesmo sentido, acredita-se que seria interessante a aplicação dos conceitos de GRASP Reativo que tem sido frequentemente utilizado com sucesso para outros problemas de semelhante complexidade, como em Silva et al. (2007).

6 Conclusões

Neste trabalho, o Problema da Alocação Dinâmica de Espaços é estudado. Este problema é relativamente novo na literatura mas, embora de definição recente, tem sua importância justificada por poder ser aplicado a casos nos quais mover um recurso é uma tarefa difícil ou dispendiosa. Além disso, o PADE pode ser correlacionado a um dos problemas mais examinados da área, o PQA, além do PDFL e o PQAG.

Dois algoritmos baseados em GRASP e Busca Tabu são propostos e os resultados são comparados com os resultados ótimos (quando estes estão disponíveis) e com os melhores resultados da literatura. Nesta comparação nota-se que a heurística baseada em busca tabu é, na média, capaz de produzir soluções de custos melhores que os demais em questão, para o conjunto de instâncias testadas.

Outra contribuição do trabalho também está na correção dos limites inferiores das instâncias P25, P26, P28 e P32, apresentados incorretamente na literatura, além de resolver instâncias da literatura que estavam em aberto.

Referências

- Koopmans, T. C. e Beckmann, M.** (1957). Assignment problems and the location of economic activities. *Econometrica*, 25, 53-76.
- Lee, C-G, e Ma, Z.,** The generalized quadratic assignment problem, *Working paper*, U. of Toronto, Dept. of Mechanical and Industrial Engineering, 2005.
- Loiola, E. M., Abreu, N. M. M. A., Boaventura-Netto, P. O., Hahn, P. e Querido, T.M.** (2007), A survey for the quadratic assignment problem, *European Journal of Operational Research*, 176, 657–690.
- McKendall Jr., A.** (2008), Improved tabu search heuristic for the dynamic space allocation problem, *Computers & Operations Research*, 35, 3347–3359.
- McKendall Jr., A. e Jaramillo, J.** (2006). A tabu search heuristic for the dynamic space allocation problem, *Computers & Operations Research*, 33, 768–789.
- McKendall Jr., A., Noble, J. e Klein, C.** (2005). Simulated annealing heuristics for managing resources during planned outages at electric power plants, *Computers & Operations Research*, 32, 107–125.
- Resende, M. G. C.** (2008). Metaheuristic hybridization with Greedy Randomized Adaptive Search Procedures. *TutORials in Operations Research*, Zhi-Long Chen and S. Raghavan (Eds.), 295-319, INFORMS, 2008
- Rosenblatt, M. J.** (1986), The dynamics of plant layout, *Management Science*, 32, 76–86.
- Santos, H. G., Ochi, L. S. e Souza, M. J. F.** (2005). A Tabu Search Heuristic with Efficient Diversification Strategies for the Class/Teacher Timetabling Problem, *The ACM Journal of Experimental Algorithmics*, 10, 1-16.
- Silva, G. C., de Andrade, M. R. Q., Ochi, L. S., Martins, S. L. e Plastino, A.** (2007). New heuristics for the maximum diversity problem, *Journal of Heuristics*, 13, 315-336.

**OTIMIZAÇÃO DO FLUXO DE PRODUTOS DE
UMA EMPRESA MINERADORA**

TÚLIO ÂNGELO MACHADO TOFFOLO

ORIENTADOR: GERALDO ROBSON MATEUS

OTIMIZAÇÃO DO FLUXO DE PRODUTOS DE UMA EMPRESA MINERADORA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

Maio de 2009

© 2009, Túlio Ângelo Machado Toffolo.
Todos os direitos reservados.

Toffolo, Túlio Ângelo Machado

Otimização do Fluxo de Produtos de uma Empresa
Mineradora / Túlio Ângelo Machado Toffolo. — Belo
Horizonte, 2009

xviii, 77 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas
Gerais

Orientador: Geraldo Robson Mateus

1. Pesquisa Operacional. 2. Otimização. 3. Mineração.
I. Título.

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={nome do arquivo}`
ao comando `\ppgccufmg`.

Aos meus pais.

Agradecimentos

Agradeço, em primeiro lugar, a Deus, pela saúde e oportunidades que me fizeram chegar até aqui.

Aos meus orientadores, Prof. Geraldo Robson Mateus e Prof. Marcone Jamilson Freitas Souza, pelos quais tenho grande admiração, gostaria de prestar minha sincera gratidão pelos ensinamentos, e, sobretudo, pela compreensão e amizade demonstrados nos momentos mais difíceis.

Aos meus pais Rodrigo e Renata, agradeço pelo amor e carinho, não só durante a realização deste trabalho, mas por toda a minha vida. Eu definitivamente não seria ninguém sem meus pais, as pessoas mais importantes da minha vida.

Ao meu irmão e melhor amigo Rodrigo, pelo eterno apoio, amizade e paciência.

Ao DCC da UFMG e seus professores, que não mediram esforços para me ajudar em todo o mestrado e que com grande sabedoria me conduziram por todo este tempo para que eu chegasse até aqui.

Agradeço a todos meus amigos que me encorajaram nos momentos mais difíceis desta trajetória, em especial à Rinat, aos companheiros da Optilog, Fred, Reinaldo, José Maria e Cecília e aos colegas do LaPo. Obrigado queridos amigos!

Agradeço também à minha avó, tios, primos e todos os demais que direta ou indiretamente contribuíram para a minha formação tanto como profissional quanto como pessoa.

Resumo

É notória a importância do setor de extração mineral para o Brasil, tendo este gerado apenas em Minas Gerais uma receita de cerca de R\$ 29 bilhões em 2008. Neste contexto o desenvolvimento de tecnologias que aprimorem este setor é de grande relevância. Periodicamente, as mineradoras devem tomar decisões relacionadas à produção e transporte dos minérios, tomando como base suas capacidades logísticas e produtivas, bem como demandas dos mercados interno e externo (exportação). Estas decisões geram um plano de fluxo dos produtos, que consiste em determinar o curso dos minérios provenientes das diferentes minas, desde a produção até a venda, com o objetivo de maximizar o atendimento às metas de qualidade dos clientes e, ao mesmo tempo, otimizar toda a cadeia logística. Neste processo, a escolha do minério a ser utilizado na composição dos produtos e uma complexa malha de transporte que inclui mineriodutos, correias de longa distância, terminais ferroviários, rodoviários e portuários devem ser considerados.

Esta dissertação propõe algoritmos para o Problema do Planejamento do Fluxo dos Produtos (FP), que engloba alguns problemas clássicos da literatura de forma integrada, tais como Mistura de Minérios, Planejamento de Transporte e Planejamento e Sequenciamento da Produção. O FP foi tratado em diferentes horizontes de planejamento: anual, trimestral, mensal e dia-a-dia. Um modelo multi-objetivo baseado em programação linear por metas foi proposto, sendo capaz de resolver apenas instâncias dos horizontes anual e mensal em tempo aceitável. Para tratar as instâncias do dia-a-dia, foram desenvolvidos algoritmos heurísticos baseados nas técnicas *relax-and-fix*, GRASP e ILS. As diferentes metodologias foram validadas através de testes em instâncias geradas a partir da realidade de uma empresa mineradora brasileira de grande porte.

Abstract

It is well-known that mineral extraction industry is very important to Brazil, having raised R\$ 29 billions only in Minas Gerais, during the year of 2008. In this context, the development of technologies that can help these industries is of great relevance. Periodically, the mining companies make decisions related to their production and the transportation of the minerals, considering their logistic and productivity capacities as well as the demands of the market. Such decisions generate a plan of products flow, which consists in determining the flow of the minerals produced in the different mines, from production to distribution, having the goals to minimize the quality gap between the demanded and the delivered product and to optimize the logistics chain. In this process, the quality of the minerals to be used in the composition of the products to be sold and a complex transport system that includes mine pipes, long-distance belts, roads, railroads terminals and harbors must be considered.

This dissertation proposes algorithms to deal with the Product Flow Problem, which includes some classic problems in the literature, such as the Ore Blending Problem, Transport Planning Problem and Sequence Production Planning Problem. The problem was considered in different planning horizons: annual, monthly and day-by-day. A multiobjective model based on goal programming was proposed for the problem, being able to solve only annual and monthly term instances in acceptable time. To deal with the day-by-day instances, heuristics algorithms based on relax-and-fix, GRASP and ILS techniques were developed. The different methodologies were validated through tests on instances based on the reality of a huge Brazilian mining company.

Sumário

1	Introdução	3
1.1	Motivação	3
1.2	Objetivos	5
1.2.1	Objetivo Geral	5
1.2.2	Objetivos Específicos	5
1.3	Organização da Dissertação	6
2	Trabalhos Relacionados	7
2.1	Mistura de Minérios	7
2.2	Planejamento de Transporte	11
2.2.1	Transporte Ferroviário de Minério da MBR	12
2.3	Planejamento e Sequenciamento da Produção	21
2.3.1	Planejamento de Produção e Vendas da MBR	23
2.4	Outros Trabalhos	29
3	Descrição do Problema	31
3.1	O Processo Produtivo da Empresa Mineradora	31
3.1.1	Fase de Lavra	32
3.1.2	Fase de Beneficiamento	34
3.1.3	Estocagem	34
3.1.4	Movimentação	35

3.2	O Fluxo de Produtos da Empresa Mineradora	36
4	Formulação Matemática	41
4.1	Otimização Multiobjetivo	41
4.1.1	Conjunto Pareto-ótimo	42
4.1.2	Formas de Resolução	44
4.2	Modelo Matemático	45
5	Algoritmos Heurísticos	55
5.1	Conceitos Iniciais	55
5.1.1	Relax-And-Fix	56
5.1.2	GRASP	57
5.1.3	ILS	60
5.1.4	VND	62
5.2	Heurística <i>Relax-And-Fix</i> aplicada ao FP	63
5.3	Heurística GRASP-ILS aplicada ao FP	64
5.3.1	Representação de uma Solução	64
5.3.2	Geração de Soluções Iniciais	65
5.3.3	Estruturas de Vizinhança	66
5.3.4	Busca Local	67
5.3.5	Avaliação Multiobjetivo	68
6	Resultados Obtidos	69
7	Considerações Finais	71
	Referências Bibliográficas	73

Lista de Figuras

2.1	Sistema produtivo da MBR.	13
2.2	Representação do problema de transporte de minérios na MBR.	14
3.1	Processo de Produção de ROM	33
3.2	Processo de Geração do Produto Primário	34
3.3	Representação do FP	37
3.4	Representação do FP	38
3.5	Entrada e Saída do FP.	40
4.1	Espaço de soluções unidimensional (otimização mono-objetivo)	41
4.2	Espaço de soluções multidimensional (otimização multiobjetivo)	42
4.3	Conjunto de soluções Pareto-ótimas	43

Lista de Tabelas

Lista de Algoritmos

5.1	Procedimento <i>RelaxAndFix</i> Básico	57
5.2	Procedimento <i>GRASP</i> Básico	58
5.3	Procedimento <i>Construcao</i> (α) do GRASP	59
5.4	Procedimento <i>BuscaLocal</i> ($N(\cdot), s_0$) do GRASP	59
5.5	Procedimento ILS Básico	61
5.6	VND Básico	62
5.7	Procedimento de Avaliação de Soluções	68

Capítulo 1

Introdução

Este capítulo tem por objetivo apresentar a motivação que impulsionou a realização deste trabalho (seção 1.1) bem como expor, em linhas gerais, os objetivos que se pretendeu alcançar ao planejá-lo (seção 1.2). Por fim, o conteúdo de cada capítulo é descrito sucintamente (seção 1.3).

1.1 Motivação

Uma das maiores reservas de minério do mundo está localizada no Brasil. De acordo com o IBRAM [Instituto Brasileiro de Mineração, 2009], a produção mineral brasileira em 2008 foi de cerca de R\$ 54 bilhões e mais de 160 mil postos de trabalho foram gerados, sendo que Minas Gerais foi o maior produtor de bens minerais do Brasil (53,9%). Assim, o desenvolvimento de tecnologias que permitam melhorar atividades relacionadas à extração, transporte ou venda de minérios é de grande interesse no cenário nacional, com destaque para o estado de Minas Gerais.

Grandes mineradoras possuem diversas minas com diferentes capacidades, instalações de tratamento de minério (ITM), áreas de estocagem, usinas de pelotização, além de uma complexa malha de transporte que pode incluir mineriodutos, correias de longa distância, terminais ferroviários, rodoviários e portuários. Periodicamente, as mineradoras devem tomar decisões relacionadas com a produção e transporte dos minérios,

tomando como base suas capacidades produtivas e demandas dos mercados interno e externo (exportação). Estas decisões geram um plano de Fluxo de Produtos (FP). No caso de uma empresa mineradora, o FP consiste em um plano de médio prazo com o objetivo de determinar o fluxo dos minérios produzidos, desde a extração até a venda. Cada minério possui diferentes características físicas e químicas, e deve contribuir com uma qualidade apropriada para que o produto final de venda esteja o mais próximo possível das metas previamente definidas. O planejamento do FP envolve os custos com deslocamento do minério e o atendimento às demandas de massa e qualidade sujeitos a restrições de produção e escoamento. Assim, cabe ao planejador decidir quais minérios extrair e beneficiar, a quantidade, a forma como se dará o transporte, e quando estes deverão ser utilizados para formar os produtos de venda.

O FP responde a importantes questões:

- A capacidade atual de escoamento é suficiente para atendimento à demanda?
- Levando em conta massa (quantidade) e qualidade, a produção da mineradora é capaz de atender às demandas? Se não, quais demandas podem ser atendidas de forma satisfatória?
- Se a demanda for menor do que a produção, então em que, onde e de quanto a produção deve ser reduzida?

Existem hoje diversas aplicações de Pesquisa Operacional na indústria da mineração, mas poucas tratam o fluxo dos minérios considerando quantidade e qualidade de forma integrada. Os poucos que realizam este tratamento [Alves et al., 2007] ignoram restrições operacionais que influenciam de forma significativa todo o processo. No que diz respeito à resolução do FP, a literatura é ainda mais escassa, sendo que os trabalhos encontrados apresentam modelos que ignoram restrições operacionais que simplesmente inviabilizam sua utilização pela maioria das empresas brasileiras.

O planejamento do FP é uma tarefa complexa. A forma como ele é resolvido atualmente por algumas empresas, sem o amparo de nenhum modelo de otimização, resulta em soluções de qualidade questionável. Como o número de variáveis é elevado,

e devido à natureza combinatória do problema, é praticamente impossível ao olhar humano detectar as melhores soluções. Na realidade, detectar uma única solução viável já é por si só uma tarefa árdua, que requer vários dias de dedicação por parte dos profissionais responsáveis pelo planejamento. Além disso, pela maneira como o FP vem sendo resolvido, é difícil estudar quantidade e qualidade ao mesmo tempo. Por estes motivos, modelos de otimização capazes de encontrar soluções de boa qualidade para o FP são muito desejados pelas empresas mineradoras.

1.2 Objetivos

1.2.1 Objetivo Geral

O presente trabalho tem por objetivo geral o desenvolvimento de sistemas computacionais que possibilitem uma maior eficiência na tomada de decisão do planejamento do Fluxo de Produtos de uma empresa mineradora. Para tanto, foram desenvolvidos modelos de otimização baseados em programação matemática e em técnicas heurísticas.

1.2.2 Objetivos Específicos

- Estudar o problema do planejamento do Fluxo de Produtos em uma empresa mineradora.
- Avaliar diversos modelos de otimização propostos na literatura, reunindo-os em modelos mais amplos, de forma a contemplar os requisitos típicos de uma mineradora.
- Elaborar um modelo matemático de otimização para o Problema do Fluxo de Produtos de uma empresa mineradora, buscando metas de qualidade e quantidade e levando em consideração as restrições de capacidade e operacionais de produção e escoamento.

- Aplicar as metaheurísticas *Greedy Randomized Adaptative Search Procedure* (GRASP) e *Iterated Local Search* (ILS) na resolução do problema.
- Desenvolver uma metodologia heurística híbrida *relax-and-fix* para o problema abordado.
- Testar as metodologias desenvolvidas.

1.3 Organização da Dissertação

Esta dissertação está organizada como se segue. No capítulo 2 são apresentados trabalhos e problemas correlatos.

O capítulo 3 apresenta o processo produtivo de uma empresa mineradora e detalha o problema abordado por esta dissertação.

O capítulo 4, por sua vez, destina-se a apresentar o modelo de programação inteira multi-objetivo desenvolvido e detalhes de sua implementação.

No capítulo 5 as metodologias heurísticas *relax-and-fix*, GRASP e ILS, bem como suas aplicações no problema, são discutidas em detalhes.

O capítulo 6 detalha os cenários de teste aos quais os algoritmos de otimização foram submetidos, apresentando e analisando os resultados obtidos.

Por fim, no capítulo 7 são apresentadas as conclusões a que se chegou mediante a análise dos resultados e as principais ideias para trabalhos futuros, bem como algumas considerações finais.

Capítulo 2

Trabalhos Relacionados

O Problema do Fluxo de Produtos (FP) de uma empresa mineradora engloba alguns problemas clássicos da literatura de forma integrada, tais como Mistura de Minérios, Planejamento de Transporte e Planejamento e Sequenciamento da Produção. Estes problemas são apresentados e discutidos neste capítulo nas seções 2.1, ?? e 2.3, respectivamente.

Além destes problemas, a descrição de outros trabalhos de otimização e simulação aplicados à mineração encontrados na literatura é feita na Seção 2.4.

2.1 Mistura de Minérios

O Problema da Mistura ou Blendagem de Minérios consiste em determinar a quantidade de cada minério, proveniente de um conjunto de frentes ou pilhas, que deve ser blendada para formar um produto final de venda com características que atendam as exigências de um determinado cliente. Os minérios extraídos possuem características diferentes, tais como o custo de lavra, o teor de determinado elemento químico ou o percentual de minério em determinada faixa granulométrica. Assim, ao se blendar os minérios é necessário atentar às proporções escolhidas, para que a mistura atenda as metas de quantidade e qualidade requeridas. No entanto, dada a grande variabilidade dos minérios normalmente encontrados nas minas, geralmente é impossível atingir as metas

estabelecidas. Por esta razão, são criados limites de tolerância para cada um dos parâmetros de controle.

O Problema da Mistura de Minérios costuma ser erroneamente confundido com o Problema da Homogeneização de Minério. O termo *blendagem* (mistura) diz respeito a uma mistura, em proporções definidas, de minérios de características diferentes com o objetivo de se obter uma massa com características específicas. O termo *homogeneização*, por outro lado, se refere ao manuseio ou mistura de quantidades de minério com o objetivo de se obter um conjunto que tenha composição ou características uniformes [Moraes et al., 2005]. Um estudo detalhado dos conceitos de *blendagem* e *homogeneização* pode ser encontrado em Schofield [1980].

Diversos autores tratam do Problema da Mistura de Minérios. Chanda e Dagdelen [1995] afirmam que a base para qualquer modelo de *blendagem* é a informação precisa sobre a composição do material a ser utilizado, informação esta que muitas vezes é apenas aproximada. Neste trabalho, os autores aplicaram um modelo de programação linear por metas [Charnes e Cooper, 1961], alegando que modelos de programação linear clássicos permitem que apenas um objetivo seja analisado por vez. Tal limitação foi contornada através da transformação de restrições em metas, que quando não cumpridas são penalizadas na função objetivo [Lee, 1972].

Everett [2001] identificou quatro possíveis estágios para formação dos minérios de venda por meio da *blendagem*:

1. Seleção de blocos com composição específica a serem lavrados em uma dada mina;
2. Determinação da sequência de trens contendo minérios de composições específicas para o transporte das minas até o porto;
3. Seleção da pilha (no porto) na qual cada trem deve descarregar seu conteúdo;
4. Seleção de pilhas com composições específicas a serem utilizadas no carregamento de um determinado navio.

Levando em conta o primeiro estágio, Costa [2005] apresentou um modelo de programação linear por metas na resolução de um problema de mistura de minérios em que

a quantidade de minério retirada em uma frente é múltipla da capacidade da caçamba do equipamento de carga em operação naquela frente. Para apresentar o modelo, seja a seguinte notação:

M : Conjunto de frentes de minério;

S : Conjunto dos parâmetros de controle analisados no produto final;

t_{ik} : Valor do parâmetro de controle $k \in S$ na frente de minério $i \in M$ (%);

tr_k : Valor requerido para o parâmetro de controle $k \in S$ no produto final (%);

tl_k : Valor mínimo admissível para o parâmetro de controle $k \in S$ no produto final (%);

tu_k : Valor máximo admissível para o parâmetro de controle $k \in S$ no produto final (%);

Pr : Meta de produção (t);

Pl : Quantidade mínima a ser produzida (t);

Pu : Quantidade máxima a ser produzida (t);

α_k^- : Penalidade por desvio negativo para o parâmetro de controle $k \in S$ no produto final;

α_k^+ : Penalidade por desvio positivo para o parâmetro de controle $k \in S$ no produto final;

β^- : Penalidade por desvio negativo da produção;

β^+ : Penalidade por desvio positivo da produção;

Ql_i : Quantidade mínima a ser utilizada da frente $i \in M$ (t);

Qu_i : Quantidade máxima a ser utilizada da frente $i \in M$ (t);

Cc_i : Capacidade da caçamba da carregadeira alocada à frente $i \in M$ (t).

Considerando as seguintes variáveis de decisão:

x_i : Quantidade de minério a ser utilizada da frente $i \in M$ (t);

N_i : O número de caçambadas a serem efetuadas na frente $i \in M$;

d_k^- : Desvio negativo do parâmetro de controle $k \in S$ no produto final;

d_k^+ : Desvio positivo do parâmetro de controle $k \in S$ no produto final;

P^- : Desvio negativo da produção requerida (t);

P^+ : Desvio positivo da produção requerida (t).

tem-se, pelas equações (2.1) - (2.14), o modelo de programação por metas relativo ao problema da mistura de minérios.

Minimizar

$$\sum_{k \in S} \alpha_k^- d_k^- + \sum_{k \in S} \alpha_k^+ d_k^+ + \beta^- P^- + \beta^+ P^+ \quad (2.1)$$

Sujeito a:

$$\sum_{i \in M} (t_{ik} - tu_k) x_i \leq 0 \quad \forall k \in S \quad (2.2)$$

$$\sum_{i \in M} (t_{ik} - tl_k) x_i \geq 0 \quad \forall k \in S \quad (2.3)$$

$$\sum_{i \in M} (t_{ik} - tr_k) x_i + d_k^- - d_k^+ = 0 \quad \forall k \in S \quad (2.4)$$

$$\sum_{i \in M} x_i + P^- - P^+ = Pr \quad (2.5)$$

$$\sum_{i \in M} x_i \leq Pu \quad (2.6)$$

$$\sum_{i \in M} x_i \geq Pl \quad (2.7)$$

$$x_i \leq Qu_i \quad \forall i \in M \quad (2.8)$$

$$x_i \geq Ql_i \quad \forall i \in M \quad (2.9)$$

$$x_i - Cc_i N_i = 0 \quad \forall i \in M \quad (2.10)$$

$$N_i \in \mathbb{Z}^+ \quad \forall i \in M \quad (2.11)$$

$$x_i \geq 0 \quad \forall i \in M \quad (2.12)$$

$$d_k^+, d_k^- \geq 0 \quad \forall k \in S \quad (2.13)$$

$$P^+, P^- \geq 0 \quad (2.14)$$

Neste modelo observam-se as restrições clássicas do problema de mistura. As restrições (2.2) e (2.3) definem limites máximos e mínimos para os parâmetros de controle, enquanto as restrições (2.6) e (2.7) limitam a quantidade máxima e a mínima de minério no produto final. As restrições (2.8) e (2.9) garantem o atendimento às quantidades máximas e mínimas a serem utilizadas de cada frente de lavra, definidas pelo planejador. As restrições (2.10) definem que a quantidade de minério utilizada de uma frente de lavra é obtida multiplicando-se a capacidade da caçamba da pá-carregadeira pelo número de caçambadas. As restrições (2.11) determinam que o número de caçambadas a serem efetuadas em uma frente de lavra deve ser um valor inteiro positivo.

As restrições (2.4) e (2.5), utilizadas por Chanda e Dagdelen [1995] e desenvolvidas a partir do método de programação por metas, visam medir os desvios de qualidade e produção, respectivamente, em relação aos valores requeridos. As restrições (2.12), (2.13) e (2.14) impedem que valores negativos sejam aceitos para as variáveis de decisão. Com a inclusão das restrições (2.4) e (2.5), a função de avaliação mono-objetivo (2.1) trata, de forma ponderada, dois objetivos distintos: a minimização dos desvios de produção e qualidade em relação aos valores requeridos.

2.2 Planejamento de Transporte

O planejamento do transporte ou distribuição define o escoamento dos produtos aos diferentes clientes, por meio de diversos modais de transporte e fazendo uso de centros de estocagem. Sabe-se que cerca de 40% do custo do minério de ferro é dado por operações logísticas. Assim, estas operações possuem impacto relevante.

Alguns trabalhos da literatura abordam problemas de Planejamento de Transporte na mineração. Lu et al. [2005] apresentam uma pesquisa sobre a logística de importação de minério de ferro na China. Neste trabalho, os autores consideram uma cadeia de suprimentos que envolve portos, ferrovias e siderúrgicas. Eles propõem um sistema de controle de pedidos, transporte e estoque, o qual utiliza um modelo baseado em programação inteira mista. Neste modelo, a função objetivo procura minimizar o custo total de circulação de minério na cadeia de suprimentos.

Mateus et al. [1994] tratam um problema de planejamento do transporte ferroviário de uma mineradora brasileira, através de um modelo de programação inteira mista. Este modelo é detalhado na Seção 2.2.1.

2.2.1 Transporte Ferroviário de Minério da MBR

Mateus et al. [1994] apresentam um modelo para planejamento do transporte ferroviário de minério de ferro aplicado ao cenário da empresa MBR (Minerações Brasileiras Reunidas), a qual foi adquirida pela Vale em 2006. O valor do transporte ferroviário, no contexto da época, representava cerca de 40% a 45% do preço final do produto exportado. A Figura 2.1 apresenta o sistema produtivo da extinta MBR.

O modelo desenvolvido tem como fronteiras os estoques de produção das instalações de tratamento de minério e o embarque de navios, não incluindo aspectos de produção, para os quais o planejamento de lavra já oferece suporte adequado. O problema de transporte de minérios visa assim minimizar os custos de movimentação de produtos, dado um conjunto de ofertas e demandas no tempo, um conjunto de capacidades de estoques espacialmente distribuídos, capacidades de equipamentos para remoção, capacidades de carregamento de terminais ferroviários, necessidade de blendagem de minérios provenientes de diferentes minas e taxas de transferência alcançáveis no transporte ponto a ponto. O planejamento do transporte é considerado dinamicamente, com os estoques de minérios nas minas, pátios e porto representando os elementos de ligação entre os vários estágios de tempo. O problema é representado por um grafo direcionado, em que os nós representam as minas, os terminais de carregamento ferroviário



Figura 2.1. Sistema produtivo da antiga empresa MBR, composto por minas, correias, rodovias, ferrovias e porto. Fonte: MBR.

e o terminal portuário, enquanto as arestas representam as ligações entre as diversas instalações, com custos e capacidades associados.

Para o modelo de Mateus et al. [1994], sejam os seguintes dados de entrada:

M : Conjunto de minas (nós de oferta);

E : Conjunto de pontos de estocagem em Minas Gerais;

C : Conjunto de terminais de carregamento ferroviário;

F : Conjunto de pontos de início de ferrovia;

G : Conjunto de áreas operacionais de terminais portuários;

H : Conjunto de pátios de estocagem em terminais portuários;

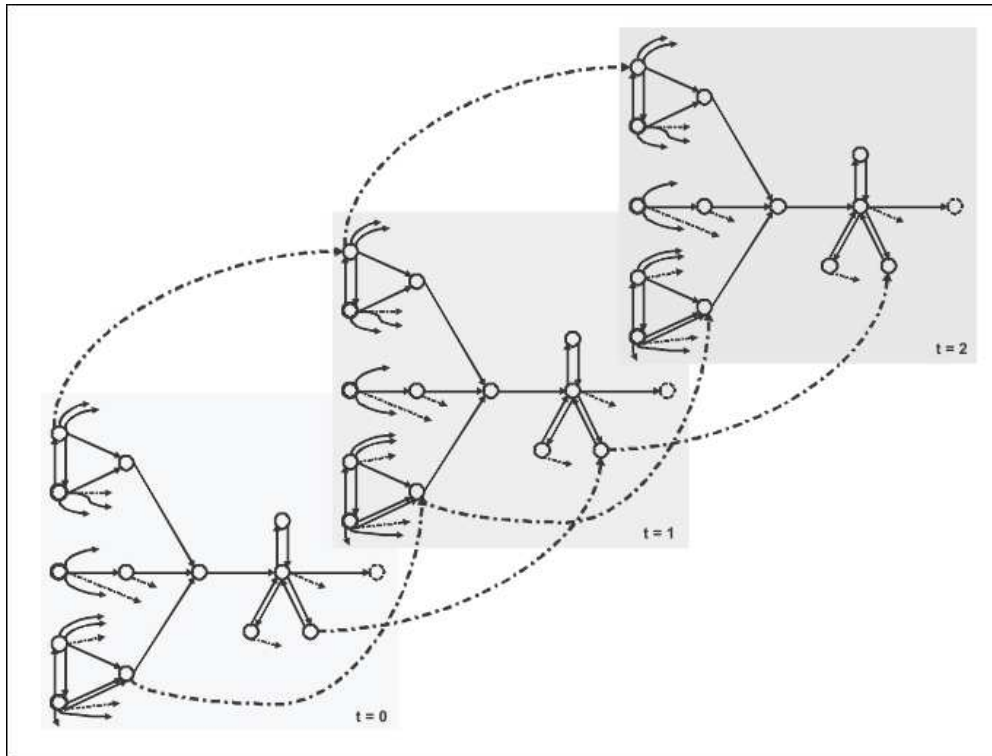


Figura 2.2. Representação do problema de transporte de minérios na MBR, com os estoques representando a ligação entre os diversos estágios de tempo no planejamento.

D : Conjunto de navios (nós de demanda);

N : Conjunto de nós do problema, onde $N = M \cup E \cup C \cup F \cup G \cup H \cup D$;

A^M : Conjunto de arcos (i, j) tais que $i \in (M \cup E \cup C), j \in (M \cup E \cup C \cup F)$, representando os fluxos entre centros de oferta (minas) e ferrovias;

A^E : Conjunto de arcos (i, j) tais que $i \in (F \cup G \cup H), j \in (G \cup H \cup D)$, representando os fluxos entre o transporte ferroviário e o carregamento portuário;

A : Conjunto de arcos do problema, onde $A = (A^M \cup A^E)$;

X : Conjunto de minérios originais que são vendidos em sua forma pura;

Y : Conjunto de produtos blendados;

Z : Conjunto de minérios originais que formam os produtos blendados;

P : Conjunto de todos os produtos, onde $P = (X \cup Y \cup Z)$;

P^b : Subconjunto de minérios ($P^b \subseteq Z$) que formam o produto blendado $b \in Y$;

T : Conjunto de estágios de tempo de planejamento;

c_{ij}^t : Custo unitário do fluxo de qualquer produto¹ no arco $(i, j) \in A$ no instante de tempo $t \in T$;

o_i^{pt} : Oferta do produto $p \in (X \cup Z)$ associada à mina $i \in M$, no instante de tempo $t \in T$;

d_i^{pt} : Demanda pelo produto $p \in (X \cup Y)$ associada ao navio $i \in D$, no instante de tempo $t \in T$;

ρ_p^{bt} : Fração do minério original $p \in P^b$ na composição do produto blendado $b \in Y$, no instante de tempo $t \in T$, tal que $\sum_{p \in P^b} \rho_p^{bt} = 1, \forall b \in Y, \forall t \in T$;

l_{ij}^{pt} : Limite inferior sobre o fluxo do produto $p \in X$ no arco $(i, j) \in A$, no instante de tempo $t \in T$;

L_{ij}^{pt} : Limite superior sobre o fluxo do produto $p \in X$ no arco $(i, j) \in A$, no instante de tempo $t \in T$;

v_{ij}^{pt} : Limite inferior sobre o fluxo do produto $p \in Y$ no arco $(i, j) \in A^E$, no instante de tempo $t \in T$;

V_{ij}^{pt} : Limite superior sobre o fluxo do produto $p \in Y$ no arco $(i, j) \in A^E$, no instante de tempo $t \in T$;

w_{ij}^{pt} : Limite inferior sobre o fluxo do produto $p \in Z$ no arco $(i, j) \in A^M$, no instante de tempo $t \in T$;

W_{ij}^{pt} : Limite superior sobre o fluxo do produto $p \in Z$ no arco $(i, j) \in A^M$, no instante de tempo $t \in T$;

¹Os autores assumem que todos os produtos têm os mesmos custos de transporte.

u_i^{pt} : Estoque mínimo do produto $p \in P$ no nó $i \in N$, ao final do estágio de tempo $t \in T$;

U_i^{pt} : Estoque máximo do produto $p \in P$ no nó $i \in N$, ao final do estágio de tempo $t \in T$;

$\tilde{u}_i^t$: Estoque mínimo total de produtos no nó $i \in N$, ao final do estágio de tempo $t \in T$;

$\tilde{U}_i^t$: Estoque máximo total de produtos (capacidade total de estocagem do sistema MBR, incluindo minas, porto e eventuais nós intermediários) no nó $i \in N$, ao final do estágio de tempo $t \in T$;

$\tilde{l}_{ij}^t$: Limite inferior sobre o fluxo total de produtos no arco $(i, j) \in A^M$, no instante de tempo $t \in T$;

$\tilde{L}_{ij}^t$: Limite superior (Capacidade total de movimentação entre os estoques das minas e o carregamento ferroviário) sobre o fluxo total de produtos no arco $(i, j) \in A^M$, no instante de tempo $t \in T$;

$\tilde{v}_{ij}^t$: Limite inferior sobre o fluxo total de produtos no arco $(i, j) \in A^E$, no instante de tempo $t \in T$;

$\tilde{V}_{ij}^t$: Limite superior (capacidade total de transporte ferroviário e da estocagem e carregamento portuário) sobre o fluxo total de produtos no arco $(i, j) \in A^E$, no instante de tempo $t \in T$.

Sejam ainda as seguintes variáveis de decisão:

x_{ij}^{pt} : Fluxo de minérios originais $p \in X$, que podem ser vendidos em sua forma pura, no arco $(i, j) \in A$, no instante de tempo $t \in T$;

y_{ij}^{pt} : Fluxo de produtos blendados $p \in Y$ por meio do arco $(i, j) \in A^E$, no instante de tempo $t \in T$;

z_{ij}^{pt} : Fluxo de insumos para blendagem $p \in Z$ por meio do arco $(i, j) \in A^M$, no instante de tempo $t \in T$;

e_i^{pt} : Estoque do produto $p \in P$ deixado no nó $i \in N$ ao final do estágio de tempo $t \in T$, sendo que nos instantes $t = 0$ e $t = T$ esse valor é uma constante definida a priori, representando, respectivamente, o estoque inicial disponível e o estoque mínimo final do produto $p \in P$ deixado no nó $i \in N$.

A formulação matemática de Mateus et al. [1994] para o problema de transporte de minérios é dada pelas equações (2.15) - (2.44). Nesta formulação, $S(\cdot)$ representa o fluxo de saída e $E(\cdot)$ o fluxo de entrada em um nó qualquer.

Minimizar

$$\sum_{t \in T} \sum_{p \in X} \sum_{(i,j) \in A} c_{ij}^t x_{ij}^{pt} + \sum_{t \in T} \sum_{p \in Y} \sum_{(i,j) \in A^E} c_{ij}^t y_{ij}^{pt} + \sum_{t \in T} \sum_{p \in Z} \sum_{(i,j) \in A^M} c_{ij}^t z_{ij}^{pt} \quad (2.15)$$

Sujeito a

$$\sum_{j \in S(m)} x_{mj}^{p1} - \sum_{k \in E(m)} x_{km}^{p1} = o_m^{p0} + e_m^{p0} - e_m^{p1} \quad \forall m \in M, \forall p \in X, t = 1 \quad (2.16)$$

$$\sum_{j \in S(m)} x_{mj}^{pt} - \sum_{k \in E(m)} x_{km}^{pt} \leq o_m^{pt} + e_m^{p,t-1} - e_m^{pt} \quad \forall m \in M, \forall p \in X, t = 2, \dots, T-1 \quad (2.17)$$

$$\sum_{j \in S(m)} x_{mj}^{pT} - \sum_{k \in E(m)} x_{km}^{pT} = o_m^{pT} + e_m^{p,T-1} - e_m^{pT} \quad \forall m \in M, \forall p \in X, t = T \quad (2.18)$$

$$\sum_{j \in S(i)} x_{ij}^{p1} - \sum_{k \in E(i)} x_{ki}^{p1} = e_i^{p0} - e_i^{p1} \quad \forall i \in (E \cup C), \forall p \in X, t = 1 \quad (2.19)$$

$$\sum_{j \in S(i)} x_{ij}^{pt} - \sum_{k \in E(i)} x_{ki}^{pt} = e_i^{p,t-1} - e_i^{pt} \quad \forall i \in (E \cup C), \forall p \in X, t = 2, \dots, T-1 \quad (2.20)$$

$$\sum_{j \in S(i)} x_{ij}^{pT} - \sum_{k \in E(i)} x_{ki}^{pT} = e_i^{p,T-1} - e_i^{pT} \quad \forall i \in (E \cup C), \forall p \in X, t = T \quad (2.21)$$

$$\sum_{j \in S(g)} x_{gj}^{p1} - \sum_{k \in E(g)} x_{kg}^{p1} = e_g^{p0} - e_g^{p1} \quad \forall g \in (G \cup H), \forall p \in X, t = 1 \quad (2.22)$$

$$\sum_{j \in S(g)} x_{gj}^{pt} - \sum_{k \in E(g)} x_{kg}^{pt} = e_g^{p,t-1} - e_g^{pt} \quad \forall g \in (G \cup H), \forall p \in X, t = 2, \dots, T-1 \quad (2.23)$$

$$\sum_{j \in S(g)} x_{gj}^{pT} - \sum_{k \in E(g)} x_{kg}^{pT} \geq e_g^{p,T-1} - e_g^{pT} \quad \forall g \in (G \cup H), \forall p \in X, t = T \quad (2.24)$$

$$\sum_{j \in S(f)} x_{fj}^{pt} - \sum_{k \in E(f)} x_{kf}^{pt} = 0 \quad \forall f \in F, \forall p \in X, \forall t \in T \quad (2.25)$$

$$- \sum_{i \in E(i)} x_{ki}^{pt} = d_i^{pt} \quad \forall i \in D, \forall p \in X, \forall t \in T \quad (2.26)$$

$$\sum_{j \in S(g)} y_{gj}^{p1} - \sum_{k \in E(g)} y_{kg}^{p1} = e_g^{p0} - e_g^{p1} \quad \forall g \in (G \cup H), \forall p \in Y, t = 1 \quad (2.27)$$

$$\sum_{j \in S(g)} y_{gj}^{pt} - \sum_{k \in E(g)} y_{kg}^{pt} = e_g^{p,t-1} - e_g^{pt} \quad \forall g \in (G \cup H), \forall p \in Y, t = 2, \dots, T-1 \quad (2.28)$$

$$\sum_{j \in S(g)} y_{gj}^{pT} - \sum_{k \in E(g)} y_{kg}^{pT} \geq e_g^{p,T-1} - e_g^{pT} \quad \forall g \in (G \cup H), \forall p \in Y, t = T \quad (2.29)$$

$$- \sum_{k \in E(i)} y_{ki}^{pt} = d_i^{pt} \quad \forall i \in D, \forall p \in Y, \forall t \in T \quad (2.30)$$

$$\sum_{j \in S(m)} z_{mj}^{p1} - \sum_{k \in E(m)} z_{km}^{p1} = o_m^{p1} + e_m^{p0} - e_m^{p1} \quad \forall m \in M, \forall p \in Z, t = 1 \quad (2.31)$$

$$\sum_{j \in S(m)} z_{mj}^{pt} - \sum_{k \in E(m)} z_{km}^{pt} = o_m^{pt} + e_m^{p,t-1} - e_m^{pt} \quad \forall m \in M, \forall p \in Z, t = 2, \dots, T-1 \quad (2.32)$$

$$\sum_{j \in S(m)} z_{mj}^{pT} - \sum_{k \in E(m)} z_{km}^{pT} \leq o_m^{pT} - e_m^{pT} \quad \forall m \in M, \forall p \in Z, t = T \quad (2.33)$$

$$\sum_{j \in S(i)} z_{ij}^{p1} - \sum_{k \in E(i)} z_{ki}^{p1} = e_i^{p0} - e_i^{p1} \quad \forall i \in (E \cup C), \forall p \in Z, t = 1 \quad (2.34)$$

$$\sum_{j \in S(i)} z_{ij}^{pt} - \sum_{k \in E(i)} z_{ki}^{pt} = e_i^{p,t-1} - e_i^{pt} \quad \forall i \in (E \cup C), \forall p \in Z, t = 2, \dots, T-1 \quad (2.35)$$

$$\sum_{j \in S(i)} z_{ij}^{pT} - \sum_{k \in E(i)} z_{ki}^{pT} = e_i^{p,T-1} - e_i^{pT} \quad \forall i \in (E \cup C), \forall p \in Z, t = T \quad (2.36)$$

$$\sum_{j \in S(f)} y_{fj}^{bt} - \sum_{p \in P^b} \rho_p^{bt} \sum_{k \in E(f)} z_{kf}^{pt} = 0 \quad \forall f \in F, \forall b \in Y, \forall t \in T \quad (2.37)$$

$$l_{ij}^{pt} \leq x_{ij}^{pt} \leq L_{ij}^{pt} \quad \forall (i, j) \in A, \forall p \in X, \forall t \in T \quad (2.38)$$

$$v_{ij}^{pt} \leq y_{ij}^{pt} \leq V_{ij}^{pt} \quad \forall (i, j) \in A^E, \forall p \in Y, \forall t \in T \quad (2.39)$$

$$w_{ij}^{pt} \leq z_{ij}^{pt} \leq W_{ij}^{pt} \quad \forall (i, j) \in A^M, \forall p \in Z, \forall t \in T \quad (2.40)$$

$$u_i^{pt} \leq e_i^{pt} \leq U_i^{pt} \quad \forall i \in N, \forall p \in P, \forall t \in T \quad (2.41)$$

$$\tilde{l}_{ij}^t \leq \sum_{p \in X} x_{ij}^{pt} + \sum_{p \in Z} z_{ij}^{pt} \leq \tilde{L}_{ij}^t \quad \forall (i, j) \in A^M, \forall t \in T \quad (2.42)$$

$$\tilde{v}_{ij}^t \leq \sum_{p \in X} x_{ij}^{pt} + \sum_{p \in Y} y_{ij}^{pt} \leq \tilde{V}_{ij}^t \quad \forall (i, j) \in A^E, \forall t \in T \quad (2.43)$$

$$\tilde{u}_i^t \leq \sum_{p \in P} e_i^{pt} \leq \tilde{U}_i^t \quad \forall i \in N, \forall p \in P, \forall t \in T \quad (2.44)$$

A função objetivo (2.15) contabiliza o custo variável total associado ao fluxo dos vários produtos nos arcos. O primeiro termo avalia o custo variável associado ao fluxo de minério vendido em sua forma pura por meio de toda a rede de distribuição. O segundo termo, por sua vez, contabiliza apenas o custo variável associado ao fluxo de produtos blendados nos arcos entre o início da ferrovia, a área operacional dos terminais portuários, seus pátios de estocagem e os navios. O terceiro termo corresponde ao custo variável associado ao fluxo de minérios originais que participam da composição dos produtos blendados através dos arcos entre as minas, as áreas de estocagem em Minas Gerais, os terminais de carga de trens e o início da ferrovia.

O primeiro conjunto de restrições representa as equações de balanço de fluxo de minérios que são vendidos em sua forma pura. As restrições (2.16), (2.17) e (2.18) limitam a capacidade de oferta de minérios originais pelas minas, considerando sua capacidade de estocagem. As restrições (2.19), (2.20) e (2.21) garantem o balanço

de fluxo de minérios originais nos nós de estocagem e carregamento ferroviário. As restrições (2.22), (2.23) e (2.24) garantem o balanço de fluxo de minérios originais nos nós de estocagem e carregamento portuário, enquanto as restrições (2.25) garantem o balanço de fluxo de minérios originais nos nós de ferrovia, onde não há estocagem de produtos (nós de transbordo). As restrições (2.26) asseguram o atendimento da demanda dos navios por produtos originais.

O segundo conjunto de restrições representa as equações de balanço de fluxo de produtos blendados. As restrições (2.27), (2.28) e (2.29) garantem o balanço de fluxo de produtos blendados nos nós de estocagem e carregamento portuário, enquanto as restrições (2.30) garantem o atendimento da demanda dos navios por produtos blendados.

O terceiro conjunto de restrições representa as equações de balanço de fluxo de minérios que participam da composição de produtos blendados. As restrições (2.31), (2.32) e (2.33) limitam a capacidade de oferta pelas minas de minérios que participam da composição de produtos blendados, considerando sua capacidade de estocagem. As restrições (2.34), (2.35) e (2.36) garantem o balanço de fluxo de minérios que compõem produtos blendados nos nós de estocagem e carregamento ferroviário.

As restrições (2.37) integram o segundo e o terceiro grupos de restrições descritos acima, definindo a participação de cada minério original na composição dos vários produtos blendados. Além disso, essas restrições modelam tanto a demanda por minérios originais que compõem produtos blendados quanto a capacidade de oferta de produtos blendados propriamente dita.

O quarto grupo de restrições representa os limites sobre o valor individual do fluxo de produtos através dos arcos. As restrições (2.38), (2.39) e (2.40) limitam o fluxo de minérios vendidos em sua forma pura, o fluxo de produtos blendados e o fluxo de minérios originais que compõem produtos blendados, respectivamente. As restrições (2.41) estabelecem limites para os níveis de estoque de cada produto em cada um dos nós do problema, ao final de cada instante de tempo.

O quinto grupo de restrições representa os limites para o fluxo total através dos arcos, não fazendo distinção entre os vários tipos de produtos, mas definindo a capaci-

dade total do fluxo de produtos pelos arcos do problema. As restrições (2.42) limitam o fluxo total de minérios (aqueles vendidos em sua forma pura e aqueles que participam da composição de produtos blendados) desde as minas até o início da ferrovia, enquanto as restrições (2.43), limitam o fluxo destes minérios após a blendagem desde o início da ferrovia até os navios. As restrições (2.44) impõem limites para os níveis totais de estoque de todos os produtos em cada um dos nós do problema, ao final de cada instante de tempo.

Os autores utilizam uma estratégia de decomposição baseada em relaxação Lagrangeana para reduzir a dimensão do problema e o tempo de solução. Tal estratégia é coerente com os procedimentos de gerenciamento com informações descentralizadas. Na prática, de acordo com Mateus et al. [1994], o problema como um todo pode ser dividido em subproblemas menores, sem perda da otimalidade global.

2.3 Planejamento e Sequenciamento da Produção

O planejamento da produção consiste em determinar a quantidade de itens a ser produzida em uma ou várias máquinas em cada período ao longo de um horizonte de tempo finito, de modo a atender uma certa demanda, sujeito a limitações de capacidade. Também deve determinar os níveis de estoque e os recursos necessários para implementar tal plano, que geralmente tem como objetivo minimizar custos e/ou maximizar o lucro. Quando o problema tem dimensão muito grande, técnicas de agregação são comumente utilizadas (por exemplo, agrupar produtos parecidos como itens únicos). Diversos trabalhos que tratam da otimização do planejamento da produção são encontrados na literatura, com aplicações nas mais diversas áreas.

Segundo Arenales et al. [2007], problemas na área de produção podem ser classificados em três níveis hierárquicos: estratégico, tático e operacional. O nível mais alto é o estratégico, em que as decisões são de longo prazo e envolvem altos investimentos. Esse nível trata da escolha e do projeto do processo, relacionados ao arranjo de máquinas e outros equipamentos e com a determinação da capacidade destes, em função de uma demanda futura. O nível tático trata do planejamento das atividades, que consiste de

dois subníveis: o planejamento agregado da produção e o planejamento de quantidades de produção. Por fim, o nível operacional controla as atividades diárias baseando-se nas ordens de produção provenientes do nível tático.

Diversos trabalhos que tratam da otimização do planejamento da produção são encontrados na literatura. Paiva e Morabito [2006] baseiam-se em modelos clássicos de dimensionamento de lotes para representar um sistema de produção de açúcar, álcool e melão, que inclui decisões da etapa agrícola, das fases de corte, carregamento e transporte de cana e, principalmente, decisões de moagem, escolha do processo produtivo e estoque dos produtos finais. As decisões são tomadas em períodos semanais e o horizonte de planejamento são as semanas de safra.

Junqueira e Morabito [2006] propõem um modelo de programação linear para apoiar as decisões do planejamento tático da produção, estocagem e transporte de sementes de milho, de forma a minimizar os custos de produção, logísticos e fiscais, atendendo a restrições de programação da colheita, capacidade das plantas e demanda dos clientes.

Kimms et al. [2005] apresentam uma formulação matemática conjunta para programação da produção e dimensionamento de lotes aplicada a uma indústria de bebidas. Nela são considerados diversos aspectos como capacidade disponível limitada, custos de armazenamento, custos de produção, custos e tempos de troca dependentes da sequência e um conjunto de máquinas paralelas entre outros.

Aires et al. [2005] apresentam um modelo matemático de programação linear inteira mista aplicado a um problema real de programação da produção de curto prazo de gasolina de uma refinaria responsável pelo abastecimento do mercado da Grande São Paulo.

Ferreira et al. [2005] propõem um modelo de otimização para auxiliar a tomada de decisões no planejamento e controle da produção em fábricas de refrigerantes, mais especificamente no que se refere ao dimensionamento e sequenciamento da produção. O modelo matemático proposto considera várias máquinas, os estágios de envase e xaroparia, tempos e custos de troca de refrigerantes nas linhas e tempos de troca de xaropes nos tanques (dependentes do sequenciamento da produção), capacidade limitada das linhas de produção e dos tanques, entre outros fatores. Para definir o

sequenciamento dos itens, os períodos são divididos em subperíodos e é permitida a produção de apenas um refrigerante por subperíodo. O critério de otimização é a minimização dos custos de estoque, atraso e troca de refrigerantes.

2.3.1 Planejamento de Produção e Vendas da MBR

Alves et al. [2007] abordam o problema do planejamento trimestral de produção e vendas de uma empresa mineradora através de um modelo de programação linear por metas. O objetivo é minimizar a diferença entre a qualidade da demanda especificada e a qualidade obtida. Para isso, são utilizados pesos para os parâmetros de cada produto de venda. O modelo considera restrições de capacidade dos terminais de carga, e inclui facilidades para o usuário, tal como atingir uma meta, a qual é modelada como uma restrição do modelo. Não fazem parte do escopo do trabalho, entretanto, a definição das rotas dos produtos primários. Além disso, algumas restrições operacionais, como a quantidade mínima de carregamento dos trens e a participação mínima de um produto primário na composição de um produto final, não são consideradas.

O modelo de Alves et al. [2007] é dado pelas equações (2.45) - (2.61). Para maior clareza, a notação utilizada pelos autores foi substituída pela notação a seguir:

P : conjunto de produtos primários provenientes das ITMs;

F : conjunto de produtos finais destinados à venda;

S : conjunto de parâmetros de qualidade analisados nos produtos finais;

C : conjunto de terminais de carga;

F : subconjunto dos terminais de carga, $F \subseteq C$, que aplicam sobre o produto carregado um fator de manuseio.

B : conjunto de possíveis misturas (blendagens) (i, j) entre produtos primários $(i \in P)$ e produtos finais $(j \in F)$;

T : conjunto de trimestres, $T = \{1, 2, 3, 4\}$;

wdd_j : peso, na função objetivo, do desvio no atendimento da demanda do produto final $j \in F$;

wdt_{jk} : peso, na função objetivo, do desvio no atendimento do teor do parâmetro $k \in S$ no produto final $j \in F$;

o_i^E : quantidade, em toneladas, de produto primário $i \in P$ disponível no estoque inicial;

o_{it}^P : capacidade de produção, em toneladas, do produto primário $i \in P$ no trimestre $t \in T$;

pr_{jt} : demanda, em toneladas, do produto final $j \in F$ no trimestre $t \in T$;

e_i : quantidade, em toneladas, de minério que deve ser eliminada do produto primário $i \in P$, isto é, a quantidade mínima fixada pelo planejador que deve ser utilizada;

b_{ijl}^E : quantidade, em toneladas, de minério armazenado no estoque inicial do produto primário $i \in P$ que deve ser blendada no produto final $j \in F$ no trimestre $l \in T$ (assume valor -1 quando não há quantidade fixada);

b_{ijl}^P : quantidade, em toneladas, de minério produzido no trimestre $t \in T$ do produto primário $i \in P$ que deve ser blendada no produto final $j \in F$ no trimestre $l \in T$ (assume valor -1 quando não há quantidade fixada);

tc_i : terminal de carga, $tc_i \in C \forall i \in P$, que transporta o produto primário $i \in P$;

cap_{ct} : capacidade de carregamento do terminal ferroviário de carga $c \in C$ no trimestre $l \in T$;

q_{ik}^E : teor do parâmetro $k \in S$ do produto primário $i \in P$ disponível no estoque inicial;

q_{ikt}^P : teor do parâmetro $k \in S$ no produto primário $i \in P$ produzido no trimestre $t \in T$;

qf_{ik}^E : teor do parâmetro $k \in S$ no produto primário $i \in P$ disponível no estoque inicial, aplicado o fator de manuseio mina-trem;

qf_{ikt}^P : teor do parâmetro $k \in S$ no produto primário $i \in P$ produzido no trimestre $t \in T$, aplicado o fator de manuseio mina-trem;

tr_{jkt} : meta, do teor típico, desejada do parâmetro $k \in S$ para o produto final $j \in F$ no trimestre $t \in T$;

tl_{jk} : limite inferior do parâmetro $k \in S$ para o produto final $j \in F$;

tu_{jk} : limite superior do parâmetro $k \in S$ para o produto final $j \in F$;

ta_{jk} : meta de teor desejada do parâmetro $k \in S$ para o produto final $j \in F$ no ano;

tt_{jkl}^R : assume valor 1 caso a restrição de meta trimestral do parâmetro de qualidade $k \in S$ do produto final $j \in F$ no trimestre $l \in T$ seja rígida, isto é, deve ser atingida; assume valor 0 caso contrário;

ta_{jk}^R : assume valor 1 caso a restrição de meta anual do parâmetro de qualidade $k \in S$ do produto final $j \in F$ seja rígida; assume valor 0 caso contrário;

São definidas ainda as seguintes variáveis de decisão:

x_{ijl}^E : quantidade, em toneladas, de minério armazenado no estoque inicial do produto primário $i \in P$ blendado para formar o produto final $j \in F$ no trimestre $l \in T$;

x_{ijtl}^P : quantidade, em toneladas, de minério produzido no trimestre $t \in T$ do produto primário $i \in P$ blendado para formar o produto final $j \in F$ no trimestre $l \in T$;

dt_{jkl}^+ : desvio positivo de teor em relação à meta do parâmetro $k \in S$ no produto final $j \in F$ no trimestre $l \in T$;

dt_{jkl}^- : desvio negativo de teor em relação à meta do parâmetro $k \in S$ no produto final $j \in F$ no trimestre $l \in T$;

dl_{jkl}^+ : desvio positivo de teor em relação ao limite superior do parâmetro $k \in S$ no produto final j no trimestre $l \in T$;

dl_{jkl}^- : desvio negativo de teor em relação ao limite inferior do parâmetro $k \in S$ no produto final j no trimestre $l \in T$;

dd_{jl} : desvio do atendimento à demanda do produto final $j \in F$ no trimestre $l \in T$;

A formulação matemática proposta por Alves et al. [2007] é dada a seguir, pelas equações (2.45)-(2.61).

Minimizar

$$\begin{aligned} \sum_{l \in T} \sum_{k \in S} \sum_{j \in F} wdt_{kj} (dt_{jkl}^+ + dt_{jkl}^- + dl_{jkl}^+ + dl_{jkl}^-) \\ + \sum_{j \in F} wdd_j \cdot dd_j \end{aligned} \quad (2.45)$$

Sujeito a

$$\sum_{i|(i,j) \in B} \sum_{t=1}^l (x_{ijtl}^P + x_{ijtl}^E) + dd_{jt} = pr_{jt} \quad \forall j \in F, \forall l \in T \quad (2.46)$$

$$\sum_{j|(i,j) \in B} \sum_{t \in T} x_{ijtl}^P \leq o_{il}^P \quad \forall i \in P, \forall l \in T \quad (2.47)$$

$$\sum_{j|(i,j) \in B} x_{ijl}^E \leq o_i^E \quad \forall i \in P \quad (2.48)$$

$$\sum_{j|(i,j) \in B} \sum_{l \in T} \left(\sum_{t=1}^l x_{ijtl}^P + x_{ijl}^E \right) \geq e_i \quad \forall i \in P \quad (2.49)$$

$$\sum_{\substack{i \in P, \\ tc_i = c}} \sum_{j|(i,j) \in B} \left(\sum_{t=1}^l x_{ijtl}^P + x_{ijl}^E \right) \leq cap_{cl} \quad \forall c \in C, \forall l \in T \quad (2.50)$$

$$x_{ijl}^E = b_{ijl}^E \quad \begin{array}{l} \forall i \in P, \forall j \in F, \\ \forall l \in T \mid b_{ijl}^E \geq 0 \end{array} \quad (2.51)$$

$$x_{ijtl}^P = b_{ijtl}^P \quad \begin{array}{l} \forall i \in P, \forall j \in F, \forall t \in T, \\ \forall l \in T \mid b_{ijtl}^P \geq 0 \end{array} \quad (2.52)$$

$$\begin{aligned} & \sum_{i|(i,j) \in B, c_{ij} \notin F} \left(\sum_{t=1}^l (q_{ikt}^P - tl_{jk}) x_{ijtl}^P + (q_{ik}^E - tl_{jk}) x_{ijl}^E \right) + \\ & \sum_{i|(i,j) \in B, c_{ij} \in F} \left(\sum_{t=1}^l (qf_{ikt}^P - tl_{jk}) x_{ijtl}^P + (qf_{ik}^E - tl_{jk}) x_{ijl}^E \right) + \\ & \quad \forall j \in F, \forall k \in S, \\ & \quad \forall l \in T, tt_{jkl}^R = 0 \\ & \quad Dl_{jkl}^- \geq 0 \end{aligned} \quad (2.53)$$

$$\begin{aligned} & \sum_{i|(i,j) \in B, c_{ij} \notin F} \left(\sum_{t=1}^l (q_{ikt}^P - tu_{jk}) x_{ijtl}^P + (q_{ik}^E - tu_{jk}) x_{ijl}^E \right) + \\ & \sum_{i|(i,j) \in B, c_{ij} \in F} \left(\sum_{t=1}^l (qf_{ikt}^P - tu_{jk}) x_{ijtl}^P + (qf_{ik}^E - tu_{jk}) x_{ijl}^E \right) - \\ & \quad \forall j \in F, \forall k \in S, \\ & \quad \forall l \in T, tt_{jkl}^R = 0 \\ & \quad Dl_{jkl}^+ \leq 0 \end{aligned} \quad (2.54)$$

$$\begin{aligned} & \sum_{i|(i,j) \in B, c_{ij} \notin F} \left(\sum_{t=1}^l (q_{ikt}^P - tr_{jk}) x_{ijtl}^P + (q_{ik}^E - tr_{jk}) x_{ijl}^E \right) + \\ & \sum_{i|(i,j) \in B, c_{ij} \in F} \left(\sum_{t=1}^l (qf_{ikt}^P - tr_{jk}) x_{ijtl}^P + (qf_{ik}^E - tr_{jk}) x_{ijl}^E \right) + \\ & \quad \forall j \in F, \forall k \in S, \\ & \quad \forall l \in T, tt_{jkl}^R = 0 \\ & \quad Dt_{jkl}^- - Dt_{jkl}^+ = 0 \end{aligned} \quad (2.55)$$

$$\begin{aligned}
& \sum_{i|(i,j) \in B, c_{ij} \notin F} \left(\sum_{t=1}^l (q_{ikt}^P - tt_{jk})x_{ijtl}^P \right. \\
& \quad \left. + (q_{ik}^E - tt_{jk})x_{ijl}^E \right) + \quad \forall j \in F, \forall k \in S, \\
& \sum_{i|(i,j) \in B, c_{ij} \in F} \left(\sum_{t=1}^l (qf_{ikt}^P - tt_{jk})x_{ijtl}^P \right. \\
& \quad \left. + (qf_{ik}^E - tt_{jk})x_{ijl}^E \right) = 0 \quad \forall l \in T, tt_{jkl}^R = 1
\end{aligned} \tag{2.56}$$

$$\begin{aligned}
& \sum_{i|(i,j) \in B, c_{ij} \notin F} \sum_{l \in T} \left(\sum_{t=1}^l (q_{ikt}^P - ta_{jk})x_{ijtl}^P \right. \\
& \quad \left. + (q_{ik}^E - ta_{jk})x_{ijl}^E \right) + \quad \forall j \in F, \forall k \in S, \\
& \sum_{i|(i,j) \in B, c_{ij} \in F} \sum_{l \in T} \left(\sum_{t=1}^l (qf_{ikt}^P - ta_{jk})x_{ijtl}^P \right. \\
& \quad \left. + (qf_{ik}^E - ta_{jk})x_{ijl}^E \right) = 0 \quad ta_{jk}^R = 1
\end{aligned} \tag{2.57}$$

$$x_{ijl}^E \geq 0 \quad \forall i \in P, \forall j \in F, \forall l \in T \tag{2.58}$$

$$x_{ijtl}^P \geq 0 \quad \forall i \in P, \forall j \in F, \forall t \in T, \forall l \in T \tag{2.59}$$

$$dt_{jkl}^+, dt_{jkl}^-, dl_{jkl}^+, dl_{jkl}^- \geq 0 \quad \forall j \in F, \forall k \in S, \forall l \in T \tag{2.60}$$

$$dd_{jl} \geq 0 \quad \forall j \in F, \forall l \in T \tag{2.61}$$

A função objetivo (2.45) é composta por duas partes: (i) a que busca a minimização dos desvios de qualidade; e (ii) a que busca a minimização dos desvios das metas de demanda. Assim, embora o problema seja multiobjetivo, Alves et al. [2007] o tratam através de uma função mono-objetivo, utilizando pesos (wdt_{kj} e wdd_j) para definir a importância das duas grandezas envolvidas: desvio do atendimento à demanda e desvio do atendimento às metas de qualidade. Os autores relatam ainda que os valores dos pesos foram definidos a partir de testes empíricos.

As equações (2.46) tem por objetivo medir os desvios da demanda dos produtos finais, enquanto as restrições (2.47) e (2.48) definem a capacidade de produção dos produtos primários e a sua disponibilidade em estoque.

As restrições (2.49) impõem um limite inferior para a utilização dos produtos primários, e as restrições (2.50) garantem que a capacidade trimestral de transporte dos

terminais de carga será respeitada. As restrições (2.51) e (2.52) asseguram que, caso haja imposição de blindagem, esta será verificada.

As equações (2.53) e (2.54) são responsáveis por mensurar quanto os limites inferior e superior, respectivamente, foram violados. As restrições (2.55) são responsáveis por medir os desvios das metas trimestrais dos parâmetros de controle. Quando as restrições são rígidas ($tt_{jkl}^R = 1$), isto é, quando os desvios são fixados em zero, a equação (2.56) é utilizada, obrigando ao atendimento das metas de qualidade. A equação (2.57) garante o atendimento da meta anual dos parâmetros de controle. Assim, quando as restrições de meta anual forem rígidas ($ta_{jk}^R = 1$), a qualidade do produto final poderá variar durante os trimestres desde que sua média anual atinja a meta.

Por fim, as restrições (2.58) a (2.61) tratam da não-negatividade das variáveis de decisão.

2.4 Outros Trabalhos

A literatura sobre aplicações de pesquisa operacional na mineração está muito concentrada na solução de problemas de planejamento de lavra e alocação e despacho de caminhões de mina. Merschmann [2002] desenvolveu um sistema computacional, denominado OTISIMIN (Otimizador e Simulador para Mineração), para resolver problemas de otimização e simulação em mineração. Neste sistema, a otimização contempla a construção e a resolução de um modelo matemático, baseado em programação linear, cujo objetivo é determinar o ritmo de lavra a ser implementado em cada frente levando-se em consideração a qualidade do minério da frente, a relação estéril/minério desejada, a produção requerida, as características dos equipamentos de carga e transporte e as características operacionais da mina.

A alocação e despacho de caminhões em mina é também tratada em Costa et al. [2004, 2005]. Estes trabalhos apresentam modelos que representam uma evolução em relação ao trabalho de Merschmann [2002], por incluírem metas de qualidade e produção, além de reduzirem significativamente o número de restrições necessárias. Em Guimaraes et al. [2007], a formulação de programação matemática de Costa et al.

[2004] é aperfeiçoada com a inclusão de restrições relativas à taxa de utilização de caminhões. Os resultados da otimização foram, ainda, validados por meio de um modelo de simulação computacional.

A literatura é muito escassa no que diz respeito a trabalhos voltados à mineração que dão ênfase a decisões relacionadas ao transporte entre mina e porto. Dentre os encontrados, os trabalhos de Mateus et al. [1994], Alves et al. [2007] e Pimentel et al. [2009] tratam de problemas de empresas mineradoras brasileiras. As proposições de Mateus et al. [1994] e Alves et al. [2007] são discutidas nas seções 2.2.1 e 2.3.1, respectivamente.

Pimentel et al. [2009] propõem uma abordagem baseada no conceito de cadeia de suprimentos. Nesta abordagem, o fluxo de matérias-primas e produtos acabados, bem como as operações de transformação, armazenagem e distribuição são tratados de forma integrada.

Capítulo 3

Descrição do Problema

Este capítulo tem como objetivo descrever o problema abordado, o qual se refere ao de uma empresa mineradora brasileira que conta com diversas minas situadas no Quadrilátero Ferrífero, em Minas Gerais. Como ponto de partida, o processo produtivo desta mineradora é discutido na Seção 3.1, estabelecendo as características do cenário alvo do trabalho. Em seguida, na Seção 3.2, é apresentada uma definição formal e estendida do problema tratado.

3.1 O Processo Produtivo da Empresa Mineradora

A atividade de mineração pode ser subdividida em uma série de fases produtivas, iniciando-se pela exploração do minério, sua transformação em produto, estocagem, manuseios diversos e transporte até o ponto de embarque. Todas estas fases são caracterizadas por um forte componente logístico, envolvendo movimentações de grandes massas, diferentes formas de transporte, estocagens e retomagens diversas.

Por razões diversas, que vão desde aspectos geológicos até outros ligados à necessidade de minimização dos impactos ambientais, o processo produtivo tornou-se complexo e descentralizado. Muitas são as situações em que o minério que é extraído em um local é beneficiado em outro, sendo transportado para um Terminal Ferroviário em um terceiro ponto para somente depois ser entregue ao cliente ou ser embarcado

em um Terminal Portuário.

Além disso, a escolha da forma de transporte impacta não somente no tempo e custo da movimentação, mas também nos parâmetros químicos e físicos que determinam a qualidade do minério. Por exemplo, quando um minério é transportado via ferrovia pode ocorrer quebra nas pelotas, alterando as propriedades granulométricas do minério.

As seções seguintes descrevem o processo produtivo, desde a produção até a venda. A Seção 3.1.1 trata da fase de lavra, quando o minério é extraído. As seções 3.1.2 e 3.1.3 abordam, respectivamente, as fases de beneficiamento e estocagem do minério. Por fim, a Seção 3.1.4 trata da fase de movimentação e venda do minério.

3.1.1 Fase de Lavra

A Fase de Lavra consiste basicamente na extração do ROM, do inglês *Run-of-Mine*, que representa o minério na forma bruta. Para realizar a extração são comumente utilizadas pás carregadeiras (??), que devem ser alocadas a frentes cujo minério atenda a restrições de qualidade pré-estabelecidas. Uma vez extraído, o ROM deve ser transportado para a área de estoque. Este transporte é geralmente feito por meio de caminhões (??), e tem sua eficiência determinada pelo trajeto que leva da frente à área de estoque. Estes trajetos podem ser extremamente complexos, cobrindo grandes áreas de terrenos irregulares. A alocação de equipamentos em frentes de lavra é uma tarefa complexa, e se constitui em um dos aspectos mais importantes no gerenciamento de minas a céu aberto, representando cerca de 50% dos custos operacionais totais nessas minas [Costa, 2005]. Segundo Rodrigues [2006], em 2006 cerca de 35 minas faziam uso de sistemas de alocação de caminhões no Brasil, com diferentes níveis de automação.

Além da produção de ROM, a atividade de extração mineral traz consigo a produção de uma quantidade variável de materiais de pouco ou nenhum valor econômico, respectivamente minério de baixa qualidade e estéril. Assim, de forma simplificada, a produção de ROM consiste na extração de minério e também na remoção de estéril. A Figura 3.1 mostra um esquema do “processo de produção” de ROM.

Pela Figura 3.1, observa-se que a obtenção do ROM pode se dar não somente através

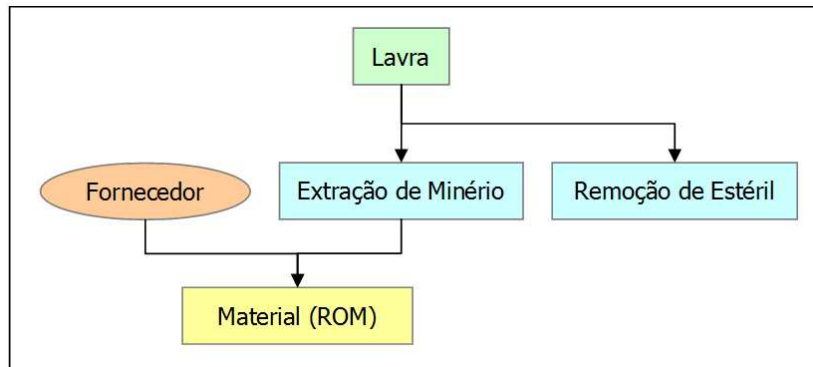


Figura 3.1. Processo de Produção de ROM

da Lavra como também a partir de compra de outros fornecedores.

Diversos trabalhos da literatura abordam problemas relacionados à fase de Lavra. Costa et al. [2005] desenvolveram uma formulação de programação matemática baseada em programação por metas - *Goal Programming* [Charnes e Cooper, 1961] - para resolver um problema de planejamento operacional de lavra em minas a céu aberto, considerando alocação estática de caminhões. O problema abordado consiste em determinar o número de viagens que cada caminhão deve fazer a cada frente de lavra, de forma que a produção horária de minério atinja o ritmo e a especificação de qualidade estabelecida. A função objetivo utilizada procura minimizar o desvio de produção e qualidade de diversos parâmetros de controle químicos e físicos.

O mesmo problema foi abordado de forma heurística por Guimarães et al. [2006b], que utilizaram um algoritmo baseado na metaheurística *Iterated Local Search* [Lourenço et al., 2003]. Em em Guimarães et al. [2006a], uma versão adaptada deste algoritmo é utilizada para tratar o problema considerando alocação dinâmica. A técnica utilizada mostrou-se adequada para o problema, obtendo bons resultados rapidamente, se comparados aos resultados do modelo de programação matemática.

3.1.2 Fase de Beneficiamento

Uma vez extraído, o minério passa para a fase de beneficiamento. Esta fase consiste em processar o minério a fim de modificar sua granulometria, bem como para elevar a concentração de determinadas frações do minério visando gerar produtos que sejam mais próximos das demandas dos clientes. Em função da redução das reservas de hematita (minério de ferro de altíssimo teor) e da abundância das reservas de itabiritos (minério de ferro de teor mais baixo), o processo de beneficiamento se faz cada vez mais necessário.

A partir do beneficiamento são gerados os Produtos Primários, que futuramente serão utilizados para formar os Produtos Finais de venda. Em casos (cada vez mais raros) em que o minério produzido possui as frações adequadas, o beneficiamento é desnecessário. A Figura 3.2 mostra um esquema que representa esta etapa. Nesta figura, observa-se que os Produtos Primários podem ser gerados a partir do Beneficiamento de ROM, a partir da própria ROM (sem o beneficiamento - no caso de hematitas, por exemplo) ou mesmo serem comprados de um fornecedor.

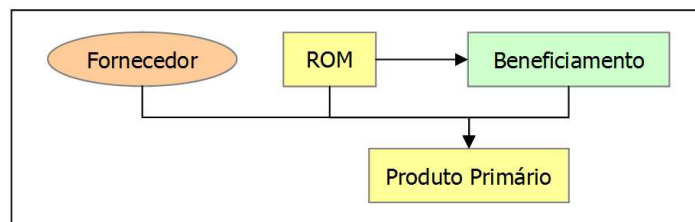


Figura 3.2. Processo de Geração do Produto Primário

3.1.3 Estocagem

A operação de estocagem consiste em armazenar os diferentes minérios em pilhas nos diversos pátios de estocagem. Após o beneficiamento, pode-se considerar que as características físicas e químicas dos minérios já são bem conhecidas. Assim, o objetivo da estocagem é basicamente armazenar minérios de diferentes características em pilhas específicas nos pátios de estocagem para posteriormente serem utilizados na composição

dos produtos finais. No caso dos minérios de ferro, estes são classificados em famílias de acordo com sua faixa granulométrica:

- **LO (*Lump Ore*)** - Faixa granulométrica de 6 mm a 31 mm, apresentando teor de ferro acima de 67% e baixo nível de impurezas, tais como aquelas quimicamente associadas a alumina (Al_2O_3), sílica (SiO_2) e fósforo (P).
- **SF (*Sinter Feed*)** - Faixa granulométrica de 0,15 mm a 6 mm, com teor de ferro em torno de 67% e baixos níveis de impurezas, principalmente sílica (SiO_2) e fósforo (P).
- **PFF (*Pellet Feed Fine*)** - Faixa granulométrica de 0,05 mm a 0,2 mm. Material muito fino, com teor de ferro variando de 67% a 68% e baixos níveis de impurezas.
- **HEM (*Hematitinha*)** - Faixa granulométrica de 6 mm a 14 mm, teor de ferro de 67% e baixos níveis de impurezas.

Em algumas mineradoras, existe a classificação de GR (Granulados), que é basicamente uma junção dos minérios das famílias LO (*Lump Ore*) e HEM (*Hematitinha*) em um único grupo.

Os minérios das diferentes famílias devem ser estocados em pilhas distintas, sendo que geralmente eles não são utilizados na composição de um mesmo produto. Uma vez estocados, os minérios poderão ser vendidos diretamente para o mercado interno ou ser transportados para terminais de carregamento ferroviário, onde poderão ser levados até clientes do mercado interno ou até portos para exportação.

3.1.4 Movimentação

Após serem extraídos, beneficiados e em alguns casos estocados, os minérios poderão ser utilizados para formar os produtos de venda. Para isso, no entanto, eles deverão ser transportados até o cliente. A fase de movimentação trata do transporte dos diversos Produtos Primários gerados para a formação dos Produtos Finais, que serão vendidos aos clientes.

No cenário brasileiro, o minério é geralmente distribuído para os clientes finais por meio de diversos modais de transporte: mineriodutos, ferrovias, hidrovias, portos e até mesmo rodovias.

Lu et al. [2005] afirmam que a alta demanda atual exige grandes fluxos de carga, em que necessidades de economia de escala sugerem a utilização (prioritária) de ferrovias e embarcações de alta capacidade. Assim, dos modais de transporte disponibilizados, o modal ferroviário caracteriza-se, especialmente, por sua capacidade de transportar grandes volumes, com elevada eficiência energética, principalmente em casos de deslocamentos a médias e grandes distâncias. Apresenta, ainda, maior segurança em relação ao modal rodoviário, com menor índice de acidentes e menor incidência de furtos e roubos. De acordo com a ANTT [Agência Nacional de Transportes Terrestres, 2009], o sistema ferroviário nacional é o maior da América Latina, em termos de carga transportada, atingindo 162,2 bilhões de tku (tonelada quilômetro útil), em 2001. No contexto da mineração no Brasil, trata-se do modal mais utilizado.

No processo de transporte, algumas restrições devem ser observadas:

- O minério deve ser disponibilizado no terminal de carga, ou seja, deve ser movimentado até o terminal por algum meio (correias de longa distância, mineriodutos, etc.).
- Um fator de manuseio (alteração na faixa granulométrica) do minério deve ser considerado, dependendo do modal de transporte utilizado.
- A quantidade de minério a ser transportado deve ser múltiplo de um valor que viabilize a operação. Por exemplo, em terminais ferroviários a quantidade de minério a ser movimentada deve ser múltipla da capacidade dos trens.

3.2 O Fluxo de Produtos da Empresa Mineradora

O planejamento do Fluxo de Produtos (FP) de uma empresa mineradora consiste em um plano de curto a médio prazo com o objetivo de determinar o fluxo dos minérios

produzidos, desde a extração até a venda. Cada minério possui diferentes características físicas e químicas, tais como o teor de determinado elemento químico ou a distribuição granulométrica. Assim, cada minério deve contribuir com uma qualidade apropriada para que o produto final de venda esteja o mais próximo possível das metas de qualidade. Logo, para compor os Produtos Finais é necessário misturar os diversos Produtos Primários, obtendo assim um minério que atenda as especificações de qualidade exigidas pelos clientes. Este processo de mistura de minérios é conhecido na mineração como “blendagem” (seção 3.1.4).

Em resumo, o FP consiste em determinar, para cada um dos períodos do planejamento, as quantidades de Produtos Primários a serem blendadas de modo a compor os Produtos Finais, indicando como se dará todo o transporte e atendendo da melhor forma possível às demandas e especificações dos clientes, respeitando diversas restrições operacionais da empresa. A Figura 3.3 mostra um “esquema” da solução do FP.

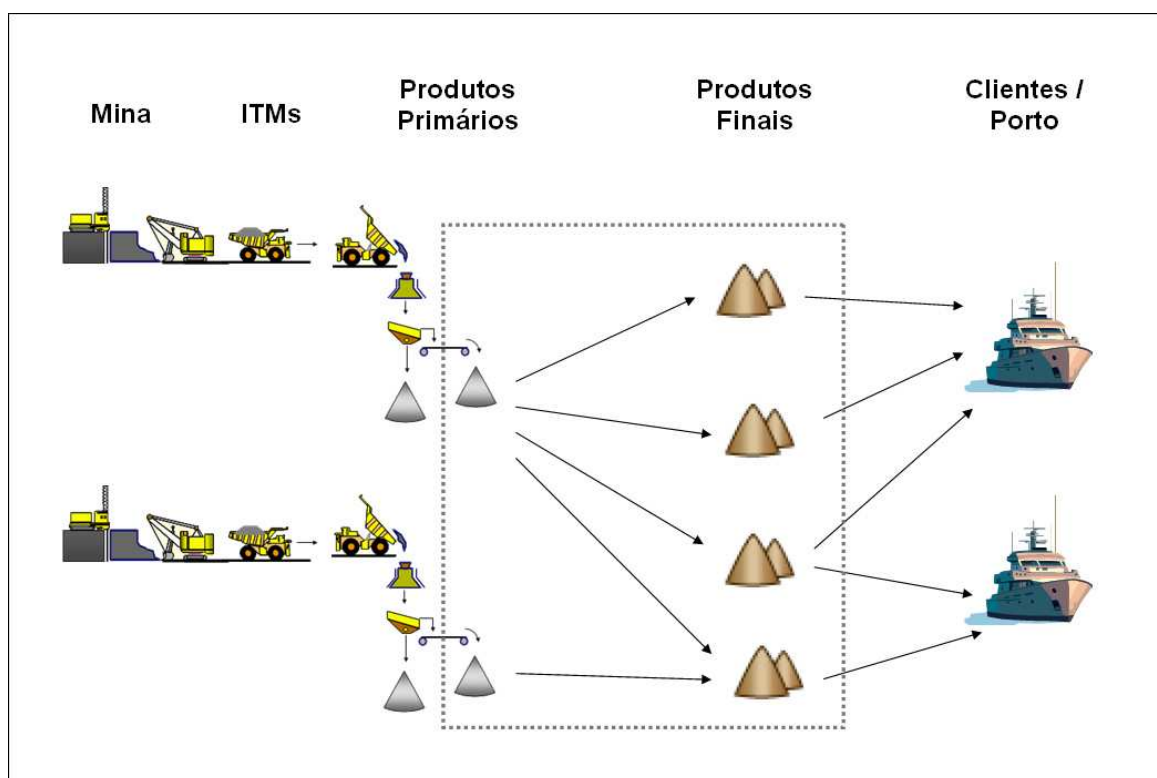


Figura 3.3. Representação do FP.

Na Figura 3.3 é apresentado um resumo do escopo abordado pelo FP. Inicialmente, o

minério é extraído da mina, sendo submetido em seguida ao processo de beneficiamento na ITM. Uma vez processado, o minério pode ser estocado ou imediatamente transportado para participar da composição de um Produto Final. Uma vez transformado em Produto Final, o minério poderá ser vendido ou transportado a um terminal portuário, onde será exportado. Nesta figura, nota-se que a área pontilhada faz referência ao transporte e à transformação do Produto Primário em Produto Final.

A Figura 3.4 apresenta alguns detalhes sobre como se dá o transporte e a blendagem dos produtos. A blendagem pode se dar em algum terminal de carga (ponto de origem) ou mesmo no porto (ponto de destino). Na figura, o conceito de rota é levemente simplificado para facilitar a compreensão. Na prática, podem haver diversos pontos intermediários entre uma origem e um destino. Ainda nesta figura, observa-se onde incorrem os custos com produção e transporte.

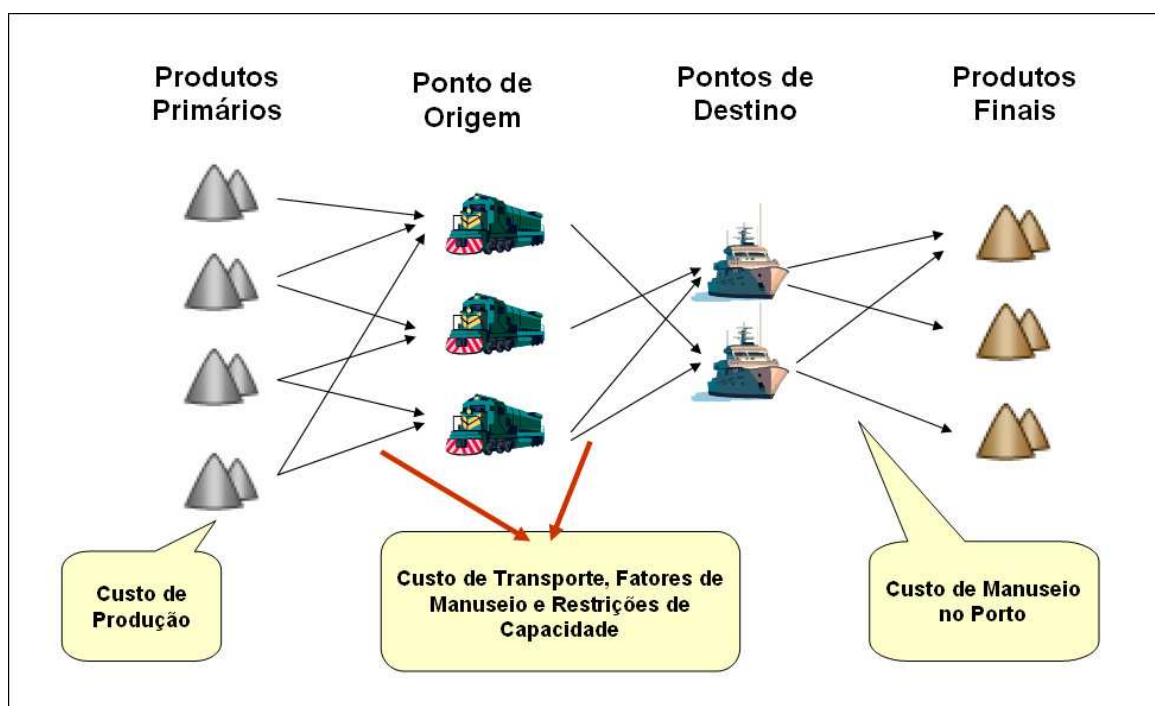


Figura 3.4. Representação do Transporte.

A Figura 3.5 mostra um esquema de entrada e saída de dados do FP. Esta figura apresenta os seguintes dados de entrada:

- **Produção:** capacidades de produção de cada Produto Primário de cada mina e o valor de seus parâmetros de qualidade (percentual de ferro, sílica, alumina, granulometria, etc.).
- **Estoque:** quantidade e qualidade dos produtos em estoque nos diferentes pátios de minério.
- **Transporte e Transferências entre UOs:** relação de rotas de transporte com custo, capacidade e respectivos fatores de manuseio¹. Estas rotas são dadas através de grafos onde os nós representam os pontos de carga e descarga e os arcos representam os caminhos a serem percorridos.
- **Compras:** relação de minérios cuja produção pode ser terceirizada, isto é, minérios que podem ser comprados de outras mineradoras.
- **Vendas:** demandas dos clientes, sendo que o ponto de entrega, a quantidade, as metas e limites para os parâmetros de qualidade desejados são especificados.

Como saída do FP, têm-se:

- **Fluxo dos Produtos:** definição do fluxo dos produtos, ou seja, das misturas a partir das quais os Produtos Finais serão gerados, bem como as rotas de transporte utilizadas.
- **Estoques Previstos:** quantidade e qualidade dos estoques resultantes após cada período de tempo.
- **Qualidade:** previsão do atendimento ou não-atendimento das metas de qualidade definidas.
- **Custos e Racionalização do Transporte:** relação das rotas utilizadas e identificação de gargalos na logística de transporte como um todo, além da otimização da mesma.

¹Fatores de manuseios são pequenas oscilações nas características do minério ocasionadas pelo seu transporte ou manipulação.

- **Transporte de Produtos por Terminal de Carga:** definição das metas de qualidade e de quantidade de cada terminal de carga em cada período de tempo.

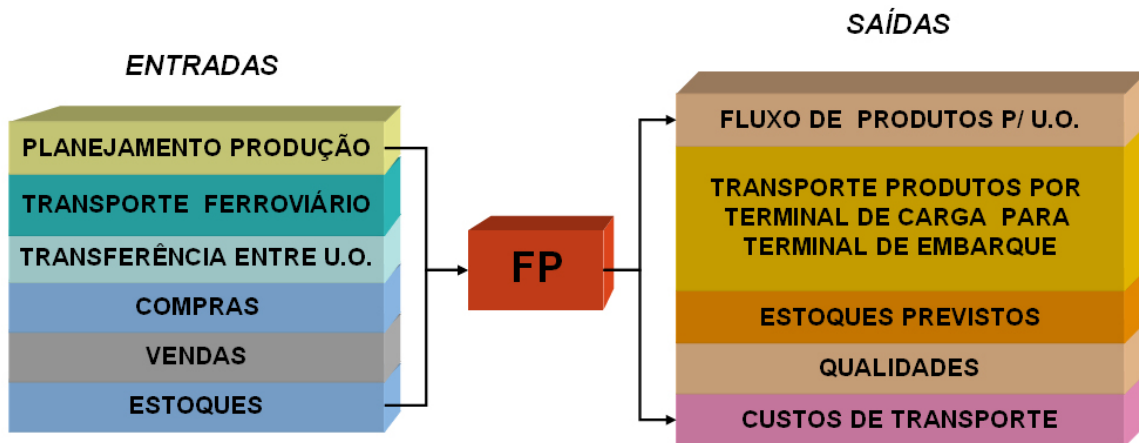


Figura 3.5. Representação dos principais dados de entrada e saída do FP.

Neste trabalho, são considerados quatro níveis de planejamento do FP: anual, trimestral, mensal e dia-a-dia com horizonte mensal.

Capítulo 4

Formulação Matemática

Este capítulo apresenta a formulação matemática do FP, através da abordagem de otimização multiobjetivo. A Seção 4.1 descreve os principais conceitos relacionados a otimização multiobjetivo, enquanto a Seção 4.2 apresenta o modelo proposto.

4.1 Otimização Multiobjetivo

Otimização multiobjetivo ou multicritério é caracterizada pela presença de n funções objetivos f_i , $i = 0, \dots, n$, sujeito às restrições do problema. A principal diferença entre a abordagem de otimização multiobjetivo e a mono-objetivo é o espaço de soluções de cada uma. Em otimização mono-objetivo, o espaço de soluções é unidimensional, enquanto na otimização multiobjetivo este espaço é multidimensional. As figuras 4.1 e 4.2 ilustram, respectivamente, os espaços de solução unidimensional e multidimensional.



Figura 4.1. Espaço de soluções unidimensional (otimização mono-objetivo)

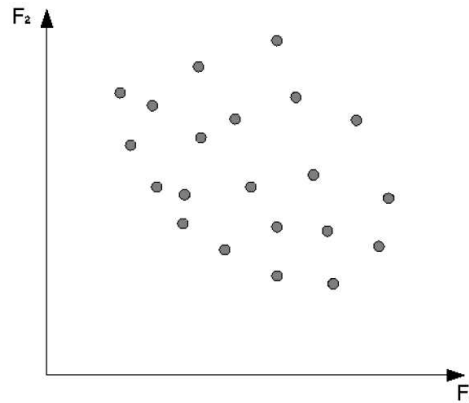


Figura 4.2. Espaço de soluções multidimensional (otimização multiobjetivo)

4.1.1 Conjunto Pareto-ótimo

O objeto fundamental da otimização multiobjetivo consiste em um conjunto de soluções X^* , denominado conjunto Pareto-ótimo, que contém as possíveis soluções x^* de um problema de otimização multiobjetivo. Para definir estes elementos, é necessário antes apresentar o conceito de dominância (Definição 4.1). Para isto, considere que $f_1, \dots, f_n$ são as n funções objetivo do problema de otimização, e que X é o conjunto de todas as soluções.

Definição 4.1 (Dominância). *Num problema de minimização, diz-se que o ponto $x_1 \in X$ domina o ponto $x_2 \in X$ se $f_i(x_1) \leq f_i(x_2)$, $i = 1, \dots, n$ e $\exists i \mid f_i(x_1) \neq f_i(x_2)$.*

A otimização das soluções em um problema multiobjetivo está diretamente relacionada com o conceito de dominância. Isto porque o processo de otimização consiste em encontrar soluções $x^* \in X^*$ não dominadas através da investigação das soluções $x \in X$. Neste contexto, Takahashi [2007] define os elementos do conjunto Pareto-ótimo por meio da Definição 4.2.

Definição 4.2 (Solução Pareto-ótima). *Diz-se que x^* é uma solução Pareto-ótima de um problema de otimização se não existir nenhuma outra solução $x \in X$ tal que $f_i(x) \leq f_i(x^*)$, $i = 1, \dots, n$ e $\exists i \mid f_i(x) \neq f_i(x^*)$, ou seja, se x^* não for dominada por nenhuma outra solução.*

Basicamente, pode-se dizer que todas as soluções não-dominadas constituem o conjunto Pareto-ótimo. Ainda assim, este conjunto pode ter cardinalidade infinita. Considerando um problema de otimização com duas funções objetivo, F_1 e F_2 , a Figura 4.3 mostra pontos que representam soluções não-dominadas, ou seja, soluções Pareto-ótimas.

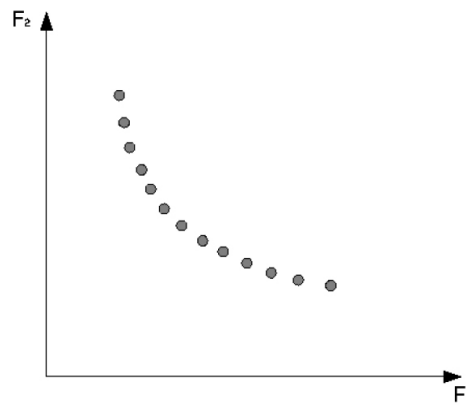


Figura 4.3. Conjunto de soluções Pareto-ótimas

Problemas multiobjetivos são muito comuns, uma vez que, em geral, nos problemas reais deseja-se otimizar mais de um recurso. Além disso, geralmente as funções objetivo do problema são conflitantes, ou seja, uma melhora num critério pode acarretar uma piora em outro. Para ilustrar esta situação, imagine que alguém deseja comprar um carro. Este comprador deseja um carro com maior potência, mas cujo custo seja baixo. Trata-se de um problema multiobjetivo em que, quanto maior a potência maior o preço e, de forma análoga, quanto menor o preço, menor a potência. Estes conflitos levam a uma maior diversidade entre as soluções Pareto-ótimas.

Em casos onde não existe nenhum conflito ao otimizar diversas funções objetivo, uma solução Pareto-ótima será chamada solução utópica. Takahashi [2007] descreve uma solução utópica por meio da Definição 4.3.

Definição 4.3 (Solução Utópica). *Diz-se que uma solução y^* é utópica quando $y_i^* = f_i(x_i)$, $i = 1, \dots, n$, onde: $x_i = \underset{x \in X}{\operatorname{argmin}} f_i(x)$.*

A Definição 4.3 afirma que a solução utópica é aquela que apresenta valores ótimos para todas as funções objetivo do problema.

4.1.2 Formas de Resolução

Em problemas de otimização multiobjetivo, tanto o espaço dos objetivos quanto o espaço das variáveis é multidimensional. Assim, existe um número maior de avaliações a serem feitas para obter todas as soluções do problema e, portanto, pode-se dizer que problemas desta classe são mais difíceis de se resolver.

Existem diversas formas de resolução do problemas multiobjetivos. Destas, duas se destacam pela simplicidade da abordagem [Takahashi, 2007]. São elas:

- **Ponderação dos objetivos:** esta técnica consiste em definir multiplicadores λ_i , $i = 1, \dots, n$ para cada uma das n funções objetivo do problema, de forma que $\sum_{i=1}^n \lambda_i = 1$. As n funções objetivo do problema são multiplicadas por seus respectivos λ_i e os valores obtidos são somados. Assim, o problema multiobjetivo passa a ser tratado como um problema de otimização mono-objetivo. Esta simplificação é uma das principais vantagens desta abordagem. Variando-se os multiplicadores λ_i , é possível obter outras soluções Pareto-ótimas. Ainda assim, em casos em que as ordens de grandezas associadas aos diferentes objetivos são muito diferentes, é possível que mesmo grandes variações nos valores das variáveis λ_i levem à mesma solução Pareto-ótima.
- **Problema ϵ -Restrito:** nesta abordagem, problemas multiobjetivo também são convertidos em problemas mono-objetivo. No entanto, apenas uma função objetivo é otimizada por vez, enquanto as demais são transformadas em restrições. Para cada função objetivo considerada como restrição, um valor ϵ_i , $i = 1, \dots, n - 1$ é associado. Este valor representa o limite da restrição. Variações dos valores ϵ_i geram novos problemas de otimização mono-objetivo, podendo resultar na obtenção de diferentes soluções Pareto-ótimas. Como desvantagem, podemos citar o aumento na complexidade do problema ao considerar $n - 1$ restrições adicionais, além da dificuldade de se obter valores adequados para as variáveis ϵ_i .

4.2 Modelo Matemático

Para modelar o Problema de Planejamento do Fluxo de Produtos (FP), foi utilizada a técnica programação por metas ou *goal programming* [Charnes e Cooper, 1961]. Nesta modelagem, o problema é tratado como multiobjetivo, levando em conta os três objetivos a seguir:

- Atendimento à demanda;
- Atendimento às metas de qualidade;
- Redução do custo total com transporte.

O modelo multi-objetivo do FP, dado pelas equações (4.1) - (4.21), utiliza a seguinte notação:

T : Conjunto de estágios de tempo do planejamento;

M : Conjunto de pátios de estocagem de minério das diferentes minas;

C : Conjunto de terminais de carga;

C^F : Subconjunto de terminais de carga, $C^F \subseteq C$, que submetem os produtos primários a fatores de manuseios (mudança de granulometria durante o carregamento);

C^N : Subconjunto de terminais de carga, $C^N \subseteq C$, que não submetem os produtos primários a fatores de manuseios (mudança de granulometria durante o carregamento);

G : Conjunto de pontos de descarga (portos ou clientes);

P : Conjunto de produtos primários produzidos pelas minas ou em estoque nos diferentes pátios de minério;

F : Conjunto de produtos finais (demandas dos clientes);

- S : Conjunto dos parâmetros de qualidade analisados nos produtos finais;
- B : Conjunto de possíveis misturas (blendagens) (i, j) entre produtos primários $(i \in P)$ e produtos finais $(j \in F)$;
- R : Conjunto de todas as rotas de transporte disponíveis;
- $R^B(i, j)$: Subconjunto de rotas, $R^B(i, j) \subseteq R$, capazes de transportar o produto primário $i \in P$ para compor o produto final $j \in F$;
- $R^C(i, j, c)$: Subconjunto de rotas ($R^C(i, j, c) \subseteq R^B(i, j)$) que utilizam o terminal de carga $c \in C$ no transporte;
- $R^G(i, j, g)$: Subconjunto de rotas ($R^G(i, j, g) \subseteq R^B(i, j)$) que desembocam no ponto de descarga $g \in G$;
- $R^M(i, j, m, c)$: Subconjunto de rotas ($R^M(i, j, m, c) \subseteq R^C(i, j, c)$) que transportam o produto primário $i \in P$ do pátio de estocagem $m \in M$ ao terminal de carga $c \in C$ para futuramente ser utilizado na composição do produto final $j \in F$;
- $R^P(i, j, c, g)$: Subconjunto de rotas ($R^P(i, j, c, g) \subseteq R^G(i, j, g)$) que transportam o produto primário $i \in P$ do terminal de carga $c \in C$ para ser utilizado na composição do produto final $j \in F$ no ponto de descaga $g \in G$;
- cc_{ct} : Valor do custo da ativação do terminal de carga $c \in C$ no instante de tempo $t \in T$;
- cs_t^r : Valor do custo de transportar 1 Kt (quilotoneladas) de um produto primário¹ através da rota $r \in R$ no instante de tempo $t \in T$;
- o_{it} : Quantidade ofertada do produto primário $i \in P$ (Kt) no instante de tempo $t \in T$;
- q_{ikt} : Valor do parâmetro de qualidade $k \in S$ do produto primário $i \in P$ (%) no instante de tempo $t \in T$;

¹Assume-se que todos os produtos têm os mesmos custos de transporte.

- pr_j : Quantidade demandada do produto final $j \in F$ (Kt);
- pl_j : Quantidade da demanda do produto final $j \in F$ (Kt) que deve ser obrigatoriamente atendida;
- pt_j : Instante de tempo limite, $t \in T$, em que a demanda pode ser atendida;
- tr_{jk} : Valor requerido para o parâmetro de qualidade $k \in S$ do produto final $j \in F$ (%);
- tl_{jk} : Valor mínimo admissível para o parâmetro de qualidade $k \in S$ do produto final $j \in F$ (%);
- tu_{jk} : Valor máximo admissível para o parâmetro de qualidade $k \in S$ do produto final $j \in F$ (%);
- fm_{ik} : Fator de manuseio do parâmetro $k \in S$ sobre o produto primário $i \in P$ quando este é carregado por meio de um terminal de carga $c \in C^F$;
- u_{cg} : Quantidade unitária transportada do terminal de carga $c \in C$ ao ponto de descarga $g \in G$;
- v_j : Quantidade mínima percentual ($0 \leq v_j \leq 1$) da participação de um produto primário na composição do produto final $j \in F$;
- ll_{ct}^C : Quantidade mínima de minério transportado pelo terminal de carga $c \in C$ (Kt) que torne viável a sua utilização no instante de tempo $t \in T$;
- lu_{ct}^C : Capacidade máxima de carga do terminal de carga $c \in C$ (Kt) no instante de tempo $t \in T$;
- lu_{gt}^G : Capacidade máxima de carga do ponto de descarga $g \in G$ (Kt) no instante de tempo $t \in T$;
- lu_{mct}^M : Capacidade máxima de transporte entre o pátio de minérios $m \in M$ e o terminal de carga $c \in C$ no instante de tempo $t \in T$;

lu_{cgt}^G : Capacidade máxima de transporte entre o terminal de carga $c \in C$ e o ponto de descarga $g \in G$ no instante de tempo $t \in T$;

α_{jkt}^- : Penalidade por desvio negativo do parâmetro de qualidade $k \in S$ no produto final $j \in F$ no instante de tempo $t \in T$;

α_{jkt}^+ : Penalidade por desvio positivo do parâmetro de qualidade $k \in S$ no produto final $j \in F$ no instante de tempo $t \in T$;

β_{jt}^- : Penalidade por desvio negativo do atendimento à demanda do produto final $j \in F$ no instante de tempo $t \in T$;

ω_{jk} : Fator de correção do parâmetro de qualidade $k \in S$ do produto final $j \in F$;

São definidas ainda as seguintes variáveis de decisão:

x_{ijt}^r : Quantidade do produto primário $i \in P$ a ser utilizado na composição do produto final $j \in F$, sendo transportado pela rota $r \in R$ no instante de tempo $t \in T$;

y_{cgt} : Quantidade de trens carregados no terminal de carga $c \in C$ e descarregados no ponto de descarga $g \in G$ no instante de tempo $t \in T$;

z_{ij} : Variável binária que assume valor 1 se um produto primário $i \in P$ fizer parte da composição do produto final $j \in F$; e 0, caso contrário;

z_{ct}^C : Variável binária que assume valor 1 se o terminal de carga $c \in C$ for utilizado no instante de tempo $t \in T$; e 0, caso contrário;

dt_{jk}^- : Desvio negativo do parâmetro de qualidade $k \in S$ no produto final $j \in F$;

dt_{jk}^+ : Desvio positivo do parâmetro de qualidade $k \in S$ no produto final $j \in F$;

dp_j^- : Desvio negativo do atendimento à demanda do produto final $j \in F$;

A formulação matemática proposta para resolver o Problema de Planejamento do Fluxo de Produtos (FP) é dada a seguir pelas equações (4.1) - (4.21):

Minimizar

$$\left\{ \begin{array}{l} F_1 = \sum_{j \in F} \beta_j^- dp_j^- \\ F_2 = \sum_{j \in F} \sum_{k \in S} \alpha_{jkt}^- \omega_{jk} dt_{jk}^- + \sum_{j \in F} \sum_{k \in S} \alpha_{jkt}^+ \omega_{jk} dt_{jk}^+ \\ F_3 = \sum_{(i,j) \in B} \sum_{r \in R^B(i,j)} \sum_{t=1}^{pt_j} cs_t^r x_{ijt}^r + \sum_{c \in C} \sum_{t \in T} cc_{ct} z_{ct}^C \end{array} \right. \quad (4.1)$$

Sujeito a

$$\sum_{i|(i,j) \in B} \sum_{r \in R^B(i,j)} \sum_{t=1}^{pt_j} x_{ijt}^r = pl_j \quad \forall j \in F \quad (4.2)$$

$$\sum_{i|(i,j) \in B} \sum_{r \in R^B(i,j)} \sum_{t=1}^{pt_j} x_{ijt}^r + dp_j^- = pr_j \quad \forall j \in F \quad (4.3)$$

$$\begin{aligned} & \sum_{i|(i,j) \in B} \sum_{r \in R^C(i,j,n)} \sum_{t=1}^{pt_j} (q_{ikt} - tl_{jk}) x_{ijt}^r + \\ & \sum_{i|(i,j) \in B} \sum_{r \in R^C(i,j,f)} \sum_{t=1}^{pt_j} (q_{ikt} + fm_{ik} - tl_{jk}) x_{ijt}^r \geq 0 \end{aligned} \quad \begin{array}{l} \forall j \in F, \forall k \in S, \\ \forall n \in C^N, \forall f \in C^F \end{array} \quad (4.4)$$

$$\begin{aligned} & \sum_{i|(i,j) \in B} \sum_{r \in R^C(i,j,n)} \sum_{t=1}^{pt_j} (q_{ikt} - tu_{jk}) x_{ijt}^r + \\ & \sum_{i|(i,j) \in B} \sum_{r \in R^C(i,j,f)} \sum_{t=1}^{pt_j} (q_{ikt} + fm_{ik} - tu_{jk}) x_{ijt}^r \leq 0 \end{aligned} \quad \begin{array}{l} \forall j \in F, \forall k \in S, \\ \forall n \in C^N, \forall f \in C^F \end{array} \quad (4.5)$$

$$\begin{aligned} & \sum_{i|(i,j) \in B} \sum_{r \in R^C(i,j,n)} \sum_{t=1}^{pt_j} (q_{ikt} - tr_{jk}) x_{ijt}^r + \\ & \sum_{i|(i,j) \in B} \sum_{r \in R^C(i,j,f)} \sum_{t=1}^{pt_j} (q_{ikt} + fm_{ik} - tr_{jk}) x_{ijt}^r \end{aligned} \quad \begin{array}{l} \forall j \in F, \forall k \in S, \\ \forall n \in C^N, \forall f \in C^F \end{array} \quad (4.6)$$

$$dt_{jk}^- - dt_{jk}^+ = 0$$

$$\sum_{j|(i,j) \in B} \sum_{r \in R^B(i,j)} x_{ijt}^r \leq o_{it} \quad \forall i \in P, \forall t \in T \quad (4.7)$$

$$\sum_{(i,j) \in B} \sum_{r \in R^P(i,j,c,g)} x_{ijt}^r - y_{cgt} u_{cg} = 0 \quad \forall c \in C, \forall g \in G, \forall t \in T \quad (4.8)$$

$$\sum_{r \in R^B(i,j)} \sum_{t \in T} x_{ijt}^r - z_{ij} v_j p r_j \geq 0 \quad \forall i \in P, \forall j \in F \quad (4.9)$$

$$z_{ij} - \sum_{r \in R^B(i,j)} \sum_{t \in T} \frac{x_{ijt}^r}{p r_j} \geq 0 \quad \forall i \in P, \forall j \in F \mid p r_j > 0 \quad (4.10)$$

$$\sum_{(i,j) \in B} \sum_{r \in R^C(i,j,c)} x_{ijt}^r - z_{ct}^C l l_{ct}^C \geq 0 \quad \forall c \in C, \forall t \in T \quad (4.11)$$

$$\sum_{(i,j) \in B} \sum_{r \in R^C(i,j,c)} x_{ijt}^r \leq l u_{ct}^C \quad \forall c \in C, \forall t \in T \quad (4.12)$$

$$\sum_{(i,j) \in B} \sum_{r \in R^G(i,j,g)} x_{ijt}^r \leq l u_{gt}^G \quad \forall g \in G, \forall t \in T \quad (4.13)$$

$$\sum_{(i,j) \in B} \sum_{r \in R^M(i,j,m,c)} x_{ijt}^r \leq l u_{mct}^M \quad \forall m \in M, \forall c \in C, \forall t \in T \quad (4.14)$$

$$\sum_{(i,j) \in B} \sum_{r \in R^P(i,j,c,g)} x_{ijt}^r \leq l u_{cgt}^P \quad \forall c \in C, \forall g \in G, \forall t \in T \quad (4.15)$$

$$x_{ijt}^r \geq 0 \quad \begin{array}{l} \forall i \in P, \forall j \in F, \forall t \in T, \\ \forall r \in R^B(i,j) \end{array} \quad (4.16)$$

$$y_{cgt} \geq 0, \quad y_{cgt} \in \mathbb{Z} \quad \forall c \in C, \forall g \in G, \forall t \in T \quad (4.17)$$

$$0 \leq z_{ij} \leq 1, \quad z_{ij} \in \mathbb{Z} \quad \forall i \in P, \forall j \in F \quad (4.18)$$

$$0 \leq z_{ct}^C \leq 1, \quad z_{ct}^C \in \mathbb{Z} \quad \forall c \in C, \forall t \in T \quad (4.19)$$

$$dt_{jk}^-, dt_{jk}^+ \geq 0 \quad \forall j \in F, \forall k \in S \quad (4.20)$$

$$dp_j^- \geq 0 \quad \forall j \in F \quad (4.21)$$

O modelo possui três funções objetivo: F_1 , F_2 e F_3 , dadas pelas equações (4.1). A função F_1 representa a soma, em Kt (quilotoneladas), dos desvios de atendimento às diferentes demandas por minério. Como na prática existe uma hierarquia de importância entre os produtos finais de venda, pesos β_j^- são aplicados no não atendimento à demanda de cada produto. O valor deste peso é definido pelo planejador e, geralmente, representa o retorno financeiro de venda do produto. De forma geral, quanto maior for a demanda satisfeita, menor o valor de F_1 ; se F_1 atingir valor igual a 0, isto implica que todas as demandas foram atendidas.

A função F_2 representa o desvio do atendimento às metas de qualidade dos diferentes minérios comercializados. De forma análoga à função objetivo anterior, F_2 utiliza uma matriz de pesos para definir a importância de cada parâmetro de qualidade de cada produto final de venda. Este parâmetro, α_{jk}^+ , é definido pelo planejador de acordo com a importância dada aos parâmetros e aos produtos. Além do peso α_{jk}^+ , um fator de correção ω_{jk} também é utilizado, uma vez que os diferentes parâmetros de qualidade são de ordem de grandeza diferentes (ex. enquanto o percentual de ferro é da ordem de 66%, o de fósforo é da ordem de 0,03%). Assim, este fator de correção ω_{jk} objetiva equilibrar os valores dos diferentes parâmetros, sendo calculado de acordo com a equação (4.22).

$$\omega_{jk} = \left(\frac{tu_{jk} - tr_{jk}}{2} + \frac{tr_{jk} - tl_{jk}}{2} \right)^{-1} \quad \begin{array}{l} \forall j \in F, \forall k \in S, \\ tu_{jk} - tr_{jk} > 0 \text{ ou } tr_{jk} - tl_{jk} > 0 \end{array} \quad (4.22)$$

Analisando a equação (4.22) e as restrições (4.6), conclui-se que $\omega_{jk}dt_{jk}^- \in [0, 1]$ e $\omega_{jk}dt_{jk}^+ \in [0, 1]$, para qualquer parâmetro $k \in S$ de um produto final $j \in F$.

F_3 , a terceira e última função objetivo, representa a soma de todos os custos com transporte. Esta soma inclui o transporte da mina ao terminal de carga e do terminal de carga ao ponto de descarga, além dos valores fixos pela utilização dos terminais.

O primeiro conjunto de restrições trata da demanda dos produtos finais. As restrições (4.2) asseguram que uma quantidade mínima pl_j da demanda do produto final $j \in F$ será obrigatoriamente atendida. As restrições (4.3), por outro lado, mensuram

o quanto a demanda de cada produto não foi atendida, sendo este valor armazenado na variável dp_j^- para depois ser avaliado na função objetivo.

O segundo conjunto de restrições trata da qualidade dos produtos gerados. As restrições (4.4) e (4.5) garantem, respectivamente, que os limites inferior e superior dos parâmetros de qualidade dos produtos finais serão respeitados. As restrições (4.6) computam o valor das variáveis dt_{jk}^- e dt_{jk}^+ , que representam, respectivamente, os desvios inferior e superior da meta de qualidade do parâmetro $k \in S$ do produto final $j \in F$. Estas restrições levam em conta, ainda, os fatores de manuseio que um determinado produto pode sofrer de acordo com a forma de transporte escolhida.

A quantidade disponível de cada produto primário, dada de acordo com as capacidades produtivas e do estoque, é limitada pelas restrições (4.7).

As restrições (4.8) garantem que a quantidade de um produto primário utilizada na composição de um produto final será múltiplo de u_{cg} , onde c representa o terminal de carga e g o ponto de descarga. Essa restrição é necessária pois cada par terminal de carga e ponto de descarga pode ter capacidade distinta. Por exemplo, a capacidade de um trem em um determinado terminal de carga enviando para um certo ponto de descarga é de 13 Kt, enquanto em outro pode ser de 6 Kt.

A quantidade mínima de utilização de um produto primário na composição de um determinado produto final é tratado pelas restrições (4.9) e (4.10). A definição deste valor mínimo evita situações operacionalmente inviáveis, como a alocação de uma quantidade muito pequena de um dado produto primário na composição de um produto final. Esta restrição é ainda importante na busca da homogeneização dos estoques dos produtos finais obtidos.

As restrições (4.11) garantem que se um determinado terminal de carga $c \in C$ for utilizado no instante de tempo $t \in T$, uma quantidade mínima ll_{ct}^C deverá ser carregada por ele. As restrições (4.12), por outro lado, asseguram que a capacidade máxima de transporte do terminal em um determinado instante de tempo será respeitada.

As restrições (4.13) limitam a quantidade de minério que um ponto de descarga pode receber em um determinado intervalo de tempo, enquanto as restrições (4.14) e (4.15)

impedem, repectivamente, que as capacidades de transporte entre minas e terminais de carga e entre terminais de carga e pontos de descarga sejam desrespeitadas.

Por fim, o último grupo de restrições - equações (4.16) a (4.21) - trata dos limites e integralidade das variáveis.

Capítulo 5

Algoritmos Heurísticos

Em alguns problemas combinatórios, o uso de métodos exatos pode se tornar bastante restrito, uma vez que eles podem não ser capazes de encontrar uma solução de qualidade em tempo aceitável. Este é o motivo pelo qual os pesquisadores têm concentrado esforços na utilização de heurísticas para solucionar problemas deste nível de complexidade.

Dada a dificuldade de resolução de instâncias maiores do FP através da abordagem de programação matemática por meio de pacotes comerciais de otimização, metodologias heurísticas foram desenvolvidas para tratar o problema.

Este capítulo tem por objetivo apresentar estes algoritmos heurísticos. Os conceitos básicos das metodologias utilizadas são apresentados na Seção 5.1. Em seguida, na Seção 5.2, a abordagem baseada na técnica *relax-and-fix* é explanada. Por fim, a Seção 5.3 apresenta o algoritmo híbrido GRASP+ILS aplicado ao problema.

5.1 Conceitos Iniciais

Nesta seção, os princípios das metodologias heurísticas *relax-and-fix* (Seção 5.1.1), GRASP (Seção 5.1.2), ILS (Seção 5.1.3) e VND (Seção 5.1.4) são apresentados.

5.1.1 Relax-And-Fix

A heurística *relax-and-fix* [Dillenberger et al., 1994; Wolsey, 1998] é uma abordagem de solução baseada em métodos exatos. Essa abordagem tem sido usada na solução de diversos tipos de problemas de forma pura ou híbrida. Nesta heurística, o conjunto de variáveis inteiras de um problema de otimização inteira mista é particionado em P conjuntos disjuntos, Q_i , $i = 1, \dots, P$, de diferentes importâncias. O número P de conjuntos determina o número de iterações da heurística. Em uma iteração n , apenas as variáveis do conjunto Q_n são definidas como inteiras, enquanto as demais variáveis são relaxadas. O submodelo resultante é então resolvido. O valor retornado para as variáveis inteiras é fixado e o próximo submodelo, gerado pela fixação dos conjuntos $Q_1, \dots, Q_n$ e pela integralização das variáveis de Q_{n+1} , é resolvido.

Durante as iterações da heurística, é possível que algum submodelo seja inviável. Neste caso, não é possível encontrar uma solução viável para o problema na iteração n com as variáveis dos conjuntos Q_i , $i = 1, \dots, n - 1$ assumindo os valores fixados.

A característica principal da heurística *relax-and-fix* é a redução do número de variáveis inteiras dos submodelos de programação inteira mista, tornando-os mais fáceis de resolver. Assim, a partição do conjunto de variáveis e o critério de seleção das variáveis a serem fixadas têm grande influência na dificuldade de solução dos submodelos.

No trabalho de Federgruen et al. [2007] é apresentada uma heurística de intervalos progressivos, onde a *relax-and-fix* é tida como um caso particular. O autor considera que nesta heurística, não há fixação de variáveis contínuas, o que dá o máximo de flexibilidade na obtenção de soluções viáveis. O caso extremo de menor flexibilidade é a fixação de todas as variáveis contínuas da iteração. Estas duas estratégias de fixação de variáveis são denominadas respectivamente por Heurística de Horizonte Expandido e Heurística de Particionamento Estrito.

Na heurística *relax-and-fix*, a primeira iteração do método é feita sobre uma relaxação do problema. Assim, o *gap* pode ser obtido de acordo com a equação (5.1), onde UB representa a solução viável final obtida pelo método e LB representa o valor da

função objetivo obtida na primeira iteração do método.

$$gap = 1 - \frac{LB}{UB} \quad (5.1)$$

O Algoritmo 5.1 mostra o pseudocódigo de um procedimento *relax-and-fix* básico.

Algoritmo 5.1 Procedimento *RelaxAndFix* Básico

- 1: divida o conjunto de variáveis inteiras em N subconjuntos disjuntos Q_i , $i = 1, \dots, N$;
 - 2: relaxe todas as variáveis inteiras do problema;
 - 3: **enquanto** $i < N$ **faça**
 - 4: retorne as variáveis do conjunto Q_i ao seu estado original;
 - 5: resolva o modelo resultante;
 - 6: **se** uma solução viável for obtida **então**
 - 7: fixe as variáveis do conjunto Q_i com os valores obtidos;
 - 8: **senão**
 - 9: **retorne**: “solução inviável”;
 - 10: **fim se**;
 - 11: **fim enquanto**;
 - 12: **retorne** a solução viável obtida;
-

É importante observar que a heurística *relax-and-fix* não apresenta garantia de viabilidade para cada subproblema. Neste contexto, Escudero e Salmeron [2005] propuseram que, se em uma das iterações uma solução viável não for encontrada, o algoritmo retirasse os valores fixados na iteração anterior, tentando resolver novamente a iteração em que a inviabilidade foi detectada. Assim, no pior caso, o problema original é resolvido em uma única iteração.

5.1.2 GRASP

GRASP, *Greedy Randomized Adaptive Search Procedure* ou Procedimento de busca adaptativa gulosa e randomizada, é um método iterativo, proposto por Feo e Resende [1995], que consiste de duas fases: uma fase de construção, na qual uma solução inicial é gerada e uma fase de busca local, na qual um ótimo local na vizinhança da solução construída é pesquisado. A melhor solução encontrada ao longo de todas as

iterações GRASP realizadas é retornada como resultado. O pseudocódigo descrito pelo Algoritmo 5.2 ilustra o procedimento GRASP.

Algoritmo 5.2 Procedimento *GRASP* Básico

```

1: repita
2:    $s \leftarrow Construc\alpha o(\alpha)$ ;   {onde  $\alpha$  é um parâmetro a ser definido}
3:    $s \leftarrow BuscaLocal(s)$ ;
4:   se a solução  $s$  for melhor do que  $s^*$  então
5:      $s^* \leftarrow s$ ;
6:   fim se;
7: até que  $iter_{max}$  iterações sejam efetuadas
8: retorne:  $s^*$ ;

```

Na fase de construção, uma solução é iterativamente construída, elemento por elemento. A cada iteração dessa fase, os próximos elementos candidatos a serem incluídos na solução são colocados em uma lista C de candidatos, seguindo um critério de ordenação pré-determinado. Esse processo de seleção é baseado em uma função adaptativa gulosa $g : C \rightarrow \mathbb{R}$, que estima o benefício da seleção de cada um dos elementos. A heurística é adaptativa porque os benefícios associados com a escolha de cada elemento são atualizados em cada iteração da fase de construção para refletir as mudanças oriundas da seleção do elemento anterior. A componente probabilística do procedimento reside no fato de que cada elemento é selecionado de forma aleatória a partir de um subconjunto restrito formado pelos melhores elementos que compõem a lista de candidatos. Este subconjunto recebe o nome de lista de candidatos restrita (RCL - *restricted candidate list*). Esta técnica de escolha permite que diferentes soluções sejam geradas em cada iteração GRASP. O pseudocódigo representado pela Algoritmo 5.3 descreve a fase de construção GRASP.

O parâmetro α do algoritmo exibido na Figura 5.3 controla o nível de “gulosidade” (aleatoriedade) do procedimento $Construc\alpha o(.)$, sendo que $\alpha \in [0, 1]$. Um valor $\alpha = 0$ faz gerar soluções puramente gulosas, enquanto $\alpha = 1$ faz produzir soluções totalmente aleatórias.

Assim como em muitas técnicas determinísticas, as soluções geradas pela fase de construção do GRASP provavelmente não são localmente ótimas com respeito à definição de vizinhança adotada. Daí a importância da fase de busca local, a qual objetiva

Algoritmo 5.3 Procedimento *Construcao*(α) do GRASP

```

1: Seja  $s$  uma solução inicial vazia;
2: Seja  $g(\cdot)$  uma função que retorna o valor (custo) de um elemento;
3: enquanto  $s$  não for uma solução completa faça
4:   classifique os elementos da solução por um critério guloso e adicione-os a uma
     lista  $C$ ;
5:    $g_{min} \leftarrow$  valor do melhor elemento da lista  $C$ ;
6:    $g_{max} \leftarrow$  valor do pior elemento da lista  $C$ ;
7:    $RCL \leftarrow \{e \in C \text{ tais que } g(e) \leq g_{min} + \alpha \times (g_{max} - g_{min})\}$ 
8:   selecione aleatoriamente um elemento  $e \in RCL$  e o inclua em  $s$ ;
9: fim enquanto;
10: retorne:  $s$ ;

```

melhorar a solução construída. O Algoritmo 5.4 descreve o pseudocódigo de um procedimento básico de busca local com respeito a uma certa vizinhança $N(\cdot)$ de s .

Algoritmo 5.4 Procedimento *BuscaLocal*($N(\cdot), s_0$) do GRASP

```

1:  $s^* \leftarrow s_0$ ; {solução inicial, passada por parâmetro};
2:  $V \leftarrow$  conjunto de soluções vizinhas  $N(s^*)$ ;
3: enquanto  $|V| > 0$  faça
4:   selecione uma solução  $s \in V$  qualquer;
5:   se a solução  $s$  for melhor que  $s^*$  então
6:      $s^* \leftarrow s$ ;
7:      $V \leftarrow$  conjunto de soluções vizinhas  $N(s^*)$ ;
8:   senão
9:      $V \leftarrow V \setminus \{s\}$ ;
10:  fim se;
11: fim enquanto;
12: retorne:  $s^*$ ;

```

A eficiência da busca local depende da qualidade da solução construída. O procedimento de construção tem então um papel importante na busca local, uma vez que as soluções construídas constituem bons pontos de partida para a busca local, permitindo assim acelerá-la. O parâmetro α , que determina o tamanho da lista de candidatos restrita, é basicamente o único parâmetro a ser ajustado na implementação de um procedimento GRASP. Em Feo e Resende [1995], discute-se o efeito do valor de α na qualidade da solução e na diversidade das soluções geradas durante a fase de construção. Valores de α que levam a uma lista de candidatos restrita de tamanho muito

limitado (ou seja, α com valor próximo de 0) implicam em soluções finais de qualidade muito próxima àquela obtida de forma puramente gulosa, obtidas com um baixo esforço computacional. Em contrapartida, provocam uma baixa diversidade de soluções construídas. Já uma escolha de α próxima da seleção puramente aleatória leva a uma grande diversidade de soluções construídas mas, por outro lado, muitas das soluções construídas são de qualidade inferior, tornando mais lento o processo de busca local.

O algoritmo GRASP procura, portanto, conjugar bons aspectos dos algoritmos puramente gulosos, com aqueles dos procedimentos aleatórios de construção de soluções. Procedimentos GRASP mais sofisticados incluem estratégias adaptativas para o parâmetro α . O ajuste desse parâmetro ao longo das iterações GRASP, por critérios que levam em consideração os resultados obtidos nas iterações anteriores, produz soluções melhores do que aquelas obtidas considerando-o fixo [Prais e Ribeiro, 1999, 2000].

5.1.3 ILS

O método *Iterated Local Search* (ILS) é baseado na idéia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local.

Para aplicar um algoritmo ILS, quatro componentes têm que ser especificadas: (a) Procedimento *SolucaoInicial*(.), que gera uma solução inicial s_0 para o problema; (b) Procedimento *BuscaLocal*(s'), que retorna uma solução s'' possivelmente melhorada em relação a s' ; (c) Procedimento *Perturbacao*(s), que modifica a solução corrente s guiando a uma solução intermediária s' e (d) Procedimento *CriterioAceitacao*(.), que decide de qual solução a próxima perturbação será aplicada.

No Algoritmo 5.5 mostra-se o pseudocódigo do ILS básico.

O sucesso do ILS é centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha do método de busca local, das perturbações e do critério de aceitação. Em princípio, qualquer método de busca local pode ser usado, mas o desempenho do ILS com respeito à qualidade da solução final e a velocidade de convergência depende fortemente do método escolhido. Normalmente um método de descida é usado,

Algoritmo 5.5 Procedimento ILS Básico

```
1:  $s_0 \leftarrow SolucaoInicial()$ ;  
2:  $s \leftarrow BuscaLocal(s_0)$ ;  
3: enquanto critério de parada não for satisfeito faça  
4:    $s' \leftarrow Perturbacao(s)$ ;  
5:    $s'' \leftarrow BuscaLocal(s')$ ;  
6:    $s \leftarrow CriterioAceitacao(s, s'')$ ;  
7: fim enquanto;  
8: retorne:  $s^*$ ;
```

mas também é possível aplicar algoritmos mais sofisticados, tais como Busca Tabu ou outras metaheurísticas.

A intensidade da perturbação deve ser forte o suficiente para permitir escapar do ótimo local corrente e permitir explorar diferentes regiões. Ao mesmo tempo, ela precisa ser fraca o suficiente para guardar características do ótimo local corrente.

O critério de aceitação é usado para decidir de qual solução se continuará a exploração, bem como qual será a perturbação a ser aplicada. Um aspecto importante do critério de aceitação e da perturbação é que eles induzem aos procedimentos de intensificação e diversificação. A intensificação consiste em permanecer na região do espaço onde a busca se encontra, procurando explorá-la de forma mais efetiva; enquanto a diversificação consiste em se deslocar para outras regiões do espaço de soluções. A intensificação da busca no entorno da melhor solução encontrada é obtida, por exemplo, pela aplicação de “pequenas” perturbações sobre ela. A diversificação, por sua vez, pode ser realizada aceitando-se quaisquer soluções s'' e aplicando “grandes” perturbações na solução ótima local.

Um critério de aceitação comumente utilizado é mover-se para o ótimo local s'' somente se ele for melhor que o ótimo local corrente s , isto é, somente se $f(s'') < f(s)$ em um problema de minimização, ou se $f(s'') > f(s)$ em um problema de maximização.

5.1.4 VND

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto por Mladenović e Hansen [1997], é um método de refinamento que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada [Souza, 2009]. O pseudo-código deste método, em que se considera o refinamento de uma solução s utilizando uma função de avaliação f , a ser minimizada, e um conjunto de r diferentes vizinhanças $N = \{N^{(1)}, N^{(2)}, \dots, N^{(r)}\}$, é apresentado pelo Algoritmo 5.6.

Algoritmo 5.6 VND Básico

```

1: seja  $r$  o número de estruturas de vizinhanças diferentes;
2:  $k \leftarrow 1$ ; // tipo de estrutura de vizinhança corrente
3: enquanto  $k \leq r$  faça
4:   encontre um vizinho  $s' \in N^{(k)}(s)$  de melhora; se não encontrar faça  $s \leftarrow \emptyset$ ;
5:   se ( $s = \emptyset$ ) então
6:      $k \leftarrow k + 1$ ;
7:   senão
8:      $k \leftarrow 1$ ;
9:   fim se;
10: fim enquanto;
11: retorne:  $s$ ;

```

Dependendo do problema abordado, a busca pelo melhor vizinho pode ser cara computacionalmente. Nesta situação é comum fazer a busca pela primeira solução de melhora. Outra alternativa, bastante utilizada, é aplicar explorar a vizinhança de forma aleatória, ou seja, executar a busca em ordem aleatória, ao invés de executá-la de forma sequencial.

Segundo os autores, o método VND baseia-se em três princípios básicos:

- Um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança;

- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
- Para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximas.

Ainda de acordo com Mladenović e Hansen [1997], o último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Este é o caso em que os ótimos local e global compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

5.2 Heurística *Relax-And-Fix* aplicada ao FP

A heurística *relax-and-fix* foi implementada sobre a formulação matemática apresentada no Capítulo 4. Na implementação utilizada neste trabalho, apenas as variáveis inteiras são fixadas, dando assim maior flexibilidade na obtenção de soluções viáveis.

Duas estratégias de particionamento foram implementadas na adaptação da heurística *relax-and-fix* para o FP. Inicialmente, foi proposto um critério de seleção aleatório, ou seja, as variáveis do problema são divididas de forma aleatória em n conjuntos Q_i , $i = 1, \dots, n$, de mesmo tamanho. Doravante, esta implementação será referenciada como RF-Rnd. Uma segunda abordagem utilizou o conceito de tempo para particionar as variáveis. Nesta estratégia, cada conjunto Q_i de variáveis inteiras abrange as variáveis de um dado período de tempo. Assim, um total de $|T|$ conjuntos são gerados, onde T representa o conjunto de estágios de tempo. A esta abordagem é atribuída o acrônimo RF-Per.

Na abordagem RF-Per, duas formas de fixação de variáveis são utilizadas, sendo que ambas particionam o conjunto de variáveis inteiras de acordo com os estágios de tempo T . A primeira estratégia, denominada RF-PerA, relaxa e fixa as variáveis do primeiro ao último período de tempo. A estratégia RF-PerB, por outro lado, utiliza a ordenação inversa.

Uma vez que na heurística *relax-and-fix* não existe garantia de viabilidade, a estratégia proposta por Escudero e Salmeron [2005] foi utilizada (vide Seção 5.1.1). Assim, se em uma das iterações uma solução viável não for encontrada, o algoritmo ignora os valores fixados na iteração anterior, e re-executa a iteração corrente sobre a união dos dois subconjuntos de variáveis inteiras envolvidas. No pior caso, o algoritmo será executado sobre o conjunto não particionado de variáveis, resolvendo assim todo o problema original numa única iteração.

Se durante as iterações do *relax-and-fix*, uma variável relaxada, que no problema original era inteira, assumir um valor inteiro, esta também é fixada. O objetivo desta estratégia é agilizar ainda mais a heurística, reduzindo o tamanho dos subproblemas gerados pelas iterações posteriores.

5.3 Heurística GRASP-ILS aplicada ao FP

Neste trabalho foi desenvolvida uma heurística híbrida para o FP, doravante denominada GRASP-ILS, que combina fundamentos e ideias das metaheurísticas GRASP e ILS (seções 5.1.2 e 5.1.3) e utiliza um procedimento VND (Seção 5.1.4) como busca local. Nesta abordagem híbrida, o algoritmo ILS adaptado ao FP é utilizado na fase de busca local do GRASP. O GRASP implementado neste trabalho é apresentado no Algoritmo 5.2, da página 58.

O procedimento de geração de soluções iniciais é apresentado na Seção 5.3.2. As estratégias de vizinhança utilizadas são discutidas na Seção 5.3.3, enquanto a busca local e o algoritmo ILS utilizado são explanados nas seções ?? e 5.3.4.

5.3.1 Representação de uma Solução

De forma análoga à formulação matemática apresentado no Capítulo 4, uma solução s para o FP é representada por um conjunto de variáveis x_{ijt}^r , representando a quantidade do produto primário i a ser utilizado na composição do produto final j , sendo transportado pela rota r no instante de tempo t . Ainda seguindo esta analogia, as res-

trições consideradas pelos modelos heurísticos são as mesmas consideradas pelo modelo matemático, podendo ser verificadas nas equações (4.1) - (4.21).

5.3.2 Geração de Soluções Iniciais

Inicialmente, ocorre uma fase de pré-processamento em que as rotas são agrupadas de acordo com os pontos que as compõem. Uma vez definidas todas as rotas possíveis entre cada par Produto Primário $\times$ Produto Final, uma solução inicial é gerada.

Esta geração é feita por meio de um procedimento parcialmente guloso, em que os primeiros estágios de tempo são analisados primeiro. Os Produtos Finais são ordenados de acordo com sua urgência (com data limite de atendimento à demanda mais próxima), sendo os mais urgentes melhor classificados. Um segundo critério de classificação é a quantidade mínima a ser obrigatoriamente atendida. Quanto maior for esta quantidade, melhor será a classificação.

Destes Produtos Finais, os $\alpha\%$ melhor classificados são colocados em uma Lista Restrita de Candidatos (*RCL*). Desta lista, um Produto é escolhido ao acaso para ser composto. Para compor este produtos, os Produtos Primários que podem ser blendados e que utilizam rotas disponíveis são classificados de acordo com a sequência de critérios definidos a seguir, nesta ordem:

1. Produto Primário com utilização mínima ainda não satisfeita;
2. Produto Primário que utilize rotas em que o limite inferior de utilização ainda não tenha sido atingido;
3. Produto Primário que minimize os desvios dos parâmetros de qualidade;

A definição do Produto Primário a ser utilizado é feito de forma análoga à escolha do Produto Final. Os $\alpha\%$ melhor classificados são submetidos a um sorteio, sendo que o sorteado é o produto escolhido. A rota a ser utilizada deve também ser definida. Para realizar esta seleção, as rotas são ordenadas de acordo com os critérios a seguir:

1. Rota que não tenha seu limite inferior de utilização atingido;
2. Rota que minimize os desvios dos parâmetros de qualidade do Produto Primário selecionado (considerando os fatores de manuseio);
3. Rota com menor custo;

Uma vez que todas as restrições de utilização mínima dos Produtos Primários e das rotas tenham sido satisfeitas, a escolha do Produto Primário passa a considerar apenas a qualidade, ou seja, os produtos que atenderem melhor aos parâmetros de controle de qualidade são utilizados.

A cada iteração do método, a lista de candidatos é atualizada, através da remoção do último Produto Final analisado e o procedimento se repete até que todos sejam analisados. Para formar a lista de Produtos Primários candidatos, algumas condições devem ser satisfeitas. O produto deve pertencer à lista de produtos blendáveis com aquele Produto Final, deve estar disponível na quantidade mínima exigida para entrar na composição do Produto Final e as rotas pelas quais ele pode passar devem ser capazes de transportar no mínimo esta quantidade.

5.3.3 Estruturas de Vizinhança

A definição das estruturas de vizinhança a serem utilizadas é crucial no desempenho e na eficiência de uma heurística de refinamento. Idealmente, de uma solução s do espaço de soluções deve ser sempre possível atingir qualquer outra solução s' em um número finito de passos, utilizando um ou mais tipos de movimento.

De forma a diversificar o espaço de busca, diversas estruturas de vizinhança foram definidas para o FP:

1. ***Change Day***: troca o dia de uma alocação;
2. ***Change Route***: troca a rota pela qual um Produto Primário é transportado, adequando a quantidade à nova rota;

3. ***Change Product***: troca o Produto Primário em uma blendagem para gerar um Produto Final.

É importante notar que o número de movimentos possíveis nas estruturas de vizinhança definidas em geral não é elevado. Isso porque, nas instâncias baseadas na realidade, muitos destes movimentos geram soluções inviáveis.

Nas estruturas de vizinhança consideradas, além de movimentos de realocação foram desenvolvidos também movimentos de troca.

Uma método de busca local baseado no algoritmo VND (vide Seção 5.1.4) foi desenvolvido, considerando todas as estruturas de vizinhança. Para agilizar a busca, o primeiro vizinho de melhora encontrado é aceito. Outro detalhe a respeito da implementação utilizada é que a busca é sempre feita em ordem aleatória, o que aumenta a diversidade das soluções geradas.

5.3.4 Busca Local

No algoritmo GRASP-ILS, o método ILS desenvolvido funciona como a busca local do algoritmo GRASP. O pseudocódigo do algoritmo ILS implementado é dado pelo Algoritmo 5.5, apresentado na página 61.

A busca local utilizada pelo ILS é descrita na Seção 5.3.3, e utiliza as estruturas de vizinhança apresentadas. Como critério de parada, é definido um número fixo de iterações.

As perturbações consistem em zerar o valor de algumas variáveis de decisão, escolhidas ao acaso, e re-executar o procedimento de geração de solução inicial a partir daquele ponto. O número de variáveis a serem zeradas define o nível de perturbação. Assim, quanto maior o nível de perturbação, maior será o número de variáveis zeradas.

Quanto ao critério de aceitação de uma solução, é utilizado o procedimento de avaliação explicado na Seção 5.3.5, sendo aceitas apenas soluções de melhora.

5.3.5 Avaliação Multiobjetivo

Na Seção 4.1 foram apresentados os conceitos iniciais sobre otimização multiobjetivo. Na metodologia heurística implementada, os mesmo conceitos podem ser aplicados. Assim, o algoritmo deveria identificar e armazenar as soluções não-dominadas, assumindo que estas estão mais próximas das soluções Pareto-ótimas. Entretanto, uma vez que no FP existe uma hierarquia bem definida entre as funções objetivo, é possível definir uma função não-linear de ordenação. O Algoritmo 5.7 apresenta o pseudocódigo do método de ordenação de soluções, assumindo que a função objetivo f_i é sempre prioritária em relação à função f_{i+1} , $\forall i = 1, \dots, n - 1$.

Algoritmo 5.7 Procedimento de Avaliação de Soluções

- 1: sejam x_1 e x_2 as soluções em avaliação;
 - 2: **para** $i = 1$ até n **faça**
 - 3: **se** $f_i(x_1) < f_i(x_2)$ **então**
 - 4: **retorne:** solução x_1 domina a solução x_2 ;
 - 5: **senão se** $f_i(x_2) < f_i(x_1)$ **então**
 - 6: **retorne:** solução x_2 domina a solução x_1 ;
 - 7: **fim se**;
 - 8: **fim para**;
 - 9: **retorne:** soluções possuem mesmos valores;
-

Capítulo 6

Resultados Obtidos

Diversos testes experimentais, considerando diferentes cenários e horizontes de planejamento foram feitos para se fazer um estudo das diferentes metodologias desenvolvidas.

(Robson, está faltando apenas esta seção. Ainda estou terminando de executar os testes, e acredito que até segunda ou terça terei este capítulo completo. Estou incluindo tudo sobre as instâncias e, por isso, deve ficar bem extenso).

Capítulo 7

Considerações Finais

Neste trabalho foram propostas soluções para tratar o Problema do Planejamento do Fluxo de Produto (FP) visando múltiplos objetivos simultaneamente: (i) minimizar o não atendimento às demandas, (ii) minimizar o não atendimento às metas de qualidade e (iii) minimizar custos com transporte. Para tanto, inicialmente foi desenvolvido um modelo multiobjetivo de programação matemática. Este modelo mostrou-se eficaz na resolução de instâncias do horizonte mensal e anual, não sendo capaz de resolver instâncias que consideram o dia-a-dia.

Para tratar estas instâncias, abordagens heurísticas foram consideradas. Uma heurística *relax-and-fix* e um algoritmo baseado nas metaheurísticas GRASP e ILS foram propostos. Estas metodologias mostraram-se adequadas no tratamento das instâncias que consideram o dia-a-dia, gerando boas soluções rapidamente.

Finalmente, com o objetivo de avaliar as abordagens propostas, instâncias baseadas na realidade de uma grande empresa mineradora brasileira foram utilizadas nos testes. As soluções - tão rapidamente obtidas - foram, então, analisadas pelos planejadores da empresa, que aprovaram a qualidade das mesmas. Assim, este trabalho contribui com a diminuição drástica do tempo de tomada da decisão, além de disponibilizar ao planejador a possibilidade de simular diversos cenários, tarefa impossível de ser feita em tempo hábil de forma manual.

Outro aspecto que valida a importância deste trabalho é a maior interação entre

a universidade e empresas mineradoras da região, possibilitando a formação de profissionais especialistas em otimização de processos. Adicionalmente, contribui com a divulgação, para o meio empresarial, de métodos de otimização na abordagem de problemas típicos de empresas de mineração.

Referências Bibliográficas

- Agência Nacional de Transportes Terrestres (2009). Transporte ferroviário. In *www.antt.gov.br*, acessado em junho de 2009.
- Aires, M. A. C.; Joly, M.; Rocha, R.; Smania Filho, P. e Fampa, M. H. C. (2005). Programação da produção de gasolina em refinaria: modelagem matemática e um algoritmo de solução. *Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional*, 1:2140–2151.
- Alves, J. M.; Guimarães, F. A.; Toffolo, T. A. M. e Souza, M. J. F. (2007). Um sistema de otimização para o planejamento da produção e vendas de um empresa mineradora. *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional*, 1:12.
- Arenales, M.; Armentano, V.; Morabito, R. e Yanasse, H. (2007). *Pesquisa Operacional Para Cursos de Engenharia. Rio de Janeiro: Editora Campus*. Editora Campus, Rio de Janeiro.
- Chanda, E. K. C. e Dagdelen, K. (1995). Optimal blending of mine production using goal programming and interactive graphics systems. *International Journal of Surface Mining, Reclamation and Environment*, 9:203–208.
- Charnes, A. e Cooper, W. W. (1961). *Management models and industrial applications of linear programming*. Wiley, New York.
- Costa, F. P. (2005). *Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto*. Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto.

- Costa, F. P.; Souza, M. J. F. e Pinto, L. R. (2004). Um modelo de alocação dinâmica de caminhões. *Brasil Mineral*, 231:26–31.
- Costa, F. P.; Souza, M. J. F. e Pinto, L. R. (2005). Um modelo de programação matemática para alocação estática de caminhões visando ao atendimento de metas de produção e qualidade. *Revista Escola de Minas*, 78(1):77–88.
- Dillenberger, C.; Escudero, L. F.; Wollensak, A. e Zhang, W. (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operations Research*, 75(1):275–286.
- Escudero, L. F. e Salmeron, J. (2005). On a relax-and-relax framework for a class of project scheduling problems. *Annals of Operations Research*, 140:163–188.
- Everett, J. (2001). Iron ore production scheduling to improve product quality. *European Journal of Operational Research*, 129(1):355–361.
- Federgruen, A.; Meissner, J. e Tzur, M. (2007). Progressive interval heuristics for multiitem capacitated lot sizing problems. *Operations Research*, 55(3):490–502.
- Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.
- Ferreira, D.; Morabito, R. e Rangel, S. (2005). Aplicação de um modelo de otimização multi-item multimáquina na programação da produção em uma fábrica de bebidas. *Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional*, 1:2473–2484.
- Guimaraes, I. F.; Souza, M. J. F. e Pantuza Júnior, G. (2007). Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. In *Anais do XIV SIMPEP*, p. 11p.
- Guimarães, F. A. C.; Souza, M. J. F.; Costa, T. A. e Costa, P. F. (2006a). Iterated local search aplicado ao planejamento operacional de lavra em minas a céu aberto considerando alocação dinâmica de caminhões. *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, 1:1369–1380.

- Guimarães, F. A. C.; Souza, M. J. F.; Costa, T. A.; Costa, P. F. e do Carmo Bento Alves, J. M. (2006b). Iterated local search aplicado ao planejamento operacional de lavra em minas a céu aberto considerando alocação estática de caminhões. *Anais do IV Congresso Brasileiro de Mina a Céu Aberto, IBRAM, Belo Horizonte*, 1:1–16.
- Instituto Brasileiro de Mineração (2009). Produção mineral brasileira. In <http://www.ibram.org.br>, acessado em maio de 2009.
- Junqueira, R. A. R. e Morabito, R. (2006). Um modelo de otimização linear para o planejamento agregado da produção e logística de sementes de milho. *Revista Produção*, 16(3):510–525.
- Kimms, A.; Toledo, C. F. M. e França, P. M. (2005). Modelo conjunto de programação da produção e dimensionamento de lotes aplicado a uma indústria de bebidas. *Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional*, 1:1947–1958.
- Lee, S. M. (1972). *Goal programming for decision analysis*. Auerback, Philadelphia.
- Lourenço, H. R.; Martin, O. e Stützle, T. (2003). Iterated local search. In Glover, F. e Kochenberger, G., editores, *Handbook of Metaheuristics*, pp. 321–353. Kluwer Academic Publishers.
- Lu, J.; Fu, M. e Sha, J. (2005). Research of import iron ore logistics system based on the minimum cost theory. In *Proceedings of the International Conference on Services Systems and Services Management, Chongqing, China*, pp. 391–396.
- Mateus, G. R.; Luna, H. P. L.; Fontelle Júnior, J.; Vieira, R. L. F.; Costa, R. A. V. e do Patrocínio Júnior, Z. K. G. (1994). Transporte de minério de ferro na mbr: caracterização e modelo. In *Departamento de Ciência da Computação, UFMG, Belo Horizonte, MG, Tech. Rep. RT-009/94*.
- Merschmann, L. H. C. (2002). *Desenvolvimento de um Sistema de Otimização e Simulação para Cenários de Produção em Minas a Céu Aberto*. COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

- Mladenović, N. e Hansen, P. (1997). Variable Neighborhood Search. *Computers and Operations Research*, 24:1097–1100.
- Moraes, E. F.; Alves, J. M.; Souza, M. J. F.; Cabral, I. E. e Martins, A. X. (2005). Um modelo de programação matemática para otimizar a composição de lotes de minério de ferro da mina cauê da cvrd. *Revista Escola de Minas*, 59:299–306.
- Paiva, R. P. O. e Morabito, R. (2006). Modelagem matemática de otimização aplicada ao planejamento agregado da produção em usinas de açúcar e Álcool: Formulação e resultados. *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, 1:24–34.
- Pimentel, S. B.; Mateus, G. R. e Almeida, F. A. (2009). Mathematical models for optimizing the global mining supply chain. Nag, B., editor, *Intelligent Systems in Operations: Models, Methods and Applications*. IGI Global.
- Prais, M. e Ribeiro, C. (1999). Parameter variation in GRASP implementations. In *Proceedings of the Third Metaheuristics International Conference*, Angra dos Reis, Brazil.
- Prais, M. e Ribeiro, C. (2000). Reactive GRASP: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, 12:164–176.
- Rodrigues, L. F. (2006). Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto. Dissertação de mestrado, Programa de Pós-Graduação em Engenharia de Produção/UFGM, Belo Horizonte.
- Schofield, C. G. (1980). Homogenization/blending systems design and control for minerals processing. In *Trans Tech Publications*.
- Souza, M. J. F. (2009). Inteligência computacional para otimização - notas de aula. In *Technical Report, UFOP - Ouro Preto*.
- Takahashi, R. H. C. (2007). Otimização escalar e vetorial - notas de aula. In *Technical Report 1, UFGM - Belo Horizonte*.

Wolsey, L. A. (1998). *Integer Programming*. Wiley, New York.



MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
Escola de Minas da Universidade Federal de Ouro Preto
Departamento de Engenharia de Minas
Programa de Pós-Graduação em Engenharia Mineral - PPGEM



UM ALGORITMO HEURÍSTICO HÍBRIDO PARA MINIMIZAR OS CUSTOS COM A ANTECIPAÇÃO E O ATRASSO DA PRODUÇÃO EM AMBIENTES COM JANELAS DE ENTREGA E TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQUÊNCIA

Autor: Puca Huachi Vaz Penna

Orientador: Marcone Jamilson Freitas Souza

Trabalho de Dissertação apresentado ao Programa de Pós-Graduação em Engenharia Mineral do Departamento de Engenharia de Minas da Escola de Minas da Universidade Federal de Ouro Preto, como parte integrante dos requisitos para a obtenção do título de Mestre em Engenharia Mineral.

Área de concentração: Lavra de Minas.

Ouro Preto, Março de 2009

**UM ALGORITMO HEURÍSTICO HÍBRIDO PARA
MINIMIZAR OS CUSTOS COM A ANTECIPAÇÃO E O
ATRASO DA PRODUÇÃO EM AMBIENTES COM
JANELAS DE ENTREGA E TEMPOS DE PREPARAÇÃO
DEPENDENTES DA SEQUÊNCIA**

Puca Huachi Vaz Penna

Esta dissertação foi apresentada em sessão pública e aprovada em 03 de março de 2009, pela Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Marcone Jamilson Freitas Souza (Orientador - DECOM/UFOP)

Prof. Dr. Carlile Campos Lavor (DMA/UNICAMP)

Prof. Dr. Frederico Gadelha Guimarães (DECOM/UFOP)

Agradecimentos

“A natureza é exatamente simples, se conseguirmos encará-la de modo apropriado. Essa crença tem-me auxiliado, durante toda a minha vida, a não perder as esperanças, quando surgem grandes dificuldades de investigação.”
[Albert Einstein]

Gostaria de expressar meus agradecimentos a todos, que de alguma forma, contribuíram para a realização deste trabalho, em especial:

A Deus pelo dom da vida.

Aos meus pais Tattu Penna e Neide Vaz que me ensinaram que o verdadeiro alicerce de nossas vidas é a família.

Às minhas irmãs Uiara, Violeta Camélia, aos irmãos Taiandir e Rairu e aos meus tios, tias, primos e primas, que juntos, me fazem entender e acreditar nos ensinamentos de meus pais.

À Temis, meu amor, que com seu apoio e carinho tem deixado nossas vidas mais felizes.

Ao meu orientador Prof. Marcone pela amizade e ensinamentos, que por sua dedicação a docência e a pesquisa ter despertado em mim esse gosto.

Aos amigos do mestrado, em especial ao Fred, por caminharmos juntos.

Aos professores do Programa de Pós-graduação em Engenharia Mineral da UFOP pelos ensinamentos.

Ao Programa de Pós-graduação em Engenharia Mineral da UFOP pela oportunidade.

À república Quarto Crescente pela amizade.

Resumo

Este trabalho de dissertação tem seu foco no problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. São considerados tempos de preparação da máquina dependentes da sequência de produção, bem como a existência de janelas de entrega distintas. Para resolução do problema, desenvolveu-se um algoritmo heurístico de três fases. A primeira fase baseada em GRASP e Descida em Vizinhança Variável para a geração da solução inicial, a segunda fase baseada em Busca Tabu para refinamento da solução, e por fim, a Reconexão por Caminhos como estratégia de pós-otimização, na terceira fase. Para cada sequência gerada pela heurística é utilizado um algoritmo de tempo polinomial para determinar a data ótima de início de processamento de cada tarefa. Os resultados computacionais mostraram que houve melhoria em relação a um algoritmo da literatura, tanto com relação à qualidade da solução final quanto em relação ao desvio médio.

Palavras-Chaves: Sequenciamento em uma máquina; GRASP; Busca Tabu; Descida em Vizinhança Variável; Reconexão por Caminhos.

Abstract

This work deals with the single machine scheduling problem with earliness and tardiness penalties. Sequence dependent setup times and distinct due windows are considered. To solve this problem, a three-phase heuristic approach was developed. The first phase is based on GRASP and Variable Neighborhood Descent to generate an initial solution; the second phase is based on a Tabu Search for solution refining, and finally Path Relinking is used as a mechanism of post-optimization as a third phase. For each job sequence generated by the heuristic, an optimal timing algorithm is used to determine the completion time for each job in the job sequence. Computational experiments carried out show that previous algorithms found in related literature have been improved, regarding the quality of the final solution and the average gap.

Keywords: *Single Machine Scheduling; GRASP; Tabu Search; Variable Neighborhood Descent; Path Relinking.*

Sumário

1	Preliminares	1
1.1	Introdução	1
1.2	Justificativa e Relevância	6
1.3	Objetivos	7
1.3.1	Objetivo Geral	7
1.3.2	Objetivos Específicos	7
1.4	Organização do Trabalho	7
2	O Problema de Sequenciamento	9
2.1	Tipos de Problemas	9
2.2	O Problema de Sequenciamento na Mineração	10
2.3	Problema Abordado	17
3	Revisão Bibliográfica	19
3.1	Introdução	19
4	Metodologia	26
4.1	Modelo Matemático	29
4.2	Procedimento de Determinação das Datas Ótimas de Início de Processamento das Tarefas	31

4.3	Modelo Heurístico	38
4.3.1	Representação de uma Solução	38
4.3.2	Vizinhança de uma Solução	38
4.3.3	Função de Avaliação	39
4.3.4	Geração da solução inicial	40
4.3.5	Busca Tabu aplicada ao PSUMAA	43
4.3.6	Reconexão por Caminhos	46
4.3.7	Algoritmo GTSPR	51
5	Resultados Computacionais	52
5.1	Problemas-teste	52
5.2	Resultados do PLIM	53
5.3	Calibração e Parâmetros do Método Heurístico	54
5.4	Resultados do Algoritmo Heurístico	59
6	Conclusões e Trabalhos Futuros	65
	Referências Bibliográficas	66
I	Resultados Detalhados	71
I.1	Execução do GTSPR I	71
I.2	Execução do GTSPR II	76
II	Publicações	81

Lista de Tabelas

2.1	Representação de uma solução do problema de alocação de caminhões	12
2.2	Solução de um problema de alocação de caminhões	13
2.3	Tempo de operação dos caminhões	14
2.4	Sequenciamento da Solução da Frente M4	16
2.5	Sequenciamento da Solução da Frente E1	16
2.6	Exemplificação dos dados de entrada	18
5.1	Resultados do PLIM	53
5.2	Parâmetros do Algoritmo GTSPR - Solução Completa	59
5.3	Comparação de resultados de cada fase do algoritmo GTSPR $\times$ Gomes Jr. <i>et al.</i> (2007) $\times$ CPLEX (Parâmetros Principais)	61
5.4	Comparação de resultados de cada fase do algoritmo GTSPR $\times$ Gomes Jr. <i>et al.</i> (2007) $\times$ CPLEX (Parâmetros Secundários)	62
5.5	Comparação dos tempos computacionais GTSPR I $\times$ GTSPR II $\times$ Gomes Jr. <i>et al.</i> (2007) corrigido	64
I.1	Resultados detalhados GTSPR I $\times$ Gomes Jr. <i>et al.</i> (2007) (8 a 10 tarefas)	72
I.2	Resultados detalhados GTSPR I $\times$ Gomes Jr. <i>et al.</i> (2007) (11 a 15 tarefas)	73

I.3	Resultados detalhados GTSPR I $\times$ Gomes Jr. <i>et al.</i> (2007) (20 a 30 tarefas)	74
I.4	Resultados detalhado GTSPR I $\times$ Gomes Jr. <i>et al.</i> (2007) (40 a 75 tarefas)	75
I.5	Resultados detalhados GTSPR II $\times$ Gomes Jr. <i>et al.</i> (2007) (8 a 10 tarefas)	77
I.6	Resultados detalhados GTSPR II $\times$ Gomes Jr. <i>et al.</i> (2007) (11 a 15 tarefas)	78
I.7	Resultados detalhados GTSPR II $\times$ Gomes Jr. <i>et al.</i> (2007) (20 a 30 tarefas)	79
I.8	Resultados detalhado GTSPR II $\times$ Gomes Jr. <i>et al.</i> (2007) (40 a 75 tarefas)	80

Lista de Figuras

1.1	Exemplo de sequenciamento sem tempo ocioso entre tarefas	4
1.2	Exemplo de sequenciamento com tempo ocioso entre tarefas	4
2.1	Problema Estudado	18
4.1	Metodologia heurística de solução para o PSUMAA	29
4.2	Programação das tarefas de um bloco na sequência: 1º Passo	33
4.3	Programação das tarefas de um bloco na sequência: 2º Passo	34
4.4	Programação das tarefas de um bloco na sequência: 3º Passo	35
4.5	Programação das tarefas de um bloco na sequência: 4º Passo	36
4.6	Procedimento PDDOIP	37
4.7	Movimento de troca - N^T	38
4.8	Movimento de realocação - N^R	39
4.9	Movimento de realocação de um bloco - N^{Or}	39
4.10	Procedimento GRASP-VND aplicado ao PSUMAA	40
4.11	Procedimento Constroi_Solucao_GRASP aplicado ao PSUMAA	41
4.12	Procedimento VND ₁ aplicado ao PSUMAA	42
4.13	Procedimento Busca Tabu aplicado ao PSUMAA	45
4.14	Procedimento PR aplicado ao PSUMAA	50
4.15	Algoritmo GTSPR	51

5.1	Calibração do parâmetro $GRASP_{max}$ em função de Fo	54
5.2	Calibração do parâmetro $GRASP_{max}$ em função do tempo	55
5.3	Calibração do parâmetro MRD_{max} em função de Fo	55
5.4	Calibração do parâmetro MRD_{max} em função do tempo	56
5.5	Calibração do parâmetro TS_{max} em função de Fo	56
5.6	Calibração do parâmetro TS_{max} em função do tempo	57
5.7	Calibração do parâmetro $ T $ em função de Fo	57
5.8	Calibração do parâmetro $ T $ em função do tempo	57
5.9	Calibração do parâmetro $ GE $ em função de Fo	58
5.10	Calibração do parâmetro $ GE $ em função do tempo	58
5.11	Calibração do parâmetro $divElite$ em função de Fo	58
5.12	Calibração do parâmetro $divElite$ em função do tempo	59

Lista de Siglas

ACO	<i>Ant Colony Optimization</i>
AG	Algoritmos Genéticos
BT	Busca Tabu
FIFO	<i>First In First Out</i>
GE	Grupo Elite
GRASP	<i>Greedy Randomized Adaptive Search Procedures</i>
ILS	<i>Iterated Local Search</i>
LC	Lista de Candidatos
LRC	Lista Restrita de Candidatos
PCV	Problema do Caixeiro Viajante
PDDOIP	Procedimento de Determinação das Datas Ótimas de Início de Processamento das tarefas
PLIM	Programação Linear Inteira Mista
PO	Pesquisa Operacional
PR	<i>Path Relinking</i>
PSUM	Problema de sequenciamento em uma máquina
PSUMAA	Problema de sequenciamento em uma máquina com minimização das penalidades por antecipação e atraso
SA	<i>Simulated Annealing</i>
VND	<i>Variable Neighborhood Descent</i>
VNS	<i>Variable Neighborhood Search</i>

Capítulo 1

Preliminares

1.1 Introdução

A Pesquisa Operacional (PO) é a área do conhecimento que usa o método científico para a tomada de decisões, procurando determinar como melhor projetar e operar um sistema, usualmente sob condições que requerem a alocação de recursos escassos (Arenales *et al.*, 2007).

As aplicações em PO são diversas. Dentro do conjunto de problemas estudados encontra-se, com grande destaque, atualmente, a classe de problemas de programação de tarefas (*job scheduling*).

De acordo com França Filho (2007), o estudo de problemas de programação de tarefas envolvendo penalizações pela antecipação e pelo atraso é mais recente que aquele voltado para problemas em que o objetivo envolve uma função não-decrescente do instante de conclusão do processamento da tarefa, tais como tempo médio de fluxo, soma ponderada de atrasos e *makespan* (momento em que termina a execução da última tarefa). Para estes, custos mais elevados decorrem apenas do adiamento da conclusão das tarefas. Entretanto, com a filosofia *just-in-time* adotada por muitas empresas, o foco atual é penalizar também a conclusão das tarefas antes do instante em que elas são requeridas. Isto é justificado pelo fato de que concluir uma tarefa antecipadamente pode resultar em custos financeiros extras pela necessidade antecipada de capital e/ou espaço para armazenamento e/ou de outros recursos para manter e gerenciar o estoque.

Com relação às entregas, são encontrados na literatura três tipos de problemas:

1. os com datas de entrega comuns (*common due date*);
2. os com datas de entrega distintas (*distinct due dates*) e
3. os que têm janelas de entrega distintas (*distinct due windows*).

No primeiro tipo, existe uma única data para entregar todas as tarefas, enquanto no segundo existe uma data de entrega específica associada a cada tarefa. No terceiro tipo, há um período para a conclusão de cada tarefa. Segundo Wan e Yen (2002), esta última situação aparece em situações de incertezas ou tolerâncias com relação às datas de entrega e não há custos se as tarefas forem concluídas dentro da janela de entrega.

Para problemas de produção envolvendo penalizações pela antecipação e pelo atraso com datas comuns de entrega, há muitas propriedades que podem ser consideradas nos algoritmos de solução e que permitem uma redução acentuada do esforço computacional na exploração do espaço de busca. Por exemplo, de acordo com Baker e Scudder (1990), se C_i representa a data de conclusão da tarefa i e $f(C_i)$ o valor da função custo relativo à tarefa i , então na sequência ótima, os pontos $(s_i, f(C_i))$ formam um V (diz-se que a sequência é V – *shaped*). Em outras palavras, na sequência ótima, as tarefas antecipadas devem ser ordenadas de forma não-crescente pela relação entre o tempo de processamento e a penalização pela antecipação, enquanto as tarefas atrasadas devem ser ordenadas de forma não-decrescente pela relação entre o tempo de processamento e a penalização pelo atraso. Outra propriedade interessante é que na sequência ótima não há tempos ociosos entre as tarefas.

Para o caso em que as datas de entrega são distintas, assim como para o caso em há janelas de entrega distintas, tais propriedades não são necessariamente válidas, tornando mais complexa a exploração do espaço de busca. Para esses tipos de problemas é necessário saber se é permitida a ociosidade das máquinas ou não.

Segundo França Filho (2007), existem situações em que a ociosidade não é permitida por gerar custos maiores que aqueles decorrentes da conclusão antecipada das tarefas. Entretanto, há situações em que vale a pena manter uma máquina ociosa, mesmo que haja uma tarefa em condições de ter seu processamento iniciado. Para ilustrar esta situação, sejam duas tarefas i e j , que devem ser consecutivas em uma sequência de produção envolvendo uma máquina, às quais estão associados os custos unitários por atraso de 100 e 20, respectivamente, custos unitários por antecipação de 22 e 2, e que tenham como datas de entrega os instantes $T_i = 50$ e $T_j = 55$, respectivamente, e 15 e 25 unidades de tempo de processamento, respectivamente. Suponha que a máquina já esteja disponível desde o instante 20. Se a tarefa i começar no instante 20, será concluída no instante 35 e, portanto, haverá uma penalização igual a 330 ($= 15 \times 22$) pela antecipação. Nesse caso, a tarefa j poderá começar somente no instante 35. Considerando seu tempo de processamento, ela será concluída no instante 60, portanto 5 unidades de tempo atrasada em relação à data de entrega, resultando em um custo por atraso igual a 100 ($= 5 \times 20$), e um custo total de 430 ($= 330 + 100$), conforme pode ser visto na figura 1.1. Considere, agora, que o processamento da tarefa i seja iniciado no instante 35, mesmo sabendo que a máquina está disponível desde o instante 20. Neste caso, a tarefa i seria concluída no instante 50, sem multa nem por antecipação nem por atraso. A tarefa j , então, já poderia começar, terminando no instante 75, com 20 unidades de tempo de atraso, resultando em uma multa por atraso de 400 ($= 20 \times 20$), mostrado na figura 1.2. O custo total nesta última situação seria de 400 unidades, menor que aquele no qual a tarefa i começava desde o instante de liberação da máquina. Assim, não havendo restrições à ociosidade das máquinas, determinar a melhor data para iniciar o processamento de cada tarefa ou, equivalentemente, inserir tempos ociosos entre tarefas, poderá produzir soluções de menores custos.

Com relação à influência do tempo de preparação das tarefas na sequência de produção, chamado de tempo de *setup*, observa-se que a maioria dos trabalhos científicos considera que tais tempos são negligenciáveis ou, então, podem ser adicionados aos tempos de processamento das tarefas. Isto é, considera-se que os tempos de *setup* são independentes da sequência de tarefas. Contudo, muitos estudos, como

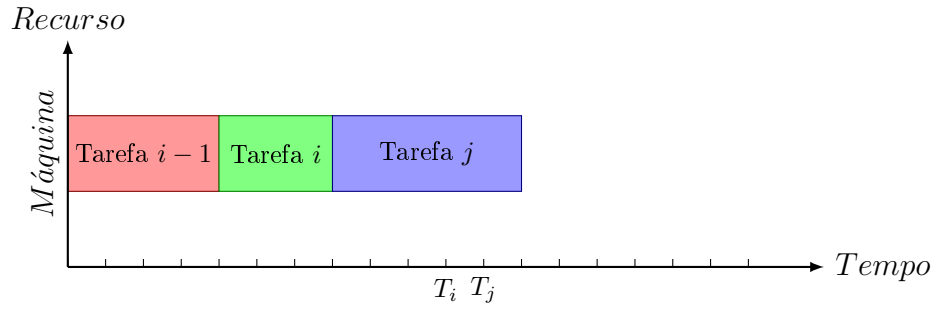


Figura 1.1: Exemplo de sequenciamento sem tempo ocioso entre tarefas

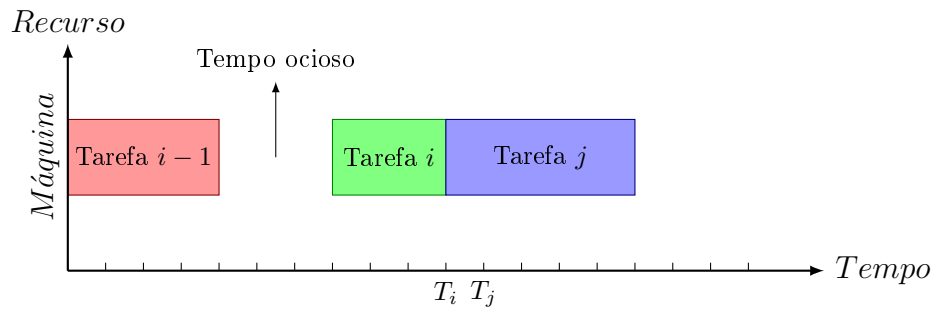


Figura 1.2: Exemplo de sequenciamento com tempo ocioso entre tarefas

os de Panwalkar *et al.* (1973), *apud* Gupta e Smith (2006), indicam que tempos de *setup* significativos aparecerem com frequência em muitas situações práticas sempre que há alteração da tarefa a ser processada na máquina. Segundo os autores, com base em informações prestadas por gerentes de produção, cerca de 70% dos entrevistados afirmaram que em pelo menos 25% das operações, o tempo de preparação era dependente da sequência de produção. De acordo com Kim e Bobrowski (1994), para modelar melhor casos práticos, esses tempos devem ser considerados explicitamente sempre que eles forem significativamente maiores que os tempos de processamento. Uma revisão de trabalhos sobre sequenciamento da produção com tempos de preparação pode ser encontrada em Allahverdi *et al.* (1999).

Ainda segundo Allahverdi *et al.* (1999), os problemas de sequenciamento envolvendo tempo de preparação podem ser classificados em dois tipos de problemas:

1. com tempo de preparação independente da sequência (*sequence-independent*);
2. com tempo de preparação dependente da sequência (*sequence-dependent*).

No primeiro tipo de problema o tempo de preparação depende somente da tarefa a ser processada, já no segundo, o tempo de preparação depende da tarefa a ser processada e da tarefa que a imediatamente precede. Neste caso, de acordo com Allahverdi *et al.* (1999), alguns autores resolveram este problema através do Problema do Caixeiro Viajante - PCV (*traveling salesman problem* - TSP), em que os tempos de preparação são tratados como distâncias entre as cidades.

Dentre os problemas de produção, o de sequenciamento em uma máquina com minimização das penalidades por antecipação e atraso da produção (*Single Machine Scheduling for Minimizing Earliness and Tardiness Penalties*), representado simplesmente por PSUMAA, é um dos mais simples.

Além da importância prática desse problema, com inúmeras aplicações industriais (Allahverdi *et al.*, 1999), trata-se de um problema difícil de ser resolvido na otimalidade em tempos computacionais aceitáveis, por pertencer à classe NP-difícil (Du e Leung, 1990; Liaw, 1999; Biskup e Feldmann, 2001; Wan e Yen, 2002). Essa conjunção de aplicabilidade e dificuldade de resolução na otimalidade tem motivado os pesquisadores a desenvolverem algoritmos eficientes para resolvê-lo.

Neste trabalho, trata-se o PSUMAA com janelas de entrega distintas e tempos de preparação da máquina dependentes da sequência da produção. Surpreendentemente, essa variante do PSUMAA ainda não tem merecido a devida atenção. O único trabalho encontrado na literatura, com essas características, é o de Gomes Jr. *et al.* (2007).

Para resolvê-lo, propõe-se um algoritmo de 3 fases, combinando os procedimentos GRASP (Feo e Resende, 1995), *Variable Neighborhood Descent* - VND (Mladenović e Hansen, 1997), Busca Tabu (Glover, 1986) e Reconexão por Caminhos (*Path Relinking*) (Glover, 1996). Na primeira fase é utilizado um procedimento baseado em GRASP para gerar uma solução inicial. Nesta fase, cada solução construída é refinada por um procedimento baseado em Descida em Vizinhança Variável (VND), que utiliza movimentos de realocação e troca de tarefas para explorar o espaço de soluções. Na segunda fase, a solução proveniente do GRASP é refinada por um procedimento Busca Tabu. Durante essa fase é formada uma lista de soluções

elite (grupo elite), contendo soluções de alta qualidade e diferenciadas entre si. Como pós-otimização, na terceira e última fase, é aplicada a Reconexão por Caminhos a pares de soluções do grupo elite.

1.2 Justificativa e Relevância

O problema de sequenciamento da produção está entre os mais estudados desta década. Segundo Allahverdi *et al.* (2008) em seu trabalho de revisão bibliográfica sobre o problema de sequenciamento com tempo de preparação, entre os anos de 1999 e 2005, foram publicados em periódicos mais de 300 trabalhos sobre o assunto, uma média superior a 40 trabalhos por ano. Este número é muito superior aos 190 trabalhos encontrados entre as décadas de 60 e 90.

Apesar do grande número de trabalhos publicados sobre o problema de sequenciamento, na revisão de Allahverdi *et al.* (2008) não foram encontrados trabalhos sobre o PSUMAA com janelas de entrega distintas e tempos de preparação da máquina dependentes da sequência da produção, foco deste trabalho.

De acordo ainda com Allahverdi *et al.* (2008), a metaheurística Busca Tabu é uma das mais utilizadas para resolver esta classe de problemas. Como esta técnica requer uma solução inicial de qualidade, nada mais natural que utilizar a metaheurística GRASP para gerá-la. Por sua vez, somente a fase de construção GRASP não guia a ótimos locais com relação às estruturas de vizinhanças adotadas. Assim, aplica-se também um método de busca local para melhorar as soluções construídas. A metaheurística VND foi usada para esse fim, pois requer poucos parâmetros (basicamente a definição das vizinhanças e a ordem de exploração destas), produz soluções de boa qualidade (Mladenović e Hansen, 1997) e é de fácil implementação. A aplicação de Reconexão por Caminhos deveu-se ao sucesso desta técnica na solução de vários outros problemas combinatórios (Glover e Kochenberger, 2003).

Este trabalho apresenta relevância por tratar um subproblema de sequenciamento ainda pouco estudado e contribuir com o desenvolvimento de uma ferramenta computacional para a otimização de processos produtivos nas indústrias têxtil, pe-

troquímica, siderúrgica e mineração, dentre outras.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho possui como objetivo geral desenvolver um algoritmo eficiente de otimização, baseado em técnicas metaheurísticas, para resolver o problema de sequenciamento em uma máquina com minimização das penalidades por antecipação e atraso da produção, com tempo de preparação dependente da sequência de produção.

1.3.2 Objetivos Específicos

Os objetivos específicos a serem atingidos são os seguintes:

1. Fazer uma revisão de literatura sobre as metodologias utilizadas para resolver o problema de sequenciamento em uma máquina;
2. Fazer uma revisão de literatura sobre as técnicas metaheurísticas GRASP, *Variable Neighborhood Descent* - VND, Busca Tabu e Reconexão por Caminhos (*Path Relinking*);
3. Desenvolver uma metodologia de otimização híbrida, baseada em metaheurísticas, para resolver eficientemente o problema;
4. Contribuir com a formação de recursos humanos especializados nessa área do conhecimento;
5. Possibilitar maior eficiência no planejamento do sequenciamento da produção.

1.4 Organização do Trabalho

O restante deste trabalho está estruturado como segue.

No capítulo 2, as características do problema estudado são apresentadas em detalhes, bem como uma possível aplicação na área de Mineração.

Uma revisão dos trabalhos correlatos e das técnicas heurísticas utilizadas são apresentadas no capítulo 3.

No capítulo 4 é descrito o algoritmo desenvolvido para resolução do PSUMAA.

No capítulo 5 são apresentados e discutidos os resultados computacionais.

Por fim, o capítulo 6 conclui o trabalho e aponta perspectivas futuras para melhoramento do algoritmo proposto.

Capítulo 2

O Problema de Sequenciamento

Neste capítulo é apresentado os principais tipos de problemas de sequenciamento da produção e, em detalhes, o problema estudado neste trabalho.

2.1 Tipos de Problemas

Na revisão de Allahverdi *et al.* (2008), os autores apresentam os principais tipos de problema de sequenciamento, são eles:

1. **Uma máquina (*single-machine*)**: Neste tipo de problema, foco deste trabalho, todas as tarefas são processadas em uma única máquina;
2. **Máquinas paralelas (*parallel machines*)**: Existem m máquinas em paralelo, que podem ser idênticas ou não. Cada tarefa pode ser processada em qualquer uma das máquinas;
3. ***Flow shop***: Em um ambiente *flow shop* com m máquinas há m estágios em série, com uma ou mais máquinas em cada estágio. Cada tarefa tem que ser processada em cada um dos m estágios na mesma ordem. O tempo de processamento para cada tarefa em diferentes estágios podem ser diferentes. Os problemas *flow shop* podem ser classificados como:
 - (a) ***regular flow shop***: Existe uma máquina em cada estágio;

- (b) ***no-wait flow shop***: Uma operação sucessora inicia imediatamente após sua antecessora terminar;
 - (c) ***flexible (hybrid) flow shop***: mais de uma máquina existe em pelo menos um estágio;
 - (d) ***assembly flow shop***: Em um ambiente *assembly flow shop* de dois estágios, cada tarefa passa por $m - 1$ operações específicas em uma máquina predeterminada no primeiro estágio, seguida de uma operação de montagem (*assembly operation*) na máquina do segundo estágio;
4. ***Job shop***: Consiste em m máquinas diferentes e cada tarefa possui uma dada ordem de execução, na qual algumas máquinas podem não ser usadas e outras podem ser repetidas;
 5. ***Open shop***: Este ambiente é semelhante ao anterior, porém cada tarefa deve ser processada uma única vez em cada uma das m máquinas, passando por elas em qualquer ordem.

2.2 O Problema de Sequenciamento na Mineração

A mineração realiza suas atividades em minas subterrâneas ou a céu aberto. Em uma mina a céu aberto, o planejamento operacional de lavra consiste no planejamento de curto prazo, onde o principal objetivo é a determinação do ritmo de lavra a ser implementado em cada frente, fornecendo à usina de beneficiamento uma alimentação adequada.

Vários trabalhos tratam esse problema em suas várias etapas, entre elas o problema de mistura de minérios, alocação de equipamentos de carga e transporte e o problema de despacho, de forma independente ou integrada.

Segundo Gershon (1982), *apud* Costa (2005), otimizar o problema de planejamento de lavra em partes independentes pode gerar conflitos que inviabilizam a implementação das soluções obtidas em cada parte.

Costa (2005) estudou três problemas do planejamento operacional de lavra,

a mistura de minérios, mistura de minérios com alocação dinâmica de caminhões e mistura de minérios com alocação estática de caminhões. Os problemas foram resolvidos através de duas metodologias, uma baseada em programação matemática e outra utilizando técnicas heurísticas.

Rodrigues (2006) fez uma análise comparativa das metodologias de despacho de caminhões em minas a céu aberto. Para resolver o problema foi utilizado programação linear, programação dinâmica e uma heurística baseada em Algoritmos Genéticos (AG).

Araújo (2008) tratou o planejamento operacional de lavra com alocação dinâmica de caminhões. O problema abordado consistia em determinar o número de viagens que cada caminhão devia fazer a cada frente de lavra, bem como decidir em quais frentes de lavra alocar as carregadeiras, com a finalidade de atender às metas de produção e qualidade requeridas para o minério a ser produzido, fazendo o melhor aproveitamento da frota de veículos disponível. Na resolução do problema foi empregado um procedimento heurístico baseado na metaheurística *Iterated Local Search*, e um modelo matemático foi formulado para validar o método heurístico proposto.

A presente pesquisa tem seu foco no sequenciamento da produção, que é uma etapa do planejamento operacional de lavra que surge após a alocação dos equipamentos de carga (carregadeiras) e transporte (caminhões) às frentes.

Costa (2005) descreve que o problema da mistura de minérios com alocação de caminhões objetiva determinar o ritmo de lavra de cada frente de acordo com sua capacidade de produção. A capacidade de produção de cada frente é determinada pelos equipamentos de carga nela alocada e pelos caminhões que realizam o transporte do material até o ponto de basculamento. Neste problema há operação de lavra de minério e estéril, a mina conta com uma frota limitada de equipamentos de carga, os quais devem ser alocados às frentes de lavra e operar em uma faixa de produtividade que torne viável sua utilização. O transporte do material retirado da frente de lavra é realizado por uma frota de caminhões com capacidade de carga diferentes.

Esses caminhões são alocados às frentes de lavra, tentando-se evitar a formação de filas, ou seja, o caminhão é alocado a um ponto de carga ou basculamento que proporcione o menor tempo de fila possível. Entretanto, um caminhão somente pode ser direcionado à uma frente que possua um equipamento de carga compatível.

Para exemplificá-lo, seja a Tabela 2.1, a qual apresenta a solução do problema de alocação dinâmica de caminhões de Araújo (2008). Nesta tabela, a primeira coluna indica as frentes de lavra; a segunda, as carregadeiras e o *status* da alocação (1 se ela está em operação e 0, caso está inativa) e as demais colunas, os caminhões. Cada célula desta tabela indica o número de viagens que cada caminhão faz à frente de lavra. Um número negativo indica incompatibilidade entre o caminhão e a carregadeira alocada à frente. Pode-se observar que a carregadeira 5 está alocada e em operação na frente de lavra F_1 . Na frente F_2 a carregadeira 3 está alocada, porém não está em operação. Já na F_3 não há nenhuma carregadeira alocada. Em relação aos caminhões, a tabela indica que o Cam_1 efetua 5 viagens à frente F_1 , não é compatível com a carregadeira da frente F_2 , não faz viagens a frente F_3 , faz 7 viagens a frente F_4 , e assim por diante.

Tabela 2.1: Representação de uma solução do problema de alocação de caminhões

$Frentes$	$Carga$	Cam_1	Cam_2	$\dots$	$Cam_{ V }$
F_1	(5, 1)	5	0	$\dots$	2
F_2	(3, 0)	-1	-1	$\dots$	0
F_3	(-1, 0)	0	0	$\dots$	-1
F_4	(6, 1)	7	-1	$\dots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$F_{ F }$	(2, 1)	4	0	$\dots$	1

Para mostrar como o problema de sequenciamento aparece no planejamento operacional de lavra, suponhamos a solução do problema de alocação de caminhões, dada na Tabela 2.2. Nesta tabela são descritos o número de viagens alocadas para cada caminhão, C_0 à C_{27} , às frentes de lavra, sendo as linhas M_1 a M_{12} , frentes de minério e E_1 à E_5 frentes de estéril.

Tabela 2.2: Solução de um problema de alocação de caminhões

[illegible]

A Tabela 2.3 mostra os tempos de operação dos caminhões em cada frente de lavra, onde as linhas representam as frentes de lavra, e as colunas o tempo de deslocamento vazio, o tempo de deslocamento cheio, o tempo de carga, o tempo de descarga e o tempo total, respectivamente.

Tabela 2.3: Tempo de operação dos caminhões

Frentes	TDVazio	TDCheio	Tcarga	Tdescar	Ttotal
M_1	2	3	3	1	9 min
M_2	3	4	3	1	11 min
M_3	3	4	3	1	11 min
M_4	3	4	3	1	11 min
M_5	4	6	3	1	14 min
M_6	2	3	3	1	9 min
M_7	2	4	3	1	10 min
M_8	3	4	3	1	11 min
M_9	2	3	3	1	9 min
M_{10}	4	5	3	1	13 min
M_{11}	2	4	3	1	10 min
M_{12}	3	4	3	1	11 min
E_1	3	5	3	1	12 min
E_2	2	4	3	1	10 min
E_3	2	3	3	1	9 min
E_4	3	4	3	1	11 min
E_5	4	6	3	1	14 min

As Tabelas 2.4 e 2.5 representam uma possibilidade de sequenciamento nas frentes M_4 e E_1 , respectivamente, para o problema apresentado na Tabela 2.2, em um intervalo de uma hora de produção.

Nas Tabelas 2.4 e 2.5, as colunas indicam os minutos e as linhas a operação de cada caminhão. O caracter c indica que o caminhão está sendo carregado naquele instante; o caracter d , que o caminhão está basculando (descarregando o material) e uma célula vazia indica que o caminhão está em viagem (cheio ou vazio) ou aguardando disponibilidade da carregadeira para carregamento. O caracter o indica a existência de tempo ocioso da carregadeira. Esse último caracter aparece somente na última linha, não vinculado a nenhum caminhão, apenas para mostrar a ociosidade da carregadeira. Neste exemplo, considera-se que os caminhões possuem a mesma capacidade e são compatíveis com as carregadeiras alocadas a cada frente.

Os tempos adotados foram aqueles apresentados na Tabela 2.3.

Nesta situação, pode-se considerar uma tarefa como o carregamento de um caminhão e uma máquina como sendo a carregadeira alocada à frente. Assim, tem-se que a carregadeira alocada à frente M_4 , dita máquina M_4 , executa 17 tarefas, com tempo total de processamento de 59 minutos. Uma solução para o seu sequenciamento pode ser dado pelo vetor $v_{M_4} = \{C_{02}, C_{04}, C_{07}, C_{02}, C_{04}, C_{07}, C_{09}, C_{10}, C_{07}, C_{09}, C_{10}, C_{07}, C_{09}, C_{10}, C_{07}, C_{09}, C_{10}\}$. Já a máquina E_1 executa 18 tarefas, com tempo total de processamento de 55 minutos, cujo vetor solução pode ser $v_{E_1} = \{C_{05}, C_{08}, C_{10}, C_{12}, C_{05}, C_{08}, C_{12}, C_{02}, C_{04}, C_{05}, C_{15}, C_{02}, C_{04}, C_{05}, C_{15}, C_{02}, C_{04}, C_{05}\}$.

É possível observar que ao tratar as carregadeiras alocadas às frentes de lavra como máquinas, caracteriza-se a existência de várias máquinas; portanto, o problema é classificado como sequenciamento em máquinas paralelas. A solução deste problema envolve, assim, o sequenciamento em cada máquina, foco deste estudo. Vale ressaltar que este trabalho trata o problema de sequenciamento de forma bem mais ampla e genérica, sendo possível ajustar seus parâmetros e/ou acrescentar outros critérios otimizantes para que se comporte como descrito acima.

Ao tratar o sequenciamento da produção, no intervalo de apenas uma hora de produção, é possível identificar características do PSUM, como a máquina, a tarefa, o tempo de processamento e a ociosidade entre as tarefas. Porém, ao se ampliar o horizonte de planejamento, pensando em termos semanais, por exemplo, é possível também identificar outras características como tempo de *setup*, que pode ser visto como o tempo necessário para uma carregadeira se deslocar de uma frente para outra. Ao se estabelecer a produção de materiais diferentes e um período de entrega distinto para cada um desses materiais, pode-se ter como objetivo entregar os produtos nos períodos indicados, minimizando-se os custos com o atraso e/ou com a antecipação da produção. O objetivo de minimizar os custos com a antecipação pode ser devido ao custo de estocagem do material até sua entrega, enquanto que o objetivo de minimizar os custos com o atraso visa à redução das multas contratuais por não entregar o produto na data requerida.

O problema tratado neste trabalho é apresentado a seguir na seção 2.3.

Tabela 2.4: Sequenciamento da Solução da Frente M4

[illegible]

Tabela 2.5: Sequenciamento da Solução da Frente E1

[illegible]

2.3 Problema Abordado

O problema de sequenciamento em uma máquina com penalidades por antecipação e atraso (PSUMAA) tratado neste trabalho tem as seguintes características:

- (i) Uma máquina deve processar um conjunto de n tarefas;
- (ii) A cada tarefa i está associado um tempo de processamento P_i e uma janela de entrega $[E_i, T_i]$, indicando uma data inicial E_i e uma data final T_i desejadas para o término de seu processamento. Se a tarefa i for finalizada antes de E_i há um custo α_i por unidade de tempo de antecipação. Caso a tarefa seja finalizada após T_i então há um custo β_i por unidade de tempo de atraso. As tarefas concluídas dentro da janela de entrega não proporcionam nenhum custo;
- (iii) A máquina pode executar no máximo uma tarefa por vez e uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção;
- (iv) Todas as tarefas estão disponíveis para processamento na data 0;
- (v) Entre duas tarefas i e j consecutivas é necessário um tempo S_{ij} de preparação da máquina, chamado tempo de setup. Assume-se que a máquina não necessita de tempo de preparação inicial, ou seja, o tempo de preparação da máquina para o processamento da primeira tarefa na sequência é igual a 0;
- (vi) É permitido tempo ocioso entre a execução de duas tarefas consecutivas.

A Figura 2.1 exemplifica o PSUMAA com janelas de entrega distintas e tempos de preparação da máquina dependentes da sequência da produção, foco deste trabalho.

Segundo a notação $a|b|c$ de Allahverdi *et al.* (2008), o primeiro campo, a , indica o tipo de máquina; o segundo campo, b , indica detalhes das características de processamento, como tempo de preparação e datas de entrega e, por fim, o terceiro campo, c , indica o objetivo a ser minimizado. Desta forma, podemos descrever o PSUMAA com a seguinte notação $1|ST_{sd}, E_i \leq d_i \leq T_i|\sum_{i=1}^n (\alpha_i e_i + \beta_i t_i)$, onde:

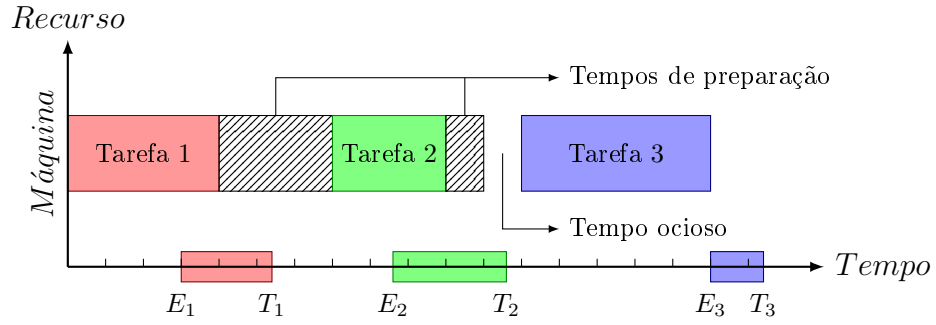


Figura 2.1: Problema Estudado

1. Tipo de máquina (Campo *a*):
 - (a) 1, indicando uma máquina (*single machine*);
2. Característica de processamento (Campo *b*):
 - (a) ST_{sd} , representando tempo de preparação dependente da sequência (*sequence-dependent setup time*);
 - (b) $E_i \leq d_i \leq T_i$, indicando janela de entrega distinta (*distinct due window*);
3. Função objetivo (Campo *c*):
 - (a) $\sum_{i=1}^n (\alpha_i e_i + \beta_i t_i)$, representando penalização por atraso e antecipação (*total weighted earliness and tardiness*).

Para exemplificar o problema, seja a Tabela 2.6, relativa a 4 tarefas, em que se apresenta o tempo de processamento (P_i), o custo por antecipação (α_i), o custo por atraso (β_i), os tempos de preparação da máquina (S_{ij}), a data inicial (E_i) e a data final (T_i) desejadas para entrega.

Tabela 2.6: Exemplificação dos dados de entrada

Tarefas	Dados do Problema					Tempo de Preparação			
	P_i	E_i	T_i	α_i	β_i	1	2	3	4
1	3	3	7	9	18	0	1	1	2
2	5	11	16	10	20	1	0	3	2
3	4	5	9	7	14	1	3	0	2
4	5	7	13	8	16	2	2	2	0

Capítulo 3

Revisão Bibliográfica

Neste capítulo é feita uma revisão dos trabalhos relacionados ao problema aqui estudado.

3.1 Introdução

Nos últimos anos um grande número de empresas tem adotado a produção de bens sob encomenda através da filosofia *just-in-time*. Com a adoção deste modelo, faz-se necessário um planejamento criterioso das atividades produtivas, visto que antecipações ou atrasos podem gerar custos extras, como por exemplo, custos de armazenagem e multas contratuais, respectivamente.

Na indústria mineral esta realidade não é diferente, os prazos de entrega se tornam cada vez mais rígidos. As indústrias trabalham, normalmente, em sua capacidade máxima e com as produções já vendidas. Desta forma, são necessários métodos eficientes para planejamento do sequenciamento da produção, buscando sempre a minimização dos custos de produção, visto que antecipações ou atrasos na produção acarretam para a empresa custos, seja de estocagem ou multas contratuais, respectivamente.

O problema de sequenciamento em uma máquina com minimização das penalidades por antecipação e atraso da produção (PSUMAA), modela um dos mais simples problemas de planejamento da produção no contexto da filosofia *just-in-time*. Contudo, trata-se de um problema difícil de ser resolvido na otimalidade em tempos

computacionais aceitáveis, visto pertencer à classe NP-difícil (Du e Leung, 1990; Wan e Yen, 2002).

Vários trabalhos tratam o PSUMAA com datas de entrega. Gordon *et al.* (2002) apresenta um estudo sobre datas comuns de entrega em problemas de sequenciamento em uma máquina e em máquinas paralelas. De acordo com o autor a data de entrega se tornou um fator importante a partir de meados da década de 80 devido à filosofia *just-in-time*; porém Jackson (1955), *apud* Gordon *et al.* (2002), foi o ponto de partida sobre seu estudo. Vários métodos de formulação da função objetivo foram apresentados, além de algumas propriedades para datas comuns de entrega.

Coleman (1992) resolveu o PSUMAA com data de entrega distinta e tempo de preparação dependente da sequência através de um modelo de programação inteira mista. Em seu trabalho, Coleman (1992) resolveu problemas-teste utilizando o otimizador LINDO, com 4 e 8 tarefas.

Lee e Choi (1995) aplicaram Algoritmos Genéticos (AG) ao PSUMAA com datas de entrega distintas. Para determinar a data ótima de conclusão de processamento de cada tarefa da sequência produzida pelo AG, desenvolveram um algoritmo específico, de complexidade polinomial, que explora as características do problema.

Sridharan e Zhou (1996) estudaram o PSUMAA dinâmico com datas de entrega distintas. De acordo com os autores, para problemas com tempos de entrega distintos a solução ótima pode conter tempo ocioso entre as tarefas. A maioria dos trabalhos que estudam este problema utilizam um algoritmo de duas fases para resolver o problema. Na primeira fase é gerada uma sequência de produção e, na segunda, é calculada a data ótima de execução, podendo-se inserir tempo ocioso entre as tarefas. Esses autores, no entanto, propuseram uma heurística de uma única fase baseado em teoria de decisão para construção do sequenciamento com tempo ocioso embutido.

James (1997) resolveu o PSMUAA com data comum de entrega sem tempo ocioso entre as tarefas utilizando a metaheurística Busca Tabu. O autor utilizou os

seguintes movimentos para explorar o espaço de busca: troca entre tarefas adjacentes, troca entre duas tarefas quaisquer e inserção de uma tarefa à frente de outro. Experimentos mostraram que a troca adjacente obteve os piores resultados e um movimento híbrido de troca e inserção obteve os melhores resultados.

Li (1997), Liaw (1999) e Biskup e Feldmann (2001) trataram o PSUMAA com datas comuns de entrega, sem permitir tempo ocioso de máquina. O primeiro autor decompõe o problema em dois subproblemas com uma estrutura mais simples, de forma que o limite inferior do problema é a soma dos limites inferiores desses dois subproblemas. O limite inferior de cada subproblema é determinado por relaxação lagrangiana. Um algoritmo *branch-and-bound* é apresentado e usado para resolver instâncias de até 50 tarefas, dobrando a dimensão de problemas que podiam ser resolvidos na otimalidade com algoritmos exatos até aquela data. O autor propôs, também, procedimentos heurísticos baseados em busca local para resolver problemas de dimensões mais elevadas. No segundo trabalho é apresentado um algoritmo *branch-and-bound* que faz uso de procedimentos para determinar limites inferiores e superiores fortes. Regras de dominância são usadas para tentar eliminar nós não promissores na árvore de busca. É analisado o desempenho do algoritmo para resolver problemas de até 50 tarefas. O terceiro trabalho apresenta duas heurísticas para resolver o problema, porém o foco do trabalho está em apresentar um algoritmo para geração de problemas-teste, com objetivo principal de gerar instâncias que pudessem ser utilizadas na comparação de performance de diferentes métodos de resolução do problema.

Chang (1999) tratou o PSUMAA com datas distintas de entrega por meio de um algoritmo *branch-and-bound*. É analisado o desempenho desse algoritmo para resolver problemas com até 45 tarefas. Também é desenvolvido um esquema para delimitar o cálculo de diferentes limites inferiores baseados no procedimento de eliminação de sobreposição de uma sequência *just-in-time*. Propriedades e teoremas sobre procedimento de eliminação de sobreposição são apresentados.

Gagné *et al.* (2001) resolveram o PSUM com tempo de preparação dependente da sequência de produção com penalização por atraso utilizando a metaheu-

rística Colônia de Formigas (*Ant Colony Optimization* - ACO). O método ACO proposto foi adaptado do método ACO utilizado para resolver o PCV.

Feldmann e Biskup (2003) resolveram o PSUMAA com datas comuns de entrega por meio de três métodos: um algoritmo evolutivo, *Simulated Annealing* (SA) e uma versão aperfeiçoada da heurística *Threshold Accepting*, sendo esta última uma variante de *Simulated Annealing*. O mesmo problema foi tratado por Hino *et al.* (2005), os quais apresentaram métodos baseados nas metaheurísticas Busca Tabu e Algoritmos Genéticos, bem como hibridizações destas.

Ventura e Radhakrishnan (2003) estudaram o PSUMAA com o tempo de preparação incluído no tempo de processamento. Os autores apresentaram algumas propriedades e desenvolveram uma heurística para encontrar uma solução inicial e então resolveram o problema por meio de relaxação lagrangeana com um algoritmo de subgradiente.

Rabadi *et al.* (2004) focaram no PSUMAA com datas comuns de entrega e tempo de preparação da máquina dependente da sequência de produção. Os autores apresentam um procedimento *branch-and-bound* que é capaz de resolver na otimalidade, em tempos aceitáveis, problemas-teste de até 25 tarefas, o que representava um avanço porque até aquela data algoritmos exatos para essa classe de problemas eram capazes de resolver somente problemas-teste de até 8 tarefas.

Tian *et al.* (2005) mostraram que, fazendo uso de algumas propriedades, o problema de sequenciamento em uma máquina com minimização do tempo total de atraso e datas de entrega distintas pode ser resolvido em tempo polinomial usando o algoritmo *shortest processing time* (SPT), para um determinado número de datas de entrega distintas.

Hassin e Shani (2005) apresentaram um estudo sobre o PSUMAA com datas de entrega distintas, com algumas variações, tempos de processamentos comuns e distintos, além de um novo conceito denominado não-execução, onde uma tarefa pode não ser executada, atribuindo portanto uma penalidade por não-execução. Para resolver o problema os autores adaptaram um algoritmo de tempo polinomial

da literatura, por eles denominado GTW.

Gupta e Smith (2006) propuseram dois métodos, um baseado em GRASP e outro na heurística *space-based search*, para a resolução do problema de sequenciamento em uma máquina considerando a existência de data de entrega para cada tarefa e tempo de preparação de máquina dependente da sequência de produção, tendo como objetivo a minimização do tempo total de atraso. O método GRASP proposto foi dividido em três fases: Construção, Refinamento e Reconexão por Caminhos. Segundo os autores, a contribuição está em uma nova função custo para a fase de construção, uma nova variação do método VND para a fase de refinamento e uma fase de Reconexão por Caminhos usando diferentes vizinhanças.

Os autores Lin *et al.* (2007a,b) também estudaram o PSUMAA com datas comuns de entrega. No primeiro trabalho foram utilizadas as metaheurísticas Algoritmos Genéticos (AG) e *Simulated Annealing* (SA), nas quais é utilizada uma busca local gulosa. No segundo, os autores desenvolveram uma heurística baseada em troca sequencial. Em ambos, os autores fizeram uso de três propriedades da literatura aplicadas ao problema com datas comuns de entrega para melhorar o desempenho das heurísticas.

Hallah (2007) trata do PSUMAA com datas de entrega distintas, não sendo permitida a existência de tempo ocioso entre as tarefas. O autor propõe um método híbrido que combina heurísticas de busca local (regras de despacho, método da descida e *Simulated Annealing*) e um algoritmo evolucionário.

O PSUM com penalidade por atraso e data de entrega distinta foi estudado por Bilge *et al.* (2007), os quais utilizaram a metaheurística Busca Tabu para resolvê-lo. Para gerar a solução inicial os autores utilizaram vários métodos, entre eles as heurísticas *earliest due date* (EDD) e *weighted shortest processing time* (WSPT). A BT desenvolvida usou uma estrutura de vizinhança híbrida com movimentos de troca e realocação, porém os autores utilizaram uma estratégia para criação de uma lista de candidatos, se valendo de algumas propriedades do problema, para não permitir movimentos que não levariam a boas soluções. Foi também utilizado um tempo tabu (*tabu tenure*) dinâmico para evitar a ciclagem.

Ying (2008) trata do PSUMAA com datas comuns de entrega das tarefas e com tempo de *setup* de cada tarefa incluído em seu tempo de processamento e independente da sequência de produção. O problema é resolvido pelo algoritmo *Recovering Beam Search* (RBS), que é uma versão aperfeiçoada do algoritmo *Beam Search* (BS). Este, por sua vez, consiste em um algoritmo *branch-and-bound* em que somente os w nós mais promissores de cada nível da árvore de busca são retidos para ramificação futura, enquanto os nós restantes são podados permanentemente. Para evitar decisões equivocadas com respeito à poda de nós que conduzam à solução ótima, o algoritmo RBS utiliza uma fase de recobrimento que busca por soluções parciais melhores que dominem aquelas anteriormente selecionadas.

Alguns trabalhos, como Chen (1997) e Uzsoy e Velásquez (2008), tratam o tempo de preparação dependente da sequência por famílias, onde cada família consiste em um conjunto de tarefas que não possuem tempo de preparação entre si. O tempo de preparação somente existe entre as execuções das famílias de tarefas. Uzsoy e Velásquez (2008) utilizaram várias heurísticas para resolver o problema, entre elas uma baseada unicamente na heurística EDD e outra, baseada em EDD com uma busca local.

Wan e Yen (2002) apresentaram um método baseado na metaheurística Busca Tabu (BT) para resolver o PSUMAA com janelas de entrega distintas. Para cada sequência de tarefas gerada pela Busca Tabu é acionado um procedimento de complexidade polinomial para determinar a data de conclusão ótima do processamento de cada tarefa da sequência. Este último procedimento é adaptado daquele proposto em Lee e Choi (1995), em que se consideram janelas de entrega em lugar de datas de entrega.

Gomes Jr. *et al.* (2007) desenvolveram um modelo de programação linear inteira mista para o PSUMAA com janelas de entrega e tempo de preparação dependente da sequência de produção. Esta aplicação foi inspirada na dissertação de Bustamante (2006). Neste último trabalho, considera-se uma aplicação na indústria siderúrgica, em que uma máquina é considerada como sendo uma sequência de laminadores que representam o gargalo do sistema, e cada tarefa representa um conjunto

de operações realizadas nesses laminadores que resulte em um dado produto (candeeira, perfil U, vergalhão etc.). Para cada produto fabricado têm-se um conjunto de operações de mesma natureza realizadas nos mesmos laminadores, mas que requerem tempos de processamento diferentes, bem como ajustes na configuração dos equipamentos da laminação. Como cada ajuste nos equipamentos dura um tempo para ser executado, a composição destes tempos gera o tempo para troca entre cada produto.

O modelo de Gomes Jr. *et al.* (2007) foi utilizado para resolver na otimalidade problemas de até 12 tarefas. Esta modelagem serviu para comparar os resultados obtidos por um algoritmo heurístico baseado em GRASP, ILS e VND, também proposto pelos autores. Para cada sequência gerada pela heurística, é acionado um algoritmo para determinar a data ótima de conclusão do processamento de cada tarefa. Tal algoritmo foi adaptado de Wan e Yen (2002), e inclui no tempo de processamento de uma tarefa o tempo de preparação da máquina, já que quando ele é acionado, já se conhece a sequência de produção. O algoritmo desenvolvido pelos autores foi capaz de encontrar todas as soluções ótimas conhecidas.

Uma revisão mais abrangente de trabalhos sobre sequenciamento com tempos de preparação dependentes da sequência pode ser encontrada em Allahverdi *et al.* (1999) e Allahverdi *et al.* (2008).

Capítulo 4

Metodologia

Neste capítulo são apresentadas duas metodologias de solução do problema tratado. Na seção 4.1, é apresentado um modelo de programação matemática para resolver a otimalidade problemas-teste do PSUMAA de pequenas dimensões. Na seção 4.3, é apresentado um algoritmo heurístico para encontrar soluções sub-ótimas do PSUMAA.

A utilização de procedimentos heurísticos para resolver o PSUMAA se deve à complexidade combinatória do problema tratado. O algoritmo proposto baseia-se em metaheurísticas, as quais, ao contrário dos procedimentos heurísticos clássicos, têm mecanismos que possibilitam escapar das armadilhas dos ótimos locais. Mais especificamente, propõe-se um algoritmo de três fases, denominado GTSPR, combinando os procedimentos metaheurísticos GRASP (Feo e Resende, 1995), VND (Mladenović e Hansen, 1997), Busca Tabu (Glover e Laguna, 1997) e a técnica Reconexão por Caminhos (Glover, 1996).

Na primeira fase, é construída uma solução inicial utilizando-se GRASP e VND. Na segunda fase essa solução é refinada por um procedimento baseado em Busca Tabu. Na terceira e última fase, como pós-otimização, é aplicado Reconexão por Caminhos a pares de soluções elite produzidas durante a Busca Tabu.

A fase GRASP-VND do algoritmo envolve dois estágios, construção e refinamento, os quais são aplicados consecutivamente por várias iterações. No estágio de construção, uma solução é construída passo a passo, elemento por elemento. Para

decidir qual elemento deve ser inserido na solução em construção é feita uma lista restrita de candidatos (LRC) com os melhores elementos candidatos. Esses candidatos são avaliados segundo uma certa função que leva em consideração os elementos já inseridos e os benefícios com a inserção do elemento candidato. A seguir, um dos elementos da LRC é escolhido de acordo com uma certa probabilidade, normalmente igual para todos da LRC. Terminado o estágio de construção, a solução construída é refinada pelo procedimento VND.

A vantagem do procedimento GRASP é que ele combina uma característica positiva dos métodos construtivos puramente gulosos (a boa qualidade da solução construída, facilitando o refinamento), com uma característica positiva dos procedimentos totalmente aleatórios (a diversificação do espaço de busca). Como construir uma solução para o PSUMAA é um procedimento relativamente rápido, demandando pouco esforço computacional, e refinar a solução construída tem um alto custo computacional, propõe-se fazer o refinamento mais “leve” após cada construção, seguido de um refinamento mais “pesado” na melhor das soluções construídas e refinadas, na fase GRASP-VND do algoritmo.

Na segunda fase, a solução proveniente da fase GRASP-VND é refinada pelo procedimento metaheurístico Busca Tabu (Glover, 1986). A Busca Tabu é um método de otimização, baseado em busca local, que admite soluções de piora para escapar das armadilhas dos ótimos locais e utiliza uma estrutura de memória para não retornar às soluções já geradas. Em sua forma original, a cada iteração, da Busca Tabu, procura-se um ótimo local selecionando-se o melhor vizinho s' da vizinhança $N(s)$ da solução corrente s . Independentemente de $f(s')$ ser melhor ou pior que $f(s)$, s' será sempre a nova solução corrente. Entretanto, este mecanismo, por si só, não é suficiente para escapar de ótimos locais, uma vez que pode haver retorno a uma solução previamente gerada. Para evitar isso, o algoritmo usa o conceito de lista tabu. Esta lista define todos os movimentos executados, baseados em um certo atributo, como sendo tabu por um determinado número de iterações, conhecido como tempo tabu (*tabu tenure*). Tais movimentos são proibidos a menos que a solução satisfaça a um certo critério de aspiração, em geral que essa solução seja melhor que

a melhor solução encontrada até então. Os atributos são escolhidos para prevenir o retorno a soluções visitadas recentemente e são escolhidos por características fáceis de detectar.

Por fim, na terceira e última fase do algoritmo proposto, é utilizado o procedimento de Reconexão por Caminhos (RC) como estratégia de pós-refinamento. Proposto por Glover (1996), a RC é uma estratégia de intensificação normalmente utilizada como pós-otimização de uma solução, ou refinamento de ótimos locais obtidos durante a busca. Dado um par de soluções, o objetivo da técnica é partir de uma delas, dita solução base, e chegar à outra, dita solução guia, por meio da adição na solução base de atributos da solução guia.

O algoritmo heurístico proposto, após gerar uma sequência de produção, utiliza um procedimento para programá-la, isto é, determinar as datas ótimas de início e término de cada tarefa na sequência gerada. Esse procedimento está descrito na seção 4.2. A Figura 4.1 descreve o modelo de obtenção da solução para o PSUMAA abordado.

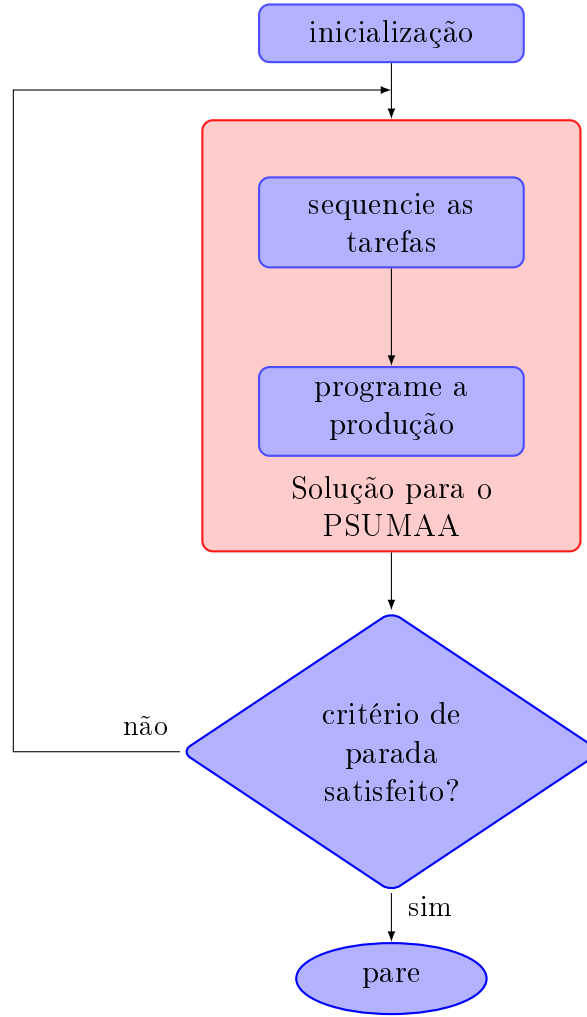


Figura 4.1: Metodologia heurística de solução para o PSUMAA

4.1 Modelo Matemático

O modelo matemático apresentado a seguir, relativo ao PSUMAA abordado, foi desenvolvido por Gomes Jr. *et al.* (2007), e é aqui reproduzido.

Nesse modelo são utilizadas duas tarefas fictícias, 0 (zero) e $n + 1$, de tal forma que a tarefa 0 antecede imediatamente a primeira operação e a tarefa $n + 1$ sucede imediatamente a última na sequência de produção. Considera-se M uma constante de valor suficientemente grande, P_0 e P_{n+1} iguais a zero e $S_{0i} = 0$ e $S_{i,n+1} = 0, \forall i = 1, \dots, n$.

As variáveis de decisão do modelo são:

- s_i , representando a data de início do processamento da tarefa i ($s_i \geq 0$);
- e_i , indicando o tempo de antecipação da tarefa i ;
- t_i , indicando o tempo de atraso da tarefa i e
- y_{ij} , uma variável binária definida da seguinte forma:

$$y_{ij} = \begin{cases} 1, & \text{se a tarefa } j \text{ é sequenciada imediatamente após a tarefa } i; \\ 0, & \text{caso contrário.} \end{cases}$$

O modelo de programação linear inteira mista de Gomes Jr. *et al.* (2007) é definido pelas expressões (4.1) a (4.10), a seguir:

minimizar:

$$\sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (4.1)$$

sujeito a:

$$s_j - s_i - y_{ij}(M + S_{ij}) \geq P_i - M \quad \forall i = 0, 1, \dots, n \quad (4.2)$$

$$\forall j = 1, 2, \dots, n+1 \text{ e } i \neq j$$

$$\sum_{i=0, i \neq j}^n y_{ij} = 1 \quad \forall j = 1, 2, \dots, n+1 \quad (4.3)$$

$$\sum_{j=1, j \neq i}^{n+1} y_{ij} = 1 \quad \forall i = 0, 1, \dots, n \quad (4.4)$$

$$s_i + P_i + e_i \geq E_i \quad \forall i = 1, 2, \dots, n \quad (4.5)$$

$$s_i + P_i - t_i \leq T_i \quad \forall i = 1, 2, \dots, n \quad (4.6)$$

$$s_i \geq 0 \quad \forall i = 0, 1, \dots, n+1 \quad (4.7)$$

$$e_i \geq 0 \quad \forall i = 1, 2, \dots, n \quad (4.8)$$

$$t_i \geq 0 \quad \forall i = 1, 2, \dots, n \quad (4.9)$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j = 0, 1, \dots, n+1 \quad (4.10)$$

A função objetivo, representada pela equação (4.1), tem como critério de otimização a minimização do custo total de antecipação e atraso. As restrições (4.2) definem a sequência de operações na máquina utilizada, garantindo a existência de um tempo suficiente para completar uma tarefa i antes de se começar outra tarefa j . Quando $y_{ij} = 1$, as restrições (4.2) se reduzem a $s_j \geq s_i + S_{ij} + P_i$, isto é, a tarefa j só pode começar após iniciada a tarefa predecessora i , processado-a e preparada a máquina para executar a tarefa j . Quando $y_{ij} = 0$, as restrições (4.2) se reduzem a $s_j \geq s_i + P_i - M$, e portanto, são redundantes, pois M é uma constante de valor arbitrariamente grande. As restrições (4.4) e (4.3) garantem que cada tarefa tenha somente uma tarefa imediatamente antecessora e uma tarefa imediatamente sucessora, respectivamente. As restrições (4.5) e (4.6) definem a quantidade de atraso e de antecipação de cada tarefa i . As restrições (4.7) a (4.10) definem o tipo das variáveis do problema.

4.2 Procedimento de Determinação das Datas Ótimas de Início de Processamento das Tarefas

Dada uma sequência de tarefas, o procedimento de determinação das datas ótimas de início de processamento (PDDOIP) determina a melhor data para se iniciar o processamento de cada tarefa, de acordo com a janela de entrega da mesma.

O PDDOIP implementado é o utilizado em Gomes Jr. *et al.* (2007), o qual foi baseado em Wan e Yen (2002) e Lee e Choi (1995), incorporando-se o tempo de preparação S_{ij} ao tempo de processamento da tarefa j . Tal consideração pode ser feita uma vez que, para sua aplicação, a sequência de produção é conhecida.

Sejam V a sequência de tarefas dada e C_k a data de conclusão do processamento da tarefa k . Suponha que um subconjunto de tarefas $B_j = \{J_0, J_1, \dots, J_m\} \subseteq V$ forme um bloco no sequenciamento, ou seja, as tarefas em B_j são sequenciadas consecutivamente sem tempo ocioso entre elas. Além disso, assuma que há l blocos em V e sejam $prim(j)$ e $ult(j)$ a primeira e a última tarefa no bloco B_j , respectivamente.

O processamento da primeira tarefa (J_1) da sequência V é programado para ser finalizado na data E_{J_1} se $P_{J_1} \leq E_{J_1}$ ou iniciado na data 0 (finalizando na data P_{J_1}) se $P_{J_1} > E_{J_1}$.

As demais tarefas são programadas da seguinte forma:

- Se $C_{J_k} + S_{(J_k)(J_{k+1})} + P_{J_{k+1}} < E_{J_{k+1}}$, então a tarefa J_{k+1} tem seu processamento programado para ser finalizado na data $E_{J_{k+1}}$, isto é, a tarefa J_{k+1} tem seu processamento programado para ser finalizado no início de sua janela de entrega, desta forma, iniciando um novo bloco.
- Por outro lado, se $C_{J_k} + S_{(J_k)(J_{k+1})} + P_{J_{k+1}} \geq E_{J_{k+1}}$, então a tarefa J_{k+1} tem seu processamento programado para ser finalizado na data $C_{J_k} + S_{(J_k)(J_{k+1})} + P_{J_{k+1}}$ e é incluído como último elemento do bloco corrente.

Após a determinação da data de conclusão do processamento de cada tarefa, é necessário verificar o posicionamento dos blocos. O custo mínimo do bloco B_j ocorre nos pontos extremos de sua função custo, isto é, no início ou no final da janela de entrega de uma das tarefas no bloco. Por causa da natureza linear convexa por partes da função custo, o custo mínimo pode ser facilmente obtido pela comparação dos somatórios das penalidades das tarefas no bloco, no início e no final de cada janela de entrega no bloco (penalidades por antecipação são consideradas negativas enquanto penalidades por atraso são consideradas positivas). Estes somatórios fornecem as inclinações das retas (m) pertencentes à função custo. O custo mínimo do bloco B_j se dá no ponto extremo onde a inclinação m torna-se maior ou igual a zero. Quando o ponto mínimo do bloco B_j é encontrado, todo o bloco é movido em direção ao ponto mínimo até que um dos três casos seguintes aconteça:

1. $s_{prim(j)} = 0$.
2. O ponto mínimo é alcançado.
3. $s_{prim(j)}$ torna-se igual a $C_{ult(j-1)} + S_{(ult(j-1))(prim(j))}$.

No caso (3), o bloco B_j é unido ao bloco B_{j-1} , resultando em um novo bloco B_{j-1} . Então, o ponto mínimo do novo B_{j-1} deve ser obtido. O procedimento anterior deve ser realizado até que o caso (1) e (2) ocorram para cada bloco.

Proposição 1 (Wan e Yen, 2002) - *O custo total de uma dada sequência de tarefas alcança seu valor ótimo se cada bloco B_j na sequência alcançar seu ponto mínimo, com exceção de $s_{prim(1)}$, que pode ser igual a zero para o primeiro bloco.*

Para exemplificar o método, serão utilizados os dados da Tabela 2.6, apresentada na página 18 da seção 2.3, relativa a 4 tarefas.

Dada a sequência $V = \{3, 4, 1, 2\}$, a primeira tarefa da sequência, tarefa 3, denotada por J_1 , tem seu processamento programado para ser finalizado na data $E_3 = 5$, que representa o início da sua janela de entrega. Para calcular as inclinações m da reta que indica o ponto de mínimo do bloco para o primeiro elemento, a penalidade por antecipação ($\alpha_3 = 7$) assume valor negativo e a por atraso ($\beta_3 = 14$) assume valor positivo. A Figura 4.2 descreve o primeiro passo da programação de tarefas.

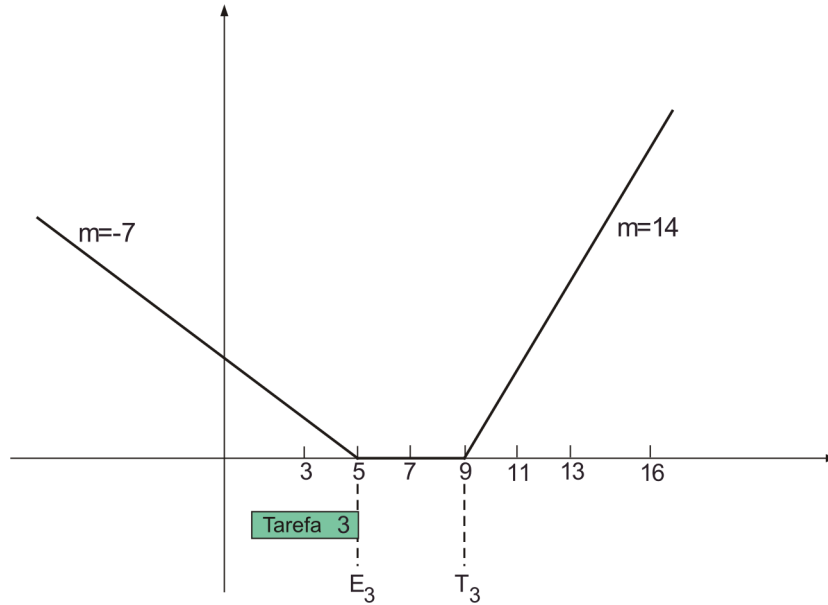


Figura 4.2: Programação das tarefas de um bloco na sequência: 1º Passo

A Figura 4.3 exemplifica a programação da segunda tarefa da sequência, tarefa $J_2 = 4$, que tem sua conclusão programada para ser finalizada no instante

$C_4 = 12$, pois $C_3 + S_{34} + P_4 > E_4$. A inclinação m da reta, para instantes inferiores a -4 , é dado por $m = -7 - 8 = -15$, momento que todas as tarefas se encontram adiantadas. Entre os instantes -4 e 1 , somente a tarefa 3 está adiantada, portanto $m = -7$. A partir do instante 2 as tarefas passam a ficar atrasadas, acarretando em uma inclinação positiva. Até o instante 3 somente a tarefa 4 está atrasada, sendo $m = 16$. Após este instante as tarefas 3 e 4 tornam-se atrasadas, portanto $m = 14 + 16 = 30$ a partir do instante 3.

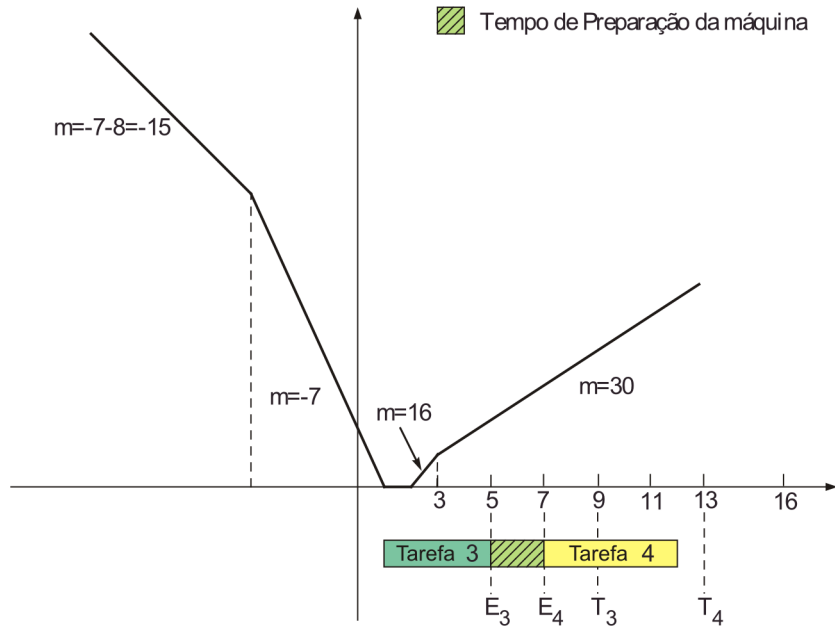


Figura 4.3: Programação das tarefas de um bloco na sequência: 2º Passo

Na Figura 4.4 são apresentadas as tarefas 3 e 4 já programadas, e mostra-se como determinar a data de conclusão do processamento da tarefa 1, os valores das inclinações m e o ponto mínimo do bloco. Até este momento, a tarefa $J_1 = 3$ se inicia no instante 1 e a tarefa $J_2 = 4$ no instante 7, ambos terminando dentro de sua janela de tempo. Como $C_{J_2} + S_{(J_2)(J_3)} + P_{J_3} = 17 \geq E_{J_3} = 3$, então $C_{J_3} = C_1 = 17$, isto é, a tarefa $J_3 = 1$ deve ser concluída no instante 17 e iniciada no instante dado por $s_{J_3} = s_1 = 17 - 3 = 14$. Neste caso, a tarefa 1 é inserida como último elemento do bloco corrente, conforme pode ser observado nesta figura. Conhecida a data de conclusão da tarefa 1, deseja-se agora verificar a posição do bloco, para saber se ele está em seu ponto mínimo. Para isto, primeiramente, são determinadas as possíveis datas de início do processamento do bloco. Essas datas são obtidas posicionando

cada tarefa do bloco no início e no final de sua janela de entrega, pois como foi visto, o custo mínimo do bloco se dá no início ou no final da janela de entrega de uma das tarefas no bloco.

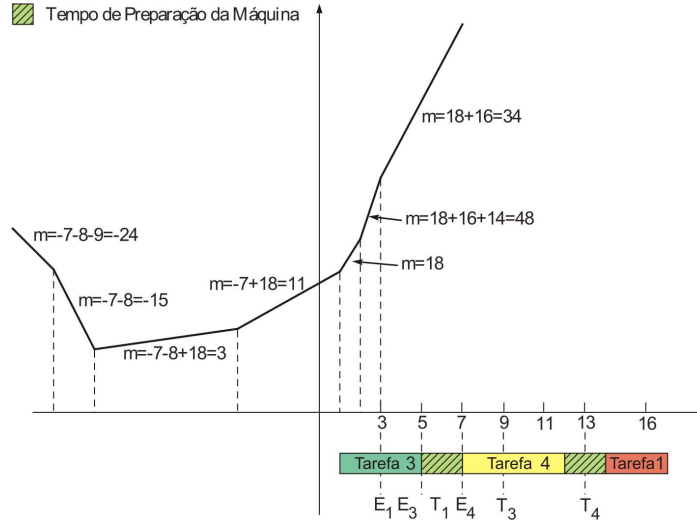


Figura 4.4: Programação das tarefas de um bloco na sequência: 3º Passo

No exemplo mostrado, as possíveis datas de início do processamento do bloco são: $-13, -11, -4, 1, 2$ e 5 . Determinadas as possíveis datas de início de processamento, calculam-se então as inclinações m de acordo com essas datas. No exemplo apresentado, se o processamento do bloco iniciar antes da data -13 , ou seja, com todas as tarefas sendo antecipadas, então a inclinação m é dada pelo somatório das penalidades por antecipação das tarefas 3, 4 e 1, portanto $m = -7 - 8 - 9 = -24$. Agora, se o processamento do bloco iniciar entre a data -13 e -11 , a tarefa 1 deixa de estar antecipada, e a inclinação m passa a ser $m = -7 - 8 = -15$ (somatório das penalidades por antecipação das tarefas 3 e 4). Entre a data -11 e -4 a tarefa 1 passa a estar atrasada, então a inclinação m é dada pelo somatório das penalidades por antecipação das tarefas 3 e 4 e da penalidade por atraso da tarefa 1, ou seja, $m = -7 - 8 + 18 = 3$. Este processo se repete até a última data. Graficamente, este procedimento está representado na Figura 4.4. O ponto mínimo, o qual se deseja encontrar, se dá no ponto aonde a inclinação m se torna maior ou igual a zero. No exemplo, esse ponto corresponde à data -11 . Determinado o ponto de mínimo, então todo o bloco deve ser movido para a esquerda até que um dos três casos citados anteriormente ocorra. No exemplo considerado, o bloco é empurrado

até o caso (1) ocorrer, ou seja, $s_{prim(1)} = 0$ (Figura 4.5).

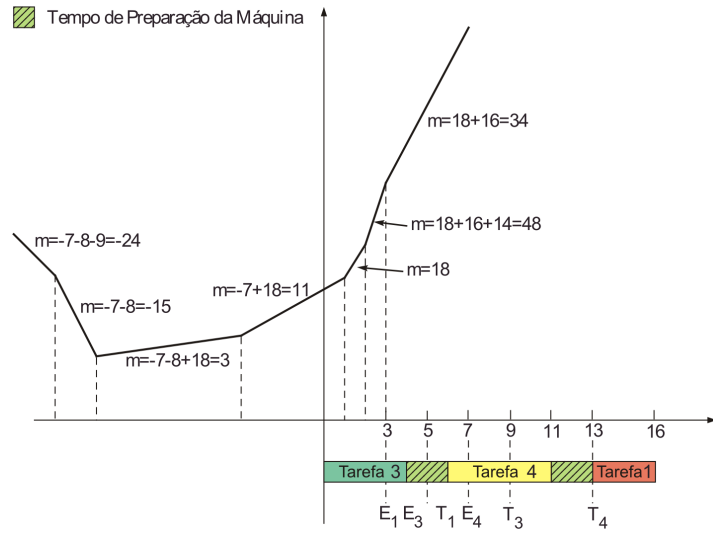


Figura 4.5: Programação das tarefas de um bloco na sequência: 4º Passo

O PDDOIP para uma sequência de tarefas dadas é descrito na Figura 4.6, onde o procedimento MudaBloco (Linha 26 da Figura 4.6) movimenta o bloco B_j até seu ponto de mínimo e, caso seja necessário, combina-o com o bloco anterior.

Procedimento DetermineDatasOtimasInicio($n, v, f(v)$)

```

1 início
2    $B \leftarrow 1$                                      // Inicialização
3    $prim(B) \leftarrow 1$ 
4    $ult(B) \leftarrow 1$ 
5    $s_1 \leftarrow \max\{0, E_1 - P_1\}$ 
6    $C_1 \leftarrow \max\{E_1, P_1\}$                      // Demais Tarefas
7   para  $i = 2$  até  $n$  faça
8     se  $(C_{i-1} + P_i + S_{(i-1)(i)} < E_i)$  então
9        $B \leftarrow B + 1$ 
10       $prim(B) \leftarrow 1$ 
11       $ult(B) \leftarrow i$ 
12       $s_i \leftarrow E_i - P_i + S_{(i-1)(i)}$ 
13       $C_i \leftarrow E_i$ 
14    senão
15      se  $(C_{i-1} + P_i + S_{(i-1)(i)} = E_i)$  então
16         $ult(B) \leftarrow i$ 
17         $s_i \leftarrow C_{i-1} + S_{(i-1)(i)}$ 
18         $C_i \leftarrow E_i$ 
19      senão
20         $ult(B) \leftarrow i$ 
21         $s_i \leftarrow C_{i-1} + S_{(i-1)(i)}$ 
22         $C_i \leftarrow s_i + P_i$ 
23    fim
24  fim
25  repita
26     $MudaBloco(B)$ 
27  até  $((\text{até todos os blocos estiverem no ponto mínimo}) \text{ ou } (s_{prim(1)} = 0))$ 
28 fim
29 retorna  $f(v)$ 
30 fim

```

Figura 4.6: Procedimento PDDOIP

4.3 Modelo Heurístico

4.3.1 Representação de uma Solução

Uma solução do PSUMAA é representada por um vetor v de n elementos, onde cada posição $i = 1, 2, \dots, n$ indica a ordem de produção da i -ésima tarefa da sequência. Assim, na sequência $v = \{5, 3, 2, 1, 4, 6\}$, a tarefa 5 é a primeira a ser realizada e a tarefa 6, a última.

4.3.2 Vizinhaça de uma Solução

Para explorar o espaço de soluções são utilizados três tipos de movimentos: troca da ordem de processamento de duas tarefas da sequência de produção, realocação de uma tarefa para outra posição na sequência de produção e realocação de um bloco de k tarefas, $2 \leq k \leq n - 2$. Esses movimentos definem, respectivamente, as estruturas de vizinhaça N^T , N^R e N^{Or} .

Dado um conjunto com n tarefas, há $n(n - 1)/2$ vizinhos possíveis na vizinhaça N^T . Por exemplo, a solução $v' = \{5, 4, 2, 1, 3, 6\}$ é um vizinho da solução v na vizinhaça N^T , pois é obtido pela troca da tarefa 3, que está na segunda posição de v , com a tarefa 4, que está na quinta posição de v , como pode ser visto na Figura 4.7.

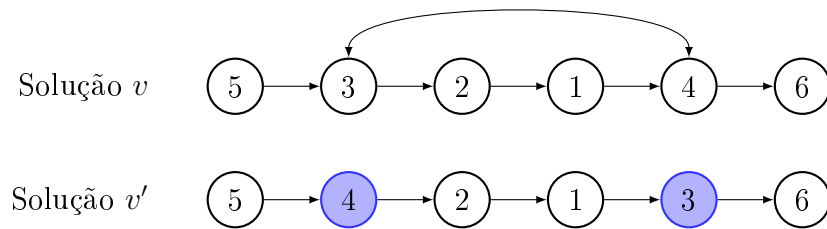


Figura 4.7: Movimento de troca - N^T

Na segunda estrutura de vizinhaça, N^R , há $(n - 1)^2$ vizinhos possíveis. Por exemplo, a solução $v' = \{5, 2, 1, 4, 3, 6\}$ é vizinha de v na vizinhaça N^R , pois é obtida pela realocação da tarefa 3 para depois da tarefa 4 na sequência de produção v , conforme exemplificado na Figura 4.8.

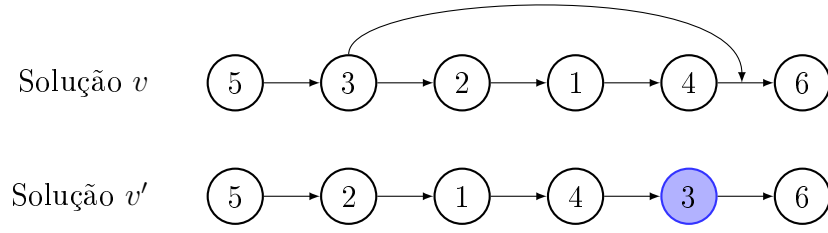


Figura 4.8: Movimento de realocação - N^R

Já na terceira estrutura de vizinhança, N^{Or} , há $(n - k - 1)^2$ vizinhos possíveis. Por exemplo, a solução $v' = \{5, 2, 1, 4, 3, 6\}$ é vizinha de v na vizinhança N^{Or} , pois é obtida pela realocação do bloco de tarefas 5 e 3, com $k = 2$, para depois da tarefa 1 na sequência de produção v , mostrado na Figura 4.9. Nas vizinhanças N^R e N^{Or} são permitidos movimentos para posições sucessoras ou antecessoras àquela em que a tarefa se encontra na sequência de produção.

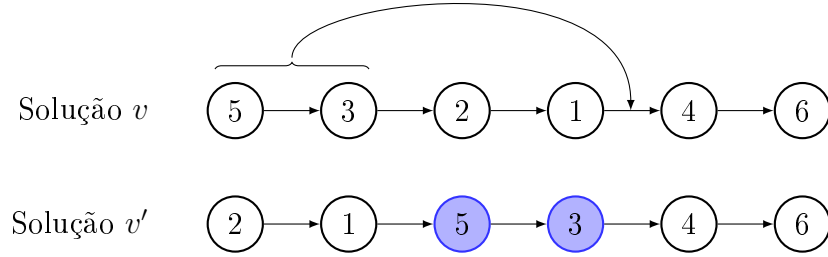


Figura 4.9: Movimento de realocação de um bloco - N^{Or}

4.3.3 Função de Avaliação

Uma solução v é avaliada pela função f a seguir, a qual deve ser minimizada:

$$f(v) = \sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (4.11)$$

em que e_i e t_i ($e_i, t_i \geq 0$) indicam, respectivamente, o tempo de antecipação e atraso da tarefa i e α_i e β_i são as penalidades respectivas.

Para determinar as datas ótimas de início de processamento das tarefas da

sequência dada e, conseqüentemente, determinar os valores e_i e t_i acima, utiliza-se o procedimento PDDOIP de Gomes Jr. *et al.* (2007).

Como os movimentos adotados não geram soluções inviáveis, pode-se observar que a função de avaliação f é a própria função objetivo do PLIM.

4.3.4 Geração da solução inicial

A solução inicial é gerada por um procedimento que combina GRASP (Feo e Resende, 1995) com Descida em Vizinhança Variável (*Variable Neighborhood Descent* - VND) (Mladenović e Hansen, 1997). A Figura 4.10 esquematiza esse procedimento.

Procedimento GRASP-VND($v, \gamma, GRASP_{max}, MRD_{max}$)	
1	início
2	$f^* \leftarrow \infty$
3	para $i=1$ até $GRASP_{max}$ faça
4	Escolha $\gamma \in \Gamma$
5	$v \leftarrow \text{Constroi\_Solucao\_GRASP}(\gamma)$
6	$v' \leftarrow \text{VND}_1(v, MRD_{max})$
7	se $(f(v') < f^*)$ então
8	$f^* \leftarrow f(v')$
9	$v^* \leftarrow v'$
10	fim
11	fim
12	$v \leftarrow \text{VND}_2(v^*, MRD_{max})$
13	retorna v
14	fim

Figura 4.10: Procedimento GRASP-VND aplicado ao PSUMAA

Na fase de construção (Linha 5 da Figura 4.10), uma solução é formada gradativamente, sendo que a cada passo é adicionada uma única tarefa à sequência. Para se escolher a tarefa a ser adicionada é feita uma lista de candidatos (LC) com todas as tarefas ainda não sequenciadas. Essas são ordenadas pela data de início da janela de entrega de cada tarefa, sendo a com a data mais cedo (E_{min}) a primeira da lista e a de data mais tarde (E_{max}) a última, tal como na heurística EDD (*Earliest Due Date*) (Baker e Scudder, 1990).

A partir de LC é formada uma lista restrita de candidatos (LRC), com

as tarefas melhor classificadas segundo o critério da data de início da janela de entrega das tarefas. O tamanho dessa lista restrita de candidatos é definido por um parâmetro $\gamma \in [0, 1]$, escolhido em um conjunto Γ , tal como em Gomes Jr. *et al.* (2007). Fazem parte da LRC todas as tarefas i cujas datas E_i sejam menores ou iguais à $E_{min} + \gamma(E_{max} - E_{min})$. A seguir, é escolhida aleatoriamente uma tarefa dessa lista, sendo a mesma adicionada à solução parcial. A fase de construção é encerrada quando todas as tarefas forem alocadas. Este procedimento é apresentado na Figura 4.11.

Procedimento Constroi_Solucao_GRASP(v, γ)	
1	início
2	$v \leftarrow \emptyset$
3	Inicialize o conjunto LC de tarefas candidatas
4	enquanto ($LC \neq \emptyset$) faça
5	$E_{min} = \min_{i \in LC} \{E_i\}$
6	$E_{max} = \max_{i \in LC} \{E_i\}$
7	$LRC = \{i \in LC \mid E_i \leq E_{min} + \gamma(E_{max} - E_{min})\}$
8	Selecione, aleatoriamente, uma tarefa $i \in LRC$
9	$v \leftarrow v \cup \{i\}$
10	Atualize o conjunto LC de tarefas candidatas
11	fim
12	retorna v
13	fim

Figura 4.11: Procedimento Constroi_Solucao_GRASP aplicado ao PSUMAA

Na fase de refinamento do procedimento GRASP-VND (Linha 6 da Figura 4.10) é aplicado o procedimento VND₁ (Figura 4.12) a cada solução construída. Essa heurística de refinamento explora o espaço de soluções usando as vizinhanças N^R e N^T , nesta ordem. Inicialmente é feita uma descida randômica usando-se movimentos de realocação. Para tanto, é escolhida aleatoriamente uma tarefa e uma posição para inserí-la. Se esta nova sequência produzir uma solução com um valor menor para a função de avaliação, a nova sequência é aceita e passa a ser a solução corrente; caso contrário, é testada outra realocação aleatória. A fase de realocação termina quando houver $MRDmax$ iterações consecutivas sem melhora na função de avaliação. Nesta última situação, passa-se a fazer movimentos de troca, também de forma aleatória.

Se for produzida uma solução global de melhor qualidade, interrompe-se esta fase e volta-se para a fase de realocação; caso contrário, prosseguem-se com as trocas. O procedimento é encerrado quando não houver melhora na solução global nem com movimentos de realocação nem com movimentos de troca.

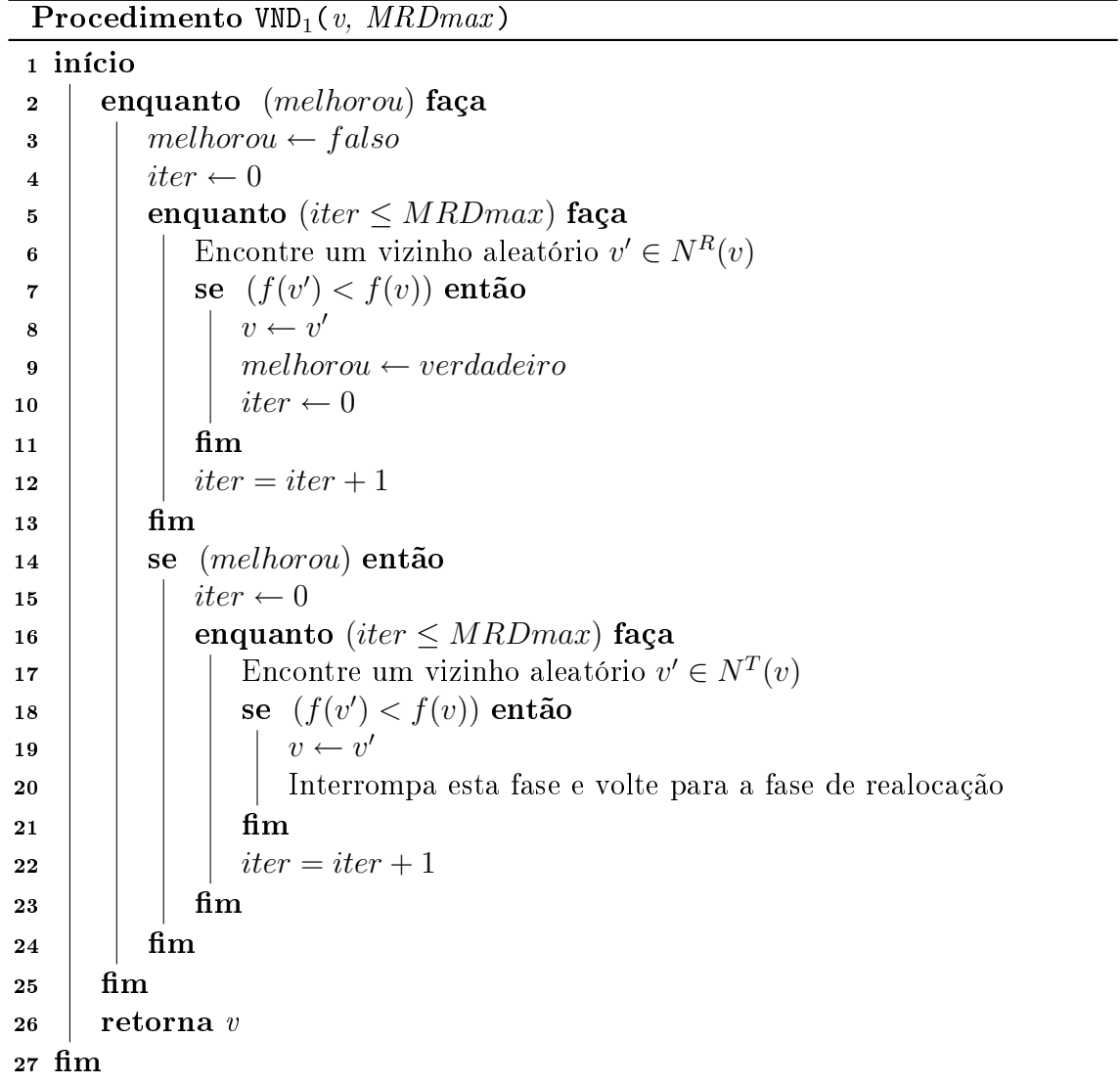


Figura 4.12: Procedimento VND₁ aplicado ao PSUMAA

Deve-se observar que a solução resultante desta busca não é necessariamente um ótimo local com relação às vizinhanças consideradas, visto que a vizinhança completa não é analisada a cada iteração. Em virtude desse fato, decorridas $GRASP_{max}$ iterações, a melhor das soluções obtidas é submetida a novo refinamento (Linha 12 da Figura 4.10), desta vez pelo procedimento VND₂. Este procedimento consiste

em explorar o espaço de soluções usando-se, além dos movimentos tradicionais de realocação e troca, também o movimento *Or* descrito na Subseção 4.3.2.

No procedimento VND₂, a busca é feita de acordo com a seguinte estratégia de exploração do espaço de busca:

1. descida randômica com movimentos de realocação;
2. descida randômica com movimentos de troca;
3. descida randômica com movimentos *Or* de realocação de um bloco de k tarefas ($2 \leq k \leq 3$);
4. descida completa com movimentos de realocação;
5. descida completa com movimentos de troca;
6. descida completa com movimentos *Or* de realocação de um bloco de k tarefas ($2 \leq k \leq n - 2$).

Nas descidas com realocação de um bloco de k tarefas, k assume inicialmente seu valor mais baixo. Havendo alguma melhora, passa-se para a primeira estratégia; caso contrário, k é aumentado progressivamente até atingir seu valor máximo. Em todas as descidas randômicas para-se quando houver *MRDmax* iterações sem melhora. Observa-se que o procedimento VND₂ retorna um ótimo local com respeito às vizinhanças N^R , N^T e N^{Or} .

4.3.5 Busca Tabu aplicada ao PSUMAA

O procedimento Busca Tabu (*Tabu Search - TS*), (Glover e Laguna, 1997) implementado começa sua execução partindo de uma solução inicial gerada pelo procedimento GRASP-VND. A cada iteração todos os vizinhos desta solução são avaliados, sendo escolhido o melhor deles que não seja tabu ou, se tabu, satisfaça a condição de aspiração, no caso, de o vizinho gerar um valor melhor que o da melhor solução até então. A exploração da vizinhança é feita tendo por base as estruturas de vizinhança N^T e N^R , alternando-as a cada iteração, isto é, na primeira iteração explora-se o

melhor vizinho com movimentos de troca, na segunda iteração com movimentos de realocação e assim por diante. O objetivo de se verificar se o movimento é tabu ou não, é evitar ciclos de soluções recentemente visitadas.

Quando um vizinho $v' \in N^T(v) \cup N^R(v)$ é escolhido, ele se torna a solução corrente, independentemente de ele ser melhor ou pior que a solução corrente e um atributo que caracterize esta solução é inserido em uma lista tabu T , proibindo o retorno a ele durante um prazo. Dada a troca entre uma tarefa i e uma tarefa j ($i < j$), o atributo que é inserido na lista tabu é a subsequência (tarefa i , tarefa sucessora de i da solução v). No caso de realocação de uma tarefa i para frente (realocação progressiva), o movimento tabu é a subsequência (tarefa i , sucessor da tarefa i na solução v); caso a realocação seja para trás (realocação regressiva) o movimento tabu é a subsequência (tarefa antecessora a i na solução v , tarefa i). A lista tabu é limitada a $|T|$ elementos. Quando ela está completa, o elemento mais antigo é retirado da lista, e aquele recentemente obtido é inserido, tal como em uma fila (*FIFO*).

Para mostrar o funcionamento da lista tabu, considere a sequência $v = \{5, 3, 2, 1, 4\}$ e um movimento envolvendo a troca da tarefa 5 com a tarefa 1. O resultado é a sequência vizinha $v' = \{1, 3, 2, 5, 4\}$, sendo na lista tabu T incluída a subsequência formada por (tarefa 5, tarefa subsequente a 5 na sequência v), isto é, $T \leftarrow T \cup (5, 3)$.

Já para o movimento de realocação, considere primeiro a realocação da tarefa 5 para depois da tarefa 1, resultando a sequência vizinha $v' = \{3, 2, 1, 5, 4\}$, sendo adicionada na lista tabu T a subsequência formada por (tarefa 5, tarefa subsequente a 5 na sequência v), ou seja, $T \leftarrow T \cup (5, 3)$. Agora considere que o movimento de realocação seja realocar a tarefa 1 para antes da tarefa 5, que resultará na solução vizinha $v' = \{1, 5, 3, 2, 4\}$, sendo que o movimento tabu armazenado, neste caso, será a subsequência formada por (tarefa antecessora a 1, tarefa 1 na sequência v), isto é, $T \leftarrow T \cup (2, 1)$.

Sempre que há melhora na solução global, é aplicado o procedimento VND₂ descrito na Subseção 4.3.4. Caso ele seja bem sucedido, a lista tabu é inicializada

novamente, já que se muda a região do espaço de busca onde a exploração está sendo feita. O procedimento Busca Tabu é encerrado quando forem realizadas TS_{max} iterações sem melhora na solução global.

O pseudocódigo da Busca Tabu está descrito na Figura 4.13. A atualização do Grupo Elite (GE), linhas 11 e 18 da Figura 4.13, é feita de acordo com a Subseção 4.3.6.

Procedimento $TS(v, T , divElite, GE, TS_{max}, MRD_{max})$	
1	início
2	$v^* \leftarrow v$ // Melhor Solução
3	$Iter \leftarrow 0$
4	$MelhorIter \leftarrow 0$ // Iteração mais recente que forneceu v^*
5	$T \leftarrow \emptyset$ // Lista Tabu
6	$GE \leftarrow \emptyset$ // Grupo Elite
7	enquanto $(Iter - MelhorIter \leq TS_{max})$ faça
8	$Iter \leftarrow Iter + 1$
9	Seja $v' \leftarrow v \oplus m$ o melhor elemento de $V \subseteq N(v)$ ($N = N^T$ se $Iter$ for ímpar e $N = N^R$ se $Iter$ for par) tal que o movimento m não seja tabu ($m \notin T$) ou, se tabu, v' atenda a condição de aspiração $f(v') < f(v^*)$
10	Atualize a Lista Tabu T
11	Atualize o Grupo Elite GE
12	$v \leftarrow v'$
13	se $f(v) < f(v^*)$ então
14	$v_1 \leftarrow VND_2(v, MRD_{max})$
15	se $(f(v_1) < f(v))$ então
16	$v \leftarrow v_1$
17	$T \leftarrow \emptyset$
18	Atualize o Grupo Elite GE
19	fim
20	$v^* \leftarrow v$
21	$MelhorIter \leftarrow Iter$
22	fim
23	fim
24	$v \leftarrow v^*$
25	retorna v
26	fim

Figura 4.13: Procedimento Busca Tabu aplicado ao PSUMAA

4.3.6 Reconexão por Caminhos

Reconexão por Caminhos (*Path Relinking* - PR), proposto por Glover (1996), é uma estratégia de intensificação normalmente utilizada como pós-otimização de uma solução, ou refinamento de ótimos locais obtidos durante a busca. Dado um par de soluções, o objetivo da técnica é partir de uma delas, dita solução base, e chegar à outra, dita solução guia, por meio da adição na solução base de atributos da solução guia.

No algoritmo proposto, a Reconexão por Caminhos é utilizada como ferramenta de pós-refinamento aplicada após a execução da Busca Tabu (BT). Para tanto, durante a exploração do espaço de busca pelo procedimento BT é gerado um conjunto de soluções, conhecido como Grupo Elite (GE), que é utilizado pelo procedimento PR. Para fazer parte desse grupo, cada solução candidata deve satisfazer a um dos dois seguintes critérios:

1. ser melhor que a melhor das $|GE|$ soluções do grupo elite;
2. ser melhor que a pior das $|GE|$ soluções do grupo elite e se diferenciar delas de determinado percentual dos atributos, dado por $divElite$.

O atributo considerado é o par de tarefas i e j consecutivas. Assim, por exemplo, dadas as sequências $v_1 = \{1, 4, 3, 2, 5, 6\}$ e $v_2 = \{5, 3, 2, 1, 4, 6\}$, elas têm 40% de atributos iguais, a saber, as subsequências (1, 4) e (3, 2). O objetivo desta estratégia é evitar a inclusão de soluções similares no grupo elite. Estando o grupo elite já formado, quando uma solução entra, a de pior avaliação sai.

O procedimento PR foi implementado como segue. Para cada par de soluções elite, caminha-se de forma bidirecional, isto é, tanto da pior solução para a melhor (dita Reconexão por Caminhos Progressiva - *Forward Path Relinking*), quanto da melhor para a pior solução (dita Reconexão por Caminhos Regressiva - *Backward Path Relinking*).

Determinadas as soluções base e guia, o procedimento é executado inserindo-se gradativamente um atributo da solução guia na solução base. Por exemplo, dada

a solução guia = $\{5, 3, 2, 1, 4, 6\}$, a cada iteração insere-se um par de tarefas consecutivas desta solução, a saber, $(5, 3)$, $(3, 2)$, $(2, 1)$, $(1, 4)$, $(4, 6)$, na solução base. Cada par é inserido na ordem em que aparece na solução guia. Após cada inserção, as tarefas da solução base que foram substituídas saem da posição que ocupavam e assumem as posições daquelas que entraram. A solução base sofre, então, uma busca local, no caso, uma descida randômica usando os movimentos de troca e realocação, tal como descrito na Subseção 4.3.4 onde se fixam os atributos da solução guia que já foram incorporados à solução base. A cada iteração, a solução base recebe todos os atributos ainda não incorporados da solução guia, um de cada vez, gerando uma nova sequência para cada atributo específico da solução guia. Ao final da iteração, o atributo inserido que produziu a melhor solução após a busca local é incorporado em definitivo à solução base, se tornando fixo. O procedimento se encerra quando a solução guia é alcançada, isto é, quando a solução base passa a ter todos os atributos da solução guia.

Para ilustrar o funcionamento do procedimento, é mostrado, a seguir, um exemplo no qual a solução guia é a sequência $\{5, 3, 2, 1, 4, 6\}$ e a solução base é $\{2, 6, 5, 1, 4, 3\}$.

Passo 1: O objetivo neste passo é avaliar a inserção, na solução base, de cada um dos 5 pares de subsequências existentes da solução guia, $\{(5, 3), (3, 2), (2, 1), (1, 4), (4, 6)\}$, na ordem em que aparecem e verificar qual inserção traz o melhor resultado segundo a função de avaliação. Para tanto, procede-se como segue:

Passo 1.1) Inserir as tarefas 5 e 3 do par $(5, 3)$ nas duas primeiras posições da solução base. Para tanto, a tarefa 5 ocupará a primeira posição da solução base, onde está a tarefa 2. Esta, por sua vez, sairá da primeira posição e irá para a terceira posição, onde está a tarefa 5, resultando na sequência $\{5, 6, 2, 1, 4, 3\}$. A seguir, é inserida a tarefa 3 na segunda posição da sequência base. Como nesta posição está a tarefa 6, esta se deslocará para a última posição, onde está localizada a tarefa 3. Após esta inserção obtém-se a sequência $\{5, 3, 2, 1, 4, 6\}$.

Passo 1.2) Inserir as tarefas 3 e 2 do par $(3, 2)$ na segunda e terceira posição, respectivamente, da solução base. Para tanto, a tarefa 3 passará a ocupar a segunda

posição da solução base, onde está a tarefa 6. Esta, por sua vez, sairá da segunda posição e irá para a última, onde está a tarefa 3, resultando na sequência $\{2, 3, 5, 1, 4, 6\}$. A seguir, é inserida a tarefa 2 na terceira posição da sequência base. Como nesta posição está a tarefa 5, esta se deslocará para a primeira posição, onde está localizada a tarefa 2. Após esta inserção obtém-se $\{5, 3, 2, 1, 4, 6\}$.

Passo 1.3) Inserir as tarefas 2 e 1 da subsequência (2, 1) na terceira e quarta posição, respectivamente, da solução base e prosseguir como anteriormente.

Passo 1.4) Inserir as tarefas 1 e 4 da subsequência (1, 4) na quarta e quinta posição, respectivamente, da solução base e prosseguir como nos passos 1.1 e 1.2.

Passo 1.5) Inserir as tarefas 4 e 6 da subsequência (4, 6) na quinta e sexta posição, respectivamente, da solução base e prosseguir como nos passos 1.1 e 1.2.

Dois estratégias de busca local foram consideradas. Na primeira, a cada inserção faz-se uma busca local com o procedimento VND₃ descrito adiante e a inserção que produzir o melhor resultado para a função de avaliação, dentre os passos 1.1 a 1.5, é realizado. Na segunda estratégia, cada inserção é avaliada e apenas aquela que melhorara valor da função de avaliação é submetida a refinamento pelo procedimento VND₃. Este procedimento consiste em uma Descida Randômica usando movimentos de troca seguida de Descida Completa com movimentos de realocação. Neste procedimento, sempre que a Descida Completa com movimentos de realocação produzir uma solução de melhora, retorna-se à Descida Randômica com movimentos de troca. Durante a aplicação do VND₃ é proibida qualquer movimentação das tarefas da solução guia que foram inseridas. Por exemplo, se a sequência do Passo 1 resultar em melhor valor para a função de avaliação, as tarefas 5 e 3 não podem movimentar-se. Assim, ter-se-á inserido na solução base um atributo da solução guia. A inserção dos demais atributos segue um procedimento semelhante, adicionando-se o fato de que durante a busca local, não somente o par recentemente inserido é fixado, mas todos os demais pares de subsequências da solução guia já inseridos também permanecem fixos.

Toda vez que o procedimento PR gera uma solução melhor que a melhor

do grupo elite, a inserção de atributos é interrompida, o grupo elite é atualizado e o processo é reiniciado com a aplicação da Reconexão por Caminhos envolvendo a melhor solução e as demais do Grupo Elite. Se encerradas as inserções dos atributos, o procedimento tiver gerado uma solução melhor que a pior, satisfazendo aos critérios de diversidade estabelecidos anteriormente, o grupo elite é atualizado com a saída de seu pior elemento. Neste caso, também aplica-se a Reconexão por Caminhos envolvendo a solução alterada e as demais do Grupo Elite. O procedimento termina após a Reconexão ser aplicada a todos os pares de solução elite e não houver alteração na melhor solução. Neste caso, retorna-se a melhor solução encontrada, bem como o conjunto elite.

O pseudocódigo do método de Reconexão por Caminhos (*PR*) está descrito na Figura 4.14.

Procedimento PR(v , $divElite$, GE)

```

1 início
2    $f^* \leftarrow \infty$ 
3    $P \leftarrow \emptyset$  // Lista que armazena soluções usadas no processo
4   enquanto (Critério de parada não atingido) faça
5      $Corrente \leftarrow$  Melhor solução do grupo elite
6      $Guia \leftarrow$  Solução escolhida aleatoriamente de  $GE$ , que não esteja
       presente em  $P$ 
7      $P \leftarrow P \cup \{Guia\}$ 
8     enquanto ( $Corrente \neq Guia$ ) faça
9        $melhorAresta \leftarrow \emptyset$ 
10       $melhorValor \leftarrow \infty$ 
11      para  $j = 1$  até total de arestas da solução guia faça
12         $v \leftarrow Corrente \cup \{aresta\ j\ da\ solução\ guia\}$ 
13         $v' \leftarrow BuscaLocal(v)$ 
14        se ( $f(v') < melhorValor$ ) então
15           $melhorAresta \leftarrow j$ 
16           $melhorValor \leftarrow f(v')$ 
17          se ( $melhorValor < f^*$ ) então
18             $v^* \leftarrow v'$ 
19             $melhorValor \leftarrow f^*$ 
20             $Corrente \leftarrow Corrente \cup \{aresta\ melhorAresta\ da\}$ 
              solução guia
21          fim
22        fim
23      fim
24       $v \leftarrow Corrente \cup \{aresta\ melhorAresta\ da\ solução\ guia\}$ 
25    fim
26    se (melhorValor é melhor que a pior solução elite e a solução
       associada diferencia das demais soluções do grupo elite em divElite%)
       então
27      Atualize o grupo elite inserindo a solução corrente no grupo elite e
       elimine a pior solução do grupo elite
28    fim
29  fim
30   $v \leftarrow v^*$ 
31  retorna  $v$  e  $GE$ 
32 fim

```

Figura 4.14: Procedimento PR aplicado ao PSUMAA

4.3.7 Algoritmo GTSPR

O algoritmo proposto, denominado GTSPR, possui três fases e combina características dos procedimentos GRASP, *Variable Neighborhood Descent* (VND), Busca Tabu (*Tabu Search*) e Reconexão por Caminhos (*Path Relinking*). Seu pseudocódigo é apresentado na Figura 4.15.

Algoritmo 7: GTSPR($v, \gamma, GRASP_{max}, MRD_{max}$)	
1	início
2	$v_0 \leftarrow \text{GRASP-VND}(v, \gamma, GRASP_{max}, MRD_{max})$
3	$v_1 \leftarrow \text{TS}(v_0, T , TS_{max}, divElite, GE, MRD_{max})$
4	$v \leftarrow \text{PR}(GE, divElite, MRD_{max})$
5	retorna v
6	fim

Figura 4.15: Algoritmo GTSPR

Capítulo 5

Resultados Computacionais

Neste capítulo são apresentados, na seção 5.1, os problemas-teste utilizados para avaliar os modelos exato e heurístico desenvolvidos no capítulo 4. A calibração e os parâmetros dos métodos heurísticos são apresentados na seção 5.3. Por fim, os resultados computacionais obtidos com a aplicação dos modelos exato e heurístico são apresentados e discutidos, respectivamente, nas seções 5.2 e 5.4.

5.1 Problemas-teste

Para testar o algoritmo proposto foram utilizados os mesmos problemas-teste usados em Gomes Jr. *et al.* (2007). Essa base de dados envolve número de tarefas igual a 8, 9, 10, 11, 12, 15, 20, 25, 30, 40, 50 e 75, e foram geradas por estes autores utilizando os mesmos parâmetros adotados em Wan e Yen (2002), Liaw (1999) e Mazzini e Armentano (2001).

Nessa base de dados os tempos de processamento P_i das tarefas são números inteiros entre 1 e 100, enquanto os centros das janelas de entrega estão distribuídos no intervalo $[(1 - TF - RDD/2)/MS, (1 - TF + RDD/2)/MS]$, sendo MS o tempo total de processamento de todas as tarefas, TF o fator de atraso e RDD a variação relativa da janela de entrega.

Os valores de TF são 0,1; 0,2; 0,3 e 0,4 enquanto que para RDD são 0,8; 1,0 e 1,2. Os tamanhos das janelas de entrega são inteiros no intervalo $[1, MS/n]$. Os custos por atraso da produção (α_i) são números inteiros no intervalo $[20, 100]$,

enquanto que os custos por antecipação da produção (β_i) são k vezes o custo por atraso da mesma tarefa, sendo k um número real aleatório no intervalo $[0, 1]$. Essa última consideração é feita tendo em vista que o atraso da produção é menos desejável do que a sua antecipação na maioria dos casos reais.

Os tempos de preparação da máquina (S_{ij}) são números inteiros no intervalo $[0, 50]$ e são simétricos, ou seja, $S_{ij} = S_{ji}$. Todas as distribuições são uniformes. Considerando-se os diferentes valores de TF e RDD , há 144 problemas-teste no total, sendo 12 para cada número de tarefas.

5.2 Resultados do PLIM

Para realização dos experimentos computacionais do modelo matemático apresentado na Seção 4.1, o modelo foi implementado usando a ferramenta de modelagem AMPL e resolvido pelo software de otimização CPLEX, versão 9.0 da ILOG. Foram resolvidos os problemas-teste de dimensões menores, no caso, de 8 a 12 tarefas, uma vez que acima deste número de tarefas o otimizador parava por estouro de memória. Os resultados encontrados são apresentados na Tabela 5.1. Nesta tabela, a primeira coluna apresenta o número de tarefas do conjunto de problemas-teste, a segunda coluna apresenta o tempo computacional médio gasto para se chegar à solução ótima e a última coluna apresenta o percentual de soluções ótimas encontradas dentro dos 12 problemas-teste relativos a cada número de tarefas. Considerou-se como critério de parada, para obtenção de uma solução ótima, um tempo limite de 3600 segundos.

Tabela 5.1: Resultados do PLIM

Número de tarefas	Tempo médio de processamento (s)	Percentual de Soluções Ótimas Encontradas (%)
8	1,14	100,00
9	24,13	100,00
10	55,29	100,00
11	1.519,30	75,00
12	1.869,55	58,33

5.3 Calibração e Parâmetros do Método Heurístico

O algoritmo heurístico proposto foi desenvolvido na linguagem C++, utilizando o ambiente *NetBeans* 6.5 e o compilador *MinGW* (GCC 4.2.1). Todos os experimentos foram realizados em um computador *Pentium Core 2 Quad* (Q6600) 2,4 GHz com 4 GB de memória RAM e sistema operacional *Windows XP*. Apesar de o processador deste equipamento possuir 4 núcleos, o algoritmo desenvolvido não foi otimizado para multiprocessamento. Cada problema-teste foi executado 30 vezes.

Os parâmetros do algoritmo foram calibrados em uma bateria inicial de testes. A exceção foi o conjunto Γ de valores γ da fase de construção GRASP, que foi o mesmo adotado por Gomes Jr. *et al.* (2007). Todos os testes de calibração foram feitos usando-se um problema-teste de porte médio, de 25 tarefas, com 10 execuções. Calibrado um parâmetro, este era fixado na calibração dos parâmetros subsequentes.

O primeiro parâmetro a ser calibrado foi o número máximo de iterações GRASP, $GRASP_{max}$. O gráfico da Figura 5.1 apresenta os valores da função objetivo (Fo), para o problema-teste adotado, variando-se o valor de $GRASP_{max}$ de 1 a 30. É possível observar que para $GRASP_{max}$ superior a 20, praticamente não há alteração do valor de Fo .

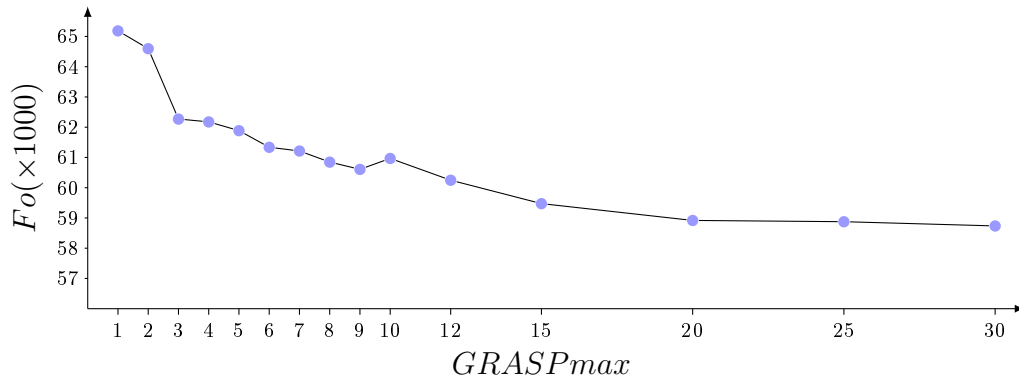


Figura 5.1: Calibração do parâmetro $GRASP_{max}$ em função de Fo

Já o gráfico da Figura 5.2 apresenta o tempo computacional, em segundos, gasto para resolver o problema. É possível observar que o tempo cresce linearmente ao se variar o valor de $GRASP_{max}$.

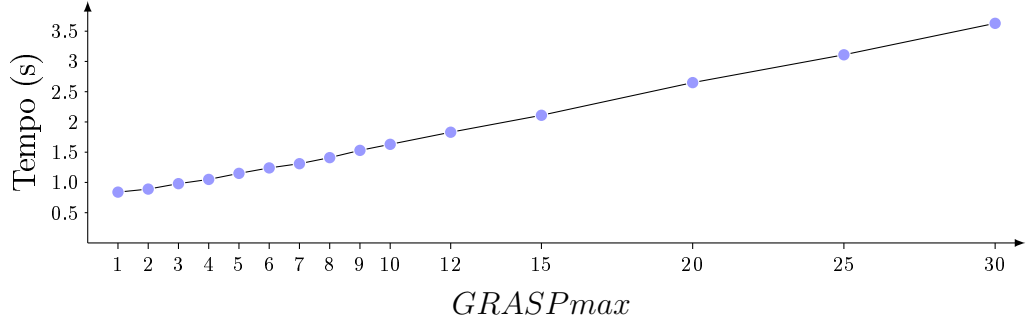


Figura 5.2: Calibração do parâmetro $GRASPmax$ em função do tempo

Ao selecionar o valor de $GRASPmax$, foram adotados dois valores, o primeiro, chamado de principal, foi adotado o valor $GRASPmax = 20$, pois a partir deste ponto o valor de Fo permanece, praticamente, constante. Para o segundo valor, chamado de secundário, deu-se prioridade ao tempo de resposta do método com uma boa solução, sendo o valor adotado $GRASPmax = 6$, pois este apresenta um bom valor para Fo , com um baixo tempo computacional. Para $GRASPmax = 20$, por exemplo, obtém-se uma melhora na solução inferior a 5% mas, em contrapartida, dobra-se o esforço computacional.

O próximo parâmetro calibrado foi o número máximo de iterações sem melhora dos procedimentos de Descida Randômica ($MRDmax$). Neste teste de calibração, os valores de $MRDmax$ foram variados em função de n , onde n é o número de tarefas do problema, de $1 \times n$ a $10 \times n$. A Figura 5.3, apresenta os testes em função de Fo e a Figura 5.4 em função do tempo. O valor adotado foi $MRDmax = 7 \times n$, visto que a partir deste ponto o valor de Fo permanece, praticamente, constante.

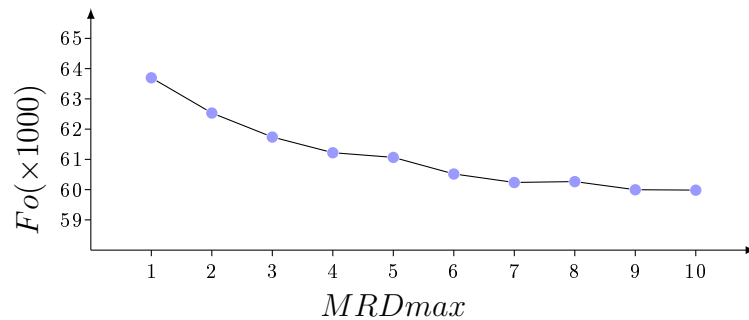


Figura 5.3: Calibração do parâmetro $MRDmax$ em função de Fo

Assim como os parâmetros do GRASP, foram calibrados os parâmetros da

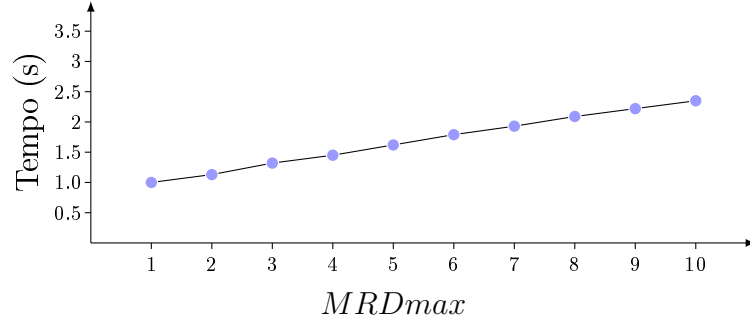


Figura 5.4: Calibração do parâmetro $MRDmax$ em função do tempo

Busca Tabu, $TSmax$, número máximo de iterações sem melhora da Busca Tabu, e $|T|$, tamanho máximo da Lista Tabu.

Os gráficos das Figuras 5.5 e 5.6 apresentam os testes de calibração do parâmetro $TSmax$, em função do valor de Fo e do tempo, respectivamente. É possível observar que o valor de Fo reduz gradativamente para valores de $TSmax$ entre $1 \times n$ e $4 \times n$; a partir deste ponto, Fo praticamente permanece constante. Assim como em $GRASPmax$ foram adotados dois valores, o principal foi $TSmax = 4 \times n$. O valor secundário selecionado para este parâmetro foi $TSmax = 2 \times n$. A justificativa deveu-se ao tempo computacional requerido por $TSmax = 4 \times n$ que produziu solução de melhor qualidade, mas requereu o dobro de tempo daquele demandado por $TSmax = 2 \times n$ para um ganho de cerca de 2%.

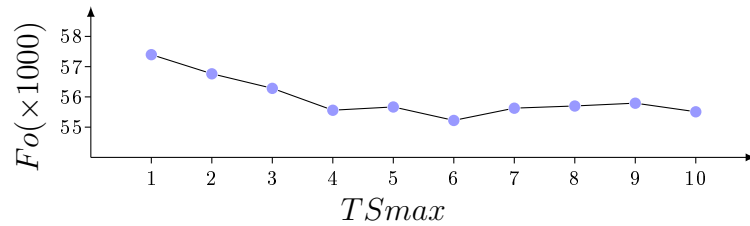


Figura 5.5: Calibração do parâmetro $TSmax$ em função de Fo

Outro parâmetro do método Busca Tabu calibrado foi o tamanho da lista tabu, $|T|$. Os gráficos das Figuras 5.7 e 5.8 apresentam os testes de calibração do mesmo.

Pelo gráfico da Figura 5.7 é possível observar que para valores de $|T|$ entre $2 \times n$ e $8 \times n$, o valor de Fo é praticamente constante. O gráfico mostra, ainda,

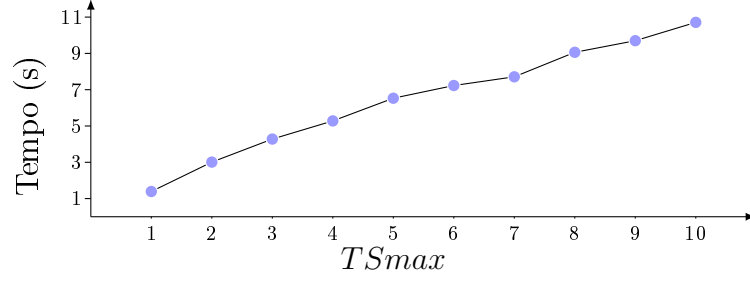


Figura 5.6: Calibração do parâmetro TS_{max} em função do tempo

que para valores muito grandes de $|T|$, no caso, acima de $8 \times n$, os movimentos permitidos ficam restritos porque o valor da função objetivo se degrada. De acordo com o gráfico da Figura 5.8 pode-se observar que o tempo de processamento não sofre influência significativa do tamanho da lista tabu. Assim, adotou-se $|T| = 2 \times n$.

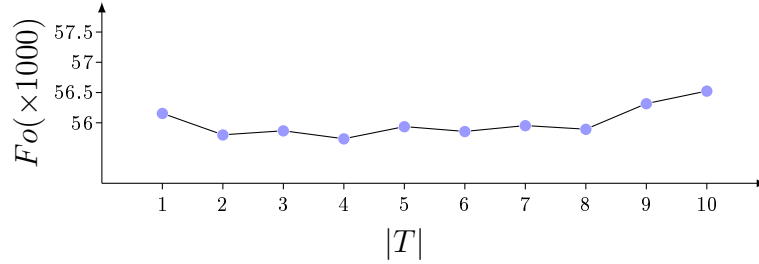


Figura 5.7: Calibração do parâmetro $|T|$ em função de Fo

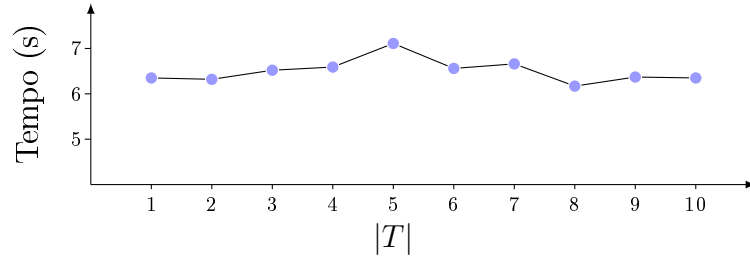


Figura 5.8: Calibração do parâmetro $|T|$ em função do tempo

Por fim, foram calibrados os parâmetros do método Reconexão por Caminhos. O tamanho máximo do Grupo Elite, $|GE|$ e o percentual mínimo de diversificação entre os membros do Grupo Elite, $divElite$.

Os gráficos das Figuras 5.9 e 5.10 mostram os valores de Fo e o tempo de processamento, respectivamente, para a calibração do parâmetro $|GE|$. Levando-se em consideração estes gráficos, foi adotado $|GE| = 5$.

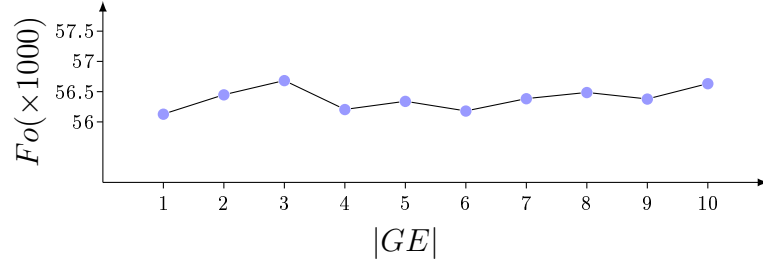


Figura 5.9: Calibração do parâmetro $|GE|$ em função de Fo

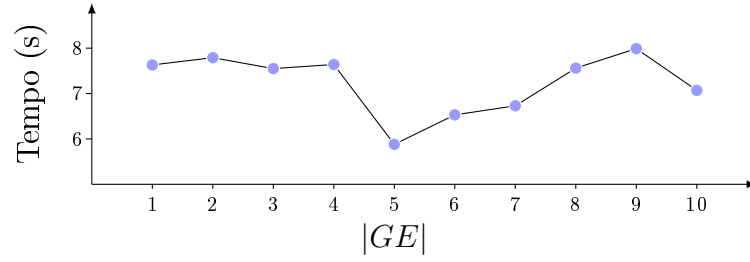


Figura 5.10: Calibração do parâmetro $|GE|$ em função do tempo

Os gráficos das Figuras 5.11 e 5.12 detalham os valores de Fo e o tempo de processamento, respectivamente, para a calibração do parâmetro *divElite*. Por análise semelhante a dos casos anteriores foi adotado $divElite = 40\%$.

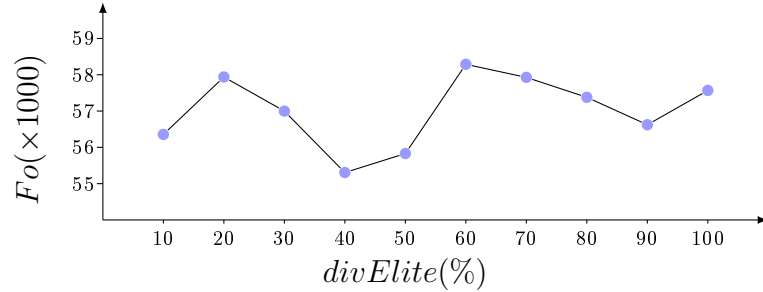


Figura 5.11: Calibração do parâmetro *divElite* em função de Fo

Os parâmetros adotados a partir dos testes de calibração estão sintetizados na Tabela 5.2, onde n representa o número de tarefas.

Nesta tabela, a segunda coluna apresenta os valores que geraram os melhores resultados nos testes de calibração. Já a terceira coluna apresenta os valores para os parâmetros *GRASPmax* *TSmax* que obtiveram resultados satisfatórios, porém com um esforço computacional bem inferior.

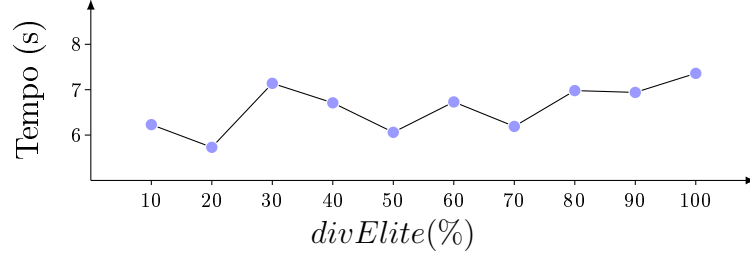


Figura 5.12: Calibração do parâmetro *divElite* em função do tempo

Tabela 5.2: Parâmetros do Algoritmo GTSPR - Solução Completa

Parâmetro	Valores Principais	Valores Secundários
Γ	$\{0; 0,02; 0,04; 0,12; 0,14\}$	$\{0; 0,02; 0,04; 0,12; 0,14\}$
$GRASP_{max}$	20	6
MRD_{max}	$7 \times n$	$7 \times n$
TS_{max}	$4 \times n$	$2 \times n$
$ T $	$2 \times n$	$2 \times n$
$ GE $	5	5
<i>divElite</i>	0,4	0,4

5.4 Resultados do Algoritmo Heurístico

Os resultados encontrados nos experimentos computacionais estão sintetizados nas Tabelas 5.3 e 5.4, onde a primeira coluna indica o número de tarefas do problema, e a cada conjunto de três colunas é apresentada a contribuição de cada fase do algoritmo GTSPR no esforço de minimização da função de avaliação f . A coluna “Fase GRASP-VND” diz respeito à fase de obtenção da solução inicial, a coluna “Fase TS” à fase de refinamento por Busca Tabu e a coluna “Fase PR”, à fase de pós-otimização por Reconexão por Caminhos. As colunas “Tempo” dizem respeito ao tempo médio de processamento acumulado, em segundos.

Para cada fase do algoritmo são aplicadas duas medidas de desempenho, dadas pelas equações (5.1) e (5.2). Nestas equações, para cada problema-teste i do grupo, f_i^{GTSPR*} representa o melhor valor encontrado pelo algoritmo proposto, f_i^{GJr*} é o melhor valor encontrado por Gomes Jr. *et al.* (2007), f_i^* é o valor da melhor solução conhecida e $\bar{f}_i^{GTSPR}$ é o valor médio encontrado pelo algoritmo proposto nas diversas execuções para o i -ésimo problema-teste.

$$imp_i^{best} = \frac{f_i^{GJr^*} - f_i^{GTSPR^*}}{f_i^{GJr^*}} \quad (5.1)$$

$$gap_i^{avg} = \frac{\bar{f}_i^{GTSPR} - f_i^*}{f_i^*} \quad (5.2)$$

Na primeira medida de desempenho, verifica-se o quanto o algoritmo proposto superou o melhor resultado de Gomes Jr. *et al.* (2007). Para tanto, para cada problema-teste, aplica-se a equação (5.1) e apresenta-se nas Tabelas 5.3 e 5.4, para cada fase do algoritmo, a média percentual desses valores para cada grupo de problemas-teste com mesmo número de tarefas.

Na segunda medida de desempenho, é utilizada a equação (5.2) para calcular o desvio das soluções médias do algoritmo em relação às melhores soluções existentes. Nesta medida, quanto menor o desvio (*gap*), menor a variabilidade das soluções finais e mais robusto é o algoritmo. Nas Tabelas 5.3 e 5.4 são apresentadas, para cada fase do algoritmo, as médias percentuais dos valores dos desvios médios para cada grupo de problemas-teste.

A Tabela 5.3 apresenta os resultados adotando-se os parâmetros principais dos testes de calibração utilizando-se a primeira estratégia da Reconexão por Caminhos, isto é, aplicar a busca local VND₃ a cada inserção e escolher a inserção que produzir o melhor resultado.

Pela Tabela 5.3 é possível observar que para problemas com até 20 tarefas, apenas a fase GRASP-VND do algoritmo é suficiente para encontrar as melhores soluções, com um desvio médio de até 1,94%. Nesses problemas-teste, o desvio médio é reduzido na fase TS, ficando em até 0,99%, enquanto que essa variabilidade das soluções finais é reduzida para um máximo de 0,64%, na média, com a fase PR.

Para problemas com 25 a 50 tarefas, observa-se que a fase GRASP-VND retorna resultados piores que os de Gomes Jr. *et al.* (2007). Entretanto, a fase TS consegue reduzir essa diferença e melhorar os resultados destes autores para todos os problemas-teste desta classe. Com a fase PR, as melhores soluções de Gomes Jr. *et al.* (2007) para os problemas-teste de 25 e 30 tarefas são encontradas, enquanto que

Tabela 5.3: Comparação de resultados de cada fase do algoritmo GTSPR $\times$ Gomes Jr. *et al.* (2007) $\times$ CPLEX (Parâmetros Principais)

#	Fase GRASP-VND			Fase TS			Fase PR			Gomes Jr. CPLEX	
Tar.	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	tempo ⁽¹⁾	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	tempo ⁽¹⁾	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	Tempo ⁽¹⁾	Tempo ⁽²⁾	Tempo ⁽¹⁾
	%	%	(s)	%	%	(s)	%	%	(s)	(s)	(s)
8	0,00	0,00	0,03	0,00	0,00	0,04	0,00	0,00	0,06	0,04	1,14
9	0,00	0,00	0,05	0,00	0,00	0,06	0,00	0,00	0,09	0,07	24,13
10	0,00	0,00	0,07	0,00	0,00	0,09	0,00	0,00	0,15	0,11	55,29
11	0,00	0,13	0,10	0,00	0,00	0,14	0,00	0,00	0,25	0,20	1519,30
12	0,00	0,07	0,14	0,00	0,01	0,20	0,00	0,00	0,37	0,29	1869,55
15	0,00	1,15	0,33	0,00	0,63	0,54	0,00	0,47	1,13	0,94	-
20	0,00	1,94	1,06	0,00	0,99	2,03	0,00	0,64	4,93	4,35	-
25	-0,57	2,94	2,59	0,00	1,46	6,13	0,00	1,09	14,90	13,29	-
30	-0,22	4,54	6,24	0,00	2,13	16,78	0,00	1,68	39,93	40,07	-
40	-1,09	7,26	20,77	0,15	3,81	70,20	0,18	3,24	193,61	155,79	-
50	-1,15	10,08	57,36	0,51	5,45	258,63	0,74	4,61	630,77	492,28	-
75	1,38	9,71	399,09	5,32	7,14	2460,59	5,75	6,56	6308,74	1368,08	-
Avg	-0,14	3,17	-	0,50	1,80	-	0,56	1,52	-	-	-

⁽¹⁾ Tempo de CPU, em um PC Intel Core 2 Quad (Q6600) 2,40 GHz com 4 GB de RAM

⁽²⁾ Tempo de CPU, em um PC Athlon XP 64 Bits 3000+ (aprox. 2 GHz) com 1 GB de RAM

para os demais problemas-teste, o algoritmo proposto consegue superar os resultados destes autores na média em até 0,74%. Por fim, os problemas-teste envolvendo 75 tarefas apresentaram melhoras significativas já na primeira fase de 1,38%, passando a 5,75% na terceira fase. Vale ressaltar que para os problemas-teste com 75 tarefas, Gomes Jr. *et al.* (2007) utilizaram parâmetros de calibração diferentes, em seus testes, como meio de reduzir o esforço computacional. Com relação ao desvio médio para esta classe de problemas, a variabilidade das soluções finais reduziu de 10,08% na fase GRASP-VND para 7,14% na fase TS e para 6,56% na fase PR. Observa-se que no algoritmo de Gomes Jr. *et al.* (2007), os desvios médios finais foram de até 18,53%. Este último percentual foi ajustado de acordo com as novas melhores soluções encontradas.

Em termos de tempos computacionais, uma comparação direta de desempenho não é possível, uma vez que as máquinas utilizadas para teste foram diferentes. Contudo, uma comparação mais justa é feita mais adiante, por meio da Tabela 5.5, que apresenta um comparativo de desempenho entre os algoritmos com um ajuste no tempo de processamento do algoritmo de Gomes Jr. *et al.* (2007).

Ainda sobre os tempos computacionais, pode-se verificar que o tempo gasto pelo GTSPR, para os problemas até 12 tarefas foi bem inferior ao tempo gasto pelo CPLEX, apresentado na última coluna, para encontrar a solução ótima. Vale ressaltar que não houve desvio na solução final apresentada pelo algoritmo proposto.

Com a finalidade de melhorar os tempos computacionais exigidos pelo algoritmo proposto, uma segunda bateria de testes foi executada, desta vez, utilizando os parâmetros secundários dos testes de calibração e a segunda estratégia da Reconexão por Caminhos, isto é, a busca local VND₃ é aplicada somente se a inserção de um arco produzir uma melhora. A Tabela 5.4 apresenta estes resultados.

Tabela 5.4: Comparação de resultados de cada fase do algoritmo GTSPR $\times$ Gomes Jr. *et al.* (2007) $\times$ CPLEX (Parâmetros Secundários)

# Tar.	Fase GRASP-VND			Fase TS			Fase PR			Gomes Jr. CPLEX	
	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	tempo ⁽¹⁾	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	tempo ⁽¹⁾	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	Tempo ⁽¹⁾	Tempo ⁽²⁾	Tempo ⁽¹⁾
	%	%	(s)	%	%	(s)	%	%	(s)	(s)	(s)
8	0,00	0,00	0,01	0,00	0,00	0,02	0,00	0,00	0,03	0,04	1,14
9	0,00	0,21	0,02	0,00	0,00	0,03	0,00	0,00	0,03	0,07	24,13
10	0,00	0,35	0,02	0,00	0,08	0,04	0,00	0,08	0,05	0,11	55,29
11	0,00	0,83	0,04	0,00	0,20	0,06	0,00	0,18	0,09	0,20	1519,30
12	0,00	0,83	0,05	0,00	0,52	0,08	0,00	0,43	0,13	0,29	1869,55
15	0,00	3,97	0,12	0,00	2,31	0,24	0,00	2,08	0,32	0,94	-
20	0,00	5,35	0,42	0,00	2,36	1,04	0,00	2,05	1,32	4,35	-
25	-0,66	6,65	1,09	0,00	3,03	3,29	0,00	2,82	4,11	13,29	-
30	-0,67	8,07	2,75	-0,05	3,62	9,83	0,00	3,48	11,63	40,07	-
40	-1,24	10,12	9,98	0,17	5,27	42,29	0,17	5,07	47,64	155,79	-
50	-2,70	13,54	29,88	0,60	7,25	165,99	0,61	7,10	181,12	492,28	-
75	0,58	16,99	256,31	4,53	9,84	1728,05	4,65	9,65	1828,88	1368,08	-
Avg	-0,39	5,58	-	0,44	2,87	-	0,45	2,75	-	-	-

Com esta segunda bateria de testes, como era de se esperar, pode-se observar que os tempos computacionais tiveram uma redução significativa, ficando bem abaixo dos tempos demandados pelo algoritmo de Gomes Jr. *et al.* (2007), mesmo sem o ajuste, com exceção dos problemas com 75 tarefas. Apesar da significativa diminuição dos tempos verifica-se que a qualidade das soluções foi pouco afetada. Para as problemas com até 20 tarefas, continuou sendo necessária apenas a primeira fase do algoritmo proposto para encontrar a melhor solução, com um desvio médio máximo de 5,35%, passando a 2,36% na segunda fase e caindo para 2,05% na ter-

ceira fase. Para os problemas com 25 ou mais tarefas, a fase TS foi suficiente para resolvê-los e melhorar os resultados, à exceção dos problemas com 30 tarefas, em que foi necessária a fase PR. As soluções finais tiveram uma melhora de até 4,65%, com desvio máximo de 9,65%, valor este ainda melhor que os 18,53% obtidos por Gomes Jr. *et al.* (2007). Observa-se que este último percentual foi ajustado de acordo com as novas melhores soluções encontradas.

Pelas Tabelas 5.3 e 5.4 observa-se que a fase PR demanda um tempo computacional elevado e sua contribuição é relativamente baixa, servindo basicamente para diminuir a variabilidade da qualidade da solução final. Observa-se, ainda, que apenas as fases GRASP-VND e TS são suficientes para resolver o problema com um esforço computacional muito menor e sem detrimento significativo da qualidade da solução final.

A Tabela 5.5 apresenta uma comparação dos tempos computacionais das duas baterias de testes do algoritmo GTSPR. A segunda coluna, GTSPR I, apresenta os tempos computacionais com os parâmetros principais, enquanto a terceira coluna, GTSPR II, os tempos com os parâmetros secundários. A quarta coluna apresenta os tempos demandados pelo algoritmo de Gomes Jr. *et al.* (2007) reduzindo-os em 30%. Este valor foi obtido baseado em testes encontrados na página [http://www.tomshardware.com/charts/cpu-charts-2008-q1-2008/compare,372.html?prod\[1275\]=on&prod\[1218\]=on](http://www.tomshardware.com/charts/cpu-charts-2008-q1-2008/compare,372.html?prod[1275]=on&prod[1218]=on), que compara os tempos gastos na execução de programas como antivírus e compactadores nas máquinas envolvidas no presente trabalho. Nestes testes os programas executados em máquinas com processador PC Intel Core 2 Quad (Q6600) obtiveram tempos computacionais em média 30% inferiores aos mesmos programas executados em máquinas com processador PC Athlon XP 64 3000+.

Ao se comparar os tempos computacionais dos algoritmos, vê-se que GTSPR II demandou menor tempo de processamento para produzir soluções com qualidade melhor ou igual àsquelas geradas pelo algoritmo de Gomes Jr. *et al.* (2007). A exceção ocorreu nos problemas de 75 tarefas, nos quais foi exigido um tempo duas vezes maior, mas que resultou, em contrapartida, em soluções finais melhores em

até 4,65%, na média. Além disso, já na fase GRASP-VND, GTSPR II demandou 256,31 segundos para obter soluções, em média 0,58% melhores do que o algoritmo de Gomes Jr. *et al.* (2007) em 957,66 segundos.

Tabela 5.5: Comparação dos tempos computacionais GTSPR I $\times$ GTSPR II $\times$ Gomes Jr. *et al.* (2007) corrigido

# Tarefas	GTSPR I Tempo (s)	GTSPR II Tempo (s)	Gomes Jr. <i>et al.</i> (2007) Tempo corrigido* (s)
08	0,06	0,03	0,03
09	0,09	0,03	0,05
10	0,15	0,05	0,08
11	0,25	0,09	0,14
12	0,37	0,13	0,20
15	1,13	0,32	0,66
20	4,93	1,32	3,05
25	14,90	4,16	9,30
30	39,93	11,54	28,05
40	173,58	47,64	109,05
50	630,77	182,12	344,60
75	6308,74	1828,88	957,66

(*) Tempo de CPU corrigido em 30%, com base em testes comparativos entre performance de processadores

No Anexo I são apresentados os resultados detalhados para os problemas-teste utilizados. Nas Tabelas I.1 a I.4 são mostrados os resultados para a execução GTSPR I e nas Tabelas I.5 a I.8, os resultados do GTSPR II.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho teve seu foco na resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA), considerando janelas de entrega e tempo de preparação da máquina dependente da sequência de produção.

Foi proposto um algoritmo de três fases, denominado GTSPR, que combina os procedimentos GRASP e Descida em Vizinhança Variável (*Variable Neighborhood Descent* - VND), para geração de solução inicial; Busca Tabu (*Tabu Search* - TS) para refinamento e Reconexão por Caminhos (*Path Relinking* - PR), como mecanismo de pós-otimização. Para cada sequência gerada pelo algoritmo são determinadas as datas ótimas de conclusão das tarefas por meio de um procedimento de tempo polinomial da literatura. Para explorar o espaço de soluções são utilizados três tipos de movimentos, baseados em troca entre tarefas, realocação de uma tarefa para outra posição e realocação de um bloco de tarefas para outra posição.

Inicialmente o algoritmo proposto foi comparado com o otimizador CPLEX aplicado a problemas-teste envolvendo até 12 tarefas. Os resultados obtidos mostraram que todas as soluções ótimas foram encontradas em tempos computacionais muito inferiores aos demandados pelo otimizador e o desvio máximo foi de 0,43%.

A seguir, o algoritmo GTSPR foi aplicado a problemas-teste envolvendo até 75 tarefas e comparado com um algoritmo da literatura. Os experimentos computacionais mostraram que, para problemas-teste de até 20 tarefas, apenas a fase

GRASP-VND do algoritmo foi suficiente para alcançar as melhores soluções existentes. Todas as demais fases do algoritmo tiveram sua contribuição para a melhora da qualidade da solução final, reduzindo sua variabilidade. Com o algoritmo proposto foi possível superar os resultados da literatura em até 5,75%, com uma variabilidade média de 6,56%, no máximo. Em termos de tempo computacional, o algoritmo proposto também foi competitivo, sendo capaz de produzir soluções melhores em menor tempo de processamento.

Como trabalhos futuros, indicam-se os seguintes aperfeiçoamentos:

1. Aplicar periodicamente a estratégia de Reconexão por Caminhos durante a exploração da busca, e não somente como mecanismo de pós-otimização;
2. Aplicar o princípio da otimalidade próxima durante a fase de construção de uma solução;
3. Estudar propriedades que possam ser aplicadas ao problema, de forma a reduzir o espaço de busca e, conseqüentemente, o tempo de processamento do algoritmo.

Referências Bibliográficas

- Allahverdi, A.; Gupta, J. N. D. e Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *OMEGA*, v. 27, p. 219–239.
- Allahverdi, Ali; Ng, C.T.; Cheng, T.C.E. e Kovalyov, Mikhail Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, v. 187, n. 3, p. 985–1032.
- Araújo, Francisco César Rodrigues de. (2008). Planejamento operacional de lavra com alocação dinâmica de caminhões: Abordagens exata e heurística. Dissertação de mestrado, Departamento de Engenharia de Minas/EM/UFOP, Ouro Preto.
- Arenales, M.; Armentano, V.; Morabito, R. e Yanasse, H. (2007). *Pesquisa Operacional para cursos de Engenharia*. Editora Campus, Rio de Janeiro.
- Baker, K. R. e Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A review. *Operations Research*, v. 38, p. 22–36.
- Bilge, Ümit; Kurtulan, Müjde e Kirac, Furkan. (2007). A tabu search algorithm for the single machine total weighted tardiness problem. *European Journal of Operational Research*, v. 176, p. 1423–1435.
- Biskup, D. e Feldmann, M. (2001). Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers and Operations Research*, v. 28, p. 787–801.
- Bustamante, L. M. (2006). Minimização do custo de antecipação e atraso para o problema de seqüenciamento de uma máquina com tempo de preparação dependente da seqüência: Aplicação em uma usina siderúrgica. Dissertação de mestrado, Programa de Pós-Graduação em Engenharia de Produção, UFMG, Belo Horizonte.
- Chang, P. C. (1999). A branch and bound approach for single machine scheduling with earliness and tardiness penalties. *Computers & Mathematics with Applications*, v. 37, n. 10, p. 133–144.
- Chen, Zhi-Long. (1997). Scheduling with batch setup times and earliness-tardiness penalties. *European Journal of Operational Research*, v. 96, p. 518–537.

- Coleman, B. J. (1992). A simple model for optimizing the single machine early/tardy problem with sequence-dependent setups. *Production and Operation Management*, v. 1, n. 2, p. 225–228.
- Costa, Felipe Pereira. (2005). Aplicações de técnicas de otimização a problemas de planejamento operacional de lavra em minas a céu aberto. Dissertação de mestrado, Departamento de Engenharia de Minas/EM/UFOP, Ouro Preto.
- Du, J. e Leung, J. Y. T. (1990). Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, v. 15, p. 483–495.
- Feldmann, Martin e Biskup, Dirk. (2003). Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers and Industrial Engineering*, v. 44, p. 307–323.
- Feo, T.A. e Resende, M.G.C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133.
- França Filho, M. F. (2007). Grasp e busca tabu aplicados a problemas de programação de tarefas em máquinas paralelas. Tese de doutorado, Departamento de Engenharia de Sistemas/UNICAMP, Campinas.
- Gagné, Caroline; Price, Wilson L. e Gravel, Marc. (2001). Scheduling a single machine with sequence dependent setup times using ant colony optimization. Relatório Técnico 2001-003, Faculté des sciences de l’administration, Université Laval.
- Gershon, M. (1982). A linear programming approach to mine scheduling optimization. *Proceedings of the 17th Application of computers and operations research in the mineral industry*, p. 483–493, New York.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, v. 5, p. 553–549.
- Glover, F. (1996). *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, Capítulo Tabu search and adaptive memory programming - Advances, applications and challenges, p. 1–75. Kluwer Academic Publishers.
- Glover, F. e Kochenberger, G. (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers.
- Glover, F. e Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Boston.
- Gomes Jr., A. C.; Carvalho, C. R. V.; Munhoz, P. L. A. e Souza, M. J. F. (2007). Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional, SBPO*, p. 1649–1660, Fortaleza.

- Gordon, Valery; Proth, Jean-Marie e Chu, Chengbin. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, v. 139, p. 1–25.
- Gupta, S. R. e Smith, J. S. (2006). Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, v. 175, p. 722–739.
- Hallah, R. M. (2007). Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. *Computers and Operations Research*, v. 34, p. 3126–3142.
- Hassin, Refael e Shani, Mati. (2005). Machine scheduling with earliness, tardiness and non-execution penalties. *Computers and Operations Research*, v. 32, p. 683–705.
- Hino, C. M.; Ronconi, D. P. e Mendes, A. B. (2005). Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, v. 160, p. 190–201.
- Jackson, J. R. (1955). Scheduling a production line to minimize maximum tardiness. Relatório técnico, UCLA, Management Science Research Project.
- James, R. J. W. (1997). Using tabu search to solve the common due date early/tardy machine scheduling problem. *Computers and Operations Research*, v. 24, n. 3, p. 199–208.
- Kim, S. C. e Bobrowski, P. M. (1994). Impact os sequence dependent setup time on job shop scheduling performance. v. 32, n. 7, p. 1503–1520.
- Lee, C. Y. e Choi, J. Y. (1995). A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers and Operations Research*, v. 22, p. 857–869.
- Li, F. (1997). Single machine earliness and tardiness scheduling. *European Journal of Operational Research*, v. 96, p. 546–558.
- Liaw, C. F. (1999). A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers and Operations Research*, v. 26, p. 679–693.
- Lin, Shih-Wei; Chou, Shuo-Yan e Chen, Shih-Chieh. (2007)a. Meta-heuristic approaches for minimizing total earliness and tardiness penalties of single-machine scheduling with a common due date. *Journal of Heuristic*, v. 13, p. 151–165.
- Lin, Shih-Wei; Chou, Shuo-Yan e Ying, Kuo-Ching. (2007)b. A sequential exchange approach for minimizing earliness-tardiness penalties of single-machine scheduling with a common due date. *European Journal of Operational Research*, v. 177, p. 1294–1301.

- Mazzini, R. e Armentano, V. A. (2001). A heuristic for single machine scheduling with early and tardy costs. *European Journal of Operational Research*, v. 128, p. 129–146.
- Mladenović, N. e Hansen, P. (1997). Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097–1100.
- Panwalkar, S. S.; Dudek, R. A. e Smith, M. L. (1973). Sequencing research and the industrial scheduling problem. SE, Elmaghraby, editor, *Symposium on the Theory of Scheduling and its Applications*, p. 29–38, Springer: Berlin.
- Rabadi, G.; Mollaghasemi, M. e Anagnostopoulos, G. C. (2004). A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers and Operations Research*, v. 31, p. 1727–1751.
- Rodrigues, Lásara Fabrícia. (2006). Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto. Dissertação de mestrado, Departamento de Engenharia de Produção/Escola de Engenharia/UFGM, Belo Horizonte.
- Sridharan, V. e Zhou, Z. (1996). A decision theory based scheduling procedure for single-machine weighted earliness and tardiness problems. *European Journal of Operational Research*, v. 94, p. 292–301.
- Tian, Z.J.; Ng, C.T. e Cheng, T.C.E. (2005). On the single machine total tardiness problem. *European Journal of Operational Research*, v. 165, p. 843–846.
- Uzsoy, Reha e Velásquez, Juan Diego. (2008). Heuristics for minimizing maximum lateness on a single machine with family-dependent set-up times. *Computers and Operations Research*, v. 35, p. 2018–2033.
- Ventura, José A. e Radhakrishnan, Sanjay. (2003). Single machine scheduling with symmetric earliness and tardiness penalties. *European Journal of Operational Research*, v. 144, p. 598–612.
- Wan, G. e Yen, B. P. C. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, v. 142, p. 271–281.
- Ying, Kuo-Ching. (2008). Minimizing earliness-tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm. *Computers and Industrial Engineering*, v. In Press, Corrected Proof. doi: 10.1016/j.cie.2008.01.008.

Anexo I

Resultados Detalhados

I.1 Execução do GTSPR I

As Tabelas I.1 a I.4 mostram os resultados detalhados para os problemas-teste de 08 a 75 tarefas. Nestas tabelas, imp_i^{avg} é dado pela Expressão(I.1) e representa a melhora do algoritmo GTSPR I sobre o algoritmo de Gomes Jr. *et al.* (2007) em relação a solução média encontrado por cada um.

Na Expressão (I.1), $\bar{f}_i^{GJr}$ representa o valor médio obtido pelo algoritmo de Gomes Jr. *et al.* (2007) em trinta execuções e o outro membro é o mesmo da Equação (5.2).

$$imp_i^{avg} = \frac{\bar{f}_i^{GJr} - \bar{f}_i^{GTSPR}}{\bar{f}_i^{GJr}} \quad (I.1)$$

Tabela I.1: Resultados detalhados GTSPR I $\times$ Gomes Jr. *et al.* (2007) (8 a 10 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR I	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR I	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR I
0801	4928	4928	0,00	4928	4928	0,00	0,00	0,05	0,06
0802	2739	2739	0,00	2739	2739	0,00	0,00	0,03	0,05
0803	4074	4074	0,00	4074	4074	0,00	0,00	0,02	0,05
0804	17149	17149	0,00	17149	17149	0,00	0,00	0,05	0,07
0805	6195	6195	0,00	6195	6195	0,00	0,00	0,03	0,05
0806	4004	4004	0,00	4004	4004	0,00	0,00	0,03	0,06
0807	10689	10689	0,00	10689	10689	0,00	0,00	0,05	0,06
0808	14059	14059	0,00	14059	14059	0,00	0,00	0,04	0,06
0809	13705	13705	0,00	13751	13705	0,33	0,00	0,05	0,06
0810	19855	19855	0,00	19855	19855	0,00	0,00	0,05	0,08
0811	22774	22774	0,00	22774	22774	0,00	0,00	0,04	0,06
0812	6184	6184	0,00	6184	6184	0,00	0,00	0,03	0,05
Média			0,00			0,03	0,00	0,04	0,06
0901	11801	11801	0,00	11801	11801	0,00	0,00	0,07	0,08
0902	8561	8561	0,00	8561	8561	0,00	0,00	0,05	0,08
0903	4734	4734	0,00	4734	4734	0,00	0,00	0,05	0,09
0904	31100	31100	0,00	31124	31100	0,08	0,00	0,08	0,09
0905	2986	2986	0,00	2986	2986	0,00	0,00	0,06	0,09
0906	13845	13845	0,00	13845	13845	0,00	0,00	0,06	0,09
0907	16400	16400	0,00	16512	16400	0,68	0,00	0,08	0,09
0908	20817	20817	0,00	20817	20817	0,00	0,00	0,07	0,10
0909	14431	14431	0,00	14431	14431	0,00	0,00	0,05	0,07
0910	25664	25664	0,00	25664	25664	0,00	0,00	0,09	0,09
0911	33202	33202	0,00	33202	33202	0,00	0,00	0,08	0,10
0912	13068	13068	0,00	13068	13068	0,00	0,00	0,06	0,09
Média			0,00			0,06	0,00	0,07	0,09
1001	14411	14411	0,00	14456	14411	0,31	0,00	0,13	0,15
1002	8349	8349	0,00	8349	8349	0,00	0,00	0,11	0,15
1003	11605	11605	0,00	11605	11605	0,00	0,00	0,10	0,14
1004	12486	12486	0,00	12486	12486	0,00	0,00	0,12	0,16
1005	5679	5679	0,00	5679	5679	0,00	0,00	0,12	0,14
1006	2897	2897	0,00	2897	2897	0,00	0,00	0,09	0,13
1007	5515	5515	0,00	5515	5515	0,00	0,00	0,12	0,15
1008	9534	9534	0,00	9534	9534	0,00	0,00	0,10	0,16
1009	9461	9461	0,00	9461	9461	0,00	0,00	0,09	0,14
1010	22273	22273	0,00	22273	22273	0,00	0,00	0,12	0,16
1011	20514	20514	0,00	20514	20514	0,00	0,00	0,09	0,16
1012	45554	45554	0,00	45554	45554	0,00	0,00	0,10	0,14
Média			0,00			0,03	0,00	0,11	0,15

Tabela I.2: Resultados detalhados GTSPR I $\times$ Gomes Jr. *et al.* (2007) (11 a 15 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR I	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR I	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR I
1101	16530	16530	0,00	16530	16530	0,00	0,00	0,16	0,23
1102	13252	13252	0,00	13252	13252	0,00	0,00	0,18	0,28
1103	15251	15251	0,00	15251	15251	0,00	0,00	0,14	0,23
1104	17573	17573	0,00	17573	17573	0,00	0,00	0,19	0,23
1105	9347	9347	0,00	9347	9347	0,00	0,00	0,19	0,23
1106	15737	15737	0,00	15737	15737	0,00	0,00	0,22	0,22
1107	12311	12311	0,00	12311	12311	0,00	0,00	0,21	0,24
1108	17361	17361	0,00	17374	17361	0,07	0,00	0,21	0,25
1109	18202	18202	0,00	18337	18202	0,74	0,00	0,21	0,22
1110	28886	28886	0,00	28977	28886	0,31	0,00	0,23	0,27
1111	30328	30328	0,00	30328	30328	0,00	0,00	0,20	0,26
1112	76689	76689	0,00	76911	76689	0,29	0,00	0,24	0,29
Média			0,00			0,12	0,00	0,20	0,25
1201	8137	8137	0,00	8183	8137	0,56	0,00	0,26	0,37
1202	9848	9848	0,00	9858	9848	0,10	0,00	0,23	0,28
1203	8002	8002	0,00	8019	8002	0,21	0,00	0,23	0,25
1204	29466	29466	0,00	29477	29466	0,04	0,00	0,35	0,38
1205	8468	8468	0,00	8468	8468	0,00	0,00	0,27	0,34
1206	10982	10982	0,00	10982	10982	0,00	0,00	0,22	0,28
1207	26939	26939	0,00	27033	26939	0,35	0,00	0,36	0,43
1208	11335	11335	0,00	11335	11335	0,00	0,00	0,27	0,40
1209	28610	28610	0,00	28967	28610	1,23	0,00	0,31	0,37
1210	23406	23406	0,00	23420	23406	0,06	0,00	0,37	0,41
1211	23858	23858	0,00	23858	23858	0,00	0,00	0,31	0,45
1212	41983	41983	0,00	41983	41983	0,00	0,00	0,29	0,40
Média			0,00			0,21	0,00	0,29	0,37
1501	18276	18276	0,00	18276	18276	0,00	0,00	0,64	0,93
1502	19622	19622	0,00	20198	19622	2,85	0,00	1,23	1,23
1503	11505	11505	0,00	11952	11505	3,74	0,00	1,09	1,23
1504	15164	15164	0,00	15250	15164	0,56	0,00	1,00	1,16
1505	12924	12924	0,00	13141	12924	1,65	0,00	0,79	1,01
1506	9396	9396	0,00	9423	9396	0,29	0,00	0,60	0,61
1507	46544	46544	0,00	46977	46724	0,54	0,39	1,23	1,26
1508	24899	24899	0,00	26201	26220	-0,07	5,31	1,19	1,14
1509	14457	14457	0,00	14457	14457	0,00	0,00	0,56	0,90
1510	33128	33128	0,00	33556	33128	1,28	0,00	1,11	1,39
1511	42522	42522	0,00	42866	42522	0,80	0,00	1,12	1,41
1512	12793	12793	0,00	12793	12793	0,00	0,00	0,68	1,29
Média			0,00			0,97	0,47	0,94	1,13

Tabela I.3: Resultados detalhados GTSPR I $\times$ Gomes Jr. *et al.* (2007) (20 a 30 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR I	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR I	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR I
2001	16294	16294	0,00	16294	16294	0,00	0,00	1,95	3,69
2002	18107	18107	0,00	18660	18173	2,61	0,37	3,63	3,75
2003	20249	20249	0,00	20461	20425	0,18	0,87	3,07	3,04
2004	29941	29941	0,00	30283	29952	1,09	0,04	4,43	4,17
2005	59158	59158	0,00	59736	59535	0,34	0,64	4,60	5,85
2006	31053	31053	0,00	32456	32016	1,35	3,10	4,86	5,13
2007	40438	40438	0,00	42005	41127	2,09	1,70	4,84	5,83
2008	29186	29186	0,00	29368	29186	0,62	0,00	3,18	3,93
2009	67827	67827	0,00	69246	67827	2,05	0,00	5,61	6,49
2010	34283	34283	0,00	34656	34424	0,67	0,41	5,52	5,32
2011	54538	54538	0,00	54911	54692	0,40	0,28	5,90	5,54
2012	79599	79599	0,00	80171	79780	0,49	0,23	4,58	6,45
Média			0,00			0,99	0,64	4,35	4,93
2501	23397	23397	0,00	23480	23397	0,35	0,00	13,27	13,23
2502	48540	48540	0,00	48801	48968	-0,34	0,88	13,24	12,04
2503	18503	18503	0,00	18526	18579	-0,28	0,41	9,32	12,05
2504	25645	25645	0,00	25785	25645	0,54	0,00	13,44	13,33
2505	35865	35865	0,00	36796	35931	2,35	0,18	14,14	17,13
2506	14189	14189	0,00	14219	14189	0,21	0,00	7,85	9,94
2507	37313	37313	0,00	37857	37377	1,27	0,17	15,26	12,72
2508	44638	44638	0,00	45150	44638	1,13	0,00	13,23	15,83
2509	12839	12839	0,00	12874	12839	0,27	0,00	10,52	17,64
2510	51415	51415	0,00	57174	54850	4,06	6,68	18,25	19,11
2511	44808	44808	0,00	47610	46682	1,95	4,18	15,87	17,78
2512	34197	34197	0,00	35255	34408	2,40	0,62	15,10	18,03
Média			0,00			1,16	1,09	13,29	14,90
3001	43673	43673	0,00	44778	43760	2,27	0,20	38,91	37,30
3002	41983	41983	0,00	42111	42021	0,21	0,09	39,48	32,27
3003	21951	21951	0,00	22375	21951	1,89	0,00	24,60	32,00
3004	64943	64943	0,00	68138	66751	2,04	2,78	57,49	47,78
3005	74100	74100	0,00	76040	75170	1,14	1,44	46,12	45,98
3006	29829	29829	0,00	30832	30872	-0,13	3,50	33,51	38,17
3007	74336	74336	0,00	75378	74553	1,09	0,29	41,04	38,50
3008	69770	69770	0,00	74097	73579	0,70	5,46	53,20	47,03
3009	21335	21335	0,00	21641	21392	1,15	0,27	22,73	25,67
3010	73702	73702	0,00	77065	75359	2,21	2,25	48,36	55,35
3011	35190	35190	0,00	38152	36054	5,50	2,45	31,09	36,78
3012	83976	83976	0,00	86008	85138	1,01	1,38	44,30	42,31
Média			0,00			1,59	1,68	40,07	39,93

Tabela I.4: Resultados detalhado GTSPR I $\times$ Gomes Jr. *et al.* (2007) (40 a 75 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR I	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR I	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR I
4001	57086	57086	0,00	57751	57765	-0,02	1,19	160,29	164,00
4002	49306	49306	0,00	49705	49587	0,24	0,57	108,29	144,03
4003	28643	28643	0,00	28834	28643	0,66	0,00	24,42	49,44
4004	99871	99828	0,04	102171	101837	0,33	2,01	210,85	183,61
4005	29863	29863	0,00	30594	30441	0,50	1,94	96,74	120,22
4006	32303	32303	0,00	33403	32637	2,29	1,03	83,50	91,80
4007	1E+05	116387	0,56	125443	120900	3,62	3,88	186,75	276,44
4008	44847	44847	0,00	46653	47026	-0,80	4,86	122,02	184,43
4009	77378	77378	0,00	82618	80884	2,10	4,53	189,65	212,59
4010	93591	93225	0,39	97763	98435	-0,69	5,59	230,40	245,15
4011	1E+05	105285	1,11	119856	114125	4,78	8,40	202,31	385,15
4012	1E+05	127269	0,00	135908	133552	1,73	4,94	254,26	230,48
Média			0,18			1,23	3,24	155,79	190,61
5001	105563	105426	0,13	111204	110175	0,93	4,50	585,6	683,99
5002	69631	69244	0,56	73810	72342	1,99	4,47	258,97	418,61
5003	60259	60259	0,00	61219	60548	1,10	0,48	233,27	400,68
5004	96858	96837	0,02	101444	100143	1,28	3,41	529,48	644,10
5005	80597	80140	0,57	82473	80837	1,98	0,87	323,06	440,39
5006	64500	64500	0,00	65746	65204	0,82	1,09	364,45	423,80
5007	129284	129284	0,00	140856	135675	3,68	4,94	680,62	852,58
5008	77165	76565	0,78	80279	78290	2,48	2,25	426,34	507,99
5009	55213	54234	1,77	60435	58728	2,83	8,29	584,56	690,93
5010	171912	163308	5,00	181267	177134	2,28	8,47	700,17	1004,17
5011	176332	176332	0,00	196956	187016	5,05	6,06	739,21	908,14
5012	83686	83686	0,00	93811	92443	1,46	10,46	481,63	593,82
Média			0,74			2,16	4,61	492,28	630,77
7501	258776	254755	1,55	277913	261408	5,94	2,61	1809,40	5572,48
7502	107779	106674	1,03	115331	109111	5,39	2,28	906,60	4861,01
7503	85960	85514	0,52	92820	87205	6,05	1,98	238,10	2625,59
7504	311870	300068	3,78	324516	313573	3,37	4,50	1416,50	6498,56
7505	116346	114722	1,40	130908	121305	7,34	5,74	1201,70	4297,68
7506	142313	139119	2,24	152408	142931	6,22	2,74	608,40	2784,47
7507	332419	320697	3,53	369624	330764	10,51	3,14	1876,10	12394,99
7508	218812	206014	5,85	255181	231806	9,16	12,52	1838,50	4450,51
7509	202863	175759	13,36	224472	198164	11,72	12,75	1417,60	6490,84
7510	359904	320983	10,81	392750	345894	11,93	7,76	1889,00	11881,95
7511	221263	185530	16,15	274439	215045	21,64	15,91	1673,90	7610,41
7512	137306	125184	8,83	155229	133749	13,84	6,84	1541,30	6236,35
Média			5,75			9,43	6,56	1368,09	6308,74

I.2 Execução do GTSPR II

As Tabelas I.5 a I.8 mostram os resultados detalhados para os problemas-teste de 08 a 75 tarefas, para o algoritmo GTSPR II.

As colunas representadas pelas expressões imp^{best} , imp^{avg} e gap^{avg} , são obtidas pelas Equações (5.1), (I.1) e (5.2), respectivamente.

Tabela I.5: Resultados detalhados GTSPR II $\times$ Gomes Jr. *et al.* (2007) (8 a 10 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR II	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR II	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR II
0801	4928	4928	0,00	4928	4928	0,00	0,00	0,05	0,03
0802	2739	2739	0,00	2739	2739	0,00	0,00	0,03	0,02
0803	4074	4074	0,00	4074	4074	0,00	0,00	0,02	0,02
0804	17149	17149	0,00	17149	17149	0,00	0,00	0,05	0,03
0805	6195	6195	0,00	6195	6195	0,00	0,00	0,03	0,02
0806	4004	4004	0,00	4004	4004	0,00	0,00	0,03	0,03
0807	10689	10689	0,00	10689	10689	0,00	0,00	0,05	0,03
0808	14059	14059	0,00	14059	14059	0,00	0,00	0,04	0,03
0809	13705	13705	0,00	13751	13705	0,33	0,00	0,05	0,03
0810	19855	19855	0,00	19855	19855	0,00	0,00	0,05	0,03
0811	22774	22774	0,00	22774	22774	0,00	0,00	0,04	0,03
0812	6184	6184	0,00	6184	6184	0,00	0,00	0,03	0,03
Média			0,00			0,03	0,00	0,04	0,03
0901	11801	11801	0,00	11801	11801	0,00	0,00	0,07	0,03
0902	8561	8561	0,00	8561	8561	0,00	0,00	0,05	0,03
0903	4734	4734	0,00	4734	4734	0,00	0,00	0,05	0,03
0904	31100	31100	0,00	31124	31100	0,08	0,00	0,08	0,07
0905	2986	2986	0,00	2986	2986	0,00	0,00	0,06	0,03
0906	13845	13845	0,00	13845	13845	0,00	0,00	0,06	0,03
0907	16400	16400	0,00	16512	16400	0,68	0,00	0,08	0,03
0908	20817	20817	0,00	20817	20817	0,00	0,00	0,07	0,04
0909	14431	14431	0,00	14431	14431	0,00	0,00	0,05	0,03
0910	25664	25664	0,00	25664	25664	0,00	0,00	0,09	0,03
0911	33202	33202	0,00	33202	33202	0,00	0,00	0,08	0,03
0912	13068	13068	0,00	13068	13068	0,00	0,00	0,06	0,03
Média			0,00			0,06	0,00	0,07	0,03
1001	14411	14411	0,00	14456	14411	0,31	0,00	0,13	0,06
1002	8349	8349	0,00	8349	8349	0,00	0,00	0,11	0,05
1003	11605	11605	0,00	11605	11605	0,00	0,00	0,10	0,05
1004	12486	12486	0,00	12486	12486	0,00	0,00	0,12	0,06
1005	5679	5679	0,00	5679	5735	-0,98	0,98	0,12	0,05
1006	2897	2897	0,00	2897	2897	0,00	0,00	0,09	0,05
1007	5515	5515	0,00	5515	5515	0,00	0,00	0,12	0,06
1008	9534	9534	0,00	9534	9534	0,00	0,00	0,10	0,04
1009	9461	9461	0,00	9461	9461	0,00	0,00	0,09	0,04
1010	22273	22273	0,00	22273	22273	0,00	0,00	0,12	0,05
1011	20514	20514	0,00	20514	20514	0,00	0,00	0,09	0,06
1012	45554	45554	0,00	45554	45554	0,00	0,00	0,10	0,06
Média			0,00			-0,06	0,08	0,11	0,05

Tabela I.6: Resultados detalhados GTSPR II $\times$ Gomes Jr. *et al.* (2007) (11 a 15 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR II	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR II	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR II
1101	16530	16530	0,00	16530	16530	0,00	0,00	0,16	0,07
1102	13252	13252	0,00	13252	13252	0,00	0,00	0,18	0,07
1103	15251	15251	0,00	15251	15251	0,00	0,00	0,14	0,06
1104	17573	17573	0,00	17573	17573	0,00	0,00	0,19	0,07
1105	9347	9347	0,00	9347	9451	-1,11	1,11	0,19	0,07
1106	15737	15737	0,00	15737	15737	0,00	0,00	0,22	0,08
1107	12311	12311	0,00	12311	12311	0,00	0,00	0,21	0,09
1108	17361	17361	0,00	17374	17361	0,07	0,00	0,21	0,10
1109	18202	18202	0,00	18337	18356	-0,10	0,84	0,21	0,10
1110	28886	28886	0,00	28977	28939	0,13	0,18	0,23	0,11
1111	30328	30328	0,00	30328	30328	0,00	0,00	0,20	0,10
1112	76689	76689	0,00	76911	76708	0,26	0,02	0,24	0,12
Média			0,00			-0,06	0,18	0,20	0,09
1201	8137	8137	0,00	8183	8333	-1,84	2,41	0,26	0,14
1202	9848	9848	0,00	9858	9875	-0,18	0,28	0,23	0,09
1203	8002	8002	0,00	8019	8002	0,21	0,00	0,23	0,10
1204	29466	29466	0,00	29477	29493	-0,05	0,09	0,35	0,14
1205	8468	8468	0,00	8468	8532	-0,75	0,75	0,27	0,11
1206	10982	10982	0,00	10982	10982	0,00	0,00	0,22	0,09
1207	26939	26939	0,00	27033	27093	-0,22	0,57	0,36	0,15
1208	11335	11335	0,00	11335	11335	0,00	0,00	0,27	0,14
1209	28610	28610	0,00	28967	28726	0,83	0,40	0,31	0,15
1210	23406	23406	0,00	23420	23563	-0,61	0,67	0,37	0,15
1211	23858	23858	0,00	23858	23858	0,00	0,00	0,31	0,13
1212	41983	41983	0,00	41983	41983	0,00	0,00	0,29	0,15
Média			0,00			-0,22	0,43	0,29	0,13
1501	18276	18276	0,00	18276	18276	0,00	0,00	0,64	0,23
1502	19622	19622	0,00	20198	20486	-1,42	4,40	1,23	0,34
1503	11505	11505	0,00	11952	12204	-2,11	6,07	1,09	0,32
1504	15164	15164	0,00	15250	15164	0,56	0,00	1,00	0,31
1505	12924	12924	0,00	13141	13275	-1,02	2,71	0,79	0,23
1506	9396	9396	0,00	9423	9396	0,29	0,00	0,60	0,28
1507	46544	46544	0,00	46977	48238	-2,68	3,64	1,23	0,35
1508	24899	24899	0,00	26201	26762	-2,14	7,48	1,19	0,40
1509	14457	14457	0,00	14457	14457	0,00	0,00	0,56	0,23
1510	33128	33128	0,00	33556	33343	0,63	0,65	1,11	0,42
1511	42522	42522	0,00	42866	42522	0,80	0,00	1,12	0,44
1512	12793	12793	0,00	12793	12793	0,00	0,00	0,68	0,30
Média			0,00			-0,59	2,08	0,94	0,32

Tabela I.7: Resultados detalhados GTSPR II $\times$ Gomes Jr. *et al.* (2007) (20 a 30 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR II	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR II	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR II
2001	16294	16294	0,00	16294	16294	0,00	0,00	1,95	0,63
2002	18107	18107	0,00	18660	18690	-0,16	3,22	3,63	1,04
2003	20249	20249	0,00	20461	20523	-0,30	1,35	3,07	0,94
2004	29941	29941	0,00	30283	30305	-0,07	1,22	4,43	1,26
2005	59158	59158	0,00	59736	59828	-0,15	1,13	4,60	1,60
2006	31053	31053	0,00	32456	32869	-1,27	5,85	4,86	1,46
2007	40438	40438	0,00	42005	41970	0,08	3,79	4,84	1,66
2008	29186	29186	0,00	29368	29329	0,13	0,49	3,18	0,92
2009	67827	67827	0,00	69246	69638	-0,57	2,67	5,61	2,01
2010	34283	34283	0,00	34656	35340	-1,97	3,08	5,52	1,52
2011	54538	54538	0,00	54911	55168	-0,47	1,15	5,90	1,35
2012	79599	79599	0,00	80171	80086	0,11	0,61	4,58	1,51
Média			0,00			-0,39	2,05	4,35	1,32
2501	23397	23397	0,00	23480	23397	0,35	0,00	13,27	4,56
2502	48540	48540	0,00	48801	49723	-1,89	2,44	13,24	3,92
2503	18503	18503	0,00	18526	18612	-0,46	0,59	9,32	2,85
2504	25645	25645	0,00	25785	25774	0,04	0,50	13,44	3,31
2505	35865	35865	0,00	36796	36294	1,37	1,20	14,14	4,69
2506	14189	14189	0,00	14219	14247	-0,19	0,41	7,85	2,34
2507	37313	37313	0,00	37857	37441	1,10	0,34	15,26	4,04
2508	44638	44638	0,00	45150	44725	0,94	0,19	13,23	4,77
2509	12839	12839	0,00	12874	12926	-0,41	0,68	10,52	3,70
2510	51415	51415	0,00	57174	56951	0,39	10,77	18,25	5,89
2511	44808	44808	0,00	47610	48314	-1,48	7,82	15,87	4,82
2512	34197	34197	0,00	35255	37232	-5,61	8,88	15,10	4,42
Média			0,00			-0,49	2,82	13,29	4,11
3001	43673	43673	0,00	44778	45494	-1,60	4,17	38,91	11,47
3002	41983	41983	0,00	42111	42356	-0,58	0,89	39,48	10,42
3003	21951	21951	0,00	22375	22640	-1,18	3,14	24,60	7,52
3004	64943	64943	0,00	68138	68012	0,18	4,73	57,49	13,09
3005	74100	74100	0,00	76040	76219	-0,24	2,86	46,12	11,04
3006	29829	29829	0,00	30832	31223	-1,27	4,67	33,51	9,13
3007	74336	74336	0,00	75378	75977	-0,79	2,21	41,04	13,57
3008	69770	69770	0,00	74097	75045	-1,28	7,56	53,20	14,99
3009	21335	21335	0,00	21641	21430	0,98	0,44	22,73	8,91
3010	73702	73702	0,00	77065	76423	0,83	3,69	48,36	15,18
3011	35190	35190	0,00	38152	36080	5,43	2,53	31,09	12,34
3012	83976	83976	0,00	86008	88115	-2,45	4,93	44,30	11,92
Média			0,00			-0,16	3,48	40,07	11,63

Tabela I.8: Resultados detalhado GTSPR II $\times$ Gomes Jr. *et al.* (2007) (40 a 75 tarefas)

Melhor Valor				Valor Médio				Tempo (s)	
Inst.	Gomes Jr.	GTSPR II	$\overline{imp}^{best}$ %	Gomes Jr.	GTSPR II	$\overline{imp}^{avg}$ %	$\overline{gap}^{avg}$ %	Gomes Jr.	GTSPR II
4001	57086	57086	0,00	57751	58647	-1,55	2,73	160,29	41,36
4002	49306	49306	0,00	49705	49710	-0,01	0,82	108,29	31,44
4003	28643	28643	0,00	28834	28734	0,35	0,32	24,42	9,79
4004	99871	99958	-0,09	102171	103701	-1,50	3,88	210,85	51,90
4005	29863	29863	0,00	30594	31172	-1,89	4,38	96,74	27,60
4006	32303	32303	0,00	33403	33226	0,53	2,86	83,50	28,44
4007	117047	115809	1,06	125443	123347	1,67	6,51	186,75	77,35
4008	44847	45016	-0,38	46653	47087	-0,93	4,99	122,02	44,48
4009	77378	77378	0,00	82618	84155	-1,86	8,76	189,65	50,66
4010	93591	93225	0,39	97763	100000	-2,29	7,27	230,40	65,16
4011	106470	105285	1,11	119856	115764	3,41	9,95	202,31	74,57
4012	127269	127269	0,00	135908	137977	-1,52	8,41	254,26	68,97
Média			0,17			-0,47	5,07	155,79	47,64
5001	105563	105563	0,00	111204	113621	-2,17	7,77	585,6	183,06
5002	69631	69244	0,56	73810	73037	1,05	5,48	258,97	109,61
5003	60259	60259	0,00	61219	61170	0,08	1,51	233,27	87,32
5004	96858	98256	-1,44	101444	102947	-1,48	6,31	529,48	182,79
5005	80597	80140	0,57	82473	82202	0,33	2,57	323,06	109,37
5006	64500	64500	0,00	65746	65425	0,49	1,43	364,45	108,38
5007	129284	130142	-0,66	140856	139967	0,63	8,26	680,62	254,06
5008	77165	76565	0,78	80279	79509	0,96	3,84	426,34	152,00
5009	55213	54234	1,77	60435	60088	0,57	10,79	584,56	205,22
5010	171912	162543	5,45	181267	178738	1,39	9,96	700,17	365,52
5011	176332	175894	0,25	196956	196028	0,47	11,45	739,21	258,43
5012	83686	83686	0,00	93811	96938	-3,33	15,84	481,63	157,65
Média			0,61			-0,08	7,10	492,28	181,12
7501	258776	251835	2,68	277913	268341	3,44	6,55	1809,4	2055,98
7502	107779	108941	-1,08	115331	113506	1,58	6,40	906,6	798,65
7503	85960	85518	0,51	92820	89363	3,72	4,50	238,1	518,65
7504	311870	306688	1,66	324516	316658	2,42	5,53	1416,5	2090,87
7505	116346	115275	0,92	130908	124542	4,86	8,56	1201,7	1508,24
7506	142313	139592	1,91	152408	142360	6,59	2,33	608,4	689,80
7507	332419	323935	2,55	369624	340627	7,84	6,21	1876,1	2872,38
7508	218812	207038	5,38	255181	238090	6,70	15,57	1838,5	2207,77
7509	202863	172563	14,94	224472	204257	9,01	18,37	1417,6	1551,93
7510	359904	326491	9,28	392750	352438	10,26	9,80	1889	3403,45
7511	221263	200971	9,17	274439	222299	19,00	19,82	1673,9	2174,02
7512	137306	126587	7,81	155229	140441	9,53	12,19	1541,3	2074,80
Média			4,65			7,08	9,65	1368,09	1828,88

Anexo II

Publicações

Lista-se a seguir os trabalhos oriundos desta pesquisa que foram apresentados em eventos nacionais e internacionais.

Trabalhos apresentados em eventos internacionais

Título: *GRASP, Tabu Search and Path Relinking for solving total earliness/tardiness single machine scheduling problem with distinct due windows and sequence-dependent setups.*

Autores: Marcone Jamilson Freitas Souza, Puca Huachi Vaz Penna, Luiz Satoru Ochi e Frederico Augusto de Cezar Almeida Gonçalves.

Evento: *XXIX CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering.*

Local: Maceió/AL - Brasil.

Período: 4 a 7 de novembro de 2008.

Trabalhos apresentados em eventos nacionais

Título: Uma heurística híbrida para minimizar custos com antecipação e atraso em sistemas de produção com janelas de entrega e tempos de preparação dependentes da sequência.

Autores: Marcone Jamilson Freitas Souza, Puca Huachi Vaz Penna, Luiz Satoru Ochi e Frederico Augusto de Cezar Almeida Gonçalves.

Evento: SPOLM 2008 - Simpósio de Pesquisa Operacional e Logística da Marinha.

Local: Rio de Janeiro - RJ.

Período: 5 e 6 de agosto de 2008.

Título: GRASP, VND, Busca Tabu e Reconexão por Caminhos para o Problema de Sequenciamento em uma máquina com tempos de preparação dependentes da sequência de produção, janelas de entrega distintas e penalidades por antecipação e atraso da produção.

Autores: Marcone Jamilson Freitas Souza, Puca Huachi Vaz Penna, e Frederico Augusto de Cezar Almeida Gonçalves.

Evento: XL SBPO - Simpósio Brasileiro de Pesquisa Operacional 2008.

Local: João Pessoa - PB.

Período: 2 a 5 de setembro de 2008.



CEFET-MG

**CENTRO FEDERAL DE EDUCAÇÃO
TECNOLÓGICA DE MINAS GERAIS**

Diretoria de Pesquisa e Pós-Graduação

**Programa de Mestrado em Modelagem
Matemática e Computacional**

HEURÍSTICAS PARA O PROBLEMA DE SEQUENCIAMENTO EM UMA MÁQUINA COM PENALIDADES POR ANTECIPAÇÃO E ATRASO DA PRODUÇÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, como parte dos requisitos necessários para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Aluno: Bruno Ferreira Rosa

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza (UFOP)

Co-Orientador: Prof. Dr. Sérgio Ricardo de Souza (CEFET-MG)

Belo Horizonte, Dezembro de 2009.

Dedico este trabalho aos meus pais José Lázaro e Isilene, e à minha namorada Fernanda, que estão ao meu lado em todos os momentos e foram os principais motivadores deste trabalho. Amo muito vocês!

Agradecimentos

Agradeço a Deus pela vida, por me guardar nas muitas viagens necessárias para a execução deste trabalho.

Aos meus pais pelo exemplo de amor, união, honestidade e determinação.

Aos meus irmãos pela amizade e por estarem sempre ao meu lado, principalmente nos momentos de descontração.

Ao grande amor da minha vida Fernanda por ser minha confidente, minha companheira, pela compreensão e pelo seu amor incondicional.

A todos os meus familiares que sempre me apoiaram em minhas escolhas.

Aos professores Marcone e Sérgio por me aceitarem, por acreditarem na minha capacidade, pelas essenciais orientações, pela paciência e disposição em me ajudar.

Aos professores e funcionários do Programa de Pós-Graduação do CEFET-MG pela oportunidade de buscar e aprimorar meus conhecimentos.

Aos amigos do MMC que estiveram presentes na execução deste trabalho. A amizade e colaboração de vocês foram fundamentais.

Aos colegas que fiz na “estrada” pelas caronas, pela confiança depositada em mim e por compartilharem suas histórias de vida.

Ao CEFET-MG pelo apoio e incentivo.

Enfim, os meus sinceros agradecimentos a todos aqueles que colaboraram, mesmo que indiretamente, para a realização deste trabalho.

Resumo

Este trabalho trata do problema de sequenciamento de tarefas em uma máquina. No problema abordado, os tempos de preparação da máquina são dependentes da sequência de produção e cada tarefa está associada a um tempo de processamento e uma janela de tempo, dentro da qual ela deve ser preferencialmente concluída. O objetivo é minimizar a soma ponderada dos atrasos e das antecipações na execução de tais tarefas. Em termos práticos, as penalidades por antecipação são devidas aos custos gerados pela necessidade de estocagem, enquanto as penalidades por atraso são consequências de multas contratuais. São propostas duas novas formulações de programação matemática para representar o problema, sendo a primeira delas um aperfeiçoamento de uma formulação da literatura, e, a outra, uma formulação indexada no tempo. Propôs-se, também, um algoritmo heurístico para determinar a melhor sequência de produção. Tal algoritmo é composto de duas fases. Na primeira, gera-se uma solução com base na metaheurística GRASP, no Princípio da Otimalidade Próxima e na Descida em Vizinhança Variável. Na segunda, faz-se o pós-refinamento da solução proveniente da fase anterior por meio de outra Descida em Vizinhança Variável. Com o objetivo de reduzir o custo computacional, adicionalmente é proposta uma estratégia de redução do espaço de busca. Devido a particularidade dos problemas-teste da única base de dados existente na literatura referente ao problema tratado, foram geradas outras duas novas bases de dados, uma das quais mais genérica. Experimentos computacionais mostram que a formulação indexada no tempo possibilita resolver problemas maiores e com maior eficiência, quando comparada com as formulações de programação matemática da literatura. O algoritmo heurístico proposto mostrou-se competitivo com os demais algoritmos existentes na literatura, tendo o baixo custo computacional como o seu grande aliado. Desta forma, foi possível resolver, em tempo computacional viável, problemas de dimensões maiores que os resolvidos até então.

Palavras-Chave: Sequenciamento em uma máquina, Formulação indexada no tempo, GRASP, Princípio da Otimalidade Próxima, Descida em Vizinhança Variável.

Abstract

This work deals with the problem of job scheduling in a single-machine. In the considered problem, the machine's setup time is dependent on production sequence and each job has a processing time and a time window within which it should preferably be completed. The objective is to minimize the weighted sum of the tardiness and earliness in the execution of such jobs. In practical terms, the advance penalties are due to the costs incurred by the need for storage, whereas late-payment penalties are the contractual penalties result. It was proposed two new mathematical programming formulations to problem representation, the first being an improved literature formulation, and the other a time-indexed formulation. It was also proposed an heuristic algorithm to determine the best production sequence. This algorithm is composed of two phases. At first, it generates a solution based on GRASP, Proximate Optimality Principle and Variable Neighborhood Descent. In the second, it is the solution post-refinement from the previous phase through another Variable Neighborhood Descent. In order to reduce the computational cost is further proposed a strategy to reduce the search space. Due to the instances particularity exist only one database in the literature for the problem approached, we generated two other new databases, one of them more generic. Computational experiments show that time-indexed formulation allows solving larger instances and more efficiently when compared with others mathematical programming formulations. The heuristic algorithm was shown to be competitive with other literature algorithms, and having low computational cost as its ally. Thus, it was possible to resolve in computing time feasible, greater dimensions instances than those solved until this date.

Keyword: Single-machine scheduling, Time-indexed formulation, GRASP, Proximate Optimality Principle, Variable Neighborhood Descent.

Sumário

Lista de Abreviaturas e Siglas	viii
Lista de Figuras	x
Lista de Algoritmos	xi
Lista de Tabelas	xii
1 Introdução	1
1.1 Objetivos	3
1.1.1 Objetivo Geral	3
1.1.2 Objetivos Específicos	3
1.2 Motivação	3
1.3 Estrutura do Trabalho	5
2 Problema Abordado e Métodos de Resolução	6
2.1 Características do PSUMAA-JE-TPD	6
2.2 Métodos de Resolução	8
2.2.1 Métodos Exatos	8
2.2.2 Métodos Heurísticos	10
2.2.3 Metaheurísticas	13
2.2.4 Estratégias de Intensificação	23
2.3 Trabalhos relacionados	24
2.3.1 PSUMAA com Datas de Entrega Comuns	24
2.3.2 PSUMAA com Datas de Entrega Distintas	26
2.3.3 PSUMAA com Janelas de Entrega Distintas	27
3 Formulações de Programação Matemática	29
3.1 Modelo de Gomes Júnior <i>et al.</i> (2007) - MPLIM-G	29
3.2 Primeiro Modelo Proposto - MPLIM-BG	30
3.3 Segundo Modelo Proposto - MPLIM-IT	31
4 Algoritmo Proposto	33
4.1 GPV	33
4.2 Representação de uma Solução	33
4.3 Vizinhança de uma Solução	33
4.4 Função de Avaliação	35
4.4.1 Algoritmo de Determinação das Datas Ótimas de Início de Proce- samento das Tarefas	35
4.5 Construção de uma Solução	38

4.6	VND ₁	40
4.7	VND ₂	41
4.8	Redução do Espaço de Busca - GPV-REB	41
5	Resultados Computacionais	44
5.1	Problemas-teste	44
5.2	Resultados Obtidos com as Formulações de Programação Matemática . . .	46
5.2.1	Primeira Bateria de Testes - BDS	47
5.2.2	Segunda Bateria de Testes - BDNS	47
5.2.3	Terceira Bateria de Testes - BDDT	48
5.3	Resultados Obtidos com os Algoritmos GPV e GPV-REB	49
5.3.1	Definição dos Parâmetros	50
5.3.2	Primeira Bateria de Testes - BDS	54
5.3.3	Segunda Bateria de Testes - BDNS	56
5.3.4	Terceira Bateria de Testes - BDDT	58
5.4	Teste de Probabilidade Empírica	60
6	Conclusão	62
	Publicações	64
	Referências Bibliográficas	65

Lista de Abreviaturas e Siglas

ADDOIPT Algoritmo de Determinação das Datas Ótimas de Início de Processamento das Tarefas

AE Algoritmos Evolutivos

AG Algoritmos Genéticos

BB *Branch-and-Bound*

BL Busca Local

EDD *Earliest Due Date* - data de entrega mais cedo

FIFO *First-In-First-Out* - o primeiro que entrar é o primeiro a sair

GRASP *Greedy Randomized Adaptive Search Procedure* - Procedimento de busca adaptativa gulosa e randomizada

HC Heurísticas Construtivas

ILS *Iterated Local Search*

JIT *Just in Time*

LRC Lista Restrita de Candidatos

PD Programação Dinâmica

PLIM Programação Linear Inteira Mista

POP Princípio da Otimalidade Próxima

PR *Path Relinking* - Reconexão por Caminhos

PSUMAA Problema de sequenciamento em uma máquina com penalidades por antecipação e atraso na produção

PSUMAA-JE-TPD PSUMAA com janelas de entrega distintas, tempos de *setup* dependentes da sequência de produção e tempos ociosos entre as execuções de tarefas consecutivas permitidos

SA *Simulated Annealing*

SS *Scatter Search*

TS *Tabu Search* - Busca Tabu

VND *Variable Neighborhood Descent* - Busca em Vizinhança Variável

Lista de Figuras

1.1	Aplicação em uma indústria siderúrgica.	4
2.1	Exemplo de dois posicionamentos distintos para as tarefas de uma mesma sequência de produção.	7
2.2	Exemplo de vizinhança de uma solução.	11
2.3	Comportamento das heurísticas de refinamento.	14
2.4	Estrutura básica de um Algoritmo Evolutivo.	22
4.1	Exemplo de situação em que a estratégia de redução do espaço de busca é utilizada.	42
5.1	Resultados dos testes para definição de γ	51
5.2	Resultados dos testes para definição de MRD_{max}	51
5.3	Resultados dos testes para definição de VND_{max}	52
5.4	Resultados dos testes para definição de $GRASP_{max}$	53
5.5	Teste de probabilidade empírica.	61

Lista de Algoritmos

1	Método da descida.	12
2	Método primeira melhora.	13
3	Método descida randômica.	14
4	Busca em Vizinhança Variável.	16
5	GRASP.	16
6	Fase de construção de um algoritmo GRASP.	17
7	Busca Tabu.	19
8	<i>Simulated Annealing</i>	20
9	<i>Iterated Local Search</i>	21
10	Método GPV	34
11	Algoritmo para determinar as datas ótimas de início de processamento das tarefas.	37
12	Construção de uma solução com POP.	38

Lista de Tabelas

4.1	Resultados apresentados pelas funções adaptativas gulosas testadas.	39
5.1	Comparação MPLIM-G $\times$ MPLIM-BG aplicados à BDS.	47
5.2	Comparação MPLIM-G $\times$ MPLIM-BG aplicados à BDNS.	48
5.3	Comparação MPLIM-G $\times$ MPLIM-BG $\times$ MPLIM-IT aplicados à BDDT.	49
5.4	Parâmetros utilizados.	53
5.5	Comparação GPV $\times$ Gomes Júnior (2007) $\times$ Penna (2009) $\times$ Ribeiro (2009) aplicados à BDS.	55
5.6	Comparação GPV $\times$ GPV-REB aplicados à BDS.	56
5.7	Comparação GPV $\times$ GPV-REB $\times$ Ribeiro (2009) aplicados à BDNS.	57
5.8	Comparação GPV $\times$ GPV-REB $\times$ Ribeiro (2009) aplicados à BDDT.	59
5.9	Resultados do MPLIM-IT associado ao GPV.	60

Capítulo 1

Introdução

O surgimento do sistema de administração *Just in Time* (JIT), que ocorreu nos meados da década de 70, evidenciou a importância de um planejamento criterioso das atividades produtivas. Visando a redução dos custos provenientes de processos produtivos, a filosofia JIT desencoraja, além dos atrasos, também as antecipações das tarefas. Há antecipação de uma tarefa quando ela é concluída antes da data desejada para sua entrega e há atraso quando ela é concluída após tal data. Deste modo, ambas as situações acarretam penalidades.

De acordo com Liaw (1999), concluir uma tarefa com atraso pode resultar em multas contratuais, perda de credibilidade da empresa e redução de vendas. Do mesmo modo, concluir uma tarefa antecipadamente pode resultar em custos financeiros extras pela necessidade de disponibilização antecipada de capital, necessidade de espaço para armazenamento ou necessidade de outros recursos para manter e gerenciar o estoque (França Filho, 2007).

O problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção, de agora em diante denotado por PSUMAA, consiste em sequenciar e determinar o momento em que as tarefas devem ser executadas em uma máquina, com o objetivo de minimizar a soma ponderada das antecipações e dos atrasos na produção de tais tarefas. Tal problema, segundo Baker & Scudder (1990), reflete melhor ambientes de produção administrados conforme a filosofia JIT.

Outros objetivos relacionados a essa filosofia de produção também são encontrados na literatura, entre eles: minimizar a soma das penalidades por antecipações e atrasos juntamente com os tempos ociosos gerados durante a produção das tarefas (Sourd, 2005), minimizar a soma da maior antecipação com o maior atraso ocorridos na sequência de execução das tarefas (Tavakkoli-Moghaddam *et al.*, 2006) e minimizar a soma dos quadrados das antecipações e dos atrasos ocorridos no decorrer do processo de produção (Weng & Ventura, 1996; Mondal, 2002).

Com relação às datas de entrega das tarefas, o PSUMAA pode ser dividido em três variantes:

- (i) datas de entrega comuns (*common due date*): todas as tarefas devem ser, preferencialmente, concluídas em uma única data pré-determinada;
- (ii) datas de entrega distintas (*distinct due dates*): existe uma data de entrega específica associada a cada tarefa, na qual a tarefa deve ser preferencialmente concluída;

- (iii) janelas de entrega distintas (*distinct due windows*): há um determinado período de tempo associado a cada tarefa, dentro do qual a tarefa deve ser preferencialmente concluída.

O caso de janelas de entrega distintas ocorre quando existem tolerâncias em torno das datas desejadas para a entrega de cada tarefa (Koulamas, 1996). Estas tolerâncias estão relacionadas às características individuais das tarefas e influenciam nos tamanhos das janelas de entrega. As tarefas concluídas dentro de suas respectivas janelas de entrega não incorrem em nenhuma penalidade. Já aquelas concluídas fora de suas janelas de entrega, são penalizadas. A produção de bens perecíveis é um exemplo relacionado. Assuma que um fabricante de produtos químicos combina uma certa substância A , que deteriora rapidamente, com uma segunda substância B para produzir um produto C . Se A for produzida suficientemente antes de B , ela deteriorará. Por outro lado, se A for produzida muito depois de B , o custo da produção de C será maior.

Nas indústrias em que são produzidos diferentes tipos de produtos e existe uma troca frequente do tipo de tarefa executada em uma máquina, após a conclusão de uma tarefa, geralmente é necessário preparar a máquina antes do início da execução da tarefa seguinte (Allahverdi *et al.*, 1999). Este tempo de preparação, chamado tempo de *setup*, inclui os tempos gastos para trocar as ferramentas, preparar o material, limpar a máquina etc. A maioria dos trabalhos em problemas de sequenciamento assume que os tempos de *setup* são independentes da sequência de produção, isto é, que eles são desprezíveis ou podem ser acrescentados aos tempos de processamento das tarefas (Gupta & Smith, 2006). No entanto, de acordo com Panwalkar *et al.* (1973), *apud* Gupta & Smith (2006), em grande parte das situações práticas, tais tempos são dependentes da sequência de produção. Christoforetti (2002) cita o exemplo de uma fábrica de papel que produz diversos tipos de folhas com diferentes cores, espessuras e texturas e que realiza frequentemente a preparação de certas máquinas para obter os diferentes tipos de produtos. Em problemas como este, os tempos de *setup* variam de acordo com sequência de produção e representam uma parcela de tempo considerável em relação ao tempo total de processamento. Portanto, eles não podem ser desconsiderados.

Considerações sobre a continuidade do funcionamento da máquina também podem ser impostas ao PSUMAA. Parte dos trabalhos da literatura não permite a inserção de tempos ociosos na sequência de produção (Chang, 1999). Conforme Li (1997), existem casos em que o custo por manter a máquina inativa é maior que o preço pago pela antecipação de uma tarefa e, neste caso, vale a pena antecipar a produção. Outra situação em que isto ocorre é quando a capacidade da máquina é inferior à demanda. No entanto, há casos em que vale a pena manter a máquina parada, mesmo que exista uma tarefa disponível para ser processada (Souza *et al.*, 2008).

Este trabalho trata o PSUMAA com janelas de entrega distintas, tempos de preparação dependentes da sequência de produção e são permitidos tempos ociosos entre as execuções de tarefas consecutivas. Para clareza de entendimento e evitar dúvidas, o PSUMAA com tais características será doravante denotado por PSUMAA-JE-TPD. Apesar de este problema ser uma generalização do PSUMAA, ele não tem recebido a atenção merecida na literatura.

1.1 Objetivos

Nesta Seção são apresentados os objetivos deste trabalho.

1.1.1 Objetivo Geral

Este trabalho trata o problema de sequenciamento de tarefas em uma máquina com penalidades por antecipação e atraso da produção na sua forma mais genérica. O objetivo principal é desenvolver um algoritmo heurístico robusto e capaz de encontrar soluções de boa qualidade para tal problema em tempo de execução viável.

1.1.2 Objetivos Específicos

Com a finalidade de alcançar o objetivo geral, os seguintes objetivos específicos foram traçados:

- Fazer um levantamento dos trabalhos da literatura que tratam o problema abordado e problemas relacionados, bem como os métodos utilizados para resolvê-los;
- Gerar novos conjuntos de problemas-teste do problema em estudo;
- Desenvolver um algoritmo heurístico para resolver o problema;
- Desenvolver novos modelos de programação matemática para representar o problema abordado;
- Resolver os problemas-teste de dimensões menores por meio dos modelos de programação matemática desenvolvidos e disponíveis na literatura;
- Resolver os problemas-teste disponíveis por meio do algoritmo heurístico implementado e, em seguida, analisar e discutir os resultados encontrados.
- Divulgar os resultados do trabalho em eventos científicos, bem como em periódicos especializados.

1.2 Motivação

O PSUMAA com janelas de entregas distintas e tempos de *setup* dependentes da sequência de produção possui muitas aplicações práticas em indústrias metalúrgicas, têxteis, de tintas, entre outras. O caso real de uma indústria siderúrgica é tratado em Bustamante (2007). Em tal trabalho, uma máquina é considerada como sendo uma sequência de laminadores e cada tarefa representa a produção de um determinado produto (barra chata, cantoneira, vergalhão etc.). Antes da fabricação de cada produto, é necessário realizar um conjunto de ajustes de mesma natureza na sequência de laminadores. Estes ajustes dependem do produto a ser fabricado e do produto fabricado anteriormente. Como cada ajuste exige um tempo de execução, a soma destes tempos configura o tempo de *setup*. Esta situação é ilustrada na Figura 1.1.

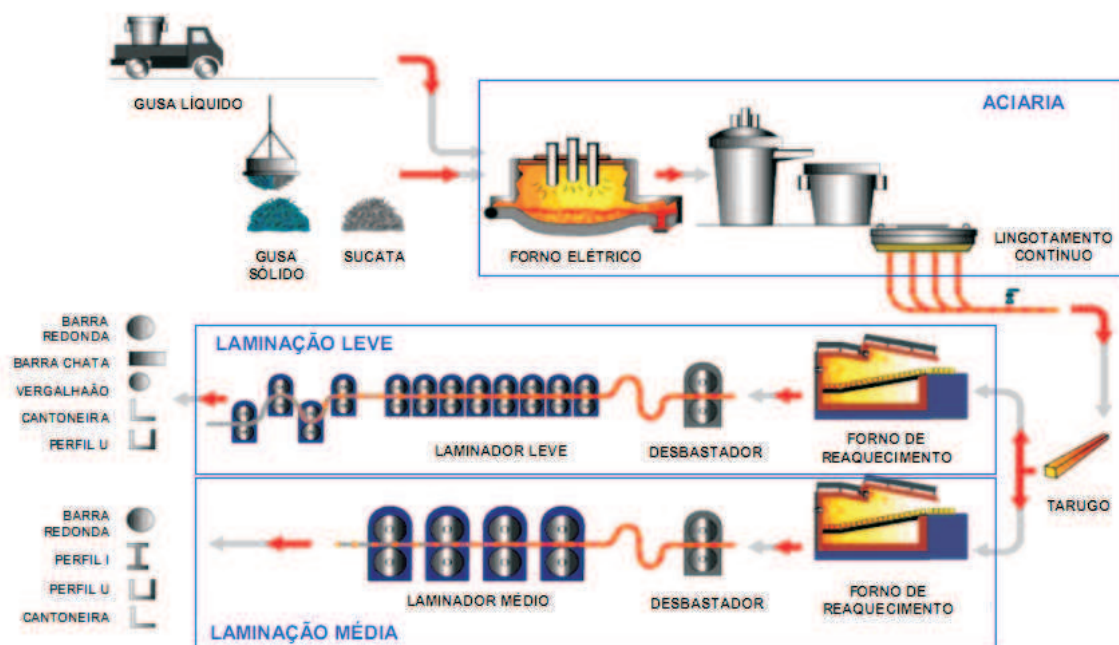


Figura 1.1: Aplicação em uma indústria siderúrgica.

Além do grande número de aplicações, associado ao PSUMAA está a dificuldade de resolvê-lo de forma ótima. Tal dificuldade cresce fatorialmente à medida em que se aumenta a quantidade de tarefas a serem sequenciadas. Esta união entre aplicabilidade e dificuldade de resolução motiva o desenvolvimento de algoritmos eficientes para resolvê-lo.

A utilização de métodos exatos pode demandar um tempo proibitivo para encontrar a solução ótima do PSUMAA, mesmo para problemas de médio porte. Isso justifica a utilização de algoritmos heurísticos, que são procedimentos capazes de encontrar uma “boa solução”, não necessariamente a solução ótima, em tempo computacional viável.

Neste trabalho são utilizados o GRASP (Feo & Resende, 1995) e o Princípio da Otimidade Próxima (Glover & Laguna, 1997) como base para o algoritmo proposto devido aos seguintes motivos: o primeiro por ser uma metaheurística que possui poucos parâmetros para serem ajustados e que se mostrou eficiente na resolução de diversos problemas combinatórios e o segundo, para refinar as soluções parciais e, assim, reduzir o custo computacional da fase de busca local do GRASP.

O desenvolvimento de novas formulações de programação matemática para o problema é motivada pelo fato de existir na literatura apenas um modelo que representa o PSUMAA-JE-TPD. A utilização de um modelo indexado no tempo é motivada pelo trabalho de de Paula (2008), o qual utilizou tal modelagem num problema semelhante e obteve bons resultados. Uma formulação de programação matemática similar à de Gomes Júnior (2007), porém com menor número de variáveis e de restrições, também é proposta na esperança de que ela possibilite resolver problemas de maiores dimensões, quando comparada à formulação de tais autores.

1.3 Estrutura do Trabalho

O restante deste trabalho está organizado como segue.

No Capítulo 2 é feita uma descrição detalhada do PSUMAA-JE-TPD. Um resumo das principais técnicas pertinentes à resolução desse problema também é apresentado. O capítulo é encerrado com a descrição dos trabalhos mais relevantes da literatura em relação ao PSUMAA-JE-TPD.

A formulação de programação matemática proposta por Gomes Júnior (2007) para representar o problema, uma nova formulação baseada nesta última e um modelo de programação matemático indexado no tempo são apresentados no Capítulo 3.

No Capítulo 4 é apresentado o algoritmo heurístico proposto para resolver o PSUMAA-JE-TPD.

Os resultados computacionais obtidos com as formulações de programação matemática propostas e com o algoritmo heurístico desenvolvido são apresentados e discutidos no Capítulo 5.

O Capítulo 6 conclui o trabalho.

Capítulo 2

Problema Abordado e Métodos de Resolução

Neste capítulo é feita uma apresentação das características do problema abordado, seguida pela apresentação dos métodos existentes para resolver problemas similares. Na Seção 2.3 é feita uma revisão da literatura relacionada ao PSUMAA-JE-TPD e problemas correlatos.

2.1 Características do PSUMAA-JE-TPD

O problema de sequenciamento de tarefas em uma máquina tratado neste trabalho (PSUMAA-JE-TPD) possui as seguintes características:

- Uma máquina deve processar um conjunto I de n tarefas;
- Associado a cada tarefa $i \in I$ está:
 - um tempo de processamento P_i ;
 - uma janela de entrega $[E_i, T_i]$ dentro da qual a tarefa i deve ser preferencialmente concluída;
 - um custo α_i por unidade de tempo de antecipação;
 - um custo β_i por unidade de tempo de atraso;
- Há antecipação de uma tarefa $i \in I$ quando seu processamento é concluído antes de E_i ;
- Há atraso de uma tarefa $i \in I$ quando seu processamento é concluído depois de T_i ;
- As tarefas que forem concluídas dentro de suas respectivas janelas de entrega não geram penalidades;
- A máquina executa no máximo uma tarefa por vez e uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção;
- Todas as tarefas estão disponíveis para processamento na data 0;

- Entre duas tarefas i e $j \in I$ consecutivas é necessário um tempo S_{ij} de preparação da máquina, chamado tempo de *setup*. Assume-se que o tempo de preparação da máquina para o processamento da primeira tarefa na sequência é igual a 0;
- É permitido tempo ocioso entre a execução de duas tarefas consecutivas.

O objetivo é determinar uma sequência de produção e as datas de início de produção das tarefas de forma a minimizar a soma ponderada das antecipações e atrasos.

É interessante observar que no objetivo do PSUMAA-JE-TPD está embutido dois subproblemas que devem ser resolvidos paralelamente. Um deles é determinar a sequência de produção, isto é, determinar a ordem em que as tarefas serão realizadas. O outro subproblema consiste em programar a data de início de produção de cada tarefa de forma que a soma ponderada das antecipações e atrasos seja a menor possível.

A Figura 2.1 ilustra dois posicionamentos possíveis para as tarefas de uma mesma sequência de produção envolvendo três tarefas. No primeiro posicionamento, as datas de início de processamento das tarefas foram programadas de modo que a tarefa 3 é concluída antecipadamente, acarretando em penalidade por antecipação, e a tarefa 2 dentro de sua janela de entrega, não gerando penalidades. Já no segundo posicionamento, as datas de início de processamento das tarefas foram programadas de forma que a tarefa 2 é concluída com atraso, gerando penalidade, e a tarefa 3 é concluída dentro de sua janela de entrega, não agregando penalidade. Nesta figura, também são ilustrados os tempos de *setup* e de ociosidade de máquina. De acordo com os valores de penalidade incorridos, um destes posicionamentos será o mais adequado para as tarefas desta sequência.

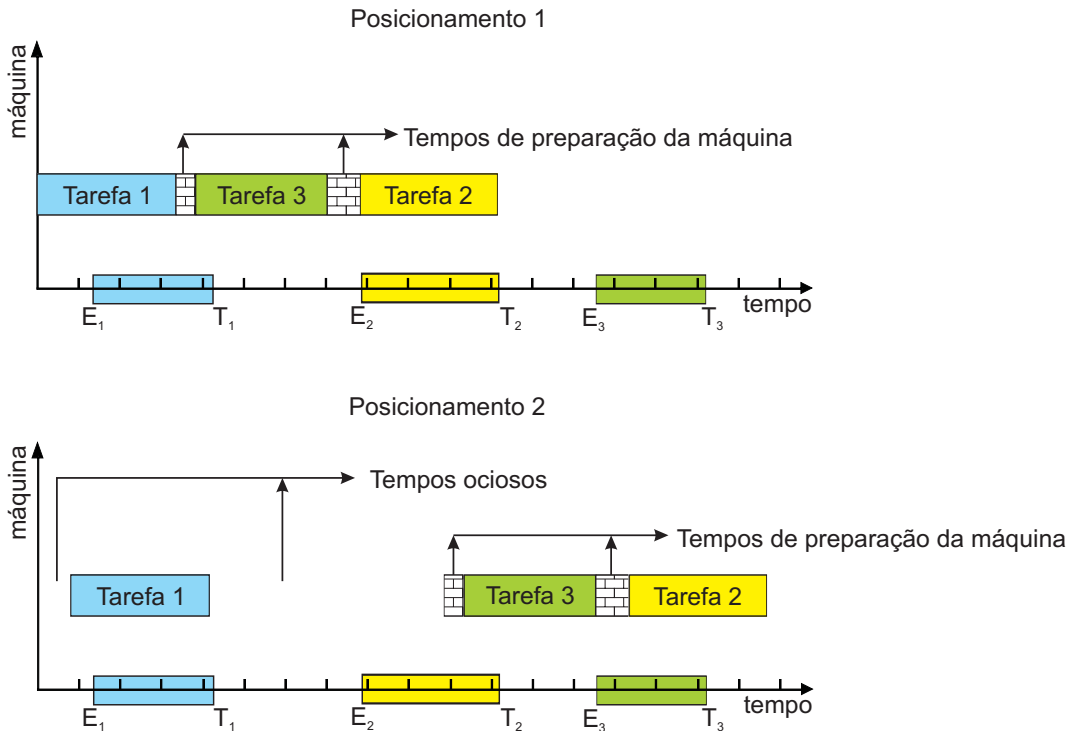


Figura 2.1: Exemplo de dois posicionamentos distintos para as tarefas de uma mesma sequência de produção.

2.2 Métodos de Resolução

O problema de sequenciamento de tarefas é, em seu caso geral, de natureza combinatória. Apesar de possuir domínio finito (para um problema com n tarefas existem $n!$ sequências possíveis) e de simples enumeração, a ideia de testar todas as possíveis sequências é computacionalmente inviável, mesmo para uma quantidade moderada de tarefas. Por exemplo, para o problema com 16 tarefas, um computador que avalia uma sequência em aproximadamente 10^{-6} segundos, gastaria mais de 8 meses para encontrar a melhor sequência.

O PSUMAA com datas de entrega comum é NP-difícil (Baker & Scudder, 1990), ou seja, ainda não é conhecido nenhum algoritmo de complexidade polinomial que o resolva. Como o caso de datas de entrega comum pode ser visto como um caso particular do problema abordado neste projeto, tem-se que o PSUMAA-JE-TPD também é NP-difícil.

Os métodos utilizados para resolver problemas combinatórios, e consequentemente o PSUMAA, podem ser divididos em dois grupos distintos: métodos exatos e métodos heurísticos. Estes métodos são descritos nas seções seguintes.

2.2.1 Métodos Exatos

Os métodos exatos são algoritmos determinísticos, fundamentados em programação matemática e capazes de encontrar a solução ótima de um problema. Uma solução é dita globalmente ótima, ou simplesmente ótima, se, sob um certo critério de otimização, não existir outra solução para o problema com o valor melhor que o dela. Os algoritmos exatos varrem, implicitamente ou não, o conjunto das soluções possíveis para um problema e armazenam a melhor solução encontrada, por isso tem-se a garantia de que a solução retornada é, ou não é, uma solução ótima do problema tratado. Além disso, é possível mensurar o quanto a melhor solução encontrada até um dado instante é inferior à possível solução ótima.

Apesar de serem capazes de encontrar a solução ótima de um problema, devido ao elevado custo computacional, esses métodos são limitados à resolução de problemas de pequeno porte. Como exemplificado anteriormente, para problemas combinatórios, o número de soluções possíveis cresce abruptamente à medida em que se aumenta a dimensão do problema. Deste modo, a simples enumeração das soluções, ainda que implicitamente, se torna inviável para problemas de dimensão mediana.

Os métodos exatos ainda possuem como desvantagens: modelagem mais complexa e nem sempre conseguem encontrar uma solução viável rapidamente. Apesar de suas limitações, os algoritmos exatos são de grande importância na validação de algoritmos não exatos.

A seguir é feita uma breve apresentação de alguns dos métodos exatos já utilizados para resolver o PSUMAA.

Enumeração Total

O método de enumeração total é o mais intuitivo e simples de se implementar entre os algoritmos exatos. Ele consiste em enumerar todas as soluções possíveis e compará-las

entre si. A melhor solução é retornada. Este método é fortemente limitado aos problemas com poucas tarefas.

Branch-and-Bound

O método *Branch-and-Bound* (BB) é um algoritmo de enumeração implícita, ou seja, um método dotado de uma certa “sabedoria”, que retorna a solução ótima de um problema de otimização mesmo enumerando apenas parte do conjunto de soluções possíveis. O BB é baseado no paradigma de divisão e conquista. Tal paradigma consiste em dividir o problema a ser resolvido em partes menores, encontrar soluções para as partes, e então combinar as soluções obtidas em uma solução global (Ziviane, 2004).

De acordo com Gray *et al.* (1997), o método inicia-se considerando o problema original com todas as possíveis soluções, dito problema raiz. Em seguida, são aplicados procedimentos que calculam limitantes superior e inferior para o problema raiz. Se estes limites forem iguais, então uma solução ótima foi encontrada e o procedimento termina. Caso contrário, o problema é dividido em subproblemas menores. Estes subproblemas se tornam filhos do problema raiz. O algoritmo é aplicado recursivamente, gerando assim uma árvore de subproblemas. Quando a solução ótima de um subproblema é encontrada, os demais subproblemas deixam de ser considerados, ou seja, eles são eliminados da árvore de subproblemas. A busca continua até que todos os subproblemas sejam resolvidos ou eliminados, ou até atingir um dado limiar entre a melhor solução encontrada e os limitantes inferiores de todos os subproblemas não resolvidos.

Um limitante superior para um problema de minimização é um valor superior ou igual ao valor ótimo deste problema. O valor da função objetivo de qualquer solução factível, por exemplo, é um limitante superior para o problema. Analogamente, um limitante inferior para este problema é um valor para o qual tem-se a garantia de que a solução ótima é superior ou igual a ele.

Para o cálculo de limites superiores podem ser utilizadas heurísticas, enquanto os limites inferiores são geralmente obtidos por relaxação do problema original. Relaxar um problema consiste em considerar apenas parte de suas restrições. Deste modo, tem-se que a solução do problema original também é uma solução para o problema relaxado. Esta técnica parte do princípio de que o problema relaxado é mais fácil de ser resolvido quando comparado com o problema original. Dentre as técnicas de relaxação existentes, a mais utilizada na resolução do PSUMAA é a relaxação lagrangeana. A técnica de relaxação lagrangeana consiste em adicionar um conjunto de restrições à função objetivo do problema original e resolver o problema relaxado iterativamente. Cada restrição é adicionada com um determinado peso, denominados *multiplicadores de Lagrange*.

Programação Dinâmica

A técnica de programação dinâmica (PD) consiste em dividir o problema em subproblemas. O cálculo das soluções dos subproblemas é feito a partir dos subproblemas menores para os maiores e os resultados são armazenados em uma tabela. Deste modo, uma vez que um subproblema é resolvido, a resposta é armazenada em uma tabela e nunca mais é recalculada. De acordo com Ziviane (2004), a PD é indicada quando a divisão de um problema de tamanho n resulta em n subproblemas de tamanho $n - 1$ facilmente resolvíveis.

2.2.2 Métodos Heurísticos

Como dito anteriormente, o PSUMAA pertence ao conjunto dos problemas de otimização combinatória, para os quais encontrar uma solução ótima de forma exata é uma tarefa difícil. Mesmo com a utilização de estratégias que reduzem o espaço de busca (ver seção 2.2.1), haverá sempre uma quantidade de tarefas a partir da qual a utilização de métodos exatos necessita de um tempo de processamento inviável. Além disso, na maioria das situações práticas, é mais interessante encontrar rapidamente uma “boa solução”, ao invés de perder tempo à procura da solução ótima. Por isso, os métodos heurísticos têm sido muito utilizados na resolução de problemas de natureza combinatória.

Souza (2008) define heurística como sendo uma técnica inspirada em processos intuitivos que procura uma boa solução a um custo computacional aceitável, sem, no entanto, estar capacitada a garantir sua otimalidade, bem como garantir quão próximo está da solução ótima.

Segundo Weise (2008), uma heurística é uma parte de um algoritmo de otimização que usa informações conhecidas para ajudar a decidir qual solução candidata será a próxima a ser testada ou qual a próxima solução a ser produzida. Normalmente, as heurísticas são dependentes das classes de problemas.

Ainda de acordo com Weise (2008), normalmente empregam-se heurísticas em algoritmos determinísticos para definir a ordem de processo das soluções candidatas. Já algoritmos probabilísticos podem considerar somente os elementos do espaço de busca que foram selecionados por uma heurística. Isto se deve ao fato de que, apesar de não garantirem a otimalidade da solução obtida, os métodos heurísticos são de fácil implementação e podem gerar boas soluções rapidamente.

Deste modo, uma heurística pode ser vista como uma técnica que se utiliza de informações particulares dos problemas para encontrar boas soluções a baixo custo computacional.

As heurísticas podem ser divididas em duas classes diferentes: heurísticas construtivas e heurísticas de refinamento. As seções seguintes são destinadas a descrever tais classes.

Heurísticas Construtivas

Heurísticas construtivas (HC) são algoritmos que têm por objetivo a construção de uma solução de forma incremental. A cada passo um elemento é inserido na solução parcial de acordo com uma função de avaliação, que por sua vez depende do problema em questão, até gerar uma solução completa. Nas heurísticas clássicas, os elementos candidatos são geralmente ordenados segundo uma função gulosa, que estima o benefício da inserção de cada elemento, e somente o “melhor” elemento é inserido a cada passo (Souza, 2008). As HC gulosas (heurísticas que utilizam uma função gulosa) são determinísticas e muito utilizadas para gerar soluções iniciais que, possivelmente, serão melhoradas por outros procedimentos heurísticos, ou seja, as soluções gulosas podem ser utilizadas como um limiar superior da solução.

Uma heurística construtiva frequentemente encontrada na literatura para o PSUMAA com datas de entrega distintas é a EDD (*Earliest Due Date* - data de entrega mais cedo). A EDD avalia uma tarefa de acordo com a data de início da sua respectiva janela de entrega. A cada passo, a tarefa com menor data de entrega e ainda não sequenciada é inserida no final da subsequência corrente. O algoritmo é interrompido quando não houver mais tarefa a ser sequenciada.

Outra maneira de se construir uma solução é de forma aleatória, ou seja, em cada passo, o elemento que será inserido na solução parcial é escolhido aleatoriamente dentre os elementos que ainda estão fora dela. Este método é bastante utilizado devido sua facilidade de implementação e por gerar soluções diferentes, isto é, não é determinístico. Porém, testes empíricos mostram que as soluções obtidas geralmente são de baixa qualidade e, por isso, necessitam de um esforço maior na fase de refinamento (Souza, 2008).

Heurísticas de Refinamento

As heurísticas de refinamento são algoritmos que têm por objetivo melhorar uma dada solução inicial por meio de modificações em suas componentes. Uma solução inicial pode ser obtida, por exemplo, através de uma HC. As heurísticas pertencentes a esta classe também são conhecidas como técnicas de busca local (BL) e são baseadas na noção de vizinhança.

Seja S o conjunto de todas as soluções possíveis para um determinado problema de otimização e $f : S \rightarrow \mathbb{R}$ a função objetivo a ser minimizada¹. Uma vizinhança é qualquer função $V : S \rightarrow P(S)$, onde $P(S)$ representa o conjunto das partes de S . Isto é, vizinhança é uma função que leva cada solução $s \in S$ em um subconjunto $V(s) \subseteq S$. Um elemento $s' \in V(s)$ é dito um vizinho de s . Todo vizinho $s' \in V(s)$ é obtido de s por meio de uma operação m , denominada movimento. Esta operação é representada por $s' \leftarrow s \oplus m$.

A Figura 2.2 ilustra a noção de vizinhança. Nela, $V(s) = \{s^1, s^2, s^3, s^4, s^5, s^6\}$ representa a vizinhança da solução $s \in S$. Os movimentos que levam s a cada um dos seus vizinhos são representados pelas setas.

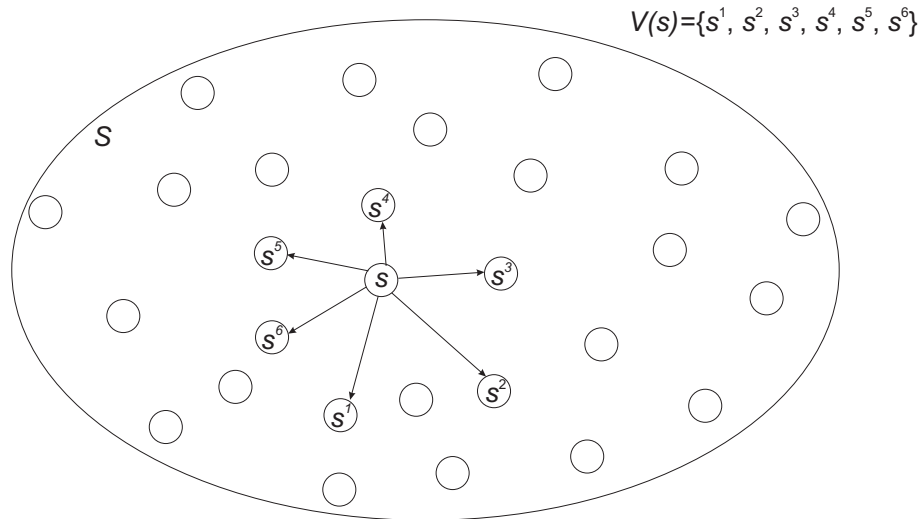


Figura 2.2: Exemplo de vizinhança de uma solução.

O movimento de troca da ordem de processamento de duas tarefas é, por exemplo, um movimento que define uma possível vizinhança para o PSUMAA. Segundo este movimento, a vizinhança de uma solução $s \in S$ é o conjunto das soluções $s' \in S$ obtidas de s por meio da troca da ordem de processamento entre duas tarefas.

¹Caso o problema seja de maximização, ele pode ser facilmente transformado em um problema equivalente de minimização.

Dada uma vizinhança, as técnicas de BL clássicas consistem em caminhar iterativamente de vizinho para vizinho, tentando melhorar a solução construída. Claramente, uma BL é sensível a vizinhança adotada. Deste modo, a escolha de uma vizinhança que permite chegar em qualquer solução do espaço de busca, a partir de qualquer outra solução deste espaço e em um número finito de iterações, é de fundamental importância para o sucesso da técnica.

Outros conceitos importantes surgem com a noção de vizinhança.

Sejam f , V e S como definidos anteriormente. Dizemos que uma solução $s \in S$ é um mínimo local com relação à vizinhança V se $f(s) \leq f(s')$ para todo $s' \in V(s)$. Se uma solução $s^* \in S$ é tal que $f(s^*) \leq f(s')$, para todo $s' \in S$, s^* é dito um mínimo global.

As heurísticas de refinamento clássicas são: método da descida, método de primeira melhora e método de descida randômica.

Dadas uma vizinhança e uma solução inicial, iterativamente, o método da descida analisa todos os vizinhos da solução corrente e o melhor vizinho é selecionado. Se o melhor vizinho representar uma solução melhor que a corrente, ele passa a ser a solução corrente. Caso contrário, o método pára. Portanto a solução retornada é um ótimo local em relação a vizinhança adotada.

O método da descida é descrito no Algoritmo 1, no qual f é a função a ser minimizada, V é uma vizinhança e s uma solução inicial.

procedimento *Descida*($f(\cdot)$, $V(\cdot)$, s)

- 1 $V \leftarrow \{s' \in V(s) \mid f(s') < f(s)\};$
- 2 **enquanto** ($|V| > 0$) **faça**
- 3 Selecione $s' \in V$, onde $s' = \arg \min\{f(s) \mid s \in V\};$
- 4 $s \leftarrow s';$
- 5 $V \leftarrow \{s' \in V(s) \mid f(s') < f(s)\};$
- 6 **fim-enquanto**;
- 7 Retorne s ;

fim *Descida*;

Algoritmo 1: Método da descida.

Como o método da descida analisa toda a vizinhança em cada iteração, ele demanda muito tempo computacional para resolver problemas de maior dimensão. Isto torna necessário técnicas alternativas para estes casos.

O método de primeira melhora (*first improvement method*) é uma variante do método da descida e evita a busca exaustiva pelo melhor vizinho. Neste método, a exploração da vizinhança é interrompida quando um vizinho melhor é encontrado. Quando isto acontece, a solução corrente é atualizada e a exploração da nova vizinhança é inicializada. Deste modo, toda a vizinhança é explorada apenas no pior caso, ou seja, quando somente o último vizinho é de melhora ou todos os vizinhos não são de melhora. O método é interrompido quando um mínimo local é encontrado. É desejável que a ordem de exploração das soluções vizinhas seja alterada a cada passo. Do contrário, privilegia-se apenas um caminho determinístico no espaço de soluções (Souza, 2008).

O Algoritmo 2 representa o método primeiro de melhora. A função a ser minimizada f , a vizinhança V e a solução inicial s são os dados de entrada do método.

```

procedimento Primeira – Melhora( $f(\cdot)$ ,  $V(\cdot)$ ,  $s$ )
1 continua  $\leftarrow$  nao; /* Verifica se a busca na vizinhança corrente pode ser
   interrompida */
2 enquanto (continua = nao) faça
3   continua  $\leftarrow$  sim;
4   para ( $s' \in V(s)$ ) faça
5     se ( $f(s') < f(s)$ ) então
6        $s \leftarrow s'$ ;
7       continua  $\leftarrow$  nao;
8     Sai do para; /* A exploração da vizinhança é interrompida para
       explorar a vizinhança da nova solução corrente */
9   fim-se;
10  fim-para;
11 fim-enquanto;
12 Retorne  $s$ ;
fim Primeira – Melhora;

```

Algoritmo 2: Método primeira melhora.

Uma outra alternativa para se evitar a busca exaustiva de toda a vizinhança é o método de descida randômica. Este método também é uma variante do método da descida. Iterativamente, um vizinho é escolhido de forma aleatória e analisado. Se este vizinho for de melhora, ele passa a ser a solução corrente. Caso contrário, a solução corrente permanece a mesma e passa-se para a próxima iteração. O algoritmo é interrompido após um número pré-determinado de iterações sem melhora na solução corrente.

A solução retornada pelo método randômico de descida não é necessariamente um ótimo local, pois nesta técnica nem todos os vizinhos da solução corrente são explorados.

O refinamento de uma solução s por meio do método randômico de descida, num problema de minimização de uma função f , é representado pelo Algoritmo 3. A vizinhança V e o número máximo de iterações sem melhora *IterMax* também são dados de entrada do método.

2.2.3 Metaheurísticas

Como visto na Seção 2.2.2, as heurísticas de refinamento clássicas são fortemente dependentes da solução inicial e sempre convergem para o primeiro ótimo local em relação à vizinhança adotada. Isto é um grande problema, pois um ótimo local em relação a uma determinada vizinhança pode não ser o ótimo global do problema.

A Figura 2.3 ilustra o comportamento das heurísticas de refinamento para o problema de minimização de uma função f . Observe que os ótimos locais encontrados a partir de s_1 e s_3 não correspondem ao ótimo global, ao contrário do ótimo local relativo à s_2 .

Como o objetivo é encontrar soluções o mais próximo da ótima possível e em tempo reduzido, torna-se necessário a utilização de algoritmos dotados de certa “inteligência”, que sejam capazes de encontrar ótimos locais, mas escapar destes em busca de um ótimo global. Tais algoritmos são comumente chamados de metaheurísticas.

```

procedimento Descida – Randomica( $f(\cdot)$ ,  $V(\cdot)$ ,  $IterMax$ ,  $s$ )
1   $Iter \leftarrow 0$  ;                               /* Contador de iterações sem melhora */
2  enquanto ( $Iter < IterMax$ ) faça
3       $Iter \leftarrow Iter + 1$ ;
4      Selecione aleatoriamente  $s' \in V(s)$ ;
5      se ( $f(s') < f(s)$ ) então
6           $Iter \leftarrow 0$  ;
7           $s \leftarrow s'$  ;
8      fim-se;
9  fim-enquanto;
10 Retorne  $s$ ;
fim Descida – Randomica;

```

Algoritmo 3: Método descida randômica.

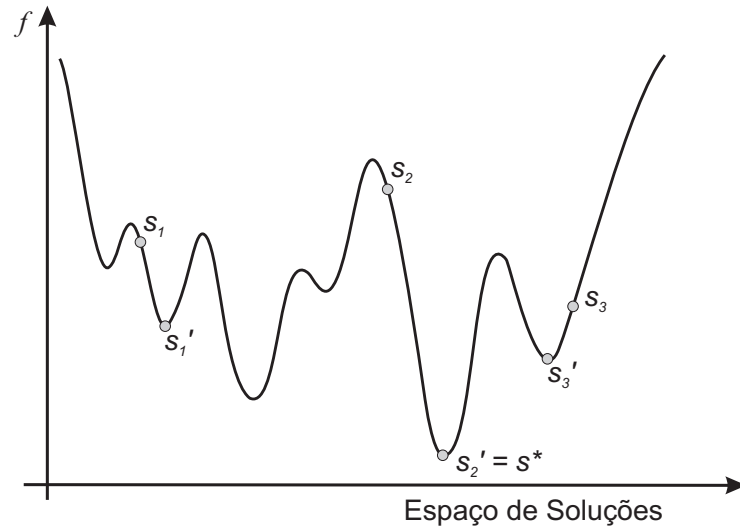


Figura 2.3: Comportamento das heurísticas de refinamento.

De acordo com Weise (2008), uma metaheurística é um método heurístico para resolver uma classe mais geral de problemas. Ela combina funções objetivo ou heurísticas de um modo abstrato e esperançosamente eficiente, normalmente sem utilizar conhecimentos mais profundos do problema, isto é, tratando-o como uma “caixa-preta”.

Para Ribeiro (1996), as metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico.

Portanto, neste trabalho considera-se que uma metaheurística é um algoritmo de caráter geral, ao contrário das heurísticas convencionais, com capacidade de escapar de mínimos locais e encontrar o mínimo global de um problema.

Uma metaheurística se diferencia de outra basicamente pela técnica utilizada para fugir de um ótimo local. A forma com que exploram o espaço de busca divide as metaheurísticas em duas categorias distintas: uma baseada em busca local e a outra baseada em busca populacional.

As metaheurísticas fundamentadas em busca local exploram o espaço de busca ele-

mento a elemento, utilizando a noção de vizinhança, a qual varia de acordo com problema abordado (ver Seção 2.2.2). Já os métodos baseados em busca populacional partem de um conjunto de soluções e aplicam, nestas soluções, operadores que visam a melhoria de todo o conjunto.

A seguir serão apresentadas algumas metaheurísticas clássicas.

Busca em Vizinhança Variável

Proposto por Mladenović & Hansen (1997), a Busca em Vizinhança Variável (*Variable Neighborhood Descent*, VND) é um método de busca local que consiste em explorar o espaço de busca por meio de trocas sistemáticas de vizinhanças.

Conforme esses autores, o VND é baseado nos três princípios básicos apresentados a seguir, sendo que o último deles é de natureza empírica e sugere que um ótimo local fornece informações importantes sobre o ótimo global.

- Um ótimo local em relação à uma dada vizinhança pode não ser um ótimo local relativo à outra vizinhança;
- Um ótimo global é um ótimo local em relação a todas as possíveis vizinhanças;
- Para muitos problemas, ótimos locais com relação a uma ou mais vizinhanças são relativamente próximos.

O método VND clássico recebe como entrada uma solução inicial e um conjunto finito e ordenado de vizinhanças. A cada iteração, uma vizinhança é explorada até que se encontre um ótimo local em relação à ela. Se este ótimo local representar uma melhora na solução corrente, ele é aceito e retorna-se à primeira vizinhança. Caso contrário, passa-se a explorar a vizinhança seguinte. O método pára quando um ótimo local em relação a todas as vizinhanças é encontrado.

O Algoritmo 4 representa o método VND básico. Nele é considerado o problema de minimização de uma função f , dados um conjunto $V = \{V^{(1)}, V^{(2)}, \dots, V^{(r)}\}$, composto de r vizinhanças diferentes, e uma solução inicial s .

Segundo Souza (2008), dependendo do problema abordado, a busca pelo ótimo local (linha 3 do Algoritmo 4) pode ser custosa computacionalmente. Nesta situação, é comum fazer a busca pela primeira solução de melhora (ver Seção 2.2.2). Outra alternativa é considerar a exploração apenas em um certo percentual da vizinhança, sendo este percentual um parâmetro do método. Uma terceira alternativa, bastante utilizada nestas situações, é aplicar o método randômico de descida (também descrito na Seção 2.2.2) em cada vizinhança explorada.

Mais informações sobre o VND podem ser encontradas em Mladenović & Hansen (1997) e Hansen & Mladenović (2003), dentre outros trabalhos.

GRASP

O GRASP (*Greedy Randomized Adaptive Search Procedure* - Procedimento de busca adaptativa gulosa e aleatória) é um método iterativo, proposto por Feo & Resende (1995), constituído basicamente de duas fases: uma fase de construção e outra de busca local. Na fase de construção, uma solução é gerada, elemento a elemento, de forma parcialmente

```

procedimento VND( $f(\cdot)$ ,  $V(\cdot)$ ,  $r$ ,  $s$ )
1  $k \leftarrow 1$ ;                                /* Vizinhaça corrente */
2 enquanto ( $k \leq r$ ) faça
3   Encontre o melhor vizinho  $s' \in V^{(k)}(s)$ ;
4   se ( $f(s') < f(s)$ ) então
5      $s \leftarrow s'$ ;
6      $k \leftarrow 1$ 
7   senão
8      $k \leftarrow k + 1$ 
9   fim-se
10 fim-enquanto;
11 Retorne  $s$ ;
fim VND;

```

Algoritmo 4: Busca em Vizinhaça Variável.

gulosa. Já a fase de busca local consiste em refinar a solução construída na primeira fase. Este procedimento é repetido $GRASP_{max}$ vezes e a melhor solução encontrada é retornada como resultado.

O Algoritmo 5 descreve o procedimento básico do GRASP aplicado ao problema de minimização de uma função f . A função adaptativa gulosa g , a vizinhaça V , $GRASP_{max}$ e γ são os parâmetros do método.

```

procedimento GRASP( $f(\cdot)$ ,  $g(\cdot)$ ,  $V(\cdot)$ ,  $GRASP_{max}$ ,  $\gamma$ )
1  $f^* \leftarrow \infty$ ;                          /* Menor valor de  $f$  obtido */
2 para ( $Iter = 1, 2, \dots, GRASP_{max}$ ) faça
3    $s \leftarrow Construcão - GRASP(g(\cdot), \gamma)$ ;
4    $BuscaLocal(f(\cdot), V(\cdot), s)$ ;
5   se ( $f(s) < f^*$ ) então
6      $s^* \leftarrow s$ ;
7      $f^* \leftarrow f(s)$ ;
8   fim-se
9 fim-para;
10 Retorne  $s^*$ ;
fim GRASP;

```

Algoritmo 5: GRASP.

Como mencionado na Seção 2.2.2, os procedimentos gulosos são determinísticos e, assim, retornam sempre a mesma solução para um determinado problema. Além disso, a solução obtida geralmente converge rapidamente para um ótimo local. Por outro lado, as construções aleatórias geram soluções diversas e, em geral, de baixa qualidade; ou seja, uma construção aleatória gera uma solução diferente em cada execução e uma busca local nestas soluções normalmente demoram para encontrar um ótimo local.

A fase de construção do GRASP visa conciliar as qualidades de uma construção gulosa com as qualidades de uma construção aleatória, isto é, ser capaz de gerar soluções

diferentes e que convergem rapidamente para um ótimo local. A cada iteração desta fase, os elementos que ainda estão fora da solução são ordenados através de uma função que estima o benefício associado a inclusão de cada elemento na solução parcial. Os melhores elementos, segundo esta função, são inseridos em uma lista restrita de candidatos (LRC). Desta lista, aleatoriamente escolhe-se o elemento que será incrementado à solução parcial naquela iteração.

O tamanho da LRC e o nível de gulosidade e de aleatoriedade da fase de construção do GRASP são determinados por um parâmetro $\gamma \in [0, 1]$. É interessante observar que $\gamma = 0$ implica em uma solução completamente gulosa e $\gamma = 1$ implica em uma solução totalmente aleatória.

O Algoritmo 6 descreve a fase de construção de um algoritmo GRASP. A função adaptativa gulosa g é quem estima o benefício da seleção de cada um dos elementos e determina a LRC em cada iteração.

procedimento *Construcao – GRASP*($g(\cdot)$, γ)

- 1 $s \leftarrow \emptyset$;
- 2 Inicialize o conjunto C de candidatos;
- 3 enquanto ($C \neq \emptyset$) faça
- 4 $g(t_{min}) \leftarrow \min\{g(t) \mid t \in C\}$;
- 5 $g(t_{max}) \leftarrow \max\{g(t) \mid t \in C\}$;
- 6 $LRC \leftarrow \{t \in C \mid g(t) \leq g(t_{min}) + \gamma(g(t_{max}) - g(t_{min}))\}$;
- 7 Aleatoriamente, selecione um elemento $t \in LRC$;
- 8 $s \leftarrow s \cup \{t\}$;
- 9 Atualize o conjunto C de candidatos;
- 10 fim-enquanto;
- 11 Retorne s ;

fim *Construcao – GRASP*;

Algoritmo 6: Fase de construção de um algoritmo GRASP.

Na fase de busca local do GRASP pode-se, por exemplo, utilizar uma das técnicas apresentadas na Seção 2.2.2.

Pode-se notar que um ótimo local é encontrado em cada iteração do GRASP. Como os ótimos locais são encontrados a partir de soluções iniciais distintas, eles podem ser diferentes. O melhor ótimo local encontrado é retornado pelo método como a solução para o problema.

Feo & Resende (1995), Resende & Ribeiro (2003) e Souza (2008) apresentam mais informações sobre o GRASP.

Busca Tabu

Proposta por Glover (1986), a Busca Tabu (*Tabu Search*, TS) é uma metaheurística baseada em busca local.

Dada uma vizinhança e uma solução inicial, a TS consiste basicamente em, iterativamente, explorar a vizinhança atual e mudar da solução corrente para o melhor vizinho encontrado, mesmo que o melhor vizinho não represente uma solução de melhora. Isto é feito até que um critério de parada seja satisfeito.

A estratégia de aceitar soluções de piora é utilizada para escapar das armadilhas dos ótimos locais; porém, esta tática pode fazer com que o algoritmo caia em ciclos, isto é, o algoritmo pode voltar a uma solução já gerada anteriormente. Por exemplo, quando a solução corrente é um ótimo local, seu melhor vizinho não será de melhora e será aceito. Na iteração seguinte, o ótimo local será novamente gerado por ser o melhor vizinho da solução corrente e o algoritmo ficará cíclico.

Para evitar ciclagem, o ideal seria armazenar todas as soluções já geradas pelo método em uma lista, denominada lista tabu. No entanto, isto é computacionalmente inviável. Uma alternativa é armazenar apenas as últimas $|LT|$ soluções geradas. Mas uma lista deste tamanho evitaria ciclos de até $|LT|$ iterações, além de poder ser inviável armazenar $|LT|$ soluções e testar se uma solução está ou não na lista tabu.

Na prática, o mais comum é criar uma lista tabu de movimentos proibidos. A lista tabu clássica contém os movimentos reversos aos últimos $|LT|$ movimentos realizados (onde $|LT|$ é um parâmetro do método) e funciona como uma fila FIFO (*First-In-First-Out*).

Apesar de reduzir o risco de ciclagem, uma lista tabu de movimentos pode ser muito restritiva (impede o retorno a uma solução já gerada anteriormente e também que novas soluções sejam geradas). Para tentar corrigir este problema, usa-se um critério de aspiração, que retira, sob certas condições, o *status* tabu de um movimento. Um exemplo de critério de aspiração é aceitar um movimento, mesmo que tabu, se ele melhorar o valor da função objetivo global. Esse é o chamado critério de aspiração por objetivo, sendo o mais usado em implementações Busca Tabu. Outra possibilidade é realizar o movimento tabu mais antigo se todos os possíveis movimentos forem tabus (aspiração por *default*). Uma relação completa de outros critérios de aspiração usualmente utilizados em Busca Tabu é encontrada em Glover & Laguna (1997).

Geralmente, o método é interrompido após $BTmax$ iterações sem melhora na melhor solução encontrada, sendo $BTmax$ um parâmetro do método.

O Algoritmo 7 descreve a metaheurística TS básica aplicada ao problema de minimização de uma função f . A vizinhança V , a função de aspiração A , o conjunto de soluções vizinhas testadas em cada iteração (SVT), o tamanho da lista tabu $|LT|$, o número máximo de iterações sem melhora $BTmax$ e a solução inicial s são as entradas do método.

Mais informações a respeito da TS podem ser encontradas em Glover (1986), Glover (1996), Glover & Laguna (1997), Gendreau (2003) e Souza (2008), entre outros trabalhos.

Simulated Annealing

O *Simulated Annealing* (SA) é uma técnica de busca local probabilística proposta por Kirkpatrick *et al.* (1983) e fundamentada na termodinâmica, ao simular o processo de recozimento (nome dado à operação de resfriamento de um conjunto de átomos aquecidos).

Para se obter metais mais estáveis, estruturalmente fortes e de menor energia interna, eles são aquecidos a uma temperatura suficientemente alta e depois são lentamente resfriados. Durante o recozimento, o material passa por vários estados possíveis.

Como o SA simula o processo de recozimento dos metais, são feitas as seguintes correspondências com um problema de otimização. As soluções do espaço de busca correspondem aos possíveis estados de um metal. O valor da função objetivo corresponde à energia interna em cada estado. O valor de um ótimo local (possivelmente global) corresponde à energia mínima.

```

procedimento Busca – Tabu( $f(\cdot)$ ,  $V(\cdot)$ ,  $A(\cdot)$ ,  $SVT$ ,  $|LT|$ ,  $BTmax$ ,  $s$ )
1  $s^* \leftarrow s$ ;                                /* Melhor solução obtida */
2  $Iter \leftarrow 0$ ;                            /* Contador do número de iterações sem melhora de  $s^*$  */
3  $LT \leftarrow \emptyset$ ;                        /* Lista tabu */
4 Inicialize a função de aspiração  $A$ ;
5 enquanto ( $Iter \leq BTmax$ ) faça
6    $Iter \leftarrow Iter + 1$ ;
7   Seja  $s' \leftarrow s \oplus m$  o melhor elemento de  $SVT \subset V(s)$  tal que o movimento  $m$  não
   é tabu ( $m \notin LT$ ) ou a condição de aspiração de  $s'$  ( $A(s')$ ) é atendida;
8   Atualize a  $LT$ ;
9    $s \leftarrow s'$ ;
10  se ( $f(s) < f(s^*)$ ) então
11     $s^* \leftarrow s$ ;
12     $Iter \leftarrow 0$ ;
13  fim-se
14  Atualize  $A$ ;
15 fim-enquanto;
16 Retorne  $s^*$ ;
fim Busca – Tabu;

```

Algoritmo 7: Busca Tabu.

O método parte de uma solução inicial qualquer e cada iteração consta basicamente de um *loop*. Em cada iteração deste *loop* interno é gerado, aleatoriamente, um único vizinho s' da solução corrente s . Seja $\Delta = f(s') - f(s)$ e considere que o problema seja de minimização. Se $\Delta < 0$ então a solução vizinha passa a ser a solução corrente. Caso contrário ($\Delta \geq 0$), o vizinho pode ser aceito com uma probabilidade $e^{-\frac{\Delta}{kT}}$, onde k é a constante de Boltzmann e T representa a temperatura corrente.

Inicialmente, a temperatura T assume um valor suficientemente alto T_0 . A cada iteração i do método, a temperatura é diminuída, ou seja, $T_i \leftarrow \alpha \cdot T_{i-1}$, onde $0 < \alpha < 1$. Os valores de T_0 e α são parâmetros do método.

Observe que quanto maior a temperatura, maior a probabilidade de se aceitar uma solução de piora. Portanto, a cada iteração do algoritmo a chance de uma solução de piora ser aceita é reduzida. As soluções de piora são aceitas para que se escape das armadilhas dos ótimos locais.

O método é interrompido quando a temperatura atinge um valor próximo de zero e somente vizinhos de melhora são aceitos, evidenciando o encontro de um ótimo local.

O Algoritmo 8 descreve o método SA básico para o problema de minimização de uma função f . A estrutura de vizinhança V , a razão de resfriamento α , o número de iterações em cada temperatura (SAm_{ax}), a temperatura inicial T_0 , um número real próximo de zero (ϵ) e uma solução inicial s são as entradas do método.

Mais detalhes do método podem ser encontrados em Kirkpatrick *et al.* (1983), Ribeiro (1996), Henderson *et al.* (2003) e Souza (2008), entre outros trabalhos.

```

procedimento  $SA(f(\cdot), V(\cdot), \alpha, SAmax, T_0, \epsilon, s)$ 
1  $s^* \leftarrow s$ ;                                /* Melhor solução obtida */
2  $IterT \leftarrow 0$ ;                            /* Número de iterações na temperatura  $T$  */
3  $T \leftarrow T_0$ ;                             /* Temperatura corrente */
4 enquanto ( $T > \epsilon$ ) faça
5   enquanto ( $IterT < SAmax$ ) faça
6      $IterT \leftarrow IterT + 1$ ;
7     Aleatoriamente, escolha  $s' \in V(s)$ ;
8      $\Delta \leftarrow f(s') - f(s)$ ;
9     se ( $\Delta < 0$ ) então
10       $s \leftarrow s'$ ;
11      se ( $f(s') < f(s^*)$ ) então  $s^* \leftarrow s'$ ;
12    senão
13      Aleatoriamente, escolha  $x \in [0, 1]$ ;
14      se ( $x < e^{-\frac{\Delta}{kT}}$ ) então  $s \leftarrow s'$ ;
15    fim-se
16  fim-enquanto;
17   $T \leftarrow \alpha \cdot T$ ;
18   $IterT \leftarrow 0$ 
19 fim-enquanto;
20 Retorne  $s^*$ ;
fim  $SA$ ;

```

Algoritmo 8: *Simulated Annealing.**Iterated Local Search*

O *Iterated Local Search* (ILS) é um método de busca local que faz uso de perturbações para escapar das armadilhas dos ótimos locais e, possivelmente, encontrar um ótimo global. Tais perturbações devem ser fortes o suficiente para permitir a fuga do ótimo local corrente e que a busca local explore diferentes regiões do espaço de busca. Ao mesmo tempo, elas devem ser suficientemente fracas para preservar características do ótimo local corrente e evitar reinícios aleatórios.

De acordo com Souza (2008), é necessário especificar quatro componentes para a aplicação do ILS:

1. Procedimento *GeraSolucaoInicial*, que gera uma solução inicial. Pode ser utilizado, por exemplo, uma HC (ver Seção 2.2.2);
2. Procedimento *BuscaLocal*, que retorna uma solução possivelmente melhorada. Por exemplo, pode ser utilizada uma heurística de refinamento (ver seção 2.2.2);
3. Procedimento *Perturbacao*, que modifica a solução corrente, guiando ela a uma solução intermediária;
4. Procedimento *CriterioAceitacao*, que decide em qual solução a próxima perturbação será aplicada.

Um critério de aceitação muito utilizado consiste em aceitar um ótimo local s'' somente se ele for melhor que o ótimo local corrente s . O Algoritmo 9 ilustra a aplicação do ILS básico a um problema de minimização de uma função f .

```

procedimento  $ILS(f(.))$ 
1  $s_0 \leftarrow GeraSolucaoInicial();$ 
2  $s \leftarrow BuscaLocal(s_0);$ 
3  $s^* \leftarrow s;$                                 /* Melhor solução obtida */
4 enquanto (critério de parada não é satisfeito) faça
5    $s' \leftarrow Perturbacao(s);$ 
6    $s'' \leftarrow BuscaLocal(s');$ 
7   se ( $f(s'') < f(s^*)$ ) então
8      $s^* \leftarrow s'';$ 
9      $s \leftarrow CriterioAceitacao(s, s'');$ 
10  fim-se
11 fim-enquanto;
12 Retorne  $s^*$ ;
fim  $ILS;$ 

```

Algoritmo 9: *Iterated Local Search.*

Geralmente, utiliza-se um número máximo de iterações sem melhora na melhor solução encontrada como critério de parada do algoritmo.

Mais detalhes do método podem ser encontrados em Souza (2008) e Lourenço *et al.* (2003).

Algoritmos Evolutivos

Algoritmos evolutivos (AE's) são metaheurísticas baseadas em busca populacional e que utilizam mecanismos inspirados na biologia, como mutação, *crossover*, seleção natural e condições de sobrevivência, para refinar, iterativamente, um conjunto de soluções candidatas (Weise, 2008).

Essencialmente, um AE parte de um conjunto de soluções candidatas, denominado população inicial, e, em cada iteração, todos os elementos (indivíduos) da população atual são avaliados pela função de avaliação. Caso o critério de parada seja satisfeito, o método é interrompido e os melhores elementos são listados. Caso contrário, com o auxílio de uma função de aptidão, que pode não depender somente da função de avaliação, os indivíduos são novamente avaliados. A função de aptidão tem o objetivo de auxiliar na seleção dos indivíduos da população que darão origem a novos indivíduos. A geração de novos indivíduos pode ser feita por meio de *crossover* ou mutação. No *crossover*, dois indivíduos pais dão origem a filhos que possuem características de ambos os pais. A mutação consiste em alterar uma característica do indivíduo. Terminada a etapa de geração de novos indivíduos, também com o auxílio da função de aptidão, é determinada a população sobrevivente, isto é, os indivíduos que farão parte da população da próxima iteração.

A Figura 2.4 ilustra a estrutura básica de um AE.

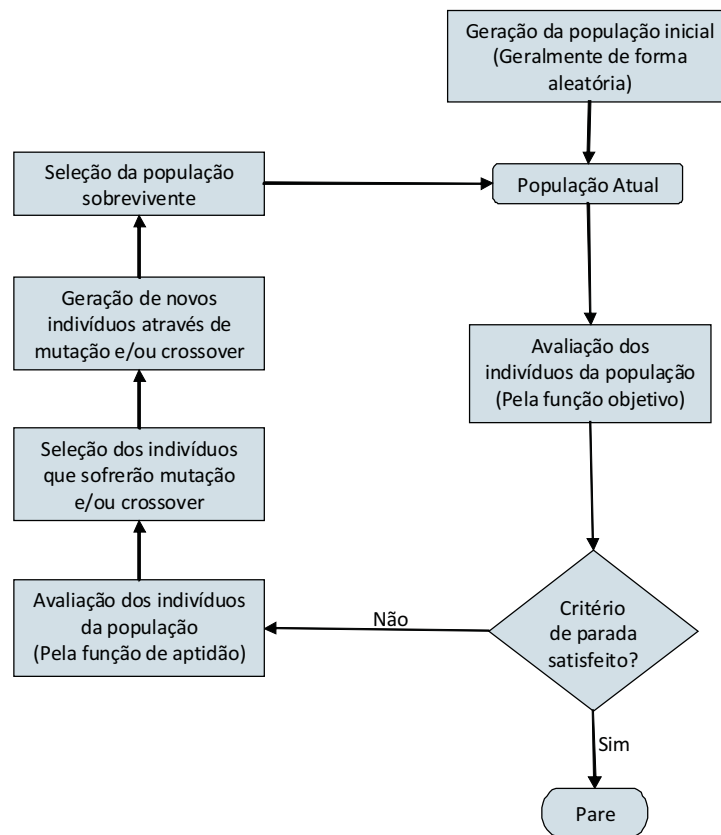


Figura 2.4: Estrutura básica de um Algoritmo Evolutivo.

De acordo com Souza (2008), os critérios de parada mais comuns são: quando um certo número de gerações é atingido, quando não há melhora na melhor solução encontrada após um certo número de iterações ou quando o desvio padrão da população é inferior a um certo valor (situação que evidencia uma homogeneidade da população).

Mais informações sobre os AE's podem ser encontradas, por exemplo, em Weise (2008).

Algoritmos Genéticos

Segundo Weise (2008), os Algoritmos Genéticos (AG's) compõem uma subclasse dos AE's, na qual os elementos do espaço de busca são vetores de componentes binários ou arranjos de tipos elementares.

É muito comum encontrar na literatura trabalhos que, por conveniência, consideram os espaços de soluções dos AG's formados por vetores de números inteiros. Neste caso, a definição de AG coincide com a definição de AE.

De qualquer forma, pode-se dizer que a Figura 2.4 ilustra a estrutura de um Algoritmo Genético básico.

Reeves (2003), Weise (2008) e Souza (2008) apresentam mais detalhes a respeito dos AG's.

Scatter Search

O *Scatter Search* (SS) é um método baseado em busca populacional que apresenta similaridades com os Algoritmos Genéticos; difere no tamanho da população e no uso de estratégias determinísticas, ao invés de probabilísticas, para atingir os objetivos de diversificação e intensificação (Belfiore *et al.*, 2006).

Ainda conforme Belfiore *et al.* (2006), o SS opera sobre um conjunto de soluções, chamado conjunto de referência, combinando estas soluções para criar novas soluções de modo a melhorar a soluções originais. É, portanto, um método evolutivo. O tamanho do conjunto de referência em *scatter search* é pequeno comparado à população nos Algoritmos Genéticos.

Segundo (Glover & Laguna, 2003), o processo do SS é organizado para:

1. Capturar informações não contidas separadamente nas soluções originais;
2. Tirar proveito de métodos de soluções heurísticas auxiliares (para avaliar as combinações produzidas e gerar ativamente novas soluções);
3. Utilizar estratégias específicas no lugar de métodos aleatórios de busca.

Uma abordagem mais detalhada do método é apresentada em Glover & Laguna (2003) e Belfiore *et al.* (2006).

2.2.4 Estratégias de Intensificação

As técnicas de intensificação são métodos que exploram com maior intensidade as soluções próximas de uma dada solução. A seguir serão apresentadas duas destas técnicas.

Princípio da Otimalidade Próxima

O Princípio da Otimalidade Próxima (POP) é uma técnica que refina uma parte da solução com o objetivo de melhorar toda a solução. Este método é baseado na ideia de que boas soluções em um nível estão próximas de boas soluções em um nível adjacente (Glover & Laguna, 1997).

Na prática, o POP é muito utilizado em métodos construtivos. Periodicamente, antes de se inserir um novo elemento na solução em construção, esta é submetida à uma busca local.

Reconexão por Caminhos

A Reconexão por Caminhos (*Path Relinking*, PR) é uma técnica de intensificação proposta em Glover (1996). O método parte de duas soluções dadas, sendo uma chamada de solução base e a outra de solução guia. Iterativamente, um atributo da solução guia é adicionado na solução base. A solução base sofre, então, uma busca local em que se fixam todos os atributos da solução guia que já foram incrementados à solução base. O método pára quando a solução base possui todos os atributos da solução guia.

O objetivo do método é, portanto, encontrar, durante as iterações, uma solução que possua os atributos bons contidos nas duas soluções dadas, isto é, uma solução melhor que as duas soluções iniciais.

O PR pode ser utilizado como um método de pós-otimização ou então para intensificar a busca em torno de ótimos locais. No primeiro caso, as melhores soluções distintas encontradas por algum procedimento heurístico são armazenadas em um conjunto e posteriormente aplica-se a PR em pares de soluções deste conjunto. Tal conjunto é chamado de conjunto elite e seu tamanho é um parâmetro do método. No segundo caso, sempre que um ótimo local é encontrado por um determinado procedimento, aplica-se a PR no par de soluções formado pelo ótimo local e a melhor solução até então encontrada pelo algoritmo.

Mais detalhes desta técnica podem ser encontrados em Glover (1996), Glover & Laguna (2003) e Souza (2008).

2.3 Trabalhos relacionados

Muitos trabalhos são encontrados na literatura com o objetivo de resolver o PSUMAA em suas diversas variantes. Uma breve descrição dos trabalhos mais relevantes são apresentados a seguir. Por questões didáticas, os trabalhos foram separados de acordo com as datas de entrega. Primeiramente são apresentados os trabalhos que tratam do PSUMAA com datas de entrega comum, em seguida os com datas de entrega distintas e, por último, os com janelas de entrega distintas.

2.3.1 PSUMAA com Datas de Entrega Comuns

Esta seção é destinada à apresentação de alguns trabalhos que tratam do PSUMAA com datas de entrega comuns.

Dentre as versões do PSUMAA, a mais estudada é a com datas de entrega comuns e tempos de *setup* independentes da sequência de produção. Para este caso, existem propriedades que podem ser utilizadas nos algoritmos de resolução, de modo a reduzir o espaço de busca e conseqüentemente o tempo computacional. As principais propriedades que são utilizadas por todos os autores são:

Propriedade 1 *Em uma sequência ótima não há tempo ocioso entre duas tarefas consecutivas.*

Propriedade 2 *Em uma sequência ótima a conclusão de uma tarefa coincide com a data de entrega ou o processamento da primeira tarefa inicia no instante zero.*

Propriedade 3 *Em uma sequência ótima, as tarefas antecipadas estão ordenadas de forma não-crescente pela razão entre o tempo de processamento e a penalidade por unidade de tempo de antecipação, enquanto as tarefas atrasadas estão ordenadas de forma não-decrescente pela razão entre o tempo de processamento e a penalidade por unidade de tempo de atraso.*

Se uma sequência satisfaz a Propriedade 3, ela é dita *V-shaped* (Biskup & Feldmann, 2001).

As justificativas para estas propriedades podem ser encontradas em Szwarc (1996). Neste trabalho, também foram estudadas propriedades da otimalidade no caso em que as

penalidades são dependentes das tarefas e as razões entre os tempos de processamento e as penalidades satisfazem certas condições. As propriedades estudadas foram utilizadas em três algoritmos baseados em programação dinâmica e de complexidade pseudo-polinomial, sendo que estes algoritmos refletiam o estado da arte até então. Experimentos em problemas-teste com até 100 tarefas evidenciaram a melhoria proporcionada pela utilização das propriedades apresentadas.

Lee & Kim (1995) resolveram o problema por meio de um algoritmo genético paralelo com representação binária, sendo este uma variante do AG convencional. Uma condição necessária para que um arranjo em série de genes seja *V-shaped* dentro de uma sequência é apresentada. Foram resolvidos problemas-teste com até 100 tarefas.

Alidaee & Dragan (1997) apresentaram um algoritmo de complexidade $O(n \log n)$ que resolve de forma ótima os casos em que as penalidades por antecipação e atraso são iguais para a mesma tarefa e proporcionais aos tempos de processamento das tarefas.

Biskup & Feldmann (2001) propuseram um gerador de problemas-teste. Foram utilizadas duas novas heurísticas para resolver e obter limites superiores para os problemas-teste, com até 1000 tarefas, gerados. As heurísticas se mostraram eficientes, pois encontraram resultados próximos dos ótimos conhecidos de problemas de menor porte. O gerador, junto com os limites superiores, são disponibilizados pelos autores para servirem como referências de trabalhos futuros sobre o PSUMAA.

Feldmann & Biskup (2003) apresentam uma nova e apropriada representação para o problema, baseada nas propriedades deste caso do PSUMAA. Foram utilizadas três metaheurísticas para resolver o problema: um Algoritmo Evolutivo, um *Simulated Annealing* e um *Threshold Accepting*, sendo este último uma variante do SA. Estas metaheurísticas foram aplicadas em problemas-teste propostos por Biskup & Feldmann (2001).

Uma construção gulosa e heurísticas baseadas em Busca Tabu e Algoritmos Genéticos foram propostas por Hino *et al.* (2005). Estratégias híbridas também foram analisadas com o objetivo de melhorar o desempenho destes métodos. Foram resolvidos problemas-teste com até 1000 tarefas.

Belfiore *et al.* (2006) aplicaram a metaheurística *Scatter Search* neste variante do PSUMAA. O algoritmo proposto foi utilizado para resolver todos os problemas-teste propostos por Biskup & Feldmann (2001).

Ying (2008) resolveu o problema por meio de um procedimento *Branch-and-Bound* que faz uso de um algoritmo *Recovering Beam Search* para caminhar na árvore de busca. O autor também utilizou a base de dados proposta por Biskup & Feldmann (2001) para realizar experimentos computacionais.

O PSUMAA com datas de entrega comum pode ser classificado em restrito ou irrestrito. Quando a soma dos tempos de processamento de todas as tarefas é maior que a data de entrega, diz-se que o problema é restrito. Caso contrário, o problema é dito irrestrito.

James (1997) testou diferentes formas de Busca Tabu para resolver o caso restrito do problema, inclusive uma que aplica as propriedades numa nova técnica que reduz o tamanho do espaço de busca de $n!$ para 2^n . Foram resolvidos problemas-teste com até 250 tarefas e a TS que faz uso da nova técnica mostrou-se mais eficiente que as demais.

Mondal & Sen (2001) consideraram que as penalidades por antecipação e atraso são independentes das tarefas. Foi apresentado um algoritmo com espaço de busca em grafo para encontrar a solução ótima de problemas-teste maiores do caso restrito. Ao contrário da programação dinâmica, o algoritmo encontrou soluções ótimas mesmo quando a me-

mória era limitada. O algoritmo se mostrou mais rápido que a programação dinâmica e *depth-first-branch-bound* (*depth-first* é uma heurística utilizada para caminhar na árvore de busca do método BB) e consegue resolver muitos problemas-teste maiores em tempo razoável. Também foram propostos e utilizados novos limites inferiores e superiores para o problema. Os problemas-teste resolvidos são com até 2000 tarefas.

Rabadi *et al.* (2004) consideraram o PSUMAA com datas de entrega comuns, penalidades iguais (o objetivo é minimizar a soma simples das antecipações e dos atrasos) e tempos de *setup* dependentes da sequência de produção. Foi proposto um algoritmo *Branch-and-Bound* para obter soluções ótimas do caso irrestrito. Os autores resolveram problemas-teste com até 25 tarefas em tempo computacional aceitável, o que representou um avanço para a época, já que os algoritmos exatos de então resolviam, na otimalidade, apenas problemas desta classe com até 8 tarefas.

2.3.2 PSUMAA com Datas de Entrega Distintas

Esta seção é destinada à apresentação de alguns trabalhos que tratam do PSUMAA com datas de entrega distintas.

M'Hallah (2007) considerou o problema com as penalidades iguais (o objetivo é minimizar a soma simples das antecipações e dos atrasos) e os tempos de *setup* independentes da sequência de produção, não sendo permitida a existência de tempo ocioso entre a execução de tarefas consecutivas. Foi proposto um método heurístico híbrido que combina heurísticas de busca local (regras de despacho, métodos de descida e SA) e um algoritmo evolutivo baseado em AG. Os efeitos de cada heurística no método são mostrados via experimentos com problemas-teste com até 4000 tarefas.

Chang (1999) tratou do problema com penalidades iguais e tempos de *setup* independentes da sequência de produção, sendo permitida a ociosidade de máquina. O autor utilizou um algoritmo *branch-and-bound* para resolver o problema. Um esquema para calcular diferentes limites inferiores baseados no procedimento de eliminação de sobreposição de uma sequência *Just-In-Time* também foi apresentado. Propriedades e teoremas deste procedimento de eliminação foram demonstradas. O algoritmo foi utilizado para resolver problemas-teste com até 45 tarefas. Ventura & Radhakrishnan (2003) abordaram o mesmo problema, para o qual também propuseram um algoritmo BB. Neste trabalho foram derivadas propriedades de uma sequência ótima. Um procedimento de relaxação lagrangeana que utiliza o algoritmo de subgradiente foi implementado para atingir soluções próximas da otimidade e ajustar os limites inferiores. O algoritmo foi testado em problemas-teste com até 100 tarefas e apresentou uma divergência relativa menor que 3% das soluções ótimas.

Li (1997) estudou o caso com tempos de *setup* independentes da sequência de produção, não sendo permitida a existência de tempos ociosos entre a execução de tarefas consecutivas. Neste trabalho, o problema foi decomposto em dois subproblemas com estruturas mais simples. O limite inferior do problema foi, então, dado pela soma dos limites inferiores dos dois subproblemas. O limite inferior de cada subproblema foi obtido por relaxação lagrangeana. Também foram desenvolvidos dois procedimentos de ajuste multiplicador, de complexidade $O(n \log n)$, para resolver os duais dos subproblemas. Um algoritmo *branch-and-bound* baseado nestes procedimentos foi apresentado e utilizado para resolver problemas-teste com até 50 tarefas, dobrando a dimensão dos problemas resol-

vidos, na otimidade, por algoritmos exatos até aquela data. O propôs, também, uma heurística baseada em busca local e resolveu problemas com até 3000 tarefas. Almeida & Centeno (1998) aplicaram um método heurístico composto, que combina TS, SA e técnicas de descida, em problemas-teste com até 50 tarefas. Liaw (1999), Valente & Alves (2005b) e Valente & Alves (2005a) também trataram este mesmo problema. O primeiro autor utilizou um algoritmo BB baseado em procedimentos que determinam limites inferiores e superiores eficientes. Foram derivadas regras de dominância para ajudar a eliminar nós não promissores da árvore de busca. O desempenho do algoritmo foi testado em problemas-teste com até 50 tarefas. No segundo trabalho foram introduzidas duas novas heurísticas, sendo uma regra de despacho e um procedimento guloso. Também foram utilizadas regras de dominância, visto que elas melhoram os resultados das heurísticas e demandam pouco tempo adicional. Os experimentos computacionais foram realizados em problemas-teste com até 500 tarefas. O terceiro trabalho propõe dois algoritmos BB. Cada BB proposto se utiliza de uma variante do algoritmo *beam-search* para caminhar na árvore de busca, sendo que um deles se utiliza de um *Filtered Beam Search* e o outro de um *Recovering Beam Search*. Os dois métodos foram testados em problemas-teste com até 500 tarefas.

Lee & Choi (1995) trataram do caso com tempos de *setup* independentes da sequência de produção, sendo permitida a ociosidade de máquina. Foi apresentado um algoritmo de complexidade polinomial que determina a data ótima de início de uma tarefa em uma dada sequência de produção. Também aplicou-se ao problema um algoritmo genético que faz uso deste algoritmo de data ótima. Foram resolvidos problemas-teste com até 80 tarefas. Mazzini & Armentano (2001) estudaram este mesmo problema. Neste trabalho, uma heurística construtiva que determina a sequência de produção e simultaneamente insere tempos ociosos foi proposta e utilizada para resolver problemas-teste com até 80 tarefas.

No trabalho de Bustamante (2007) foi estudado o problema com tempos de *setup* dependentes da sequência de produção, sendo permitida a existência de tempos ociosos de máquina. São desenvolvidos dois modelos de programação linear inteira mista, sendo que tais modelos exigem que os tempos de *setup* satisfaçam a desigualdade triangular. O autor também sugeriu alterações nos modelos que permitem resolver o caso com janelas de entrega distintas, mas os testes computacionais foram realizados apenas em problemas com datas de entrega distintas. Foram resolvidos, na otimalidade, problemas com até 10 tarefas por meio do software de otimização GLPK 4.8.

2.3.3 PSUMAA com Janelas de Entrega Distintas

Alguns trabalhos da literatura que tratam o PSUMAA com janelas de entrega distintas são apresentados a seguir.

Wan & Yen (2002) trataram o problema com tempos de *setup* independentes da sequência de produção, permitindo a existência de tempos ociosos de máquina. Primeiramente, foi apresentada uma formulação matemática do problema junto com várias propriedades importantes para sua resolução. Em seguida, foi proposto um procedimento de complexidade polinomial para determinar a data de conclusão ótima de processamento de cada tarefa em uma dada sequência, sendo este procedimento uma extensão do algoritmo proposto por Lee & Choi (1995). Por fim, uma TS, que faz uso do procedimento de datas

ótimas, foi proposta para resolver o problema. Foram realizados testes computacionais em problemas envolvendo até 80 tarefas.

Koulamas (1996) focou no problema com janelas de entrega distintas, porém com uma sensível distinção dos demais trabalhos da literatura em relação à antecipação. Em seu trabalho, há antecipação de uma tarefa quando seu processamento é iniciado antes do início de sua janela de entrega. O restante da literatura considera que há antecipação quando uma tarefa é concluída antes do início de tal janela. O autor também considerou tempos de *setup* independentes da sequência de produção, sendo permitida a ociosidade de máquina. O autor adaptou heurísticas já utilizadas em outros casos do PSUMAA e as aplicou nessa versão. Também foi proposto um algoritmo que insere tempos ociosos, de modo ótimo, em uma dada sequência de produção. Foram realizados experimentos computacionais em problemas-teste com até 200 tarefas.

Gomes Júnior *et al.* (2007), Penna (2009) e Ribeiro (2009) correspondem ao estado da arte em relação ao PSUMAA-JE-TPD. No primeiro trabalho foi proposto um modelo de programação matemática para representar o problema, no qual deixa de ser necessário que o problema satisfaça a desigualdade triangular. Também foi proposto um método heurístico baseado em GRASP, ILS e VND. Para cada sequência de tarefas gerada pela heurística proposta, é utilizado um algoritmo de complexidade polinomial que determina a data ótima de início de processamento das tarefas na sequência dada, sendo este algoritmo uma extensão daquele proposto por Wan & Yen (2002). Foram realizados experimentos computacionais em problemas-teste com os tempos de *setup* simétricos e com até 75 tarefas, utilizando a heurística, e até 12 tarefas, utilizando o modelo matemático. O segundo trabalho propõe uma heurística baseada em GRASP, VND, TS e PR, enquanto no terceiro trabalho é proposto um Algoritmo Genético Adaptativo para resolver o problema. Nos dois últimos trabalhos foram realizados experimentos nos mesmos problemas-teste utilizados por Gomes Júnior *et al.* (2007), sendo mostrada a superioridade dos algoritmos propostos em relação a este último. Ribeiro (2009) também realizou testes em outras duas bases de dados em que os tempos de *setup* não são necessariamente simétricos. Foram resolvidos problemas-teste com até 100 tarefas por meio do AG proposto.

Capítulo 3

Formulações de Programação Matemática

Como foi mencionado na Seção 2.2.1, os métodos exatos são de grande importância por serem capazes de encontrar a solução ótima do problema e auxiliarem na validação dos procedimentos heurísticos. Com esta finalidade, nesta seção são apresentadas três formulações de programação matemática para representar o PSUMAA-JE-TPD. A primeira delas foi proposta por Gomes Júnior (2007), enquanto as outras duas são propostas neste trabalho. A segunda formulação é um aperfeiçoamento da primeira e a última é indexada no tempo.

3.1 Modelo de Gomes Júnior *et al.* (2007) - MPLIM-G

A seguir é reproduzido o modelo de programação linear inteira mista (PLIM) proposto por Gomes Júnior (2007) para representar o PSUMAA-JE-TPD, doravante denotado por MPLIM-G.

Para auxiliar na modelagem, são utilizadas duas tarefas fictícias, 0 (zero) e $n + 1$, que devem ser sequenciadas necessariamente na primeira e na última posição, respectivamente. Admite-se que P_0 e P_{n+1} são iguais a zero e que $S_{0i} = S_{i0} = 0$ e $S_{i,n+1} = S_{n+1,i} = 0$, $\forall i \in I$. A data de início do processamento da tarefa $j \in I \cup \{0, n + 1\}$ e os tempos de antecipação e atraso da tarefa $i \in I$ são representados por s_j , e_i e t_i , respectivamente.

Sejam y_{ij} variáveis que determinam a sequência de produção, sendo $y_{ij} = 1$ se a tarefa j for sequenciada imediatamente após a tarefa i e $y_{ij} = 0$, caso contrário, $\forall i, j \in I \cup \{0, n + 1\}$. Considere, ainda, uma constante M de valor suficientemente grande. O

modelo MPLIM-G é representado pelas equações (3.1)-(3.10):

$$\min \quad Z = \sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (3.1)$$

$$\text{s.a. } s_i + P_i + y_{ij}(M + S_{ij}) - M \leq s_j \quad \forall i \in I \cup \{0\}, \\ \forall j \in I \cup \{n+1\} \text{ e } i \neq j \quad (3.2)$$

$$\sum_{j=1, j \neq i}^{n+1} y_{ij} = 1 \quad \forall i \in I \cup \{0\} \quad (3.3)$$

$$\sum_{i=0, i \neq j}^n y_{ij} = 1 \quad \forall j \in I \cup \{n+1\} \quad (3.4)$$

$$s_i + P_i + e_i \geq E_i \quad \forall i \in I \quad (3.5)$$

$$s_i + P_i - t_i \leq T_i \quad \forall i \in I \quad (3.6)$$

$$s_i \geq 0 \quad \forall i \in I \cup \{0, n+1\} \quad (3.7)$$

$$e_i \geq 0 \quad \forall i \in I \quad (3.8)$$

$$t_i \geq 0 \quad \forall i \in I \quad (3.9)$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j \in I \cup \{0, n+1\} \quad (3.10)$$

A função objetivo, representada pela equação (3.1), busca a minimização da soma ponderada das antecipações e atrasos. As restrições (3.2) garantem que existe tempo suficiente para executar uma tarefa i e preparar a máquina antes do início do processamento da tarefa seguinte j , ou seja, elas garantem que o processamento da tarefa j não será iniciado antes do término do processamento da tarefa i . As restrições (3.3) e (3.4) garantem que cada tarefa terá apenas uma tarefa imediatamente sucessora e uma tarefa imediatamente antecessora, respectivamente. As restrições (3.5) e (3.6) definem as antecipações e os atrasos de acordo com as respectivas janelas de entrega de cada tarefa. As restrições (3.7), (3.8), (3.9) e (3.10) dizem respeito ao tipo de variáveis.

3.2 Primeiro Modelo Proposto - MPLIM-BG

O modelo de programação linear inteira mista (PLIM) apresentado a seguir é baseado naquele proposto por Gomes Júnior *et al.* (2007) (ver Seção 3.1, página 29) e será denotado por MPLIM-BG. Diferentemente de tais autores, é utilizada apenas uma tarefa fictícia, denominada 0 (zero), para auxiliar na modelagem. Esta tarefa é sequenciada duas vezes, sendo uma na primeira posição e outra na última. Considera-se que $P_0 = 0$ e que $S_{0i} = S_{i0} = 0$, $\forall i \in I$.

Considerando s_i , e_i , t_i e y_{ij} tais como no MPLIM-G, tem-se que as variáveis $y_{n+1,i}$ e $y_{i,n+1}$, $\forall i \in I \cup \{0\}$, e s_{n+1} deixam de ser necessárias no MPLIM-BG. Sendo assim, se comparado ao MPLIM-G, no MPLIM-BG há redução de $2n+3$ variáveis e $n+1$ restrições, com n representando o número de tarefas a serem sequenciadas.

Finalmente, o MPLIM-BG é obtido do MPLIM-G substituindo-se as restrições (3.2), (3.3), (3.4), (3.7) e (3.10) pelas restrições (3.11), (3.12), (3.13), (3.14) e (3.15), respectivamente, apresentadas a seguir:

$$s_i + P_i + y_{ij}(M + S_{ij}) - M \leq s_j \quad \forall i \in I \cup \{0\}, \forall j \in I \text{ e } i \neq j \quad (3.11)$$

$$\sum_{j=0, j \neq i}^n y_{ij} = 1 \quad \forall i \in I \cup \{0\} \quad (3.12)$$

$$\sum_{i=0, i \neq j}^n y_{ij} = 1 \quad \forall j \in I \cup \{0\} \quad (3.13)$$

$$s_i \geq 0 \quad \forall i \in I \cup \{0\} \quad (3.14)$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j \in I \cup \{0\} \quad (3.15)$$

3.3 Segundo Modelo Proposto - MPLIM-IT

A seguir é proposto um modelo de programação linear inteira mista (PLIM) indexado no tempo para representar o PSUMAA abordado. Tal modelo, denotado por MPLIM-IT, é baseado no trabalho de Paula (2008), que utilizou a discretização do tempo para modelar o problema de minimização dos atrasos em sequenciamento de máquinas paralelas com tempos de preparação dependentes da sequência. O MPLIM-IT é descrito como segue.

Seja $H = \{h_0, h_1, h_2, \dots, h_L\}$ o conjunto com as possíveis datas de início de processamento das tarefas de I e considere x_{ih} variáveis binárias, sendo $x_{ih} = 1$ se a tarefa i é programada para iniciar na data h e $x_{ih} = 0$, caso contrário, $\forall i \in I$ e $\forall h \in H$.

Se e_i e t_i representam, respectivamente, as unidades de tempo de antecipação e tempo de atraso da tarefa $i \in I$, a formulação MPLIM-IT pode ser escrita assim:

$$\min \quad Z = \sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (3.16)$$

$$\text{s.a. } x_{ih} + \sum_{u=h}^{\min(h+P_i+S_{ij}-1, H_L)} x_{ju} \leq 1 \quad \forall i, j \in I, \quad \forall h \in H \quad (3.17)$$

e $i \neq j$

$$\sum_{h=H_0}^{H_L} x_{ih} = 1 \quad \forall i \in I \quad (3.18)$$

$$\sum_{h=H_0}^{H_L} x_{ih} h + P_i + e_i \geq E_i \quad \forall i \in I \quad (3.19)$$

$$\sum_{h=H_0}^{H_L} x_{ih} h + P_i - t_i \leq T_i \quad \forall i \in I \quad (3.20)$$

$$e_i \geq 0 \quad \forall i \in I \quad (3.21)$$

$$t_i \geq 0 \quad \forall i \in I \quad (3.22)$$

$$x_{ih} \in \{0, 1\} \quad \forall i \in I \text{ e } \forall h \in H \quad (3.23)$$

A função objetivo representada pela equação (3.16) tem como critério de otimização a minimização da soma ponderada das antecipações e dos atrasos. As restrições (3.17) garantem que existe tempo suficiente para executar uma tarefa i e preparar a máquina antes do início do processamento da tarefa seguinte j , ou seja, elas garantem que o processamento da tarefa j não será iniciado antes do término do processamento da tarefa i . As restrições (3.18) garantem que cada tarefa seja executada uma única vez. As restrições (3.19) e (3.20) definem as antecipações e os atrasos de acordo com as respectivas janelas de entrega de cada tarefa. As restrições (3.21), (3.22) e (3.23) determinam os domínios das variáveis do problema.

É importante observar que, diferentemente do MPLIM-G e do MPLIM-BG, o MPLIM-IT somente é válido se o problema satisfazer a desigualdade triangular, ou seja, se as condições (3.24) a seguir forem verificadas:

$$S_{ik} \leq S_{ij} + S_{jk} + P_j \quad \forall i, j, k \in I, i \neq j, i \neq k \text{ e } j \neq k \quad (3.24)$$

No PSUMAA-JE-TPD abordado, a execução de qualquer tarefa pode ser iniciada em qualquer momento futuro ($H = [0, +\infty[$). Portanto, caso se tenha a garantia de que na sequência ótima o início do processamento da primeira tarefa não ocorre antes de h_{inf} e que o início do processamento da última tarefa é no máximo h_{sup} , pode-se discretizar o conjunto H no intervalo $[h_{inf}, h_{sup}]$, isto é, adotar $H = \{h_{inf}, h_{inf}+1, h_{inf}+2, \dots, h_{sup}\}$. Deste modo, como o MPLIM-IT é fortemente sensível à cardinalidade de H , quanto menor for o intervalo $[h_{inf}, h_{sup}]$, menor será o número de variáveis do modelo.

Capítulo 4

Algoritmo Proposto

Na prática, a aplicação das formulações de programação matemática apresentadas na Seção 3 é limitada a problemas de pequeno porte, ou seja, a problemas com poucas tarefas. Com o objetivo de resolver casos do PSUMAA-JE-TPD de maior dimensão (com um número de tarefas mais elevado), neste capítulo são propostos algoritmos heurísticos que combinam os procedimentos GRASP (ver Seção 2.2.3), Princípio da Otimalidade Próxima - POP (ver Seção 2.2.4) e Descida em Vizinhança Variável - VND (ver Seção 2.2.3).

4.1 GPV

Basicamente, a metodologia proposta – denotada por GPV – é composta de duas fases. Na primeira (linhas 1 a 12 do Algoritmo 10), gera-se uma solução com base na metaheurística GRASP, no POP e na metaheurística VND. Na segunda (linha 13 do Algoritmo 10), faz-se o pós-refinamento da solução proveniente da fase anterior por meio de outro VND. O GPV pode ser representado pelo Algoritmo 10. A função de avaliação f , a função adaptativa gulosa g , a vizinhança V , o número máximo de iterações sem melhora ($GRASP_{max}$) do GRASP, o número máximo de iterações sem melhora (VND_{max}) do VND, o número máximo de iterações sem melhora (MRD_{max}) do MRD e o parâmetro γ da fase de construção GRASP são as entradas do método.

4.2 Representação de uma Solução

Uma solução (sequência) para o PSUMAA-JE-TPD de n tarefas é representada por um vetor v de n posições, em que cada posição $i = 1, 2, \dots, n$ indica a ordem de produção da tarefa v_i . Por exemplo, dada a sequência $v = (5, 3, 2, 1, 4, 6)$, a tarefa 5 é a primeira a ser executada e a tarefa 6 a última.

4.3 Vizinhança de uma Solução

Para explorar o espaço de soluções são usados três tipos de movimentos: troca da ordem de processamento de duas tarefas da sequência de produção, realocação de uma

Algoritmo $GPV(f(\cdot), g(\cdot), V(\cdot), GRASP_{max}, VND_{max}, MRD_{max}, \gamma)$ <pre> 1 $f^* \leftarrow +\infty;$ 2 $Iter \leftarrow 0;$ 3 <u>enquanto</u> ($Iter < GRASP_{max}$) <u>faça</u> 4 $Iter \leftarrow Iter + 1;$ 5 $v_0 \leftarrow Constroi - Solucao - com - POP(f(\cdot), g(\cdot), N^R(\cdot), MRD_{max}, \gamma);$ 6 $v \leftarrow VND_1(v_0, V(\cdot), VND_{max});$ 7 <u>se</u> ($f(v) < f^*$) <u>então</u> 8 $v^* \leftarrow v;$ 9 $f^* \leftarrow f(v);$ 10 $Iter \leftarrow 0;$ 11 <u>fim-se</u> 12 <u>fim-enquanto</u>; 13 $v^* \leftarrow VND_2(v^*, V(\cdot));$ 14 Retorne $v^*;$ fim $GPV;$ </pre>

Algoritmo 10: Método GPV

tarefa para outra posição da sequência e realocação de um bloco de tarefas para outra posição da sequência. Esses movimentos definem, respectivamente, as vizinhanças N^T , N^R e N^{RB} .

Na vizinhança N^T , um exemplo de vizinho da solução $v = (5, 3, 2, 1, 4, 6)$ é a solução $v' = (5, 4, 2, 1, 3, 6)$, pois v' é obtida por meio da troca da ordem de processamento da segunda com a quinta tarefa de v . Dada uma sequência de produção de n tarefas, cada tarefa na sequência pode trocar de posição com qualquer uma das outras $n - 1$ tarefas. Ademais, trocar a ordem de processamento da i -ésima tarefa na sequência com a j -ésima tarefa é equivalente à troca da ordem de processamento da j -ésima tarefa na sequência com a i -ésima tarefa, para todo $i, j \in \{1, 2, \dots, n\}$. Sendo assim, para uma solução do PSUMAA-JE-TPD com n tarefas, há $n(n - 1)/2$ vizinhos distintos na vizinhança N^T .

A solução $v' = (5, 2, 1, 4, 3, 6)$ é um exemplo de vizinho da solução $v = (5, 3, 2, 1, 4, 6)$ na vizinhança N^R , pois é obtida pela realocação da tarefa 3, que está na segunda posição em v , para a quinta posição na sequência de produção. Dada uma sequência de produção de n tarefas, cada tarefa na sequência pode ser realocada para outras $n - 1$ posições distintas. Além disso, realocar a tarefa da i -ésima posição para a posição $i + 1$ na sequência é equivalente a realocar a tarefa da posição $i + 1$ para a i -ésima posição na sequência, para todo $i \in \{1, 2, \dots, n - 1\}$. Sendo assim, para uma solução do PSUMAA-JE-TPD com n tarefas, há $n(n - 1) - (n - 1) = (n - 1)^2$ vizinhos distintos na vizinhança N^R .

Já na vizinhança N^{RB} , um exemplo de vizinho da solução $v = (5, 3, 2, 1, 4, 6)$ é a solução $v' = (5, 4, 6, 3, 2, 1)$, pois é obtida pela realocação do bloco de três tarefas $\langle 3, 2, 1 \rangle$, que está sequenciado após a tarefa 5 em v , para depois da tarefa 6 na sequência de produção, ou seja, para duas posições sucessoras. Dada uma sequência de produção de n tarefas, existem $n - k + 1$ blocos distintos de k tarefas, para todo $k \in \{1, 2, \dots, n\}$, e cada um destes blocos podem ser realocados para $n - k$ posições distintas na sequência

de produção. Além disso, realocar um bloco de i tarefas para j posições sucessoras é equivalente à realocação de um bloco de j tarefas para i posições antecessoras, para todo $i, j \in \{1, 2, \dots, n\}$ e $i + j \leq n$. Sendo assim, para uma solução do PSUMAA-JE-TPD com n tarefas, há $\left(\sum_{k=1}^n (n-k+1)(n-k)\right)/2 = \left(\sum_{k=1}^n (n-k)^2 + \sum_{k=1}^n (n-k)\right)/2 = \left(n(2n-1)(n-1)/6 + n(n-1)/2\right)/2 = n(n-1)(n+1)/6$ vizinhos distintos na vizinhança N^{RB} .

Com o objetivo de evitar a visita a vizinhos já visitados e consequentemente reduzir o custo computacional, todas as observações mencionadas nos parágrafos acima foram consideradas na implementação computacional do algoritmo proposto.

4.4 Função de Avaliação

Como os movimentos utilizados não geram soluções inviáveis, uma sequência v é avaliada pela função f dada pela equação (4.1), a qual deve ser minimizada:

$$f(v) = \sum_{i=1}^n (\alpha_i e_i + \beta_i t_i) \quad (4.1)$$

Para determinar os valores de e_i e t_i , é utilizado o algoritmo de determinação das datas ótimas de início de processamento das tarefas (ADDOIPT) apresentado a seguir.

4.4.1 Algoritmo de Determinação das Datas Ótimas de Início de Processamento das Tarefas

Dada uma sequência de produção, é utilizado o algoritmo de determinação das datas ótimas de início de processamento das tarefas (ADDOIPT) proposto por Gomes Júnior *et al.* (2007) e baseado nos trabalhos de Wan & Yen (2002) e Lee & Choi (1995). O restante desta subseção é destinado à apresentação deste algoritmo.

Seja $V = (J_1, J_2, \dots, J_n)$ uma sequência do conjunto de tarefas I . Logo $J_i \in I$ e $J_i \neq J_j$, para todo $i, j \in \{1, 2, \dots, n\}$. Diz-se que uma sub-sequência de tarefas $B = (J_u, J_{u+1}, \dots, J_v) \subseteq V$ forma um bloco na sequência V se as tarefas em B são sequenciadas consecutivamente sem tempo ocioso entre elas e existe ociosidade entre as tarefas J_{u-1} e J_u e as tarefas J_v e J_{v+1} .

Considere que há l blocos em V , ou seja, $V = B_1 \cup B_2 \cup \dots \cup B_l$. Sejam $prim(j)$ e $ult(j)$ a primeira e a última tarefa no bloco B_j , respectivamente.

O processamento da primeira tarefa (J_1) da sequência V é programado para ser finalizado na data E_{J_1} se $P_{J_1} \leq E_{J_1}$ ou iniciado na data 0 (finalizando na data P_{J_1}) se $P_{J_1} > E_{J_1}$.

Seja C_k a data de conclusão da tarefa k .

As demais tarefas são programadas da seguinte forma:

- Se $C_{J_k} + S_{(J_k)(J_{k+1})} + P_{J_{k+1}} \geq E_{J_{k+1}}$, a tarefa J_{k+1} tem seu processamento programado para ser finalizado na data $C_{J_k} + S_{(J_k)(J_{k+1})} + P_{J_{k+1}}$ e passa a ser a última tarefa do bloco corrente.

- Caso contrário, se $C_{J_k} + S_{(J_k)(J_{k+1})} + P_{J_{k+1}} < E_{J_{k+1}}$, a tarefa J_{k+1} tem seu processamento programado para ser finalizado na data $E_{J_{k+1}}$ e um novo bloco é iniciado.

Após a determinação da data de conclusão do processamento de cada tarefa, os blocos são reposicionados em suas respectivas posições ótimas na subsequência corrente. Deste modo, após a inserção da última tarefa na sequência, todos os blocos estão otimamente alocados.

A antecipação e o atraso de uma tarefa i na sequência são dados, respectivamente por $\max\{0, E_i - C_i\}$ e $\max\{0, C_i - T_i\}$, respectivamente. Consequentemente, o custo gerado por antecipação e atraso de uma tarefa i concluída na data C_i pode ser dado pela função $g_i(C_i) = \alpha_i \cdot \max\{0, E_i - C_i\} + \beta_i \cdot \max\{0, C_i - T_i\}$.

Logo o custo para deslocar um bloco B_j , x unidades de tempo para a esquerda, pode ser expresso pela equação:

$$Custo_j(x) = \sum_{i \in B_j} g_i(C_i - x), \quad (4.2)$$

em que $C_i - x$ é a nova data de conclusão do processamento da tarefa i .

Claramente a função $g_i(C_i - x)$ é uma função linear convexa por partes em relação a x para todo $i \in \{1, 2, \dots, n\}$.

Lema 1 (Wan & Yen, 2002) *O somatório de duas funções lineares convexas por partes é também uma função linear convexa por partes.*

Baseado no Lema 1, tem-se:

Proposição 1 (Wan & Yen, 2002) *$Custo_j(x)$ é uma função linear convexa por partes em relação a x , para todo $j \in \{1, 2, \dots, l\}$.*

Devido a natureza linear convexa por partes da função custo, o custo mínimo do bloco B_j ocorre nos pontos extremos de sua função $Custo_j$, isto é, no início ou no final da janela de entrega de uma das tarefas no bloco. Tal custo mínimo pode ser facilmente obtido pela comparação dos somatórios das penalidades das tarefas no bloco, onde cada um destes somatórios é obtido ao programar uma tarefa do bloco para ser concluída no início ou no final de sua janela de entrega. Todos os possíveis somatórios são calculados e o menor deles corresponde ao ponto de custo mínimo do bloco.

Quando o ponto mínimo do bloco B_j é encontrado, todo o bloco é movido em direção ao ponto mínimo até que um dos três casos seguintes aconteça:

- (i) $s_{prim(j)} = 0$;
- (ii) O ponto mínimo é alcançado;
- (iii) $s_{prim(j)}$ torna-se igual a $C_{ult(j-1)} + S_{ult(j-1)(prim(j))}$.

Se ocorrer o caso (iii), o bloco B_j é unido ao bloco B_{j-1} , resultando em um novo bloco B_{j-1} . Logo o ponto mínimo do novo bloco B_{j-1} deve ser obtido.

O procedimento anterior deve ser realizado até que ocorra o caso (i) ou (ii) para cada bloco.

A seguinte proposição garante que o algoritmo apresentado determina as datas ótimas de início do processamento das tarefas de uma dada sequência.

Proposição 2 (Wan & Yen, 2002) *O custo total de uma dada sequência de tarefas alcança seu valor ótimo se cada bloco B_j na sequência alcançar seu ponto mínimo, exceto $s_{prim(1)}$, que pode ser igual a zero para o primeiro bloco.*

O Algoritmo 11, adaptado de Gomes Júnior (2007), descreve o procedimento ADDOIPT. Nele, o procedimento *MudaBloco* movimenta o bloco J até seu ponto de mínimo e, se necessário, combina-o com o bloco anterior.

```

procedimento Determina – Datas – Otimas – Inicio( $n, s$ )
1   $B \leftarrow 1$ ;
2   $prim(B) \leftarrow ult(B) \leftarrow 1$ ;
3   $s_1 \leftarrow \max(0, E_1 - P_1)$ ;
4   $C_1 \leftarrow \max(E_1, P_1)$ ;
5  para ( $i = 2, \dots, n$ ) faça
6    se ( $C_{i-1} + P_i + S_{(i-1)(i)} < E_i$ ) então
7       $B \leftarrow B + 1$ ;
8       $prim(B) \leftarrow ult(B) \leftarrow i$ ;
9       $s_i \leftarrow E_i - P_i + S_{(i-1)(i)}$ ;
10      $C_i \leftarrow E_i$ ;
11   senão
12     se ( $C_{i-1} + P_i + S_{(i-1)(i)} = E_i$ ) então
13        $ult(B) \leftarrow i$ ;
14        $s_i \leftarrow C_{i-1} + S_{(i-1)(i)}$ ;
15        $C_i \leftarrow E_i$ ;
16     senão
17        $ult(B) \leftarrow i$ ;
18        $s_i \leftarrow C_{i-1} + S_{(i-1)(i)}$ ;
19        $C_i \leftarrow s_i + P_i$ ;
20   fim-se
21 fim-se
22 repita (até todos os blocos estiverem no ponto mínimo ou  $s_{prim(1)} = 0$ )
23   MudaBloco( $B$ );
24 fim-repita;
25 fim-para;
26    $f(s) \leftarrow \sum_{i=1}^n \alpha_i \cdot \max\{0, E_i - C_i\} + \beta_i \cdot \max\{0, C_i - T_i\}$ ;
27 Retorne  $f(s)$ ;
fim Determina – Datas – Otimas – Inicio;

```

Algoritmo 11: Algoritmo para determinar as datas ótimas de início de processamento das tarefas.

Mais detalhes do ADDOIPT podem ser encontrados em Gomes Júnior (2007).

4.5 Construção de uma Solução

Nesta etapa do GPV (linha 5 do Algoritmo 10), uma solução é formada, tarefa por tarefa, de forma parcialmente gulosa, seguindo as ideias da fase de construção do algoritmo GRASP (ver Seção 2.2.3) e do POP (ver Seção 2.2.4). O Algoritmo 12 representa o procedimento proposto para construir uma solução.

procedimento *Constroi – Solucao – com – POP*($f(\cdot)$, $g(\cdot)$, $V(\cdot)$, $MRDmax$, γ)

- 1 $s \leftarrow \emptyset$;
- 2 Inicialize o conjunto C de candidatos;
- 3 enquanto ($C \neq \emptyset$) faça
- 4 $g(t_{min}) \leftarrow \min\{g(t) \mid t \in C\}$;
- 5 $g(t_{max}) \leftarrow \max\{g(t) \mid t \in C\}$;
- 6 $LRC \leftarrow \{t \in C \mid g(t) \leq g(t_{min}) + \gamma(g(t_{max}) - g(t_{min}))\}$;
- 7 Aleatoriamente, selecione um elemento $t \in LRC$;
- 8 $s \leftarrow s \cup \{t\}$;
- 9 $s \leftarrow MRD(s, N^R(\cdot), MRDmax)$; /* Aplica o POP */
- 10 Atualize o conjunto C de candidatos;
- 11 fim-enquanto;
- 12 Retorne s ;

fim *Constroi – Solucao – com – POP*;

Algoritmo 12: Construção de uma solução com POP.

Foram experimentadas 11 funções adaptativas gulosas para estimar o benefício da inserção de cada tarefa j ainda não sequenciada na iteração i , sendo elas:

- $g_1(j) = E_j$;
- $g_2(j) = T_j$;
- $g_3(j) = E_j + T_j$;
- $g_4(j) = T_j - P_j$;
- $g_5(j) = E_j - P_j$;
- $g_6(j) = \frac{(E_j - P_j)\alpha_j + (T_j - P_j)\beta_j}{\alpha_j + \beta_j}$;
- $g_7(j) = E_j + T_j - 2P_j$;
- $g_8(j) = \frac{2E_j}{\max\{E_k \mid k \in F_i\}} + \frac{2T_j}{\max\{T_k \mid k \in F_i\}} + \frac{\alpha_j}{\max\{\alpha_k \mid k \in F_i\}} - \frac{\beta_j}{\max\{\beta_k \mid k \in F_i\}} + \frac{P_j}{\max\{P_k \mid k \in F_i\}} + \frac{S_{i-1,j}}{\max\{S_{i-1,k} \mid k \in F_i\}}$, onde F_i representa o conjunto das tarefas ainda não sequenciadas até a i -ésima iteração (para $i = 1$, no lugar de $\frac{S_{i-1,j}}{\max\{S_{i-1,k} \mid k \in F_i\}}$, é utilizado $-\frac{\text{média}\{S_{kj} \mid k \in I \text{ e } k \neq j\}}{\max\{S_{kl} \mid k, l \in I \text{ e } k \neq l\}}$);

- $g_9(j) = \frac{E_j}{\max\{E_k | k \in F_i\}} + \frac{2T_j}{\max\{T_k | k \in F_i\}} + \frac{\alpha_j}{\max\{\alpha_k | k \in F_i\}} - \frac{\beta_j}{\max\{\beta_k | k \in F_i\}} + \frac{P_j}{\max\{P_k | k \in F_i\}} + \frac{S_{i-1,j}}{\max\{S_{i-1,k} | k \in F_i\}}$, onde F_i representa o conjunto das tarefas ainda não sequenciadas até a i -ésima iteração (para $i = 1$, no lugar de $\frac{S_{i-1,j}}{\max\{S_{i-1,k} | k \in F_i\}}$, é utilizado $-\frac{\text{média}\{S_{kj} | k \in I \text{ e } k \neq j\}}{\max\{S_{kl} | k, l \in I \text{ e } k \neq l\}}$);
- $g_{10}(j) = \frac{E_j}{\max\{E_k | k \in F_i\}} + \frac{T_j}{\max\{T_k | k \in F_i\}} + \frac{\alpha_j}{\max\{\alpha_k | k \in F_i\}} - \frac{\beta_j}{\max\{\beta_k | k \in F_i\}} + \frac{P_j}{\max\{P_k | k \in F_i\}} + \frac{S_{i-1,j}}{\max\{S_{i-1,k} | k \in F_i\}}$, onde F_i representa o conjunto das tarefas ainda não sequenciadas até a i -ésima iteração (para $i = 1$, no lugar de $\frac{S_{i-1,j}}{\max\{S_{i-1,k} | k \in F_i\}}$, é utilizado $-\frac{\text{média}\{S_{kj} | k \in I \text{ e } k \neq j\}}{\max\{S_{kl} | k, l \in I \text{ e } k \neq l\}}$);
- $g_{11}(j) = \frac{2E_j}{\max\{E_k | k \in F_i\}} + \frac{T_j}{\max\{T_k | k \in F_i\}} + \frac{\alpha_j}{\max\{\alpha_k | k \in F_i\}} - \frac{\beta_j}{\max\{\beta_k | k \in F_i\}} + \frac{P_j}{\max\{P_k | k \in F_i\}} + \frac{S_{i-1,j}}{\max\{S_{i-1,k} | k \in F_i\}}$, onde F_i representa o conjunto das tarefas ainda não sequenciadas até a i -ésima iteração (para $i = 1$, no lugar de $\frac{S_{i-1,j}}{\max\{S_{i-1,k} | k \in F_i\}}$, é utilizado $-\frac{\text{média}\{S_{kj} | k \in I \text{ e } k \neq j\}}{\max\{S_{kl} | k, l \in I \text{ e } k \neq l\}}$);

Cada função g_k , $k = 1, 2, \dots, 11$, foi utilizada para construir uma solução gulosa para cada um dos 608 problemas-teste com até 100 tarefas apresentados na Subseção 5.1. A qualidade da solução encontrada por cada função adaptativa gulosa g_k , aplicada a um determinado problema-teste, foi mensurada pela equação (4.3), onde f_{g_k} representa o valor da função objetivo da solução encontrada pela função adaptativa gulosa g_k e f_{g^*} denota o valor da função objetivo da melhor solução encontrada pelas 11 funções gulosas.

$$Desvio_{g_k} = \frac{f_{g_k} - f_{g^*}}{f_{g^*}} \quad (4.3)$$

Os resultados encontrados estão sintetizados na Tabela 4.1. Nesta tabela, a primeira linha indica as funções adaptativas gulosas testadas. Já na segunda linha, são apresentadas as médias dos desvios encontrados pelas respectivas funções aplicadas aos 608 problemas-teste.

Tabela 4.1: Resultados apresentados pelas funções adaptativas gulosas testadas.

Função	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}
Desvio Médio	0,47	0,46	0,45	0,54	0,54	0,56	0,53	0,28	0,38	0,57	0,36

Na Tabela 4.1 é possível observar que as funções g_1 , g_2 , g_3 , g_8 , g_9 e g_{11} apresentaram desvios médios menores que 0,5. Sendo assim, optou-se por utilizar estas seis funções, uma em cada iteração do GRASP. Tais funções são utilizadas segundo a ordem crescente do

desvio médio encontrado, ou seja, na primeira iteração utiliza-se a função g_8 , na segunda utiliza-se a função g_{11} e assim por diante, até que todas as seis funções sejam utilizadas; repetindo-se a aplicação desta sequência a seguir.

Ainda na etapa de construção de uma solução, é aplicado o POP (linha 9 do Algoritmo 12). Assim, sempre que uma nova tarefa é inserida na solução em formação, esta é submetida a uma busca local. A busca local utilizada consiste em um método de descida, o qual é descrita a seguir.

Para determinar a vizinhança utilizada no método de descida relativo ao POP, foi realizado o seguinte procedimento em um problema-teste com 30 tarefas. Cada uma das seis funções adaptativas gulosas determinadas acima foi utilizada para construir uma solução para o problema. As soluções construídas foram, então, submetidas ao procedimento de Descida de Primeira Melhora com a vizinhança N^T . Este mesmo procedimento foi realizado com a vizinhança N^R . Como a ordem de visita aos vizinhos é determinante em um método de descida, o procedimento descrito foi realizado 20 vezes com cada vizinhança, sendo a ordem de visitas sempre escolhida aleatoriamente. No final de todas as execuções, as soluções obtidas com a vizinhança N^R apresentaram uma média 9,55% menor que a média das soluções obtidas com a vizinhança N^T . Além disso, a melhor solução obtida com a vizinhança N^R foi 7,63% melhor que a melhor solução obtida com a N^T . Estes resultados sugerem que a vizinhança N^R é capaz de encontrar soluções de melhor qualidade e com melhor média que as soluções obtidas com a vizinhança N^T e, por isso, foi a escolhida para o método de descida usado no POP.

Como o POP é aplicado sempre que uma tarefa é inserida na solução em formação, utilizou-se uma descida randômica com vizinhança N^R , visto que este método de descida demanda menos tempo computacional que a Descida de Primeira Melhora. Sendo assim, dada uma solução, aleatoriamente escolhe-se uma tarefa na sequência e uma nova posição para ela. Se a nova sequência produzir uma solução com um valor menor para a função de avaliação, a nova sequência é aceita e passa a ser a solução corrente; caso contrário, é testada outra realocação aleatória. A busca é interrompida após $MRDMax$ realocações consecutivas sem melhora na solução corrente, sendo $MRDMax$ um parâmetro do método.

4.6 VND₁

Para refinar as soluções contruídas na primeira fase do GPV (linha 6 do Algoritmo 10), utiliza-se um procedimento baseado em Descida em Vizinhança Variável (ver Seção 2.2.3), denotado por VND₁. Basicamente o VND₁ consiste em três passos:

Passo 1: Descida randômica com a vizinhança N^T ;

Passo 2: Descida randômica com a vizinhança N^R ;

Passo 3: Descida randômica com a vizinhança N^{RB} ;

No *Passo 1*, duas tarefas são escolhidas de forma aleatória e a ordem de seus processamentos na sequência de produção são trocadas. Se a nova sequência produzir uma solução com um valor menor para a função de avaliação, a nova sequência é aceita e passa a ser a solução corrente; caso contrário, é testada outra troca aleatória. O *Passo 1* termina

quando ocorrer $MRDmax$ trocas consecutivas sem melhora na solução corrente, sendo $MRDMax$ um parâmetro do método. Neste último caso, passa-se para o *Passo 2*.

No *Passo 2*, aleatoriamente escolhe-se uma tarefa na sequência e uma nova posição para ela. Se a nova sequência produzir uma solução com um valor menor para a função de avaliação, a nova sequência passa a ser a solução corrente e volta-se ao *Passo 1*; caso contrário, é testada outra realocação aleatória. O *Passo 2* é interrompido após $MRDMax$ realocações consecutivas sem melhora na solução corrente. Neste último caso, passa-se para o *Passo 3*.

No *Passo 3*, aleatoriamente escolhe-se um bloco de tarefas na sequência de produção (o tamanho do bloco também é escolhido de forma aleatória) e uma nova posição para ele. Se a nova sequência produzir uma solução com um valor menor para a função de avaliação, a nova sequência passa a ser a solução corrente e volta-se ao *Passo 1*; caso contrário, é testada outra realocação de bloco aleatória. O *Passo 3* é interrompido após $MRDMax$ realocações de blocos consecutivas sem melhora na solução corrente. Neste último caso, o VND_1 é interrompido e a melhor solução encontrada é retornada.

É interessante observar que a solução oriunda do VND_1 não é necessariamente um ótimo local em relação às vizinhanças adotadas, visto que nem toda a vizinhança é analisada em cada cada passo.

4.7 VND_2

Como a solução proveniente da primeira fase do GPV (linhas 1 a 12 do Algoritmo 10) não é necessariamente um ótimo local em relação às vizinhanças adotadas, ela é submetida a uma busca local mais efetiva, no caso, também baseada na Busca em Vizinhança Variável (VND_2). Nesta, a exploração do espaço de soluções é realizada de acordo com os seguintes passos:

Passo 1: Descida Primeiro de Melhora com a vizinhança N^T ;

Passo 2: Descida completa com a vizinhança N^{RB} ;

Se no *Passo 1*, todos os vizinhos com relação à vizinhança N^T não são de melhora, passa-se para o *Passo 2*.

No *Passo 2*, em que são realizadas realocações de blocos de tarefas, inicialmente testam-se todas as possíveis realocações de blocos com apenas uma tarefa e, quando não for mais possível melhorar a solução com um determinado tamanho de bloco, passa-se a explorar movimentos com blocos de tamanho imediatamente superior. Se uma solução de melhora é encontrada, volta-se ao *Passo 1*. O procedimento é interrompido quando um ótimo local com relação às vizinhanças N^T e N^{RB} é encontrado. Observa-se que a vizinhança N^R corresponde a realocar blocos com apenas uma tarefa. Logo, a vizinhança N^R está contida na vizinhança N^{RB} e um ótimo local em relação à esta última também é um ótimo local em relação à primeira.

4.8 Redução do Espaço de Busca - GPV-REB

A avaliação de todos os vizinhos gerados em uma busca local requer um grande e muitas vezes injustificável esforço computacional (França Filho, 2007). Por isso, é inte-

ressante utilizar estratégias que tenham por objetivo evitar a avaliação de soluções que apresentam determinadas características julgadas ruins e, assim, reduzir o esforço computacional. Como consequência, mais vizinhos podem ser explorados ou, então, problemas de dimensões maiores podem ser resolvidos.

No caso do PSUMAA-JE-TPD, se duas tarefas $i, j \in I$ satisfazem as condições (4.4), (4.5), (4.6) e (4.7):

$$E_i \leq E_j, \quad (4.4)$$

$$\alpha_i \leq \alpha_j, \quad (4.5)$$

$$T_i \leq T_j \text{ e} \quad (4.6)$$

$$\beta_i \geq \beta_j, \quad (4.7)$$

espera-se que boas soluções possuem a tarefa i sequenciada antes da tarefa j . Deste modo, se i é sequenciada antes de j na sequência corrente, é proposta a seguinte estratégia de redução do espaço de busca:

- Durante a exploração da vizinhança N^T , a solução produzida pela troca das ordens de processamentos da tarefa i com a tarefa j é rejeitada sem ser avaliada.
- Durante a exploração da vizinhança N^R , as realocações da tarefa i para posições sucessoras à tarefa j são descartadas sem serem avaliadas. Do mesmo modo, as realocações da tarefa j para as posições antecessoras à tarefa i também são descartadas.
- Durante a exploração da vizinhança N^{RB} , as realocações de blocos que levam a tarefa i a ser sequenciada depois da tarefa j na sequência de produção são descartadas sem serem avaliadas.

A Figura 4.1 ilustra a situação em que $E_3 \leq E_2$, $\alpha_3 \leq \alpha_2$, $T_3 \leq T_2$ e $\beta_3 \geq \beta_2$. Neste caso, as soluções obtidas pela troca das ordens de processamento da tarefa 3 com a tarefa 2 ou por realocações, de tarefas ou de blocos, que levam a tarefa 3 a ser sequenciada depois da tarefa 2 são descartadas antes de serem avaliadas.

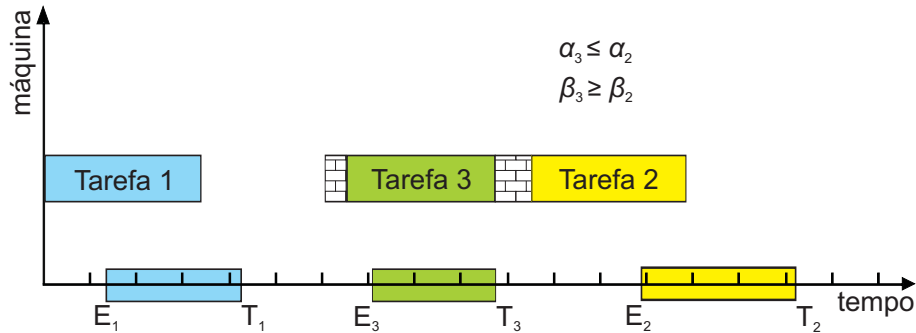


Figura 4.1: Exemplo de situação em que a estratégia de redução do espaço de busca é utilizada.

A estratégia de redução do espaço de busca proposta acima foi utilizada em um algoritmo derivado do algoritmo GPV, denotado por GPV-REB. O GPV-REB se diferencia

do GPV pela utilização da estratégia de redução do espaço de busca nas buscas locais da primeira fase do algoritmo, ou seja, na descida randômica aplicada nas soluções parciais da fase de construção de uma solução (linha 9 do Algoritmo 12) e no VND₁ (linha 6 do Algoritmo 10). A segunda fase dos dois algoritmos é a mesma.

Capítulo 5

Resultados Computacionais

Neste capítulo são apresentados os resultados computacionais obtidos pela aplicação das formulações de programação matemática propostas, bem como dos algoritmos heurísticos desenvolvidos. Primeiramente, são apresentados os conjuntos de problemas-teste utilizados. Na Seção 5.2 são apresentados os resultados obtidos com as formulações de programação matemática apresentadas no Capítulo 3. Na Seção 5.3 são apresentados os resultados obtidos com os algoritmos GPV e GPV-REB, bem como os procedimentos utilizados na determinação dos parâmetros de tais algoritmos.

Todos os experimentos computacionais foram realizados em um computador Intel(R) Core(TM) 2 Duo E8400 2.99 GHz, com 2 GB de memória RAM e sistema operacional Windows XP. Apesar de o processador deste equipamento possuir 2 núcleos, o algoritmo desenvolvido não foi otimizado para multiprocessamento.

5.1 Problemas-teste

Foram utilizados três conjuntos de problemas-teste. O primeiro deles, denotado por BDS, foi proposto por Gomes Júnior (2007), sendo baseado nos trabalhos de Wan & Yen (2002), Liaw (1999) e Mazzini & Armentano (2001). Esta base de dados é constituída por grupos de problemas-teste com 8, 9, 10, 11, 12, 15, 20, 25, 30, 40, 50 e 75 tarefas.

Dada uma tarefa i , o tempo de processamento (P_i), o custo por unidade de atraso (β_i) e o custo por unidade de antecipação (α_i) são valores inteiros selecionados aleatoriamente dentro dos intervalos $[1, 100]$, $[20, 100]$ e $[0, \beta_i]$, respectivamente.

O centro da janela de entrega de $i \in I$ é um valor aleatório dentro do intervalo $\left[\left(1 - FA - \frac{VRJ}{2}\right) \times TTP, \left(1 - FA + \frac{VRJ}{2}\right) \times TTP \right]$, onde TTP é o tempo total de processamento de todas as tarefas, FA é o fator de atraso e VRJ é a variação relativa da janela de entrega. São utilizados os valores 0,1; 0,2; 0,3 e 0,4 para FA e 0,8; 1,0 e 1,2 para VRJ .

Os tamanhos das janelas de entrega são uniformemente e discretamente distribuídos no intervalo $\left[1, \frac{TTP}{n}\right]$, onde n é o número de tarefas de I .

Para toda tarefa $j \in I$ e $j \neq i$, o tempo de *setup* (S_{ij}) é um inteiro aleatório dentro do intervalo $[0, 50]$. Utilizou-se tempos de preparação simétricos, ou seja, $S_{ij} = S_{ji}$.

Como são utilizados quatro valores distintos de FA e três valores distintos de VRJ , há 12 problemas-teste para cada número de tarefas, totalizando 144 problemas-teste. Doravante, esta base de dados será denotada por BDS.

O segundo conjunto de problemas-teste utilizado, denotado por BDNS, foi gerado considerando-se a metodologia apresentada nos trabalhos de Wan & Yen (2002) e Rabadi *et al.* (2004). Foram gerados grupos de problemas-teste com 6, 7, 8, 9, 10, 11, 12, 15, 20, 30, 40, 50, 75, 100 e 150 tarefas.

Dada uma tarefa $i \in I$, o tempo de processamento (P_i), o custo por unidade de atraso (β_i) e o custo por unidade de antecipação (α_i) são valores inteiros selecionados aleatoriamente dentro dos intervalos $[1, 100]$, $[1, 10]$ e $[1, \beta_i]$, respectivamente.

O centro da janela de entrega de i é um valor inteiro aleatório dentro do intervalo $\left[\left(1 - FA - \frac{VRJ}{2}\right) \times TTP, \left(1 - FA + \frac{VRJ}{2}\right) \times TTP \right]$, onde TTP é o tempo total de processamento de todas as tarefas, FA é o fator de atraso e VRJ é a variação relativa da janela de entrega. São utilizados os valores 0,1; 0,3; 0,5 e 0,8 para FA e 0,4; 0,7; 1,0 e 1,3 para VRJ .

O tamanho da janela de entrega de i é um número inteiro selecionado aleatoriamente dentro do intervalo $\left[0, \frac{TTP}{n} \right]$, onde n é o número de tarefas de I .

Para toda tarefa $j \in I$ e $j \neq i$, o tempo de *setup* (S_{ij}) é um inteiro aleatório dentro do intervalo $[5, 30]$. Nesta base de dados, os tempos de *setup* são genéricos e, assim, não são necessariamente simétricos como na base anterior.

Como são utilizados 4 valores distintos de TTP e 4 de VRJ , são 16 problemas-teste de cada tamanho, totalizando 240 problemas-teste. Esta nova base de dados será denotada por BDNS.

O terceiro conjunto de problemas-teste, denotado por BDDT, também foi baseado nos trabalhos de Rabadi *et al.* (2004) e Wan & Yen (2002), mas diferem do anterior na caracterização dos tempos de preparação, que, neste caso, satisfazem a desigualdade triangular. Os problemas-teste deste conjunto foram gerados do modo descrito a seguir.

Dada uma tarefa $i \in I$, o tempo de processamento (P_i), o custo por unidade de atraso (β_i) e o custo por unidade de antecipação (α_i) são números inteiros selecionados aleatoriamente dentro dos intervalos $[1, 40]$, $[1, 10]$ e $[1, \beta_i]$, respectivamente.

O centro da janela de entrega de i é um valor inteiro aleatório dentro do intervalo $\left[\left(1 - FA - \frac{VRJ}{2}\right) \times TTP, \left(1 - FA + \frac{VRJ}{2}\right) \times TTP \right]$, em que TTP é o tempo total de processamento de todas as tarefas, FA é o fator de atraso e VRJ é a variação relativa da janela de entrega.

O tamanho da janela de entrega é um valor inteiro selecionado aleatoriamente no intervalo $\left[0, \frac{TTP}{n} \right]$, sendo n o número de tarefas.

Para toda tarefa $j \in I$ e $j \neq i$, o tempo de *setup* (S_{ij}) é um número inteiro escolhido aleatoriamente dentro do intervalo $[5, 15]$. Assim, os tempos de *setup* também não são necessariamente simétricos.

Foram gerados conjuntos de problemas-teste com 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 e 20 tarefas, respectivamente, sendo utilizados os valores 0,1; 0,3; 0,5 e 0,8 para FA e 0,4; 0,7; 1,0 e 1,3 para VRJ . Deste modo, há 16 problemas-teste em cada conjunto,

totalizando 240 problemas-teste.

Durante a geração dos problemas-teste desta última base de dados, sempre que um problema gerado não satisfazia a desigualdade triangular, ele era descartado e outro problema com os mesmos valores para FA e VRJ era gerado. Portanto, todos os problemas-teste desta base de dados satisfazem a desigualdade triangular.

5.2 Resultados Obtidos com as Formulações de Programação Matemática

Os modelos de programação matemática apresentados na Seção 3 foram implementados no modelador AMPL e resolvidos pelo otimizador CPLEX, versão 10.1, da ILOG, em sua configuração padrão. Foram realizados experimentos nos problemas-teste de pequeno porte das três bases de dados apresentadas na Seção 5.1, totalizando três baterias de testes.

Para os modelos MPLIM-G e MPLIM-BG, foi utilizado $M = 1000$, valor adotado por Gomes Júnior (2007). Para cada problema-teste, o conjunto H do MPLIM-IT foi determinado pelas expressões (5.1) e (5.2):

$$h_{inf} = \max \left(0, \min \left\{ T_i \mid i \in I \right\} + \min \left\{ S_{ij} \mid i, j \in I \text{ e } i \neq j \right\} - \sum_{i \in I} P_i - \sum_{i \in I} \max \left\{ S_{ij} \mid j \in I \text{ e } i \neq j \right\} \right) \quad (5.1)$$

$$\begin{aligned} h_{sup} = & \max \left\{ E_i \mid i \in I \right\} + \sum_{i \in I} P_i + \sum_{i \in I} \max \left\{ S_{ij} \mid j \in I \text{ e } i \neq j \right\} \\ & - \min \left\{ S_{ij} \mid i, j \in I \text{ e } i \neq j \right\} - \min \left\{ P_i \mid i \in I \right\} \\ & - \min \left(\min \left\{ P_i \mid i \in I \right\}, \max \left\{ E_i \mid i \in I \right\} \right) \end{aligned} \quad (5.2)$$

em que h_{inf} garante que todas as tarefas podem ser concluídas sem atraso (ou o processamento da primeira tarefa é iniciado na data 0) e h_{sup} assegura que todas as tarefas podem ser executadas sem antecipação. Portanto, tem-se a garantia de que as datas de início de processamento das tarefas na sequência ótima pertencem ao conjunto $H = \{h_{inf}, h_{inf} + 1, h_{inf} + 2, \dots, h_{sup}\}$.

Considerou-se, ainda, o limite de 3600 segundos para a resolução de cada problema-teste pelo CPLEX. Para os problemas em que este limite de tempo foi atingido, a solução retornada não é necessariamente ótima; porém, um limite inferior para o ótimo de tal problema é retornado pelo CPLEX. Este limite foi utilizado para medir a qualidade da solução retornada, que é obtida pela equação (5.3):

$$gap = \frac{(f^{CPLEX} - L)}{L} \times 100\% \quad (5.3)$$

em que f^{CPLEX} e L representam, respectivamente, o valor da função objetivo da solução e o limite inferior encontrados pelo otimizador. Deste modo, se uma solução ótima é encontrada, tem-se $gap = 0$.

Os resultados obtidos em cada bateria de testes são apresentados a seguir.

5.2.1 Primeira Bateria de Testes - BDS

A primeira bateria de testes foi realizada com os problemas-teste da base de dados BDS. Os problemas-teste dessa base de dados possuem os tempos de *setup* simétricos e não satisfazem, necessariamente, a desigualdade triangular. Deste modo, apenas os modelos MPLIM-G e MPLIM-BG foram utilizados nesta bateria, visto que o MPLIM-IT não permite resolver problemas em que a desigualdade triangular não é satisfeita.

Os resultados encontrados são resumidos na Tabela 5.1, onde a primeira coluna indica o número de tarefas dos problemas-teste de cada conjunto. Ademais, para cada conjunto de problemas, as colunas “Soluções Ótimas” mostram (em porcentagem) em quantos problemas o CPLEX encontrou uma solução ótima com os respectivos modelos, as colunas “ $\overline{gap}$ ” mostram as médias dos $\overline{gap}$ ’s retornados pelos respectivos modelos e as colunas “Tempo Médio” mostram as médias dos tempos (em segundos) demandados por tal otimizador na resolução dos problemas-teste com os respectivos modelos.

Tabela 5.1: Comparação MPLIM-G $\times$ MPLIM-BG aplicados à BDS.

# Tarefas	MPLIM-G			MPLIM-BG		
	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)
08	100,00	0,00	1,52	100,00	0,00	1,61
09	100,00	0,00	35,12	100,00	0,00	31,71
10	100,00	0,00	86,68	100,00	0,00	77,16
11	66,67	15,78	1850,01	75,00	10,53	1974,97
12	50,00	30,51	1957,23	50,00	30,98	1962,74
Média	83,33	9,26	786,11	85,00	8,30	809,64

Na Tabela 5.1, observa-se que o MPLIM-G e o MPLIM-BG proporcionaram ao CPLEX desempenhos semelhantes para a base de dados em questão. Usando-se esses dois modelos, o CPLEX foi capaz de resolver na otimalidade todos os problemas-teste com até 10 tarefas e com tempos médios de processamentos bem próximos uns dos outros. Para os problemas-teste com 11 tarefas, o MPLIM-BG permitiu resolver 75,00% dos casos, contra 66,67% do MPLIM-G, e proporcionou $\overline{gap}$ inferior ao propiciado pelo MPLIM-G. Para estes problemas-teste, o tempo médio demandado pelo CPLEX com o MPLIM-G foi sensivelmente inferior ao despendido com o MPLIM-BG. Para os problemas-teste com 12 tarefas, ambos os modelos proporcionaram ao otimizador resolver 50,00% dos problemas, com $\overline{gap}$ ’s próximos de 31% e tempos médios próximos de 1960 segundos.

5.2.2 Segunda Bateria de Testes - BDNS

A segunda bateria de testes foi realizada com os problemas-teste da base de dados BDNS. Os problemas-teste dessa base de dados também não satisfazem, necessariamente, a desigualdade triangular. Do mesmo modo, apenas os modelos MPLIM-G e MPLIM-BG

foram utilizados nesta bateria, já que o MPLIM-IT não permite resolver problemas em que a desigualdade triangular não é satisfeita.

A Tabela 5.2 resume os resultados encontrados. Nela, a primeira coluna indica o número de tarefas em cada problema-teste dos conjuntos. Além disso, para cada conjunto de problemas, as colunas “Soluções Ótimas” mostram (em porcentagem) em quantos problemas o CPLEX encontrou uma solução ótima com os respectivos modelos, as colunas “ $\overline{gap}$ ” mostram as médias dos gap ’s com os respectivos modelos e as colunas “Tempo Médio” mostram as médias dos tempos (em segundos) demandados por tal otimizador na resolução dos problemas-teste com os respectivos modelos.

Tabela 5.2: Comparação MPLIM-G $\times$ MPLIM-BG aplicados à BDNS.

# Tarefas	MPLIM-G			MPLIM-BG		
	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)
06	100,00	0,00	0,10	100,00	0,00	0,09
07	100,00	0,00	0,76	100,00	0,00	0,76
08	100,00	0,00	5,89	100,00	0,00	5,96
09	100,00	0,00	92,46	100,00	0,00	83,15
10	93,75	2,40	823,97	93,75	2,55	832,44
11	43,75	31,30	2275,84	50,00	27,53	2265,33
12	25,00	54,92	2887,26	25,00	54,53	2857,60
Média	80,36	12,66	869,47	81,25	12,09	863,62

Como pode ser observado na Tabela 5.2, os modelos MPLIM-G e MPLIM-BG também propiciaram ao CPLEX comportamentos semelhantes para a base de dados BDNS. Ambos os modelos permitiram resolver na otimalidade todos os problemas-teste com até 9 tarefas e, aproximadamente, a mesma proporção dos demais conjuntos de problemas. Os $\overline{gap}$ ’s e os tempos médios também foram bem próximos. Essencialmente, a diferença mais sensível ocorreu com os problemas-teste com 11 tarefas. Para este conjunto, o CPLEX conseguiu resolver 50,00% dos problemas por meio do MPLIM-BG, enquanto o MPLIM-G proporcionou ao otimizador resolver apenas 43,75% dos mesmos. Os tempos médios demandados com os dois modelos foram bem próximos uns dos outros, mas o $\overline{gap}$ com o MPLIM-BG foi um pouco menor que o $\overline{gap}$ com o MPLIM-G, 27,53% contra 31,30%.

5.2.3 Terceira Bateria de Testes - BDDT

Na terceira bateria de testes utilizou-se os problemas da base de dados BDDT. Como descrito anteriormente, os problemas-teste dessa base de dados satisfazem a desigualdade triangular. Sendo assim, os três modelos de programação matemática apresentados na Seção 3 foram utilizados nesta bateria.

Os resultados encontrados para o MPLIM-G, para o MPLIM-BG e para o MPLIM-IT são resumidos na Tabela 5.3, na qual a primeira coluna indica o número de tarefas dos problemas-teste de cada conjunto. Para cada conjunto de problemas, as colunas “Soluções Ótimas” mostram (em porcentagem) em quantos problemas o CPLEX encontrou uma

solução ótima com os respectivos modelos, as colunas “ $\overline{gap}$ ” mostram as médias dos gap ’s com os respectivos modelos e as colunas “Tempo Médio” mostram as médias dos tempos (em segundos) demandados por tal otimizador na resolução dos problemas-teste com os respectivos modelos.

Tabela 5.3: Comparação MPLIM-G $\times$ MPLIM-BG $\times$ MPLIM-IT aplicados à BDDT.

# Tarefas	MPLIM-G			MPLIM-BG			MPLIM-IT		
	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)
06	100,00	0,00	0,08	100,00	0,00	0,07	100,00	0,00	3,10
07	100,00	0,00	0,88	100,00	0,00	0,87	100,00	0,00	7,15
08	100,00	0,00	8,07	100,00	0,00	9,19	100,00	0,00	11,47
09	100,00	0,00	102,79	100,00	0,00	108,71	100,00	0,00	24,21
10	93,75	2,88	1049,76	93,75	2,59	1019,47	100,00	0,00	57,82
11	43,75	32,42	2377,33	43,75	33,80	2327,45	100,00	0,00	128,14
12	12,50	59,79	3322,83	6,25	60,41	3377,30	100,00	0,00	278,23
13	18,75	66,27	2988,67	18,75	66,40	2999,26	87,50	0,58 ¹	1246,11
14	12,50	68,50	3198,65	12,50	69,40	3168,35	37,50	5,36 ²	2758,40
Média	64,58	25,54	1449,90	63,89	25,85	1445,63	91,67	0,66	501,63

¹ gap médio relativo aos 93,75% problemas-teste em que uma solução viável foi encontrada.

² gap médio relativo aos 50,00% problemas-teste em que uma solução viável foi encontrada.

Pela Tabela 5.3, observa-se que, novamente, o MPLIM-G e o MPLIM-BG apresentaram comportamentos bem semelhantes. Usando-se esses dois modelos, o CPLEX não foi capaz de resolver na otimalidade todos os problemas-teste com mais de 9 tarefas, para os quais o $\overline{gap}$ variou de 2,59% à 60,40%. Por outro lado, com o modelo MPLIM-IT, o CPLEX conseguiu resolver na otimalidade todos os problemas-teste com até 12 tarefas. Esta última formulação também exigiu, na maioria dos casos, um menor tempo médio computacional para se alcançar a otimalidade. De fato, com exceção dos três primeiros conjuntos de problemas-teste, a formulação MPLIM-IT demandou até 18 vezes, aproximadamente, menos tempo que as demais formulações.

Ainda na Tabela 5.3, nota-se que o CPLEX aplicado à formulação MPLIM-IT não foi capaz de encontrar, em uma hora de processamento, uma solução viável em 6,25% dos problemas-teste com 13 tarefas, bem como em 50,00% dos problemas-teste com 14 tarefas. Contudo, para os casos em que essa formulação conseguiu gerar uma solução viável, os $\overline{gap}$ ’s foram bem baixos (0,58% em problemas com 13 tarefas e 5,36% nos de 14 tarefas), se comparados com aqueles produzidos pelas formulações MPLIM-G e MPLIM-BG, que foram sempre superiores a 66%.

5.3 Resultados Obtidos com os Algoritmos GPV e GPV-REB

Os algoritmos heurísticos propostos (GPV e GPV-REB), bem como os procedimentos utilizados para a definição dos seus parâmetros, foram implementados em linguagem C,

usando-se o compilador Borland C++ Builder 5.0. Foram realizados experimentos com todos os problemas-teste das três bases de dados apresentadas na Seção 5.1, totalizando três baterias de testes.

Para auxiliar na apresentação e comparação dos resultados, utilizou-se duas medidas de desempenho, dadas pelas equações (5.4) e (5.5). Nestas equações, para um dado problema-teste i , f_i^* é o valor da função objetivo da melhor solução conhecida para o problema e $f_i^{X^*}$ e $\bar{f}_i^X$ são, respectivamente, o valor da função objetivo da melhor solução encontrada e a média dos valores da função objetivo aplicada nas soluções encontradas durante todas as execuções do algoritmo X para o problema-teste i .

$$imp_i^{best} = \frac{f_i^{X^*} - f_i^*}{f_i^*} \quad (5.4)$$

$$imp_i^{avg} = \frac{\bar{f}_i^X - f_i^*}{f_i^*} \quad (5.5)$$

A primeira equação calcula o desvio do valor da função objetivo da melhor solução encontrada pelo algoritmo X em torno da função objetivo da melhor solução conhecida para o respectivo problema-teste. Já a segunda equação calcula a variabilidade média dos valores da função objetivo das soluções encontradas pelo algoritmo X em torno da melhor solução conhecida para o respectivo problema.

A seguir são apresentados os procedimentos utilizados para definir cada parâmetro do GPV (utilizou-se os mesmos parâmetros no GPV-REB) e os resultados obtidos em cada bateria de testes.

5.3.1 Definição dos Parâmetros

Cada parâmetro do algoritmo GPV foi definido por meio de experimentos em um problema-teste de porte médio (30 tarefas). Definido um parâmetro, este era fixado nos procedimentos de definição dos demais parâmetros.

O primeiro parâmetro a ser definido foi o parâmetro γ da fase de construção de uma solução (ver Seção 4.5). Para isto, dado um valor para γ , foi construída uma solução com cada uma das seis funções adaptativas gulosas definidas na Seção 4.5 e a melhor destas soluções era retornada. Como o objetivo era determinar o valor de γ que proporciona construir soluções de melhor qualidade, não foi utilizado o POP durante as construções das soluções, ou seja, as soluções parciais não foram submetidas à uma busca local. Este procedimento foi realizado 20 vezes para cada um dos seguintes valores para γ : 0; 0,02; 0,04; 0,06; 0,08; 0,10; 0,12; 0,14; 0,16; 0,18; 0,20; 0,22; 0,24; 0,26; 0,28 e 0,30. Os resultados encontrados estão sintetizados no gráfico da Figura 5.1, onde é mostrado o menor valor e a média dos valores retornados para a função objetivo (fo) das soluções produzidas nas 20 execuções dos respectivos valores para γ .

Como pode ser observado no gráfico da Figura 5.1, o menor valor para fo foi encontrado para γ igual a 0,12. Já a menor média para fo foi encontrada para γ igual a 0. Desde modo, definiu-se γ igual a 0 na primeira vez em que cada uma das seis funções adaptativas gulosas é utilizada e γ igual a 0,12 nas demais.

O segundo parâmetro a ser definido foi o parâmetro $MRDmax$ da descida randômica (MRD) utilizada no POP (ver Seção 4.5). Para isto, dado um valor para $MRDmax$, foi

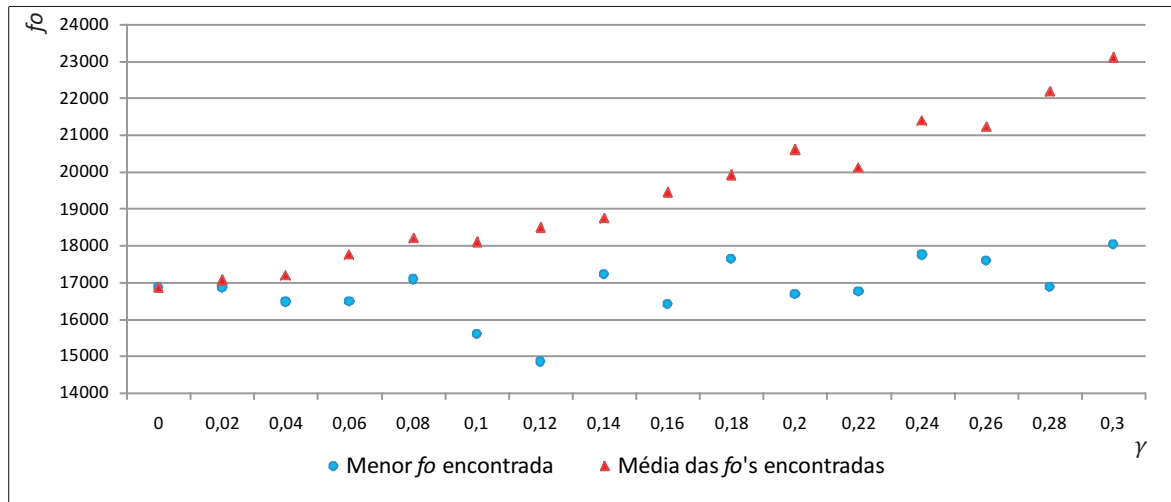


Figura 5.1: Resultados dos testes para definição de γ .

construída uma solução com cada uma das seis funções adaptativas gulosas definidas na Seção 4.5 e a melhor destas soluções era retornada. Como o objetivo era determinar o valor de MRD_{max} que proporciona encontrar soluções de melhor qualidade a partir de uma mesma solução inicial, o valor de γ foi fixado em 0. Deste modo, a ordem de inclusão de uma tarefa na solução parcial é sempre a mesma para uma dada função adaptativa gulosa. Este procedimento foi realizado 20 vezes para cada valor de MRD_{max} . Os valores de MRD_{max} testados são da forma $k \times n$, onde n representa o número de tarefas a serem sequenciadas e $k \in \{1, 2, 3, \dots, 15\}$. Os resultados encontrados estão apresentados no gráfico da Figura 5.2, onde é mostrado o menor valor e a média dos valores retornados para a função objetivo (f_0) das soluções encontradas nas 20 execuções dos respectivos valores de k .

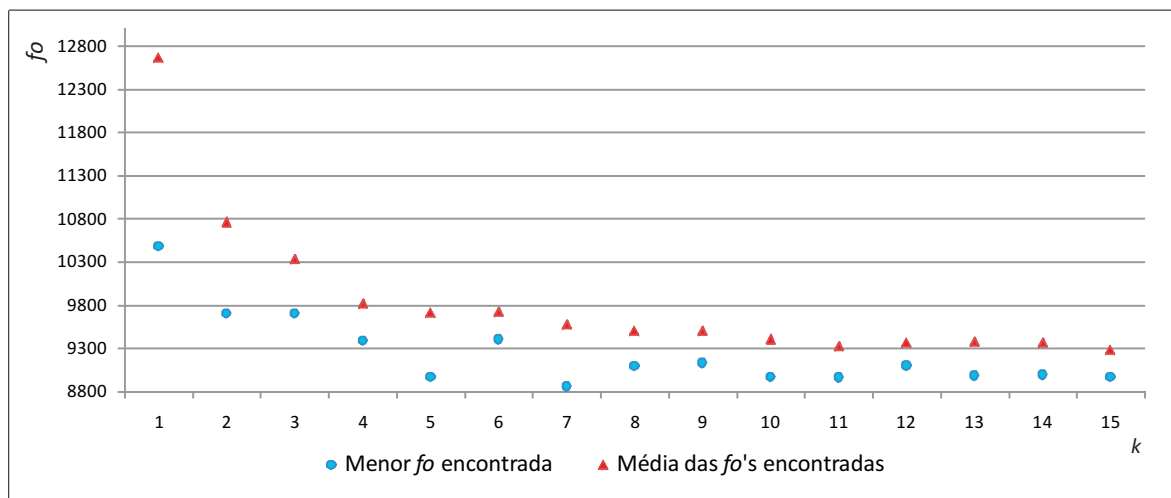


Figura 5.2: Resultados dos testes para definição de MRD_{max} .

Como pode ser observado no gráfico da Figura 5.2, o menor valor para f_0 foi encontrado

para k igual a 7, ou seja, para $MRDmax$ igual a $7 \times n$. Deste modo, definiu-se $MRDmax$ igual a $7 \times n$. Devido ao tempo computacional despendido nos problemas com 75 ou mais tarefas ($n \geq 75$), utilizou-se $MRDmax$ igual a $5 \times n$ para estes problemas.

Definidos os parâmetros da fase de construção de uma solução, o próximo parâmetro a ser definido foi o parâmetro $VNDmax$ do VND_1 (ver Seção 4.6). Para isto, dado um valor para $VNDmax$, foi construída uma solução com cada uma das seis funções adaptativas gulosas definidas na Seção 4.5 e, a seguir, cada solução construída foi submetida ao VND_1 . A melhor solução encontrada era retornada. Para construir as soluções, γ foi fixado em 0 e o $MRDmax$ foi $7 \times n$. Este procedimento foi realizado 20 vezes para cada valor de $VNDmax$. Os valores de $VNDmax$ testados também são da forma $k \times n$, onde n representa o número de tarefas a serem sequenciadas e $k \in \{1, 2, 3, \dots, 15\}$. Os resultados encontrados estão resumidos no gráfico da Figura 5.2, onde é mostrado o menor valor e a média dos valores retornados para a função objetivo (fo) das soluções encontradas nas 20 execuções dos respectivos valores de k .

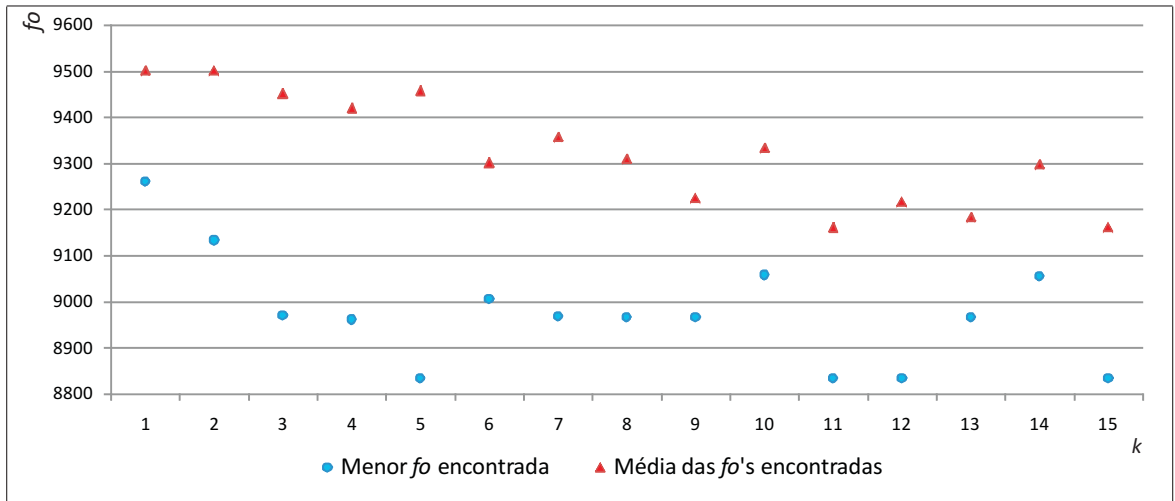


Figura 5.3: Resultados dos testes para definição de $VNDmax$.

Conforme o gráfico da Figura 5.3, o menor valor para fo foi encontrado para k igual a 5, 11, 12 e 15, ou seja, para $VNDmax$ igual a $5 \times n$, $11 \times n$, $12 \times n$ e $15 \times n$. Como a menor média para a fo foi encontrada para $VNDmax$ igual a $11 \times n$, definiu-se $VNDmax$ igual a $11 \times n$. Devido ao tempo computacional despendido para problemas com 75 ou mais tarefas ($n \geq 75$), nestes problemas utilizou-se $VNDmax$ igual a $5 \times n$.

Finalmente, o último parâmetro a ser definido foi o parâmetro $GRASPmax$ ¹ da primeira fase da metodologia proposta (ver Algoritmo 10). Para isto, dado um valor para $GRASPmax$, apenas a primeira fase do algoritmo GPV era aplicada ao problema e a melhor solução encontrada era retornada. Para construir as soluções, utilizou-se γ igual a 0 na primeira vez em que cada uma das seis funções adaptativas gulosas era utilizada e γ igual a 0,12 nas demais. $MRDmax$ e $VNDmax$ foram fixados em $7 \times n$ e $11 \times n$, respectivamente. Este procedimento foi realizado 20 vezes para cada valor de $GRASPmax$.

¹É importante lembrar que esta fase é baseada no procedimento GRASP, mas, diferentemente do GRASP tradicional, utiliza-se $GRASPmax$ iterações sem melhora na solução corrente e não simplesmente $GRASPmax$ iterações.

Os valores de $GRASP_{max}$ testados foram: 6, 12, 18, 24 e 30. Testou-se múltiplos de 6 para $GRASP_{max}$ por terem sido utilizadas seis funções adaptativas gulosas distintas para construir soluções. Os resultados encontrados estão sintetizados no gráfico da Figura 5.2, onde é mostrado o menor valor e a média dos valores retornados para a função objetivo (fo) das soluções encontradas durante as 20 execuções dos respectivos valores para $GRASP_{max}$.

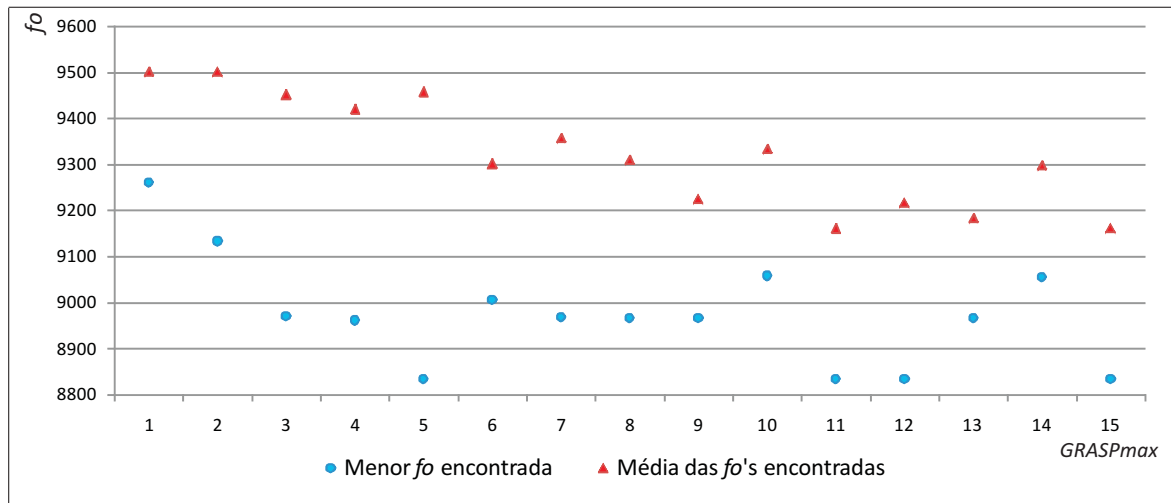


Figura 5.4: Resultados dos testes para definição de $GRASP_{max}$.

De acordo o gráfico da Figura 5.4, o menor valor para fo foi encontrado para $GRASP_{max}$ igual a 12, 18, 24 e 36. Como a menor média para a fo foi encontrada para $GRASP_{max}$ igual a 24, definiu-se $GRASP_{max}$ igual a 24. Devido ao tempo computacional despendido nos problemas com 75 ou mais tarefas ($n \geq 75$), utilizou-se $GRASP_{max}$ igual a 12 para estes problemas.

Os parâmetros utilizados estão resumidos na Tabela 5.4, onde n representa o número de tarefas a serem sequenciadas.

Tabela 5.4: Parâmetros utilizados.

Parâmetros	Valores
γ	0 e 0,12
MRD_{max}	$7 \times n$, se $n \leq 50$ e $5 \times n$, se $n \geq 75$
VND_{max}	$11 \times n$, se $n \leq 50$ e $5 \times n$, se $n \geq 75$
$GRASP_{max}$	24, se $n \leq 50$ e 12, se $n \geq 75$

5.3.2 Primeira Bateria de Testes - BDS

A primeira bateria de experimentos foi realizada com os problemas-teste da base de dados BDS. Os problemas-teste de tal base de dados possuem os tempos de *setup* simétricos. Cada problema-teste foi resolvido 30 vezes com cada algoritmo proposto, exceto aqueles com 75 tarefas, os quais foram resolvidos apenas 10 vezes em virtude do maior tempo computacional demandado. Os resultados obtidos pelo algoritmo GPV foram comparados com os alcançados em Gomes Júnior (2007), Penna (2009) e Ribeiro (2009), visto que estes autores também realizaram experimentos computacionais com esta base de dados.

Um resumo dos resultados obtidos pelo algoritmo GPV e pelos algoritmos da literatura que também fizeram experimentos com a base de dados em questão é apresentado na Tabela 5.5. Nesta tabela, a primeira coluna indica o número de tarefas nos problemas-teste de cada grupo. Nas colunas “Gomes Júnior (2007)”, “Penna (2009)”, “Ribeiro (2009)” e “GPV” são apresentados os resultados obtidos pelos respectivos algoritmos. Para um grupo de problemas-teste com i tarefas, as colunas “Tempo” mostram as médias dos tempos demandados (em segundos) pelos respectivos algoritmos, as colunas “ $\overline{imp}^{best}$ ” apresentam as médias dos imp_i^{best} s (em porcentagem) obtidos pelos respectivos algoritmos aplicados nos problemas deste conjunto e as colunas “ $\overline{imp}^{avg}$ ” apresentam as médias dos imp_i^{avg} s obtidos pelos respectivos algoritmos aplicados nos problemas-teste do conjunto.

Pela Tabela 5.5, observa-se que o algoritmo GPV e o algoritmo proposto por Gomes Júnior (2007) foram capazes de encontrar as melhores soluções conhecidas em todos os problemas-teste com até 25 tarefas. Além disso, para estes mesmos problemas, os $\overline{imp}^{avg}$ s obtidos pelo GPV foram sempre menores que os obtidos por tal autor, sendo o maior $\overline{imp}^{avg}$ apresentado pelo GPV igual a 1,53% e o maior obtido pelo algoritmo de Gomes Júnior (2007) igual a 2,32%. Já para os problemas-teste com mais de 25 tarefas, os $\overline{imp}^{best}$ s obtidos pelos dois algoritmos alternaram os menores valores e os $\overline{imp}^{avg}$ s apresentados pelo algoritmo de Gomes Júnior (2007) foram sempre superiores que os respectivos $\overline{imp}^{avg}$ s apresentados pelo GPV, sendo que, para os problemas com 75 tarefas, o $\overline{imp}^{avg}$ obtido pelo GPV foi de 13,49%, contra os 19,22% obtido pelo algoritmo deste autor. Além disso, para os problemas com mais de 30 tarefas os tempos médios demandados pelo algoritmo de Gomes Júnior (2007) foram de 30% a 74% maiores que os demandados pelo GPV, apesar dos testes terem sido realizados em máquinas diferentes.

Ainda na Tabela 5.5 é possível observar que os algoritmos propostos por Penna (2009) e Ribeiro (2009) obtiveram os menores valores para $\overline{imp}^{best}$ e $\overline{imp}^{avg}$ na maioria dos casos. Porém, para os problemas com mais de 20 tarefas, número de tarefas a partir do qual o tempo demandado passa a ser significativo, os dois algoritmos demandaram tempos médios superiores aos demandados pelo GPV nos respectivos conjuntos de problemas. Apesar de terem sido utilizadas máquinas diferentes, os tempos médios demandados pelo algoritmo de Penna (2009) e Ribeiro (2009) em tais conjuntos de problemas chegaram a ser aproximadamente 800% e 360%, respectivamente, superiores aos demandados pelo algoritmo GPV nos respectivos conjuntos de problemas.

Tabela 5.5: Comparação GPV $\times$ Gomes Júnior (2007) $\times$ Penna (2009) $\times$ Ribeiro (2009) aplicados à BDS.

# Tarefas	Gomes Júnior (2007)			Penna (2009)			Ribeiro (2009)			GPV		
	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ¹ (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ² (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ³ (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ⁴ (s)
08	0,00	0,03	0,04	0,00	0,00	0,06	0,00	0,00	0,94	0,00	0,00	0,11
09	0,00	0,06	0,07	0,00	0,00	0,09	0,00	0,15	1,26	0,00	0,00	0,17
10	0,00	0,02	0,11	0,00	0,00	0,15	0,00	0,24	1,60	0,00	0,00	0,24
11	0,00	0,12	0,20	0,00	0,00	0,25	0,00	0,03	2,21	0,00	0,06	0,46
12	0,00	0,21	0,29	0,00	0,00	0,37	0,00	0,07	2,81	0,00	0,02	0,74
15	0,00	1,47	0,94	0,00	0,47	1,13	0,00	0,76	6,02	0,00	0,51	1,47
20	0,00	1,65	4,35	0,00	0,64	4,93	0,00	0,73	20,60	0,00	1,49	5,19
25	0,00	2,32	13,29	0,00	1,09	14,90	0,00	1,02	45,72	0,00	1,53	14,22
30	0,20	3,34	40,07	0,00	1,68	39,93	0,00	1,60	112,06	0,04	2,41	30,84
40	0,44	4,66	155,79	0,08	3,32	190,61	0,00	2,41	335,88	1,10	3,89	111,52
50	1,37	7,29	492,28	0,32	4,95	630,77	0,00	4,08	896,10	1,65	6,48	309,03
75	7,89	19,22	1368,08	0,82	7,46	6308,74	0,99	7,61	2005,05	6,08	13,49	787,90
Média	0,82	3,37	172,96	0,10	1,63	599,33	0,08	1,56	285,85	0,74	2,49	105,16

¹Testes realizados em um PC Athlon XP 64 Bits 3000+ (aproximadamente 2 GHz), com 1 GB de RAM.

²Testes realizados em um PC Intel Core 2 Quad (Q6600) 2,40 GHz, com 4 GB de RAM.

³Testes realizados em um PC Pentium Core 2 Duo 2,1 GHz, com 4 GB de memória RAM.

⁴Testes realizados em um PC Intel Core 2 Duo (E8400) 2.99 GHz, com 2 GB de memória RAM.

Os resultados encontrados nos testes computacionais realizados com o algoritmo GPV-REB estão sintetizados na Tabela 5.6. Na primeira coluna é indicado o número de tarefas nos problema-teste de cada conjunto. Nas colunas “GPV” e “GPV-REB” são apresentados os resultados obtidos pelos respectivos algoritmos. Para um grupo de problemas-teste com i tarefas, as colunas “Tempo”, “ $\overline{imp}^{best}$ ” e “ $\overline{imp}^{avg}$ ” são tais como na Tabela 5.5.

Conforme pode ser observado na Tabela 5.6, para os conjuntos de problemas-teste com até 50 tarefas, o algoritmo GPV-REB obteve valores de $\overline{imp}^{best}$ menores ou iguais a 2,13%. De fato, em dois destes conjuntos o $\overline{imp}^{best}$ foi próximo de 1,60% e nos demais, menores que 1,00%. Para os problemas-teste com 75 tarefas o $\overline{imp}^{best}$ foi de 8,63%, valor próximo daquele obtido com o algoritmo GPV. Já os $\overline{imp}^{avg}$ s obtidos por meio do GPV-REB foram sempre superiores aos apresentados pelo GPV nos conjuntos correspondentes. No entanto, os valores dos $\overline{imp}^{avg}$ s obtido pelo GPV-REB aproximam-se dos obtidos com o GPV à medida em que considera-se conjuntos de problemas com mais tarefas. Finalmente, para os problemas em que o tempo demandado é mais significativo, isto é, problemas com mais de 20 tarefas, os tempos demandados pelo GPV foram de 60% a 200%, aproximadamente, superiores aos demandados pelo GPV-REB nos conjuntos de problemas correspondentes.

Tabela 5.6: Comparação GPV $\times$ GPV-REB aplicados à BDS.

# Tarefas	GPV			GPV-REB		
	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo (s)
08	0,00	0,00	0,11	0,23	0,23	0,06
09	0,00	0,00	0,17	0,00	0,00	0,13
10	0,00	0,00	0,24	0,00	0,04	0,13
11	0,00	0,06	0,46	0,15	0,19	0,37
12	0,00	0,02	0,74	0,00	0,98	0,52
15	0,00	0,51	1,47	1,48	2,53	0,90
20	0,00	1,49	5,19	0,04	3,77	3,15
25	0,00	1,53	14,22	1,57	3,86	6,96
30	0,04	2,41	30,84	0,72	4,55	19,26
40	1,10	3,89	111,52	1,77	6,73	60,36
50	1,65	6,48	309,03	2,13	8,70	151,74
75	6,08	13,49	787,90	8,63	16,34	467,71
Média	0,74	2,49	105,16	1,39	3,99	59,27

5.3.3 Segunda Bateria de Testes - BDNS

Na segunda bateria de testes utilizou-se os problemas da base de dados BDNS. Os problemas-teste dessa base de dados não possuem, necessariamente, os tempos de *setup* simétricos e satisfazendo a desigualdade triangular, sendo assim, mais genéricos. Assim como na primeira bateria de testes, cada problema foi resolvido 30 vezes, exceto aqueles com mais de 50 tarefas, os quais foram resolvidos apenas 10 vezes devido ao maior tempo computacional demandado. Os resultados obtidos pelos algoritmos GPV e GPV-REB

foram comparados com os resultados do algoritmo proposto por Ribeiro (2009), dado que este é o único autor da literatura que realizou experimentos com esta base de dados.

Uma síntese dos resultados encontrados pelos algoritmos GPV e GPV-REB e pelo algoritmo de Ribeiro (2009) é apresentada na Tabela 5.7. Nela, a primeira coluna indica o número de tarefas dos problemas de cada grupo de problemas-teste. Nas colunas “Ribeiro (2009)”, “GPV” e “GPV-REB” são apresentados os resultados obtidos pelos respectivos algoritmos. Para um grupo de problemas-teste com i tarefas, as colunas “Tempo” mostram as médias dos tempos demandados (em segundos) pelos respectivos algoritmos, as colunas “ $\overline{imp}^{best}$ ” apresentam as médias dos imp_i^{best} ’s (em porcentagem) obtidos pelos respectivos algoritmos aplicados nos problemas deste grupo e as colunas “ $\overline{imp}^{avg}$ ” apresentam as médias dos imp_i^{avg} ’s obtidos pelos respectivos algoritmos aplicados nos problemas do grupo.

Tabela 5.7: Comparação GPV $\times$ GPV-REB $\times$ Ribeiro (2009) aplicados à BDNS.

# Tarefas	Ribeiro (2009)			GPV			GPV-REB		
	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ¹ (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ² (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ² (s)
006	0,00	0,00	0,76	0,00	0,00	0,04	0,00	0,00	0,03
007	0,00	0,00	0,76	0,00	0,00	0,07	0,00	0,00	0,03
008	0,00	0,00	1,03	0,00	0,00	0,30	0,00	0,76	0,25
009	0,00	0,00	1,35	0,00	0,00	0,37	0,01	0,18	0,29
010	0,00	0,00	1,61	0,00	0,20	0,45	0,47	0,86	0,34
011	0,00	0,00	2,20	0,00	0,01	0,40	0,11	0,16	0,40
012	0,00	0,00	3,32	0,00	0,00	0,52	0,00	0,43	0,42
015	0,00	0,01	7,49	0,00	0,01	1,26	0,00	0,54	0,73
020	0,00	0,30	23,88	0,00	0,10	4,52	0,49	1,83	2,85
030	0,19	1,40	172,51	0,01	0,58	36,06	0,51	1,99	18,17
040	0,18	1,17	801,67	0,06	0,79	132,94	0,36	1,51	67,04
050	0,30	1,45	1575,11	0,06	1,28	440,49	0,65	2,57	215,60
075	5,90	5,90	4978,02	3,10	5,64	1187,76	4,14	6,66	636,86
100	2,36	2,62	22299,79	0,21	2,32	4612,79	0,57	3,79	3015,16
150	-	-	-	0,14	2,53	49524,76	0,83	2,62	32833,61
Média	0,64	0,92	2133,54	0,24	0,90	3729,52	0,54	1,59	2452,79

¹Testes realizados em um PC Pentium Core 2 Duo 2,1 GHz, com 4 GB de memória RAM.

²Testes realizados em um PC Intel Core 2 Duo (E8400) 2.99 GHz, com 2 GB de memória RAM.

De acordo com a Tabela 5.7, o GPV obteve valores de $\overline{imp}^{best}$ sempre menores ou iguais que os respectivos valores obtidos pelo algoritmo de Ribeiro (2009). Isso significa que as soluções produzidas pelo GPV foram, em média, melhores ou iguais às produzidas pelo algoritmo de tal autor. Ambos os algoritmos encontraram $\overline{imp}^{avg}$ ’s próximos de zero nos conjuntos de problemas-teste com até 20 tarefas. Já nos conjuntos de problemas com mais de 20 tarefas, os valores dos $\overline{imp}^{avg}$ ’s obtidos pelo GPV foram relativamente inferiores aos respectivos valores encontrados pelo algoritmo de Ribeiro (2009). Ademais, os tempos médios demandados pelo algoritmo de Ribeiro (2009) para resolver os problemas de cada

conjunto de problemas-teste foram sempre significativamente maiores que os demandados pelo GPV nos problemas dos respectivos conjuntos. Para o conjunto de problemas-teste com 100 tarefas, por exemplo, o tempo médio demandado pelo algoritmo proposto por Ribeiro (2009) foi aproximadamente 4,8 vezes superior ao demandado pelo GPV.

Na Tabela 5.7 também é possível observar que o algoritmo GPV obteve valores de $\overline{imp}^{best}$ menores que 1,00% para todos os conjuntos de problemas-teste, exceto para o conjunto de problemas com 75 tarefas. Mesmo assim, para este último conjunto, o $\overline{imp}^{best}$ foi menor que o obtido pelo algoritmo de Ribeiro (2009). Os $\overline{imp}^{avg}$ s foram relativamente baixos, sendo o maior valor igual a 6,66% e, para os problemas com 150 tarefas, próximo do valor obtido pelo GPV. Os tempos médios demandados também foram inferiores os demandados pelos outros dois algoritmos. Se comparado aos tempos médios demandados pelo GPV, os demandados pelo GPV-REB foram de 34% a 50% menores.

5.3.4 Terceira Bateria de Testes - BDDT

A terceira bateria de experimentos foi realizada com os problemas-teste da base de dados BDDT. Os tempos de processamento e de *setup* dos problemas-teste de tal base de dados satisfazem a desigualdade triangular. Cada problema-teste foi resolvido 30 vezes. Novamente os resultados obtidos pelos algoritmos GPV e GPV-REB foram comparados com os resultados de Ribeiro (2009), pois este último é o único autor da literatura que realizou experimentos com a base de dados em questão.

Os resultados obtidos pelos algoritmos GPV e GPV-REB e por Ribeiro (2009) estão resumidos na Tabela 5.8, onde a primeira coluna indica o número de tarefas nos problemas-teste de cada conjunto de problemas. Nas colunas “Ribeiro (2009)”, “GPV” e “GPV-REB” são apresentados os resultados obtidos pelos respectivos algoritmos. Para um grupo de problemas-teste com i tarefas, as colunas “Tempo” mostram as médias dos tempos demandados (em segundos) pelos respectivos algoritmos, as colunas “ $\overline{imp}^{best}$ ” apresentam as médias dos imp_i^{best} s (em porcentagem) obtidos pelos respectivos algoritmos aplicados nos problemas deste grupo e as colunas “ $\overline{imp}^{avg}$ ” apresentam as médias dos imp_i^{avg} s obtidos pelos respectivos algoritmos aplicados nos problemas-teste do grupo.

Pela Tabela 5.8, observa-se que o algoritmo GPV, bem como o algoritmo proposto por Ribeiro (2009), foi capaz de encontrar todas as respectivas melhores soluções conhecidas para os problemas-teste da base de dados BDDT. Apesar de o mesmo não ter ocorrido com o algoritmo GPV-REB, o maior $\overline{imp}^{best}$ obtido com este algoritmo foi de 0,35%. Os três algoritmos obtiveram valores de $\overline{imp}^{avg}$ menores que 1,00%. Em relação aos tempos médios demandados em cada conjunto de problemas-teste, novamente o algoritmo de Ribeiro (2009) demandou tempos médios maiores que os respectivos tempos médios demandados pelo algoritmo GPV, que por sua vez demandou tempos médios maiores que os despendidos pelo algoritmo GPV-REB. Tais tempos médios foram, no máximo, 3,43 segundos para o algoritmo GPV-REB, 5,26 segundos para o algoritmo GPV e 30,32 segundos para o algoritmo de Ribeiro (2009).

Dado um determinado problema-teste da base de dados BDDT, a melhor solução produzida para ele foi utilizada para determinar o conjunto $H = \{h_{inf}, h_{inf} + 1, h_{inf} + 2, h_{sup}\}$ necessário na formulação de programação matemática MPLIM-IT (ver Seção 3.3). Os valores de h_{inf} e de h_{sup} utilizados são dados, respectivamente, pela data de início da primeira tarefa e pela data de conclusão da última tarefa na melhor sequência encontrada

Tabela 5.8: Comparação GPV $\times$ GPV-REB $\times$ Ribeiro (2009) aplicados à BDDT.

# Tarefas	Ribeiro (2009)			GPV			GPV-REB		
	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ¹ (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ² (s)	$\overline{imp}^{best}$ (%)	$\overline{imp}^{avg}$ (%)	Tempo ² (s)
06	0,00	0,00	0,67	0,00	0,00	0,04	0,00	0,00	0,02
07	0,00	0,00	0,86	0,00	0,00	0,17	0,00	0,06	0,04
08	0,00	0,00	1,13	0,00	0,00	0,29	0,00	0,27	0,06
09	0,00	0,00	1,47	0,00	0,00	0,34	0,00	0,02	0,09
10	0,00	0,00	1,78	0,00	0,00	0,41	0,00	0,03	0,15
11	0,00	0,00	2,43	0,00	0,04	0,32	0,00	0,15	0,19
12	0,00	0,00	3,88	0,00	0,00	0,49	0,01	0,09	0,31
13	0,00	0,00	4,87	0,00	0,01	0,69	0,00	0,25	0,44
14	0,00	0,00	5,75	0,00	0,04	0,91	0,00	0,11	0,56
15	0,00	0,02	8,50	0,00	0,04	1,25	0,00	0,19	0,71
16	0,00	0,06	11,64	0,00	0,14	2,36	0,04	0,61	1,20
17	0,00	0,04	1,47	0,00	0,04	2,34	0,09	0,36	1,40
18	0,00	0,42	15,23	0,00	0,05	3,03	0,05	0,37	1,95
19	0,00	0,04	22,16	0,00	0,01	3,80	0,35	0,89	2,59
20	0,00	0,02	30,32	0,00	0,14	5,26	0,10	0,88	3,43
Média	0,00	0,04	7,48	0,00	0,03	1,45	0,04	0,29	0,88

¹Testes realizados em um PC Pentium Core 2 Duo 2,1 GHz, com 4 GB de memória RAM.

²Testes realizados em um PC Intel Core 2 Duo (E8400) 2.99 GHz, com 2 GB de memória RAM.

pelo algoritmo GPV para o problema. Este procedimento foi utilizado para determinar o conjunto H de cada problema-teste da base de dados e, em seguida, utilizou-se o CPLEX para resolver o MPLIM-IT relativo à cada problema.

Considerou-se o limite de 3600 segundos para a resolução de cada problema-teste pelo CPLEX e a qualidade das soluções retornadas foram medidas via a expressão (5.3) apresentada na Seção 5.2.

Os resultados encontrados são resumidos na Tabela 5.9. Nela, a primeira coluna indica o número de tarefas nos problemas de cada conjunto de problemas-teste. Para cada conjunto de problemas, a coluna “Soluções Ótimas” mostra (em porcentagem) em quantos problemas o CPLEX encontrou uma solução ótima com o modelo MPLIM-IT, a coluna “ $\overline{gap}$ ” mostra as médias dos gap ’s e a coluna “Tempo Médio” mostra as médias dos tempos demandados (em segundos) por tal otimizador na resolução dos problemas-teste com o MPLIM-IT.

Segundo a Tabela 5.9, o modelo MPLIM-IT associado ao algoritmo GPV permitiu ao CPLEX resolver na otimalidade todos os problemas-teste com até 13 tarefas. Nos conjuntos de problemas-teste com 14 e 15 tarefas, o maior $\overline{gap}$ encontrado foi igual a 1,61%.

Ainda na Tabela 5.9, observa-se que o CPLEX aplicado à formulação MPLIM-IT associada ao algoritmo GPV não foi capaz de encontrar, em uma hora de processamento, uma solução viável em 12,50% dos conjuntos de problemas-teste com 16 e 17 tarefas. Contudo, para os casos em que essa formulação conseguiu gerar uma solução viável, os

Tabela 5.9: Resultados do MPLIM-IT associado ao GPV.

# Tarefas	MPLIM-IT		
	Soluções Ótimas (%)	$\overline{gap}$ (%)	Tempo Médio (s)
06	100,00	0,00	1,93
07	100,00	0,00	5,06
08	100,00	0,00	10,44
09	100,00	0,00	16,99
10	100,00	0,00	41,70
11	100,00	0,00	105,65
12	100,00	0,00	297,89
13	100,00	0,00	795,70
14	87,50	0,50	944,55
15	68,75	1,61	1593,53
16	50,00	1,92 ¹	2563,40
17	37,50	5,86 ¹	2911,82
Média	86,98	0,82	774,05

¹ $\overline{gap}$ médio relativo aos 87,50% dos problemas-teste em que uma solução viável foi encontrada.

$\overline{gap}$'s também foram baixos (1,92% nos problemas com 16 tarefas e 5,86% nos de 17 tarefas).

5.4 Teste de Probabilidade Empírica

O comportamento do algoritmo GPV também foi comparado com os algoritmos propostos por Gomes Júnior (2007), Penna (2009) e Ribeiro (2009) via um teste de probabilidade empírica proposto por Aiex *et al.* (2000). Para realizar tal teste, escolheu-se um problema com 40 tarefas da base de dados BDS. Em seguida, determinou-se o valor da função objetivo da melhor solução conhecida para o problema como sendo o alvo a ser atingido (portanto, um alvo difícil). Cada um destes algoritmos foi executado 100 vezes no problema escolhido e cada execução foi interrompida somente quando o alvo foi atingido. Neste último momento, registrou-se o tempo despendido. Para cada algoritmo, os tempos iguais foram eliminados e o restante foi ordenado de forma crescente. Ao tempo da posição i , t_i , associou-se a probabilidade $p_i = \frac{i - 0,5}{N}$, onde N é o número de tempos não repetidos.

Os gráficos com os pontos (t_i, p_i) relativos aos algoritmos analisados são apresentados na Figura 5.5. Segundo este gráfico, os algoritmos propostos por Gomes Júnior (2007), Penna (2009) e o GPV apresentam comportamentos semelhantes para o problema-teste em questão, com vantagem dos dois primeiros, que requerem um menor tempo de processamento para obter um dado nível de probabilidade de alcançar o valor alvo. Com os algoritmos GPV e o proposto por Penna (2009), a partir de 140 segundos, aproximadamente, tem-se mais de 90% de probabilidade de atingir o alvo para o problema-teste escolhido. Já o algoritmo de Gomes Júnior (2007) possui tal probabilidade somente a

partir de 190 segundos, aproximadamente. O algoritmo proposto por Ribeiro (2009) é o mais lento entre os algoritmos analisados, pois, para qualquer tempo dado, ele possui a probabilidade de alcançar o alvo menor que os demais algoritmos.

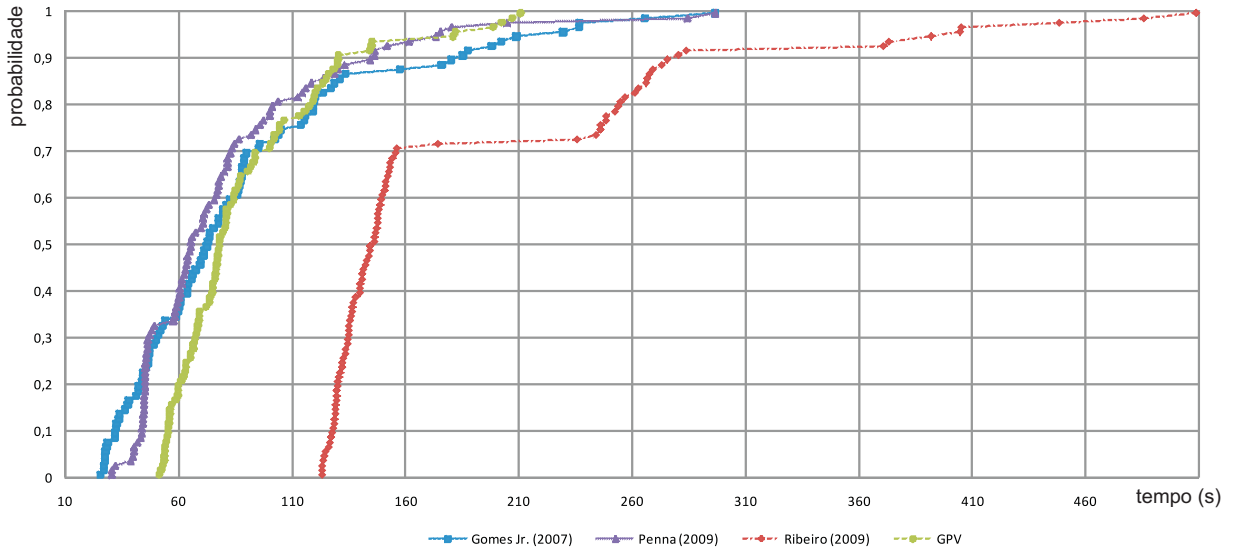


Figura 5.5: Teste de probabilidade empírica.

O teste de probabilidade empírica também foi realizado com outros dois problemas com 30 tarefas da base de dados BDS. Os resultados encontrados foram semelhantes aos apresentados na Figura 5.5.

Capítulo 6

Conclusão

Este trabalho tratou o problema de sequenciamento de tarefas em uma máquina com penalidades por antecipação e atraso da produção, considerando janelas de entrega distintas e tempo de preparação da máquina dependente da sequência, denotado por PSUMAA-JE-TPD.

Foram propostas, inicialmente, duas novas formulações de programação matemática para representar o problema. A primeira delas, denotada por MPLIM-BG, consiste em um aperfeiçoamento da formulação proposta por Gomes Júnior *et al.* (2007) (MPLIM-G), pois exige menos variáveis e restrições que a formulação de tais autores. A segunda formulação proposta, denotada por MPLIM-IT, é indexada no tempo.

Propôs-se, também, um algoritmo heurístico para determinar a melhor sequência de produção. Tal algoritmo, denominado GPV, é composto de duas fases. Na primeira, gera-se uma solução com base na metaheurística GRASP, no Princípio da Otimalidade Próxima (POP) e na Busca em Vizinhança Variável (*Variable Neighborhood Descent* - VND). Na segunda, faz-se o pós-refinamento da solução proveniente da fase anterior por meio de outro VND. Para cada sequência gerada pelo algoritmo, um procedimento de complexidade polinomial encontrado na literatura é acionado para determinar as dadas ótimas de conclusão das tarefas. Com o objetivo de reduzir o custo computacional, adicionalmente é proposta uma estratégia de redução do espaço de busca. Essa estratégia foi utilizada durante a primeira fase do GPV, originando, assim, o algoritmo denominado GPV-REB.

Visto que os problemas-teste da única base de dados (BDS) existente na literatura referente ao problema tratado possuem os tempos de preparação da máquina simétricos, foram geradas outras duas novas bases de dados. Em uma delas, BDDT, os problemas-teste envolvem até 20 tarefas e satisfazem a desigualdade triangular com relação aos tempos de preparação da máquina. Na outra, BDNS, os problemas-teste envolvem até 150 tarefas, e os tempos de preparação da máquina não satisfazem a uma propriedade particular, sendo, portanto, mais genérica.

O otimizador CPLEX 10.1 foi utilizado para resolver as três formulações matemáticas em tela, aplicadas em problemas-teste com até 14 tarefas. Os resultados computacionais mostram que as formulações MPLIM-G e MPLIM-BG proporcionam resultados semelhantes e que a formulação MPLIM-IT é mais eficiente que estas duas na resolução de problemas com até 12 tarefas, dado que esta última proporcionou ao CPLEX encontrar soluções com garantia de melhor qualidade e com menor tempo médio de processamento

que as demais. Para os problemas-teste com mais de 12 tarefas, devido ao limite de tempo estabelecido para o CPLEX retornar uma solução, os resultados obtidos não permitem uma comparação efetiva entre os três modelos.

Os algoritmos GPV e GPV-REB foram aplicados em todos os problemas-teste disponíveis, divididos em três baterias de testes. A primeira bateria foi realizada com a base de dados BDS, que é composta de problemas-teste com até 75 tarefas. Os resultados encontrados pelo GPV foram comparados com os resultados de Gomes Júnior (2007), Penna (2009) e Ribeiro (2009). O GPV apresentou desempenho superior ao algoritmo proposto por Gomes Júnior (2007), tanto em relação à qualidade das soluções encontradas, quanto em relação ao tempo computacional médio demandado. Apesar de os resultados encontrados pelo GPV serem um pouco inferiores aos encontrados pelo algoritmo de Penna (2009) e pelo algoritmo de Ribeiro (2009), os tempos computacionais médios demandados pelo GPV foram significativamente inferiores aos tempos médios demandados pelos algoritmos de tais autores.

Na segunda e na terceira bateria de testes foram utilizadas, respectivamente, as bases de dados BDNS e BDDT. Os resultados encontrados pelo GPV foram comparados com os de Ribeiro (2009), que foi o único autor a utilizar também esta base de dados. No conjunto de problemas-teste BDNS, o GPV mostrou-se mais eficiente que o algoritmo de tal autor, encontrando soluções de melhor qualidade e com tempo computacional bastante inferior. Já com a base de dados BDDT, as soluções encontradas por ambos os algoritmos foram semelhantes, porém o GPV exigiu menor tempo computacional. Ainda na terceira bateria de testes, utilizou-se o GPV para determinar o horizonte de planejamento do MPLIM-IT, o que possibilitou ao CPLEX encontrar soluções bem próximas das ótimas em todos os problemas-teste com até 15 tarefas.

Nas três baterias de testes, os tempos computacionais demandados pelo algoritmo GPV-REB foram sempre significativamente inferiores aos respectivos tempos despendidos pelo GPV. Apesar de as soluções encontradas pelo GPV-REB serem um pouco inferiores às aquelas encontradas pelo GPV, esta diferença diminui à medida em que se aumenta o número de tarefas do problema. Isto sugere que vale a pena investir em técnicas e propriedades que permitam reduzir o espaço de busca.

Enfim, conclui-se que a metodologia proposta é competitiva com aquelas existentes na literatura, tendo o baixo custo computacional como o seu grande aliado. Desta forma, foi possível resolver problemas-teste com até 150 tarefas em tempo computacional viável.

Publicações

A seguir são listados os trabalhos provenientes da presente pesquisa que foram apresentados e aceitos para publicação nos anais dos respectivos eventos.

Título: Princípio da Otimalidade Próxima e GRASP para o problema de sequenciamento de tarefas em uma máquina com penalidades por antecipação e atraso da produção.

Autores: Bruno Ferreira Rosa e Marcone Jamilson Freitas Souza.

Evento: IV Semana de Ciência & Tecnologia do CEFET-MG.

Local: Belo Horizonte/MG - Brasil.

Período: 20 a 26 de outubro de 2008.

Título: Uma nova formulação de programação matemática indexada no tempo para uma classe de problemas de sequenciamento em uma máquina.

Autores: Bruno Ferreira Rosa e Marcone Jamilson Freitas Souza.

Evento: XLI SBPO - Simpósio Brasileiro de Pesquisa Operacional.

Local: Porto Seguro/BA - Brasil.

Período: 1 a 4 de setembro de 2009.

Título: Formulações de programação matemática para o problema de sequenciamento em uma máquina com janelas de entrega distintas e tempo de preparação dependente da sequência de produção.

Autores: Bruno Ferreira Rosa, Marcone Jamilson Freitas Souza e Sérgio Ricardo de Souza.

Evento: XXXII CNMAC - Congresso Nacional de Matemática Aplicada e Computacional.

Local: Cuiabá/MT - Brasil.

Período: 8 a 11 de setembro de 2009.

Título: Princípio da Otimalidade Próxima aplicado ao problema de sequenciamento de tarefas em uma máquina com penalidades por antecipação e atraso da produção.

Autores: Bruno Ferreira Rosa, Marcone Jamilson Freitas Souza e Sérgio Ricardo de Souza.

Evento: IX SBAI - Simpósio Brasileiro de Automação Inteligente.

Local: Brasília/DF - Brasil.

Período: 20 a 23 de setembro de 2009.

Referências Bibliográficas

- Aiex, Renata M., Resende, Mauricio G. C., Ribeiro, Celso C., Celso, & Ribeiro, C. 2000. Probability Distribution Of Solution Time In Grasp: An Experimental Investigation. *Journal of Heuristics*, **8**, 200–2.
- Alidaee, B., & Dragan, I. 1997. A note on minimizing the weighted sum of tardy and early completion penalties in a single machine: A case of small common due date. *European Journal of Operational Research*, **96**, 559–563.
- Allahverdi, A., Gupta, J. N.D., & Aldowaisan, T. 1999. A review of scheduling research involving setup considerations. *Omega: The International Journal of Management Science*, **27**, 219–239.
- Almeida, M. T., & Centeno, M. 1998. A composite heuristic for the single machine early/tardy job scheduling problem. *Computers & Operations Research*, **25**(7/8), 625–635.
- Baker, K. R., & Scudder, G. D. 1990. Sequencing with Earliness and Tardiness Penalties: A Review. *Operations Research*, **38**, 22–36.
- Belfiore, P. P., da Silva, V. F., & Ronconi, D. P. 2006. Scatter Search para programação de tarefas em uma única máquina com penalidades de adiantamento e atraso e data de entrega comum. In: *Anais do XXVI Encontro Nacional de Engenharia de Produção*. Associação Brasileira de Engenharia de Produção, Fortaleza, CE, Brasil.
- Biskup, D., & Feldmann, M. 2001. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & Operations Research*, **28**, 787–801.
- Bustamante, L. M. 2007. *Minimização do custo de antecipação e atraso para o problema de sequenciamento de uma máquina com tempo de preparação dependente da sequência: aplicação em uma usina siderúrgica*. M.Phil. thesis, UFMG, Belo Horizonte.
- Chang, P. C. 1999. A Branch and Bound Approach for Single Machine Scheduling with Earliness and Tardiness Penalties. *Computers & Mathematics with Applications*, **37**, 133–144.
- Christofoletti, L. M. 2002. *Métodos de Reinício Aplicados ao Sequenciamento em Uma Máquina com Tempos de Preparação e Datas de Entrega*. M.Phil. thesis, UNICAMP, Campinas.

- de Paula, Mateus Rocha. 2008. *Heurísticas para a minimização dos atrasos em seqüenciamento de máquinas paralelas com tempos de preparação dependentes da seqüência*. M.Phil. thesis, UFMG, Belo Horizonte.
- Feldmann, M., & Biskup, D. 2003. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers & Industrial Engineering*, **44**, 307–323.
- Feo, T. A., & Resende, M. G. C. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, **6**, 109–133.
- França Filho, M. F. 2007. *GRASP e Busca Tabu aplicados a problemas de programação de tarefas em máquinas paralelas*. Ph.D. thesis, UNICAMP, Campinas.
- Gendreau, M. 2003. An Introduction to Tabu Search. 37-54 2 of: Glover, F., & Kochenberger, G. A. (eds), *Handbook of Metaheuristics*. Kluwer Academic Publishers.
- Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, **13**(5), 533–549.
- Glover, F. 1996. *Tabu Search and adaptive memory programming - advances, applications and challenges*. Interfaces in computer science and operations research edn. Kluwer Academic Publishers. Pages 1–75.
- Glover, F., & Laguna, M. 1997. *Tabu Search*. Boston: Kluwer Academic Publishers.
- Glover, F., & Laguna, M. 2003. Scatter Search and Path Relinking: Advances and Applications. Chap. 1, pages 1–35 of: Glover, F., & Kochenberger, G. A. (eds), *Handbook of Metaheuristics*. Kluwer Academic Publishers.
- Gomes Júnior, A. C. 2007. *Problema de Seqüenciamento em uma Máquina com Penalidades por Antecipação e Atraso: Modelagem e Resolução*. M.Phil. thesis, UFMG, Belo Horizonte.
- Gomes Júnior, A. C., Carvalho, C. R. V., Munhoz, P. L. A., & Souza, M. J. F. 2007. Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. Pages 1649–1660 of: *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional*. SOBRAPO.
- Gray, P., Hart, W., Painton, L., Phillips, C., Trahan, M., & Wagner, J. 1997. *A Survey of Global Optimization Methods*. Tech. rept. Sandia National Laboratories. Disponível em < <http://www.cs.sandia.gov/opt/survey> >. Acesso em: 27 nov. 2008.
- Gupta, S. R., & Smith, J. S. 2006. Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, **175**, 722–739.
- Hansen, P., & Mladenović, N. 2003. Variable Neighborhood Search. 145-184 6 of: Glover, F., & Kochenberger, G. A. (eds), *Handbook of Metaheuristics*. Kluwer Academic Publishers.

- Henderson, D., Jacobson, S. H., & Johnson, A. W. 2003. The Theory and Practice of Simulated Annealing. *Chap. 10, pages 287–319 of: Glover, F., & Kochenberger, G. A. (eds), Handbook of Metaheuristics*. Kluwer Academic Publishers.
- Hino, C. M., Ronconi, D. P., & Mendes, A. B. 2005. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, **160**, 190–201.
- James, R. J. W. 1997. Using tabu search to solve the common due date early/tardy machine scheduling problem. *Computers & Operations Research*, **24**(3), 199–208.
- Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. 1983. Optimization by simulated annealing. *Science*, **220**, 671–680.
- Koulamas, C. 1996. Single-machine scheduling with time windows and earliness/tardiness penalties. *European Journal of Operational Research*, **91**, 190–202.
- Lee, C. Y., & Choi, J. Y. 1995. A Genetic Algorithm for Job Sequencing Problems with Distinct Due Dates and General Early-Tardy Penalty Weights. *Computers and Operations Research*, **22**(8), 857–869.
- Lee, C. Y., & Kim, S. J. 1995. Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights. *Computers & industrial engineering*, **28**(2), 231–243.
- Li, G. 1997. Single machine earliness and tardiness scheduling. *European Journal of Operational Research*, **96**, 546–558.
- Liaw, C.-F. 1999. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers & Operations Research*, **26**, 679–693.
- Lourenço, H. R., Martin, O. C., & Stützle, T. 2003. Iterated Local Search. *Chap. 11, pages 321–353 of: Glover, F., & Kochenberger, G. A. (eds), Handbook of Metaheuristics*. Kluwer Academic Publishers.
- Mazzini, R., & Armentano, V. A. 2001. A heuristic for single machine scheduling with early and tardy costs. *European Journal of Operational Research*, **128**, 129–146.
- M'Hallah, R. 2007. Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. *Computers & Operations Research*, **34**, 3126–3142.
- Mladenović, N., & Hansen, P. 1997. Variable Neighborhood Search. *Computers and Operations Research*, **24**(11), 1097–1100.
- Mondal, S. A. 2002. Minimization of squared deviation of completion times about a common due date. *Computers & Operations Research*, **29**, 2073–2085.
- Mondal, S. A., & Sen, A. K. 2001. Single machine weighted earliness tardiness penalty problem with a common due date. *Computers & Operations Research*, **28**, 649–669.

- Panwalkar, S. S., Dudek, R. A., & Smith, M. L. 1973. Sequencing research and the industrial scheduling problem. *Pages 29–38 of: Elmaghraby, S. E. (ed), Symposium on the Theory of Scheduling and its Applications.*
- Penna, Puca Huachi Vaz. 2009. *Um algoritmo heurístico híbrido para minimizar os custos com a antecipação e o atraso da produção em ambientes com janelas de entrega e tempos de preparação dependentes da sequência.* M.Phil. thesis, UFOP, Ouro Preto.
- Rabadi, G., Mollaghasemi, M., & Anagnostopoulos, G. C. 2004. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers & Operations Research*, **31**, 1727–1751.
- Reeves, C. 2003. Genetic Algorithms. *Chap. 3, pages 55–82 of: Glover, F., & Kochenberger, G. A. (eds), Handbook of Metaheuristics.* Kluwer Academic Publishers.
- Resende, M. G.C., & Ribeiro, C. C. 2003. Greedy Randomized Adaptive Search Procedures. *219–249 of: Glover, F., & Kochenberger, G. A. (eds), Handbook of Metaheuristics.* Kluwer Academic Publishers.
- Ribeiro, C. C. 1996. Metaheuristics and Applications. In *Advanced School on Artificial Intelligence*, Estoril, Portugal.
- Ribeiro, Fábio Fernandes. 2009. *Um algoritmo genético adaptativo para a resolução do problema de sequenciamento em uma máquina com penalização por antecipação e atraso da produção.* M.Phil. thesis, CEFET-MG, Belo Horizonte.
- Sourd, F. 2005. Optimal timing of a sequence of tasks with general completion costs. *European Journal of Operational Research*, **165**, 82–96.
- Souza, M. J. F. 2008. *Inteligência Computacional para Otimização.* Notas de aula. Disponível em < [http : //www.decom.ufop.br/prof/marcone/ Disciplinas/InteligenciaComputacional/InteligenciaComputacional.htm](http://www.decom.ufop.br/prof/marcone/ Disciplinas/InteligenciaComputacional/InteligenciaComputacional.htm) >. Acesso em: 20 nov. 2008.
- Souza, M. J. F., Penna, P. H. V., Gonçalves, F. A. C. A., & Ochi, L. S. 2008. Uma heurística híbrida para minimizar custos com antecipação e atraso em sistemas de produção com janelas de entrega e tempos de preparação dependentes da sequência. *Pages 1–16 of: Anais do XI Simpósio de Pesquisa Operacional e Logística da Marinha.*
- Szwarc, W. 1996. The weighted common due date single machine scheduling problem revisited. *Computers and Operations Research*, **23**(3), 255–262.
- Tavakkoli-Moghaddam, R., Moslehi, G., Vasei, M., & Azaron, A. 2006. A branch-and-bound algorithm for a single machine sequencing to minimize the sum of maximum earliness and tardiness with idle insert. *Applied Mathematics and Computation*, **174**, 388–408.
- Valente, J. M.S., & Alves, R. A. F. S. 2005a. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. *Computers & Industrial Engineering*, **48**, 363–375.

- Valente, J. M.S., & Alves, R. A. F. S. 2005b. Improved heuristics for the early/tardy scheduling problem with no idle time. *Computers & Operations Research*, **32**, 557–569.
- Ventura, J. A., & Radhakrishnan, S. 2003. Single machine scheduling with symmetric earliness and tardiness penalties. *European Journal of Operational Research*, **144**, 598–612.
- Wan, G., & Yen, B. P.-C. 2002. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, **142**, 271–281.
- Weise, T. 2008. *Global Optimization Algorithms: Theory and Application*. 2ª edição edn. Disponível em: < [http : //www.it – weise.de/](http://www.it-weise.de/) >. Acesso em: 01 dez. 2008.
- Weng, X., & Ventura, J. A. 1996. Scheduling about a given common due date to minimize mean squared deviation of completion times. *European Journal of Operational Research*, **88**, 328–335.
- Ying, K.-C. 2008. Minimizing earliness-tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm. *Computers & Industrial Engineering*, doi:10.1016/j.cie.2008.01.008.
- Ziviane, N. 2004. *Projeto de Algoritmos com implementações em PASCAL e C*. 2ª edição revista e ampliada edn. Thomson.

**UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO**

**Programação integrada de veículos e tripulações de ônibus
urbano**

Marcone Jamilson Freitas Souza /DECOM/UFOP - Coordenador

Anexo do Relatório Técnico Científico Final
apresentado ao CNPq, referente ao processo
CNPq, 474831/2007-8, desenvolvido no
período dezembro de 2007 a dezembro de
2009.

Ouro Preto – Minas Gerais - Brasil
Janeiro de 2010

Problema de programação integrada de veículos e tripulações de ônibus urbano

RESUMO

Este trabalho trata da resolução integrada de duas etapas do planejamento de transportes no Sistema de Transporte Público, no caso, a programação de veículos (PPV) e a programação de tripulações (PPT). O primeiro problema consiste em criar uma rotina diária de operação para uma frota de veículos disponibilizada para uso de uma empresa, a partir de um conjunto de viagens com seus respectivos horários, pontos de partida e chegada, e tempo de viagem. O objetivo do PPV é reduzir o custo operacional de tal atividade, fazendo o melhor aproveitamento da frota. O PPT recebe como dados de entrada a programação dos veículos e retorna como resultado um conjunto de jornadas de trabalho para as tripulações (motoristas e trocadores). Estes problemas são da classe NP-difícil e, assim, ainda não há algoritmos em tempo polinomial para encontrar soluções ótimas. Aqui, estes problemas são tratados conjuntamente e alterações realizadas na programação de veículos têm reflexo na programação das tripulações. Dada a natureza combinatória dos problemas considerados e, em particular, do problema integrado, este é resolvido por meio de métodos heurísticos. Neste trabalho, foi desenvolvido um método híbrido, baseado nas metaheurísticas *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e Busca Tabu (BT). Para testar o algoritmo proposto são utilizados dados reais das empresas do Sistema de Transporte Público da cidade de Belo Horizonte (MG), que atuam em uma bacia de linhas de tal cidade. Duas funções de avaliação foram consideradas: uma baseada em pesos empíricos e outra, em custos. Os resultados obtidos são comparados com aqueles produzidos por uma metodologia clássica de solução dessas duas etapas de solução do PPVT, que consiste em resolver os problemas PPV e PPT de forma independente e seqüencial. Os experimentos computacionais realizados mostram que o algoritmo proposto é superior, além de robusto.

Índice

LISTA DE FIGURAS	V
LISTA DE TABELAS.....	VI
1 INTRODUÇÃO	1
1.1 O PROBLEMA DE PROGRAMAÇÃO DE VEÍCULOS E TRIPULAÇÕES	1
1.2 ESTRUTURA DO TRABALHO	3
2 OBJETIVOS DO TRABALHO	4
3 REVISÃO BIBLIOGRÁFICA.....	5
3.1 INTRODUÇÃO.....	5
3.2 HEURÍSTICAS.....	8
3.2.1 Construção Gulosa.....	9
3.2.2 Método de Descida.....	9
3.2.3 Método Descida Randômica	10
3.3 METAHEURÍSTICAS.....	11
3.3.1 Iterated Local Search	11
3.3.2 Variable Neighborhood Descent.....	12
3.3.3 Busca Tabu.....	13
3.4 MÉTODO DE INTENSIFICAÇÃO	14
3.4.1 Relaxação Adaptativa	14
4 O PROBLEMA DE PROGRAMAÇÃO DE VEÍCULOS E TRIPULAÇÕES ABORDADO.....	16
4.1 RESTRIÇÕES E OBJETIVOS DA PROGRAMAÇÃO DE VEÍCULOS	18
4.2 RESTRIÇÕES E OBJETIVOS DA PROGRAMAÇÃO DE TRIPULAÇÕES	19
5 METODOLOGIA	20
5.1 REPRESENTAÇÃO DO PPVT.....	20
5.2 ESTRUTURAS DE VIZINHANÇA	21
5.2.1 Vizinhaça N_v^R	21
5.2.2 Vizinhaça N_v^T	21
5.2.3 Vizinhaça N_t^R	22
5.2.4 Vizinhaça N_t^T	22
5.2.5 Vizinhaça N_v^{RP}	22
5.2.6 Vizinhaça N_t^{TP}	23
5.3 AVALIAÇÃO DE UMA SOLUÇÃO - 1ª ABORDAGEM	23
5.3.1 Avaliação da programação de veículos	23
5.3.2 Avaliação da programação de tripulações	26
5.4 AVALIAÇÃO DE UMA SOLUÇÃO - 2ª ABORDAGEM	28
5.4.1 Avaliação dos veículos	28
5.4.2 Avaliação das tripulações	29
5.4.3 Definindo os pesos da avaliação.....	30
5.5 GERAÇÃO DE UMA SOLUÇÃO INICIAL PARA O PPV	31
5.5.1 Método Guloso aplicado à geração de uma solução inicial para o PPV.....	31
5.5.2 Método Guloso aplicado à geração de uma solução inicial para o PPT.....	35
5.6 METAHEURÍSTICAS APLICADAS AO PPVT	38

5.6.1	<i>Iterated Local Search</i>	38
5.6.2	<i>Variable Neighborhood Descent</i>	39
5.6.3	<i>Busca Tabu com Relaxação Adaptativa</i>	40
5.6.4	<i>O procedimento Vehicle Random Descent</i>	41
5.6.5	<i>O procedimento Integrated Vehicle and Crew Random Descent Procedure...</i>	42
6	RESULTADOS	44
6.1	RESULTADOS – 1ª ABORDAGEM	44
6.2	RESULTADOS – 2ª ABORDAGEM	48
7	CONCLUSÕES	51
	REFERÊNCIAS BIBLIOGRÁFICAS	52

Lista de Figuras

Figura 1.1 – Relação entre as diversas operações no Sistema de Transporte Público	1
Figura 3.1 – Pseudocódigo da Construção Gulosa.....	9
Figura 3.2 – Pseudocódigo do Método da Descida.....	10
Figura 3.3 – Pseudocódigo do Método Descida Randômica	10
Figura 3.4 – Pseudocódigo do algoritmo <i>Iterated Local Search</i>	12
Figura 3.5 – Algoritmo VND	13
Figura 3.6 - Algoritmo Busca Tabu	14
Figura 4.1 - Representação esquemática das viagens da Tabela 4.1.....	17
Figura 4.2 - Representação esquemática de uma lista de tarefas	18
Figura 5.1 – Exemplo de uma solução s para o PPVT	20
Figura 5.2 – Método Gerador de Solução Inicial para os Veículos	32
Figura 5.3 –Exemplo da geração de uma solução inicial para os veículos: Passo 0.....	32
Figura 5.4 – Exemplo da geração de uma solução inicial para os veículos: Passo 1.....	33
Figura 5.5 – Exemplo da geração de uma solução inicial para os veículos: Passo 2.....	33
Figura 5.6 – Exemplo da geração de uma solução inicial para os veículos: Passo 3.....	33
Figura 5.7 – Exemplo da geração de uma solução inicial para os veículos: Passo 4.....	33
Figura 5.8 – Exemplo da geração de uma solução inicial para os veículos: Passo 5.....	34
Figura 5.9 – Exemplo da geração de uma solução inicial para os veículos: Passo 6.....	34
Figura 5.10 – Exemplo da geração de uma solução inicial para os veículos: Passo 7.....	34
Figura 5.11 – Exemplo da geração de uma solução inicial para os veículos: Passo 8.....	34
Figura 5.12 – Exemplo da geração de uma solução inicial para os veículos: Passo 9.....	35
Figura 5.13 – Método Gerador de Solução Inicial para as Tripulações.....	35
Figura 5.14 – Tarefas geradas pela programação dos veículos da Figura 5.14.....	36
Figura 5.15 – Exemplo da geração de uma solução inicial para as tripulações: Passo 1	36
Figura 5.16 – Exemplo da geração de uma solução inicial para as tripulações: Passo 2	36
Figura 5.17 – Exemplo da geração de uma solução inicial para as tripulações: Passo 3.....	37
Figura 5.18 – Exemplo da geração de uma solução inicial para as tripulações: Passo 4.....	37
Figura 5.19 – Exemplo da geração de uma solução inicial para as tripulações: Passo 5.....	37
Figura 5.20 – Exemplo da geração de uma solução inicial para as tripulações: Passo 6.....	37
Figura 5.21 – Algoritmo ILS com VND_2N aplicado ao PPVT	39
Figura 5.22 – Algoritmo VND em dois níveis como refinamento do ILS.....	40
Figura 5.23 – Algoritmo Busca Tabu com Relaxação Adaptativa aplicado ao PPVT.....	41
Figura 5.24 – Procedimento <i>Vehicle Random Descent</i>	42
Figura 5.25 – Procedimento <i>Integrated Vehicle and Crew Random Descent</i>	43

Lista de Tabelas

Tabela 4.1 – Exemplo de uma tabela de viagens	16
Tabela 5.1 – Dados usados no calculo dos pesos – 2ª Abordagem	30
Tabela 5.2 – Pesos da função de avaliação – 2ª Abordagem	31
Tabela 6.1 – Parâmetros utilizados na função de avaliação do PPVT	45
Tabela 6.2 – Instâncias e seus respectivos tempos limites relativos ao PPV	45
Tabela 6.3 – Comparação de resultados (Seqüencial x Integrada)	46
Tabela 6.4 – Características das melhores soluções das metodologias seqüencial e integrada para a empresa G02	47
Tabela 6.5 – Características das melhores soluções das metodologias seqüencial e integrada para a empresa G27	48
Tabela 6.6 – Comparação de resultados dos algoritmos seqüencial e integrado com função de avaliação baseada em custos	49
Tabela 6.7 – Características das melhores soluções dos algoritmos seqüencial e integrado com função de avaliação baseada em custos.....	50

1 Introdução

1.1 O Problema de Programação de Veículos e Tripulações

A Figura 1 representa as relações existentes entre os principais problemas que surgem no processo de planejamento do Sistema de Transporte Público.

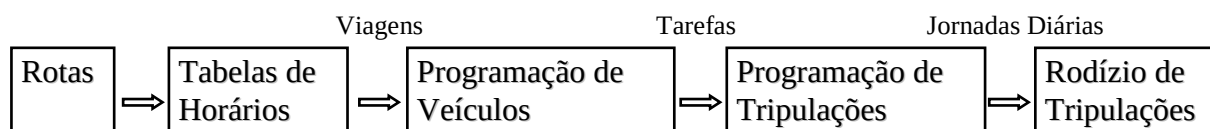


Figura 1.1 – Relação entre as diversas operações no Sistema de Transporte Público

A definição das rotas (linhas) e das tabelas de horários do sistema se baseia na disponibilidade de infra-estrutura, nos serviços requeridos pelos usuários e em aspectos de demanda. Tais decisões não fazem parte da abrangência deste projeto e são consideradas como conhecidas. As viagens de uma mesma rota podem ter suas durações alteradas ao longo do período de planejamento, entretanto, suas durações são consideradas também como conhecidas de forma determinística. As duas próximas etapas do processo são as programações dos veículos e das tripulações, que consistem em atribuir os veículos às viagens e as tripulações aos veículos, respectivamente.

Conhecidas as viagens a serem executadas pela empresa concessionária do serviço de transporte público, o problema seguinte é o da “Programação dos Veículos”. Neste, o objetivo é fazer o dimensionamento da frota e a alocação dos veículos às viagens, de forma que todas as viagens programadas para as diversas linhas da empresa sejam executadas. Mais precisamente, o Problema de Programação de Veículos (PPV) pode ser definido da seguinte forma: dado um conjunto de viagens com seus respectivos horários e pontos de partida e chegada, e dado o tempo de viagem entre todos os pares de pontos, encontrar a programação de veículos de custo mínimo tal que: 1) cada viagem seja executada por um único veículo, e 2) a cada veículo seja atribuída uma seqüência exequível de viagens. Usualmente, a função de custo é uma combinação dos custos fixos (custo do veículo) e custos variáveis (tempo de viagem fora de operação e tempo parado no terminal). A programação dos veículos resulta em *blocos de viagens*, onde cada bloco consiste na seqüência de viagens realizadas por um veículo entre uma partida e o seu retorno à garagem.

Na abordagem seqüencial assume-se conhecida a programação dos veículos para resolver a programação de tripulações, ou seja, são conhecidos os blocos de viagens dos veículos. Nesta etapa, cada bloco de viagens de um veículo é subdividido em *oportunidades de troca*, que são definidas através de um ponto (possível local de troca) e um intervalo de tempo, que juntos possibilitam que ocorra a troca da tripulação em operação por uma nova tripulação. Com base nas oportunidades de troca, as viagens são agrupadas em *tarefas*. Uma *tarefa* é definida como um conjunto de viagens compreendido entre duas oportunidades de troca consecutivas, e corresponde à menor porção de trabalho que pode ser designada a uma tripulação. Com base nessas definições, o Problema de Programação de Tripulações (PPT) pode ser definido da seguinte forma: dado um conjunto de tarefas, encontrar um conjunto de jornadas diárias com custo mínimo tal que 1) cada tarefa seja atribuída a uma única jornada, e

2) cada jornada seja uma seqüência de tarefas que pode ser executada por uma única tripulação respeitando as restrições operacionais e trabalhistas. Dentre essas restrições, destacamos: a) a duração da jornada diária de trabalho não pode superar um valor máximo; b) durante a jornada de trabalho deve haver um intervalo de tempo para descanso e alimentação; (c) as trocas de tripulações só podem ocorrer em determinados locais (pontos). Usualmente, a função de custo do PPT considera a minimização do número de jornadas, da quantidade de horas-extras existentes nas jornadas e da quantidade de trocas de tripulações, linhas e veículos. Para detalhes sobre a modelagem e a resolução de problemas desta natureza no âmbito nacional ver Silva *et al.* (2002, 2004), Souza *et al.* (2003a, 2003b, 2004a, 2004b).

Uma vez geradas as jornadas diárias de dias úteis, sábados, domingos e feriados, essas devem ser, agora, combinadas, de forma a fazer a escala mensal das tripulações ou Rodízio das Tripulações, problema conhecido na literatura inglesa como *Crew rostering*. Na elaboração da escala mensal, observa-se que uma parte da legislação trabalhista, dos acordos coletivos e das regras operacionais foi contemplada na confecção das jornadas diárias. Entretanto, outras tantas devem ser observadas, como por exemplo: (a) A cada 6 (seis) dias de trabalho o tripulante tem direito a uma folga; (b) Tripulantes que fazem dupla pegada têm folga obrigatoriamente aos sábados ou domingos. Em uma escala mensal procura-se, dentre outros objetivos: (a) minimizar, ou mesmo eliminar, as mudanças de horários de trabalho de uma tripulação; (b) equilibrar as horas mensais trabalhadas pelas tripulações; (c) reduzir as mudanças de linhas de uma tripulação ao longo do mês. Experiências com a modelagem e resolução desse problema considerando o cenário nacional são relatadas em Toffolo *et al.* (2005).

Pela Figura 1 observa-se que a solução de um problema de uma etapa do processo de planejamento do Sistema de Transporte Público depende da solução do problema da etapa anterior. Assim, na abordagem seqüencial de resolução dos problemas, se fosse possível garantir a solução ótima para o problema de cada etapa, não haveria garantia de que essa solução fosse ótima para o problema como um todo. O planejamento poderia ser mais eficiente se fosse adotada uma abordagem integrada de resolução dos problemas. Entretanto, devido à complexidade dos modelos matemáticos que consideram o problema como um todo, a grande maioria dos estudos teóricos trata os problemas de forma seqüencial.

Mais recentemente têm surgido na literatura abordagens que lidam com a integração de duas etapas consecutivas do processo de planejamento, a programação dos veículos e a programação de tripulações.

Embora haja um entendimento de que a abordagem integrada da programação de veículos e tripulações seja vantajosa, a maioria dos trabalhos encontrados na literatura segue a abordagem seqüencial, na qual a programação de veículos é definida a priori e independentemente da programação de tripulações.

O Problema de Programação Integrada de Veículos e Tripulações (PPVT), objeto de estudo neste trabalho, pode ser definido da seguinte maneira: dada uma tabela de horários de viagens a serem realizadas, encontrar a programação de custo mínimo para os veículos e as tripulações tal que ambas sejam factíveis e mutuamente compatíveis.

Os problemas dos veículos e das tripulações interagem entre si, uma vez que a especificação da programação dos veículos impõe certas restrições à programação das tripulações e vice-versa. Assim, a abordagem integrada do Problema de Programação de Veículos e de Tripulações sempre permitirá obter resultados superiores ou iguais aos resultados da abordagem seqüencial (Freling *et al.* 2003).

1.2 Estrutura do trabalho

Este trabalho está estruturado em sete capítulos, incluindo esta introdução.

Os objetivos do projeto estão apresentados no capítulo 2.

No capítulo 3 é apresentada uma revisão bibliográfica sobre alguns trabalhos desenvolvidos e suas técnicas utilizadas para resolver o PPV e o PPT. São descritos também os princípios básicos das heurísticas e metaheurísticas empregadas neste trabalho.

O capítulo 4 descreve o Problema de Programação de Veículos e de Tripulações segundo as políticas operacionais que regem o Sistema de Transporte Público da cidade de Belo Horizonte, objeto de estudo neste projeto.

A metodologia adotada neste trabalho é apresentada no capítulo 5. Aborda-se, em tal capítulo, a forma de representação de uma solução, os tipos de movimentos que constituem as estruturas de vizinhança, a forma de avaliação de uma solução e o algoritmo de exploração do espaço de busca.

No capítulo 6 são apresentados os resultados obtidos.

No capítulo 7 são apresentadas as conclusões.

2 Objetivos do Trabalho

Este trabalho tem como objetivo o desenvolvimento de um algoritmo heurístico, baseado nas metaheurísticas *Iterated Local Search*, Descida em Vizinhança Variável e Busca Tabu, para resolver de forma eficiente o Problema de Programação Integrada de Veículos e Tripulações de Ônibus Urbano.

Objetiva-se, também, comparar os resultados obtidos com o algoritmo proposto com aqueles produzidos por uma metodologia clássica de solução dessas duas fases do planejamento de transporte. Isto é, com um algoritmo sequencial tradicional, o qual inicialmente resolve a programação de veículos e depois, com a solução desta, resolve o problema de programação de tripulações.

3 Revisão Bibliográfica

3.1 Introdução

Esta seção apresenta uma revisão de literatura do Problema de Programação de Veículos e, em seguida, do Problema de Programação de Tripulações; posteriormente, faz uma revisão da classificação de Freling *et al.* 1999 para o PPVT e da integração entre esses dois problemas.

O Problema de Programação de Veículos foi um dos primeiros problemas de transporte público a serem estudados e resolvidos com o auxílio de computadores. Os primeiros trabalhos tratavam o problema tanto utilizando métodos exatos (Kirkman, 1968) quanto métodos heurísticos (Saha, 1970; Wren, 1972).

Martin-löf (1970) resolve o problema de programação de veículos considerando viagens com flexibilidade nos horários de partida. É apresentado um modelo de programação linear inteira, o qual é resolvido por um método simples de busca do tipo *branch-and-bound*. O método, entretanto, tem aplicabilidade limitada a problemas de pequeno porte.

Gavish *et al.* (1978) formulam o PPV como um problema de designação e pseudo-designação, que tem como objetivos primários minimizar o número de veículos utilizados durante o horário de pico e minimizar o tempo de deslocamento fora de operação no horário entre picos. O objetivo secundário é minimizar as mudanças em relação à programação já existente.

Hoffstadt (1981) utiliza uma heurística de encadeamento de viagens para determinar o número mínimo de veículos. Posteriormente, para minimizar o custo operacional é aplicado o algoritmo húngaro (Kuhn, 1956) a um problema de designação.

Carraresi e Gallo (1984) tratam o problema de programação de veículos de natureza periódica, isto é, após um determinado período, por exemplo, 24 horas, todas as viagens devem ser executadas novamente. Dessa forma, são praticadas apenas três programações diferentes: uma para os dias úteis, uma para os sábados e uma para os domingos e feriados.

Wren e Chamberlain (1988) incorporam a programação da tripulação ao sistema para programação de veículos, denominado VAMPIRES, de Wren (1972), que passa então a ser denominado BUSMAN, sendo a componente responsável pela programação de veículos denominada BUSPLAN.

Kwan e Rahin (1995) apresentam os módulos incorporados à componente BUSPLAN do sistema, os quais permitem estabelecer certa interatividade com o usuário. São apresentadas ferramentas semi-automáticas para coordenar a geração de novos horários das viagens com ligação infactível, além de melhoramentos nas heurísticas que tratam da programação com diferentes tipos de veículos.

Kwan e Rahin (1999) apresentam as características do BOOST, que é a versão orientada a objetos do sistema anterior, o BUSPLAN. Nesse sistema, uma solução inicial é gerada por meio de um método construtivo guloso que admite infactibilidade. A seguir, a programação é refinada por um procedimento iterativo 2-*optimal*.

Psarras *et al.* (1997) utilizam as vantagens da *Constraint Logic Programming*, para gerar uma solução inicial para o Problema de Programação de Veículos e posteriormente

avaliar as várias soluções intermediárias obtidas por uma busca local com o objetivo de verificar a conformidade com as restrições impostas ao problema. Segundo os autores, a grande vantagem desta abordagem é a obtenção de soluções em um tempo de execução satisfatório.

Em Atkinson (1998) uma heurística GRASP é aplicada a um PPV em que os intervalos de desembarque e embarque de um veículo são reduzidos, além de outras restrições adicionais. Duas formas adaptativas de busca são propostas, uma global e outra local. Em ambas, o cálculo da função gulosa é modificado por uma quantidade que mensura heurísticamente a qualidade da solução parcial que é obtida quando uma decisão é tomada. A busca global é um procedimento adaptativo tratado como uma regra de aprendizado, uma vez que ele tenta aprender com os erros anteriores fazendo uma atualização da função gulosa a cada iteração da heurística.

Baita *et al.* (2000) abordam um caso prático do PPV comparando o desempenho do modelo tradicional de designação com duas novas heurísticas propostas para o problema. O modelo de designação é resolvido com o algoritmo húngaro descrito por Carpaneto *et al.* (1988). Neste modelo, a estrutura da função de custo considera os seguintes elementos: viagens mortas entre os terminais, mudanças de linhas e tempo de espera nos terminais. A primeira heurística proposta é baseada na técnica de Inteligência Artificial denominada Programação em Lógica. Uma vez que o PPV pode ser visto como um problema de otimização multiobjetivo (minimizar frota, tempo de viagens mortas e tempo de terminal) a heurística tenta satisfazer os diferentes critérios na ordem que segue. O algoritmo tenta conectar as viagens mais próximas que pertencem à mesma linha e ao mesmo terminal, satisfazendo os tempos mínimos e máximos de terminal. Se a primeira estratégia não cobrir todas as viagens, então a restrição de pertinência à mesma linha é retirada, e são conectadas viagens que partem do mesmo terminal. Se for necessário, são conectadas viagens que partem de diferentes terminais, iniciando a busca pelos terminais mais próximos. No algoritmo genético proposto pelos autores, uma programação é representada por um gene, sendo que cada alelo (valor dentro do vetor) representa um veículo, e sua posição representa a viagem atribuída ao veículo. Para manter a factibilidade da solução após uma operação de cruzamento, aplica-se um mecanismo específico de reparação. A escolha dos indivíduos a serem reproduzidos na próxima geração é feita utilizando dois procedimentos não convencionais, além do método tradicional, segundo o qual a probabilidade de um indivíduo ser selecionado é proporcional ao valor da função objetivo.

Silva (2001) explora modelos de fluxo em redes para resolver o PPV. Duas técnicas de redução de rede são apresentadas. A primeira combina o modelo de fluxo com custo mínimo à técnica de redução de arcos, denominada Eliminação dos Arcos Longos. Apesar da redução significativa da rede, o método se mostra limitado na sua aplicabilidade. A segunda abordagem combina a técnica de Geração de Arcos Longos com o problema de pseudo-designação. A adaptação feita, denominada Geração de Arcos, considera, além da minimização do número de veículos, a redução do tempo de viagem morta e de terminal. O modelo não considera o tempo máximo de operação do veículo e se mostra pouco flexível para a inclusão de outras restrições operacionais, limitando sua aplicação.

O Problema de Programação de Tripulações de ônibus urbano (PPT) trata da geração de escalas de trabalho de motoristas e cobradores. A resolução do PPT é de grande importância, uma vez que nos gastos totais de uma empresa a mão-de-obra operacional representa um dos maiores custos. O PPT pertence a uma classe de problemas combinatoriais chamada programação de pessoal (*staff scheduling ou rostering*). Tais problemas são de difícil resolução, uma vez que apresentam um grande número de restrições operacionais e trabalhistas. Além disso, pertencem à classe de problemas NP-difíceis. Sendo assim, o uso

exclusivo de métodos exatos limita-se a resolver problemas de pequenas dimensões. Para problemas de dimensões maiores, a abordagem mais comum é através de heurísticas.

Marinho (2005) resolve o PPT por um método Busca Tabu com Relaxação Adaptativa que explora o espaço de busca por meio de duas estruturas de vizinhança: realocação e troca.

A representação de uma solução para o PPV pode ser facilmente adaptada para representar uma solução do PPT. Embora as restrições sejam diferentes, os dois problemas podem ser reduzidos a problemas de designação. No PPV, busca-se uma boa alocação de viagens em veículos, enquanto isso, no PPT a meta é encontrar uma boa alocação de tarefas às jornadas de trabalho dos tripulantes. Dessa forma, as mesmas técnicas utilizadas na resolução do PPV podem ser adaptadas para resolver o PPT.

Grande parte dos trabalhos que abordam o PPVT o faz resolvendo o PPV de maneira exata enquanto utiliza uma heurística para resolver o PPT. Esses trabalhos têm como referência o procedimento proposto por Ball *et al.* (1983), que envolve a definição de uma rede de programação, consistindo de vértices caracterizados por conjuntos de viagens denominados *d-trips* que devem ser executadas por um veículo e uma tripulação. Diversos tipos de arcos podem ser agrupados em duas categorias, aqueles que indicam que uma tripulação e veículo seguem de uma *d-trip* para outra a aqueles que indicam que apenas a tripulação segue de uma *d-trip* para outra (arco_tripulações). O procedimento de resolução é decomposto em três componentes, enfatizando a programação da tripulação: construção dos pedaços de jornadas, melhoramento dos pedaços de jornadas e geração das jornadas de trabalho das tripulações. A programação dos veículos é gerada simultaneamente excluindo os arcos_tripulações e fixando os arcos usados pelos pedaços de jornadas na solução. Heurísticas similares são propostas por Tosini & Vercellis (1988) e Patrikalakis & Xerocostas (1992).

Trabalhos mais recentes modelam o PPVT como Problema de Programação Linear Inteira Zero-Um, acrescentando ao modelo de Pseudo-designação, para a programação dos veículos, variáveis de decisão e restrições que garantem que cada tarefa seja atribuída a uma jornada. A função de avaliação é composta por componentes de custo dos veículos e das jornadas de trabalho. Uma restrição de *link*, incluída no modelo, garante que cada tarefa seja coberta por uma jornada se o arco correspondente está na solução do PPV (Haase & Friberg, 1999). Freling *et al.* (2001) realizam uma aplicação prática, considerando diferentes regras operacionais e seus impactos sobre a redução de custos.

Em Silva *et al.* (2006a) e Silva *et al.* (2006b) relatam-se experiências com a resolução integrada do problema, no caso, introduzindo na programação de veículos algumas restrições da programação das tripulações. Os resultados obtidos mostraram que a abordagem integrada produziu soluções com qualidade superior às aquelas geradas pela abordagem tradicional. Em uma instância considerada, a abordagem integrada utilizou um veículo a mais do que a tradicional, mas em contrapartida houve uma redução de 17 tripulações e ainda uma economia no número de horas extras pagas. Os autores lembram que o custo operacional das tripulações é superior ao custo operacional dos veículos, motivando o estudo de uma metodologia integrada para abordar os dois problemas.

Freling *et al.* (1999) classificam as abordagens de resolução do Problema de Programação de Veículos e Tripulações em: independente, seqüencial e integrada. Segundo os autores, a abordagem seqüencial pode ser dividida em tradicional e inversa, enquanto que a abordagem seqüencial pode ser dividida nível um e nível dois.

A abordagem seqüencial tradicional consiste em resolver o PPV e em seguida o PPT. A abordagem seqüencial inversa é uma variação dessa abordagem, consistindo em inverter a

ordem de resolução dos problemas, isto é, resolve-se primeiramente o PPT e em seguida o PPV (Souza *et al.*, 2004; Marinho *et al.*, 2004; Silva *et al.*, 2004; Atzingen, 2006).

As principais vantagens das abordagens sequenciais são a facilidade de implementação e a diminuição da complexidade do problema. A abordagem sequencial tradicional é recomendada para casos em que a programação dos veículos possui custo operacional superior ao custo operacional da programação das tripulações, e a inversa é indicada quando o custo operacional dos veículos é inferior ao das tripulações.

Na programação independente, a programação dos veículos é obtida ignorando-se a programação dos tripulantes e a programação dos tripulantes é obtida ignorando-se a programação dos veículos. Experiências desse tipo são retratadas em Bassi *et al.* 2007, que resolve o PPT sem considerar a programação dos veículos. O problema dessa abordagem é que, na maioria dos casos, os resultados obtidos são inviáveis na prática, pois as soluções obtidas para o PPV e para o PPT dificilmente são compatíveis entre si (Freling *et al.* 1999). Entretanto, essa abordagem permite determinar os limitantes inferiores para a solução ótima do PPVT.

A abordagem integrada começou a ser estudada mais recentemente, tendo em vista a evolução tecnológica dos computadores, uma vez que devido à grande complexidade do problema, o custo computacional desta abordagem era proibitivo.

A abordagem integrada nível um consiste em resolver simultaneamente o PPV e o PPT. Este é um problema complexo devido às suas grandes dimensões. Existem modelos computacionais para a sua resolução (Freling *et al.*, 1999; Friberg e Haase, 1999; Haase *et al.*, 2001), porém esses modelos não resolvem problemas de grandes dimensões em tempo aceitável devido às limitações computacionais existentes.

A abordagem integrada nível dois é baseada na resolução do PPV considerando características dos tripulantes de tal forma que a programação dos tripulantes é facilitada pela programação dos veículos.

Segundo Freling *et al.* 2003, a abordagem integrada do Problema de Programação de Veículos e de Tripulações sempre permitirá obter resultados superiores ou iguais aos resultados da abordagem sequencial, e resultados inferiores ou iguais aos resultados da abordagem independente.

3.2 Heurísticas

As heurísticas são procedimentos de busca que têm suas pesquisas guiadas por meio da quantificação de proximidade a um determinado objetivo (solução). Para tanto, emprega-se a função de avaliação que se baseia nas instâncias, características e estrutura do problema para encontrar boas soluções, possivelmente a ótima, a um custo computacional aceitável sem, contudo garantir a otimalidade ou quão próxima dela a solução se encontra.

As heurísticas são classificadas em heurísticas construtivas ou de refinamento dependendo da estratégia adotada para se convergir para a solução desejada. Uma heurística construtiva tem por objetivo construir uma solução, elemento por elemento, sendo a forma de escolha de cada elemento a ser inserido a cada passo variável de acordo com a função de avaliação adotada. As heurísticas de refinamento consistem em técnicas que realizam a exploração do espaço de soluções, através da estrutura de vizinhança que pode ser definida da seguinte forma: seja S o espaço de pesquisa de um problema de otimização, $s \in S$ uma das possíveis soluções que pode ser obtida de uma heurística construtiva ou gerada aleatoriamente

e f a função objetivo a ser otimizada. O conjunto $N(s) \subseteq S$, o qual depende da estrutura do problema tratado, reúne um número determinado de soluções s' , denominado *vizinhança* de s . Cada solução $s' \in N(s)$ é chamada de *vizinho* de s e é obtido de s a partir de uma operação chamada de *movimento*.

O obstáculo a ser vencido pelas heurísticas é encontrar, em um tempo reduzido, uma solução tão próxima da ótima quanto possível.

A seguir serão apresentadas as heurísticas referenciadas ao longo deste trabalho.

3.2.1 Construção Gulosa

A Construção Gulosa é uma heurística construtiva que, a cada iteração, integra à solução sendo gerada aquele elemento que trará mais benefícios dentre todos os candidatos, conforme uma função de avaliação (função gulosa).

A estimativa do benefício da inserção de cada elemento candidato à solução em construção deve ser calculada a cada iteração, uma vez que, esta solução parcial sofre alterações sempre que um novo elemento passa a integrá-la e conseqüentemente a relevância dos possíveis candidatos também é modificada.

O pseudocódigo da heurística construtiva gulosa aplicada a um problema de minimização é apresentado a seguir na Figura 3.1.

Deve ser salientado que os passos para se construir uma solução dependem do problema abordado e a diferentes funções de avaliação estão associadas soluções gulosas distintas.

- 1) Seja C o conjunto de elementos candidatos a integrar a solução.
 - 2) Seja s a solução sendo construída.
 - 3) $s \leftarrow \emptyset$
 - 4) enquanto $C \neq \emptyset$, faça
 - a) $f(t_{min}) = \min\{f(t) \mid t \in C\}$
 - b) $s \leftarrow s \cup \{t_{min}\}$
 - c) Atualize o conjunto C de elementos candidatos.
 - 5) Retorne a solução s .

Figura 3.1 – Pseudocódigo da Construção Gulosa

3.2.2 Método de Descida

O Método da Descida é uma técnica iterativa que parte de uma solução inicial qualquer, explora a vizinhança movendo-se sempre para o melhor vizinho segundo a função de avaliação, até atingir um ótimo local, do qual o método não é capaz de escapar.

Inicialmente, o Método da Descida explora toda a vizinhança de sua solução corrente. Se o melhor vizinho encontrado proporcionar uma melhora no valor da função de avaliação até então conhecida, ele é aceito como a nova solução a ser explorada. Caso contrário, um ótimo local foi alcançado e a solução corrente é retornada.

A Figura 3.2 mostra o pseudocódigo do Método de Descida, cujo objetivo é minimizar uma função de avaliação f .

- 1) Seja s_0 uma solução inicial conhecida.
- 2) $s \leftarrow s_0$, onde s é a solução corrente.
- 3) Faça $s' \leftarrow \text{melhorVizinho}(s)$, onde s' é o melhor dentre todos os vizinhos de uma dada solução s .
- 4) enquanto ($f(s') < f(s)$), faça
 - a) $s \leftarrow s'$
 - b) $s' \leftarrow \text{melhorVizinho}(s)$
- 5) Retorne a melhor solução encontrada, s .

Figura 3.2 – Pseudocódigo do Método da Descida

3.2.3 Método Descida Randômica

O procedimento Descida Randômica (DR) é uma variante do procedimento Descida apresentado na seção 3.2.2, e tem por finalidade evitar a busca exaustiva em toda a vizinhança de uma dada solução.

A idéia básica dessa técnica é obter a cada iteração um vizinho qualquer da solução corrente, admitindo-o como nova solução corrente caso ele proporcione melhoras no valor da função de avaliação. Caso contrário, um novo vizinho é gerado. Esse procedimento se repete até que se exceda um número determinado de iterações sem melhoras na solução.

Portanto, trata-se de um procedimento computacionalmente mais barato que o método de Descida clássico, mas que também fica preso ao primeiro ótimo local encontrado. Devido à aleatoriedade na escolha dos vizinhos, pode ocorrer de a solução retornada pelo método Descida Randômica com Primeiro de Melhora não corresponder a um ótimo local.

Na Figura 3.3 é apresentado o pseudocódigo do método Descida Randômica aplicado à minimização de uma função f .

- 6) Seja s_0 uma solução inicial conhecida.
- 7) $s \leftarrow s_0$, onde s é a solução corrente.
- 8) enquanto (*critério de parada não satisfeito*), faça
 - a) Faça $s' \leftarrow \text{primeiraMelhora}(s)$, onde s' é o primeiro vizinho de melhora de uma dada solução s .
 - b) $s \leftarrow s'$, se s' for melhor que s de acordo com f
 - c) Atualize o critério de parada
- 9) Retorne a melhor solução encontrada, s .

Figura 3.3 – Pseudocódigo do Método Descida Randômica

3.3 Metaheurísticas

Metaheurísticas podem ser vistas como sendo heurísticas genéricas mais sofisticadas, que visam explorar inteligentemente as instâncias do problema e o seu espaço de soluções. As metaheurísticas se caracterizam pela habilidade de escapar de ótimos locais e são destinadas a encontrar uma boa solução, eventualmente a ótima, para um problema de otimização.

As metaheurísticas se dividem em duas categorias, de acordo com o princípio usado para explorar o espaço de soluções: busca local e busca populacional. Nas metaheurísticas baseadas em busca local, a exploração do espaço de soluções é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora em sua vizinhança. Busca Tabu, *Simulated Annealing*, Método de Pesquisa em Vizinhança Variável, Método de Descida em Vizinhança Variável e *Iterated Local Search* são exemplos de métodos que se enquadram nesta categoria. Os métodos baseados em busca populacional, por sua vez, consistem em manter um conjunto de boas soluções e combiná-las de forma a tentar produzir soluções ainda melhores. Exemplos clássicos de procedimentos dessa categoria são os Algoritmos Genéticos e os Algoritmos Meméticos.

A seguir são apresentadas as metaheurísticas referenciadas neste trabalho.

3.3.1 *Iterated Local Search*

ILS (*Iterated Local Search* - Busca Local Iterada) é um método aplicado em conjunto com um procedimento de busca local, com a finalidade de melhorar as soluções obtidas por este através da realização de perturbações. Uma perturbação consiste em modificações realizadas sobre uma solução por meio da aplicação de um ou mais tipos de movimentos.

Para iniciar a sua exploração o ILS precisa primeiramente de uma boa solução para o problema, para tanto, é gerada uma solução inicial e esta é melhorada por meio de um procedimento de busca local.

Uma vez obtida uma solução é iniciada a fase de pesquisa sobre a sua vizinhança, que consiste em: se realizar uma perturbação sobre a solução corrente, seguida de um procedimento de busca que retornará o ótimo local da solução perturbada. Este ótimo é então avaliado e caso seja satisfeito o critério de aceitação (por exemplo, se houve melhora na função objetivo), ele será considerado a nova solução corrente e recomeça-se o processo de pesquisa pela vizinhança dessa nova solução. Caso contrário é aumentado o nível de perturbação, ou seja, torna-se a perturbação mais intensa e o processo de pesquisa é retomado. Este procedimento é repetido até que um critério de parada seja satisfeito, como por exemplo, quando o tempo de execução do algoritmo exceder um dado limite.

É importante salientar que o sucesso do ILS depende fortemente das perturbações realizadas pelo mesmo. Portanto, a perturbação aplicada a uma dada solução, deve ser dosada de maneira tal que as alterações decorrentes de sua aplicação sejam suficientes para se escapar de ótimos locais e explorar diferentes regiões sem, contudo causar um caos absoluto que implicaria na perda de características do ótimo local corrente.

Em princípio, qualquer método de busca local pode ser utilizado pelo ILS, no entanto, seu desempenho com respeito à qualidade da solução final e a velocidade de convergência dependem fortemente do método escolhido. Normalmente um método de descida é usado, mas também é possível aplicar algoritmos mais sofisticados, tais como Busca Tabu ou outras metaheurísticas.

Na Figura 3.6 é apresentado o pseudocódigo do algoritmo ILS básico em que se identificam quatro componentes:

- a) Procedimento *GeraSolucaoInicial*, que gera uma solução inicial s_0 para o problema;
- b) Procedimento *BuscaLocal*, que retorna uma solução melhorada s após geração da solução inicial e outra s'' após perturbação por meio de um método de busca local;
- c) Procedimento *Perturbacao*, que modifica a solução corrente s guiando a uma solução intermediária s' ;
- d) Procedimento *CriterioAceitacao*, decisão que se refere tanto à aceitação da solução perturbada como solução corrente, quanto quão intenso será a próxima perturbação a ser aplicada (isto é, determinação do nível de perturbação).

```

1)  $s_0 \leftarrow \text{GeraSolucaoInicial}();$ 
2)  $s \leftarrow \text{BuscaLocal}(s_0);$ 
3) enquanto (os critérios de parada não estiverem satisfeitos) faça
   a)  $s' \leftarrow \text{Perturbacao}(\text{historico}, s);$ 
   b)  $s'' \leftarrow \text{BuscaLocal}(s');$ 
   c)  $s \leftarrow \text{CriterioAceitacao}(s, s'', \text{historico});$ 
4) Retorne a solução  $s$ 

```

Figura 3.4 – Pseudocódigo do algoritmo *Iterated Local Search*

3.3.2 *Variable Neighborhood Descent*

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto por Nenad Mladenovic e Pierre Hansen (Mladenovic e Hansen, 1997), é um método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O pseudocódigo desse algoritmo, em que se considera o refinamento de uma solução s utilizando uma função de avaliação f , a ser minimizada, e um conjunto de r diferentes vizinhanças $N = N^{(1)}; N^{(2)}; \dots; N^{(r)}$, é apresentado pela Figura 3.5.

Dependendo do problema abordado, a busca pelo melhor vizinho (linha 5 da Figura 3.5) pode ser cara computacionalmente. Nessa situação é comum fazer a busca pela primeira solução de melhora. Outra alternativa é considerar a exploração apenas em certo percentual da vizinhança.

Algoritmo VND

```
1:  Entrada:  $f(\cdot)$ ,  $N(\cdot)$ ,  $r$ ,  $s$ 
2:  Seja  $r$  o número de estruturas diferentes de vizinhança
3:   $tipo \leftarrow 0$ 
4:  enquanto  $tipo < r$  faça
5:    Encontre o melhor vizinho  $s' \in N^{(tipo)}(s)$ 
6:    se (  $f(s') < f(s)$  )
7:       $s \leftarrow s'$ 
8:       $tipo \leftarrow 0$ 
9:    senão
10:      $tipo \leftarrow tipo + 1$ 
11:  fim se
12: fim enquanto
13: retorne  $s$ 
```

Figura 3.5 – Algoritmo VND

Segundo os autores, o método VND baseia-se em três princípios básicos:

- Um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança;
- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
- Para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximas.

Segundo os autores, o último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Esse é o caso em que os ótimos, tanto local quanto global, compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

3.3.3 Busca Tabu

O método de Busca Tabu, proposto independentemente por Glover (1986) e Hansen (1986), é um procedimento iterativo para a solução de problemas de otimização combinatória que aceita movimentos de piora para tentar escapar de ótimos locais.

Partindo de uma solução inicial s_0 , a cada interação, o método Busca Tabu explora um subconjunto V de uma vizinhança $N(s)$ da solução corrente s . A busca nessa vizinhança é interrompida quando um movimento que melhore a solução corrente for encontrado. A idéia de reduzir o espaço de busca se deve ao fato da iteração do método de Busca Tabu ser custosa computacionalmente para o PPVT. Essa interrupção da busca, conhecida como primeira melhora, é uma idéia aplicada em métodos heurísticos e visa reduzir o esforço computacional para explorar a vizinhança.

A escolha do melhor vizinho a cada iteração é utilizada para tentar escapar de mínimos locais. Porém, este critério de escolha pode fazer com que o algoritmo cicle, ou seja, que retorne a uma solução já visitada anteriormente. Para evitar a ciclagem, existe um mecanismo chamado memória de curto prazo ou lista tabu. O objetivo dessa lista é tentar evitar

movimentos que levem à regiões já visitadas do espaço de soluções, o que usualmente é alcançado pela proibição dos últimos movimentos realizados. Esses movimentos são mantidos nesta lista por um certo número de iterações chamado *tabu tenure*. Durante estas iterações, os movimentos armazenados ficam proibidos de serem realizados. Mas essa estratégia pode ser muito restritiva e desconsiderar soluções de alta qualidade. Para que isto não ocorra, movimentos com status tabu podem ser aceitos se a nova solução produzida satisfizer ao critério de aspiração por objetivo, isto é, se a solução, ainda que tabu, for melhor que a melhor solução gerada até então.

De forma a reduzir ainda mais a probabilidade de ciclagem, o método de Busca Tabu implementado utiliza uma lista tabu de tamanho dinâmico. No caso, a cada iteração do método, o movimento tabu permanece na lista por um certo número de iterações, sorteado no intervalo $[\text{minTabuTenure}, \text{maxTabuTenure}]$. Na Figura 3.6 é apresentado o pseudocódigo do algoritmo de Busca Tabu, aplicado a um problema de minimização.

Algoritmo BT

```

1: Entrada:  $f()$ ,  $N()$ ,  $|V|$ ,  $\text{MinTabuTenure}$ ,  $\text{MaxTabuTenure}$ ,  $s$ 
2:  $s^* \leftarrow s$ ;                                     { Melhor solução até então }
3:  $\text{ListaTabu} \leftarrow \emptyset$ ;                     { Lista Tabu }
4: enquanto ( critério de parada não satisfeito ) faça
5:   Seja  $s' \leftarrow s \oplus m$  o melhor vizinho de  $V \subset N(s)$  tal que o movimento  $m$  não seja tabu
     ( $m \notin \text{ListaTabu}$ ) ou, se tabu, satisfaça  $f(s') < f(s^*)$ ;
6:   Atualize  $\text{ListaTabu}$ ;
7:    $s \leftarrow s'$ ;
8:   se (  $f(s) < f(s^*)$  ) então
9:      $s^* \leftarrow s$ ;
10:  fim-se
11: fim-enquanto;
12:  $s \leftarrow s^*$ ;
13: retorne  $s$ ;
Fim BT;

```

Figura 3.6 - Algoritmo Busca Tabu

FONTE: Souza (2008)

3.4 Método de Intensificação

3.4.1 Relaxação Adaptativa

A Relaxação Adaptativa ou oscilação estratégica consiste em orientar movimentos em relação a um nível crítico, que pode ser identificado por um estágio de construção ou um intervalo escolhido de valores para uma função. Tal nível crítico (ou fronteira de oscilação) freqüentemente representa um ponto onde o método normalmente iria ser interrompido. Em vez de parar quando esta fronteira é alcançada, as regras para selecionar movimentos são modificadas para permitir que a região definida pelo nível crítico seja atravessada. Esta abordagem então continua até uma dada profundidade além da fronteira de oscilação e volta novamente. O limite da oscilação é novamente alcançado e atravessado, desta vez na direção oposta, e assim por diante.

O processo de repetidamente abordar e atravessar o nível crítico a partir de diferentes direções cria um comportamento oscilatório. O controle sobre esse comportamento é estabelecido por meio da geração de avaliações e regras modificadas para os movimentos, dependendo da região navegada e da direção da busca. A possibilidade de percorrer uma trajetória já visitada pode ser evitada por mecanismos tabu padrões, tais como aqueles estabelecidos pelas funções da memória de curto prazo.

Schaerf (1996) apresenta um mecanismo de relaxação adaptativa onde os pesos para cada fonte de inviabilidade da função de avaliação são ajustados dinamicamente, como proposto originalmente por Gendreau *et al.* (1994). Para cada inviabilidade i o peso β_i é multiplicado por um fator σ_i que varia de acordo com o seguinte esquema:

- No início da busca $\sigma_i \leftarrow 1$.
- A cada k movimentos:
 - se todas as k soluções visitadas são factíveis em relação à restrição i então $\sigma_i \leftarrow \sigma_i / \gamma$;
 - se todas as k soluções visitadas são infactíveis em relação à restrição i então $\sigma_i \leftarrow \sigma_i \times \gamma$;
 - se algumas soluções são factíveis e outras são infactíveis, considerando a restrição i , então σ_i permanece inalterado.

O parâmetro γ é randomicamente selecionado, a cada k movimentos, no intervalo $[1,8; 2,2]$ conforme proposto por Schaerf (1996). Cada valor de σ_i é limitado por duas constantes σ_{min} e σ_{max} , de forma a evitar que a relaxação adaptativa incremente/decremente indefinidamente os pesos para restrições que são sempre insatisfeitas/satisfeitas.

4 O Problema de Programação de Veículos e Tripulações Abordado

O presente capítulo descreve, em detalhes, o problema abordado. Na seção 4.1, apresenta-se as restrições e objetivos da programação de veículos e na seção 4.2, da programação de tripulações.

O Problema de Programação Integrada de Veículos e Tripulações (PPVT) abordado neste trabalho pode ser definido da seguinte maneira: dada uma tabela de viagens a serem realizadas, encontre a programação de custo mínimo para os veículos e as tripulações, fazendo o melhor aproveitamento da frota de veículos e dos tripulantes de uma empresa de modo que todas as viagens a ela responsabilizadas sejam realizadas.

Para que a programação de atividades da frota de veículos e do conjunto de tripulantes seja factível é necessário atender a uma série de restrições, as quais podem variar conforme as políticas operacionais que regem o Sistema de Transporte Público no qual a empresa atua. Neste trabalho é tratado o problema particular envolvendo empresas da cidade de Belo Horizonte/Brasil.

A resolução do problema de programação de veículos e de tripulantes é feita a partir de uma tabela de horários, na qual se encontram todas as viagens a serem realizadas em um determinado dia por uma empresa. A Tabela 4.1 é um exemplo de tabela de horários.

Tabela 4.1 – Exemplo de uma tabela de viagens

idViagem	Horário de Início	Ponto Inicial	Horário de Término	Ponto Final	Linha
0	100	1	290	1	LN305
1	140	1	210	1	LN305
2	310	1	380	1	LN305
3	310	1	460	1	LN150
4	470	1	540	1	LN150
5	480	1	550	1	LN305
6	575	1	790	1	LN305
7	640	1	710	1	LN150
8	730	1	880	1	LN150
9	860	1	1090	1	LN150

Na tabela de horários, cada viagem é identificada univocamente por meio de um conjunto de características que consistem em informações relevantes no processo de resolução do problema. Organizados em colunas e aparecendo sequencialmente na tabela, temos: o número da viagem, horário de início da viagem (em minutos), o ponto inicial que corresponde ao local de início da viagem, horário de término da viagem (em minutos), o ponto final que se refere ao local em que a viagem termina e a linha a qual a viagem pertence.

O tempo de embarque e o tempo de desembarque de passageiros também é um parâmetro importante em uma viagem. O primeiro dependente do horário de início da viagem e o segundo do horário de fim da mesma.

Ao longo do dia existem determinados períodos em que geralmente há um número maior de passageiros em trânsito pelo meio de transporte em questão. Esses períodos são os chamados horários de pico. As empresas estimam que na prática nesses horários o tempo necessário para o de embarque ou desembarque de passageiros é de dois minutos e fora desses

horários esse tempo é de um minuto. Existem dois horários de pico ao longo do dia: das 5:00 às 7:59 da manhã e das 4:00 às 18:59.

Além do conjunto de viagens, outra informação importante é a matriz de tempos de deslocamento entre os vários pontos de parada dos veículos e tripulantes, incluindo a garagem. De posse dessa informação é possível, por exemplo, determinar o tempo que um veículo percorre fora de operação, ou seja, sem passageiros para sair de um ponto e chegar a outro. A este tipo de deslocamento é dado o nome de viagem morta.

Quanto aos horários de início e término a tabela de viagens de exemplo (Tabela 4.1) é representada esquematicamente na Figura 4.1:

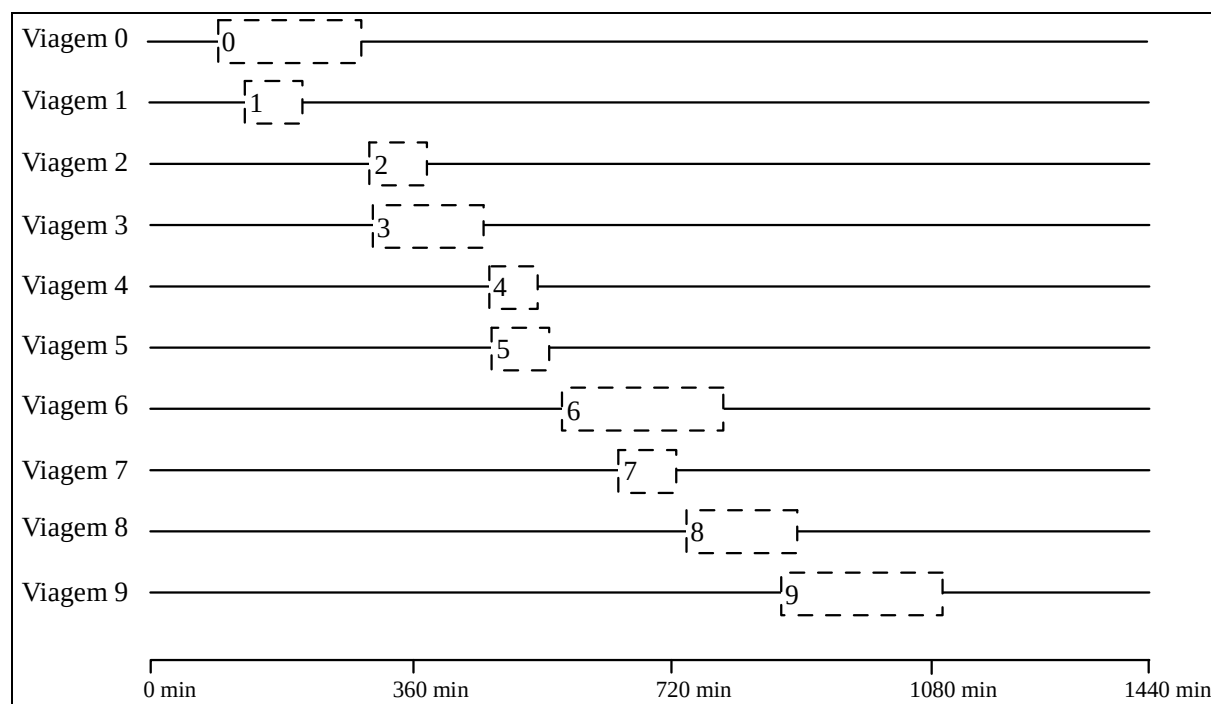


Figura 4.1 - Representação esquemática das viagens da Tabela 4.1

O PPT em um sistema de transporte público consiste na definição da jornada de trabalho de cada tripulação pertencente ao quadro de uma dada empresa. Uma tripulação é constituída por um motorista e um cobrador (quando houver), responsáveis pela operação de um veículo. As tarefas são formadas a partir de jornadas de trabalho de veículos, agrupando conjuntos de viagens. Este agrupamento é feito de maneira que cada tarefa fica compreendida entre duas *Oportunidades de Trocas (OT)*. Uma OT é um intervalo de tempo suficiente, em um ponto apropriado, que possibilita a troca de tripulações de um veículo em operação. Viagens de uma mesma tarefa são executadas pelo mesmo veículo. Durante a realização de uma tarefa, não é possível que haja a troca da tripulação. Sendo assim, uma tarefa é a mínima porção de trabalho que pode ser adicionada ou retirada da jornada de trabalho de uma tripulação.

A programação das tripulações é feita a partir de um conjunto de tarefas que, unidas, contêm todas as viagens a serem realizadas em um determinado dia por uma empresa. A Figura 4.2 representa um conjunto de tarefas, gerado a partir de uma possível organização das viagens da Tabela 4.1. Na Figura 4.2 a Tarefa 2 contém a viagem 3 e a viagem 4.

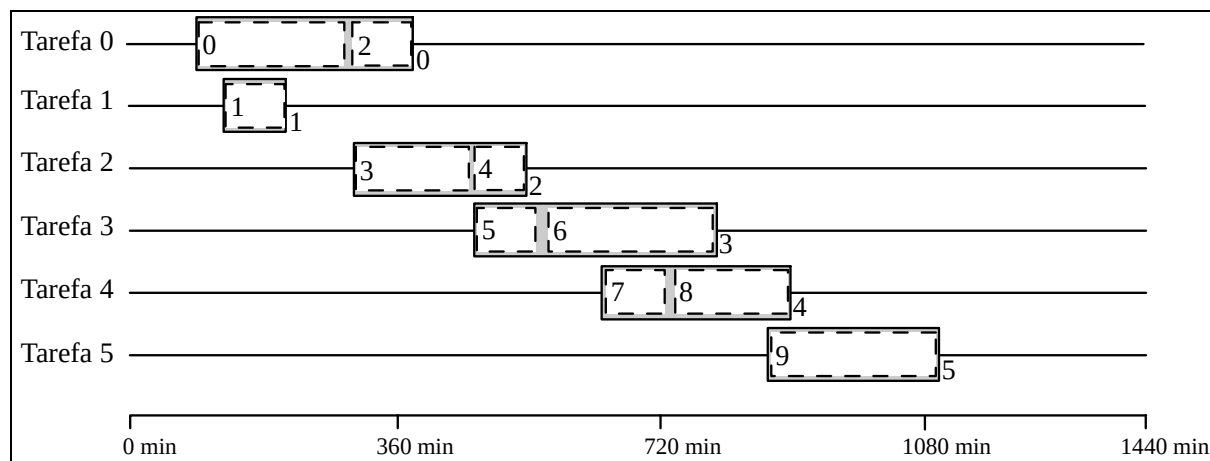


Figura 4.2 - Representação esquemática de uma lista de tarefas

4.1 Restrições e Objetivos da Programação de Veículos

As restrições que foram levadas em consideração para geração da programação diária dos veículos neste trabalho, foram as seguintes:

- Um veículo deve permanecer no mínimo trinta minutos consecutivos na garagem, por dia. Este tempo pode ser cumprido quando o veículo pára na garagem entre duas viagens consecutivas pertencentes à sua jornada diária ou mesmo depois de completada toda a sua jornada de trabalho;
- Um mesmo veículo não pode realizar duas viagens simultaneamente, o que caracterizaria uma sobreposição;
- Antes de se iniciar e após finalizar uma viagem, um veículo deve obrigatoriamente cumprir um tempo mínimo no terminal, que corresponde ao tempo necessário para o embarque e desembarque dos passageiros;
- Um veículo não deve permanecer mais de duas horas em um terminal esperando para executar a sua próxima viagem. Se o tempo de espera exceder esse limite, o veículo deverá se deslocar para a garagem ou outro local onde possa esperar o horário de início da viagem subsequente. Diz-se, nesse último caso, que o veículo está fazendo uma dupla pegada;
- Todas as viagens de responsabilidade de uma empresa, sem exceções, devem ser realizadas por algum veículo da frota.

Os objetivos do PPV considerados neste trabalho são os seguintes: (1) diminuir o tempo ocioso dos veículos, isto é, que os veículos não fiquem parados entre duas viagens consecutivas descontando-se os tempos de embarque e desembarque que são obrigatórios; (2) diminuir o número de deslocamentos fora de operação (deslocamentos que não equivalem a nenhuma viagem); (3) reduzir o tamanho da frota utilizada e (4) diminuir o número de trocas de linha entre as viagens dos veículos da frota.

A solução para o Problema de Programação de Veículos consiste, portanto, em atribuir um conjunto de viagens para um número de veículos menor ou igual à frota disponível pela empresa, considerando as restrições apresentadas acima, de tal forma que todas as viagens sejam realizadas com o menor custo operacional possível. Cada conjunto de viagens a serem executadas por um veículo é chamado de “bloco do veículo”.

4.2 Restrições e Objetivos da Programação de Tripulações

Para que a programação das jornadas de trabalho das tripulações seja factível é necessário atender a uma série de restrições. Sendo assim, o Problema de Programação de Tripulações (PPT) pode ser definido da seguinte forma: dado um conjunto de tarefas, encontrar um conjunto de jornadas diárias com custo mínimo tal que cada jornada seja uma sequência de tarefas que pode ser executada por uma única tripulação respeitando as restrições operacionais e trabalhistas. As restrições consideradas são as seguintes:

- a) Cada tarefa deve ser atribuída a uma única jornada de trabalho;
- b) Um mesmo tripulante não pode realizar duas tarefas simultaneamente, o que caracterizaria uma sobreposição;
- c) Um tripulante não deve permanecer mais de duas horas em um terminal esperando para executar a sua próxima viagem. Se o tempo de espera exceder esse limite, o tripulante deverá se deslocar para a garagem ou outro local onde possa esperar o horário de início da tarefa subsequente. Diz-se, nesse último caso, que o tripulante está fazendo uma dupla pegada;
- d) A duração da jornada diária de trabalho não pode superar 9 horas se a jornada for classificada como simples, ou 8:40 se for classificada como de dupla pegada;
- e) As trocas de tripulações só podem ocorrer em determinados locais, nos pontos;
- f) Nenhum dos tripulantes da frota deve efetuar trocas proibidas de veículos, linhas ou pontos;
- g) Todas as viagens de responsabilidade de uma empresa, sem exceções, devem ser realizadas por algum tripulante.

Uma troca de veículos é permitida quando há tempo para se efetuar a troca, caso contrário ela é proibida. Uma troca de linha é permitida se as linhas pertencerem a um mesmo grupo, caso contrário ela é proibida. A definição dos grupos das linhas é feita pela empresa de transporte. Uma troca de ponto de ônibus é permitida (1) se os pontos envolvidos na troca pertencerem ao mesmo grupo de pontos; (2) caso contrário, se as tarefas pertencerem ao mesmo veículo; (3) caso contrário, apenas se houver dupla pegada entre as duas tarefas. Uma troca de ponto é proibida caso não satisfaça nenhum dos três quesitos anteriores.

Os objetivos do PPT considerados neste trabalho são os seguintes: gerar soluções finais factíveis, ou seja, sem sobreposição de tarefas e sem excesso de trabalho, além disso, deve haver tempo suficiente entre as jornadas de trabalho de um tripulante, segundo as leis trabalhistas; o número de trocas de veículos, o número de trocas de linhas e o número de trocas de pontos devem ser mínimos, assim como o tempo ocioso e o tempo de hora extra; não deve haver excesso de duplas pegadas; e o número de tripulantes também é um item a ser minimizado.

5 Metodologia

Neste capítulo apresenta-se a metodologia proposta para resolver, de forma integrada, o problema de programação de veículos e tripulações (PPVT). Na seção 5.1 mostra-se como uma solução para o problema é representada. Na seção 5.2 são apresentados os tipos de movimentos que constituem as estruturas de vizinhança desenvolvidas para explorar o espaço de busca. Nas seções 5.3 e 5.4 são mostradas duas formas utilizadas para avaliar uma solução, sendo a primeira com pesos empíricos e a segunda, baseada em custos. A descrição de como uma solução inicial é gerada é apresentada na seção 5.5. O algoritmo proposto para resolver o PPVT é apresentado na seção 5.6.

5.1 Representação do PPVT

Uma solução s para o PPVT consiste em um par (s_V, s_T) , onde s_V é uma solução para o problema de programação de veículos e s_T uma solução para o problema de programação de tripulações.

A solução s_V consiste em um conjunto de veículos, a cada qual está associado uma lista de viagens a serem por ele executadas durante um dia de trabalho. As viagens executadas por cada veículo são mantidas ordenadas por seus horários de início. Essa abordagem é importante, uma vez que através dela será possível avaliar a jornada de trabalho de um veículo sob vários aspectos, tais como: tempo de operação em viagem, tempo de locomoção fora de operação, ou seja, deslocamentos que não correspondem a nenhuma viagem, tempo de espera em terminal e total de retornos à garagem.

A solução s_T , por sua vez, consiste em um conjunto de tripulações (motoristas e cobradores), a cada qual está associada uma lista de tarefas a serem executadas pelos tripulantes diariamente.

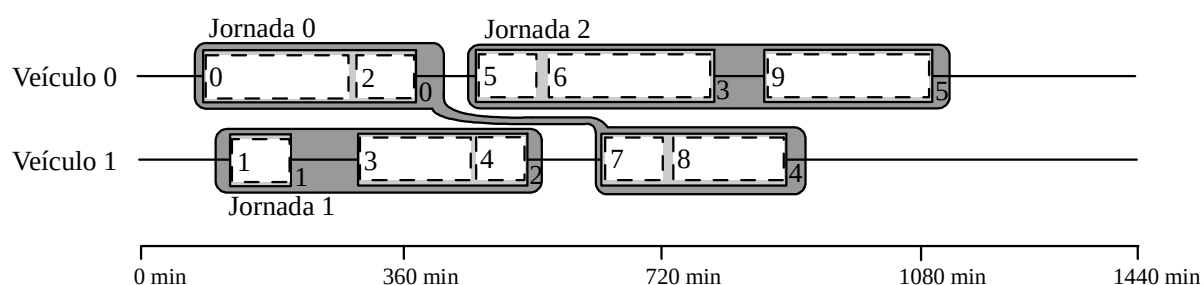


Figura 5.1 – Exemplo de uma solução s para o PPVT

A Figura 5.1 ilustra uma solução $s = (s_V, s_T)$ para o PPVT, na qual uma frota de dois veículos executa dez viagens a ela responsabilizada. Cada viagem é representada por um retângulo pontilhado de cor branca, cada tarefa é um retângulo de cor cinza claro, e cada jornada é a figura que une tarefas, cinza escuro. Nessa solução, o Veículo 1 executa, ao longo do dia, as viagens 1, 3, 4, 7 e 8. Onde, a viagem 1 constitui a tarefa 1, as viagens 3 e 4 constituem a tarefa 2 e as viagens 7 e 8 a tarefa 4. Note que o tripulante responsável pela jornada 0 executa a tarefa 0 no veículo 0 e a tarefa 4 no veículo 1.

5.2 Estruturas de vizinhança

Os métodos heurísticos usados neste trabalho são baseados na efetuação de modificações em uma dada solução com a finalidade de se melhorar sua qualidade. Essas modificações são chamadas *movimentos* e diferentes tipos de movimento podem ser aplicados em uma solução. Uma estrutura de vizinhança é definida por um tipo de movimento.

Um vizinho de uma solução s é definido pela aplicação de um movimento na solução s . Aplicando diferentes tipos de movimentos os métodos heurísticos podem explorar diferentes regiões do espaço de soluções.

Com vistas a explorar o espaço de soluções definimos 6 estruturas de vizinhança.

Duas delas N_v^R e N_v^T fazem modificações somente nos veículos, ignorando os reflexos destes movimentos na programação de tripulações, o que gera, possivelmente, soluções inviáveis.

Outras duas estruturas N_t^R e N_t^T modificam somente a programação das tripulações, preservando a compatibilidade entre veículos e tripulações.

O terceiro par de estruturas de vizinhança N_v^{RP} e N_v^{TP} é integrado e modifica ambas as partes, veículos e tripulações. Essas duas últimas estruturas consistem na realocação ou troca as viagens de veículos, seguida de uma reconstrução das jornadas de forma a tornar as programações de veículos e tripulações compatíveis entre si. Para cada veículo envolvido na modificação, são eliminadas da programação de tripulações todas as tarefas pertencentes a esses veículos. A seguir, a partir da nova configuração dos blocos dos veículos envolvidos, são geradas as novas tarefas. Essas tarefas, por sua vez, são adicionadas a qualquer tripulante, de forma gulosa de acordo com o procedimento descrito na seção anterior. Para essas duas estruturas, N_v^{RP} e N_v^{TP} , diz-se que a modificação realizada é “propagada” para as tripulações.

5.2.1 Vizinhança N_v^R

O movimento de realocação definido pela estrutura de vizinhança N_v^R consiste em transferir uma viagem pertencente à lista de viagens de um dado veículo para a lista de outro veículo. Portanto, este movimento pode ser visto como uma seqüência de decisões a serem tomadas e uma respectiva ação da seguinte forma:

1. Escolher um primeiro veículo, dentre todos os envolvidos no problema;
2. Escolher um segundo veículo que não seja o mesmo selecionado anteriormente;
3. Escolher uma viagem do primeiro veículo que será transferida para a lista de viagens do segundo;
4. Efetivar o movimento.

5.2.2 Vizinhança N_v^T

O movimento de troca definido pela estrutura de vizinhança N_v^T consiste em permutar uma tarefa da jornada de trabalho de um dado tripulante com uma tarefa pertencente a um outro tripulante. Igualmente aos processos anteriores, no movimento de troca, antes que se possa efetivamente realizar a permutação das tarefas algumas decisões devem ser tomadas:

1. Escolher um primeiro veículo, dentre todos os envolvidos no problema;
2. Escolher um segundo veículo que não seja o mesmo selecionado anteriormente;

3. Escolher uma viagem do primeiro veículo que será transferida para o bloco do segundo;
4. Escolher uma viagem do segundo veículo que será transferida para o bloco do primeiro;
5. Efetivar o movimento.

5.2.3 Vizinhaça N_t^R

O movimento de realocação definido pela estrutura de vizinhaça N_t^R consiste em transferir uma tarefa da jornada de trabalho de um dado tripulante para a jornada de trabalho de outro tripulante. Portanto, este movimento pode, também, ser visto como uma seqüência de decisões a serem tomadas e uma respectiva ação da seguinte forma:

1. Escolher um primeiro tripulante, dentre todos os envolvidos no problema;
2. Escolher um segundo tripulante que não seja o mesmo selecionado anteriormente;
3. Escolher uma tarefa do primeiro tripulante que será transferida para a jornada de trabalho do segundo;
4. Efetivar o movimento.

5.2.4 Vizinhaça N_t^T

O movimento de troca definido pela estrutura de vizinhaça N_t^T consiste em permutar uma tarefa da jornada de trabalho de um dado tripulante com uma tarefa pertencente a um outro tripulante. Igualmente aos processos anteriores, no movimento de troca, antes que se possa efetivamente realizar a permutação das tarefas algumas decisões devem ser tomadas:

6. Escolher um primeiro tripulante, dentre todos os envolvidos no problema;
7. Escolher um segundo tripulante que não seja o mesmo selecionado anteriormente;
8. Escolher uma tarefa do primeiro tripulante que será transferida para a jornada de trabalho do segundo;
9. Escolher uma tarefa do segundo tripulante que será transferida para a jornada de trabalho do primeiro.
10. Efetivar o movimento.

5.2.5 Vizinhaça N_v^{RP}

Na estrutura N_v^{RP} , um vizinho s' de uma solução $s = (s_v, s_T)$ do PPVT é gerado fazendo-se a seguinte seqüência de operações: (i) realocação de uma viagem de um veículo para outro veículo na solução s_v ; (ii) eliminação, na solução s_T , das tarefas que pertencem aos veículos envolvidos na troca; (iii) agrupamento, em tarefas, das viagens dos dois veículos envolvidos na realocação; (iv) geração de uma nova solução s_T , consistindo na distribuição das tarefas aos tripulantes por um procedimento guloso, conforme descrito na seção seguinte.

As estruturas de vizinhaça N_v^{RP} e N_v^{TP} fazem uso de um método que, de forma gulosa, aloca as tarefas geradas por uma modificação, realocação ou troca, na programação dos veículos. Nessas estruturas, a primeira modificação se dá primeiramente na programação de veículos, sendo essa modificação propagada para a programação de tripulações. Essa propagação é feita substituindo as tarefas, anteriormente geradas com base na lista de viagens

dos veículos envolvidos, pelas novas tarefas, geradas a partir da lista de viagens dos mesmos veículos depois da modificação.

5.2.6 Vizinhaça N_t^{TP}

Na estrutura N_t^{TP} , um vizinho s' de uma solução $s = (s_V, s_T)$ do PPVT é gerado fazendo-se a seguinte seqüência de operações: (i) troca de viagens entre dois veículos da solução s_V ; (ii) eliminação, na solução s_T , das tarefas que pertencem aos veículos envolvidos na troca; (iii) agrupamento, em tarefas, das viagens pertencentes aos dois veículos envolvidos na troca; (iv) geração de uma nova solução s_T , consistindo na distribuição das tarefas, geradas pelo passo anterior, aos tripulantes por um procedimento guloso, conforme descrito na seção 5.2.5.

5.3 Avaliação de uma solução - 1ª Abordagem

Nesta abordagem, baseada em pesos empíricos, uma solução $s = (s_V, s_T)$ do PPVT é avaliada por uma função f baseada em penalidades, a qual deve ser minimizada, na forma:

$$f(s) = \alpha \times f_V(s) + \beta \times f_T(s) \quad (1)$$

em que $f_V(s)$ representa a componente que avalia s com relação à programação dos veículos, $f_T(s)$ representa a componente que avalia s com relação à programação das tripulações e α e β representam pesos atribuídos à importância relativa de cada uma dessas soluções.

5.3.1 Avaliação da programação de veículos

A função de avaliação f_V considerada para a componente s_V relativa a programação dos veículos é dada pela expressão (2), baseada em Simões *et al.* (2006).

A avaliação de uma solução para os veículos considera: tempo de sobreposição de viagens, tempo de viagem morta, tempo de espera em terminal, número de trocas de linhas, excesso de duplas pegadas e número de veículos.

$$f_V(s) = \sum_{k \in F} f_V^k(s) + numTrLinha \times \omega_{TL} + excessoDuplaPegV \times \omega_{EDP} + numVeículos \times \omega_{NV} \quad (2)$$

onde:

- a) $f_V^k(s)$ é a função de custo do k -ésimo veículo da frota F de uma solução s .
- b) $numTrLinha$ é a quantidade de trocas de linha realizadas pelos veículos da frota e ω_{TL} é a penalização associada.
- c) $excessoDuplaPegV$ é a quantidade de duplas pegadas da programação de veículos que supera um certo número admissível previamente definido pela empresa e ω_{EDP} é a penalização correspondente.
- d) $numVeículos$ é a quantidade de veículos escalados para executar o conjunto de viagens e ω_{NV} é a penalização associada.

O custo $f_v^k(s)$ atribuído ao veículo k de uma solução s é calculado pela expressão (3). Sendo T o conjunto das suas n viagens diárias e i e j viagens consecutivas de sua jornada de trabalho.

$$f_v^k(s) = CD_{g1} + CD_{ng} + \sum_{(i,j) \in T} C_{ij} \quad (3)$$

Na equação (3), CD_{g1} e CD_{ng} são, respectivamente, os custos dos deslocamentos (isto é, das viagens mortas) do veículo v da garagem para o ponto de início de sua primeira viagem e do ponto final da sua última viagem para a garagem. C_{ij} é o custo associado à realização das viagens consecutivas i e j e é calculado conforme a expressão (4):

$$C_{ij} = \begin{cases} CD_{ij} + CS_{ij} & \text{se } t_{ij} < 0 \\ CT_{ij} + CD_{ij} & \text{se } 0 \leq t_{ij} \leq \text{TempoMaxTerm} \\ CD_{ig} + CD_{gj} & \text{se } t_{ij} > \text{TempoMaxTerm} \end{cases} \quad (4)$$

onde CT_{ij} , CD_{ij} e CS_{ij} correspondem, respectivamente, aos custos de permanência no terminal, deslocamento realizado fora de operação (viagem morta) e sobreposição entre as viagens i e j . Na expressão (4), TempoMaxTerm é o tempo máximo que um veículo pode permanecer no terminal, estipulado segundo políticas operacionais que regem o Sistema de Transporte Público, e t_{ij} corresponde ao período de tempo compreendido entre o fim da viagem i e o início da viagem j após descontado o tempo de viagem morta, de embarque e desembarque entre elas.

Note que t_{ij} corresponde ao tempo real de ociosidade do veículo v entre as viagens i e j . Portanto, o valor t_{ij} poderá ser:

- Negativo, indicando que há sobreposição de horário entre as viagens i e j . Assim, além da penalização associada à viagem morta realizada pelo veículo v entre as viagens i e j , há um custo atribuído ao tempo de sobreposição entre elas;
- Não-negativo e menor ou igual ao tempo máximo de terminal permitido para um veículo (TempoMaxTerm), indicando que o veículo v , após a realização da viagem i , ficará esperando no terminal pelo horário de início de sua próxima viagem (viagem j). Portanto, o custo de se realizar as viagens i e j consecutivamente é obtido penalizando-se o tempo de deslocamento do veículo fora de operação e o tempo que o mesmo permanece no terminal entre as viagens;
- Maior que o tempo máximo de terminal permitido para um veículo (TempoMaxTerm), indicando que o veículo v terá que retornar à garagem entre as viagens i e j , ou seja, realizará uma dupla pegada; não havendo, dessa forma, a necessidade de espera no terminal. O custo de se realizar as viagens, nesse caso, é calculado penalizando-se os tempos de deslocamento do veículo fora de operação.

Na prática, é desejável não apenas obter uma solução factível para o PPV com tempos de terminal e de deslocamento baixos na soma global da programação, mas também que exista uma homogeneidade na distribuição desses tempos, isto é, deve ser evitado que haja paradas no terminal excessivamente demoradas ou que os tempos de deslocamentos de um veículo fora de operação sejam muito elevados. Para cumprir esse objetivo, atribui-se um custo individual para cada espera no terminal ou para cada viagem morta de um veículo na expressão (4), de acordo com uma função exponencial. Os custos CD_{ij} e CT_{ij} são obtidos conforme as equações (5) e (6), apresentadas a seguir:

$$CD_{ij} = \begin{cases} (\alpha_1 \times d_{ij})^{\beta_1} & \text{se } d_{ij} \leq TempDesloc\_Lim \\ d_{ij} \times \omega_{TD} & \text{se } d_{ij} > TempDesloc\_Lim \end{cases} \quad (5)$$

$$CT_{ij} = \begin{cases} (\alpha_2 \times t_{ij})^{\beta_2} & \text{se } t_{ij} \leq TempTerm\_Lim \\ t_{ij} \times \omega_{TT} & \text{se } t_{ij} > TempTerm\_Lim \end{cases} \quad (6)$$

em que:

- a) t_{ij} é o tempo real de ociosidade do veículo v entre as viagens i e j ;
- b) d_{ij} é a duração da viagem morta do veículo v entre as viagens i e j ;
- c) $TempDesloc\_Lim$ é o tempo limite para a duração de uma viagem morta;
- d) ω_{TD} é a penalização, por unidade de tempo (em minutos), atribuída a uma viagem morta caso a sua duração supere o limite dado por $TempDesloc\_Lim$;
- e) α_1 e $\beta_1 \in (0,1)$ são parâmetros utilizados para o cálculo da penalização atribuída a uma viagem morta caso a sua duração seja menor ou igual ao limite dado por $TempDesloc\_Lim$;
- f) $TempTerm\_Lim$ é o tempo limite para a duração de uma parada no terminal;
- g) ω_{TT} é a penalização, por unidade de tempo (em minutos), atribuída a uma espera no terminal que exceda o limite dado por $TempTerm\_Lim$;
- h) α_2 e $\beta_2 \in (0,1)$ são parâmetros utilizados para o cálculo da penalização atribuída a uma espera no terminal caso a sua duração seja menor ou igual ao limite dado por $TempTerm\_Lim$.

Observe que os custos CT_{ij} e CD_{ij} são dependentes da duração da espera no terminal ou da viagem morta efetuada pelo veículo v entre as viagens i e j . A um tempo de terminal ou de deslocamento aceitável é associado um custo mais baixo, calculado com base em uma função exponencial. Caso esse tempo seja inaceitável, o custo é mais elevado e é calculado com base em uma função linear para diferenciar o nível de inviabilidade.

O motivo de se utilizar uma função exponencial está no fato de que ela estabelece uma variação suave do custo da espera no terminal (ou viagem morta) quando a duração é pequena e mais acentuada quando essa duração é maior. Assim, a tempos de terminal (ou viagem

morta) toleráveis estão associados custos mais próximos e para tempos quase intoleráveis, os custos são elevados.

Uma vez que a existência de sobreposição em uma solução para o PPV é uma inviabilidade, não se aplica para o cálculo de seu custo a mesma metodologia que a adotada anteriormente para penalizar o tempo de terminal e a viagem morta. Assim, o custo CS_{ij} é obtido de forma simples penalizando cada minuto de sobreposição pelo valor ω_{TS} , conforme equação (7):

$$CS_{ij} = - (t_{ij}) \times \omega_{TS} \quad (7)$$

Note que, uma vez que o valor t_{ij} é sempre negativo quando há sobreposição entre as viagens i e j , é necessário multiplicá-lo por (-1) para que um custo positivo seja associado a esta sobreposição.

5.3.2 Avaliação da programação de tripulações

A função de avaliação considerada para a componente s_T relativa a programação das tripulações é baseada em penalizações de requisitos essenciais e não essenciais. Como as restrições devem ser satisfeitas sob pena de a solução do PPT não ter aplicabilidade, os pesos dados aos requisitos essenciais são bem mais elevados que aqueles atribuídos aos requisitos não-essenciais.

A avaliação de uma solução para os tripulantes considera: tempo de sobreposição de tarefas, tempo de excesso de trabalho, tempo insuficiente entre jornadas, número de trocas de veículos, número de trocas de linhas, número de trocas de pontos, tempo ocioso, tempo de hora extra, excesso de duplas pegadas e número de tripulantes.

Uma solução s_T do PPT é avaliada com base na seguinte função $f_T(s)$, baseada em Marinho *et al.* (2004):

$$\begin{aligned} f_T(s) = & \sum_{i \in Trip} (penalidadeTempoOcioso_i + penalidadeHoraExtra_i + \\ & + \mu_3 \times numTrVPerm_i + \mu_4 \times numTrLPerm_i + \mu_5 \times numTrPPerm_i + \\ & + \lambda_1 \times tempoSobrepos_i + \lambda_2 \times tempoExcTrab_i + \lambda_3 \times tempoInsufEJorn_i + \\ & + \lambda_4 \times numTrocaVPr oib + \lambda_5 \times numTrLPr oib_i + \lambda_6 \times numTrPPr oib_i) + \\ & + \lambda_7 \times excessoDuplaPeg + \delta \times numTripulantes \end{aligned} \quad (8)$$

onde:

- (i) $Trip$ é o conjunto de tripulantes da componente s_T .
- (ii) μ_i e λ_i são as penalidades aplicadas aos requisitos não essenciais e essenciais respectivamente.
- (iii) $penalidadeTempoOcioso_i$ é a penalidade referente à quantidade total de ociosidade, em minutos, na jornada i e μ_1 é a penalização desse quesito. O valor μ_1 deve ser escolhido de tal forma que ele seja maior que o valor $penalidadeTempoOcioso_i$ caso o $tempoOcioso_i$ seja igual a 120 minutos (duas horas). O valor dessa penalidade é dado pela expressão 9, onde $tempoOcioso$ representa a quantidade de tempo, em minutos, em que uma jornada i é ociosa.

$$penalidadeTempoOcioso_i = \begin{cases} \mu_1, & \text{se } tempoOcioso_i > 120 \\ \lfloor 0.347 \times tempoOcioso_i^{0.8} \rfloor_{cc} & \end{cases} \quad (9)$$

- (iv) $penalidadeHoraExtra_i$ é a penalidade referente à quantidade total de horas extras, em minutos, na jornada i e μ_2 é a penalização desse quesito. O valor μ_2 deve ser escolhido de tal forma que ele seja maior que o valor $penalidadeHoraExtra_i$ caso o $tempoExtra_i$ seja igual a 120 minutos (duas horas). O valor dessa penalidade é dado pela expressão 10.

$$penalidadeHoraExtra_i = \begin{cases} \mu_2, & \text{se } tempoExtra_i > 120 \\ \lfloor 1.357 \times tempoExtra_i^{0.6} \rfloor_{cc} & \end{cases} \quad (10)$$

- (v) $numTrVeiculosPerm_i$ é o número de vezes em que ocorreu troca de veículos permitida durante a jornada i e μ_3 a penalização para esse quesito. Uma troca de veículos é permitida quando há tempo para se efetuar a troca.
- (vi) $numTrLPerm_i$ é o número de vezes em que ocorreu troca de linhas permitida durante a jornada i e μ_4 a penalização correspondente. Uma troca de linha é permitida se as linhas pertencerem a um mesmo grupo. A definição dos grupos das linhas é feita pela empresa de transporte.
- (vii) $numTrPPerm_i$ é o número de vezes em que ocorreu troca de pontos permitida durante a jornada i e μ_5 a penalização desse quesito. Uma troca de ponto de ônibus é permitida (1) se os pontos envolvidos na troca pertencerem ao mesmo grupo de pontos; (2) caso contrário, se as tarefas pertencerem ao mesmo veículo; (3) caso contrário, apenas se houver dupla pegada entre as duas tarefas. Uma troca de ponto é proibida caso não satisfaça nenhum dos três quesitos anteriores.
- (viii) $tempoSobrep_i$ é a quantidade de tempo, em minutos, em que duas tarefas são realizadas simultaneamente na jornada i e λ_1 a penalização para esse quesito.
- (ix) $tempoExcTrab_i$ é a quantidade de tempo excedente, em minutos, na jornada i e λ_2 a penalização correspondente. O tempo excedente em uma jornada é dado por seu tempo de trabalho menos o tempo de máxima duração de uma jornada, que é dado pelo tempo limite de trabalho segundo a classificação da jornada, simples ou pegada dupla, mais o tempo máximo de hora extra, que, neste caso é de duas horas (120 minutos). A expressão 11 apresenta o cálculo desse valor.

$$tempoExcTrab_i = \begin{cases} \max(0, tempoTrab_i - (tempoLimJornSimples + 120)), & \text{se } i \text{ for Jornada Simples} \\ \max(0, tempoTrab_i - (tempoLimJornDupla + 120)), & \text{se } i \text{ for Jornada Dupla} \end{cases} \quad (11)$$

- (x) $tempoTrab_i$ é a quantidade de tempo de trabalho, em minutos, da jornada i . Esse valor é dado pelo horário final da última tarefa, da jornada em questão, menos o horário de início da primeira, menos a duração do intervalo de dupla pegada se a jornada for classificada como de pegada dupla.
- (xi) $tempoInsufEJorn_i$ é a quantidade de tempo, em minutos, que falta para que a diferença entre o horário final da jornada i em um dia e o horário inicial no dia subsequente atinja o tempo mínimo entre jornadas de 11 (onze) horas. λ_3 é a penalização correspondente.

- (xii) $numTrVProib_i$ é o número de vezes em que ocorreu troca de veículos proibida durante a jornada i e λ_4 a penalização para esse quesito. Uma troca de veículos é proibida quando não há tempo para se efetuar a troca.
- (xiii) $numTrLProib_i$ é o número de vezes em que ocorreu troca de linhas proibida durante a jornada i e λ_5 a penalização desse quesito. Uma troca de linha é proibida se as linhas envolvidas pertencerem a grupos diferentes.
- (xiv) $numTrPProib_i$ é o número de vezes em que ocorreu troca de pontos proibida durante a jornada i e λ_6 a penalização desse quesito.
- (xv) $excessoDuplaPeg$ é a quantidade de jornadas classificadas como de dupla pegada que supera um certo número previamente definido pela empresa. λ_7 é a penalização correspondente.
- (xvi) $numTripulantes$ é o número de tripulantes escalados para executar o conjunto de tarefas. δ é a penalização correspondente.

5.4 Avaliação de uma solução - 2ª Abordagem

Na programação dos veículos, e isso não é diferente na programação das tripulações, existem requisitos operacionais que precisam ser satisfeitos e requisitos de qualidade os quais devem ser minimizados. Esses requisitos podem ser diferentes de acordo com o caso estudado. As seções 5.4.1 e 5.4.2 apresentam quais os requisitos considerados neste trabalho e como esses requisitos são computados. Nós acreditamos que os requisitos considerados nessa abordagem podem ser aplicados em todos os casos práticos.

O custo de uma solução para o PPVT é calculado de acordo com a fórmula (1), a qual deve ser minimizada. Nesta formula, $f_v(s)$ representa a componente f que avalia a solução s com respeito à programação dos veículos e $f_t(s)$ avalia a solução s com respeito a programação das tripulações. Observa-se que tal função é, ao contrário da anterior, baseada em custos, tal como em Reis (2007).

$$f(s) = f_v(s) + f_t(s) \quad (1)$$

5.4.1 Avaliação dos veículos

Quanto menor é o numero de veículos necessários para que a empresa faça todas as viagens de sua responsabilidade, melhor é a solução. Outro ponto é, quanto menor é o tempo em que um veículo está ocioso (tempo ocioso) e quanto menor é o tempo em que um veículo efetua viagens as quais não pertencem ao conjunto de viagens de sua responsabilidade (tempo de viagem morta), melhor é a solução. Assim, nesta abordagem nós tentamos minimizar os seguintes itens referentes à programação de veículos:

- Número de veículos
- Tempo ocioso
- Tempo de viagem morta

Existe uma restrição obvia sobre a programação de veículos: é impossível que um veículo efetue mais que uma viagem ao mesmo tempo. Como a representação da solução permite que isso ocorra, é necessário especificar esta restrição. Nesta abordagem nós consideramos também uma restrição específica para as empresas de transporte público consideradas, uma solução não é válida se mais que 60% do número de veículos voltam à garagem durante, digo nem no início nem no final, sua jornada de trabalho (pegada dupla). Assim, uma programação de veículos é factível se:

- Não há conflitos de viagens na programação dos veículos
- O número de pegadas duplas é menor que 60% do número de veículos

O custo da programação de veículos é dado pela seguinte fórmula:

$$f_v(s) = f(s_v) = \sum_{i \in Q_v} \alpha_i q_i^v(s_v) + \sum_{i \in O_v} \gamma_i o_i^v(s_v) \quad (2)$$

Nesta formula: Q_v e O_v são requisitos de qualidade e operacionais da programação de veículos, respectivamente; α e γ são penalidades aplicadas aos requisitos de qualidade e operacionais da solução s_v , respectivamente; e q_i^v e o_i^v são valores do i -ésimo requisito de qualidade e operacional, respectivamente.

5.4.2 Avaliação das tripulações

As minimal is the number of crews that are necessary for the enterprises to make the set of trips, better is the solution. Another point is, as minimal is the time that a crew is idle and as minimal is the time that a crew is in overtime payment, better is the solution too. The overtime of a duty is given by the time that supers seven hours of work, i.e., the duration of a crew duty that has one hour of overtime payment is eight hours. Thus, in this work, we try to minimize the follow items about the crew schedule:

Quanto menor é o numero de tripulações necessárias para que a empresa faça todas as viagens de sua responsabilidade, melhor é a solução. Outro ponto é, quanto menor o tempo em que uma tripulação está em hora extra (tempo de hora extra), melhor é a solução também. O tempo de hora extra de uma tripulação é dado pelo tempo que supera 7 (sete) horas de trabalho, isto é, a duração da jornada de uma tripulação que possui uma hora extra é oito horas. Assim, nesta abordagem nós tentamos minimizar os seguintes itens referentes à programação das tripulações:

- Número de tripulações
- Tempo de hora extra
- Tempo ocioso

Assim como na programação dos veículos, na programação das tripulações, há uma restrição óbvia: é impossível que uma tripulação efetue mais que uma tarefa ao mesmo tempo. Além disso, por aspectos legais, no Brasil, uma empresa que possui alguma tripulação trabalhando mais que nove por dia ou não possui no mínimo onze horas entre o final de uma jornada de trabalho e o início da próxima jornada é ilegal e está sujeito a multa. Nesta abordagem também foi considerada uma restrição específica para as empresas de transporte público em questão, uma solução não pode ser aplicada se mais que 25% das tripulações voltam para a garagem durante suas jornadas. Assim, uma programação de tripulações é factível se:

- Não há conflitos entre as tarefas da programação das tripulações
- Não há tripulações trabalhando mais que 9 horas por dia
- Nenhuma tripulação possui menos que 11 horas de descanso entre o fim de uma jornada de trabalho e o início da próxima
- O número de pegadas duplas for menor que 25% do número de tripulações

O custo da programação das tripulações é dado pela fórmula:

$$f_t(s) = f(s_t) = \sum_{i \in Q_t} \beta_i q_i^t(s_t) + \sum_{i \in O_t} \theta_i o_i^t(s_t) \quad (3)$$

Nesta formula: Q_t e O_t são requisitos de qualidade e operacionais da programação de tripulações, respectivamente; β e θ são penalidades aplicadas aos requisitos de qualidade e operacionais da solução s_t , respectivamente; e q_i^t e o_i^t são valores do i -ésimo requisito de qualidade e operacional, respectivamente.

5.4.3 Definindo os pesos da avaliação

O objetivo da função de avaliação é avaliar a qualidade de uma solução. Como visto em seções anteriores, esta função considera requisitos que devem ser minimizados. É fato que alguns desses requisitos são relativamente mais importantes que outros. Para contemplar fielmente a importância relativa desses requisitos, os pesos apresentados nessa abordagem são baseados em custos reais descritos em Reis (2007). A Tabela 5.1 apresenta os dados utilizados no cálculo dos pesos da função de avaliação.

Tabela 5.1 – Dados usados no calculo dos pesos – 2ª Abordagem

Descrição	Valor
Pagamento do motorista (mês)	R\$ 1.000,99
Pagamento do auxiliar	R\$ 600,59
Pagamento da tripulação	R\$ 1.601,58
Pagamento + taxas (38%)	R\$ 2.210,18
Dias trabalhos por mês	24
Tempo de trabalho durante um dia	430
Pagamento do minuto de trabalho	R\$ 0,21
Pagamento de um minuto de hora extra	R\$ 0,32
Pagamento de um dia de trabalho	R\$ 92,09
Custo fixo do veículo	R\$ 13.415,58
Dias trabalhados dos veículos	30
Custo diário de um veículo	R\$ 447,19
Velocidade média (km/h)	20
Custo por quilometro	R\$ 2,64
Custo de um minuto de viagem morta	R\$ 0,88
Custo variável por quilometro	R\$ 0,78
Custo pelo minuto ocioso do veículo	R\$ 0,26

A Tabela 5.2 apresenta os pesos utilizados, nessa abordagem, para a avaliação de uma solução do PPVT. Note que, os pesos de qualidade são baseados nos custos reais apresentados na Tabela 5.1, já os custos operacionais são empíricos e seus valores são maiores que os de qualidade. A última coluna desta tabela nos fornece um exemplo das características de uma solução exemplo. O custo dessa solução exemplo, de acordo com a esta abordagem, é 4450,84.

Tabela 5.2 – Pesos da função de avaliação – 2ª Abordagem

Tipo	Referente a	Requisito	Peso	Exemplo de s
Qualidade	Veículos	Número de veículos	447,19	5
		Tempo ocioso	0,26	1700
		Tempo de viagem morta	0,88	161
	Tripulações	Número de tripulações	92,09	11
		Tempo ocioso	0,21	1702
		Tempo de hora extra	0,32	815
Operacional	Veículos	Tempo de conflito	4000	0
		Excesso de pegadas duplas	1000	0
	Tripulações	Tempo de conflito	4000	0
		Excesso de pegadas duplas	1000	0
		Tempo insuficiente entre jornadas	4000	0
		Excesso de tempo de trabalho	4000	0

5.5 Geração de uma solução inicial para o PPV

A geração de uma solução inicial para o PPVT é feita de forma sequencial. Constrói-se uma solução para os veículos e, posteriormente, constrói-se uma solução para as tripulações.

5.5.1 Método Guloso aplicado à geração de uma solução inicial para o PPV

Uma solução inicial para o PPV é gerada por um procedimento construtivo guloso. Esse procedimento consiste na alocação a cada passo de uma nova viagem de responsabilidade da empresa na lista de viagens de algum veículo de sua frota, de tal forma que esta combinação seja a que implique no menor custo de operação possível determinado pela aplicação da função de avaliação descrita na seção 5.3.

O pseudocódigo do método guloso considerado é descrito a seguir na Figura 5.4.

Algoritmo SIG_V

Entrada: $f(\cdot)$, X , F
Seja X o conjunto de viagens a serem realizadas
Seja F uma frota de veículos, ainda sem viagens a executar
Seja s_v uma solução para os veículos
 $s_v \leftarrow F$
Para cada viagem $x \in X$ faça
 $melhorDelta \leftarrow \infty$
 Para cada veículo $k \in s_v$ faça
 $custoAnterior \leftarrow$ custo do veículo k antes de receber a viagem x
 Aloque a viagem x no veículo k
 $custoPosterior \leftarrow$ custo do veículo k após inserir a viagem x
 Remova a viagem x do veículo k
 $\Delta = custoPosterior - custoAnterior$
 Se ($\Delta < melhorDelta$) então
 $melhorVeículo \leftarrow k$
 $melhorDelta \leftarrow \Delta$
 fim se
 fim para
 Aloque a viagem x ao veículo indicado por $melhorVeículo$
 $fo \leftarrow fo + melhorDelta$, ou seja, atualize o valor da função de avaliação
fim para
Retorne s_v

Figura 5.2 – Método Gerador de Solução Inicial para os Veículos

Para exemplificar como construir uma solução para o PPV pelo procedimento proposto, suponha que a empresa que deve executar as viagens da Tabela 4.1 possua dois veículos disponíveis. A seguinte seqüência de passos mostra o funcionamento do procedimento.

Passo 0: Constrói-se uma lista com o número de veículos da empresa. A seguir, aloca-se a primeira viagem da lista de viagens, ordenada por horário de início, ao veículo que gera o menor custo. Nesse caso, a Viagem 0 é alocada ao Veículo 0.

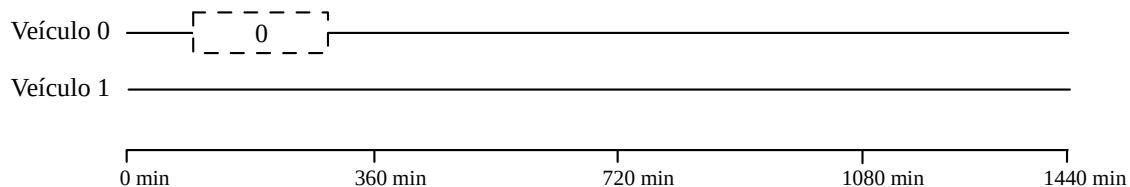


Figura 5.3 –Exemplo da geração de uma solução inicial para os veículos: Passo 0

Passo 1: Aloca-se a segunda viagem ao veículo que gera o menor custo de alocação. Nesse caso, a Viagem 1 é alocada ao Veículo 1, pois a alocação dessa viagem ao Veículo 0 geraria uma sobreposição da Viagem 0 com a Viagem 1, gerando um custo de alocação muito elevado de acordo com a função de avaliação f_T descrita na seção 5.3.

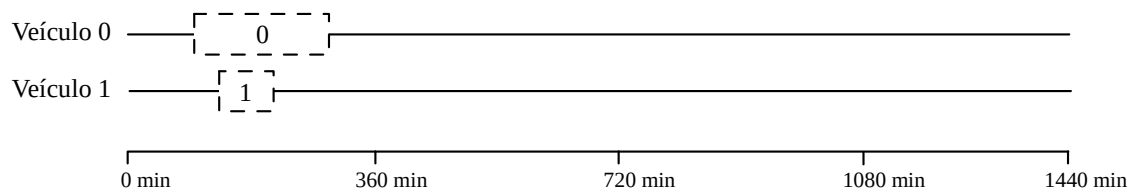


Figura 5.4 – Exemplo da geração de uma solução inicial para os veículos: Passo 1

Passo 2: Aloca-se a terceira viagem da lista ao veículo que gera o menor custo de alocação. No caso, a Viagem 2 é alocada ao Veículo 0, pois se ela fosse alocada ao Veículo 1 o tempo ocioso seria maior.

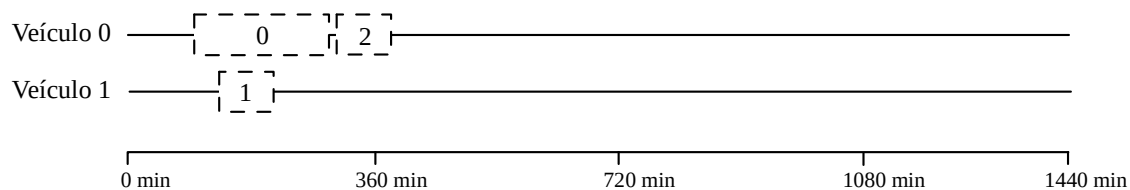


Figura 5.5 – Exemplo da geração de uma solução inicial para os veículos: Passo 2

Passo 3: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 3 é alocada ao Veículo 1. Essa viagem não poderia ser alocada ao Veículo 0 pois, neste caso, haveria sobreposição com a Viagem 2.

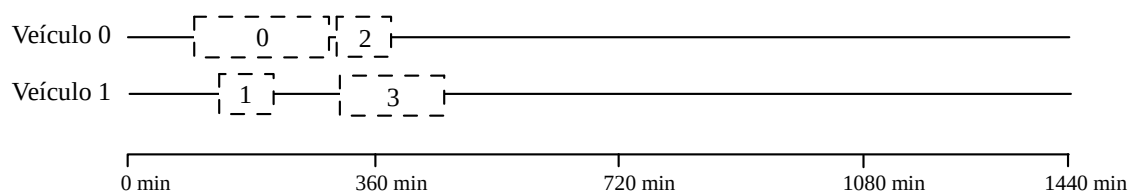


Figura 5.6 – Exemplo da geração de uma solução inicial para os veículos: Passo 3

Passo 4: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Neste caso, a Viagem 4 é alocada ao Veículo 1, uma vez que tal alocação resulta em menor tempo de ociosidade.

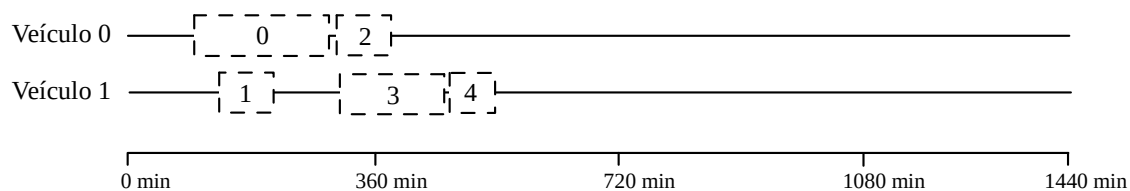


Figura 5.7 – Exemplo da geração de uma solução inicial para os veículos: Passo 4

Passo 5: Aloca-se a próxima viagem da lista ao veículo que gera o menor custo de alocação. Nesse caso, considerando que haveria sobreposição de viagens com a alocação da Viagem 5 ao Veículo 1, a Viagem 5 é alocada ao Veículo 0.

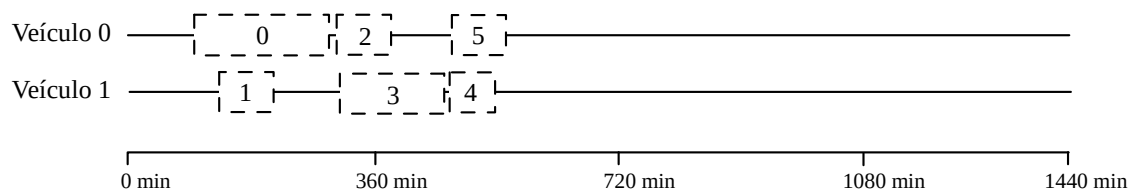


Figura 5.8 – Exemplo da geração de uma solução inicial para os veículos: Passo 5

Passo 6: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 6 é alocada ao Veículo 0.

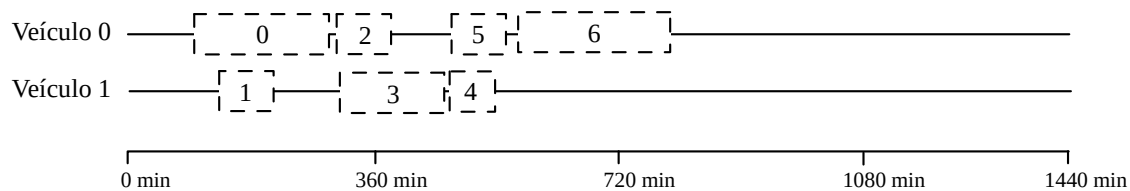


Figura 5.9 – Exemplo da geração de uma solução inicial para os veículos: Passo 6

Passo 7: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 7 é alocada ao Veículo 1.

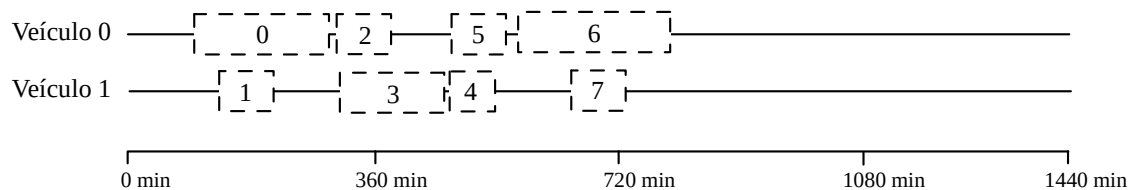


Figura 5.10 – Exemplo da geração de uma solução inicial para os veículos: Passo 7

Passo 8: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 8 é alocada ao Veículo 1.

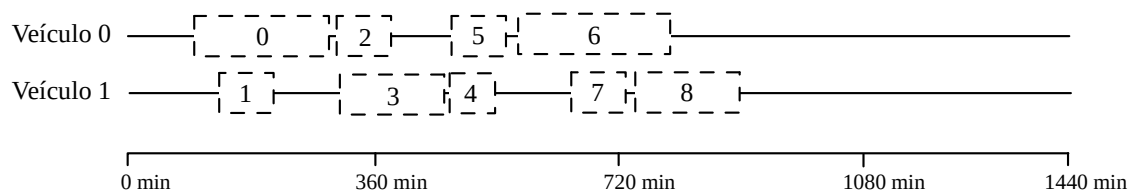


Figura 5.11 – Exemplo da geração de uma solução inicial para os veículos: Passo 8

Passo 9: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 9 é alocada ao Veículo 0. Como não há mais viagens a serem alocadas, encerra-se o procedimento de construção da solução inicial para os veículos.

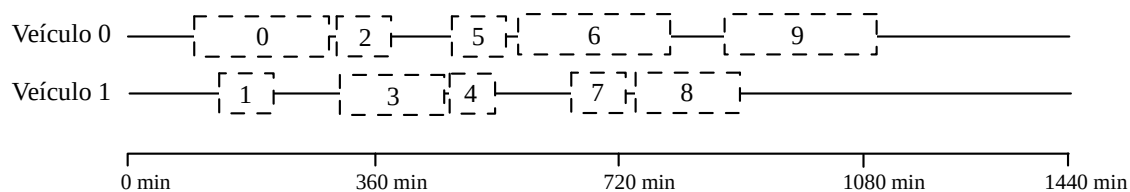


Figura 5.12 – Exemplo da geração de uma solução inicial para os veículos: Passo 9

5.5.2 Método Guloso aplicado à geração de uma solução inicial para o PPT

Nesta forma de construção, considera-se uma lista de tarefas T_i ($i = 0, 1, 2, 3, \dots$), ordenadas pela hora de início, e uma lista de jornadas, onde em cada passo, há sempre uma jornada sem nenhuma tarefa. O método começa com a alocação da tarefa T_0 à primeira jornada. A seguir, para cada tarefa T_i ($i = 1, 2, 3, \dots$) da lista de tarefas, calcula-se o custo da alocação de T_i em cada uma das jornadas já com alguma tarefa alocada e mais uma jornada sem nenhuma tarefa. O custo da alocação é calculado tendo em vista a expressão (8) da seção 5.3.2. A seguir, aloca-se a tarefa T_i à jornada que gerou o menor custo.

O pseudocódigo do método guloso considerado é descrito a seguir na Figura 5.5.

Algoritmo SIG_T

```

Entrada:  $f(\cdot)$ ,  $T$ 
Seja  $T$  o conjunto de tarefas a serem executadas
Seja  $s_T$  uma solução para as tripulações
 $s_T \leftarrow \emptyset$ 
Adicione uma jornada a  $s_T$ 
para cada  $t \in T$  faça
     $melhorDelta \leftarrow \infty$ 
    para cada jornada  $j \in s_T$  faça
         $custoAnterior \leftarrow$  custo da jornada  $j$  antes de receber a tarefa  $t$ 
        Aloque a tarefa  $t$  na jornada  $j$ 
         $custoPosterior \leftarrow$  custo da jornada  $j$  após inserir a tarefa  $t$ 
        Remova a tarefa  $t$  da jornada  $j$ 
         $\Delta = custoPosterior - custoAnterior$ 
        Se ( $\Delta < melhorDelta$ ) então
             $melhorJornada \leftarrow j$ 
             $melhorDelta \leftarrow \Delta$ 
    fim se
fim para
se  $melhorJornada$  não possuir nenhuma tarefa
    Adicione uma nova jornada à  $s_T$ 
    Aloque a tarefa  $t$  na jornada indicada por  $melhorJornada$ 
     $fo \leftarrow fo + melhorDelta$ , ou seja, atualize o valor da função de avaliação
fim para
Remova de  $s_V$  a jornada que não possui nenhuma tarefa
Retorne  $s_V$ 

```

Figura 5.13 – Método Gerador de Solução Inicial para as Tripulações

Estabelecida a programação dos veículos, é gerado um conjunto de tarefas a serem executadas pelos tripulantes da empresa. Cada tarefa é um conjunto de viagens reunidas de maneira que haja apenas duas oportunidades de troca: uma no início e outra no final da tarefa. Assim, durante sua realização não é possível a troca de tripulação.

A Figura 5.16 ilustra a lista de tarefas geradas a partir da programação dos Veículos da Figura 5.14.

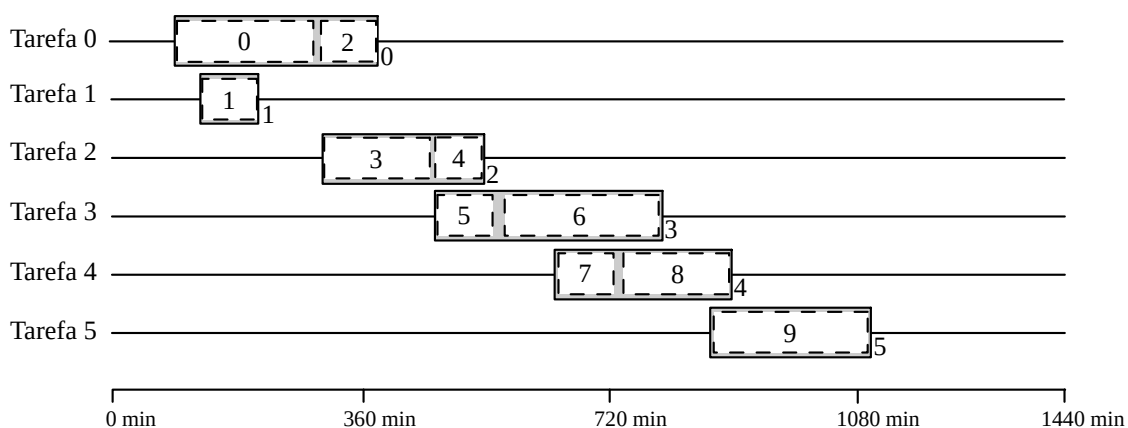


Figura 5.14 – Tarefas geradas pela programação dos veículos da Figura 5.14

Passo 0: Cria-se uma lista com apenas uma jornada, inicialmente sem nenhuma tarefa.

Passo 1: Aloca-se a primeira tarefa da lista de tarefas na única jornada da lista. Posteriormente, adiciona-se uma nova jornada na lista de jornadas. Nesse caso, a Tarefa 0 é adicionada à Jornada 0. Como deve-se sempre avaliar o caso de uma tarefa ser atribuída a uma jornada sem tarefas, a Jornada 1 é adicionada à lista de Jornadas.

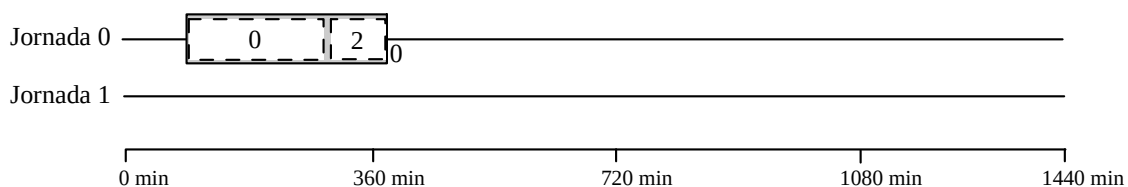


Figura 5.15 – Exemplo da geração de uma solução inicial para as tripulações: Passo 1

Passo 2: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 1 é alocada à Jornada 1. Como não há jornadas vazias, o procedimento cria uma nova jornada, a Jornada 2, que é adicionada à lista de Jornadas.

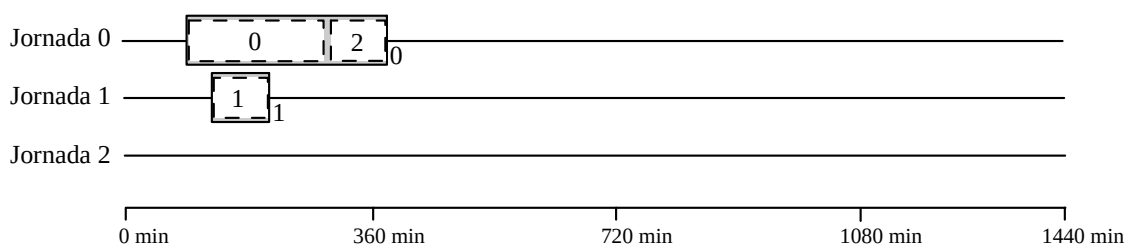


Figura 5.16 – Exemplo da geração de uma solução inicial para as tripulações: Passo 2

Passo 3: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 2 é alocada à Jornada 1, porque ela geraria uma sobreposição se fosse alocada à Jornada 0 e se fosse alocada à Jornada 2 haveria um custo mais elevado por utilizar uma nova jornada.

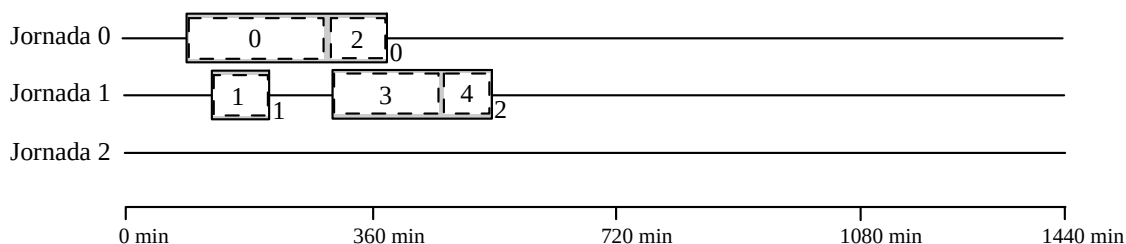


Figura 5.17 – Exemplo da geração de uma solução inicial para as tripulações: Passo 3

Passo 4: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 3 é alocada à Jornada 2.

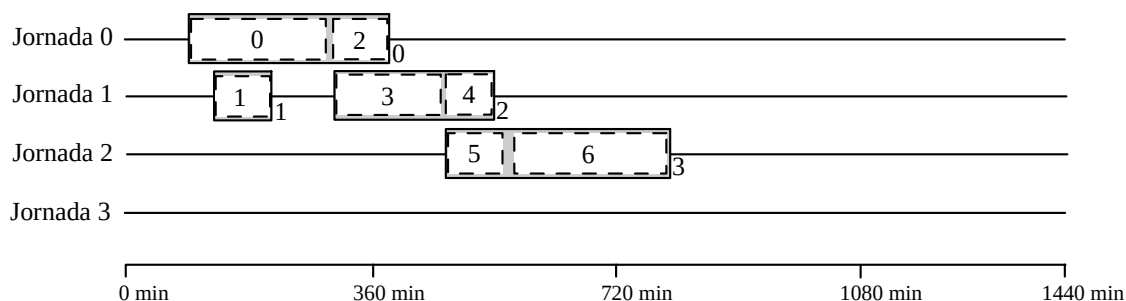


Figura 5.18 – Exemplo da geração de uma solução inicial para as tripulações: Passo 4

Passo 5: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 4 é alocada à Jornada 0.

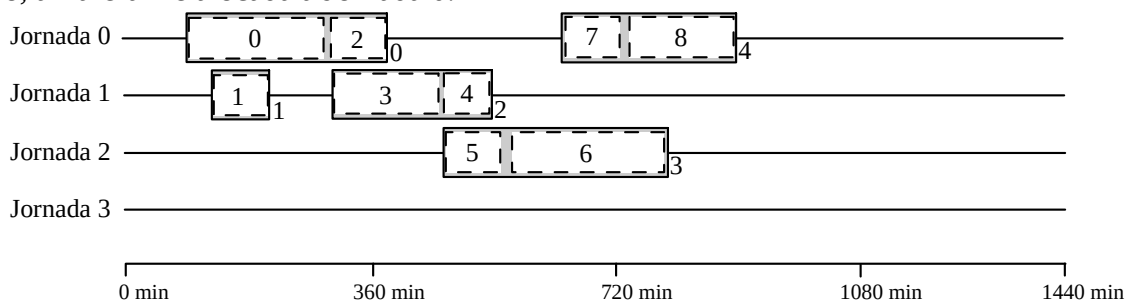


Figura 5.19 – Exemplo da geração de uma solução inicial para as tripulações: Passo 5

Passo 6: Aloca-se a última tarefa da lista na jornada que gera menor custo. Nesse caso, a Tarefa 5 é alocada à Jornada 2 e o procedimento construtivo se encerra.

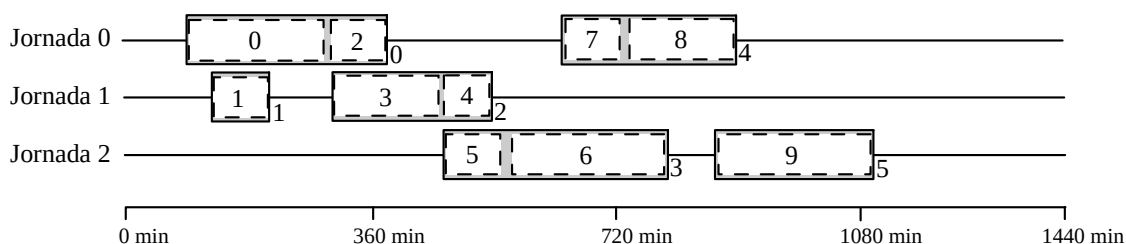


Figura 5.20 – Exemplo da geração de uma solução inicial para as tripulações: Passo 6

5.6 Metaheurísticas aplicadas ao PPVT

Neste trabalho, o refinamento de uma solução é feito pela metaheurística *Iterated Local Search* (ILS) com uma fase de refinamento que combina uma técnica baseada no procedimento *Variable Neighborhood Descent* (VND) em dois níveis com uma heurística clássica e a poderosa metaheurística Busca Tabu com Relaxação Adaptativa.

5.6.1 *Iterated Local Search*

O algoritmo *Iterated Local Search* (ILS) é uma meta-heurística que para escapar das armadilhas dos ótimos locais, efetua perturbações em uma solução s . Esse algoritmo se mostra bem sucedido em várias aplicações, motivando, assim, sua utilização na resolução do PPVT. Este algoritmo é baseado na idéia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local.

Neste trabalho, uma perturbação de um nível i consiste de efetuar $i+2$ movimentos na vizinhança N_V^{RP} escolhidos aleatoriamente. Desta forma, uma perturbação de nível 0 consiste em efetuar 2 movimentos nesta estrutura de vizinhança. Para evitar buscas exaustivas na fase de refinamento do ILS, os movimentos feitos em uma perturbação não geram inviabilidade segundo o quesito sobreposição de viagens.

O pseudocódigo do ILS aplicado ao PPVT é apresentado na Figura 5.22. Em que, $Nívelmax$ é o maior nível de perturbação a ser explorado, $ILSmax$ é o número de iterações em que um mesmo nível de perturbação é explorado, $Tempomax$ é o tempo limite de processamento do ILS e $f(.)$ é uma função que avalia uma solução.

Basicamente, este método possui duas fases a cada iteração: a primeira é a fase de perturbação, e a segunda, a de refinamento. A fase de refinamento do ILS é descrita em detalhes na seção 5.6.2.

Algoritmo ILS_PPVT

```
1:  Entrada:  $s, f(\cdot), \text{Nívelmax}, \text{ILSmax}, \text{Tempomax}$ 
2:   $s \leftarrow \text{VND\_2N}(s)$ 
3:   $s^* \leftarrow s$ 
4:  nível  $\leftarrow 0$ 
5:  enquanto nível <  $\text{Nívelmax}$  faça
6:    iter  $\leftarrow 0$ 
7:    enquanto iter <  $\text{ILSmax}$  e tempo <  $\text{Tempomax}$  faça
8:       $s \leftarrow \text{Perturbação}(s, \text{nível})$ 
9:       $s \leftarrow \text{VND\_2N}(s)$ 
10:     se  $f(s) < f(s^*)$  então
11:        $s^* \leftarrow s$ 
12:       nível  $\leftarrow 0$ 
13:       iter  $\leftarrow 0$ 
14:     senão
15:        $s \leftarrow s^*$ 
16:       iter  $\leftarrow \text{iter} + 1$ 
17:     fim se
18:   fim enquanto
19:   nível  $\leftarrow \text{nível} + 1$ 
20: fim enquanto
21: retorne  $s^*$ 
```

Figura 5.21 – Algoritmo ILS com VND_2N aplicado ao PPVT

5.6.2 Variable Neighborhood Descent

Um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança. Este é um dos princípios básicos do método de descida em vizinhança variável, *Variable Neighborhood Descent* (VND). Proposto por Nenad Mladenovic e Pierre Hans, este é um método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

Neste trabalho, o algoritmo proposto para o refinamento de uma solução após a perturbação causada no ILS é baseado na técnica VND. Aqui, o algoritmo, além de alterar a estrutura de vizinhança, também altera o método de refinamento utilizado. Esse algoritmo consiste em uma aplicação em dois níveis, um externo e um interno, da idéia VND. A Figura 5.9 é uma representação descritiva do algoritmo proposto para o refinamento do ILS.

As duas linhas dos passos 1, 2 e 3 do algoritmo em questão constituem o VND interno. No passo 1 aplica-se a uma solução s_0 o método DR2VP utilizando a vizinhança N_V^{RP} resultando uma solução s_1 . Como DR2VP é um método de descida, a solução s_1 ou é melhor que a solução s_0 ou é a própria solução s_0 . Visto que este é um problema de minimização, uma solução é melhor que outra se o valor de sua função de avaliação (seção 5.3) é inferior ao valor da função de avaliação da outra solução. E então, a solução s_1 é atribuída à solução s_0 . O algoritmo continua com a chamada do método DR2VP usando a estrutura de vizinhança N_V^{TP} , gerando a solução s_2 . Se o custo da solução s_2 for inferior ao custo de s_0 , então se inicia novamente a execução do passo 1. O primeiro VND interno termina quando os métodos DR2VP não produzirem melhoras nas soluções. A execução dos passos 2 e 3 é semelhante a deste passo: no passo 2 o refinamento de uma solução é feito por meio do método BTRA

(Busca Tabu com Relaxação Adaptativa), primeiro utilizando a vizinhança N_T^R e depois, a vizinhança N_T^T ; e no passo 3, o refinamento é feito pelo método DR2VI usando a vizinhança N_V^{RP} e depois a N_V^{TP} .

Alg.	VND_2N
Início	Seja s_0 uma solução
Passo 1	$s_1 \leftarrow \text{VDR}(s_0, \text{VDRmax}, N_V^R, N_V^{RP}, f_v)$ $s_2 \leftarrow \text{VDR}(s_1, \text{VDRmax}, N_V^T, N_V^{TP}, f_v)$ Se s_2 melhor que s_0 , faça $s_0 = s_2$ e volte ao início desse passo
Passo 2	$s_3 \leftarrow \text{BTRA}(s_2, \text{BTtempo}, N_T^R, f_l)$ $s_4 \leftarrow \text{BTRA}(s_3, \text{BTtempo}, N_T^T, f_l)$ Se s_4 melhor que s_2 , faça $s_2 = s_4$ e volte ao início desse passo
Passo 3	$s_5 \leftarrow \text{IRD}(s_4, \text{IDRmax}, N_V^R, N_V^{RP}, f, \lambda)$ $s_6 \leftarrow \text{IRD}(s_5, \text{IDRmax}, N_V^R, N_V^{RP}, f, \lambda)$ Se s_6 melhor que s_4 , faça $s_4 = s_6$ e volte ao início desse passo Se houve alguma melhora no Passo 3, volte ao Passo 2
Passo 4	Retorne s_6

Figura 5.22 – Algoritmo VND em dois níveis como refinamento do ILS

O VND externo consiste em tomar passos como tipos de vizinhança no VND original (seção 3.3.2). A terceira linha do passo 3 verifica se, nesse passo, houve alguma melhora na solução. Caso haver, o método inicia novamente sua busca a partir da primeira estrutura, aqui a partir do passo 2. O VND externo é feito somente utilizando os passos 2 e 3.

5.6.3 Busca Tabu com Relaxação Adaptativa

Para o refinamento de uma solução PPVT utilizando Busca Tabu com Relaxação Adaptativa adaptou-se o método BTAR-FI-IC proposto por Marinho (2005) para a abordagem do problema integrado. Este método, cujo pseudocódigo é apresentado na Figura 5.24, consiste no método Busca Tabu acrescido do mecanismo de intensificação Relaxação Adaptativa.

O mecanismo de Relaxação Adaptativa utilizado pelo método BTAR-FI-IC é baseado naquele descrito por Schaerf (1996), onde os pesos para cada fonte de inviabilidade são ajustados dinamicamente, como proposto originalmente por Gendreau *et al.* (1994). Para cada inviabilidade i o peso β_i é multiplicado por um fator σ_i que varia de acordo com o seguinte esquema:

- No início da busca $\sigma_i \leftarrow 1$.
- A cada k movimentos:
 - se todas as k soluções visitadas são factíveis em relação à restrição i então $\sigma_i \leftarrow \sigma_i / \gamma$;
 - se todas as k soluções visitadas são infactíveis em relação à restrição i então $\sigma_i \leftarrow \sigma_i \times \gamma$;
 - se algumas soluções são factíveis e outras são infactíveis, considerando a restrição i , então σ_i permanece inalterado.

O parâmetro γ é randomicamente selecionado, a cada k movimentos, no intervalo $[1,8; 2,2]$ conforme proposto por Schaerf (1996). Cada valor de σ_i é limitado por duas constantes σ_{min} e σ_{max} , de forma a evitar que a relaxação adaptativa incremente/decremente indefinidamente os pesos para restrições que são sempre insatisfeitas/satisfeitas.

Algoritmo Busca Tabu com Relaxação Adaptativa

```

1:  Entrada:  $s, BTtempo, N_t (N_t^R \text{ ou } N_t^I), f_t(.)$ 
2:   $s^* \leftarrow s;$                                      { Melhor solução até então }
3:   $ListaTabu \leftarrow \emptyset;$                        { Lista Tabu}
4:   $Iter \leftarrow 0;$                                    { Contador do número de iterações }
5:  repita
6:    Defina um subconjunto  $V \subset N_t(s);$ 
7:     $melhorMovimento \leftarrow movimentoRandomico(V);$    { Melhor Movimento em  $V \subset N(s)$  }
8:     $melhorCustoDinamico \leftarrow \infty;$                { Melhor custo dinâmico }
9:    para todo ( movimento  $m \in V$  ) faça
10:     se (  $f(s \oplus m) < f(s^*)$  ) então
11:        $melhorMovimento \leftarrow m;$ 
12:       interromper;
13:     senão
14:       se (  $m \notin ListaTabu$  ) então
15:         se (  $f(s \oplus m) < f(s)$  ) então
16:            $melhorMovimento \leftarrow m;$ 
17:           interromper;
18:         senão
19:           se (  $custoPorPesosDinamicos(s \oplus m) < melhorCustoDinamico$  ) então
20:              $melhorMovimento \leftarrow m;$ 
21:              $melhorCustoDinamico \leftarrow custoPorPesosDinamicos(s \oplus m);$ 
22:           fim-se
23:         fim-se
24:       fim-se
25:     fim-se
26:   fim-para
27:    $s \leftarrow s \oplus melhorMovimento;$ 
28:   se (  $f(s) < f(s^*)$  ) então
29:      $s^* \leftarrow s;$ 
30:   fim-se
31:    $AtualizarListaTabu();$ 
32:   se  $IteracaoAtualizacao()$  então
33:      $AtualizarPesosDinamicos();$ 
34:   fim-se
35: até que ( Critério de parada seja satisfeito )
36: retorne  $s^*;$ 

```

Figura 5.23 – Algoritmo Busca Tabu com Relaxação Adaptativa aplicado ao PPVT
 FONTE: Marinho (2005)

5.6.4 O procedimento *Vehicle Random Descent*

Para problemas grandes, encontrar o melhor vizinho em cada iteração de um método de descida é muito custoso. Nesses casos, normalmente são utilizadas abordagens randômicas. O clássico procedimento Descida Randômica (DR) funciona dessa forma: é gerado um ponto aleatório s' em uma dada vizinhança. Se s' for melhor que a solução atual, então s é atualizada ($s \leftarrow s'$) e a busca continua de s' . Se não houver melhora em $RDmax$ iterações, então a busca é finalizada. Note que o procedimento pode não retornar um ótimo local.

Quando se trata de problemas integrados, encontrar o melhor vizinho a cada iteração é uma operação ainda mais custosa. Isto se deve ao fato de que, nas vizinhanças integradas consideradas (N_v^{RP} e N_v^{TP}), cada modificação nos veículos tem reflexo na programação das tripulações. Para reduzir o número de propagações dessas modificações na programação das tripulações, propõe-se o procedimento *Vehicle Random Descent* (VRD), descrito na Figura 5.24.

O princípio de funcionamento desse algoritmo é bem simples e lembra bastante o clássico Descida Randômica. A principal diferença é que nesse procedimento, apenas as modificações feitas nos veículos (realocação ou troca) que melhoram a função de avaliação f_v são propagadas na programação das tripulações.

Na Figura 5.25, s é uma solução formada pelo par de soluções s_v e s_t , $VRDmax$ é o número máximo de iterações sem melhora, N_v é uma estrutura de vizinhança (realocação ou troca) específica para os veículos, m é o movimento realizado na estrutura considerada, N_v^P é a estrutura de vizinhança (realocação ou troca) com a propagação segundo a modificação m e f_v é a função de avaliação que avalia a parte s_v da solução s .

Observa-se que a solução final desse algoritmo pode, não necessariamente, ser melhor que a solução inicial em relação à função integrada f uma vez que só é usada a função de avaliação de veículos f_v .

Algoritmo VRD

```

1:  Entrada:  $s$ ,  $VRDmax$ ,  $N_v(\cdot)$ ,  $N_v^P(\cdot)$ ,  $f_v(\cdot)$ 
2:  Saída: Solução  $s^*$  melhor ou igual à solução  $s$  de acordo com a função  $f_v$ 
3:   $iter \leftarrow 0$ 
4:  enquanto  $iter < VRDmax$  faça
5:     $m \leftarrow movimentoAleatorio(s, N_v)$ 
6:     $s' \leftarrow vizinho(s, m)$ 
7:    se  $s'$  for melhor que  $s$  de acordo com  $f_v$  faça
8:       $s \leftarrow vizinho(s, N_v^P, m)$ 
9:       $iter \leftarrow 0$ 
10:   fim se
11:    $iter \leftarrow iter + 1$ 
12: fim enquanto
13:  $s^* \leftarrow s$ 
14: retorne  $s^*$ 

```

Figura 5.24 – Procedimento *Vehicle Random Descent*

5.6.5 O procedimento *Integrated Vehicle and Crew Random Descent Procedure*

Esta estratégia é semelhante à apresentada na seção anterior. Há duas diferenças básicas. A primeira é que são candidatos apenas os movimentos que produzirem soluções, com respeito aos veículos, melhores que f_v ou piores que f_v em até $(\lambda - 1) \times 100\%$. A segunda, é que se considera o custo tanto de s_v quanto de s_t , o que não acontece no *Vehicle Random Descent* que, quando aplicado à resolução do PPVT, considera apenas o custo de s_v . O objetivo desse algoritmo é refinar uma solução s usando movimentos integrados sem onerar demasiadamente o processamento, como aconteceria se o refinamento fosse feito pelo algoritmo clássico descida randômica usando unicamente movimentos integrados.

A Figura 5.26 apresenta o *Integrated Vehicle and Crew Random Descent Procedure* (IRD). Nesta figura, s é uma solução formada pelo par de soluções s_v e s_t compatíveis entre si, $IRDmax$ é o número de iterações sem melhora em f , m é um movimento realizado nos veículos de acordo com a vizinhança N_v (realocação ou troca), N_v^P é a estrutura de vizinhança

(realocação ou troca) com propagação segundo a modificação m , f_v é a função que avalia s_v e λ ($\lambda \geq 1$) é um valor usado para determinar o quanto uma solução dos veículos pode piorar para que valha a pena efetuar a parte mais custosa do refinamento, a construção e a propagação do movimento integrado. Assim, esse valor determina se é um momento propício para testar o movimento com propagação. $\lambda = 1,01$ implica que se um movimento gerar uma solução s_v' de custo maior que 101% do custo de s_v , o método não testa o movimento de propagação; caso contrário, o teste é realizado e, se a solução s' resultante do movimento de propagação for melhor que a solução s , então a solução s é atualizada, $s \leftarrow s'$. Testes realizados mostraram que, para as instancias analisadas, 1,01 é um bom valor para λ e este é o valor utilizado neste trabalho.

Algoritmo IRD

```

1:  Entrada:  $s$ ,  $IRDmax$ ,  $N_v(\cdot)$ ,  $N_v^P(\cdot)$ ,  $f(\cdot)$ , Real  $\lambda$ 
2:  Saída: Solução  $s^*$  melhor ou igual à solução  $s$  de acordo com a função  $f$ 
3:   $iter \leftarrow 0$ 
4:  enquanto  $iter < IRDmax$  faça
5:     $m \leftarrow movimentoAleatorio(s, N_v)$ 
6:     $s' \leftarrow vizinho(s, m)$ 
7:    se  $f_v(s') \leq f_v(s) \times \lambda$  faça
8:       $s'' \leftarrow vizinho(s, N_v^P, m)$ 
9:      se  $s''$  for melhor que  $s$  de acordo com  $f$  faça
10:         $s \leftarrow s''$ 
11:       $iter \leftarrow 0$ 
12:    fim se
13:  fim se
14:   $iter \leftarrow iter + 1$ 
15: fim enquanto
16:  $s^* \leftarrow s$ 
17: retorne  $s^*$ 

```

Figura 5.25 – Procedimento *Integrated Vehicle and Crew Random Descent*

6 Resultados

Apresentam-se, nesse capítulo, os resultados obtidos aplicando-se a heurística ILS com VND em dois níveis utilizando Busca Tabu com Relaxação Adaptativa, denotada por ILS-VND-BTRA, conforme descrito na seção 5.6.

O algoritmo descrito foi implementado na linguagem C++ usando o compilador Borland C++ Builder 6.0 e testado em um microcomputador com processador Intel® Pentium®, 3.00 GHz, com 1 GB de memória RAM, sob sistema operacional Windows XP Professional Service Pack 2.

Para validar as abordagens foram disponibilizadas pela empresa responsável pelo gerenciamento do Sistema de Transporte Público da cidade de Belo Horizonte, instâncias reais pertencentes a uma Bacia de Linhas de tal cidade.

A seção 6.1 apresenta os resultados obtidos utilizando a primeira abordagem da função de avaliação, enquanto na seção 6.2 apresentam-se os resultados obtidos considerando a segunda abordagem da função de avaliação.

6.1 Resultados – 1ª Abordagem

Nas tabelas 6.1 e 6.2 são apresentados os valores dos parâmetros utilizados para o cálculo da função de avaliação apresentada na seção 5.3, a qual é baseada em pesos empíricos. Na Tabela 6.2 os valores *TempDesloc_Lim* e *TempTerm_Lim* são vinculados a cada instância e representam os maiores tempos de viagem morta e terminal, nessa ordem, em minutos, encontrados na solução gerada pela empresa.

Tabela 6.1 – Parâmetros utilizados na função de avaliação do PPVT

Parâmetro	Valor
<i>MaxTimeTerm</i>	119 minutos
α_1	2
β_1	$\ln(16) / \ln(\alpha_1 \times \text{TempDesloc\_Lim})$
α_2	1
β_2	$\ln(16) / \ln(\alpha_2 \times \text{TempTerm\_Lim})$
ω_{TD}	50
ω_{TT}	40
ω_{TS}	80
ω_{EDP}	80
ω_{TL}	5
ω_{NV}	0
μ_1	18
μ_2	40
μ_3	19
μ_4	5
μ_5	5
λ_1	40
λ_2	40
λ_3	40
λ_4	45
λ_5	35
λ_6	35
λ_7	45
δ	100

Tabela 6.2 – Instâncias e seus respectivos tempos limites relativos ao PPV

Instâncias	<i>TempDesloc_Lim</i>	<i>TempTerm_Lim</i>
G02_DOM	18	73
G02_SEG	18	118
G02_SEX	18	113
G02_SAB	18	62
G27_DOM	23	110
G27_SEG	23	47
G27_SEX	23	42
G27_SAB	23	57

No cálculo de β_1 e β_2 da Tabela 6.1, o valor 16 corresponde à penalidade máxima atribuída a uma viagem morta ou tempo de espera no terminal que obedece ao limite estabelecido por *TempDesloc_Lim* ou *TempTerm_Lim* de acordo com o que se esteja penalizando. Ou seja, trata-se do valor máximo da função exponencial utilizada no cálculo da função de avaliação.

Para as empresas em questão, considerou-se o número de duplas pegadas na programação dos veículos inferior a 60% do tamanho da frota; enquanto que na programação dos tripulantes esse número foi considerado inferior a $1,25 \times \text{tamanho da frota de veículos}$.

Esta última consideração é uma estimativa para o número de tripulantes, que corresponde a 2,5 tripulantes para cada veículo.

Os parâmetros utilizados no método ILS-VND-BTRA, utilizado para a abordagem integrada, foram, na chamada do método ILS, $Tempomax = 30$ minutos, $Nivelmax = 5$ e $ILSmax = 50$; $Itermax = total\ de\ viagens \times tamanho\ máximo\ da\ frota\ de\ veículos$ nas chamadas dos métodos DR2VP e DR2VI. O valor da constante ρ , parâmetro do método DR2VI, é fixado em 1,2. Os pesos α e β relativos à programação dos veículos e das tripulações como descrito na seção 5.3, são fixados em 1,0 e 2,5, respectivamente.

Para os métodos destinados a resolver os problemas de programação de veículos e tripulações de forma seqüencial o tempo de processamento foi limitado a 15 minutos para cada fase, isto é, 15 minutos para a resolução do PPV pelo método ILS proposto em Simões *et al.* (2006) e 15 minutos para a resolução do PPT pelo método baseado em Busca Tabu, proposto em Marinho *et al.* 2004.

A Tabela 6.3 mostra uma comparação entre os resultados produzidos pelas metodologias, seqüencial e integrada, em termos da função de avaliação apresentada na seção 5.3. Foram analisadas duas das empresas que operam na Bacia contemplada, considerando os quatro dias da semana que apresentam quadros de horários distintos: segunda-feira, sexta-feira, sábado e domingo. Nesta tabela, considera-se que “FO” é o custo da programação realizada pela empresa e “Melhor FO”, “Média FO”, “Desvio”, “PM” e “PMM” são, respectivamente, o menor custo encontrado pelo método considerado, a média dos custos em 10 execuções, o percentual de desvio calculado conforme fórmula a seguir.

$$desvio = \frac{MediaFO - MelhorFO}{MelhorFO} \quad (12)$$

Tabela 6.3 – Comparação de resultados (Seqüencial x Integrada)

Instancia	Algoritmo Seqüencial			Algoritmo Integrado		
	Melhor FO	Média FO	Desvio	Melhor FO	Média FO	Desvio
G02_DOM	7036	7160,35	6,47%	6725,5	6841,5	1,72%
G02_SEG	22507,5	22828,55	8,26%	21087,5	21297,7	1,00%
G02_SEX	23045,5	23354,1	7,50%	21724,5	21910,1	0,85%
G02_SAB	12938,5	13260,5	8,51%	12220	12468,05	2,03%
G27_DOM	3497,5	3550,75	4,51%	3397,5	3442,3	1,32%
G27_SEG	8299	8463,25	7,30%	7887,5	7992,55	1,33%
G27_SEX	8429,5	8486,25	7,59%	7887,5	8015,55	1,62%
G27_SAB	5193,5	5346,65	5,73%	5057	5176,35	2,36%

Pelos resultados exibidos na Tabela 6.3, verifica-se que, quanto aos desvios, a metodologia integrada proposta apresenta-se superior à metodologia seqüencial em todos os problemas teste. Assim, em todos os dias analisados, o método proposto foi capaz de gerar, em média, soluções melhores que a da metodologia seqüencial, sob o ponto de vista da função de avaliação.

Nos domingos da G02, a metodologia integrada mostrou-se melhor por apresentar um desvio de 1,72%, muito inferior ao desvio de 6,47% da metodologia seqüencial. Nas segundas-feiras da mesma empresa, a metodologia integrada mostrou-se melhor por apresentar um desvio de 1,00%, inferior ao desvio de 8,26% da metodologia seqüencial. Nas

sextas-feiras da G02, a metodologia integrada mostrou-se melhor por apresentar um desvio de 0,85%, inferior ao desvio de 7,50% da metodologia seqüencial. E, nos sábados da G02, a metodologia integrada mostrou-se melhor por apresentar um desvio de 2,03%, inferior ao desvio de 8,51% da metodologia seqüencial. Para a empresa G27 o método integrado também obteve melhoras significativas quanto ao seqüencial.

O método integrado proposto foi capaz de gerar soluções melhores em todos os oito problemas teste (domingos, segundas, sextas e sábados das empresas G02 e G27). Além disso, ao se analisar os valores de desvio apresentados na Tabela 6.3 para a integração observa-se que eles são baixos. Este é um ponto importante, uma vez que não há grandes discrepâncias entre os custos encontrados para uma mesma instância.

As tabelas 6.4 e 6.5 mostram, detalhadamente, as características das melhores soluções geradas pelas metodologias utilizadas nos quatro dias da semana da empresa G02 e da empresa G27, respectivamente, em relação ao total de viagens a serem realizadas “NVIAG”, número de veículos utilizados “NVEIC”, tempo de espera no terminal “TE”, tempo de viagem morta “TVM”, número de duplas pegadas dos veículos “NDP(v)” e dos tripulantes “NDP(t)”, número de tripulantes “NTRIP”, tempo ocioso “TO”, tempo de hora extra “THE”, número de trocas de linha “NTL”, número de trocas de ponto “NTP”, número de trocas de veículos “NTV” função objetivo de veículos “f(v)”, função objetivo de tripulações “f(t)”, função objetivo de veículos e tripulações “f(v,t)”.

Os números em negrito apontados nessas tabelas mostram melhoras da metodologia integrada em relação à metodologia seqüencial. Assim, por exemplo, nos domingos da empresa G02, há uma redução de 2h29min no tempo de espera em terminal (de 16h13min para 13h44min), uma redução de 2 tripulantes (de 21 para 19), 4h6min no tempo de ociosidade das tripulações (de 15h13min para 11h07min) e uma redução de 4 trocas de veículos (de 6 para 2). É importante observar que na função de avaliação dos veículos não é penalizado o número de veículos.

Tabela 6.4 – Características das melhores soluções das metodologias seqüencial e integrada para a empresa G02

Empresa G02	Algoritmo Sequencial				Algoritmo Integrado			
	DOM	SEG	SEX	SAB	DOM	SEG	SEX	SAB
NVIAG	90	260	260	172	90	260	260	172
NVEIC	11	40	40	23	11	40	39	23
TE	16:13	22:26	5:56	21:02	13:44	8:12	22:14	21:02
TVM	8:24	12:50	12:14	16:48	9:50	15:06	14:58	18:36
NDP(v)	3	21	20	5	5	24	24	8
NDP(t)	12	50	50	16	7	49	43	20
NTRIP	21	62	63	37	19	62	63	34
TO	15:13	15:04	9:23	15:44	11:07	7:52	6:28	15:12
THE	5:04	6:44	7:34	20:08	14:58	8:28	12:31	19:44
NTL	1	5	7	1	4	4	6	2
NTP	2	18	19	9	3	11	8	10
NTV	6	62	66	11	2	34	34	15
f(v)	841	2510	2553	1491	893	2740	2887	1585
f(t)	2478	7999	8197	4579	2333	7339	7535	4254
f(v,t)	7036	22507.5	23045.5	12938.5	6725.5	21087.5	21724.5	12220

Tabela 6.5 – Características das melhores soluções das metodologias sequencial e integrada para a empresa G27

Empresa G27	Algoritmo Sequencial				Algoritmo Integrado			
	DOM	SEG	SEX	SAB	DOM	SEG	SEX	SAB
NVIAG	52	98	98	69	52	98	98	69
NVEIC	5	14	13	7	5	14	13	7
TE	12:57	10:25	8:45	11:12	10:47	10:22	11:08	10:54
TVM	3:50	16:06	15:20	8:26	4:36	16:52	15:20	9:12
NDP(v)	0	7	7	4	1	8	7	5
NDP(t)	6	17	16	8	7	13	13	6
NTRIP	10	24	25	15	10	23	23	14
TO	6:30	10:52	13:44	10:14	5:06	10:06	8:57	8:54
THE	6:53	5:58	6:29	7:33	6:35	11:38	14:42	13:19
NTL	0	0	0	0	0	0	0	0
NTP	0	3	2	1	0	3	3	0
NTV	4	14	12	4	2	6	5	3
f(v)	370	1019	952	661	385	1050	1005	682
f(t)	1251	2912	2991	1813	1205	2735	2753	1750
f(v,t)	3497.5	8299	8429.5	5193.5	3397.5	7887.5	7887.5	5057

Pelas tabelas 6.4 e 6.5 verifica-se que, em todos os dias e nas duas empresas, o método proposto foi capaz de gerar soluções melhores, sob o ponto de vista da função de avaliação, que a da metodologia sequencial. A aplicação da metodologia integrada desenvolvida neste trabalho é vantajosa para situações em que, para a empresa, os custos envolvidos são maiores para os tripulantes, e não para os veículos. Pelos resultados obtidos observa-se que, quanto à integração, há um aumento sutil no custo dos veículos e uma redução significativa no custo dos tripulantes, produzindo assim soluções melhores para o conjunto.

6.2 Resultados – 2ª Abordagem

Nesta abordagem, assim como na anterior, testes preliminares foram realizados com a finalidade de calibrar os parâmetros do ILS-VND-BTRA. Em especial, na segunda abordagem, o número de iterações sem melhora do ILS (ILS_{max}) foi fixado em 50. O tempo de cada chamada do BTRA (BTtempo) foi limitado a 20 segundos. O número máximo de iterações sem melhora do VRD (VRD $_{max}$) foi fixado em 100. O número de iterações sem melhora do IRD (IRD $_{max}$) foi fixado em 50. E, o tempo total de processamento foi limitado a 20 minutos.

A primeira fase da abordagem sequencial usa o algoritmo ILS descrito em Sousa *et al.* (2007) e a segunda, o algoritmo BTRA descrito em Marinho (2004). Os dois algoritmos são interrompidos após 10 minutos de processamento. Nesta abordagem, ILS_{max} é fixado em 50, uma perturbação de nível i é feita aplicando-se $i + 2$ movimentos da vizinhança N_v^R , a busca local do ILS é feita pela aplicação de dois métodos de descida randômica, o primeiro usando a vizinhança N_v^R , e o segundo usando a vizinhança N_v^S .

Para testar os algoritmos, foram examinados problemas de doze problemas-teste disponíveis em <http://www.decom.ufop.br/prof/marcone/pt/instances/VC.zip>, referentes a dados de nove empresas de transporte público.

A Tabela 6.6 apresenta uma comparação dos resultados da metodologia sequencial e da integrada em relação ao melhor valor, ao valor médio e ao desvio. Além disso, esta tabela mostra a melhora obtida em cada caso.

Tabela 6.6 – Comparação de resultados dos algoritmos sequencial e integrado com função de avaliação baseada em custos

Problema	Tradicional			Integrado			Melhora	
	Melhor	Média	Desvio	Melhor	Média	Desvio	Melhor	Média
VC52	3936,96	3988,46	2,97%	3873,31	3895,43	0,57%	1,64%	2,39%
VC90	7519,20	7561,96	1,14%	7476,57	7518,14	0,56%	0,57%	0,59%
VC98a	9638,79	9675,39	1,16%	9564,95	9594,67	0,31%	0,77%	0,84%
VC98b	9500,16	9514,50	1,47%	9376,67	9398,28	0,23%	1,32%	1,24%
VC260a	28479,80	28714,00	1,43%	28310,50	28329,11	0,07%	0,60%	1,36%
VC260b	28064,80	28215,82	0,54%	27921,40	27950,78	0,11%	0,51%	0,95%
VC498	32673,50	32756,81	0,26%	32500,10	32612,90	0,35%	0,53%	0,44%
VC538	18917,00	18964,83	0,25%	18887,20	18939,01	0,27%	0,16%	0,14%
VC639a	39478,10	39586,00	0,27%	39301,30	39398,46	0,25%	0,45%	0,48%
VC639b	42557,60	42608,22	0,12%	42336,20	42436,66	0,24%	0,52%	0,40%
VC1036	48764,60	48840,86	0,16%	48669,10	48778,18	0,22%	0,20%	0,13%
VC1038	49223,90	49335,35	0,23%	49085,40	49140,27	0,11%	0,28%	0,40%

Pela Tabela 6.6 pode-se observar que os dois algoritmos têm bom desempenho, com os melhores valores e valores médios bem próximos, e com desvios baixos. Observa-se, no entanto, um desempenho melhor do algoritmo integrado para todos os casos analisados. Por exemplo, há uma melhora de até 1,64% em relação ao valor da melhor solução encontrada em relação ao sequencial e de 2,30% em relação ao valor médio. Adicionalmente, o desvio é menor em todos os casos, o que mostra a sua robustez do algoritmo integrado proposto.

A Tabela 6.7 apresenta as características das melhores soluções encontradas em cada problema-teste, com respeito à metodologia sequencial e integrada.

Por esta tabela observa-se que o número de veículos permanece o mesmo nos dois algoritmos. Com relação ao número de tripulantes houve redução de um tripulante nos problemas teste VC52, VC260a, VC498. Nas instâncias VC90 e VC639, o número de tripulantes foi maior, mas houve uma redução significativa no tempo de hora-extra e de ociosidade. Para os outros problemas, o número de tripulantes permaneceu o mesmo pelos dois algoritmos. Uma observação interessante é que a abordagem integrada produziu soluções piores para os veículos nos problemas VC498 e VC538. Por outro lado, a programação dos tripulantes passou a ser melhor que a encontrada na metodologia sequencial. Além disso, a solução final foi sempre melhor na metodologia integrada.

Tabela 6.7 – Características das melhores soluções dos algoritmos sequencial e integrado com função de avaliação baseada em custos

Problema-teste	VC52	VC52	VC90	VC90	VC98a	VC98a	VC98b	VC98b
Algoritmo	Seq.	Int.	Seq.	Int.	Seq.	Int.	Seq.	Int.
Número de veículos	5	5	10	10	13	13	13	13
Tempo ocioso dos veículos (hh:mm)	12:57	12:57	16:13	16:13	11:58	11:58	10:21	10:21
Tempo de viagem morta (hh:mm)	03:50	03:50	8:24	08:24	15:20	15:20	14:34	14:34
Número de pegadas duplas (veículos)	0	0	4	4	7	7	6	6
Número de tripulações	12	11	20	21	27	27	25	25
Tempo ocioso das tripulações (hh:mm)	11:31	08:31	18:44	14:28	17:05	13:29	12:50	09:29
Tempo de hora extra (hh:mm)	02:25	05:52	14:13	10:00	06:38	05:09	15:13	10:59
Número de pegadas dúplas (tripulações)	2	2	1	3	7	7	5	5
f_v	2640,4	2640,4	5168,4	5168,4	6809,8	6809,75	6744,1	6744,1
f_t	1296,6	1232,9	2350,8	2308,2	2829,0	2755,2	2756,1	2632,6
f_v+f_t	3937,0	3873,3	7519,2	7476,6	9638,8	9565,0	9500,2	9376,7
Problema-teste	VC260a	VC260a	VC260b	VC260b	VC498	VC498	VC538	VC538
Algoritmo	Seq.	Int.	Seq.	Int.	Seq.	Int.	Seq.	Int.
Número de veículos	40	40	39	39	47	47	24	24
Tempo ocioso dos veículos (hh:mm)	19:15	19:15	13:22	22:02	37:38	39:27	65:26	65:28
Tempo de viagem morta (hh:mm)	37:54	37:54	37:54	37:54	23:36	24:48	18:59	20:23
Número de pegadas duplas (veículos)	22	22	23	23	25	26	3	3
Número de tripulações	76	75	76	76	94	93	53	53
Tempo ocioso das tripulações (hh:mm)	91:31	83:03	84:10	79:07	55:03	44:51	50:55	47:57
Tempo de hora extra (hh:mm)	07:14	08:46	11:28	07:19	24:36	22:17	33:17	29:48
Número de pegadas dúplas (tripulações)	17	17	17	17	19	19	2	2
f_v	20189,0	20189,0	19785,3	19785,3	22851,1	22942,8	12755,6	12830,1
f_t	8290,8	8121,5	8279,5	8136,19	9822,4	9557,3	6161,36	6057,1
f_v+f_t	28479,8	28310,5	28064,8	27921,4	32673,5	32500,1	18917	18887,2
Problema-teste	VC639a	VC639a	VC639b	VC639b	VC1036	VC1036	VC1038	VC1038
Algoritmo	Seq.	Int.	Seq.	Int.	Seq.	Int.	Seq.	Int.
Número de veículos	55	55	60	60	69	69	70	70
Tempo ocioso dos veículos (hh:mm)	50:36	51:45	50:42	52:16	66:07	69:13	67:41	69:10
Tempo de viagem morta (hh:mm)	52:03	52:37	32:59	56:06	50:28	51:41	50:32	51:46
Número de pegadas duplas (veículos)	25	25	25	27	35	35	32	33
Número de tripulações	109	110	116	116	136	136	137	137
Tempo ocioso das tripulações (hh:mm)	57:47	51:29	59:44	49:23	67:57	60:41	78:26	67:10
Tempo de hora extra (hh:mm)	30:10	17:48	31:06	22:01	43:20	37:16	30:50	26:25
Número de pegadas dúplas (tripulações)	22	22	24	24	28	27	28	28
f_v	28133,0	28180,9	30525,4	30608,8	34552,2	34664,8	35027,3	35115,6
f_t	11345,1	11120,3	12032,2	11727,4	14212,4	14004,4	14196,6	13969,8
f_v+f_t	39478,1	39301,3	42557,6	42336,2	48764,6	48669,1	49223,9	49085,4

7 Conclusões

Este projeto de pesquisa apresentou um algoritmo híbrido, baseado nas metaheurísticas *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e Busca Tabu (BT), aplicado ao Problema de Programação Integrada de Veículos e Tripulações.

Para testar o método proposto foram utilizados dados reais das empresas do Sistema de Transporte Público da cidade de Belo Horizonte (MG), que atuam na bacia do Barreiro.

Os resultados obtidos pelo método proposto são comparados com aqueles obtidos pela metodologia clássica, de resolver os problemas PPV e PPT de forma independente e seqüencial. Foram consideradas duas funções de avaliação: uma baseada em pesos empíricos e outra em custos. Os experimentos computacionais realizados mostraram que em ambos os casos, a metodologia integrada proposta é adequada, pois foi capaz de produzir resultados superiores àqueles alcançados pela clássica metodologia seqüencial.

Os resultados também mostram que a metodologia integrada proposta, em comparação com a metodologia seqüencial, pode produzir soluções piores para os veículos, mas melhores para as tripulações, gerando assim soluções melhores para o conjunto. Esse fato mostra claramente que é uma boa idéia explorar a integração desses problemas em vez de gerar a melhor solução para os veículos e depois gerar a melhor solução para as tripulações como faz a metodologia seqüencial.

Referências Bibliográficas

- Atkinson, J B (1998) A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *The Journal of the Operational Research Society*, v. 49, p. 700-708.
- Baita, F.; Pesenti, R.; Ukovich, W.; Favaretto, D. (2000) A comparison of different solution approaches to the vehicle scheduling problem in a practical case. *Computers and Operations Research*, v.27, p.1249-1269.
- Ball, M. O; Bodin, L. D. & Dial, R. (1983) A matching based heuristic for scheduling mass transit crew and vehicles. *Transportation Science*, v. 17, p. 4-31.
- Bassi, Hugo Vinícius; Marinho, Euler Horta; Souza, M. J. F.; Silva, Gustavo Peixoto. (2004). Heurísticas GRASP e Reconexão por Caminhos aplicadas ao problema de escalonamento de tripulações In: Simpósio Brasileiro de Pesquisa Operacional, 2007, Fortaleza. Anais do XXXIX SBPO. Rio de Janeiro: SOBRAPO, 2007. v.1. p.1081 – 1092.
- Bicalho, M. S. S., Silva, G. P.; Souza, M. J. F. (2005) Otimização da operação dos veículos de empresas do transporte público de Belo Horizonte. *Revista da Pesquisa e Pós-Graduação*. Universidade Federal de Ouro Preto, p. 1-8.
- Carpaneto, G.; Martello, S.; Toth, P. (1988) Algorithms and codes for the assignment problem. *Annals of Operations Research*, v.13, p. 193-223.
- Freling, R.; Huisman, D. & Wagelmans, A. P. M (2001) Applying an Integrated approach to vehicle and crew scheduling in practice. In: *Computer-Aided Scheduling of Public Transport*, S. Voss e J. R. Daduna (eds.), Springer-Verlag, Berlin, p. 73-90.
- Freling, R.; Wagelmans, A. P. M. & Paixão, J. M. P. (1999) An overview of models and techniques for integrating vehicle and crew scheduling. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p.441-459.
- Gavish, B.; Schweitzer, P.; Shlifer, E. (1978) Assigning buses to schedules in a metropolitan area. *Computers and Operations Research*, v.5, p. 129-138.
- Glover, F.; Kochenberger, G. (2003) *Handbook of Metaheuristics*. Kluwer Academic Publishers.
- Haase, K. & Friberg, C. (1999) An Exact Branch and Cut Algorithm for the Vehicle and Crew Scheduling Problem. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 63-80.
- Hoffstadt, J. (1981) Computerized vehicle and driver scheduling for the Hamburger Hochbahn Aktiengesellschaft. In: *Computer Scheduling of Public Transport*, Wren A. (ed.), North-Holland, Amsterdam, p. 35-52.
- Kirkman, F. (1968) Problems of innovation in the transport industry: a bus scheduling program. In: *Proceedings of PTRC Public Transport Analysis Seminar, Planning and Transport Research and Computation Co. Ltd.*, London, v.1, p. 1-15.
- Kwan, R. K.; Rahin, M. A. (1995) Bus scheduling with trip co-ordination and complex constraints. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; I. M. Branco; J. M. P. Paixão (eds.), Springer-Verlag, Berlin, p. 91-101.
- Kwan, R. K.; Rahin, M. A. (1999) Object oriented bus vehicle scheduling - the BOOST system. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 177-191.
- LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. (2003) Iterated Local Search. In: F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, MA, USA..
- Marinho, Euler Horta, Ochi, Luiz Satoru, Drummond, Lúcia M. A., Souza, M. J. F., Silva, Gustavo Peixoto. (2004). Busca Tabu aplicada ao Problema de Programação de Tripulações de Ônibus Urbano In: Simpósio Brasileiro de Pesquisa Operacional, 2004, São João Del Rey. Anais do XXXVI SBPO. Rio de Janeiro: SOBRAPO, 2004. v.1. p.1471 – 1482.
- Martin-Löf, A. (1970) A branch-and-bound algorithm for determining the minimal fleet size of a transportation system. *Transportation Science*, v.4, p.159-163.

- Mladenovic, N. & Hansen, P. (1997). A Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097-1100.
- Psarras, J.; Stefanitsis, E.; Christodoulou N. (1997) Combination of local search and CLP in the vehicle-fleet scheduling problem. *European Journal of Operational Research*, v. 98, p. 512-521.
- REIS, J. von A. (2007) Heurísticas baseadas em Busca em Vizinhaça Variável para o Problema de Programação Integrada de Veículos e Tripulações no Transporte Coletivo Urbano por Ônibus, Dissertação de mestrado, Escola Politécnica da USP, São Paulo.
- Saha, J. L. (1970) An algorithm for bus scheduling problems. *Operational Research Quarterly*, v.21, p. 463-474.
- Silva, G. P. (2001) Uma metodologia baseada na técnica de geração de arcos para o problema de programação de veículos. Tese de doutorado. Escola Politécnica da USP, São Paulo.
- SILVA, Gustavo Peixoto, SOUZA, M. J. F., GOMES, Helton Cristiano. (2005) Otimização integrada da tabela de horários com a programação de veículos no sistema de transporte público In: Congresso de Pesquisa e Ensino em Transportes, 2005, Recife. Panorama Nacional da Pesquisa em Transportes. Recife: ANPET, 2005. v.1. p.680 – 691.
- SILVA, Gustavo Peixoto, REIS, Jorge Von Atzingen dos, SOUZA, M. J. F. (2006a) Metaheurísticas aplicadas ao Sistema de Transporte Público In: XIII Congresso Latino Iberoamericano de Investigación Operativa, 2006, Montevideo. Anais do XIII CLAIO. Montevideo: Universidad de la República, 2006. v.1. p.1 – 6.
- SILVA, Gustavo Peixoto, REIS, Jorge Von Atzingen dos, SOUZA, M. J. F. (2006b) Resolução Integrada do Problema de Programação de Veículos e Tripulações no Sistema de Transporte Público In: Congresso Luso Brasileiro para o Planeamento Urbano, Regional, Integrado e Sustentável, 2006, Braga. Anais do 2º Congresso Luso Brasileiro para o Planeamento Urbano, Regional, Integrado e Sustentável. , 2006. v.1. p.121-132.
- SIMÕES, E. M. L.; SOUZA, M. J. F.; SILVA, G. P. (2006) Aplicação da Metaheurística Iterated Local Search à Programação de Veículos no Sistema de Transporte Público. In: XXXVIII Simpósio Brasileiro de Pesquisa Operacional, 2006, Goiânia. Anais do XXXVIII SBPO. Rio de Janeiro : SOBRAPO, 2006. v. 1. p. 1277-1288.
- SOARES, Gleidson Fonseca, SILVA, Gustavo Peixoto, MARINHO, Euler Horta, SOUZA, M. J. F. (2006) Alocação da mão-de-obra no Sistema de Transporte Público: uma visão multiobjetivo In: Congresso de Pesquisa e Ensino em Transportes, 2006, Brasília. Panorama Nacional da Pesquisa em Transportes. Rio de Janeiro: Associação Nacional de Pesquisa e Ensino em Transportes, 2006. v.1. p.693 - 704
- SOUZA, M. J. F. (2008) Inteligência Computacional para otimização, Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto, disponível em <http://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.pdf>.
- SOUZA, M. J. F.; RODRIGUES, M. M. S.; MAPA, S. M. S; SILVA, G. P. (2003a) Um estudo das heurísticas Simulated Annealing e VNS aplicadas ao problema de programação de tripulações. In: Anais do XXIII Encontro Nacional de Engenharia de Produção, Ouro Preto, 8 p., outubro de 2003.
- SOUZA, M. J. F.; SILVA, G. P.; TOFFOLO, T. A. M. (2005) Resolução do Problema de Rodízio de Tripulações de Ônibus Urbano via Simulated Annealing e Iterated Local Search. In XIX Anais do Congresso de Pesquisa e Ensino em Transportes, Recife, v. 1, p. 657-668.
- Souza, M. J. F; Cardoso, L. X. T. & Silva, G. P. (2003b) Programação de Tripulações de Ônibus Urbano: uma abordagem heurística, XXXV Simpósio Brasileiro de Pesquisa Operacional, Natal: SOBRAPO, 2003, p. 1285-1294.
- Souza, M. J. F; Xavier, L. X. T.; Silva, G. P.; Mapa, S. M. S.; Rodrigues, M. M. S. (2004a) Metaheurísticas Aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público. *Tendências em Matemática Aplicada e Computacional (TEMA)*, São José do Rio Preto, v. 5, n. 2, p.357-368.
- Souza, M. J. F; Cardoso, Silva, G. P. & Mapa, S. M. S. (2004b) Métodos de Pesquisa em Vizinhaça Variável Aplicados à Resolução do Problema de Programação Diária de Tripulações de Ônibus Urbano, XVIII Congresso de Pesquisa e Ensino em Transportes, Panorama Nacional de Pesquisa em Transportes, v. 2, p. 1492-1502.
- SOUZA, M. J. F., SILVA, Gustavo Peixoto, SIMÕES, Emiliana Mara Lopes. (2006) Programação de Veículos de Ônibus Urbano: uma abordagem heurística In: Congresso de Pesquisa e Ensino em

- Transportes, 2006, Brasília. Panorama Nacional da Pesquisa em Transportes. Rio de Janeiro: Associação Nacional de Pesquisa e Ensino em Transportes, 2006. v.1. p.705-716.
- TOFFOLO, T. A.; SOUZA, M. J. F.; PEIXOTO, G. P. (2005) Algoritmos Simulated Annealing e Iterated Local Search aplicados à resolução do Problema de Rodízio de Tripulações de Ônibus Urbano. In VIII Simpósio de Pesquisa Operacional e Logística da Marinha, Rio de Janeiro, Anais do VIII SPOLM, 12 p.
- Tosini, E. & Vercellis, C. (1988) An interactive system for extra-urban vehicle and crew scheduling problems. In: *Computer-Aided Transit Scheduling*, J. R. Daduna e A. Wren (eds.), Springer-Verlag, Berlin, p. 41-53.
- Wren, A.; Chamberlain, M. P. (1988) The development of Micro-BUSMAN: scheduling on micro-computers. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; A. Wren (eds.), Springer-Verlag, Berlin, p.160-174.
- Wren, A. (1972) Bus scheduling: an interactive computer method. *Transportation Planning and Technology*, v.1, p.115-122.

**UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO**

Planejamento operacional de lavra

Anexo do Relatório Técnico-científico Final
apresentado ao CNPq, relativo ao processo
474831/2007-8, desenvolvido no período de de-
zembro de 2007 a dezembro de 2009.

Equipe:

Marcone Jamilson Freitas Souza (DECOM/UFOP) - Coordenador

Alexandre Xavier Martins (DECEA/UFOP)

Francisco César de Araújo (mestre em Engenharia Mineral / UFOP)

Igor Machado Coelho (Bolsista de Iniciação Científica / PIBIC - CNPq)

Sabir Ribas (Bolsista de Iniciação Científica / FAPEMIG)

Data de início do projeto: 25/12/2007

Data de fim do projeto: 25/12/2009

Ouro Preto - Minas Gerais - Brasil

Janeiro de 2010

Resumo

Este trabalho trata do planejamento operacional de lavra em minas a céu aberto. Este problema consiste na mistura de minérios provenientes de várias frentes de lavra, levando-se em consideração metas de produção e qualidade do produto final, restrições operacionais e alocação dos equipamentos de carga e transporte necessários ao processo. Considera-se o sistema de alocação dinâmica de caminhões, no qual, após as descargas nos pontos de basculamento, cada caminhão pode se dirigir a uma frente diferente para novo carregamento, aumentando a produtividade da frota. Dada sua complexidade, o problema foi resolvido por técnicas heurísticas, sendo propostos dois algoritmos heurísticos híbridos. O primeiro deles, denominado GVILS, combina os procedimentos heurísticos GRASP, *Variable Neighborhood Descent* e *Iterated Local Search*, enquanto o segundo, denominado H-GVILS, adiciona ao anterior um módulo de busca local feito por um otimizador de programação matemática. Usando problemas-teste da literatura, os dois algoritmos heurísticos foram comparados entre si e com o otimizador CPLEX aplicado exclusivamente a um modelo de programação matemática. Verificou-se que os dois algoritmos propostos são competitivos com o otimizador CPLEX, pois foram capazes de encontrar soluções de boa qualidade rapidamente e com baixa variabilidade das soluções finais. Entretanto, o algoritmo H-GVILS mostrou-se superior, uma vez que é capaz de gerar soluções finais de melhor qualidade que o algoritmo GVILS e com menor variabilidade das soluções finais.

Palavras-chave: Planejamento operacional de lavra, Metaheurísticas, Programação Matemática, Ritmo de lavra, Mistura de minérios.

Conteúdo

Lista de Figuras

Lista de Tabelas

1	Introdução	p. 9
1.1	O Problema de Planejamento Operacional de Lavra	p. 9
1.2	Objetivos do trabalho	p. 10
1.3	Estrutura do trabalho	p. 11
2	Revisão Bibliográfica	p. 13
2.1	Introdução	p. 13
2.2	Modelos de Programação Matemática	p. 15
2.2.1	Modelo de Pinto e Merschmann (2001)	p. 15
2.2.2	Modelo de White, Arnold e Clevenger (1982)	p. 17
2.2.3	Modelo de White e Olson (1986)	p. 18
2.2.4	Modelo de Costa, Souza e Pinto (2004)	p. 20
2.3	Heurísticas	p. 23
2.3.1	Método Randômico de Descida (MRD)	p. 24
2.4	Metaheurísticas	p. 24
2.4.1	<i>Iterated Local Search</i> (ILS)	p. 25
2.4.2	<i>Variable Neighborhood Descent</i> (VND)	p. 26
2.4.3	<i>Greedy Randomized Adaptive Search Procedure</i> (GRASP)	p. 27

3	O Planejamento Operacional de Lavra Abordado	p. 30
3.1	Alocação Estática de Caminhões	p. 31
3.2	Alocação Dinâmica de Caminhões	p. 32
3.3	Características do Problema de Alocação Abordado	p. 32
4	Modelo de programação matemática	p. 34
5	Metodologia Heurística	p. 38
5.1	Representação de uma solução	p. 38
5.2	Geração de uma solução inicial	p. 39
5.2.1	Geração Aleatória	p. 39
5.2.2	Geração Parcialmente Gulosa	p. 40
5.3	Estruturas de vizinhança	p. 43
5.3.1	Movimento Carga - $N^{CG}(s)$	p. 43
5.3.2	Movimento Operação Frente - $N^{OF}(s)$	p. 43
5.3.3	Movimento Número de Viagens - $N^{NV}(s)$	p. 44
5.3.4	Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$	p. 45
5.3.5	Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$	p. 46
5.3.6	Movimento Operação Caminhão - $N^{OC}(s)$	p. 46
5.3.7	Movimento Troca de Viagens - $N^{VT}(s)$	p. 46
5.3.8	Movimento Troca de Carregadeiras - $N^{CT}(s)$	p. 47
5.4	Função de avaliação	p. 48
5.4.1	Produção de Minério e Estéril	p. 48
5.4.2	Qualidade da Mistura	p. 50
5.4.3	Utilização dos Caminhões	p. 51
5.4.4	Produção dos Equipamentos de Carga	p. 51
5.5	Algoritmos Propostos	p. 52

5.5.1	Algoritmo <i>GVILS</i> aplicado ao POLAD	p. 52
5.5.2	Algoritmo <i>H-GVILS</i> aplicado ao POLAD	p. 53
6	Sistema Desenvolvido	p. 55
6.1	Implementação Computacional	p. 55
7	Resultados	p. 57
7.1	Descrição dos problemas-teste	p. 57
7.2	Pesos e parâmetros utilizados	p. 58
7.3	Ambiente de desenvolvimento	p. 58
7.4	Resultados e análise	p. 59
7.4.1	CPLEX $\times$ <i>GVILS</i>	p. 59
7.4.2	CPLEX $\times$ <i>H-GVILS</i>	p. 61
7.4.3	<i>GVILS</i> $\times$ <i>H-GVILS</i>	p. 61
7.4.4	Características das melhores soluções obtidas	p. 62
8	Conclusões e Trabalhos Futuros	p. 64
	Referências	p. 66

Lista de Figuras

1	Procedimento MRD	p. 24
2	Procedimento ILS	p. 25
3	Procedimento VND	p. 27
4	Procedimento GRASP	p. 28
5	Procedimento Fase de Construção GRASP	p. 29
6	Equipamentos de carga e transporte	p. 30
7	Alocação Estática de Caminhões	p. 31
8	Alocação Dinâmica de Caminhões	p. 32
9	Construção de uma solução para o carregamento de estéril	p. 40
10	Construção de uma solução para o carregamento de minério	p. 41
11	Procedimento GRASP aplicado ao POLAD	p. 42
12	Movimento de realocação de equipamentos de carga (frente disponível) . . .	p. 43
13	Movimento de realocação de equipamentos de carga	p. 44
14	Movimento parar operação de uma frente	p. 44
15	Movimento de decréscimo no número de viagens	p. 44
16	Movimento de acréscimo no número de viagens	p. 45
17	Movimento de realocação de viagens de um caminhão	p. 45
18	Movimento de realocação de viagens de uma frente	p. 46
19	Movimento parar operação de um caminhão em uma frente	p. 46
20	Movimento troca de viagens	p. 47
21	Movimento troca de carregadeiras	p. 47
22	Algoritmo <i>GVILS</i> proposto	p. 52

23	Estrutura de camadas do sistema	p.55
24	Estrutura completa do sistema	p.56

Lista de Tabelas

1	Exemplo de características de uma solução para o POLAD	p. 39
2	Características dos problemas-teste do POLAD	p. 57
3	Pesos adotados	p. 58
4	Comparação de resultados: CPLEX $\times$ <i>GVILS</i>	p. 59
5	Comparação de resultados: CPLEX-2 $\times$ <i>GVILS</i>	p. 60
6	Comparação de resultados: CPLEX $\times$ <i>H-GVILS</i>	p. 61
7	Comparação de resultados: <i>GVILS</i> $\times$ <i>H-GVILS</i>	p. 61
8	Características das melhores soluções encontradas	p. 63

1 Introdução

1.1 O Problema de Planejamento Operacional de Lavra

As mineradoras realizam suas atividades em minas subterrâneas ou a céu aberto. Em minas a céu aberto as atividades de carregamento e transporte ocorrem da seguinte maneira: os caminhões se deslocam até a frente de lavra (que são os pontos da mina onde o minério e o estéril são retirados), são carregados pelos equipamentos de carga e em seguida se dirigem aos pontos de descarga, onde descarregam o minério e o estéril. Os pontos de descarga podem ser pilhas de estéril (material que não é aproveitado pelo processo), pilhas de homogeneização (quando é transportada uma quantidade de minério maior do que a usina pode beneficiar ou quando é necessário “misturar” os minérios antes de iniciar o beneficiamento) e usina de tratamento, onde se inicia o beneficiamento de minério.

Para fornecer minério de qualidade uniforme para o processo é necessário misturar minério de diferentes qualidades proveniente de várias partes da mina ou de diferentes minas com o objetivo de assegurar a uniformidade da alimentação, já que mudanças são usualmente acompanhadas de aumento do custo total da operação (CHANDA; DAGDELEN, 1995).

A atividade de transporte de material é um dos mais importantes aspectos na operação de minas a céu aberto (ALARIE; GAMACHE, 2002). Segundo Maran e Topuz (1988), sistemas de transporte nessas minas envolvem grande volume de capital e recursos. O objetivo do problema de transporte é mover o material retirado da mina para a usina de modo que o custo seja minimizado, uma vez que o custo associado influencia a escolha de onde retirar minério (GERSHON, 1982).

Minas a céu aberto utilizam dois critérios para o transporte de material por caminhões: alocação estática e alocação dinâmica. Na alocação estática, os caminhões seguem uma trajetória fixa entre um ponto de carga e outro de descarga, ou seja, os caminhões ficam fixos a esses dois pontos durante um determinado período de tempo. Já na alocação dinâmica, os caminhões não ficam vinculados a uma mesma rota; assim, a cada descarga, o caminhão pode ser

direcionado a um ponto de carga não necessariamente o mesmo da viagem anterior.

A alocação estática é o método mais utilizado nas minerações de pequeno e médio porte por não apresentar a obrigatoriedade de utilização de um sistema automático de alocação, conhecido como sistema de despacho. Esse método, entretanto, proporciona menor produtividade em função da possibilidade de formação de filas de caminhões e ociosidade dos equipamentos de carga (RODRIGUES, 2006).

A vantagem da alocação dinâmica de caminhões é que com essa estratégia há uma maior produtividade da frota. Esse aumento de produtividade pode refletir um aumento na produção da mina ou a redução do número de equipamentos necessários para manter o mesmo nível de produção. Um algoritmo eficiente para a alocação dinâmica de caminhões é importante porque ele integra um sistema de despacho computadorizado. Um sistema de despacho reúne, ainda, um algoritmo de seqüenciamento de viagens, um sistema de comunicação entre os equipamentos de carga e caminhões e uma central de comandos. Segundo White e Olson (1986), para que o sistema de despacho de caminhões seja completo é importante que o sistema de monitoramento dos equipamentos seja preciso e confiável, de modo que as operações da mina possam ser otimizadas em tempo real.

O custo de instalação de sistemas de despacho depende do tamanho da mina e do tipo de operação. Esse custo inibia a sua utilização por mineradoras de pequeno e médio porte. A partir da década de 90, em consequência da evolução da informática, o custo desses sistemas foi consideravelmente reduzido. Essa redução no custo levou ao aumento no número de mineradoras e empreiteiras que utilizam esse tipo de sistema. Segundo Rodrigues (2006), atualmente cerca de 35 minas fazem uso desses sistemas no Brasil, com diferentes níveis de automação.

No presente trabalho, tem-se como foco o problema de planejamento operacional de lavra, considerando o sistema de alocação dinâmica de caminhões.

1.2 Objetivos do trabalho

Este trabalho tem como objetivo geral desenvolver um algoritmo eficiente de otimização para resolver o problema de planejamento operacional de lavra em minas a céu aberto (POLAD), considerando alocação dinâmica de caminhões.

Os objetivos específicos a serem atingidos são os seguintes:

- (a) Fazer uma revisão de literatura sobre as metodologias utilizadas para resolver o problema de planejamento operacional de lavra em minas a céu aberto;

- (b) Fazer uma revisão de literatura sobre as técnicas metaheurísticas *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e GRASP, nas suas versões mais recentes;
- (c) Desenvolver uma metodologia de otimização, baseada em metaheurísticas e hibridizadas com métodos de programação matemática, para resolver o problema;
- (d) Testar a metodologia desenvolvida, usando sempre que possível, dados reais encontrados em mineradoras brasileiras;
- (e) Produzir um artigo que possa ser apresentado e publicado nos anais de um evento científico nacional;
- (f) Produzir um artigo a ser submetido a um periódico de circulação internacional;
- (g) Contribuir com a divulgação de técnicas de otimização aplicadas à resolução do problema, possibilitando à indústria extrativa nacional melhorar sua produtividade e tornar-se mais competitiva;
- (h) Contribuir com a formação de recursos humanos especializados nessa área do conhecimento;
- (i) Contribuir para a consolidação das linhas de pesquisa “Otimização e simulação de operações de lavra em minas a céu aberto e subterrâneas” e “Otimização Combinatória” do grupo de Logística e Pesquisa Operacional da UFOP;

1.3 Estrutura do trabalho

O presente trabalho está dividido em sete capítulos, incluindo esta introdução, onde o problema de planejamento operacional de lavra é contextualizado.

No Capítulo 2 é apresentada uma revisão bibliográfica sobre os diversos métodos utilizados na resolução do problema de planejamento operacional de lavra bem como a forma com que diversos autores tratam esse problema. Descreve-se, também, o Método Randômico de Descida e as metaheurísticas *Variable Neighborhood Descent* (VND), *Iterated Local Search* (ILS) e GRASP.

No Capítulo 3 é apresentado o problema abordado em detalhes.

No Capítulo 4 é desenvolvido um modelo de programação matemática para o problema.

No Capítulo 5 são apresentados os algoritmos heurísticos propostos para resolver o problema, sendo o primeiro puramente heurístico e o segundo, híbrido, combinando metaheurística com programação matemática.

No Capítulo 6 é apresentado o sistema desenvolvido, enquanto no Capítulo 7 são apresentados e analisados os resultados obtidos.

No Capítulo 8 são apresentadas as conclusões e apontados os trabalhos futuros.

2 *Revisão Bibliográfica*

2.1 Introdução

White e Olson (1986) propuseram um algoritmo que é a base para o sistema *DISPATCH*, que vem operando em muitas minas em todo o mundo. Uma solução é obtida em duas etapas. Na primeira, baseada em programação linear, realiza-se uma otimização do problema da mistura de minérios tendo como objetivo a minimização de uma função de custo que considera o ritmo de lavra, a qualidade da mistura, o atendimento às taxas de alimentação da usina de beneficiamento e o remanuseio de material. As restrições do modelo estão relacionadas às capacidades de produção dos equipamentos de carga, à qualidade da mistura e às taxas de alimentação mínima requerida da usina de beneficiamento. A segunda etapa do algoritmo, a qual é resolvida por programação dinâmica, usa um modelo semelhante ao de White, Arnold e Clevenger (1982), diferenciando-se deste por utilizar como variável de decisão o volume de material transportado por hora em uma determinada rota, ao invés da taxa de caminhões por hora. É considerada, ainda, a presença de pilhas de estocagem. Nesta segunda etapa do algoritmo, o objetivo é minimizar a necessidade de transporte de material na mina.

Chanda e Dagdelen (1995) desenvolveram um modelo de programação linear por metas para resolver um problema de mistura de minérios no planejamento de curto prazo em uma mina de carvão. A função objetivo do modelo consistia na soma ponderada de três objetivos distintos: maximizar um critério econômico, minimizar os desvios de produção requeridos e minimizar os desvios de qualidade relativos aos valores desejados para os parâmetros de controle. Nenhuma alocação de equipamento de carga e transporte foi considerada nesse modelo.

Alvarenga (1997) desenvolveu um programa para o despacho ótimo de caminhões em uma mineração de ferro, a céu aberto, com o objetivo de minimizar o tempo de fila da frota de caminhões, aumentar a produtividade desta e melhorar a qualidade do minério lavrado. No trabalho desenvolvido, que é base do sistema SMART MINE, atualmente muito utilizado em várias minas brasileiras, foi aplicada uma técnica estocástica de otimização, o algoritmo genético com processamento paralelo.

Merschmann (2002) desenvolveu um sistema de otimização e simulação para análise de cenário de produção em minas a céu aberto. O sistema, denominado OTISIMIN (Otimizador e Simulador para Mineração), foi desenvolvido em dois módulos. O primeiro corresponde ao módulo de otimização onde um modelo de programação linear foi construído e resolvido e o segundo a um módulo de simulação que permite ao usuário utilizar os resultados obtidos na resolução do modelo de programação linear como dados de entrada para a simulação. O módulo de otimização foi elaborado com o objetivo de otimizar o processo de mistura de minérios oriundos das várias frentes de lavra de forma a atender as especificações de qualidade impostas pela usina de tratamento e realizar a alocação de equipamentos (caminhões, carregadeiras e/ou escavadeiras) às frentes de lavra, considerando tanto alocação dinâmica quanto estática dos caminhões. O modelo de otimização desenvolvido não considera metas de produção e qualidade, nem a redução do número de caminhões necessários ao sistema de produção.

Em Costa, Souza e Pinto (2004) e Costa, Souza e Pinto (2005) foram apresentados e modelados problemas relativos à mistura de minérios provenientes de várias frentes de lavra, levando-se em consideração metas de produção e qualidade, alocação dinâmica e estática de caminhões, restrições operacionais e alocação dos equipamentos de carga e transporte necessários ao processo. Os modelos considerados foram baseados em programação linear por metas e representaram um avanço em relação àqueles de Merschmann (2002). A contribuição desses autores foi propor um modelo mais completo que os até então existentes, contemplando mais situações reais e exigindo um menor número de restrições e variáveis para modelar um certo grupo de restrições.

Dada a NP-completude desses problemas, em Costa (2005) o referido problema foi também modelado de forma heurística por um algoritmo baseado em *Variable Neighborhood Search*. Para explorar o espaço de soluções, o autor utilizou seis tipos de movimentos. Pelos experimentos realizados, o modelo heurístico foi capaz de gerar soluções de melhor qualidade em um menor tempo de processamento que aquelas obtidas pelo modelo exato.

Rodrigues (2006) fez uma análise comparativa de várias metodologias utilizadas para o despacho de caminhões em minas a céu aberto. As metodologias testadas, as quais eram baseadas em Programação Linear, Programação Dinâmica e em Heurísticas, são aquelas consideradas as bases de algoritmos utilizados em sistemas de despacho comercializados no Brasil. As soluções finais geradas por essas metodologias foram simuladas usando-se o ambiente ARENA, com o objetivo de reproduzir o comportamento das operações de lavra. Os resultados obtidos mostraram o desempenho dos algoritmos utilizando as metodologias sob diferentes condições em minas a céu aberto. A autora, porém, afirma que não se pode, através dos resultados obtidos,

concluir pela superioridade de uma delas.

Guimarães, Pantuza e Souza (2007) apresentaram um modelo de simulação computacional para validar resultados obtidos pela aplicação de um modelo de programação matemática na determinação do ritmo de lavra em minas a céu aberto. Os autores modelaram o problema usando o otimizador LINGO e validaram a solução pelo ARENA. Foi concluído que, ao contrário do procedimento comumente adotado nas mineradoras, o cumprimento da meta de produção não pode ser atingido simplesmente aumentando-se o número de veículos no sistema produtivo. Esta conclusão foi obtida tendo em vista que a partir de um determinado número de veículos no sistema, a produção não era alcançada e até pelo contrário, diminuía, devido ao aumento no tempo de fila com o aparecimento de mais veículos de transporte no sistema.

2.2 Modelos de Programação Matemática

Nesta seção são apresentados modelos de programação matemática que apresentam soluções para o problema de planejamento operacional de lavra com alocação dinâmica de caminhões.

2.2.1 Modelo de Pinto e Merschmann (2001)

Pinto e Merschmann (2001), Merschmann (2002) e Pinto, Biajoli e Mine (2003) abordam o problema de planejamento operacional de lavra utilizando o sistema de alocação dinâmica de caminhões. Este modelo contempla o problema da mistura e a alocação de equipamentos de carga, o atendimento da relação estéril/minério mínima e considera a alocação dinâmica dos caminhões.

O modelo de Pinto e Merschmann (2001) é apresentado pelas equações (2.1) - (2.7) e utiliza os seguintes dados de entrada:

- M : Conjunto de frentes de minério;
- E : Conjunto de frentes de estéril;
- F : Conjunto de frentes formado por $M \cup E$;
- S : Conjunto dos parâmetros de qualidade analisados no minério;
- C : Conjunto de equipamentos de carga;
- Pr : Ritmo de lavra recomendado (t/h);
- t_{ij} : Teor do parâmetro j na frente i (%);
- tl_j : Teor mínimo admissível para o parâmetro j no produto final (%);

tu_j : Teor máximo admissível para o parâmetro j no produto final (%);

rem : Relação estéril/minério requerida;

Cl_k : Produção mínima do equipamento de carga k (t/h);

Cu_k : Produção máxima do equipamento de carga k (t/h).

Define-se, ainda, as seguintes variáveis de decisão:

x_i : Ritmo de lavra da frente i (t/h);

$y_{ik} : \begin{cases} 1 & \text{se o equipamento de carga } k \text{ opera na frente } i; \\ 0 & \text{caso contrário.} \end{cases}$

$$\max \sum_{i \in M} x_i \quad (2.1)$$

$$\text{s.a:} \quad tl_j \leq \frac{\sum_{i \in M} t_{ij} x_i}{\sum_{i \in M} x_i} \leq tu_j \quad \forall j \in S \quad (2.2)$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad (2.3)$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad (2.4)$$

$$\sum_{k \in C} Cl_k y_{ik} \leq x_i \leq \sum_{k \in C} Cu_k y_{ik} \quad \forall i \in F \quad (2.5)$$

$$\sum_{i \in M} x_i \geq Pr \quad (2.6)$$

$$\frac{\sum_{i \in E} x_i}{\sum_{i \in M} x_i} \geq rem \quad (2.7)$$

Observa-se nesta formulação que a função objetivo (2.1) deve ser maximizada sujeita às restrições (2.2), que definem valores mínimos e máximos admissíveis para o parâmetro de qualidade j no produto final. Outras restrições que complementam o modelo estão relacionadas à alocação de equipamentos de carga, onde as restrições (2.3) definem que cada frente possui um único equipamento de carga, enquanto as restrições (2.4) definem que cada equipamento de carga opera em uma única frente. As restrições (2.5) estão relacionadas ao ritmo de lavra, mínimo e máximo, imposto pelos equipamentos de carga. A restrição (2.7) diz respeito ao atendimento da relação estéril/minério.

Observa-se, finalmente, que o modelo proposto pelos autores é não-linear, tendo em vista as restrições (2.2) e (2.7). Sendo assim, não há garantia de que a solução final produzida seja

ótima.

2.2.2 Modelo de White, Arnold e Clevenger (1982)

White, Arnold e Clevenger (1982) apresentam um modelo de programação linear para minimizar o número de caminhões necessários através de restrições relacionadas à continuidade do fluxo de material pelos pontos de carga e basculamento e às capacidades de produção dos pontos de carga. Para o modelo descrito pelas equações (2.8)-(2.11) sejam os seguintes dados de entrada :

- N_P : Conjunto de rotas viáveis;
- N_S : Conjunto de pontos de basculamento;
- N_C : Conjunto de pontos de carga;
- N_F : Conjunto de pontos formado por $N_S \cup N_C$;
- E_j : Conjunto de rotas viáveis que chegam no ponto j ;
- S_j : Conjunto de rotas viáveis que saem do ponto j ;
- C : Número de pontos de carga;
- Td_i : Tempo de deslocamento pela rota i (min);
- Tb_j : Tempo de basculamento no ponto j (min);
- R_j : Taxa de carregamento do ponto j (caminhões/min).

e a seguinte variável de decisão:

- P_{ij} : Taxa de caminhões que utilizam a rota i que possui ligação com um ponto de carga ou basculamento j (caminhões/min).

$$\min \sum_{i \in N_P} P_i Td_i + \sum_{j \in N_S} \sum_{i \in E_j} P_{ij} Tb_j + C \quad (2.8)$$

$$\text{s.a:} \quad \sum_{i \in E_j} P_{ij} - \sum_{i \in S_j} P_{ij} = 0 \quad \forall j \in N_F \quad (2.9)$$

$$\sum_{i \in S_j} P_{ij} - R_j = 0 \quad \forall j \in N_C \quad (2.10)$$

$$P_{ij} \geq 0 \quad \forall j \in N_C, i \in N_P \quad (2.11)$$

Nesta formulação, a função objetivo (2.8) visa minimizar a necessidade de caminhões mantendo a produção máxima dos equipamentos de carga (restrições (2.10)). As restrições (2.9)

garantem que a taxa total de entrada de caminhões em um ponto de carga ou basculamento é igual à taxa total de saída deste ponto. As restrições (2.11) não permitem valores negativos para a taxa de caminhões em uma rota.

2.2.3 Modelo de White e Olson (1986)

White e Olson (1986) apresentam um modelo de programação linear para o problema de alocação dinâmica de caminhões em mineração, o qual é dividido em duas partes. Na primeira parte do modelo, restrições (2.12)-(2.15), realiza-se uma otimização do problema da mistura de minérios tendo como objetivo a minimização da função de custo dada pela equação (2.12), a qual considera o ritmo de lavra, o atendimento às taxas de alimentação da usina de beneficiamento e de qualidade da mistura, além do remanuseio de material.

As restrições do modelo estão relacionadas às capacidades de produção dos equipamentos de carga (restrições (2.13)), às taxas de alimentação mínima requerida da usina de beneficiamento (restrições (2.14)) e à qualidade da mistura (restrições (2.15)). A formulação do problema é apresentada pelas equações (2.12)-(2.15) e considera os seguintes dados de entrada:

- N_m : Conjunto de equipamentos de carga alocados nas frentes de lavra;
- N_{est} : Conjunto de equipamentos de carga alocados nas pilhas de estoque;
- N_F : Conjunto de pontos de carga formado por $N_S \cup N_C$;
- S : Conjunto dos parâmetros de qualidade analisados no minério;
- c_m : Custo de movimentação de material (h/m^3);
- c_p : Custo associado à alimentação da usina de beneficiamento (h/m^3);
- c_s : Custo de estocagem de material (h/m^3);
- c_q : Custo associado à qualidade do minério (h/m^3);
- Pu : Produção máxima admissível (m^3/h);
- t_{ij} : Teor do parâmetro j no minério proveniente da frente ou pilha de estoque i (%);
- tl_j : Teor mínimo admissível para o parâmetro j no produto final (%);
- tu_j : Teor máximo admissível para o parâmetro j no produto final (%);
- tc_j : Teor corrente para o parâmetro j na pilha de mistura (%);
- l_j : Importância do parâmetro j ;
- R_i : Produção máxima do ponto de carga i (m^3/min);
- T_C : Intervalo de controle (h);
- M_C : Massa de controle (t);
- SG : Peso específico (t/m^3).

e a seguinte variável de decisão:

x_i : Ritmo de lavra do ponto de carga i (m³/h).

$$\min \sum_{i \in N_m} c_m x_i + c_p (Pu - \sum_{i \in N_F} x_i) + \sum_{i \in N_{est}} c_s x_i + \sum_{i \in N_F} \sum_{j \in S} l_j c_q t_{ij} x_i \quad (2.12)$$

$$\text{s.a:} \quad 0 \leq x_i \leq R_i \quad \forall i \in N_F \quad (2.13)$$

$$Pu \geq \sum_{i \in N_F} x_i \quad (2.14)$$

$$tl_j \leq tc_j + \sum_{i \in N_F} (t_{ij} - tc_j) x_i T_C / (M_C / SG) \leq tu_j \quad \forall j \in S \quad (2.15)$$

A segunda parte do modelo de White e Olson (1986) é semelhante ao modelo de White, Arnold e Clevenger (1982), diferenciando-se por utilizar como variável de decisão o volume de material transportado por hora, ao invés da taxa de caminhões por hora, que utilizam uma rota. É considerada, ainda, a presença de pilhas de estocagem. Para este modelo, apresentado pelas equações (2.16)-(2.21), sejam os seguintes dados de entrada:

N_P : Conjunto de rotas viáveis;

N_S : Conjunto de pontos de basculamento;

N_m : Conjunto de equipamentos de carga alocados nas frentes de lavra;

N_{est} : Conjunto de equipamentos de carga alocados nos pilhas de estoque;

N_F : Conjunto de pontos de carga formado por $N_S \cup N_C$;

N_D : Conjunto de pontos de carga e basculamento formado por $N_F \cup N_S$;

E_j : Conjunto de rotas viáveis que chegam no ponto j ;

S_j : Conjunto de rotas viáveis que saem do ponto j ;

E : Número de pontos de carga;

C_F : Capacidade da frota (m³);

Td_i : Tempo de deslocamento pela rota i (h);

Tb_j : Tempo de basculamento no ponto j (h);

R_i : Taxa de carregamento do ponto i (m³/h).

e a seguinte variável de decisão:

P_{ij} : Volume transportado pela rota i que possui ligação com um ponto de carga ou basculamento j (m³/h).

$$\min \sum_{i \in N_P} P_i T d_i + \sum_{j \in N_S} \sum_{i \in E_j} P_{ij} T b_j + EC_F \quad (2.16)$$

$$\text{s.a:} \quad \sum_{i \in E_j} P_{ij} - \sum_{i \in S_j} P_{ij} = 0 \quad \forall j \in N_D \quad (2.17)$$

$$\sum_{i \in S_j} P_{ij} = R_j \quad \forall j \in N_m \quad (2.18)$$

$$\sum_{i \in S_j} P_{ij} \leq R_j \quad \forall j \in N_{est} \quad (2.19)$$

$$\sum_{i \in S_j} P_{ij} = x_j \quad \forall j \in N_F \quad (2.20)$$

$$P_{ij} \geq 0 \quad \forall j \in N_D, i \in N_P \quad (2.21)$$

Nesta formulação, a função (2.16) tem por objetivo minimizar a necessidade de transporte de material na mina. As restrições (2.17) estão relacionadas com a continuidade do fluxo de material através dos pontos de carga e basculamento. O ritmo de lavra das frentes de minério deve ser igual à sua taxa de carregamento (restrições (2.18)). As restrições (2.19) definem que o ritmo de lavra em pilhas de estocagem deve ser menor ou igual à taxa de carregamento do ponto. A união desta segunda parte do modelo de White e Olson (1986) com a primeira é realizada por meio das restrições (2.20), onde define-se que o fluxo de material que sai de um ponto de carga deve ser igual ao ritmo de lavra determinado na primeira parte do modelo.

2.2.4 Modelo de Costa, Souza e Pinto (2004)

O modelo de programação matemática proposto por Costa, Souza e Pinto (2004) é uma extensão daquele proposto por Pinto e Merschmann (2001) e Pinto, Biajoli e Mine (2003) e inclui restrições de metas de produção e qualidade.

Para esse modelo de alocação dinâmica de caminhões, apresentado pelas equações (2.22)-(2.43), sejam os seguintes dados de entrada:

- M : Conjunto de frentes de minério;
- E : Conjunto de frentes de estéril;
- F : Conjunto de frentes formado por $M \cup E$;
- S : Conjunto dos parâmetros de qualidade analisados no minério;
- C : Conjunto de equipamentos de carga;
- V : Conjunto de equipamentos de transporte;

- Pr : Ritmo de lavra recomendado (t/h);
 Pl : Ritmo de lavra mínimo (t/h);
 Pu : Ritmo de lavra máximo (t/h);
 β^- : Penalidade por desvio negativo da produção;
 β^+ : Penalidade por desvio positivo da produção;
 t_{ij} : Valor do parâmetro j na frente i (%);
 tr_j : Valor recomendado para o parâmetro j na mistura (%);
 tl_j : Valor mínimo admissível para o parâmetro j na mistura (%);
 tu_j : Valor máximo admissível para o parâmetro j na mistura (%);
 α_j^- : Penalidade por desvio negativo para o parâmetro j na mistura;
 α_j^+ : Penalidade por desvio positivo para o parâmetro j na mistura;
 Ql_i : Ritmo de lavra mínimo para a frente i (t/h);
 Qu_i : Ritmo de lavra máximo para a frente i (t/h);
 rem : Relação estéril/minério requerida;
 Cl_k : Produção mínima do equipamento de carga k (t/h);
 Cu_k : Produção máxima do equipamento de carga k (t/h);
 cap_l : Capacidade do caminhão l (t);
 T_{il} : Tempo total de ciclo do caminhão l na frente i (min).
 $g_{lk} : \begin{cases} 1 & \text{se o caminhão } l \text{ é compatível com o equipamento de carga } k; \\ 0 & \text{caso contrário.} \end{cases}$

e as seguintes variáveis de decisão:

- x_i : Ritmo de lavra da frente i (t/h);
 $y_{ik} : \begin{cases} 1 & \text{se o equipamento de carga } k \text{ opera na frente } i; \\ 0 & \text{caso contrário.} \end{cases}$
 n_{il} : Número de viagens que um caminhão l realiza na frente i em uma hora;
 d_j^- : Desvio negativo do parâmetro j na mistura (t/h);
 d_j^+ : Desvio positivo do parâmetro j na mistura (t/h);
 P^- : Desvio negativo do ritmo de lavra em relação ao recomendado (t/h);
 P^+ : Desvio positivo do ritmo de lavra em relação ao recomendado (t/h).

O modelo de programação matemática relativo à alocação dinâmica de uma frota heterogênea de caminhões e equipamentos de carga, levando-se em consideração metas de produção e qualidade de minério, é apresentado pelas equações (2.22)-(2.43).

$$\min \sum_{j \in S} \alpha_j^- d_j^- + \sum_{j \in S} \alpha_j^+ d_j^+ + \beta^- P^- + \beta^+ P^+ \quad (2.22)$$

$$\text{s.a:} \quad \sum_{i \in M} (t_{ij} - tu_j) x_i \leq 0 \quad \forall j \in S \quad (2.23)$$

$$\sum_{i \in M} (t_{ij} - tl_j) x_i \geq 0 \quad \forall j \in S \quad (2.24)$$

$$\sum_{i \in M} (t_{ij} - tr_j) x_i + d_j^- - d_j^+ = 0 \quad \forall j \in S \quad (2.25)$$

$$\sum_{i \in M} x_i - Pu \leq 0 \quad (2.26)$$

$$\sum_{i \in M} x_i - Pl \geq 0 \quad (2.27)$$

$$\sum_{i \in M} x_i - Pr + P^- - P^+ = 0 \quad (2.28)$$

$$x_i - Qu_i \leq 0 \quad \forall i \in F \quad (2.29)$$

$$x_i - Ql_i \geq 0 \quad \forall i \in F \quad (2.30)$$

$$x_i \geq 0 \quad \forall i \in F \quad (2.31)$$

$$d_j^+, d_j^- \geq 0 \quad \forall j \in S \quad (2.32)$$

$$P^+, P^- \geq 0 \quad (2.33)$$

$$\sum_{i \in E} x_i - \text{rem} \sum_{i \in M} x_i \geq 0 \quad (2.34)$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad (2.35)$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad (2.36)$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in F, k \in C \quad (2.37)$$

$$x_i - \sum_{k \in C} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad (2.38)$$

$$x_i - \sum_{k \in C} Cl_k y_{ik} \geq 0 \quad \forall i \in F \quad (2.39)$$

$$n_{il} T_{il} - 60 \sum_{k \in C, g_{lk} \neq 0} y_{ik} \leq 0 \quad \forall i \in F, l \in V \quad (2.40)$$

$$\sum_{i \in F} n_{il} T_{il} - 60 \leq 0 \quad \forall l \in V \quad (2.41)$$

$$x_i - \sum_{l \in V} n_{il} cap_l = 0 \quad \forall i \in F \quad (2.42)$$

$$n_{il} \in \mathbb{Z}^+ \quad \forall i \in F, l \in V \quad (2.43)$$

Observa-se que (2.23)-(2.33) são restrições que juntamente com a função objetivo (2.22) formam o modelo de mistura de minérios com metas. A restrição (2.34) diz respeito ao aten-

dimento da relação estéril/minério mínima requerida. As demais restrições que complementam o modelo podem ser divididas em dois grupos. O primeiro diz respeito à alocação de equipamentos de carga e a faixa de produtividade que torne viável a utilização desses equipamentos. As restrições (2.35) definem que cada frente possui um único equipamento de carga, enquanto que as restrições (2.36) definem que cada equipamento de carga opera em uma única frente. As restrições (2.37) definem que as variáveis de decisão y_{ik} assumem valores binários. As restrições (2.38) e (2.39) limitam, respectivamente, o ritmo de lavra máximo e mínimo.

O segundo grupo de restrições está relacionado ao transporte de material na mina e a alocação dos caminhões. As restrições (2.40) fazem com que um caminhão somente realize viagens à uma frente onde esteja alocado um equipamento de carga compatível. As restrições (2.41) asseguram que um caminhão opere no máximo sessenta minutos. As restrições (2.42) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados à frente. As restrições (2.43) asseguram que o número de viagens que um caminhão faz à uma frente é um valor inteiro positivo.

2.3 Heurísticas

As heurísticas são procedimentos empíricos que visam a obtenção de soluções de qualidade satisfatória em um tempo computacional aceitável. Tais procedimentos, no entanto, não garantem a obtenção da solução ótima para o problema nem são capazes de garantir o quão próximo a solução obtida está da ótima.

As heurísticas podem ser construtivas ou de refinamento. As construtivas têm por objetivo construir uma solução, elemento a elemento. A escolha de cada elemento está, geralmente, relacionada a uma determinada função que o avalia de acordo com sua contribuição para a solução. Tal função é bastante relativa, pois varia conforme o tipo de problema abordado.

As heurísticas de refinamento, ou também chamados de mecanismos de busca local, são técnicas baseadas na noção de vizinhança. Para definir o que é uma vizinhança, seja S o espaço de pesquisa de um problema de otimização e f a função objetivo a minimizar. O conjunto $N(s) \subseteq S$, o qual depende da estrutura do problema tratado, reúne um número determinado de soluções s' , denominado vizinhança de s . Cada solução $s' \in N(s)$ é chamada de *vizinho* de s e é obtido de s a partir de uma operação chamada de *movimento*.

Em linhas gerais, esses métodos partem de uma solução inicial s_0 , percorrem o espaço de busca, através de um movimento, passando de uma solução para outra que seja sua vizinha.

A Subseção 2.2.1 descreve a heurística de refinamento utilizada neste trabalho, a saber, o Método Randômico de Descida.

2.3.1 Método Randômico de Descida (MRD)

O Método Randômico de Descida, ou MRD, é uma heurística de refinamento que consiste em analisar um vizinho qualquer e o aceitar somente se ele for estritamente melhor que a solução corrente. Caso esse vizinho não for melhor, a solução corrente permanece inalterada e outro vizinho é gerado. O método é finalizado quando se atinge um número máximo de iterações ($IterMRD$) sem que haja a produção de melhorias na solução corrente.

O pseudocódigo do MRD é apresentado na Figura 1.

Procedimento MRD
Entrada: Solução s_0 , Função $f(.)$ Saída: Solução s de qualidade superior ou igual à s_0 de acordo com a função f $s \leftarrow s_0$; $iter \leftarrow 0$; enquanto $iter < IterMRD$ faça $iter \leftarrow iter + 1$; Gere aleatoriamente um vizinho $s' \in N(s)$; se s' for melhor que s de acordo com a função f então $s \leftarrow s'$; $iter \leftarrow 0$; fim fim Retorne s ;

Figura 1: Procedimento MRD

2.4 Metaheurísticas

Metaheurísticas são procedimentos de busca local destinados a resolver de forma aproximada um problema de otimização e que exploram o espaço de soluções com capacidade de escapar das armadilhas dos ótimos locais. Elas diferenciam-se entre si basicamente pelas seguintes características:

- Critério de escolha de uma solução inicial;
- Definição da vizinhança $N(s)$ de uma solução s ;

- c) Critério de uma solução vizinha dentro de $N(s)$;
- d) Critério de término;

Da mesma forma que as heurísticas convencionais, as metaheurísticas são baseadas na noção de vizinhança e exploram o espaço de busca transitando de uma solução vizinha a outra.

Nas Subseções 2.4.1, 2.4.2 e 2.4.3 são apresentadas, respectivamente, as metaheurísticas *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e GRASP referenciadas ao longo deste trabalho.

2.4.1 *Iterated Local Search* (ILS)

O método *Iterated Local Search* (ILS) é baseado na idéia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local.

Para aplicar um algoritmo ILS, quatro componentes têm que ser especificadas: (a) Procedimento *GeraSolucaoInicial()*, que gera uma solução inicial s_0 para o problema; (b) Procedimento *BuscaLocal*, que retorna uma solução possivelmente melhorada s'' ; (c) Procedimento *Perturbacao*, que modifica a solução corrente s guiando a uma solução intermediária s' e (d) Procedimento *CriterioAceitacao*, que decide de qual solução a próxima perturbação será aplicada.

A Figura 2 mostra o pseudocódigo do algoritmo ILS básico.

Procedimento $_{ILS}$
Entrada: Solução s , Função $f(.)$ Saída: Solução s^* de qualidade superior ou igual à s de acordo com a função f $s^* \leftarrow \text{BuscaLocal}(s, f);$ enquanto critério de parada não satisfeito faça $s' \leftarrow \text{Perturbacao}(\text{histórico}, s^*);$ $s'' \leftarrow \text{BuscaLocal}(s', f);$ $s^* \leftarrow \text{CritérioAceitacao}(\text{histórico}, s', s'', s^*);$ fim Retorne s^* ;

Figura 2: Procedimento ILS

O sucesso do ILS é centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha do método de busca local, das perturbações e do critério de aceitação. Em princípio, qualquer método de busca local pode ser usado, mas o desempenho do ILS com

respeito à qualidade da solução final e a velocidade de convergência depende fortemente do método escolhido. Normalmente um método de descida é usado, mas também é possível aplicar algoritmos mais sofisticados, tais como Busca Tabu ou outras metaheurísticas.

A intensidade da perturbação deve ser forte o suficiente para permitir escapar do ótimo local corrente e permitir explorar diferentes regiões. Ao mesmo tempo, ela precisa ser fraca o suficiente para guardar características do ótimo local corrente.

O critério de aceitação é usado para decidir de qual solução se continuará a exploração, bem como qual será a perturbação a ser aplicada. Um aspecto importante do critério de aceitação e da perturbação é que eles induzem aos procedimentos de intensificação e diversificação. A intensificação consiste em permanecer na região do espaço onde a busca se encontra, procurando explorá-la de forma mais efetiva; enquanto a diversificação consiste em se deslocar para outras regiões do espaço de soluções. A intensificação da busca no entorno da melhor solução encontrada é obtida, por exemplo, pela aplicação de “pequenas” perturbações sobre ela. A diversificação, por sua vez, pode ser realizada aceitando-se quaisquer soluções s'' e aplicando “grandes” perturbações na solução ótima local.

Um critério de aceitação comumente utilizado é mover-se para o ótimo local s'' somente se ele for melhor que o ótimo local corrente s , isto é, somente se $f(s'') < f(s)$ em um problema de minimização, ou se $f(s'') > f(s)$ em um problema de maximização.

2.4.2 *Variable Neighborhood Descent (VND)*

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto por Mladenović e Hansen (1997), é um método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O pseudocódigo desse algoritmo, em que se considera o refinamento de uma solução s utilizando uma função de avaliação f , a ser minimizada, e um conjunto de r diferentes vizinhanças $N = \{N^{(1)}; N^{(2)}; \dots; N^{(r)}\}$, é apresentado pela Figura 3.

Dependendo do problema abordado, a busca pelo melhor vizinho (linha 4 da Figura 3) pode ser cara computacionalmente. Nessa situação é comum fazer a busca pela primeira solução de melhora. Outra alternativa é considerar a exploração apenas em um certo percentual da vizinhança.

Procedimento VND

Entrada: Solução s , Vizinhança $N(\cdot)$, Inteiro r , Função $f(\cdot)$

Saída: Solução s^* de qualidade superior ou igual à s de acordo com a função f

```

1  Seja  $r$  o número de diferentes estruturas de vizinhança;
2   $k \leftarrow 1$  {Tipo de estrutura de vizinhança corrente};
3  enquanto  $k < r$  faça
4      Encontre o melhor vizinho  $s' \in N^k(s)$ ;
5      se  $s'$  for melhor que  $s$  de acordo com a função  $f$  então
6           $s \leftarrow s'$ 
7           $k \leftarrow 1$ 
8      senão
9           $k \leftarrow k + 1$ 
10     fim
11 fim
12  $s^* \leftarrow s$ 
13 Retorne  $s^*$ ;

```

Figura 3: Procedimento VND

Segundo os autores, o método VND baseia-se em três princípios básicos:

- Um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança;
- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
- Para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximas.

Ainda segundo os autores, o último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Esse é o caso em que os ótimos local e global compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

2.4.3 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP (*Greedy Randomized Adaptive Search Procedure* - Procedimento de Busca Adaptativa Gulosa e Randomizada) é um método iterativo proposto por Feo e Resende (1995), constituído basicamente de duas fases: uma fase de construção e uma fase de busca local, cujo

Procedimento GRASP

Entrada: Inteiro $GRASP_{max}$, Função $f(\cdot)$
Saída: Solução s^* melhor quanto à função f em $GRASP_{max}$ iterações

 $f^* \leftarrow \infty$;

para cada *uma das* $GRASP_{max}$ iterações **faça**

 Construa uma solução s_0 por uma heurística parcialmente gulosa;

 Submeta s_0 a um procedimento de busca local, retornando s ;

se $f(s) < f^*$ **então**
 $s^* \leftarrow s$;

 $f^* \leftarrow f(s)$;

fim
fim

 Retorne s^* ;

Figura 4: Procedimento GRASP

objetivo é convergir à solução encontrada na fase de construção para um ótimo local. A Figura 4 apresenta o pseudocódigo básico do método GRASP para um problema de minimização.

A primeira fase do GRASP é a fase de construção, na qual uma solução viável é construída elemento a elemento. Cada elemento ainda não usado na solução é avaliado por uma função gulosa g e compõe uma lista, denominada de Lista de Candidatos (LC). Por meio de um fator $\alpha \in [0, 1]$ é criada uma Lista Restrita de Candidatos (LRC), cujos elementos i são os melhores da LC segundo a função g e satisfazem a condição $g_i \leq g_{min} + \alpha \times (g_{max} - g_{min})$, sendo g_{min} o valor do elemento com a melhor avaliação segundo g e g_{max} , o de pior avaliação. Definida a LRC, seleciona-se, aleatoriamente, um candidato da LRC e, em seguida, atualizam-se as listas LC e LRC. O método pára quando $LC = \emptyset$.

De acordo com Feo e Resende (1995), o parâmetro α , que determina o tamanho da LRC, influencia significativamente a qualidade e diversidade das soluções geradas durante a fase de construção. Valores de α muito baixos (próximos de zero), ou seja, que determinam um tamanho muito limitado para a LRC, geram soluções próximas à solução puramente gulosa e implicam em uma baixa diversidade das soluções finais. Já uma escolha de α próxima da seleção puramente aleatória (valores de α próximos a 1) leva a uma grande diversidade de soluções construídas mas, por outro lado, muitas das soluções construídas são de qualidade inferior, tornando mais lento o processo de busca local.

O pseudocódigo da fase de construção é apresentado na Figura 5:

A segunda fase do GRASP consiste em refinar a solução gerada pela fase de construção, aplicando um método de busca local. A velocidade de convergência para um ótimo local irá depender da qualidade da solução construída. Quanto melhor for a qualidade da solução gerada

Procedimento Construção GRASP

Entrada: Lista de elementos candidatos LC, Função $g(\cdot)$

Saída: Solução s construída de forma parcialmente gulosa quanto à função g

$s \leftarrow \emptyset$

enquanto *a solução s não estiver totalmente construída* **faça**

 Classifique os elementos de LC de acordo com a função g

 Crie uma LRC, composta pelos melhores elementos da LC

 Selecione aleatoriamente um elemento de LRC e inclua-o na solução s

 Atualize as listas LC e LRC, eliminando o elemento candidato inserido em s

fim

Retorne s ;

Figura 5: Procedimento Fase de Construção GRASP

pela heurística de construção, maior será a velocidade de convergência desta solução para um ótimo local.

3 O Planejamento Operacional de Lavra Abordado

O Problema de Planejamento de Lavra a Céu Aberto em Mineração envolve a alocação de máquinas e caminhões às frentes de lavra. A Figura 6 ilustra um equipamento de carga abastecendo um caminhão em uma frente.



Figura 6: Equipamentos de carga e transporte

Cada frente de lavra contém uma determinada quantidade de material (minério ou estéril), com características físicas, químicas e econômicas diferenciadas, denominadas parâmetros de controle. Como exemplo típico de parâmetros de controle, tem-se: Fe, SiO₂, H₂O, Mn, P, granulometria. Para satisfazer as especificações exigidas pelos clientes, é necessário selecionar as frentes a serem lavradas e seu ritmo de lavra, os quais devem ser determinados proporcionalmente. Para a operação de minério e estéril, a mina conta com uma frota limitada de equipamentos de carga, os quais devem ser alocados às frentes de lavra e operarem em uma faixa de produtividade que torne viável sua utilização (COSTA, 2005).

Considera-se que o transporte do material retirado da frente de lavra é realizado por uma frota de caminhões com capacidades de carga diferentes. Esses caminhões são alocados às frentes de lavra dinamicamente, tentando-se evitar a formação de filas, ou seja, o caminhão é alocado a um ponto de carga ou basculamento que proporcione o menor tempo de fila possível.

O ritmo de lavra é determinado pelas capacidades de operação dos equipamentos de carga e transporte alocados às diversas frentes.

Em minas a céu aberto, são utilizados dois critérios para a alocação de caminhões: alocação estática e alocação dinâmica.

3.1 Alocação Estática de Caminhões

Na alocação estática (Figura 7), um caminhão é alocado a uma única rota, ou seja, permanece se deslocando entre dois pontos fixos, um de basculamento e outro de carga. Esta estratégia é geralmente adotada devido à simplificação das operações e ao custo da implantação de um sistema de despacho computadorizado de caminhões.



Figura 7: Alocação Estática de Caminhões

Esse sistema tem a desvantagem de ser sensível à formação de filas durante as operações de carga e por não aproveitar a ociosidade dos caminhões, geradas ao fixá-los a uma mesma rota.

3.2 Alocação Dinâmica de Caminhões

No sistema de alocação dinâmica (Figura 8), os caminhões não ficam fixos a uma determinada rota, como no sistema de alocação estática. Eles podem ser direcionados a diferentes frentes de lavra, onde esteja um equipamento de carga compatível. Esta estratégia faz aumentar a produtividade da frota e proporciona, segundo Costa (2005), um aumento na capacidade de produção da mina ou mesmo a redução do número de equipamentos necessários para manter o mesmo nível de produção.



Figura 8: Alocação Dinâmica de Caminhões

3.3 Características do Problema de Alocação Abordado

O problema abordado neste trabalho é o de Planejamento Operacional de Lavra com alocação dinâmica de caminhões (POLAD), sendo estes de capacidades diferentes.

Sendo a alocação dinâmica, ao descarregar o material, seja no britador (ou pilhas de estoque próximas ao britador) ou na pilha de estéril, o caminhão é direcionado a uma frente, não necessariamente a mesma da viagem anterior.

Admite-se que há um conjunto de carregadeiras de diferentes produtividades, sendo este conjunto menor que o de frentes às quais elas serão alocadas.

Considera-se o planejamento para uma hora de produção, sendo este aplicado até uma frente exaurir ou ocorrer uma quebra de equipamento, situação na qual deve ser feito outro planejamento.

Dado o elevado custo de uma carregadeira, é imposto um limite mínimo de produção para cada carregadeira para justificar economicamente sua utilização.

Finalmente, considera-se uma taxa de utilização máxima para os caminhões. Por exemplo, supondo uma taxa de utilização máxima de 85%, um caminhão l de 80 t de capacidade deveria trabalhar 51 ($= 0,85 \times 60$) minutos, no máximo, em uma hora. Isso é adotado para retratar uma situação mais real, uma vez que um caminhão não fica todo o tempo em atividade. Além disso, essa taxa de utilização máxima tem por objetivo, também, modelar a variabilidade nos tempos de ciclo dos caminhões.

4 *Modelo de programação matemática*

O modelo proposto de alocação dinâmica de caminhões é uma adaptação daquele de Costa (2005). Especificamente, a equação referente à relação estéril/minério é substituída por uma equação de meta de produção de estéril. Esta substituição é feita porque pode ser conveniente estabelecer limites inferiores ou superiores para a produção de estéril que não sigam a mesma proporção da meta de minério. Para tanto, são criadas variáveis de desvio relativas ao não atendimento dessa meta. Adicionalmente, tal como em Guimarães, Pantuza e Souza (2007), são incluídas restrições relativas às taxas de utilização dos caminhões, bem como uma componente de avaliação do número de veículos usados.

O planejamento da produção é feito para uma hora, sendo replicado enquanto não houver frente exaurida e as condições operacionais forem as mesmas. O objetivo do modelo é minimizar os desvios das metas de produção e qualidade, bem como reduzir o número de veículos necessários à operação. Para sua apresentação, sejam os seguintes parâmetros:

- M : Conjunto de frentes de minério;
- E : Conjunto de frentes de estéril;
- F : Conjunto de frentes de minério e estéril, isto é, $F = M \cup E$;
- T : Conjunto de parâmetros de controle analisados no minério;
- C : Conjunto de equipamentos de carga;
- V : Conjunto de equipamentos de transporte (caminhões);
- P_r : Ritmo de lavra recomendado relativo a minério (t/h);
- P_l : Ritmo de lavra mínimo relativo a minério (t/h);
- P_u : Ritmo de lavra máximo relativo a minério (t/h);
- E_r : Ritmo de lavra recomendado relativo a estéril (t/h);
- E_l : Ritmo de lavra mínimo relativo a estéril (t/h);
- E_u : Ritmo de lavra máximo relativo a estéril (t/h);
- α^- : Penalidade por desvio negativo da produção de minério;
- α^+ : Penalidade por desvio positivo da produção de minério;

- β^- : Penalidade por desvio negativo da produção de estéril;
 β^+ : Penalidade por desvio positivo da produção de estéril;
 t_{ij} : Percentual do parâmetro de controle j na frente i (%);
 tr_j : Percentual recomendado para o parâmetro de controle j na mistura (%);
 tl_j : Percentual mínimo admissível para o parâmetro de controle j na mistura (%);
 tu_j : Percentual máximo admissível para o parâmetro de controle j na mistura (%);
 λ_j^- : Penalidade por desvio negativo para o parâmetro de controle j na mistura;
 λ_j^+ : Penalidade por desvio positivo para o parâmetro de controle j na mistura;
 ω_l : Penalidade por uso do l -ésimo caminhão;
 Qu_i : Ritmo de lavra máximo para a frente i (t/h);
 Tx_l : Taxa máxima de utilização do caminhão l (%);
 Cl_k : Produção mínima do equipamento de carga k (t/h);
 Cu_k : Produção máxima do equipamento de carga k (t/h);
 cap_l : Capacidade do caminhão l (t);
 tc_{il} : Tempo total de ciclo do caminhão l na frente i (min);
 g_{lk} : 1, se o caminhão l é compatível com o equipamento de carga k ; e 0, caso contrário.

Definamos as seguintes variáveis de decisão:

- x_i : Ritmo de lavra da frente i (t/h);
 y_{ij} : 1, se o equipamento de carga k opera na frente i ; e 0, caso contrário.
 n_{il} : Número de viagens que um caminhão l realiza na frente i ;
 P_m^- : Desvio negativo de produção de minério em relação ao recomendado (t/h);
 P_m^+ : Desvio positivo de produção de minério em relação ao recomendado (t/h);
 P_e^- : Desvio negativo de produção de estéril em relação ao recomendado (t/h);
 P_e^+ : Desvio positivo de produção de estéril em relação ao recomendado (t/h);
 d_j^- : Desvio negativo do parâmetro j na mistura (t/h);
 d_j^+ : Desvio positivo do parâmetro j na mistura (t/h);
 U_l : 1, se o veículo l é usado; e 0, caso contrário.

A seguir, é apresentado pelas equações (1)-(26), o modelo de programação matemática relativo ao problema abordado.

$$\min \sum_{j \in T} \lambda_j^- d_j^- + \sum_{j \in T} \lambda_j^+ d_j^+ + \alpha^- P_m^- + \alpha^+ P_m^+ + \beta^- P_e^- + \beta^+ P_e^+ + \sum_{l \in V} \omega_l U_l \quad (4.1)$$

$$\sum_{i \in M} (t_{ij} - tu_j)x_i \leq 0 \quad \forall j \in T \quad (4.2)$$

$$\sum_{i \in M} (t_{ij} - tl_j)x_i \geq 0 \quad \forall j \in T \quad (4.3)$$

$$\sum_{i \in M} (t_{ij} - tr_j)x_i + d_j^- - d_j^+ = 0 \quad \forall j \in T \quad (4.4)$$

$$\sum_{i \in M} x_i \leq P_u \quad (4.5)$$

$$\sum_{i \in M} x_i \geq P_l \quad (4.6)$$

$$\sum_{i \in M} x_i + P_m^- - P_m^+ = P_r \quad (4.7)$$

$$\sum_{i \in E} x_i \leq E_u \quad (4.8)$$

$$\sum_{i \in E} x_i \geq E_l \quad (4.9)$$

$$\sum_{i \in E} x_i + P_e^- - P_e^+ = E_r \quad (4.10)$$

$$x_i \leq Qu_i \quad \forall i \in F \quad (4.11)$$

$$x_i \geq 0 \quad \forall i \in F \quad (4.12)$$

$$d_j^-, d_j^+ \geq 0 \quad \forall j \in T \quad (4.13)$$

$$P_m^-, P_m^+ \geq 0 \quad (4.14)$$

$$P_e^-, P_e^+ \geq 0 \quad (4.15)$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad (4.16)$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad (4.17)$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in F, \forall k \in C \quad (4.18)$$

$$x_i - \sum_{k \in C} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad (4.19)$$

$$x_i - \sum_{k \in C} Cl_k y_{ik} \geq 0 \quad \forall i \in F \quad (4.20)$$

$$n_{il} Tc_{il} - 60 \sum_{k \in C, g_{lk}=1} y_{ik} \leq 0 \quad \forall i \in F, \forall l \in V \quad (4.21)$$

$$x_i - \sum_{l \in V} n_{il} cap_l = 0 \quad \forall i \in F \quad (4.22)$$

$$\frac{1}{60} \sum_{l \in V} n_{il} tc_{il} \leq Tx_l \quad \forall i \in F \quad (4.23)$$

$$U_l - \frac{1}{60} \sum_{i \in F} n_{il} tc_{il} \geq 0 \quad \forall l \in V \quad (4.24)$$

$$n_{il} \in Z^+ \quad \forall i \in F, \forall l \in V \quad (4.25)$$

$$U_l \in \{0, 1\} \quad \forall l \in V \quad (4.26)$$

A função objetivo (4.1) procura minimizar os desvios com relação à qualidade e produção de minério e estéril, bem como o número de caminhões utilizados. As restrições (4.2)-(4.15) tratam o problema clássico de mistura com metas. Nesse grupo, as restrições (4.7) e (4.10)

dizem respeito, respectivamente, aos atendimentos das metas de produção de minério e estéril, enquanto as restrições (4.11) limitam o ritmo de lavra máximo definido pelo usuário.

As demais restrições que complementam o modelo podem ser divididas em dois grupos. O primeiro diz respeito à alocação de equipamentos de carga e a faixa de produtividade que torna viável a utilização desses equipamentos. O segundo grupo está relacionado ao transporte de material na mina e à alocação e utilização dos caminhões.

Em relação ao primeiro grupo, as restrições (4.16) definem que em cada frente pode ser alocado, no máximo, um único equipamento de carga, enquanto que as restrições (4.17) definem que cada equipamento de carga pode operar, no máximo, em uma única frente. As restrições (4.18) definem que as variáveis y_{ik} são binárias. As restrições (4.19) e (4.20) limitam, respectivamente, o ritmo de lavra máximo e mínimo, definidos pela carregadeira alocada à frente.

No segundo grupo de restrições, cada restrição (4.21) faz com que um caminhão somente realize viagens a uma frente onde esteja alocado um equipamento de carga compatível. As restrições (4.22) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados à frente. As restrições (4.23) definem que cada caminhão opere no máximo $Tx\%$ em uma hora. As restrições (4.24), juntamente com a função objetivo, forçam com que os caminhões usados sejam penalizados. As restrições (4.25) forçam que seja inteiro positivo o número de viagens que um caminhão faz a uma frente. As restrições (4.26) indicam que as variáveis U_l são binárias.

5 Metodologia Heurística

Neste capítulo apresenta-se a metodologia heurística proposta para resolver o POLAD. A Seção 5.1 descreve como uma solução do POLAD é representada. A heurística utilizada para geração da solução inicial é apresentada na Seção 5.2, e na Seção 5.3 são apresentados os movimentos que constituem as estruturas de vizinhança utilizadas para resolução do problema. A Seção 5.4 mostra como uma solução é avaliada. Os algoritmos heurísticos propostos para resolver o POLAD são descritos na Seção 5.5.

5.1 Representação de uma solução

Uma solução do POLAD é representada por uma matriz $R_{|F| \times (1+|V|)}$ de valores inteiros, sendo F o conjunto de frentes e V o conjunto de caminhões.

Para clareza de apresentação, a matriz $R_{|F| \times (1+|V|)}$ é decomposta em duas submatrizes Y e N , com $R = [Y|N]$, sendo $Y = (y_i)_{|F| \times 1}$ e $N = (n_{il})_{|F| \times |V|}$. A submatriz $Y_{|F| \times 1}$ representa a alocação dos equipamentos de carga ao conjunto F de frentes e o respectivo *status* de cada um desses equipamentos com relação ao fato de estarem ativos ou não. Em cada célula y_i da matriz $Y_{|F| \times 1}$ representa-se a carregadeira k alocada à frente i . Um valor D significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente i , a carregadeira k associada a tal frente é considerada *inativa* e não é penalizada por produção abaixo da mínima para este equipamento de carga (restrições (4.20) da formulação de programação matemática, página 36). A submatriz $N = (n_{il})_{|F| \times |V|}$ representa o número de viagens realizadas pelos caminhões l às frentes i . Um valor 0 (zero) significa que não há viagem para aquele caminhão, enquanto um valor X informa que há incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

A Tabela 1 exemplifica uma solução para uma instância do problema. Nesta tabela, as linhas representam as frentes de lavra disponíveis no conjunto F , a coluna *CARGA* representa a alocação dos equipamentos de carga às frentes de lavra e as demais colunas indicam o número

de viagens que serão realizadas pelo conjunto V de caminhões disponíveis.

Tabela 1: Exemplo de características de uma solução para o POLAD

	<i>Carga</i>	<i>Cam₁</i>	<i>Cam₂</i>	...	<i>Cam_V</i>
F_1	$\langle Car_1, 1 \rangle$	8	X	...	X
F_2	$\langle D, 0 \rangle$	0	0	...	0
F_3	$\langle Car_8, 0 \rangle$	0	0	...	0
...	...	...	...	...	...
F_F	$\langle Car_5, 1 \rangle$	0	9	...	3

Neste exemplo observa-se, na coluna CARGA, linha F_1 , a dupla $\langle Car_1, 1 \rangle$, indicando que o equipamento de carga Car_1 está alocado à frente F_1 e em operação. Na coluna CARGA, linha F_3 , a dupla $\langle Car_8, 0 \rangle$ indica que o equipamento de carga Car_8 está alocado à frente F_3 , mas não está em operação. Observa-se, ainda, na coluna CARGA, linha F_2 , o valor $\langle D, 0 \rangle$ informando que não existe equipamento de carga alocado à frente F_2 e que, portanto, esta frente está disponível. As demais colunas representam o número de viagens a serem realizadas por um caminhão a uma frente, considerando a compatibilidade entre o caminhão e o equipamento de carga alocado à frente. As células com os valores X indicam incompatibilidade entre um caminhão e o respectivo equipamento de carga.

A partir de Y , N e dos tempos de ciclo dados na matriz $TC = (tc_{il})_{|F| \times |V|}$ são determinados o ritmo de lavra em cada frente e o somatório dos tempos de ciclo de cada caminhão.

5.2 Geração de uma solução inicial

Para gerar uma solução inicial para o POLAD, foram desenvolvidos três procedimentos: aleatório, guloso e parcialmente guloso.

5.2.1 Geração Aleatória

Na geração aleatória, a formação de uma solução inicial é feita em duas etapas. Na primeira, os equipamentos de carga disponíveis são numerados e a seguir esses valores são aleatoriamente distribuídos na submatriz Y , com cada célula recebendo um único valor, no máximo. Na segunda etapa, atribui-se à cada célula n_{il} da submatriz N , um valor aleatório compreendido entre o número mínimo e o máximo de viagens a serem realizadas pelo caminhão l à frente i , desde que exista compatibilidade entre o equipamento de carga alocado à frente i e o caminhão l .

5.2.2 Geração Parcialmente Gulosa

Uma solução inicial para o problema é obtida por um procedimento construtivo parcialmente guloso, semelhante à da fase de construção GRASP (FEO; RESENDE, 1995). A construção é feita em duas etapas. As alocações das carregadeiras e a distribuição das viagens às frentes são feitas, na primeira etapa, às frentes de estéril, e na segunda, às frentes de minério. Esta estratégia é adotada tendo em vista que nas frentes de estéril o importante é atender à produção e não é necessário observar a qualidade.

Nas figuras 9 e 10 são mostrados os pseudocódigos dos procedimentos de construção mencionados. Na classificação dos elementos candidatos a ser inseridos na solução considera-se que para as frentes de estéril, a melhor frente é a que possui a maior massa, a melhor carregadeira é a que oferece a maior produção e o melhor caminhão é o de maior capacidade. Já para as frentes de minério considera-se que a melhor frente é a que possui o menor desvio dos teores em relação às metas, a melhor carregadeira é a que oferece a maior produção e o melhor caminhão é o de menor capacidade.

Procedimento	ConstróiSoluçãoEstéril
Saída: Solução de estéril s	
enquanto <i>a produção de estéril for menor que a produção recomendada e existirem frentes de estéril não utilizadas</i> faça	
<i>frenteAtual</i> $\leftarrow$ melhor frente de estéril ainda não utilizada ;	
se <i>não há carregadeira na frente frenteAtual</i> então	
Aloque a melhor carregadeira ainda não alocada;	
$s \leftarrow s'$;	
se <i>não há caminhão na frente frenteAtual</i> então Retorne s	
para cada <i>caminhão l</i> faça	
se <i>o caminhão l for compatível com a carregadeira da frente frenteAtual</i>	
então	
enquanto <i>a produção de estéril for menor que a produção recomendada e o caminhão l pode fazer mais uma viagem</i> faça	
Aloque uma viagem ao caminhão l para a frente <i>frenteAtual</i> ;	
fim	
fim	
fim	
fim	
Retorne s ;	

Figura 9: Construção de uma solução para o carregamento de estéril

Nesse algoritmo, a cada iteração gera-se uma solução inicial parcialmente gulosa, s_0 , através de uma adaptação da construção gulosa, descrita na Subseção 2.4.3. Essa adaptação se

refere às escolhas das frentes e carregadeiras, que passam a ser escolhidas através de uma LRC (Lista Restrita de Candidatos). Nessa construção, as escolhas são realizadas de forma gulosa, isto é, a carregadeira de maior produtividade é alocada à frente de maior disponibilidade de minério/estéril. Já na fase de construção GRASP, a carregadeira a ser alocada não é necessariamente a de maior produtividade, mas uma das *GRASP_C* carregadeiras de maior produtividade. O mesmo acontece com as frentes, que são escolhidas dentre as *GRASP_F* frentes de maior disponibilidade de minério/estéril.

A solução inicial é refinada pelo procedimento de busca local VND, apresentado na Subseção 2.4.2, gerando uma solução s' . Se a solução s' for melhor que a melhor solução (s^*) encontrada até então, aceita-se s' como a nova solução s^* . O método pára quando o número máximo de iterações (*GRASPmax*) for atingido.

A escolha da frente *frenteAtual* na segunda etapa da construção (Figura 10) é feita por uma função guia, tal como em Bresina (1996). Primeiramente, todas as frentes candidatas são ordenadas tendo em vista os valores dos desvios das metas de qualidade. À r -ésima frente melhor classificada, é associada uma função guia $bias(r) = 1/(2r)$. A seguir, cada frente candidata é escolhida com probabilidade $p(r) = bias(r)/\sum bias(r)$. Nesta estratégia, dá-se maior probabilidade de escolha à frente que mais contribui para minimizar os desvios das metas.

Procedimento ConstróiSoluçãoMinério

Saída: Solução de minério s

enquanto a produção de minério for menor que a produção recomendada **e** existirem frentes de minério não utilizadas **faça**

 Ordene as frentes pelos desvios de meta (ordem crescente);

frenteAtual $\leftarrow$ Escolha uma das frentes de minério que podem ser utilizadas;

se não há carregadeira na frente *frenteAtual* **então**

 Aloque a melhor carregadeira ainda não alocada;

se todas as carregadeiras já foram alocadas **então** **Retorne** s

para cada caminho l **faça**

se o caminho l for compatível com a carregadeira da frente *frenteAtual*

então

enquanto a produção de minério for menor que a produção

 recomendada **e** o caminho l pode fazer mais uma viagem **faça**

 | Aloque uma viagem ao caminho l para a frente *frenteAtual*;

fim

fim

fim

fim

fim

Retorne s ;

Figura 10: Construção de uma solução para o carregamento de minério

A solução inicial é construída com base nos procedimentos *ConstroiSoluçãoEstéril* (Figura 9), e *ConstroiSoluçãoMinério* (Figura 10), executados nesta ordem. Se a solução s for melhor que a melhor solução (s^*) encontrada até então, aceita-se s como a nova solução s^* . O método pára quando o número máximo de iterações ($GRASP_{max}$) for atingido.

O pseudocódigo dessa metodologia é apresentado na Figura 11.

Procedimento GRASP POLAD

```

 $f^* \leftarrow \infty;$ 
 $iter \leftarrow 0;$ 
enquanto  $iter < GRASP_{max}$  faça
     $s_0 \leftarrow \text{ConstroiSoluçãoParcialmenteGulosa}();$ 
    Submeta  $s$  a um procedimento de busca local;
    se  $f(s') < f(s^*)$  então
         $s^* \leftarrow s';$ 
    fim
fim
 $s_0 \leftarrow s^* // \text{Retorne a solução } s_0;$ 

```

Figura 11: Procedimento GRASP aplicado ao POLAD

5.3 Estruturas de vizinhança

Para explorar o espaço de soluções do POLAD foram desenvolvidos 8 tipos diferentes de movimentos, apresentados a seguir, para definir oito estruturas de vizinhança $N^k(s)$. Os seis primeiros movimentos, e suas devidas estruturas de vizinhança, foram propostos em Costa (2005). As demais representam contribuições do presente trabalho.

5.3.1 Movimento Carga - $N^{CG}(s)$

Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, trocar os equipamentos de carga que operam nas frentes i e k , caso as duas frentes possuam equipamentos de carga alocados. No caso de apenas uma das frentes possuir equipamento de carga e a outra estiver disponível, esse movimento consistirá em realocar o equipamento de carga à frente disponível.

Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas juntamente com as frentes escolhidas.

As Figuras 12 e 13 ilustram a aplicação deste tipo de movimento. Na Figura 12, a frente F_4 que antes operava com o equipamento Car_4 , passa a ficar disponível; e a frente F_2 , que antes estava disponível, passa a operar com o equipamento Car_4 .

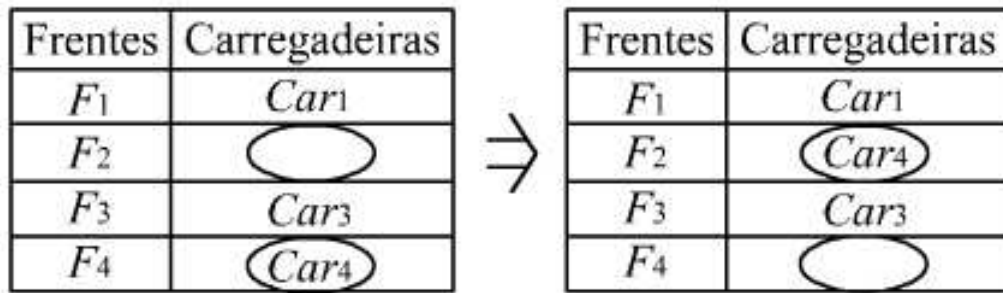


Figura 12: Movimento de realocação de equipamentos de carga (frente disponível)

Na Figura 13, as frentes F_1 e F_3 foram selecionadas e o equipamento de carga Car_1 , anteriormente alocado à frente F_1 , é realocado para a frente F_3 , carregando também as viagens de F_1 para F_3 . O mesmo ocorre em F_3 , onde antes operava a carregadeira Car_2 , que passa a operar com Car_1 .

5.3.2 Movimento Operação Frente - $N^{OF}(s)$

Consiste em retirar de operação o equipamento de carga que esteja em operação na frente i . O movimento retira todas as viagens feitas a esta frente, deixando o equipamento *inativo*. O

Frentes	Carregadeiras	<i>Cam</i> ₁	<i>Cam</i> ₂
<i>F</i> ₁	<u><i>Car</i>₁</u>	3	2
<i>F</i> ₂		0	3
<i>F</i> ₃	<u><i>Car</i>₂</u>	1	0
<i>F</i> ₄	<i>Car</i> ₃	x	0

 $\Rightarrow$

Frentes	Carregadeiras	<i>Cam</i> ₁	<i>Cam</i> ₂
<i>F</i> ₁	<u><i>Car</i>₂</u>	1	0
<i>F</i> ₂		0	3
<i>F</i> ₃	<u><i>Car</i>₁</u>	3	2
<i>F</i> ₄	<i>Car</i> ₃	x	0

Figura 13: Movimento de realocação de equipamentos de carga

equipamento retorna à operação assim que uma nova viagem é associada a ele.

Na Figura 14, a frente *F*₂ que antes estava em operação e tinha viagens de *Cam*₁ e *Cam*₂, passa a não ter mais tais viagens e sua carregadeira é considerada inativa, não sendo penalizada na função objetivo.

Frentes	Carregadeiras	<i>Cam</i> ₁	<i>Cam</i> ₂
<i>F</i> ₁	<u><i>Car</i>₁</u>	3	2
<i>F</i> ₂		0	3
<i>F</i> ₃	<u><i>Car</i>₂</u>	1	0
<i>F</i> ₄	<i>Car</i> ₃	x	0

 $\Rightarrow$

Frentes	Carregadeiras	<i>Cam</i> ₁	<i>Cam</i> ₂
<i>F</i> ₁	<u><i>Car</i>₂</u>	1	0
<i>F</i> ₂		0	3
<i>F</i> ₃	<u><i>Car</i>₁</u>	3	2
<i>F</i> ₄	<i>Car</i> ₃	x	0

Figura 14: Movimento parar operação de uma frente

5.3.3 Movimento Número de Viagens - $N^{NV}(s)$

Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão *l* a uma frente *i*, onde esteja operando um equipamento de carga compatível. Desta maneira, uma célula n_{il} da matriz *N* tem seu valor acrescido ou decrescido de uma unidade.

A Figura 15 ilustra o movimento de diminuição do número de viagens realizadas pelo caminhão *Cam*₂ na frente *F*₂, de 3 para 2 viagens.

Na Figura 16 o movimento aumenta o número de viagens realizadas pelo caminhão *Cam*₂ na frente *F*₂, alterando o número de viagens de 3 para 4.

	<i>Cam</i> ₁	<i>Cam</i> ₂
<i>F</i> ₁	3	2
<i>F</i> ₂	0	<u>3</u>
<i>F</i> ₃	1	0
<i>F</i> ₄	x	0

 $\Rightarrow$

	<i>Cam</i> ₁	<i>Cam</i> ₂
<i>F</i> ₁	3	2
<i>F</i> ₂	0	<u>2</u>
<i>F</i> ₃	1	0
<i>F</i> ₄	x	0

Figura 15: Movimento de decréscimo no número de viagens

	<i>Cam1</i>	<i>Cam2</i>
F_1	3	2
F_2	0	3
F_3	1	0
F_4	x	0

 $\Rightarrow$

	<i>Cam1</i>	<i>Cam2</i>
F_1	3	2
F_2	0	4
F_3	1	0
F_4	x	0

Figura 16: Movimento de acréscimo no número de viagens

5.3.4 Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$

Consiste em seleccionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl} . Assim, neste movimento, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

A Figura 17 ilustra este movimento, onde o caminhão Cam_1 e as frentes F_1 e F_2 são seleccionados, e uma viagem de F_1 é realocada para F_2 .

	<i>Cam1</i>	<i>Cam2</i>
F_1	3	2
F_2	0	3
F_3	1	0
F_4	x	0

 $\Rightarrow$

	<i>Cam1</i>	<i>Cam2</i>
F_1	2	2
F_2	1	3
F_3	1	0
F_4	x	0

Figura 17: Movimento de realocação de viagens de um caminhão

5.3.5 Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$

Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik} . Portanto, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Este movimento é apresentado na Figura 18, onde a frente F_2 foi selecionada e uma viagem do caminhão Cam_2 é transferida para o caminhão Cam_1 .

	Cam_1	Cam_2
F_1	3	2
F_2	0	3
F_3	1	0
F_4	x	0

 $\Rightarrow$

	Cam_1	Cam_2
F_1	3	2
F_2	1	2
F_3	1	0
F_4	x	0

Figura 18: Movimento de realocação de viagens de uma frente

5.3.6 Movimento Operação Caminhão - $N^{OC}(s)$

Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, significando retirar de atividade um caminhão l que esteja operando em uma frente i .

A Figura 19 ilustra este movimento. Na figura, observa-se que o caminhão Cam_2 e a frente F_2 foram selecionados e o número de viagens foi zerado, ou seja, Cam_2 não irá operar em F_2 .

	Cam_1	Cam_2
F_1	3	2
F_2	0	3
F_3	1	0
F_4	x	0

 $\Rightarrow$

	Cam_1	Cam_2
F_1	3	2
F_2	0	0
F_3	1	0
F_4	x	0

Figura 19: Movimento parar operação de um caminhão em uma frente

5.3.7 Movimento Troca de Viagens - $N^{VT}(s)$

Duas células da matriz N são selecionadas e uma viagem é realocada entre elas. Tal movimento pode ocorrer entre quaisquer células da matriz N , respeitando-se as restrições de compatibilidade entre equipamentos.

A Figura 20 ilustra este movimento. Nela, observa-se que a frente F_1 e o caminhão Cam_1 , assim como a frente F_4 e o caminhão Cam_2 foram selecionados, transferindo-se uma viagem de um caminhão para outro.

Frentes	Carregadeiras	Cam_1	Cam_2
F_1	Car_1	3	2
F_2		0	3
F_3	Car_2	1	0
F_4	Car_3	x	4

 $\Rightarrow$

Frentes	Carregadeiras	Cam_1	Cam_2
F_1	Car_1	2	2
F_2		0	3
F_3	Car_2	1	0
F_4	Car_3	x	5

Figura 20: Movimento troca de viagens

5.3.8 Movimento Troca de Carregadeiras - $N^{CT}(s)$

Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, trocar os equipamentos de carga que operam nas frentes i e k . Analogamente ao movimento CG , em um movimento CT os equipamentos de carga das frentes são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

Na Figura 21 observa-se que as carregadeiras Car_1 e Car_3 , alocadas às frentes F_1 e F_4 , respectivamente, são trocadas, mas as viagens feitas a estas frentes são mantidas. Porém, para manter a compatibilidade entre caminhões e carregadeiras, o caminhão Cam_1 , incompatível com a carregadeira Car_3 , tem agora suas viagens à frente F_1 zeradas visto sua incompatibilidade com a carregadeira alocada à frente. O caminhão Cam_1 pode agora fazer viagens à frente F_4 , uma vez que agora se encontra alocada nesta frente a carregadeira Car_1 , compatível com tal caminhão.

Frentes	Carregadeiras	Cam_1	Cam_2
F_1	Car_1	3	2
F_2		0	3
F_3	Car_2	1	0
F_4	Car_3	x	5

 $\Rightarrow$

Frentes	Carregadeiras	Cam_1	Cam_2
F_1	Car_3	x	2
F_2		0	3
F_3	Car_2	1	0
F_4	Car_1	0	5

Figura 21: Movimento troca de carregadeiras

5.4 Função de avaliação

Como os movimentos desenvolvidos podem gerar soluções inviáveis, uma solução é avaliada por uma função f , a ser minimizada, composta por duas parcelas. A primeira delas é a função objetivo propriamente dita, f^{PM} (equação (4.1) do modelo de programação matemática), e a segunda é composta pelas funções que penalizam a ocorrência de inviabilidade na solução corrente. Assim, a função f mensura o desvio dos objetivos considerados e penaliza o não atendimento às restrições do problema. Ela está definida pela equação (5.1).

$$f(s) = f^{PM}(s) + f^p(s) + \sum_{j \in T} f_j^q(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \quad (5.1)$$

em que:

$f^{PM}(s)$ é uma função que avalia s quanto ao atendimento às metas de produção e qualidade, bem como número de caminhões utilizados (mesma do modelo de programação matemática);

$f^p(s)$ avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril;

$f_j^q(s)$ avalia s quanto à inviabilidade em relação ao j -ésimo parâmetro de controle;

$f_l^u(s)$ avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l -ésimo caminhão;

$f_k^c(s)$, que avalia s quanto ao desrespeito aos limites de produtividade da carregadeira k .

Mostra-se, a seguir, como cada uma dessas componentes da função $f(s)$ é avaliada.

5.4.1 Produção de Minério e Estéril

Quanto à produção de minério e estéril, uma solução s é avaliada segundo a Equação (5.2).

$$\begin{aligned} f^p(s) = & \Theta^{m-} \times \max(0, P_l - P_m) + \Theta^{m+} \times \max(0, P_m - P_u) + \\ & \Theta^{e-} \times \max(0, E_l - P_e) + \Theta^{e+} \times \max(0, P_e - E_u) \end{aligned} \quad (5.2)$$

em que:

P_m : Produção de minério (t/h);

P_l : Limite mínimo de produção de minério (t/h);

P_u : Limite máximo de produção de minério (t/h);

Θ^{m+} : Peso associado à inviabilidade quanto ao limite máximo de produção de minério.

Θ^{m-} : Peso associado à inviabilidade quanto ao limite mínimo de produção de minério.

P_e : Produção de estéril (t/h);

E_l : Limite mínimo de produção de estéril (t/h);

E_u : Limite máximo de produção de estéril (t/h);

Θ^{e+} : Peso associado à inviabilidade quanto ao limite máximo de produção de estéril.

Θ^{e-} : Peso associado à inviabilidade quanto ao limite mínimo de produção de estéril.

O valor P_m da produção de minério é obtido pelo somatório de todas as viagens realizadas pelos caminhões às frentes de minério multiplicadas pelas suas respectivas capacidades de carga, conforme Equação (5.3).

$$P_m = \sum_{i \in M} \sum_{l \in V} n_{il} cap_l \quad (5.3)$$

em que:

M : Conjunto de frentes de minério;

V : Conjunto de caminhões;

n_{il} : Número de viagens que um caminhão l faz à uma frente i em uma hora;

cap_l : Capacidade do caminhão l (t).

O valor de P_e da produção de estéril é obtido analogamente pela equação (5.4).

$$P_e = \sum_{i \in E} \sum_{l \in V} n_{il} cap_l \quad (5.4)$$

em que:

E : Conjunto de frentes de estéril;

V : Conjunto de caminhões;

n_{il} : Número de viagens que um caminhão l faz à uma frente i em uma hora;

cap_l : Capacidade do caminhão $l(t)$.

5.4.2 Qualidade da Mistura

A qualidade da mistura é mensurada pelo desvio em relação às metas relativas aos parâmetros de controle. A Equação (5.5) avalia esse desvio para o j -ésimo elemento do conjunto T , de parâmetros de controle.

$$f_j^q(s) = \Delta_j \times \left[\Phi_j^- \times \max(0, Ql_j - Q_j) + \Phi_j^+ \times \max(0, Q_j - Qu_j) \right] \quad \forall j \in T \quad (5.5)$$

em que:

Q_j : Quantidade encontrada na mistura para o parâmetro de controle j (ton);

Ql_j : Quantidade mínima para o parâmetro de controle j na mistura (ton);

Qu_j : Quantidade máxima para o parâmetro de controle j na mistura (ton);

Φ_j^- : Peso associado à quantidade mínima do parâmetro j ;

Φ_j^+ : Peso associado à quantidade máxima do parâmetro j ;

Δ_j : Multiplicador associado ao parâmetro j .

Os valores de Q_j , Ql_j e Qu_j são avaliados conforme as Equações (5.6), (5.7) e (5.8), respectivamente.

$$Q_j = \sum_{i \in M} t_{ij} x_i \quad \forall j \in T \quad (5.6)$$

$$Ql_j = tl_j \times \sum_{i \in M} x_i \quad \forall j \in T \quad (5.7)$$

$$Qu_j = tu_j \times \sum_{i \in M} x_i \quad \forall j \in T \quad (5.8)$$

O multiplicador Δ_j é utilizado para equiparar os parâmetros de controle em uma mesma ordem de grandeza, de modo que os pesos Φ_j^- e Φ_j^+ sejam aplicados coerentemente.

5.4.3 Utilização dos Caminhões

Os caminhões são avaliados de acordo com a carga transportada, dada em toneladas de minério por hora, conforme Equação (5.9).

$$f_l^u(s) = \Omega^+ \times cap_l \times \max(0, TU_l - Tx_l) \quad \forall l \in V \quad (5.9)$$

em que:

TU_l : Taxa de utilização do caminhão l (t/h);

Tx_l : Taxa máxima de utilização do caminhão l (%).

A taxa de utilização de um caminhão l (TU_l) é dada pela Equação (5.10), que retorna o percentual do tempo em que o caminhão é efetivamente utilizado em uma hora de operação.

$$TU_l = \frac{\sum_{i \in F} n_{il} tc_{il}}{60} \quad \forall l \in V \quad (5.10)$$

O fator Ω^+ penaliza uma solução que apresenta um caminhão sendo utilizado acima da sua utilização máxima.

5.4.4 Produção dos Equipamentos de Carga

Cada equipamento de carga deve operar em uma faixa de produção que garanta a sua viabilidade operacional. A produção do equipamento de carga é avaliada conforme o ritmo de lavra da frente à qual está alocado, como mostra a Equação (5.11).

$$f_i^c(s) = \Psi_k^- \times \max\{0, Cl_k - x_i\} + \Psi_k^+ \times \max\{0, x_i - Cu_k\} \quad \forall i \in F \quad (5.11)$$

em que:

x_i : Ritmo de lavra da frente i (t/h);

k : Equipamento de carga que está operando na frente i ;

Cu_k : Produção máxima do equipamento de carga k alocado à frente i (t/h);

Cl_k : Produção mínima do equipamento de carga k alocado à frente i (t/h);

Os fatores Ψ_k^- e Ψ_k^+ penalizam uma solução que apresenta um equipamento de carga com produção abaixo ou acima, respectivamente, da produção mínima e máxima previamente estabelecida.

5.5 Algoritmos Propostos

Para resolver o POLAD, são desenvolvidos dois algoritmos, *GVILS* e *H-GVILS*, os quais são descritos a seguir.

5.5.1 Algoritmo *GVILS* aplicado ao POLAD

O algoritmo *GVILS* combina os procedimentos GRASP (FEO; RESENDE, 1995), *Variable Neighborhood Descent* - VND (MLADENOVIC; HANSEN, 1997) e *Iterated Local Search* - ILS (LOURENÇO; MARTIN; STÜTZLE, 2003). Seu pseudocódigo está esquematizado na Figura 22.

Procedimento <i>GVILS</i>	
Entrada: Solução s , Inteiro $ILStempo$, Inteiro $ILSmax$, Função $f(.)$	
Saída: Solução s^* de qualidade superior ou igual à s de acordo com a função f	
1	$s^* \leftarrow \text{FaseConstruçãoGRASP}(s, f);$
2	$p \leftarrow 0;$
3	enquanto critério de parada não satisfeito faça
4	$iter \leftarrow 0;$
5	enquanto $iter < ILSmax$ and critério de parada não satisfeito faça
6	$s' \leftarrow \text{perturbação}(s^*, p);$
7	$s'' \leftarrow \text{VND}(s', f);$
8	se s'' for melhor que s^* de acordo com a função f então
9	$s^* \leftarrow s'';$
10	$p \leftarrow 0;$
11	$iter \leftarrow 0;$
12	senão
13	$iter \leftarrow iter + 1;$
14	fim
15	fim
16	$p \leftarrow p + 1;$
17	fim
18	Retorne $s^*;$

Figura 22: Algoritmo *GVILS* proposto

Uma solução inicial (linha 1 da Figura 22) é gerada pelo procedimento parcialmente guloso

GRASP, conforme descrito na Seção 5.2. A busca local é feita pelo procedimento VND usando-se os movimentos descritos na seção 5.3. Estrategicamente, a busca local opera nas vizinhanças em uma ordem pré-definida, começando das que exigem menor esforço computacional para aquelas que exigem maior esforço.

Como em testes preliminares algumas vizinhanças não produziam bons ótimos locais ou despendiam muito tempo de processamento para alcançar um ótimo local, apenas um grupo reduzido de vizinhanças foi utilizado nessa busca local. Assim, o VND utiliza a seguinte ordem de exploração de vizinhanças: N^{CG} , N^{NV} , N^{VC} e N^{VF} . Os movimentos relativos às vizinhanças N^{CT} , N^{OC} , N^{OF} e N^{VT} , foram utilizados apenas como perturbação da solução corrente.

O objetivo da perturbação é diversificar a busca, gerando uma solução diferente e cada vez mais *distante* da região atual de exploração no espaço de busca. Para cumprir esta missão, são estabelecidos vários níveis de perturbação. Para um dado nível p de perturbação, são aplicados à solução corrente $p + 2$ movimentos, cada qual escolhido aleatoriamente dentre 6 daqueles descritos na Subseção 5.3. Cada movimento é escolhido de acordo com uma certa probabilidade, dada por: 30% para N^{NV} , 20% para N^{VT} , 20% para N^{CG} , 10% para N^{CT} , 10% para N^{OF} e 10% para N^{OC} . Essa probabilidade diferenciada deveu-se à influência mais significativa dos movimentos baseados em alteração do número de viagens, troca de viagens e troca dos equipamentos de carga na qualidade da solução final. A essa solução perturbada é aplicada busca local, baseada no procedimento VND (linha 7 da Fig. 22). Após $ILSMax$ iterações sem melhora em um dado nível, este é aumentado em uma unidade. No caso de se encontrar uma solução de melhora, o nível de perturbação volta ao seu nível mais baixo, no caso, $p = 0$.

5.5.2 Algoritmo *H-GVILS* aplicado ao POLAD

O algoritmo *H-GVILS* é um aperfeiçoamento do algoritmo *GVILS* descrito na Seção anterior. Neste algoritmo, a partir de um determinado nível de perturbação, a busca local (linha 7 da Figura 22) não passa a ser feita de forma heurística e sim de forma exata. Para tanto, fixa-se uma parte grande do problema e resolve-se na otimalidade uma parte pequena do POLAD. Além disso, há diferença também na maneira como são feitas as perturbações após esse determinado nível de perturbação é alcançado.

No *H-GVILS*, para cada nível p de perturbação, são aplicados à solução corrente $p + 2$ movimentos, escolhidos aleatoriamente dentre os 8 descritos na Subseção 5.3. A diferença é que após um nível H de perturbações, o método começa a fazer perturbações da seguinte maneira: inicialmente seleciona-se aleatoriamente 10% das frentes e fixa-se no modelo exato (formulação de programação matemática, descrita no Capítulo 4) todas as configurações das

outras frentes não selecionadas, ou seja, as células das matrizes Y , de carregadeiras e N , de viagens, que não pertencem a estes 10% selecionados. Desta maneira, executa-se o otimizador de programação matemática e retorna-se como solução perturbada a solução encontrada pelo método exato. A cada nível são aumentados mais 10% de frentes livres, de forma que ao término de dez níveis, o método é executado totalmente de forma exata. O procedimento de solução via o otimizador é interrompido em duas situações: (i) após encontrar a solução ótima ou (ii) após decorridos τ segundos de processamento.

6 *Sistema Desenvolvido*

6.1 Implementação Computacional

Os algoritmos propostos foram implementados na linguagem C++ e tiveram sua estrutura projetada no modelo de um *framework* (Figura 23). Tal estratégia se justifica pelo fato de que durante o desenvolvimento deste trabalho percebeu-se que muitas abstrações poderiam ser feitas no código de forma que este pode então ser aproveitado parcialmente e até integralmente para problemas semelhantes ao POLAD e mesmo para muitos outros problemas de otimização. Desta forma, reduziu-se enormemente o tempo de programação, tempo que pode ser aplicado para um melhor detalhamento e análise do problema real e melhor aproveitamento dos detalhes intrínsecos do problema abordado. Outra razão para esta abordagem de *framework* é que esta se tornou a forma mais natural de se trabalhar com uma integração híbrida entre modelos exatos e metaheurísticas.



Figura 23: Estrutura de camadas do sistema

O *framework* considera que as estruturas de vizinhanças são compostas de um número finito de movimentos, e que tais movimentos são *numeráveis*. O próprio *framework* é dividido em duas camadas, onde a primeira contém apenas classes abstratas, que devem ser implementadas e servem de esqueleto para a segunda camada. Com a primeira definida pode-se usufruir de toda a estrutura montada na segunda camada, onde estão os métodos da descida, VND, ILS, integração com modelos exatos, que podem ser chamados independentemente do problema abordado sem a necessidade de edição de uma linha de código sequer em tais métodos.

Um *método* é definido por uma classe abstrata que abstrai o método *executar* :: Solução $\rightarrow$ Solução, o qual parte de uma solução e retorna uma nova solução, sendo esta melhor ou igual à solução de entrada. Outros parâmetros, exclusivos de cada método, são colocados nos construtores de seus objetos, de forma que algum outro método pode chamá-lo no seu corpo sem precisar conhecê-lo. Para isso, basta esse outro método estar instanciado e ter definida a função *executar*. Este mecanismo facilita a integração de métodos, e como consequência, propicia maior facilidade na execução de testes diversos para uma estrutura de problema já definida.

A Figura 24 mostra a estrutura completa do *framework*, onde N^k denota a k -ésima estrutura de vizinhança, e M^k a k -ésima estrutura de representação do movimento.



Figura 24: Estrutura completa do sistema

7 *Resultados*

7.1 Descrição dos problemas-teste

Os problemas-teste utilizados para teste foram os mesmos de Costa (2005). Tais cenários se referem a dados do planejamento operacional de empresas mineradoras do quadrilátero ferrífero, situado na região central do Estado de Minas Gerais. Os parâmetros de controle são os teores químicos (Fe, SiO₂, Mn, P, H₂O etc) e granulometrias especificadas para o minério. A diferença reside apenas na função de avaliação que, ao contrário deste autor, considera a penalização pela utilização de veículos, bem como a inclusão de uma nova restrição que impede um caminhão de operar mais que uma determinada taxa de utilização, no caso, 85% de uma hora, no máximo, situações essas não contempladas no modelo por ele proposto.

Foram utilizados 4 problemas-teste do POLAD, todos retratando situações reais de uma mineradora do quadrilátero ferrífero de Minas Gerais. A Tabela 2 apresenta algumas características desses problemas-teste utilizados na execução dos algoritmos. Nesta tabela, as colunas $|F|$ e $|T|$ representam, respectivamente, o número de frentes de lavra e o número de parâmetros de controle. A coluna $|C|$ mostra o total de carregadeiras e a coluna $|V|$ o total de veículos (caminhões) disponíveis.

Tabela 2: Características dos problemas-teste do POLAD

Problema-teste	$ F $	$ T $	$ C $	$ V $
PADC01	17	10	8	30
PADC02	17	10	8	30
PADC03	32	10	7	30
PADC04	32	10	7	30

Para os problemas-teste PADC01 e PADC02, os caminhões de 1 a 15 têm capacidade 50 t e são compatíveis com as carregadeiras Car01 a Car04, enquanto os demais caminhões têm capacidade 80 t e são compatíveis com as carregadeiras Car05 a Car08. Já para os problemas-teste PADC03 e PADC04, todos os caminhões são de 50 t e compatíveis com todas as carregadeiras. Os dois últimos problemas diferem apenas na composição química e granulométrica dos

minérios presentes nas frentes de lavra.

7.2 Pesos e parâmetros utilizados

O parâmetro do método, *IterMax*, que indica o número de iterações sem melhora em um dado nível de perturbação foi fixado, após uma bateria preliminar de testes, em 60, para o método GVILS, e 50, para o método H-GVILS. O valor de H (vide Subseção 5.5.2) para o método exato foi de 4 níveis de perturbação, com um tempo τ limitado a 30 segundos.

Os pesos adotados na função de avaliação são apresentados na Tabela 3 e são os mesmos de Costa (2005), além daqueles não previstos nesse trabalho.

Tabela 3: Pesos adotados

Pesos	Descrição	Valor
γ	Penalidade por tonelada abaixo dos limites inferiores ou acima dos limites superiores de produção (estéril/minério)	1000
α	Penalidade por tonelada abaixo ou acima da meta de produção (estéril/minério)	100
Θ^+	Penalidade por tonelada acima do limite superior de produção (minério/estéril)	1000
Θ^-	Penalidade por tonelada abaixo do limite inferior de produção (minério/estéril)	1000
Φ_j	Penalidade por tonelada abaixo (acima) do limite mínimo (máximo) de especificação do parâmetro j	100
λ_j	Penalidade por tonelada abaixo ou acima da meta de qualidade do parâmetro de controle j	1
Δ_j	Multiplicador associado ao parâmetro de controle j	1
ω_l	Penalidade pelo uso do caminhão l	1
Tx_l	Taxa máxima de utilização do caminhão l	85%
Ω^+	Penalidade por utilização acima da taxa máxima de utilização mínimo de produção de um caminhão	1000
Ψ_k^-	Penalidade por tonelada abaixo do limite mínimo de produção da carregadeira k	1000
Ψ_k^+	Penalidade por tonelada acima do limite máximo de produção da carregadeira k	1000

7.3 Ambiente de desenvolvimento

Os algoritmos heurísticos *GVILS* e *H-GVILS* foram desenvolvidos em C++ usando o compilador g++ 4.0 e o IDE Eclipse 3.1. O algoritmo heurístico *H-GVILS*, que integra programação

matemática com metaheurística, usou o otimizador GLPK, versão 4.9, para resolver partes menores do problema. Os dois algoritmos foram testados em um microcomputador com processador Pentium IV, 3.0 GHZ e 2 GB de RAM, rodando Windows XP Professional.

O modelo de programação matemática foi modelado em AMPL e resolvido pelo otimizador CPLEX, versão 9.1.

Cada problema-teste foi executado 10 vezes pelos algoritmos *GVILS* e *H-GVILS*, considerando-se como critério de parada o tempo de 15 minutos. Já para o otimizador CPLEX foram considerados três critérios de parada: 15, 60 e 120 minutos de processamento.

7.4 Resultados e análise

A seguir apresentam-se os resultados obtidos pelos dois algoritmos heurísticos propostos e pelo otimizador CPLEX, tendo como referência a função de avaliação dada pela Equação (5.1), à página 48.

7.4.1 CPLEX $\times$ *GVILS*

Na Tabela 4, a coluna “Melhora” indica o quanto o algoritmo *GVILS* melhorou a solução produzida pelo CPLEX-120 (solução obtida pelo CPLEX em duas horas de processamento). Já a coluna “Desvio” mostra o desvio dos valores médios do algoritmo *GVILS* em relação à melhor solução conhecida em cada problema-teste. O algoritmo *GVILS* foi executado 10 vezes, sendo que o critério de parada adotado foi o tempo total de processamento, restrito a 900 segundos.

Tabela 4: Comparação de resultados: CPLEX $\times$ *GVILS*

Problema-teste	CPLEX-15	CPLEX-60	CPLEX-120	Média <i>GVILS</i>	Melhor <i>GVILS</i>	Melhora (%)	Desvio (%)
PADC01	227,32	227,32	227,32	227,22	227,12	0,09	0,043
PADC02	257,30	254,06	254,38	254,38	252,41	0,65	0,780
PADC03	164.040	164.034	164.034	164.089,28	164.070,52	-0,02	0,034
PADC04	164.054	164.054	164.054	164.187,48	164.140,49	-0,05	0,081

Como pode ser observado pela Tabela 4, o algoritmo *GVILS* foi capaz de, em 15 minutos de processamento, gerar soluções de melhor qualidade que o CPLEX em duas horas de processamento nos problemas-teste PADC01 e PADC02. Já nos dois últimos problemas-teste, *GVILS* apresentou soluções piores, mas apenas a um máximo de 0,05% da solução do CPLEX. A variabilidade das soluções finais também foi baixa, variando de 0,03% a 0,78%, no máximo.

O algoritmo *GVILS* não foi comparado com o de Costa (2005), visto que a função de avaliação utilizada por este era diferente e, além disso, o modelo desse autor não contemplava restrições relativas à taxa de utilização de caminhões. A Tabela 5 compara os resultados encontrados pelo algoritmo *GVILS* e pelo CPLEX considerando dois minutos de processamento. O objetivo, neste teste, é verificar a eficiência do algoritmo heurístico em produzir uma solução no tempo de uma tomada de decisão, que é tipicamente pequeno. Nessa tabela, a coluna “Problema-teste” indica o problema-teste analisado; “Melhor Lit.” representa o melhor valor encontrado na literatura para o referido problema-teste; “CPLEX-2” indica o valor encontrado pelo CPLEX após dois minutos de processamento; “*GVILS*” indica o valor médio em 10 execuções do algoritmo *GVILS* em dois minutos de processamento; “Desvio CPLEX” e “Desvio *GVILS*”, calculados conforme as equações (7.1) e (7.2), indicam o desvio percentual das soluções do CPLEX e *GVILS*, respectivamente, em relação ao melhor resultado da literatura. Observa-se que os melhores resultados da literatura foram alcançados pelo *GVILS* em 15 minutos de processamento (problemas-teste PADC01 e PADC02) e pelo CPLEX em duas horas de processamento (PADC03 e PADC04).

$$Desvio_i^{CPLEX} = \frac{f_i^{CPLEX} - f_i^*}{f_i^*} \quad (7.1)$$

$$Desvio_i^{GVILS} = \frac{\bar{f}_i^{GVILS} - f_i^*}{f_i^*} \quad (7.2)$$

Nas equações (7.1) e (7.2), para cada problema-teste i , $\bar{f}_i^{GVILS}$ é o valor médio encontrado pelo algoritmo *GVILS* em dez execuções de 2 minutos cada, f_i^{CPLEX} é o valor encontrado pelo otimizador CPLEX em 2 minutos de processamento e f_i^* é o melhor valor conhecido.

Tabela 5: Comparação de resultados: CPLEX-2 $\times$ *GVILS*

Problema -teste	Melhor Lit.	CPLEX-2	<i>GVILS</i>	Desvio CPLEX-2 (%)	Desvio <i>GVILS</i> (%)
PADC01	227,12	7.228,95	227,73	3.082,88	0,27
PADC02	252,41	7.276,39	256,72	2.782,77	1,71
PADC03	164.034,27	164.043,55	164.099,06	0,01	0,04
PADC04	164.054,04	164.085,84	164.245,03	0,02	0,12

Como pode ser observado na Tabela 5, nos problemas-teste PADC01 e PADC02, o algoritmo *GVILS* foi capaz de gerar soluções melhores que o CPLEX, com um desvio de 0,27% no primeiro problema e de 1,71% no segundo, em relação aos melhores resultados conhecidos. Já nos dois últimos problemas-teste, o algoritmo proposto apresentou soluções médias piores, mas a uma distância média de apenas 0,04% da melhor solução conhecida no primeiro problema e

de 0,12% no segundo.

7.4.2 CPLEX $\times$ *H-GVILS*

A Tabela 6 mostra os resultados da comparação entre o otimizador CPLEX e o algoritmo híbrido *H-GVILS*, o qual combina metaheurísticas com programação matemática. A notação utilizada é a mesma da seção anterior, sendo que as colunas “Média *H-GVILS*” e “Melhor *H-GVILS*” indicam, respectivamente, os valores da solução média e da melhor solução encontrada pelo algoritmo *H-GVILS* em 15 minutos de processamento.

Tabela 6: Comparação de resultados: CPLEX $\times$ *H-GVILS*

Problema-teste	CPLEX-15	CPLEX-60	CPLEX-120	Média <i>H-GVILS</i>	Melhor <i>H-GVILS</i>	Melhora (%)	Desvio (%)
PADC01	227,32	227,32	227,32	227,16	227,12	0,09	0,02
PADC02	257,30	254,06	254,38	253,45	252,41	0,65	0,42
PADC03	164.040	164.034	164.034	164.076,04	164.063,56	-0,02	0,03
PADC04	164.054	164.054	164.054	164.172,49	164.118,79	-0,04	0,07

Como pode ser observado pela Tabela 6, o algoritmo *H-GVILS* foi capaz de, em 15 minutos de processamento, gerar soluções de melhor qualidade que o CPLEX em duas horas de processamento nos problemas-teste PADC01 e PADC02. Já nos dois últimos problemas-teste, o algoritmo proposto apresentou soluções piores, mas apenas a um máximo de 0,04% da solução do CPLEX. A variabilidade das soluções finais também ficou baixa, variando de 0,02% a 0,42%, no máximo.

7.4.3 *GVILS* $\times$ *H-GVILS*

A Tabela 7 mostra os resultados da comparação entre o algoritmo heurístico *GVILS* e o algoritmo híbrido *H-GVILS*, tendo-se como referência 15 minutos de processamento. As colunas “Melhora Média” e “Melhora” indicam o percentual de melhora do algoritmo *H-GVILS* sobre *GVILS* em relação às soluções médias e as melhores soluções, respectivamente.

Tabela 7: Comparação de resultados: *GVILS* $\times$ *H-GVILS*

Problema-teste	Média <i>GVILS</i>	Média <i>H-GVILS</i>	Melhora Média (%)	Melhor <i>GVILS</i>	Melhor <i>H-GVILS</i>	Melhora (%)
PADC01	227,22	227,16	0,03	227,12	227,12	0,000
PADC02	254,38	253,45	0,37	252,41	252,41	0,000
PADC03	164.089,28	164.076,04	0,01	164.070,52	164.063,56	0,004
PADC04	164.187,48	164.172,49	0,01	164.140,49	164.118,79	0,013

Como pode ser observado pela Tabela 7, o algoritmo *H-GVILS* teve desempenho superior ao do algoritmo *GVILS*, tanto com relação às melhores soluções geradas tanto com relação à média das soluções geradas. No primeiro caso, *H-GVILS* conseguiu ainda gerar soluções até 0,013% melhores que *GVILS*; enquanto no segundo, a melhora média foi de até 0,37%. Este fato mostra a maior robustez do algoritmo, isto é, o algoritmo híbrido é capaz de gerar soluções finais de melhor qualidade e com menor variabilidade.

A comparação entre os resultados do CPLEX e do algoritmo heurístico híbrido *H-GVILS* em dois minutos de processamento não é mostrada. Isto se deve ao fato de que o algoritmo *H-GVILS* não aciona o módulo de integração com o método exato nesse tempo de processamento. Desta forma, ele se comporta apenas como o algoritmo *GVILS*, o qual já se mostra superior ao CPLEX nesse tempo de processamento.

7.4.4 Características das melhores soluções obtidas

Na Tabela 8 são apresentadas as principais características das melhores soluções encontradas nos quatro problemas-teste. As melhores soluções do PADC01 e PADC02 foram obtidas pelo algoritmo *H-GVILS*, enquanto que nos dois últimos problemas-teste, as melhores soluções foram alcançadas pelo CPLEX.

Por esta Tabela, percebe-se o quão próxima as soluções dos problemas-teste ficaram das metas estabelecidas com relação aos parâmetros de controle, produção e produtividade dos equipamentos de carga.

No primeiro problema-teste, PADC01, considerando as duas casas decimais de precisão do problema, quase não há distinção entre as metas dos parâmetros de controle e seus valores obtidos na melhor solução encontrada. Podem ser encontradas diferenças de no máximo 0,01% da especificação, observando-se os parâmetros Par02, Par05 e Par10.

No segundo problema, PADC02, podem ser encontradas maiores diferenças entre as metas de especificação e valores obtidos, sendo no máximo de 0,09% no parâmetro de controle Par01. Assim como no PADC01, não é possível atingir a meta de produção de estéril, visto que os caminhões carregam 50 t/h ou 80 t/h e a meta é de 1798 t/h.

Os problemas PADC03 e PADC04 são diferentes dos dois primeiros em termos de maior número de frentes de lavra, e menor número de carregadeiras disponíveis. Muitos parâmetros de controle foram atingidos na meta, havendo baixos desvios em alguns outros.

A meta de produção de minério foi atingida em todos os problemas-teste e todas as carrega-

deiras foram utilizadas dentro dos limites de produtividade estabelecidos. Todos os parâmetros de controle foram encontrados também dentro dos limites de especificação, sendo tais soluções, portanto, totalmente viáveis para aplicações reais.

Tabela 8: Características das melhores soluções encontradas

Problema-teste		PADC01		PADC02		PADC03		PADC04	
Características		Meta	Obtido	Meta	Obtido	Meta	Obtido	Meta	Obtido
Parâmetros	Par01	3,30%	3,30%	3,30%	3,39%	47,91%	48,00%	47,90%	47,90%
	Par02	4,24%	4,23%	4,24%	4,30%	0,47%	0,53%	0,47%	0,77%
	Par03	1,63%	1,63%	1,63%	1,71%	0,26%	0,29%	0,26%	0,36%
	Par04	3,92%	3,92%	3,92%	3,97%	0,04%	0,04%	0,04%	0,04%
	Par05	1,07%	1,06%	1,07%	1,05%	0,04%	0,05%	0,04%	0,05%
	Par06	3,03%	3,03%	3,03%	3,05%	1,88%	1,70%	1,88%	2,57%
	Par07	1,13%	1,13%	1,13%	1,15%	2,15%	1,98%	2,15%	2,13%
	Par08	1,24%	1,24%	1,24%	1,18%	16,50%	16,50%	16,50%	17,10%
	Par09	1,77%	1,77%	1,77%	1,76%	80,94%	81,20%	80,90%	81,20%
	Par10	1,71%	1,70%	1,71%	1,75%	43,00%	43,30%	43,00%	43,40%
Produto	Estéril	1798 t/h	1800 t/h	1798 t/h	1800 t/h	1640 t/h	0 t/h	1640 t/h	0 t/h
	Minério	5800 t/h	5800 t/h	5800 t/h	5800 t/h	4100 t/h	4100 t/h	4100 t/h	4100 t/h
Carregadeiras	Car01	900 t/h	900 t/h	900 t/h	900 t/h	500 t/h	500 t/h	500 t/h	500 t/h
	Car02	900 t/h	900 t/h	900 t/h	900 t/h	400 t/h	400 t/h	400 t/h	400 t/h
	Car03	900 t/h	900 t/h	900 t/h	900 t/h	400 t/h	400 t/h	400 t/h	400 t/h
	Car04	900 t/h	900 t/h	900 t/h	900 t/h	500 t/h	500 t/h	500 t/h	500 t/h
	Car05	1000 t/h	960 t/h	1000 t/h	960 t/h	800 t/h	800 t/h	800 t/h	800 t/h
	Car06	1000 t/h	960 t/h	1000 t/h	960 t/h	900 t/h	900 t/h	900 t/h	900 t/h
	Car07	1100 t/h	1040 t/h	1100 t/h	1040 t/h	600 t/h	600 t/h	600 t/h	600 t/h
	Car08	1100 t/h	1040 t/h	1100 t/h	1040 t/h	-	-	-	-
Número de Caminhões		-	25	-	25	-	17	-	17

8 *Conclusões e Trabalhos Futuros*

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões.

Em virtude da complexidade combinatória do problema, foram propostos dois algoritmos heurísticos híbridos. O primeiro deles, denominado *GVILS*, combina os procedimentos heurísticos *GRASP*, *Variable Neighborhood Descent* e *Iterated Local Search*, enquanto o segundo, denominado *H-GVILS*, adiciona ao anterior um módulo de busca local feito por um otimizador de programação matemática.

Usando problemas-teste da literatura, os dois algoritmos heurísticos foram comparados entre si e com o otimizador CPLEX aplicado exclusivamente a um modelo de programação matemática.

Dado que a tomada de decisão no problema em pauta tem que ser rápida, as soluções geradas pelos algoritmos heurísticos *GVILS* e *H-GVILS* foram comparadas com aquelas geradas pelo otimizador CPLEX impondo-se como critério de parada 2 (dois) minutos de processamento. Verificou-se que os dois algoritmos heurísticos foram capazes de gerar soluções muito melhores que as do CPLEX nesse tempo de processamento. Esse resultado valida, portanto, a utilização dos algoritmos heurísticos propostos enquanto ferramenta de apoio à decisão.

Adicionalmente, verificou-se que à medida que o tempo de processamento aumentava, os algoritmos heurísticos se tornavam mais robustos, produzindo soluções finais de qualidade ainda melhor e com baixa variabilidade. As soluções geradas pelos algoritmos heurísticos em 15 minutos de processamento foram competitivas com aquelas produzidas pelo CPLEX nesse mesmo tempo ou até em uma hora de processamento. Comparando-se os algoritmos heurísticos entre si, mostrou-se que *H-GVILS* mostrou-se superior, tanto em termos da qualidade das soluções médias, das melhores soluções e desvio em relação aos melhores resultados da literatura.

Para trabalhos futuros, sugere-se adicionar ao *H-GVILS* um módulo de Reconexão por Caminhos (*Path Relinking* - (GLOVER, 1996)) como uma estratégia de intensificação a cada

ótimo local encontrado após a fase de busca local, bem como implementar outras estratégias de integração entre métodos heurísticos e exatos, visando a geração de soluções com menores desvios e de melhor qualidade.

Referências

- ALARIE, S.; GAMACHE, M. Overview of solution strategies used in truck dispatching systems for open pit mines. *International Journal of Surface Mining, Reclamation and Environment*, v. 16, p. 59–76, 2002.
- ALVARENGA, G. B. *Despacho ótimo de caminhões numa mineração de ferro utilizando algoritmo genético com processamento paralelo*. Dissertação (Dissertação de Mestrado) — Programa de Pós-Graduação em Engenharia Elétrica/UFMG, Belo Horizonte, 1997.
- BRESINA, J. L. Heuristic-biased stochastic sampling. In: *Proceedings of the 13th National Conference on Artificial Intelligence, AAAI Press*. Portland: [s.n.], 1996. p. 271–278.
- CHANDA, E. K. C.; DAGDELEN, K. Optimal blending of mine production using goal programming and interactive graphics systems. *International Journal of Surface Mining, Reclamation and Environment*, v. 9, p. 203–208, 1995.
- COSTA, F. P. *Aplicações de Técnicas de Otimização a Problemas de Planejamento Operacional de Lavra em Minas a Céu Aberto*. Dissertação (Dissertação de Mestrado) — Departamento de Engenharia de Minas/EM/UFOP, Ouro Preto, 2005.
- COSTA, F. P.; SOUZA, M. J. F.; PINTO, L. R. Um modelo de alocação dinâmica de caminhões. *Revista Brasil Mineral*, v. 231, p. 26–31, 2004.
- COSTA, F. P.; SOUZA, M. J. F.; PINTO, L. R. Um modelo de programação matemática para alocação estática de caminhões visando ao atendimento de metas de produção e qualidade. *Revista da Escola de Minas*, v. 58, p. 77–81, 2005.
- FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.
- GERSHON, M. A linear programming approach to mine scheduling optimization. In: *Proceedings of the 17th Application of computers and operations research in the mineral industry*. New York: [s.n.], 1982. p. 483–493.
- GLOVER, F. Computing tools for modeling, optimization and simulation: Interfaces in computer science and operations research. In: _____. [S.l.]: Kluwer Academic Publishers, 1996. cap. Tabu search and adaptive memory programming - Advances, applications and challenges, p. 1–75.
- GUIMARÃES, I. F.; PANTUZA, G.; SOUZA, M. J. F. Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. In: *Anais do XIV Simpósio de Engenharia de Produção (SIMPEP)*. Bauru, CD-ROM: [s.n.], 2007. p. 11.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers, 2003.

MARAN, J.; TOPUZ, E. Simulation of truck haulage systems in surface mines. *International Journal of Surface Mining*, v. 2, p. 43–49, 1988.

MERSCHMANN, L. H. C. *Desenvolvimento de um sistema de otimização e simulação para análise de cenários de produção em minas a céu aberto*. Dissertação (Dissertação de Mestrado) — Programa de Engenharia de Produção/COPPE/UFRJ, Rio de Janeiro, 2002.

MLADENOVIĆ, N.; HANSEN, P. Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097–1100, 1997.

PINTO, L. R.; BIAJOLI, F. L.; MINE, O. M. *Uso de otimizador em planilhas eletrônicas para auxílio ao planejamento de lavra*. Relatório Técnico FAPEMIG, Programa de Pós-graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, Ouro Preto, Minas Gerais, 2003.

PINTO, L. R.; MERSCHMANN, L. H. C. Planejamento operacional da lavra de mina usando modelos matemáticos. *Revista Escola de Minas*, v. 54, n. 3, p. 211–214, 2001.

RODRIGUES, L. F. *Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto*. Dissertação (Dissertação de Mestrado) — Departamento de Engenharia de Produção/Escola de Engenharia/UFMG, Belo Horizonte, 2006.

WHITE, J. W.; ARNOLD, M. J.; CLEVINGER, J. G. Automated open-pit truck dispatching at Tyrone. *Engineering and Mining Journal*, v. 183, n. 6, p. 76–84, 1982.

WHITE, J. W.; OLSON, J. P. Computer-based dispatching in mines with concurrent operating objectives. *Mining Engineering*, v. 38, n. 11, p. 1045–1054, 1986.

ALGORITMOS HEURÍSTICOS PARA O PROBLEMA DA DIVERSIDADE MÁXIMA

Ivan Luis Duarte

Geiza Cristina da Silva (*)

Tatiana Alves Costa

Universidade Federal de Ouro Preto

Departamento de Ciências Exatas e Aplicadas – Rua 37 nº. 115 – Loanda

35930-970 João Monlevade – Minas Gerais – Brasil

iiiivan@yahoo.com.br

{geiza, tatiana}@decea.ufop.br

RESUMO

O Problema da Diversidade Máxima (PDM) é um problema da área de Otimização Combinatória que tem por objetivo selecionar um número pré-estabelecido de elementos de um dado conjunto de maneira tal que a soma das diversidades entre os elementos selecionados seja a maior possível. Por pertencer à classe de problemas NP-Difícil, o desenvolvimento de novos métodos heurísticos na solução aproximada deste problema em tempo computacional aceitável, torna-se atrativo. Neste trabalho são propostos algoritmos heurísticos baseados nos conceitos da metaheurística Iterated Local Search. Resultados computacionais mostram que os algoritmos propostos apresentam um desempenho médio superior quando comparados com as melhores heurísticas da literatura.

PALAVARAS CHAVE. Problema da Diversidade Máxima, Iterated Local Search, metaheurísticas.

Áreas Principais: MH – Metaheurística, OC - Otimização Combinatória.

ABSTRACT

The Maximum Diversity Problem (MDP) is a problem of the Combinatorial Optimization area which asks to select a specified number of elements from a given set so that the sum of diversity between the selected elements is as large as possible. MDP belongs to the NP-hard class becoming attractive to the development of new heuristics to solve the problem in acceptable computational time. In this work heuristics algorithms based on concepts of the Iterated Local Search metaheuristic are proposed. Computational results show that the proposed algorithm produces an average performance better than well known heuristics from the literature.

KEYWORDS. Maximum Diversity Problem, Iterated Local Search, metaheuristics.

Main Areas: MH – Metaheuristic, OC – Combinatorial Optimization.

1. Introdução

O Problema da Diversidade Máxima (PDM) consiste em, dado um conjunto N com n elementos, selecionar um subconjunto $M \subset N$ de forma tal que os elementos de M possuam a maior diversidade possível entre eles, sendo a diversidade expressa por uma função d_{ij} , definida sobre todo $i < j \in M$.

Existem dois critérios através dos quais o problema pode ser descrito. Na abordagem MAXSUM o objetivo é maximizar a soma das distâncias entre os elementos de M , ou seja, $\text{Max } z(M) = \sum_{i,j \in M, i < j} d(i,j)$. A abordagem MAXMIM, ao contrário, considera a pior diversidade. O objetivo, neste caso, é $\text{Max } z(M) = \min\{d_{ij}, i < j, j \in M\}$.

Kuo et al. (1993) provaram que MAXSUM é NP-Difícil, utilizando uma redução ao problema Clique em grafo. Ghosh (1996) provou que assim como MAXSUM, MAXMIM é NP-Difícil, a partir de uma redução ao problema de recobrimento de vértices.

O estudo de complexidade do problema mostra claramente a limitação do uso exclusivo de métodos exatos para a sua solução. Neste trabalho é proposto um algoritmo de construção aleatório e guloso, assim como três heurísticas de busca local. Estes algoritmos são aplicados aos conceitos da metaheurística ILS (Iterated Local Search) para instâncias do problema abordando o critério MAXSUM. Os testes computacionais realizados mostraram que os algoritmos propostos apresentam um desempenho médio superior quando comparados com as melhores heurísticas da literatura.

O restante deste artigo está organizado conforme segue. Na seção 2 o Problema da Diversidade Máxima é apresentado. Na seção 3 é feita uma revisão da literatura assim e dois algoritmos da literatura conhecidos pelos bons resultados alcançados para o problema são descritos. Na seção 4 são discutidas as propostas de algoritmos deste trabalho. A seção 5 apresenta os resultados computacionais e finalmente, a seção 6 conclui o trabalho.

2. O Problema da Diversidade Máxima

O objetivo do Problema da Diversidade Máxima (PDM) é selecionar, a partir de um conjunto N de cardinalidade n , um subconjunto $M \subset N$ com m elementos que possuam a maior diversidade entre si. Uma medida de diversidade entre dois elementos pode ser representada pela distância entre eles. A distância, neste caso, é calculada com base no conjunto de atributos (componentes) dos elementos de N .

Desta forma, para definir o problema, considera-se o conjunto de índices $N = \{1, 2, \dots, n\}$, e:

- t , o número de atributos que caracterizam os elementos de N ;
- a_{ik} , o valor do atributo k , $k = \{1, \dots, t\}$ do elemento i , $i \in N$.
- $d(i,j)$, o índice de diversidade entre dois elementos distintos, onde i e j possuem os respectivos valores de atributos $(a_{i1}, a_{i2}, \dots, a_{it})$ e $(a_{j1}, a_{j2}, \dots, a_{jt})$.

O índice de diversidade representa o grau de diferença existente entre os atributos dos elementos i e j , ou seja, a distância entre eles, que deve ser calculada usando-se uma dentre as métricas de distâncias existentes. De acordo com a área de aplicação, uma métrica pode refletir melhor o problema que outra (Kochenberg e Glover, 1999).

Seja $D [n \times n]$ uma matriz simétrica que armazena os índices de diversidade entre cada um dos elementos de N , preenchida da seguinte maneira: $d(i,j) = d(j,i)$ e $d(i,i) = 0$, $\forall (i,j) \in N$.

O valor da diversidade de M é determinado pela fórmula $\text{div}(M) = \sum_{i,j \in M, i < j} d(i,j)$.

O Problema da Diversidade Máxima pode ser aplicado nas mais diversas áreas relacionadas ao campo da pesquisa operacional como na administração de recursos humanos (Dhir et al., 1994), (Weitz e Lakshminarayanan, 1998), biologia (Agrafiotis, 1997) e preservação ecológica (Maser, 1999), (Unkel, 1985).

O trabalho de Kuo et al. (1993), além da definição do problema, apresentou os dois modelos matemáticos para o PDM mostrados a seguir.

O PDM pode ser representado como um Problema de Programação Inteira considerando-se $x_i, \forall i \in N$, variável binária igual a 1, se e somente se o elemento i estiver na solução e 0, caso contrário, e $d(i, j)$ representando a diversidade entre i e j , $i, j \in N$.

(f_1) Maximizar

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^n d(i, j) x_i x_j \quad (1)$$

sujeito a:

$$\sum_{i=1}^n x_i = m, \quad (2)$$

$$x_i \in \{0,1\}, \forall i = 1, \dots, n. \quad (3)$$

A formulação f_1 , descrita pelas equações 1-3 é um modelo não linear devido a sua função objetivo quadrática (1) e tem como meta maximizar a soma das diversidades de m elementos de N . A igualdade (2) exige que exatamente m elementos façam parte de uma solução viável e (3) indica que as variáveis do problema devem ser binárias.

Uma versão linearizada de f_1 descreve o PDM como um Problema de Programação Linear Inteira. Para isto, além das notações da formulação anterior, devem-se considerar as variáveis de decisão y_{ij} que se tornam iguais a 1 se i e j fizerem parte da solução e 0, caso contrário, $\forall i, j \in N$, $i < j$ e $Q = \{(i, j): i, j \in N, i < j\}$.

(f_2) Maximizar

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^n d(i, j) y_{ij} \quad (4)$$

sujeito a:

$$\sum_{i=1}^n x_i = m, \quad (5)$$

$$x_i + x_j - y_{ij} \leq 1, \forall (i, j) \in Q, \quad (6)$$

$$-x_i + y_{ij} \leq 0, \forall (i, j) \in Q, \quad (7)$$

$$-x_j + y_{ij} \leq 0, \forall (i, j) \in Q, \quad (8)$$

$$y_{ij} \in \{0,1\}, \forall (i, j) \in Q, \quad (9)$$

$$x_i \in \{0,1\}, \forall i \in N. \quad (10)$$

Na formulação f_2 , a função objetivo (4) deve maximizar a soma das diversidades de m elementos de N . A igualdade (5) exige que exatamente m elementos pertençam a uma solução viável. Nas restrições (6) pode-se observar que a variável binária y_{ij} assume valor 1 sempre que x_i e x_j receberem tal valor e y_{ij} assume valor 0 se x_i e x_j forem ambas iguais a 0 ou se x_i ou x_j (somente uma delas) for igual a 1. As restrições (7) exigem que no caso em que x_i pertencer à solução então deverá existir algum j tal que $y_{ij} = 1$. Da mesma forma que em (7), as restrições (8) exigem que se x_j pertencer à solução então deverá existir algum i tal que $y_{ij} = 1$. As restrições (9) e (10) representam as variáveis binárias do problema, sendo $y_{ij} = 1$ se x_i e x_j estiverem na solução e 0 caso contrário e cada x_i assumindo valor 1 se pertencer à solução e 0, caso contrário.

Outras formulações matemáticas para o PDM foram propostas em Glover (1995) para aplicações específicas na área de meio ambiente.

Além de Kuo et al. (1993), Cutler e Klastori (1997), Bhadury et al. (2000) e Martí et al. (2007) apresentam abordagens exatas para tratar o problema.

A seção 3 descreve os principais trabalhos utilizando metodologias heurísticas para o PDM.

3. Abordagens Anteriores

Em Ghosh (1996) foi proposto um algoritmo GRASP para o PDM e para comprovar a qualidade da heurística proposta foi implementado um algoritmo exato. Os resultados computacionais mostraram que o GRASP proposto alcança uma solução ótima ou uma solução muito próxima da ótima, mostrando desta forma sua eficiência ao menos para instâncias de pequeno porte ($n \leq 40$).

Agrafiotis (1997) explorou o problema de selecionar subconjuntos de componentes químicos. No trabalho introduziu duas medidas para quantificar a diversidade e apresentou uma solução para o problema baseada em simulated annealing. O método foi testado usando bases de dados formadas a partir de métodos estatísticos. Os resultados foram visualizados a partir do algoritmo de mapeamento não-linear de Sammon [Sa69]. O artigo não fornece detalhes dos testes computacionais, apenas cita que o método encontra resultados satisfatórios.

Em Cutler e Klastori (1997) foram apresentadas duas heurísticas para o PDM, uma relaxação lagrangeana e uma heurística gulosa. Para avaliar os resultados obtidos foi implementado um algoritmo branch and bound e, desta forma, os autores concluíram que a heurística gulosa apresentou bons resultados para um conjunto de testes com número de elementos pequeno ($n \leq 40$).

Weitz e Lakshminarayanan (1998) apresentaram cinco heurísticas para o PDM. Este trabalho foi realizado para resolver uma aplicação prática que é criar grupos de projetos em uma classe de estudantes, de forma tal que esses estudantes sejam inseridos em ambientes heterogêneos, baseado em critérios como, por exemplo, nacionalidade, idade e formação. As heurísticas foram testadas somente para uma instância composto de dados reais e foi implementado também um algoritmo exato. Os testes apontaram à heurística LCW (Método Lotfi-Cerveny-Weitz) como sendo a melhor para a solução do problema.

Em Glover et al. (1998) foram propostas quatro heurísticas sendo duas delas, construtivas e as outras destrutivas onde, a partir de uma solução heurística com m elementos ocorre uma eliminação progressiva até que se tenha uma solução viável com n elementos. A eficiência dessas heurísticas foi testada através de comparação com os resultados de um método exato e para isso foram criadas diversas instâncias de populações com até 30 elementos. Foi citado pelos autores que as heurísticas propostas obtiveram em média resultados próximos aos do algoritmo exato (a no máximo 2% do ótimo), com um tempo de computação muito menor.

O trabalho de Kochenberger e Glover (1999) desenvolveu uma heurística busca tabu para encontrar a máxima diversidade. Segundo os autores, a heurística foi testada gerando aleatoriamente instâncias de problemas com populações de tamanho que variam de $n = 100$ a $n = 1000$ elementos onde se desejava obter subconjuntos de m elementos mais diversos com tamanho entre 10% a 30% de n . Os autores citam que testes computacionais foram realizados, mas os resultados não foram comparados com qualquer outro método da literatura.

Silva et al. (2004) apresentaram três algoritmos construtivos e duas heurísticas de busca local utilizados para implementar várias heurísticas GRASP para o PDM. Diversos experimentos computacionais foram realizados com instâncias de médio porte propostas no trabalho e mostraram que seus procedimentos superaram métodos apresentados anteriormente.

Silva et al (2007) incorporaram à abordagem anterior um módulo de reconexão por caminhos. Os testes realizados comprovaram a eficiência do método híbrido proposto.

Aringhieri et al. (2008) propuseram um algoritmo busca tabu para o problema. A busca tabu proposta utiliza os conceitos de lista tabu dinâmica e memória secundária, esta última, armazena as segundas melhores soluções encontradas, mas não utilizadas em cada iteração da

busca tabu e é utilizada para reiniciar a busca em casos específicos. Os autores citam que os resultados alcançados puderam equiparar ou superar os outros métodos.

Em Palubeckis (2008) foi desenvolvido um algoritmo Iterated Tabu Search (ITS). Foi apresentado também um algoritmo steepest ascent. Para comparação foram considerados os dois métodos propostos e os trabalhos de Silva et al. (2007) e Martí (2007). Instâncias de maiores portes foram utilizadas e os resultados reportados demonstram boa eficiência do algoritmo ITS.

Os trabalhos de Duarte e Martí (2007) e Gallego et al. (to appear) têm sido considerados referências de algoritmos heurísticos, obtendo os melhores resultados da literatura para o PDM. Por este motivo, os algoritmos apresentados nestes trabalhos, utilizados para efeito de comparação com a metodologia aqui proposta, são descritos a seguir.

3.1. Algoritmo Tabu_D2

Proposto em Duarte e Martí (2007), implementa uma variante da heurística destrutiva D2 de Glover et al. (1998) para a construção de uma solução. Seja Sel o conjunto de elementos selecionados para compor a solução. No início do procedimento, este conjunto possui todos os elementos do conjunto N, isto é: Sel = N. A cada passo, um elemento i^* de menor $D(i^*)$ é retirado da solução, até que se obtenha Sel = M. O cálculo de $D(i^*)$ é mostrado na Equação 11.

$$D(i^*) = \min_{i \in \text{Sel}} [D(i) - \beta \times R(j) \times \frac{\text{freq}[i]}{\max\_freq} + \delta \times R(j) \times \frac{\text{qual}[i]}{\max\_qual}]. \quad (11)$$

Nesta equação, $D(i)$ é o somatório das distâncias dos pares i e j selecionados para compor as soluções prévias, isto é, $D(i) = \sum_j d_{ij} x_i x_j$; $\text{freq}[i]$ corresponde à frequência na qual o elemento i aparece em soluções prévias e $\text{qual}[i]$ é a média da qualidade (relacionada à função objetivo) das soluções anteriores que incluem o elemento i . Os valores $\max\_freq$ e $\max\_qual$ são os máximos de $\text{freq}[i]$ e $\text{qual}[i]$. Os fatores de penalidade β e δ foram definidos pelos autores como 0,1 e 0,0001, respectivamente. Além disso, $R(j) = \max_{i \in \text{Sel}} (D(i)) - \min_{i \in \text{Sel}} (D(i))$.

Como mecanismo de memória de curto prazo da busca tabu foi implementado o método LS_TS. A cada iteração, um elemento i pertencente à solução é selecionado aleatoriamente. A probabilidade da seleção do elemento i é inversamente proporcional a $D(i)$. O conjunto $N - M$ de elementos fora da solução é investigado, obtendo-se um elemento j que proporciona o primeiro movimento de melhora na troca do elemento i por j . Se nenhum movimento de melhora é obtido, o melhor movimento de piora é escolhido. A troca é realizada e os elementos i e j são incluídos em uma lista tabu. Durante um número de iteração, estes elementos são considerados tabu e, portanto, são proibidos de serem usados nas trocas. O método termina após um número consecutivo de movimentos sem melhora. É citado que o número de iterações no qual um elemento permanece tabu é $0,28 \times m$ para elementos que estão sendo inseridos na solução e de $0,028 \times m$ para os elementos que estão deixando a solução. O número de iterações sem melhora é dado por $0,1 \times n$.

3.2. Algoritmo SS

Em (Gallego, to appear) é proposta uma implementação da metaheurística scatter search, cujos conceitos são detalhados em Laguna e Martí (2003), denominado algoritmo SS. O método consiste em explorar um espaço de soluções que fazem parte de um conjunto de referência, denominado RefSet. Este conjunto armazena boas soluções que foram encontradas durante o processo de busca. Os passos do algoritmo SS são:

1. Um conjunto P com tam soluções diversas é construído utilizando Tabu_D2.
2. O procedimento LS_TS é aplicado como método de melhoria a todas as soluções em P.
3. O conjunto inicial de soluções de referência RefSet consiste das $b = 12$ soluções em P, das quais $q = 30\%$ são escolhidas de acordo com a qualidade das soluções

e $(1 - q)\%$ são escolhidas de acordo com a diversidade das mesmas. Uma medida de distância $dist$ é aplicada para definir a diversidade de uma solução em relação ao conjunto de RefSet. A distância entre uma solução t e as contidas em RefSet, é dada por: $dist(t, RefSet) = bm - \sum_{r=1}^b \sum_{i \in M} x_i^r$. Esta fórmula conta o número de vezes

que cada elemento da solução t aparece nas soluções de RefSet e subtrai este valor da distância máxima possível (bm).

4. O método de geração de subconjuntos considera cada par de soluções de RefSet como subconjuntos de soluções de tamanho 2.
5. O método de combinação das soluções utiliza `Tabu_D2` para a união de elementos das soluções contidas nos subconjuntos definidos no Passo 4.
6. Novamente, o método de melhoria é aplicado a todas as soluções geradas no Passo 5.
7. RefSet é atualizado se for obtida uma solução melhor do que as já armazenadas através dos Passos 5 e/ou 6. Se RefSet tiver sido atualizado, o procedimento retorna ao Passo 4, caso contrário, é encerrado.

4. Iterated Local Search

A metaheurística Iterated Local Search (ILS) é um procedimento iterativo no qual buscas locais são aplicadas a novas soluções de partida, obtidas através de perturbações em soluções de ótimos locais.

O método funciona da seguinte maneira (Lourenço et al, 2002): Dada uma solução corrente s^* , uma perturbação é aplicada levando a uma solução s' . Em seguida, s' é explorada através de um procedimento de busca local através do qual é obtida a solução s^{**} , que corresponde a um ótimo local. Um critério de aceitação é invocado para decidir, entre s^* e s^{**} , qual delas deve se tornar corrente na próxima iteração. A Figura 1 apresenta o pseudocódigo do algoritmo ILS.

Procedimento ILS
$s_0 = \text{GeraSolucaoInicial}$ $s^* = \text{BuscaLocal}(s_0)$ Faça $s' = \text{Perturbacao}(s^*, \text{história})$ $s^{**} = \text{BuscaLocal}(s')$ $s^* = \text{CritérioAceitacao}(s^*, s^{**}, \text{história})$ Enquanto critério de parada não for satisfeito Fim-Procedimento.

Figura 1. Pseudo-código da metaheurística ILS

Existem quatro componentes fundamentais do método a serem considerados, representados no pseudo-código por meio das funções: i) `GeraSolucaoInicial`, ii) `BuscaLocal`, iii) `Perturbação` e iv) `CritérioAceitacao`. Mais detalhadamente, i) retorna uma solução inicial. Esta pode ser gerada aleatoriamente ou através de algum procedimento guloso; ii) retorna um ótimo local; iii) perturba a solução através de um ou mais movimentos na solução corrente; e iv) decide qual solução será considerada na próxima iteração do algoritmo.

Uma perturbação não deve ser demasiadamente pequena para que o procedimento não retorne frequentemente à solução corrente e poucas novas soluções sejam exploradas, assim como, não deve ser grande demais, pois neste caso o método recairia em um algoritmo de múltiplos reinícios aleatórios.

O mecanismo de perturbação junto com o de busca local define a possível transição entre a solução corrente s^* para a solução vizinha s^{**} . O critério de aceitação determina se s^{**} é aceito ou não como a nova solução corrente. Pode servir como um procedimento de

intensificação se for definido de forma que somente melhores soluções são aceitas, isto é, para um problema onde o objetivo é a maximização tem-se: CritérioAceitacao retorna $s^{*'} se $Z(s^{*'}) > Z(s^*)$ e; s^* , caso contrário, onde Z é a função objetivo do problema.$

Neste trabalho, as quatro componentes do ILS foram definidas da seguinte maneira:

1. Construção da solução inicial: Uma solução inicial é construída da seguinte maneira: Para cada elemento $i \in N$ são selecionados os m elementos $j \in N - M$ que possuem os maiores índices de diversidade em relação ao i . As diversidades destes elementos são somadas totalizando si . A partir daí, os m elementos com maiores valores de si são escolhidos para compor a solução.
2. Mecanismo de Perturbação: Foram utilizados sete níveis de perturbação, sendo os seis primeiros definidos com base em um valor percentual que indica o número de trocas que são realizadas entre elementos dentro e fora da solução. No primeiro nível são escolhidos, aleatoriamente, 2% dos elementos pertencentes à solução e, cada um desses, é trocado com um outro elemento não pertencente à mesma, também selecionado de forma aleatória. Nos cinco níveis seguintes, os percentuais adotados são 10, 40, 50, 70 e 80%, respectivamente. A utilização do sétimo nível de perturbação é condicionada aos casos nos quais todos os níveis de perturbação definidos anteriormente resultarem em um número de trocas entre mais de dois elementos. Nestes casos, a perturbação de nível sete é utilizada para alterar levemente a solução. A cada iteração do método um nível de perturbação é escolhido aleatoriamente.
3. Algoritmos de busca local: Na fase de busca local são utilizados três algoritmos baseados nos conceitos clássicos apresentados em Sousa (2005) denominados como melhor vizinho, primeiro de melhora e busca randômica.
 - a. O primeiro é uma implementação do algoritmo de busca local proposto por Ghosh (1996). Cada elemento i pertencente à solução é trocado por cada elemento j fora desta. A cada troca é associada uma melhoria $\Delta z(i, j)$ calculada por: $\Delta z(i, j) = \sum_{u \in M - \{i\}} d(j, u) + d(i, u)$. A cada iteração, todas as possíveis trocas são testadas e a solução obtida da troca dos elementos i, j de maior Δz é considerada corrente para a próxima iteração. A busca prossegue até que, em uma iteração, todas as trocas efetuadas retornem $\Delta z(i, j) < 0$.
 - b. No segundo algoritmo de busca local é calculada a melhoria $\Delta z(i, j)$ referente às trocas e a solução resultante da primeira troca na qual $\Delta z(i, j) > 0$ torna-se a solução corrente da próxima iteração. A busca prossegue até que nenhuma solução de melhora possa ser obtida.
 - c. No terceiro algoritmo, a implementação da busca local randômica consiste em escolher aleatoriamente dois elementos $i \in M$ e $j \in N - M$ e realizar a troca entre eles. Se o valor de $\Delta z(i, j)$ associado a esta troca for maior do que 0 a solução produzida torna-se corrente na próxima iteração. Caso contrário, um novo par de elementos é selecionado aleatoriamente. A busca prossegue até que um determinado critério de parada seja alcançado. Neste trabalho os critérios de parada adotados foram número de iterações sem melhora e tempo de processamento.
4. Critério de aceitação: De acordo com o critério de aceitação adotado, uma solução $s^{*'}$ passa ser corrente caso melhore o valor da função objetivo da solução s^* , ou seja, se $Z(s^{*'}) > Z(s^*)$.

4. Resultados computacionais

Com base na metodologia descrita na Seção 3 foram propostas três diferentes versões de heurísticas ILS, que se diferenciam pelo procedimento de busca local adotado, conforme Tabela 1.

Algoritmo	Busca Local
<i>ILS1</i>	Melhor Vizinho (Ghosh, 1996)
<i>ILS2</i>	Primeiro de Melhora
<i>ILS3</i>	Randômica

Tabela 1. Algoritmos ILS propostos

Os algoritmos propostos foram implementados usando linguagem C e compilados na versão 3.3.3 do GCC. Os testes foram realizados em um computador Intel Pentium IV 3 Ghz com 1 Gb de memória RAM. Cada heurística foi executada 10 vezes com o tempo máximo de processamento de 30 segundos para cada execução.

Para efeito de comparação foi utilizado o conjunto de instâncias proposto em Silva et. al. (2004) e os algoritmos descritos na seção 3. No conjunto de instâncias utilizado, o número de variáveis $n \in \{100, 200, 300, 400, 500\}$ e o parâmetro $m \in \{0.1n, 0.2n, 0.3n, 0.4n\}$. Cada valor de diversidade $d(i, j)$ foi gerado aleatoriamente dentro do intervalo $[0, 9]$.

A Tabela 2 apresenta os valores de diversidades obtidos pelos algoritmos comparados. A coluna 1 apresenta as instâncias e as colunas 2 e 3 mostram os valores de n e m de cada uma destas. Na coluna 4 encontra-se o melhor valor conhecido para cada instância (Martí, 2008). As colunas 5 e 6 apresentam, respectivamente, os valores reportados em Gallego (to appear) para os procedimentos **TabuD2** e **ss**. Estes, correspondem ao melhor valor obtido em 10 execuções de 30 segundos com os algoritmos codificados pelos autores em linguagem Java, executados com Java Runtime Environment 1.5 e testados em um computador Pentium IV 3 GHZ com 3 Gb de memória RAM. Nas colunas 7 a 9 são apresentados os melhores valores de diversidades obtidos, em 10 execuções, pelos algoritmos propostos *ILS1*, *ILS2*, *ILS3*.

Instâncias	n	m	Melhor Valor	Literatura		Propostos		
				TabuD2	ss	ILS1	ILS2	ILS3
<i>Silva.1</i>	100	10	333	333	333	333	333	333
<i>Silva.2</i>	100	20	1195	1195	1195	1195	1195	1195
<i>Silva.3</i>	100	30	2457	2457	2457	2457	2457	2457
<i>Silva.4</i>	100	40	4142	4139	4142	4142	4142	4142
<i>Silva.5</i>	200	20	1247	1245	1247	1247	1247	1247
<i>Silva.6</i>	200	40	4450	4448	4450	4450	4450	4450
<i>Silva.7</i>	200	60	9437	9426	9437	9437	9437	9437
<i>Silva.8</i>	200	80	16225	16225	16225	16225	16225	16225
<i>Silva.9</i>	300	30	2694	2694	2694	2694	2694	2694
<i>Silva.10</i>	300	60	9689	9683	9689	9689	9689	9689
<i>Silva.11</i>	300	90	20743	20715	20743	20743	20743	20743
<i>Silva.12</i>	300	120	35881	35871	35881	35875	35881	35881
<i>Silva.13</i>	400	40	4658	4651	4658	4658	4658	4658
<i>Silva.14</i>	400	80	16950	16929	16950	16956	16956	16956
<i>Silva.15</i>	400	120	36317	36288	36317	36306	36310	36317
<i>Silva.16</i>	400	160	62487	62429	62487	62460	62446	62487
<i>Silva.17</i>	500	50	7141	7117	7141	7141	7136	7133
<i>Silva.18</i>	500	100	26258	26229	26258	26239	26242	26258
<i>Silva.19</i>	500	150	56572	56547	56572	56572	56572	56572
<i>Silva.20</i>	500	200	97344	97269	97344	97317	97344	97335

Tabela 2. Comparação entre os melhores resultados reportados em Martí (2008) e algoritmos ILS propostos

A Tabela 2 demonstra que os algoritmos propostos sempre são capazes de alcançar a melhor solução em cada instância e, para a instância *Silva.14* o melhor resultado até então conhecido na literatura pôde ser melhorado.

A Tabela 3 faz uma comparação entre a diferença percentual do melhor resultado obtido pelos algoritmos propostos e os algoritmos da literatura. Diferenças percentuais iguais a zero são representadas pelo símbolo (-).

Instâncias	Literatura		Propostos		
	TabuD2	SS	ILS1	ILS2	ILS3
<i>Silva.1</i>	-	-	-	-	-
<i>Silva.2</i>	-	-	-	-	-
<i>Silva.3</i>	-	-	-	-	-
<i>Silva.4</i>	0,00072	-	-	-	-
<i>Silva.5</i>	0,00160	-	-	-	-
<i>Silva.6</i>	0,00045	-	-	-	-
<i>Silva.7</i>	0,00117	-	-	-	-
<i>Silva.8</i>	-	-	-	-	-
<i>Silva.9</i>	-	-	-	-	-
<i>Silva.10</i>	0,00062	-	-	-	-
<i>Silva.11</i>	0,00135	-	-	-	-
<i>Silva.12</i>	0,00028	-	0,00017	-	-
<i>Silva.13</i>	0,00150	-	-	-	-
<i>Silva.14</i>	0,00159	0,00035	-	-	-
<i>Silva.15</i>	0,00080	-	0,00030	0,00019	-
<i>Silva.16</i>	0,00093	-	0,00043	0,00066	-
<i>Silva.17</i>	0,00336	-	-	0,00070	0,00112
<i>Silva.18</i>	0,00110	-	0,00072	0,00061	-
<i>Silva.19</i>	0,00044	-	-	-	-
<i>Silva.20</i>	0,00077	-	0,00028	-	0,00009
Soma	0,01669	0,00035	0,00190	0,00216	0,00121

Tabela 3. Diferenças percentuais entre algoritmos da literatura e propostos

A diferença percentual foi obtida calculando-se a diferença entre o melhor valor conhecido para cada instância e os resultados obtidos pelas heurísticas. Este valor é dividido pelo melhor valor.

Pode-se notar que, em se tratando dos algoritmos propostos, pelo menos um deles, é capaz de produzir o melhor valor em todas as instâncias.

Quanto aos algoritmos da literatura considerados, a diferença percentual média é de aproximadamente 2% no caso do algoritmo *TabuD2* e de menos de 0,04% para o algoritmo *SS*.

A robustez dos algoritmos propostos pode ser avaliada de acordo com a diferença percentual entre a melhor solução conhecida e a solução obtida por estes e, ainda, de acordo com a diferença percentual entre a melhor solução conhecida e a solução média alcançada em 10 execuções de cada algoritmo. Estes resultados são apresentados na Tabela 4.

Instâncias	<i>ILS1</i>		<i>ILS2</i>		<i>ILS3</i>	
	% melhor	% média	% melhor	% média	% melhor	% média
<i>Silva.1</i>	-	-	-	-	-	-
<i>Silva.2</i>	-	-	-	-	-	-
<i>Silva.3</i>	-	-	-	-	-	-
<i>Silva.4</i>	-	-	-	-	-	-
<i>Silva.5</i>	-	-	-	-	-	-
<i>Silva.6</i>	-	-	-	0,00022	-	0,00022
<i>Silva.7</i>	-	-	-	-	-	-
<i>Silva.8</i>	-	-	-	-	-	-
<i>Silva.9</i>	-	0,00186	-	0,00074	-	0,00037
<i>Silva.10</i>	-	0,00062	-	0,00052	-	0,00041
<i>Silva.11</i>	-	0,00092	-	0,00067	-	0,00039
<i>Silva.12</i>	0,00017	0,00020	-	-	-	0,00022
<i>Silva.13</i>	-	0,00172	-	0,00064	-	0,00086
<i>Silva.14</i>	-	0,00201	-	0,00083	-	0,00088
<i>Silva.15</i>	0,00030	0,00124	0,00019	0,00072	-	0,00058
<i>Silva.16</i>	0,00043	0,00093	0,00066	0,00078	-	0,00066
<i>Silva.17</i>	-	0,00364	0,00070	0,00224	0,00112	0,00168
<i>Silva.18</i>	0,00072	0,00107	0,00061	0,00065	-	0,00076
<i>Silva.19</i>	-	0,00016	-	0,00055	-	0,00030
<i>Silva.20</i>	0,00028	0,00111	-	0,00035	0,00009	0,00032
Média	0,00190	0,01545	0,00216	0,00891	0,00121	0,00766

Tabela 4. Diferenças percentuais entre as melhores soluções e soluções médias

Primeiramente, é feita uma análise em relação às melhores soluções (colunas 2, 4 e 6): o algoritmo *ILS3* apresenta menores diferenças percentuais, estando em média a 0,1% da melhor solução conhecida. Os algoritmos *ILS2* e *ILS1* têm diferenças percentuais de aproximadamente 0,2%, ambos.

Quanto à diferença percentual média da melhor solução para as soluções médias obtidas, os métodos *ILS1*, *ILS2* e *ILS3* são de aproximadamente 1,5%, 0,9% e 0,8%, respectivamente.

Os resultados apresentados mostram que a heurística ILS produz resultados promissores para o problema.

Conclusões

Neste artigo foi descrita uma metodologia aplicada a obtenção de soluções heurísticas para o problema da diversidade máxima. Os procedimentos implementados no trabalho incluem uma heurística de construção aleatória e gulosa, três algoritmos de busca local baseados em conceitos clássicos e um esquema de perturbação de solução que combinados produzem uma abordagem baseada na metaheurística ILS. Os métodos propostos mostraram-se simples no que se à implementação. A despeito disto, os resultados alcançados demonstram que são superiores, na média, quando comparados aos métodos da literatura para um conjunto formado por 20 instâncias. Em particular, foi possível encontrar uma nova melhor solução para uma destas instâncias. Com isto, conclui-se que o método proposto é promissor no tratamento de instâncias do PDM. A fim de comprovar a robustez do método, nos próximos passos deste estudo, deseja-se utilizar outros conjuntos de instâncias disponíveis na literatura com um número maior de variáveis.

Agradecimentos

Esta pesquisa foi parcialmente suportada pelo CT-Info/MCT/CNPq Universal 15/2007 e PROPP/UFOP. Os autores agradecem também à FAPEMIG pelo apoio financeiro.

Referências

- Agrafiotis, D. K.** (1997), Stochastic algorithms for maximizing molecular diversity, *Journal Chem. Inf. Comput. Sci.*, 37, 841-851.
- Aringhieri, R., Cordone, R. e Melzani, Y.** (2008), Tabu search versus GRASP for the maximum diversity problem, *4OR: A quarterly Journal of Operations Research*, 6, 45-60.
- Bhadury, J., Mighty, E. J., e Damar, H.** (2000), Maximizing workforce diversity in project teams: a network flow approach, *Omega*, 28, 143-153.
- Cutler, M. e Klastorin, T.**, The max diversity problem, Relatório Técnico, University of Washington, Department of Management Science, 1997.
- Dhir, K., Glover, F. e Kuo, C.-C.** (1994), Optimizing diversity for engineering management, *Proceedings of the IEEE International Engineering Management Conference*.
- Duarte, A. E Martí, R.** (2007), Tabu search for the maximum diversity problem, *European Journal of Operational Research*, 178, 71-84.
- Gallego, M., Duarte, A., Laguna, M. Martí, R.** Hybrid heuristics for the maximum diversity problem, *Computational Optimization and Applications (To appear)*.
- Glover, F., Kuo, C.-C. e Dhir, K.** (1995), A discrete optimization model for preserving biological diversity. *Applied Mathematical Modeling*, 19, 696-701.
- Glover, F., Kuo, C.-C. & Dhir, K.** (1998), Heuristic algorithms for the maximum diversity problem. *Journal of Information & Optimization Science*, 109-132.
- Ghosh, J.B.** (1996), Computational aspects of the maximum diversity problem, *Operations Research Letters*, 19, 175-181.
- Kochenberger, G. e Glover, F.**, Diversity data mining, Relatório Técnico, The University of Mississippi, 1999.
- Kuo, C.-C., Glover, F. e Dhir, K.** (1993), Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Science*, 24, 1171-1185.
- Laguna, M. e Martí, R.**, Scatter Search: Methodology and Implementations in C, Kluwer Academic Publishers, Boston, 2003.
- Lourenço, H. R., Martin, O. C. e Stützle, T.**, Iterated Local Search, *Handbook of Metaheuristics*, F. Glover and G. Kochenberger (Eds.), International Series in Operations Research & Management Science, vol. 57. Kluwer Academic Publishers, Norwell, MA., 321-353, 2002.
- Maser, C.** Ecological diversity in sustainable development: the vital and forgotten dimension. Saint Lucie Press, 1999.
- Martí, R., Gallego, M. e Duarte, A.**, An Exact Method for the Maximum Diversity Problem, Relatório Técnico, Univesitat de València, 2007.
- Martí, R.**, Rafael Martí - Departamento de Estadística e Investigación Operativa de la Universitat de València. Disponível em: < <http://www.uv.es/rmarti/> >. Acesso em: 24 de abril de 2008.
- Palubeckis, G.** (2007), Iterated tabu search for the maximum diversity problem, *Applied Mathematics and Computation*, 189, 371-383.
- Resende, M. G., Martí, R., Gallego, M. e Duarte, A.**, GRASP and path relinking for the Max-min diversity problem, Relatório Técnico, AT&T Labs Research, TD-7BSN88, 2008.
- Silva, G. C., Ochi, L. S., e Martins, S. L.** (2004), Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem, *Lecture Notes on Computer Science*, 3059, 498-512.
- Silva, G. C., Andrade, M., Ochi, L. S., Martins, S. L e Plastino, A.** (2007), New heuristics for the maximum diversity problem, *Journal of Heuristics*, 13, 315-336.
- Sousa, M. J. F.** (2008). Inteligência Computacional para Otimização - Notas de aula. Ouro Preto: UFOP. Disponível em: <<http://www.iceb.ufop.br/prof/marcone>>. Acesso em: 22 abril de 2008.

Unkel, W. C., (1985), Natural diversity and national forest. *Natural Areas Journal*, 5, 8-13.

Weitz, R. e Lakshminarayanan, S. (1998), An empirical comparison of heuristic methods for creating maximally diverse group, *Journal of the Operational Research Society*, 49, 635-646.

(*) Autor para quem as correspondências devem ser encaminhadas.

PARALLEL ITERATED LOCAL SEARCH APLICADO AO PLANEJAMENTO OPERACIONAL DE LAVRA

**Sabir Ribas, Igor Machado Coelho
Marcone Jamilson Freitas Souza, David Menotti**

Universidade Federal de Ouro Preto
Departamento de Ciência da Computação
Instituto de Ciências Exatas e Biológicas
35400-000, Ouro Preto, MG
{sabir, igor, marcone, menotti}@iceb.ufop.br

RESUMO

Este trabalho apresenta um algoritmo paralelo baseado na metaheurística *Iterated Local Search*. Para testá-lo, ele é aplicado a um problema que necessita de agilidade na geração da resposta, no caso, o planejamento operacional de lavra em minas a céu aberto. Dada a complexidade combinatória deste problema, foi desenvolvido um algoritmo heurístico híbrido, que combina a paralelização do *Iterated Local Search* com os procedimentos GRASP e *Variable Neighborhood Descent*. O algoritmo paralelo foi implementado em uma arquitetura *multi-core*. Para validá-lo, os resultados obtidos foram comparados com aqueles produzidos pelo otimizador CPLEX e uma versão sequencial do mesmo, todos limitados ao mesmo tempo de processamento. Observou-se melhoria na qualidade das suas soluções finais quando comparadas com as obtidas pela sua versão sequencial. Houve também melhora em relação às soluções encontradas pelo CPLEX.

PALAVRAS CHAVE. Algoritmos Paralelos. *Multi-core*. Metaheurísticas. *Iterated Local Search*. Planejamento operacional de lavra.

ABSTRACT

This work presents a parallel algorithm based on Iterated Local Search metaheuristic. In order to test this algorithm, it is applied to a problem which requires fast answers, *i.e.*, the open-pit mining problem. Given its combinatorial complexity, we develop a hybrid heuristic algorithm, which combines an Iterated Local Search parallel version with GRASP and Variable Neighborhood Descent procedures. The parallel algorithm was implemented on a multi-core architecture. In order to validate the proposed parallel algorithm, its results were compared with the results of both CPLEX and its sequential version, all in the same processing time. Results show improvement in the quality of its final solutions in relation to those obtained by its sequential version. In addition, the proposed algorithm overcomes the solutions found by CPLEX.

KEYWORDS. Parallel Algorithms. Multi-core. Metaheuristics. Iterated Local Search. Open-pit mining.

1. Introdução

O crescimento da demanda por sistemas computacionais de auxílio à tomada de decisão acompanha o aumento da complexidade das cadeias produtivas. Tais sistemas normalmente estão relacionados a proporcionar uma visualização objetiva e clara da situação de uma linha de produção, à resolução de um gargalo nos processos de um sistema produtivo ou à melhoria de qualidade de algum produto.

Muitos problemas encontrados em linhas produtivas são de natureza combinatória e, como tal, o desafio é encontrar a melhor entre as soluções existentes. Para resolvê-los destacam-se duas abordagens: a heurística e a exata. A vantagem de se adotar metodologias exatas é a garantia da obtenção das soluções ótimas para o problema, o que não acontece no caso de heurísticas. Porém o uso de heurísticas é muito atraente quando a prioridade é o tempo de resposta. Heurísticas são métodos de busca geralmente dotados de alguma inteligência e apresentam boas soluções em um tempo relativamente baixo se comparados com os métodos exatos. Assim, métodos heurísticos passaram a ser vastamente empregados na resolução de problemas reais.

Para melhorar o desempenho das heurísticas vêm surgindo na literatura propostas de paralelização desses procedimentos, como os de Resende e Ribeiro (2005), Muhammad (2006) e Campos *et al.* (2006). Isso se deve ao crescimento das facilidades em se montar um *cluster* e também pelo crescente investimento em multiprocessadores por parte dos fabricantes. O tempo de resolução de problemas de otimização poderia ser reduzido se os algoritmos utilizassem os recursos *multi-core* desses processadores assim como versões de sistemas operacionais especializados em ambientes distribuídos e construção rápida de *clusters*, como é o caso das distribuições Linux PelicanHPC e Parallel Knoppix.

De forma a explorar a capacidade dos processadores e das arquiteturas *multi-core* atuais, desenvolve-se neste trabalho uma proposta de paralelização da metaheurística *Iterated Local Search*. Para testá-la, esta é aplicada a um problema que necessita de agilidade na geração da resposta, no caso, o planejamento operacional de lavra em minas a céu aberto com alocação dinâmica de caminhões. A tomada de decisão nesse problema tem que ser rápida já que os custos de produção envolvidos são bastante elevados (Rodrigues, 2006).

Para resolvê-lo, as abordagens variam entre métodos exatos e heurísticos. Entre essas, apontamos os trabalhos de White e Olson (1986), Chanda e Dagdelen (1995), Alvarenga (1997), Merschmann (2002), Costa *et al.* (2004), Rodrigues (2006), Guimarães *et al.* (2007) e Coelho *et al.* (2008). Neste último trabalho é proposto um algoritmo heurístico híbrido que representa uma evolução em relação ao de Costa (2005), por incluir mais restrições e outros movimentos para explorar o espaço de soluções. O algoritmo proposto combina os procedimentos heurísticos GRASP (Feo e Resende, 1995), *Variable Neighborhood Descent* (Mladenovic e Hansen, 1997) e *Iterated Local Search* (Lourenço *et al.*, 2003) para resolver o problema em pauta. Usando quatro problemas-teste da literatura, o algoritmo heurístico foi comparado com o otimizador CPLEX 9.1 aplicado a um modelo de programação matemática. Foram realizados testes envolvendo dois minutos de processamento, considerado adequado para a tomada de decisão. Em dois dos problemas, o algoritmo proposto mostrou-se bastante superior; enquanto nos dois outros ele foi competitivo com o CPLEX, produzindo soluções médias com valores até 0,12% piores, na média.

O restante deste trabalho está organizado como segue. A Seção 2 apresenta as abordagens para a resolução do problema de planejamento de lavra. Na Seção 3 são apresentados e discutidos os resultados obtidos. Finalmente, na Seção 4, são apresentadas as conclusões e propostas para trabalhos futuros.

2. Abordagens

Nesta seção são apresentadas duas abordagens encontradas na literatura para o planejamento operacional de lavra, sendo uma exata e uma heurística sequencial, bem como uma proposta de algoritmo heurístico paralelo.

2.1 Abordagem de Programação Matemática

O modelo matemático de alocação dinâmica de caminhões é o mesmo descrito em Coelho *et*

al. (2008), o qual, por sua vez, representa um aperfeiçoamento daquele proposto em Costa *et al.* (2004). Para sua apresentação, sejam os seguintes parâmetros:

M	Conjunto de frentes de minério;
E	Conjunto de frentes de estéril;
F	Conjunto de frentes formado por $M \cup E$;
T	Conjunto de parâmetros de controle analisados no minério;
C	Conjunto de equipamentos de carga;
V	Conjunto de equipamentos de transporte;
M_r	Ritmo de lavra recomendado relativo a minério (t/h);
M_l	Ritmo de lavra mínimo relativo a minério (t/h);
M_u	Ritmo de lavra máximo relativo a minério (t/h);
E_r	Ritmo de lavra recomendado relativo a estéril (t/h);
E_l	Ritmo de lavra mínimo relativo a estéril (t/h);
E_u	Ritmo de lavra máximo relativo a estéril (t/h);
α^-	Penalidade por desvio negativo da produção de minério;
α^+	Penalidade por desvio positivo da produção de minério;
β^-	Penalidade por desvio negativo da produção de estéril;
β^+	Penalidade por desvio positivo da produção de estéril;
t_{ij}	Valor do parâmetro j na frente i (%);
tr_j	Valor recomendado para o parâmetro de controle j na mistura (%);
tl_j	Valor mínimo admissível para o parâmetro de controle j na mistura (%);
tu_j	Valor máximo admissível para o parâmetro de controle j na mistura (%);
λ_j^-	Penalidade por desvio negativo para o parâmetro de controle j na mistura;
λ_j^+	Penalidade por desvio positivo para o parâmetro de controle j na mistura;
ω_l	Penalidade por uso do l -ésimo caminhão;
Qu_i	Ritmo de lavra máximo para a frente i (t/h);
Tx_l	Taxa máxima de utilização do caminhão l ;
Cl_k	Produção mínima do equipamento de carga k (t/h);
Cu_k	Produção máxima do equipamento de carga k (t/h);
cap_l	Capacidade do caminhão l (t);
T_{il}	Tempo total de ciclo do caminhão l na frente i (min);
g_{lk}	1, se o caminhão l é compatível com o equipamento de carga k ; e 0, caso contrário.

e as seguintes variáveis de decisão:

x_i	Ritmo de lavra da frente i (t/h);
y_{ik}	1, se o equipamento de carga k opera na frente i ; e 0, caso contrário.
n_{il}	Número de viagens que um caminhão l realiza à frente i ;
P_m^-	Desvio negativo de produção de minério em relação ao recomendado (t/h);
P_m^+	Desvio positivo de produção de minério em relação ao recomendado (t/h);
P_e^-	Desvio negativo de produção de estéril em relação ao recomendado (t/h);
P_e^+	Desvio positivo de produção de estéril em relação ao recomendado (t/h);
d_j^-	Desvio negativo do parâmetro de controle j na mistura (t/h);
d_j^+	Desvio positivo do parâmetro de controle j na mistura (t/h);
U_l	1, se o veículo l é usado; e 0, caso contrário.

A seguir, pelas Equações (1)-(26), é apresentada uma formulação de programação linear inteira mista relativa à alocação dinâmica de uma frota heterogênea de caminhões e equipamentos de carga, tendo-se como objetivo alcançar as metas de produção e qualidade de minério e reduzir o número de caminhões necessários.

$$\min \sum_{j \in T} \lambda_j^- d_j^- + \sum_{j \in T} \lambda_j^+ d_j^+ + \alpha^- P_m^- + \alpha^+ P_m^+ + \beta^- P_e^- + \beta^+ P_e^+ + \sum_{l \in V} \omega_l U_l \quad (1)$$

$$\sum_{i \in M} (t_{ij} - tu_j) x_i \leq 0 \quad \forall j \in T \quad (2) \quad \sum_{i \in E} x_i \geq E_l \quad (9)$$

$$\sum_{i \in M} (t_{ij} - tl_j) x_i \geq 0 \quad \forall j \in T \quad (3) \quad \sum_{i \in E} x_i + P_e^- - P_e^+ = E_r \quad (10)$$

$$\sum_{i \in M} (t_{ij} - tr_j) x_i + d_j^- - d_j^+ = 0 \quad \forall j \in T \quad (4) \quad x_i \leq Qu_i \quad \forall i \in F \quad (11)$$

$$\sum_{i \in M} x_i \leq M_u \quad (5) \quad x_i \geq 0 \quad \forall i \in F \quad (12)$$

$$\sum_{i \in M} x_i \geq M_l \quad (6) \quad d_j^+, d_j^- \geq 0 \quad \forall j \in T \quad (13)$$

$$\sum_{i \in M} x_i + P_m^- - P_m^+ = M_r \quad (7) \quad P_m^+, P_m^- \geq 0 \quad (14)$$

$$\sum_{i \in E} x_i \leq E_u \quad (8) \quad P_e^+, P_e^- \geq 0 \quad (15)$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad (16) \quad x_i - \sum_{l \in V} n_{il} cap_l = 0 \quad \forall i \in F \quad (22)$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad (17) \quad \frac{1}{60} \sum_{i \in F} n_{il} T_{il} \leq T_{X_l} \quad \forall l \in V \quad (23)$$

$$y_{ik} \in \{0,1\} \quad \begin{matrix} \forall i \in F, \\ \forall k \in C \end{matrix} \quad (18) \quad U_l - \frac{1}{60} \sum_{i \in F} n_{il} T_{il} \geq 0 \quad \forall l \in V \quad (24)$$

$$x_i - \sum_{k \in C} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad (19) \quad n_{il} \in \mathbb{Z}^+ \quad \begin{matrix} \forall i \in F, \\ \forall l \in V \end{matrix} \quad (25)$$

$$x_i - \sum_{k \in C} Cl_k y_{ik} \leq 0 \quad \forall i \in F \quad (20) \quad U_l \in \{0,1\} \quad \forall l \in V \quad (26)$$

$$n_{il} T_{il} - 60 \sum_{k \in C, g_{ik} \neq 0} y_{ik} \leq 0 \quad \begin{matrix} \forall i \in F, \\ \forall l \in V \end{matrix} \quad (21)$$

Podemos distinguir nesta formulação três grupos de equações. O primeiro diz respeito ao problema de mistura de minérios com metas, sendo formado pelas Equações (2)-(15). O segundo (Equações (16) a (20)) modela a alocação das carregadeiras e a faixa de produtividade que torna viável a utilização desses equipamentos; enquanto o último grupo (Equações (21) a (26)) está relacionado ao transporte de material na mina e a alocação e utilização dos caminhões.

No primeiro grupo, as Equações (7) e (10) modelam o atendimento às metas de produção de minério e estéril, respectivamente; enquanto as Equações (4) visam atender as metas de qualidade estabelecidas para os diversos parâmetros de controle. Para cada uma dessas equações envolvendo metas, há variáveis de desvio que são penalizadas na função objetivo (1). As Equações (2) e (3) impedem que soluções infactíveis com respeito aos limites de especificação dos parâmetros de controle sejam aceitas. As Equações (5) e (6) asseguram que a produção de minério está dentro dos limites $[M_l, M_u]$, enquanto que as Equações (8) e (9) garantem que a produção de estéril esteja na faixa $[E_l, E_r]$. As Equações (11) asseguram que o ritmo de lavra em cada frente não supera aquele estabelecido pelo usuário. As demais equações desse grupo definem que as variáveis envolvidas são não-negativas.

Em relação ao segundo grupo, as Equações (16) definem que em cada frente pode ser alocado, no máximo, um único equipamento de carga, enquanto que as Equações (17) asseguram que cada equipamento de carga pode operar, no máximo, em uma única frente. As Equações (18)

caracterizam as variáveis y_{ik} como binárias. As Equações (19) e (20) limitam, respectivamente, o ritmo de lavra máximo e mínimo, definidos pela carregadeira alocada à frente. Já as Equações (11) limitam o ritmo de lavra máximo definido pelo usuário.

No terceiro grupo, cada Equação (21) faz com que um caminhão somente realize viagens a uma frente onde esteja alocado um equipamento de carga compatível. As Equações (22) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados à frente. As Equações (23) definem que cada caminhão opere no máximo $Tx\%$ em uma hora. As Equações (24), juntamente com a função objetivo, forçam com que os caminhões usados sejam penalizados. As Equações (25) forçam que seja inteiro positivo o número de viagens que um caminhão faz a uma frente. As Equações (26) indicam que as variáveis U_l são binárias.

2.2 Abordagem Heurística

2.2.1 Representação de uma Solução

Uma solução é representada por uma matriz $R = [Y \mid N]$, onde Y é a matriz $|F| \times 1$ e N é a matriz $|F| \times |V|$. Cada célula y_i da matriz $Y_{|F| \times 1}$ representa a carregadeira k alocada à frente i . O valor -1 significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente i , a carregadeira k associada a tal frente é considerada *inativa* e não é penalizada por produção abaixo da mínima para este equipamento de carga (Equação (18) do modelo matemático).

Na matriz $N_{|F| \times |V|}$, cada célula n_{il} representa o número de viagens do caminhão $l \in V$ a uma frente $i \in F$. O valor 0 (zero) significa que não há viagem para aquele caminhão. O valor -1 informa a incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

A partir de Y , N e os tempos de ciclo da matriz $Tc_{|F| \times |V|}$ são determinados $X_{|F| \times 1}$ e $Tcv_{1 \times |V|}$, os quais representam, respectivamente, a quantidade de massa lavrada em cada frente e o somatório dos tempos de ciclo de cada caminhão.

2.2.2 Avaliação de uma Solução

Uma solução s é avaliada pela função f da Equação (27). Esta função, que deve ser minimizada, tem como primeira componente a função objetivo propriamente dita, que corresponde à Equação (1) do modelo de programação matemática. As demais componentes penalizam a ocorrência de inviabilidades da solução s em vista do fato de que os movimentos utilizados para explorar o espaço de soluções (vide Seção 2.2.4) podem gerar soluções infactíveis.

$$f(s) = f^{PM}(s) + f^P(s) + \sum_{j \in T} f_j^q(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \quad (27)$$

Na Equação (27), $f^{PM}(s)$ é uma função que avalia s quanto ao atendimento às metas de produção e qualidade, bem como número de caminhões utilizados; $f^P(s)$ avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril; $f_j^q(s)$ avalia s quanto à inviabilidade em relação ao j -ésimo parâmetro de controle; $f_l^u(s)$ avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l -ésimo caminhão e $f_k^c(s)$, que avalia s quanto ao desrespeito aos limites de produtividade da carregadeira k .

2.2.3 Construção de uma Solução inicial

Uma solução inicial para o problema é obtida por um procedimento construtivo guloso. A construção é feita em duas etapas. As alocações das carregadeiras e a distribuição das viagens às frentes são feitas, na primeira etapa, às frentes de estéril, e na segunda, às frentes de minério. Esta estratégia é adotada tendo em vista que nas frentes de estéril o importante é atender à produção e não é necessário observar a qualidade.

Em seguida, pela Figura 1, são apresentados os procedimentos de construção mencionados. Na classificação dos elementos candidatos à inserção na solução, considera-se que para as frentes de estéril, a melhor frente é a que possui a maior massa, a melhor carregadeira é a que oferece a maior produção e o melhor caminhão é o de maior capacidade. Já para as frentes de minério considera-se que a melhor frente é a que possui o menor desvio dos teores em relação às metas, a melhor carregadeira é a de maior produção e o melhor caminhão é o de menor capacidade.

Procedimento Constrói_Solução_Estéril	
1	Seja s uma solução vazia
2	enquanto (a produção de estéril for menor que a produção recomendada) e (existirem frentes de estéril não utilizadas) faça
3	$frente\_atual \leftarrow$ melhor frente de estéril ainda não utilizada
4	se não há carregadeira na frente $frente\_atual$ então
5	Aloque a melhor carregadeira ainda não alocada
6	se todas as carregadeiras já foram alocadas então Retorne s
7	para cada caminhão l faça
8	se o caminhão l for compatível com a carregadeira da frente $frente\_atual$ então
9	enquanto (a produção de estéril for menor que a produção recomendada) e (o caminhão l pode fazer mais uma viagem) faça
10	Aloque ao caminhão l uma viagem para a frente $frente\_atual$
11	Retorne s ;
Procedimento Constrói_Solução_Minério	
1	enquanto (a produção de minério for menor que a produção recomendada) e (existirem frentes de minério não utilizadas) faça
	Ordene as frentes pelos desvios de meta (ordem crescente)
2	$frente\_atual \leftarrow$ Escolha a melhor frente de minério que pode ser utilizada
3	se não há carregadeira na frente f_atual então
4	Aloque a melhor carregadeira ainda não alocada
5	se todas as carregadeiras já foram alocadas então Retorne s
6	para cada caminhão l faça
7	se o caminhão for compatível com a carregadeira da frente $frente\_atual$ então
8	enquanto (a produção de minério for menor que a produção recomendada) e (o caminhão l pode fazer mais uma viagem) faça
9	Aloque ao caminhão l uma viagem para a frente $frente\_atual$
10	Retorne s ;

Figura 1: Procedimentos para construção de uma solução inicial

2.2.4 Estruturas de vizinhança

Para explorar o espaço de soluções do problema foram utilizados 8 movimentos, apresentados a seguir, sendo os seis primeiros propostos em Costa (2005) e os dois últimos propostos em Coelho *et al.* (2008).

Movimento Número de Viagens - $N^{NV}(s)$: Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

Movimento Carga - $N^{CG}(s)$: Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, trocar os equipamentos de carga que operam nas frentes i e k , caso as duas frentes possuam equipamentos de carga alocados. No caso de apenas uma das frentes possuir equipamento de carga e a outra estiver disponível, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas juntamente com as frentes escolhidas.

Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$: Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl} . Desta forma, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$: Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik} . Portanto, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Operação Frente - $N^{OF}(s)$: Consiste em retirar de operação a carregadeira que esteja em operação na frente i . Todas as viagens de caminhões feitas a essa frente são, também, eliminadas. A carregadeira retorna à operação assim que uma nova viagem é associada a ele.

Movimento Operação Caminhão - $N^{OC}(s)$: Consiste em selecionar uma célula $n_{il} \in N$ e zerar seu conteúdo, isto é, retirar de atividade um caminhão l que esteja operando em uma frente i .

Movimento Troca de Viagens - $N^{VT}(s)$: Duas células da matriz N são selecionadas e uma viagem é realocada de uma para outra. Tal movimento pode ocorrer entre quaisquer células da matriz N , respeitando-se as restrições de compatibilidade entre equipamentos.

Movimento Troca de Carregadeiras - $N^{CT}(s)$: Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, em trocar as carregadeiras que operam nas frentes i e k . Analogamente ao movimento CG , em um movimento CT os equipamentos de carga das frentes são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

2.2.5 Algoritmo heurístico sequencial GVILS

O algoritmo heurístico sequencial GVILS, proposto em Coelho *et al.* (2008), combina os procedimentos *Greed Randomized Adaptive Search Procedure* – GRASP (Feo e Resende, 1995), *Variable Neighborhood Descent* – VND (Mladenovic e Hansen, 1997) e *Iterated Local Search* – ILS (Lourenço *et al.*, 2003). Seu pseudocódigo é esquematizado na Figura 2.

Algoritmo GVILS (s)	
1	$s \leftarrow \text{ConstróiSoluçãoEstéril}()$
2	$s \leftarrow \text{ConstróiSoluçãoMinério}()$
3	$s \leftarrow \text{VND}(v')$
4	enquanto (Critério de parada não satisfeito) faça
5	$s' \leftarrow \text{Perturbação}(s, \text{nível})$
6	$s'' \leftarrow \text{VND}(s')$
7	se $f(s'') < f(s)$ então
8	$s \leftarrow s''$
9	fim-se
10	fim-enquanto
11	$s \leftarrow s^*$
12	retorne s

Figura 2: Algoritmo heurístico sequencial GVILS

A construção de uma solução inicial (linhas 1 e 2 do algoritmo GVILS) é feita pelos procedimentos descritos na Seção 2.2.4. A busca local é feita pelo procedimento VND usando-se os movimentos descritos na seção anterior e opera nas vizinhanças em uma ordem pré-definida, começando das que exigem menor esforço computacional para aquelas que exigem maior. Assim, o VND segue a seguinte ordem de exploração: N^{CG} , N^{NV} , N^{VC} , N^{VF} . Os movimentos relativos às vizinhanças N^{CT} , N^{OC} , N^{OF} e N^{VT} não foram utilizados na busca local, mas apenas como perturbação. A justificativa para esta estratégia é o fato de que estes últimos quatro movimentos requerem um elevado tempo computacional durante a busca local, sem um benefício proporcional ao esforço despendido.

Uma das principais características de algoritmos baseados em ILS é o conceito de perturbação. Seu objetivo é diversificar a busca, gerando uma solução diferente e cada vez mais “distante” da região atual de exploração no espaço de busca. Para cumprir esta missão, comumente são estabelecidos vários níveis de perturbação. No trabalho em questão, para cada nível n , são aplicados à solução corrente $n+2$ movimentos, escolhidos aleatoriamente dentre os 8 descritos na Seção 2.2.4. A essa solução perturbada é aplicada busca local, baseada no procedimento VND (linha 6 do algoritmo GVILS). Após *IterMax* iterações sem melhora em um dado nível, este é aumentado. Encontrando uma solução melhor, a perturbação volta ao seu nível mais baixo.

2.2.6 Parallel Iterated Local Search (PILS)

Neste trabalho, propomos uma paralelização da metaheurística *Iterated Local Search*, denominada *Parallel Iterated Local Search* (PILS). Seu pseudocódigo é mostrado na Figura 3.

Procedimento <i>Parallel Iterated Local Search</i>	
1	$s_0 \leftarrow \text{GeraSoluçãoInicial}$
2	$s^* \leftarrow \text{BuscaLocal}(s_0)$
3	repita
4	$A \leftarrow P $ cópias de s^*
5	$B \leftarrow \emptyset$
6	para cada $s \in A$ faça em paralelo
7	$s' \leftarrow \text{Perturbação}(s^*, \text{histórico})$
8	$s'' \leftarrow \text{BuscaLocal}(s')$
9	Adicione s'' ao conjunto B
10	fim
11	$s^* \leftarrow \text{CritérioAceitação}(s^*, B, \text{histórico})$
12	até critério de parada satisfeito
13	retorne s^*

Figura 3: *Parallel Iterated Local Search* (PILS)

Nesta figura, s_0 é uma solução inicial; s^* é a melhor solução obtida durante sua execução; P é o conjunto de nodos de processamento; A é um conjunto que armazena $|P|$ cópias da melhor solução até então; B é um conjunto de soluções refinadas a partir do conjunto A ; s' é a solução perturbada; e s'' , obtido pela aplicação da busca local à solução perturbada, é, potencialmente, um ótimo local melhor que as soluções de A . Como se observa na Figura 3, inicialmente, é gerada uma solução inicial e a ela é aplicada uma busca local. Em seguida vem a parte paralela do algoritmo. Para isso, a cada iteração, é criada uma sequência A de soluções idênticas à melhor solução encontrada até o momento, sendo o tamanho da sequência definida como sendo o número de processos que serão criados. Para se aproveitar ao máximo a arquitetura utilizada, o número de processos deve ser igual ao número de nodos de processamento. O próximo passo é a perturbação e o refinamento das soluções presentes em A , os quais são realizados em um nodo de processamento. As soluções refinadas em cada iteração são armazenadas em um conjunto B . Ao fim da parte paralela do algoritmo, é aplicado um critério de aceitação para se determinar a melhor solução obtida até o momento considerando-se a própria solução s^* , o conjunto B e outras soluções eventualmente armazenadas.

2.2.7 PILS Aplicado ao Planejamento Operacional de Lavra

O algoritmo proposto para tratar o Planejamento Operacional de Lavra, chamado PGVILS, é a versão paralela daquele desenvolvido em Coelho *et al.* (2008) usando a estratégia PILS da Figura 3. A fase construtiva de PGVILS é feita pelo procedimento descrito na Seção 2.2.3. A busca local é feita por um procedimento baseado na metaheurística *Variable Neighborhood Descent* (VND) usando-se os movimentos descritos na Seção 2.2.4 e adotando-se a mesma ordem de exploração apresentada na Seção 2.2.5. Além disso, a definição dos níveis de perturbação segue a mesma estratégia estabelecida na Seção 2.2.5.

2.2.8 Sistema Desenvolvido

Para implementar de forma rápida o algoritmo proposto, foi utilizada a biblioteca MapMP¹ que corresponde a uma implementação da abstração *MapReduce*.

MapReduce é uma abstração simples e poderosa, geralmente aplicada ao processamento ou geração de grandes massas de dados. Dean e Ghemawat (2004) apresentam uma visão geral sobre a implementação do *MapReduce* da Google, a qual facilita muito o trabalho de seus programadores. Esse modelo foi feito para processar grandes conjuntos de dados de uma maneira massivamente paralela e é baseado nos seguintes fatores (Lämmel, 2009): (i) iteração sobre a

¹ Disponível em <http://sourceforge.net/projects/mapreducepp>

entrada; (ii) computação sobre cada um dos pares chave/valor da entrada; (iii) agrupamento de todos os valores intermediários por chaves; (iv) iteração sobre os grupos resultantes; (v) redução de cada grupo. O usuário da biblioteca *MapReduce* expressa a computação como duas funções: *map* e *reduce*. *map*, escrita pelo usuário, recebe um par como entrada e produz um conjunto de pares intermediários também na forma chave/valor. A biblioteca *MapReduce* agrupa todos os valores intermediários associados à mesma chave intermediária e os passa à função *reduce*. A função *reduce*, também escrita pelo usuário, aceita uma chave intermediária e o conjunto de valores relacionados àquela chave. Essa função junta esses valores para formar um conjunto possivelmente menor de valores. Tipicamente, zero ou apenas um valor é produzido pela função *reduce*. Os valores intermediários são passados para o usuário da função *reduce*.

Especificamente para problemas de otimização, onde os itens a serem mapeados não necessitam ser ordenados ou agrupados por chaves, pois o que interessa é a solução e uma forma de se medir sua qualidade, o modelo pode ser simplificado. Usuários especificam as funções *map* e *reduce*. A primeira processa um item de um tipo α e gera um tipo β . A segunda mescla todos os dados intermediários, representados por um conjunto de elementos do tipo β , gerado a partir da aplicação da função *map* a um conjunto de dados do tipo α . Programas escritos nesse estilo funcional podem ser automaticamente paralelizados e executados em grandes *clusters*.

MapMP é uma implementação em C++ da abstração *MapReduce* para multiprocessadores. Tal implementação compõe o projeto MapReduce++ que atualmente conta com recursos de paralelização tanto em *threads* quanto em rede. A paralelização por *threads* permite aos usuários o desenvolvimento de procedimentos que utilizam ao máximo a capacidade dos computadores atuais, os quais possuem recursos de multiprocessamento. Já a paralelização em rede permite aos usuários o desenvolvimento de aplicações distribuídas para computação de alto desempenho em *clusters* sofisticados ou em um conjunto de computadores pessoais. O principal objetivo da biblioteca é oferecer ao usuário todo aparato necessário para o desenvolvimento rápido de aplicações paralelas.

3. Resultados e Análise

A aplicação do algoritmo proposto PGVILS ao planejamento operacional de lavra foi desenvolvida em C++ usando o compilador g++ 4.0. O modelo de programação matemático foi resolvido pelo otimizador CPLEX, versão 11. Tanto o algoritmo heurístico quanto o exato foram testados em um PC Pentium Core 2 Quad (Q6600), com 2,4 GHz e 8GB de RAM. O CPLEX foi executado no Windows Vista, e o algoritmo heurístico paralelo na distribuição Linux Ubuntu 8.04LTS, por questões de compatibilidade com a biblioteca de paralelização OpenMP utilizada.

Para testar o algoritmo proposto foram usados quatro cenários de Coelho *et al.* (2008), os quais se referem a dados do planejamento operacional de empresas mineradoras do quadrilátero ferrífero, situadas na região central do Estado de Minas Gerais. Os parâmetros de controle são os teores químicos (Fe, SiO₂, Mn, P, H₂O, entre outros) e granulometrias especificadas para o minério. Tanto os pesos da função de avaliação quanto os parâmetros do algoritmo heurístico foram os mesmos de Coelho *et al.* (2008).

Os melhores resultados atuais para esses problemas-teste são os seguintes: PADC1: 227,12; PADC2: 252,41; PADC3: 164034,27 e PADC4: 164054,04.

A Tabela 1 mostra os resultados obtidos em 15, 30, 60 e 120 segundos de processamento, usando-se 1, 2, 3 e 4 nodos de processamento. Nesta tabela, “Núcleos” indica a quantidade de nós de processamento, as demais colunas indicam o melhor valor e o valor médio da função de avaliação para cada problema-teste (PADC1, PADC2, PADC3 e PADC4) encontrado em 30 execuções do algoritmo. Observa-se que os resultados obtidos pela aplicação do PGVILS em um único núcleo equivalem à aplicação da versão sequencial GVILS.

TABELA 1: Resultados do algoritmo PGVILS

Tempo (segs)	Núcleos	Melhores valores de custo				Valores médios do custo			
		PADC1	PADC2	PADC3	PADC4	PADC1	PADC2	PADC3	PADC4
15	1	227,12	257,40	164090,31	164140,49	4408,83	4431,14	164107,81	164271,73
15	2	227,12	254,80	164089,20	164132,21	2561,24	4143,19	164104,85	164272,55
15	3	227,12	253,37	164080,36	164140,49	2796,25	2852,68	164105,95	164275,18
15	4	227,12	254,80	164058,33	164140,49	3421,54	2218,68	164103,88	164269,16
30	1	227,12	252,41	164073,84	164137,04	1947,76	1681,23	164105,82	164242,80
30	2	227,12	252,41	164080,36	164155,10	947,70	724,88	164102,00	164247,43
30	3	227,12	253,37	164080,36	164146,27	694,23	1280,13	164100,86	164239,58
30	4	227,12	252,41	164072,84	164136,89	694,31	491,40	164105,40	164232,21
60	1	227,12	252,41	164073,84	164111,47	460,77	258,25	164101,43	164231,00
60	2	227,12	252,41	164073,02	164132,21	461,11	255,97	164093,72	164237,06
60	3	227,12	252,41	164073,72	164126,92	227,38	255,02	164095,43	164227,59
60	4	227,12	252,41	164080,36	164107,97	227,42	256,88	164096,15	164203,98
120	1	227,12	252,41	164048,15	164110,29	460,64	254,29	164097,44	164204,40
120	2	227,12	252,41	164052,47	164092,26	227,29	254,97	164086,35	164185,43
120	3	227,12	252,41	164073,02	164121,09	227,34	255,06	164090,93	164173,15
120	4	227,12	252,41	164045,73	164118,79	227,26	254,46	164083,43	164172,10

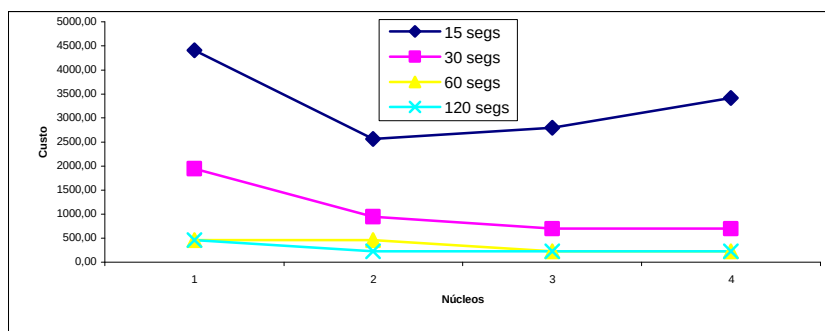


Figura 4: Comportamento médio do PGVILS no PADC1 em relação ao n° de núcleos

Pela Tabela 1 e Figura 4, pode-se observar que à medida que o número de núcleos aumenta há uma tendência de melhoria na qualidade da solução final. Entretanto, em alguns casos, como o relatado na Figura 4, soluções finais obtidas usando-se dois núcleos são melhores que aquelas com três ou quatro núcleos. Isto ocorre devido à aleatoriedade inerente ao método, a qual faz com que a busca caminhe em direções diferentes no espaço de soluções. Assim, não há garantia sempre de que a escolha das regiões de busca feita usando-se três ou quatro núcleos seja mais promissora do que aquela realizada com dois núcleos.

Na Tabela 2 são comparados os resultados obtidos nas abordagens exata (CPLEX), heurística sequencial (GVILS) e heurística paralela (PGVILS) com quatro núcleos. A última coluna apresenta a melhora do valor médio das soluções obtidas pelo PGVILS em relação aos obtidos pelo GVILS, de acordo com $MelhoraMédia = (Média_{GVILS} - Média_{PGVILS}) / Média_{PGVILS}$. O desvio é calculado como segue: $Desvio = (Média - Melhor_{Lit}) / Melhor_{Lit}$, sendo $Melhor_{Lit}$ o melhor valor encontrado na literatura para o problema-teste em questão.

Pela Tabela 2 observa-se que em quinze dos dezesseis casos analisados, o algoritmo paralelo PGVILS obteve soluções finais de maior qualidade, em média, que o GVILS. O PGVILS apresentou melhora significativa nos problemas PADC1 e PADC2. Já nos outros problemas, a melhora foi amena. Nos testes de quinze segundos, a melhora foi de até 99,72%. A maior melhora foi obtida nos testes de 30 segundos: 242,13% em relação à média. Apesar da melhora observada em 15 e 30 segundos, a resolução do problema em tais tempos não é apropriada, principalmente no caso dos problemas PADC1 e PADC2. Nesses casos, o PGVILS ainda apresenta grande variabilidade na qualidade das soluções finais encontradas. Entretanto, percebe-se que 60

segundos é tempo suficiente para a resolução satisfatória do problema. Nesse tempo, o PGVILS é superior ao CPLEX em dois dos quatro problemas. Outro ponto a ser observado é que o PGVILS foi capaz de alcançar a melhor solução da literatura para o problema PADC1 com apenas 15 segundos de processamento e, para o PADC2, a partir de 30 segundos.

TABELA 2 – Comparação do PGVILS com outras abordagens

Probl.	Tempo (segs)	CPLEX	GVILS			PGVILS			Melhora
			Melhor	Média	Desvio	Melhor	Média	Desvio	Média
PADC1	15	90238,416	227,12	4408,83	1841,19%	227,12	3421,54	1406,49%	28,855%
PADC2	15	261,834	257,40	4431,14	1655,53%	254,80	2218,68	779,00%	99,720%
PADC3	15	164043,55	164090,31	164107,81	0,04%	164058,33	164103,88	0,04%	0,002%
PADC4	15	164065,21	164140,49	164271,73	0,13%	164140,49	164269,16	0,13%	0,002%
PADC1	30	90238,416	227,12	1947,76	757,59%	227,12	694,31	205,70%	180,533%
PADC2	30	261,834	252,41	1681,23	566,07%	252,41	491,40	94,68%	242,129%
PADC3	30	164043,31	164073,84	164105,82	0,04%	164072,84	164105,40	0,04%	0,000%
PADC4	30	164065,21	164137,04	164242,80	0,12%	164136,89	164232,21	0,11%	0,006%
PADC1	60	230,3	227,12	460,77	102,88%	227,12	227,42	0,13%	102,609%
PADC2	60	259,746	252,41	258,25	2,31%	252,41	256,88	1,77%	0,531%
PADC3	60	164015,67	164073,84	164101,43	0,04%	164080,36	164096,15	0,04%	0,003%
PADC4	60	164055,68	164111,47	164231,00	0,11%	164107,97	164203,98	0,09%	0,016%
PADC1	120	230,30	227,12	460,64	102,82%	227,12	227,26	0,06%	102,692%
PADC2	120	254,22	252,41	254,29	0,74%	252,41	254,46	0,81%	-0,069%
PADC3	120	164043,55	164048,15	164097,44	0,04%	164045,73	164083,43	0,03%	0,009%
PADC4	120	164085,84	164110,29	164204,40	0,09%	164118,79	164172,10	0,07%	0,020%

4. Conclusões

Este trabalho apresentou uma proposta de paralelização da metaheurística *Iterated Local Search* (ILS) e desenvolveu um algoritmo heurístico paralelo, denominado PGVILS, aplicado ao problema de planejamento operacional de lavra com alocação dinâmica de caminhões.

O algoritmo desenvolvido combina os procedimentos heurísticos ILS, *Variable Neighborhood Descent* (VND) e a fase de construção GRASP e explora a capacidade de multiprocessamento da geração atual de computadores.

O algoritmo PGVILS foi testado usando-se 1, 2, 3 e 4 núcleos de processamento e interrompido após 15, 30, 60 e 120 segundos de execução. Observou-se, como já era esperada, melhoria na qualidade das soluções finais quando comparado com sua versão sequencial. Comparado com os resultados produzidos pelo otimizador CPLEX, o algoritmo proposto conseguiu superá-lo com relação às melhores soluções encontradas e produzir soluções médias competitivas com este.

Como trabalho futuro propõe-se a implementação de uma versão distribuída do algoritmo PGVILS de forma a executá-lo em um maior número de núcleos.

Agradecimentos

Os autores agradecem ao CNPq, processo 474831/2007-8, e à FAPEMIG, processo CEX 2991-06.1/07, pelo apoio ao desenvolvimento da presente pesquisa.

Referências

- ALVARENGA, G. B. *Despacho ótimo de caminhões numa mineração de ferro utilizando algoritmo genético com processamento paralelo*. Dissertação (Mestrado em Engenharia Elétrica), Programa de Pós-Graduação em Engenharia Elétrica, UFMG, Belo Horizonte, 1997.
- CAMPOS, G. D.; YOSHIZAKI, H. T. Y.; BELFIORE, P. P. Algoritmo Genético e computação paralela para problemas de roteirização de veículos com janelas de tempo e entregas fracionadas. *Gestão e Produção*, v. 13, p. 271-281, 2006.

- CHANDA, E. K. C.; DAGDELEN, K. Optimal blending of mine production using goal programming and interactive graphics systems, *International Journal of Surface Mining, Reclamation and Environment*, v. 9, p. 203-208, 1995.
- COELHO, I. M.; RIBAS, S.; SOUZA, M. J. F. Um algoritmo baseado em GRASP, Iterated Local Search para a otimização do planejamento operacional de lavra. In: *Anais do XI Encontro de Modelagem Computacional*, Volta Redonda (RJ), 8 p., 2008.
- COSTA, F. P.; SOUZA, M. J. F. e PINTO, L. R. Um modelo de alocação dinâmica de caminhões. *Revista Brasil Mineral*, v. 231, p. 26-31, 2004.
- COSTA, F. P. *Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto*. 141 p. Dissertação (Mestrado em Engenharia Mineral) - Programa de Pós-Graduação em Engenharia Mineral, UFOP, Ouro Preto, 2005.
- DEAN, J. e GHEMAWAT, S. MapReduce: Simplified Data Processing on Large Clusters, In OSDI'04, 6th Symposium on Operating Systems Design and Implementation, Sponsored by USENIX, in cooperation with ACM SIGOPS, pg 137-150, 2004.
- FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109-133, 1995.
- GUIMARÃES, I. F.; PANTUZA, G. e SOUZA, M. J. F. Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. In: *Anais do XIV Simpósio de Engenharia de Produção (SIMPEP)*, Bauru, CD-ROM, 11 p., 2007.
- LÄMMEL, R. Google's MapReduce Programming Model, Microsoft Corp. In www.cs.vu.nl/~ralf/MapReduce/paper.pdf Acesso em 03/03/2009.
- LOURENÇO, H. R.; MARTIN, O. C.; STUTZLE, T. Iterated Local Search. In: GLOVER, F. e KOCHENBERGER, G. (eds). *Handbook of Metaheuristics*, Kluwer Academic Publishers, 2003.
- MERSCHMANN, L.H.C. *Desenvolvimento de um sistema de otimização e simulação para análise de cenários de produção em minas a céu aberto*. Dissertação (Mestrado em Engenharia de Produção) - Programa de Engenharia de Produção/COPPE, UFRJ, Rio de Janeiro, 2002.
- MLADENOVIC, N.; HANSEN, P. A Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097-1100, 1997.
- MUHAMMAD, R. B. A Parallel Local Search Algorithm for Euclidean Steiner Tree Problem, In *IEEE Proceedings of Seventh International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2006)*, p. 157-164, 2006.
- RESENDE, M. G. C e RIBEIRO, C. C. Parallel Greedy Randomized Adaptive Search Procedures. *Parallel Metaheuristics: A New Class of Algorithms* (E. Alba, ed.), 315-346, Wiley, 2005.
- RODRIGUES, L. F. *Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto*. 86 p. Dissertação (Mestrado em Engenharia de Produção), Programa de Pós-Graduação em Engenharia de Produção, UFMG, Belo Horizonte, 2006.
- WHITE, J. W. e OLSON, J. P. Computer-based dispatching in mines with concurrent operating objectives. *Mining Engineering*, v. 38, n. 11, p. 1045-1054, 1986.

UM ALGORITMO HEURÍSTICO HÍBRIDO PARA O PLANEJAMENTO OPERACIONAL DE LAVRA

Igor Machado Coelho¹, Sabir Ribas¹, Marccone Jamilson Freitas Souza¹, Vitor Nazário Coelho¹

¹Universidade Federal de Ouro Preto, Departamento de Computação, Ouro Preto, Minas Gerais, Brasil

imcoelho@iceb.ufop.br; sabir@iceb.ufop.br; marcone@iceb.ufop.br; vncoelho@gmail.com

Abstract – This paper presents a hybrid algorithm based on Iterated Local Search metaheuristic. In order to test this algorithm, it is applied to a problem which requires fast answers, i.e., the open-pit mining problem. The proposed heuristic algorithm uses GRASP to generate an initial solution and has Variable Neighborhood Descent as a method of local search. Periodically, a mathematical programming module is triggered to solve smaller parts of the problem in optimality. To validate the proposed hybrid method its results are compared to those produced by its pure heuristic version and to the CPLEX solver. The results show the efficiency of the proposed algorithm.

Keywords – Open Pit Mining, GRASP, Variable Neighborhood Descent, Iterated Local Search, Metaheuristics

Resumo – Este trabalho apresenta um algoritmo híbrido baseado na metaheurística *Iterated Local Search* (ILS). De forma a testá-lo, ele é aplicado a um problema que requer decisões rápidas, o problema de planejamento operacional de lavra em minas a céu aberto. O algoritmo heurístico proposto utiliza GRASP para gerar uma solução inicial e tem o *Variable Neighborhood Descent* como método de busca local. Periodicamente, é acionado um módulo de programação matemática para resolver na otimalidade partes menores do problema. Para validá-lo, seus resultados são comparados com aqueles produzidos por sua versão sem o módulo de programação matemática, bem como com os do otimizador CPLEX. Experimentos computacionais mostram a efetividade da proposta.

Palavras-chave – Planejamento operacional de lavra, GRASP, *Variable Neighborhood Descent*, *Iterated Local Search*, Metaheurísticas

1. Introdução

Este trabalho trata do problema de planejamento operacional de lavra com alocação dinâmica de caminhões (POLAD). O objetivo é determinar o ritmo de lavra em cada frente, seja ela de minério ou de estéril, de forma que as metas requeridas de produção e qualidade sejam satisfeitas e o número de caminhões necessários ao processo seja reduzido.

De acordo com [1], este problema pertence à classe NP-difícil, uma vez que tem como subproblema, o Problema da Mochila Múltipla, o qual é NP-difícil. Em vista da dificuldade de resolvê-lo na otimalidade, propõe-se um algoritmo heurístico híbrido para encontrar soluções sub-ótimas. O algoritmo

proposto é baseado na metaheurística *Iterated Local Search* - ILS [2] e utiliza GRASP [3] para gerar uma solução inicial, *Variable Neighborhood Descent* - VND [4] para fazer busca local e tem um módulo de programação matemática para resolver na otimalidade partes menores do problema. O algoritmo proposto representa uma evolução em relação ao de [5], por incluir um módulo de programação matemática.

Dado que a tomada de decisão no problema em pauta deve ser rápida, o desafio é produzir soluções de boa qualidade no menor tempo. Assim, comparações com outras metodologias são feitas tendo-se como foco a qualidade da solução em um dado tempo de processamento.

O restante deste trabalho está organizado como segue. Na Seção 2 são apresentados os trabalhos correlatos. A Seção 3 apresenta a metodologia para a resolução do problema. Na Seção 4 é descrito o algoritmo proposto e a Seção 5 mostra os resultados dos experimentos computacionais. A Seção 6 conclui o trabalho.

2. Trabalhos Correlatos

Em [6] foi proposto um algoritmo, baseado em programação linear e dinâmica, que é a base para o sistema *DISPATCH*, o qual vem operando em muitas minas em todo o mundo [7].

Um modelo de programação linear por metas foi apresentado em [8] para resolver um problema de mistura de minérios no planejamento de curto prazo em uma mina de carvão. O objetivo foi maximizar um critério econômico e reduzir a soma dos desvios dos parâmetros individuais de qualidade e da tonelagem em relação aos valores desejados.

Um sistema para o despacho ótimo de caminhões em uma mineração de ferro foi desenvolvido em [9] com o objetivo de minimizar o tempo de fila da frota de caminhões, aumentar a produtividade desta e melhorar a qualidade do minério lavrado. O sistema tem o Algoritmo Genético com ferramenta de otimização e é a base do sistema SMART MINE, atualmente muito utilizado em várias minas brasileiras.

Um sistema de otimização e simulação, chamado de OTISIMIN, foi desenvolvido em [10] para análise de cenário de produção em minas a céu aberto. Esse modelo de otimização foi aperfeiçoado posteriormente por [11], incluindo-se restrições de atendimento às metas de produção e qualidade. Em [1] foi proposta uma modelagem heurística para o problema, baseada em *Variable Neighborhood Search* (VNS). O algoritmo proposto explora o espaço de soluções com base em seis tipos de movimentos, sendo estes os primeiros descritos na subseção 3.2 do presente trabalho. Pelos experimentos realizados, o VNS foi capaz de gerar soluções de melhor qualidade em um menor tempo de processamento que aquelas obtidas pelo modelo exato, resolvido pelo LINGO.

Em [12], o problema é resolvido por um algoritmo heurístico, denominado GVILS, que combina os procedimentos heurísticos GRASP, VND e ILS. Além dos movimentos de [1], o algoritmo GVILS faz uso de mais outros dois movimentos para explorar o espaço de soluções. Na modelagem heurística considerou-se, também, a minimização do número de veículos. Usando quatro problemas-teste da literatura, o GVILS foi comparado com o otimizador CPLEX 9.1 aplicado a um modelo de programação matemática. Foram realizados testes envolvendo 15 minutos de processamento. Em dois dos problemas, o algoritmo proposto mostrou-se bastante superior; enquanto nos dois outros ele foi competitivo com o CPLEX, produzindo soluções médias com valores até 0,08% piores, na média.

Em [5], são feitos testes com tempos limite menores, comparando a abordagem heurística proposta com o otimizador CPLEX 9.1 aplicado ao mesmo modelo matemático de [12]. Observou-se uma melhora no desempenho da abordagem heurística proposta em relação à resolução do problema com *solvers* exatos quando o tempo de solução é um fator crítico. Os testes foram feitos com um limite de dois minutos de processamento.

Tais resultados motivaram o desenvolvimento do presente trabalho, que busca explorar as características vantajosas de ambas as abordagens heurística e exata.

3. Metodologia

3.1 Modelo Exato

A formulação de programação matemática usada neste trabalho é a mesma de [12], a qual é uma evolução do modelo de [1]. Nesta formulação, considera-se a função de avaliação dada pela Equação (1):

$$\begin{aligned} \min \sum_{j \in T} \lambda_j^- d_j^- + \sum_{j \in T} \lambda_j^+ d_j^+ + \alpha^- P_m^- + \alpha^+ P_m^+ \\ + \beta^- P_e^- + \beta^+ P_e^+ + \sum_{l \in V} \omega_l U_l \end{aligned} \quad (1)$$

Na Equação (1) busca-se minimizar os desvios positivos (d_j^+) e negativos (d_j^-) das metas de cada parâmetro de controle j da mistura, bem como minimizar os desvios positivos e negativos das metas de produção de minério e estéril, representados pelas variáveis de decisão P_m^+ , P_m^- , P_e^+ e P_e^- , respectivamente. Nesta função também considera-se a minimização do número de veículos utilizados, representado pela variável binária U_l , que vale 1 se o veículo l for utilizado e 0, caso contrário.

As constantes λ_j^- , λ_j^+ , α^- , α^+ , β^- , β^+ e ω_l são pesos que refletem a importância de cada componente da função objetivo.

3.2 Modelo Heurístico

Representação de uma solução

Uma solução é representada por uma matriz $R = [Y|N]$, sendo Y a matriz $|F| \times 1$ e N a matriz $|F| \times |V|$. Cada célula y_i da matriz $Y_{|F| \times 1}$ representa a carregadeira k alocada à frente i . O valor -1 significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente i , a carregadeira k associada a tal frente é considerada *inativa* e não é penalizada por produção abaixo da mínima para este equipamento de carga.

Na matriz $N_{|F| \times |V|}$, cada célula n_{il} representa o número de viagens do caminhão $l \in V$ a uma frente $i \in F$. O valor 0 (zero) significa que não há viagem para aquele caminhão. O valor -1 informa a incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

Avaliação de uma solução

Como os movimentos desenvolvidos podem gerar soluções inviáveis, uma solução é avaliada por uma função f baseada em penalidades, dada pela Equação (2). Nesta equação, a componente f_s^M é a função objetivo propriamente dita (mesma do modelo matemático - Equação (1)), e avalia s quanto ao atendimento às metas de produção e qualidade, bem como número de caminhões utilizados. As demais componentes penaliza a ocorrência de inviabilidade na solução corrente. Assim, $f^p(s)$ avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril; $f_j^q(s)$ avalia s quanto à inviabilidade em relação ao j -ésimo parâmetro de controle; $f_l^u(s)$ avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l -ésimo caminhão e $f_k^c(s)$ avalia s quanto ao desrespeito aos limites de produtividade da k -ésima carregadeira.

$$f(s) = f^M(s) + f^p(s) + \sum_{j \in T} f_j^q(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \quad (2)$$

Geração de uma solução inicial

Uma solução inicial para o problema é obtida por um procedimento construtivo parcialmente guloso, semelhante à da fase de construção GRASP. A construção é feita em duas etapas. As alocações das

carregadeiras e a distribuição das viagens às frentes são feitas, na primeira etapa, às frentes de estéril, e na segunda, às frentes de minério.

Esta estratégia é adotada tendo em vista que nas frentes de estéril o importante é atender à produção, não sendo necessário observar a qualidade. A classificação dos elementos candidatos a ser inseridos na solução é feita considerando que, para as frentes de estéril a melhor frente é a que possui a maior massa, a melhor carregadeira é a que oferece a maior produção e o melhor caminhão é o de maior capacidade. Já para as frentes de minério considera-se que a melhor frente é a que possui o menor desvio dos teores em relação às metas, a melhor carregadeira é a que oferece a maior produção e o melhor caminhão é o de menor capacidade.

A escolha da frente na segunda etapa da construção (minério) é feita usando-se uma função guia, tal como em [13]. Inicialmente, todas as frentes candidatas são ordenadas tendo em vista os valores dos desvios das metas de qualidade. À r -ésima frente melhor classificada, é associada a função $bias(r) = 1/(2r)$. A seguir, cada frente candidata é escolhida com probabilidade $p(r) = bias(r) / \sum bias(r)$. Assim, dá-se maior probabilidade de escolha à frente que mais contribui para minimizar os desvios das metas.

Estruturas de Vizinhança

Para explorar o espaço de soluções do problema foram usados oito movimentos, apresentados a seguir, sendo os seis primeiros os de [1].

Movimento Número de Viagens - $N^{NV}(s)$. Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i , onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

Movimento Carga - $N^{CG}(s)$. Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, trocar os equipamentos de carga que operam nas frentes i e k , caso as duas frentes possuam equipamentos de carga alocados. Havendo apenas uma frente com equipamento de carga, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas junto com as frentes escolhidas.

Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$. Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl} . Assim, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$. Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik} . Portanto, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Operação Frente - $N^{OF}(s)$. Consiste em retirar de operação o equipamento de carga que esteja em operação na frente i . O movimento retira todas as viagens feitas a esta frente, deixando o equipamento *inativo*. O equipamento retorna à operação assim que uma nova viagem é associada a ele.

Movimento Operação Caminhão - $N^{OC}(s)$. Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, significando retirar de atividade um caminhão l que esteja operando em uma frente i .

Movimento Troca de Viagens - $N^{VT}(s)$. Duas células da matriz N são selecionadas e uma viagem é realocada de uma célula para outra. Tal movimento pode ocorrer entre quaisquer células da matriz N , respeitando-se as restrições de compatibilidade entre equipamentos.

Movimento Troca de Carregadeiras - $N^{CT}(s)$. Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, trocar os equipamentos de carga que operam nas frentes i e k . Analogamente ao movimento CG , os equipamentos de carga das frentes são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

4. Algoritmo proposto

O algoritmo proposto, denominado H-GVILS, consiste em adicionar um módulo de programação matemática ao algoritmo heurístico GVILS de [5] .

No algoritmo GVILS, o qual é baseado em ILS e cujo pseudocódigo está apresentado na Figura 1, uma solução inicial é construída pelo procedimento descrito na Subseção 3.2. A busca local é feita pelo procedimento VND usando-se os movimentos descritos na subseção 3.2. Estrategicamente, a busca local opera nas vizinhanças em uma ordem pré-definida, começando das que exigem menor esforço computacional para aquelas que exigem maior esforço. Assim, o VND segue a seguinte ordem de exploração: N^{CG} , N^{NV} , N^{VC} , N^{VF} . Os movimentos relativos às vizinhanças N^{CT} , N^{OC} , N^{OF} e N^{VT} não foram utilizados na busca local, mas apenas como perturbação. Isto foi feito tendo em vista que estes últimos movimentos requerem esforço computacional muito elevado, inadequado para o tempo computacional disponível para a tomada de decisão .

```
Algoritmo GVILS( Solução  $s$  )  
1  $s \leftarrow ConstróiSoluçãoEstéril ()$   
2  $s \leftarrow ConstróiSoluçãoMinério( s )$   
3  $s \leftarrow VND( s )$   
4 enquanto ( critério de parada não satisfeito ) faça  
5    $s' \leftarrow Perturbação ( s, nível )$   
6    $s'' \leftarrow VND ( s' )$   
7   se (  $f(s'') < f(s)$  ) então  
8      $s \leftarrow s''$   
9   fim-se  
10 fim-enquanto  
11 Retorne  $s$   
fim GVILS
```

Figura 1: Algoritmo GVILS

O objetivo da perturbação é diversificar a busca, gerando uma solução diferente e cada vez mais *distante* da região atual de exploração no espaço de busca. Para cumprir esta missão, são estabelecidos vários níveis de perturbação. Para cada nível n , são aplicados à solução corrente $n + 2$ movimentos, escolhidos aleatoriamente dentre os oito descritos na Subseção 3.2. A essa solução perturbada é aplicada busca local, baseada no procedimento VND (linha 6 da Fig. 1).

Após *IterMax* iterações sem melhora em um dado nível, este é aumentado. No caso de se encontrar uma solução de melhora, o nível de perturbação volta ao seu nível mais baixo.

O algoritmo H-GVILS difere do GVILS em dois aspectos. A partir de um determinado nível de perturbação, a busca local (linha 6 da Figura 1) não passa a ser feita de forma heurística e sim de forma exata. Para tanto, fixa-se uma parte grande do problema e resolve-se na otimalidade uma parte pequena do POLAD. Além disso, há diferença também na maneira como são feitas as perturbações após esse determinado nível de perturbação ser alcançado.

No H-GVILS, para cada nível p de perturbação, são aplicados à solução corrente $p + 2$ movimentos, escolhidos aleatoriamente dentre os oito descritos na Subseção 3.2. A diferença é que após um nível H de perturbações, o método começa a fazer perturbações da seguinte maneira: inicialmente seleciona-se aleatoriamente 10% das frentes e fixa-se no modelo exato todas as configurações das outras frentes não selecionadas, ou seja, as células das matrizes Y , de carregadeiras e N , de viagens, que não pertencem a estes 10% selecionados. Desta maneira, executa-se o otimizador de programação matemática e retorna-se como solução perturbada a solução encontrada pelo método exato. A cada nível são aumentados mais 10% de frentes livres, de forma que ao término de dez níveis, o método é executado totalmente de forma exata. O procedimento de solução via o otimizador é interrompido em duas situações: (i) após encontrar a solução

ótima ou (ii) após decorridos τ segundos de processamento.

5. Resultados Computacionais

O algoritmo H-GVILS proposto foi desenvolvido em C++ usando o framework de otimização OptFrame, disponível em <https://sourceforge.net/projects/optframe> compilado pelo g++ 4.0. O modelo exato foi resolvido pelo CPLEX 9.1 e pelo GLPK 4.9 na versão híbrida do algoritmo. Os testes foram executados em um microcomputador Pentium IV, 3.0 GHZ e 2 GB de RAM, com Windows XP Professional.

Os cenários utilizados para teste, bem como os parâmetros do algoritmo proposto, foram os mesmos de [5].

Cada problema-teste foi executado 10 vezes pelo algoritmo H-GVILS, considerando-se como critério de parada os tempos de 2 e 15 minutos. A seguir, ele foi comparado com o *solver* CPLEX 9.1, bem como o GVILS de [12], em 15 minutos de execução e de [5], em 2 minutos de execução. Não houve comparação com o algoritmo de [1] visto que este não contemplava restrições relativas à taxa de utilização de caminhões e a função de custo era diferente.

As Tabelas 1 e 2 apresentam os resultados médios das funções objetivo obtidas pelos algoritmos H-GVILS, GVILS e pelo otimizador CPLEX em 15 e 2 minutos de processamento, respectivamente.

Tabela 1: Resultados 15 min. de processamento

Problema	CPLEX	GVILS	H-GVILS
-teste	15 min	15 min	15 min
PADC01	227,32	227,22	227,16
PADC02	257,30	254,38	253,45
PADC03	164.040	164.089,28	164.076,04
PADC04	164.054	164.187,48	164.172,49

Tabela 2: Resultados 2 min. de processamento

Problema	CPLEX	GVILS	H-GVILS
-teste	2 min	2 min	2 min
PADC01	7.228,95	227,73	227,73
PADC02	7.276,39	256,72	256,72
PADC03	164.043,55	164.099,06	164.099,06
PADC04	164.085,84	164.245,03	164.245,03

Como pode ser observado pela Tabela 1, o algoritmo H-GVILS é superior a GVILS em todos os problemas-teste. Comparando-o com o CPLEX, verifica-se que ele o superou nos problemas-teste PADC01 e PADC02 em até 1,49%; no entanto, teve desempenho inferior nos outros dois problemas, mas aquém apenas a 0,07% da solução do CPLEX, no máximo. Já na Tabela 2, em que o foco está no tempo computacional reduzido, no caso de dois minutos de processamento, verifica-se que tanto o GVILS quanto o H-GVILS superam, em muito, as soluções do CPLEX nos dois primeiros problemas-teste. Entretanto, tais algoritmos têm desempenho inferior nos demais problemas, mas ficam com soluções piores em 0,1%, no máximo. Observa-se, também, que o algoritmo H-GVILS obteve os mesmos resultados do GVILS, visto que sua busca híbrida é apenas executada nas últimas perturbações do ILS, o que ocorreu após os dois minutos de execução do método.

6. Conclusões

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões. Foi proposto um algoritmo heurístico híbrido, que combina uma abordagem de programação matemática com os procedimentos heurísticos GRASP, *Variable Neighborhood Descent* e *Iterated Local Search*, para resolvê-lo. A integração proposta foi validada considerando-se que resultados melhores foram obtidos. O algoritmo heurístico também se mostrou adequado para ser usado como ferramenta de apoio à decisão, uma vez que resultados de qualidade são obtidos rapidamente.

Agradecimentos: Os autores agradecem ao CNPq, processo 474831/2007-8, e à FAPEMIG, processo CEX 2991-06.1/07, pelo apoio a este trabalho.

Referências:

- [1] COSTA, F. P. *Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto*. Dissertação (Mestrado em Engenharia Mineral) — PPGEM/UFOP, Ouro Preto, 2005.
- [2] LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. Boston: Kluwer, 2003.
- [3] FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.
- [4] MLADENOVIC, N.; HANSEN, P. A variable neighborhood search. *Computers and Operations Research*, v. 24, p. 1097–1100, 1997.
- [5] COELHO, I. M.; RIBAS, S.; SOUZA, M. J. F. Um algoritmo baseado em grasp, iterated local search para a otimização do planejamento operacional de lavra. In: *Anais do XI Encontro de Modelagem Computacional*. Volta Redonda/RJ: [s.n.], 2008.
- [6] WHITE, J. W.; OLSON, J. P. Computer-based dispatching in mines with concurrent operating objectives. *Mining Engineering*, v. 38, n. 11, p. 1045–1054, 1986.
- [7] RODRIGUES, L. F. *Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto*. Dissertação (Mestrado em Engenharia de Produção) — DEP/UFMG, Belo Horizonte, 2006.
- [8] CHANDA, E. K. C.; DAGDELEN, K. Optimal blending of mine production using goal programming and interactive graphics systems. *Int. Journal of Surface Mining, Reclamation and Environment*, v. 9, p. 203–208, 1995.
- [9] ALVARENGA, G. B. *Despacho ótimo de caminhões numa mineração de ferro utilizando algoritmo genético com processamento paralelo*. Dissertação (Mestrado em Engenharia Mineral) — DEE/UFMG, Belo Horizonte, 1997.
- [10] MERSCHMANN, L. H. C. *Desenvolvimento de um sistema de otimização e simulação para análise de cenários de produção em minas a céu aberto*. Dissertação (Mestrado em Engenharia de Produção) — PEP/COPPE/UFRJ, Rio de Janeiro, 2002.
- [11] COSTA, F. P.; SOUZA, M. J. F.; PINTO, L. R. Um modelo de alocação dinâmica de caminhões. *Rev. Brasil Mineral*, v. 231, p. 26–31, 2004.
- [12] COELHO, I. M.; RIBAS, S.; SOUZA, M. J. F. Um algoritmo baseado em grasp, vnd e iterated local search para a resolução do planejamento operacional de lavra. In: *XV Simpósio de Engenharia de Produção*. Bauru/SP: [s.n.], 2008.
- [13] BRESINA, J. L. Heuristic-biased stochastic sampling. In: *Proceedings of the 13th National Conference on Artificial Intelligence*, AAAI Press. Portland: [s.n.], 1996. p. 271–278.

**PLANEJAMENTO OPERACIONAL DE LAVRA COM ALOCAÇÃO DINÂMICA DE CAMINHÕES:
ABORDAGENS EXATA E HEURÍSTICA**

Francisco César Rodrigues de Araújo
Mestrando–PPGEM/EM/UFOP–Ouro Preto–MG
E-mail: fcesaraujo@yahoo.com.br

Marcone Jamilson Freitas Souza
Prof. Adjunto–PPGEM/EM/UFOP e DECOM/ICEB/UFOP–Ouro Preto–MG
E-mail: marcone@iceb.ufop.br

RESUMO

ESTE TRABALHO TRATA DO PLANEJAMENTO OPERACIONAL DE LAVRA COM ALOCAÇÃO DINÂMICA DE CAMINHÕES. ESTE PROBLEMA CONSISTE EM DETERMINAR O NÚMERO DE VIAGENS QUE CADA CAMINHÃO DEVE FAZER A CADA FRENTE DE LAVRA, BEM COMO DECIDIR EM QUAIS FRENTE DE LAVRA ALOCAR AS CARREGADEIRAS, DE SORTE A ATENDER AS METAS DE PRODUÇÃO E QUALIDADE REQUERIDAS PARA O MINÉRIO A SER PRODUZIDO, FAZENDO O MELHOR APROVEITAMENTO DA FROTA DE VEÍCULOS DISPONÍVEL. DADA SUA COMPLEXIDADE COMBINATÓRIA, O PROBLEMA É ABORDADO POR UM PROCEDIMENTO HEURÍSTICO BASEADO NA METAHEURÍSTICA *ITERATED LOCAL SEARCH*. TAMBÉM É FORMULADO UM MODELO DE PROGRAMAÇÃO MATEMÁTICA, O QUAL SERVE PARA VALIDAR O ALGORITMO HEURÍSTICO PROPOSTO. OS MODELOS PROPOSTOS SÃO TESTADOS POR MEIO DA UTILIZAÇÃO DE DADOS REAIS EM UM SOFTWARE DESENVOLVIDO PARA ESTE FIM.

Palavras-Chave: Planejamento de lavra em minas a céu aberto, Alocação Dinâmica de Caminhões, *Iterated Local Search*, Metaheurísticas.

ABSTRACT

THIS WORK DEALS WITH THE OPERATIONAL MINING PLANNING IN OPEN PIT MINES WITH EQUIPMENT DYNAMIC ALLOCATION. THIS PROBLEM CONSISTS IN DETERMINING THE NUMBER OF TRIPS THAT EACH TRUCK WOULD DO IN EACH MINING SITE AS WELL AS DECIDING IN WHICH MINING SITE TO ALLOCATE THE LOADERS MACHINES, IN ORDER TO ATTEND THE PRODUCTION AND QUALITY GOALS REQUIRED TO THE ORE PRODUCTION, DOING THE BEST UTILIZATION OF THE AVAILABLE VEHICLE FLEET. DUE TO THE COMBINATORIAL COMPLEXITY OF THE PROBLEM, IT IS SOLVED USING A HEURISTIC PROCEDURE BASED ON THE ITERATED LOCAL SEARCH METAHEURISTIC. ALSO IT IS PROPOSED A MODEL OF MATHEMATICAL PROGRAMMING, WHICH SERVES TO VALIDATE THE PROPOSED METAHEURISTIC ALGORITHM. THE PROPOSED MODELS ARE VALIDATED BY THE UTILIZATION OF REAL DATA IN A SOFTWARE DEVELOPED FOR THIS PURPOSE.

Keywords: Operational Mining Planning in open pit mines, Equipment Dynamic Allocation, *Iterated Local Search*, Metaheuristics.

INTRODUÇÃO

O objetivo principal da Pesquisa Operacional (PO) é encontrar a melhor utilização de recursos limitados, procurando determinar a programação otimizada de atividades ou recursos, fornecendo um conjunto de procedimentos e métodos quantitativos para tratar de forma sistêmica problemas que envolvam a utilização de recursos escassos. Dentre as aplicações da PO, destaca-se sua utilização em Planejamento de Lavra a céu aberto em Mineração, em especial, na alocação de máquinas (carregadeiras e/ou escavadeiras) e caminhões em uma frente de lavra.

Estudos como os de Chanda e Dagdelen (1995), Alvarenga (1997), Costa (2005), Pinto e Merschmann (2001), Merschmann (2002), Pinto *et al.* (2003) e Costa *et al.* (2004, 2005), têm utilizado a Pesquisa Operacional com o objetivo de aplicar metodologias para a solução destes tipos de problemas. Tais esforços se concentraram na aplicação de técnicas de otimização (programação linear e métodos heurísticos) que podem, no entanto, ser aprimoradas. Neste sentido, este trabalho tem como objetivo remodelar as metodologias referidas para a solução do problema mencionado, concentrando-se nos métodos heurísticos. Apesar de estas técnicas não garantirem a otimalidade, elas conseguem, em geral, produzir soluções de boa qualidade, são de fácil implementação e baixo custo de desenvolvimento. Dentre as heurísticas, destacam-se as chamadas metaheurísticas, as quais, ao contrário das heurísticas convencionais, têm caráter geral e são providas de mecanismos para escapar de ótimos locais (SOUZA, 2007).

O restante do trabalho está organizado como segue. Na seção a seguir, é descrito o problema abordado. Logo após, é apresentada uma modelagem de programação matemática para o problema e, ainda, uma modelagem heurística. Em seguida, são apresentados, discutidos e analisados os resultados obtidos pela aplicação dos modelos desenvolvidos a um caso real. A última seção conclui o trabalho.

DESCRIÇÃO DO PROBLEMA DE ALOCAÇÃO DINÂMICA DE CAMINHÕES

No Planejamento de Lavra a Céu Aberto em Mineração, em especial, na alocação de máquinas e caminhões em uma frente de lavra, cada frente contém uma determinada quantidade de minério, com características físicas, químicas e econômicas diferenciadas, denominadas parâmetros de controle. Para satisfazer as especificações exigidas pelos clientes, é necessário selecionar as frentes a serem lavradas, bem como os respectivos ritmos de lavra. Para a operação de minério e estéril, a mina conta com uma frota limitada de equipamentos de carga, os quais devem ser alocados às frentes de lavra e operarem em uma faixa de produtividade que torne viável sua utilização (COSTA, 2005). O transporte do material retirado da frente de lavra é realizado por uma frota de caminhões com capacidades de carga diferentes. Esses caminhões são alocados às frentes de lavra dinamicamente, tentando-se evitar a formação de filas, ou seja, o caminhão é alocado a um ponto de carga ou basculamento que proporcione o menor tempo de fila possível. O ritmo de lavra é determinado pelas capacidades de operação dos equipamentos de carga e transporte alocados às diversas frentes.

No sistema de alocação dinâmica, os caminhões não ficam fixos a uma determinada frente de lavra, como no modelo estático. Eles podem ser direcionados a diferentes frentes de lavra compatíveis, aumentando a produtividade da frota e proporcionando, segundo Costa

(2005), um aumento na capacidade de produção da mina ou a redução do número de equipamentos necessários para manter o mesmo nível de produção.

No problema abordado, há dois pontos de descarga: um para minério e outro para estéril. Considera-se também que o transporte do material às frentes e ao britador é feito por caminhões de capacidades diferentes. Sendo a alocação dinâmica, ao descarregar o material, seja no britador (ou pilhas de estoque próximas ao britador) ou na pilha de estéril, o caminhão é direcionado a uma frente, não necessariamente a mesma da viagem anterior. Admite-se, também, que há um conjunto de carregadeiras de diferentes capacidades de carregamento, sendo este conjunto menor que o conjunto das frentes às quais elas serão alocadas. Considera-se, também, o planejamento para uma hora de produção, o qual é aplicado até uma frente exaurir ou ocorrer uma parada de equipamento, situação na qual deve ser feito outro planejamento. Dado o elevado custo de uma carregadeira, normalmente impõe-se uma quantidade mínima a ser extraída para justificar economicamente sua utilização. É desejável também no planejamento de lavra considerar uma taxa de utilização para os caminhões, ou seja, pode-se desejar que cada caminhão utilizado no transporte de minério e estéril tenha uma determinada produtividade. Com a introdução deste requisito, o número de caminhões utilizados no transporte do material poderá ser, naturalmente, menor do que aquele conseguido sem a utilização dessa taxa.

MODELAGEM DO PROBLEMA ABORDADO

Introdução

Para resolver o problema, são apresentados dois modelos: um de programação matemática e outro heurístico. O primeiro modelo servirá para validar o segundo e será aplicado a instâncias de pequeno porte. Para problemas de porte mais elevado, o primeiro modelo, em geral, não é capaz de resolvê-los na otimalidade, em virtude do tempo proibitivo de execução.

Modelo de programação matemática

O modelo proposto de alocação dinâmica de caminhões é adaptado de Costa *et al.* (2004). Especificamente, a equação referente à relação estéril/minério é substituída por uma equação de meta de produção de estéril. Assim são criadas variáveis de desvio relativas ao não atendimento desta meta. Adicionalmente são incluídas restrições relativas às taxas de utilização dos veículos de transporte, bem como uma componente de avaliação do número de veículos usados.

Sejam os seguintes parâmetros de entrada:

- M : Conjunto de frentes de minério;
- E : Conjunto de frentes de estéril;
- F : Conjunto de frentes formado por $M \cup E$;
- S : Conjunto de parâmetros de qualidade analisados no minério;
- C : Conjunto de equipamentos de carga;
- V : Conjunto de equipamentos de transporte;
- Pr : Ritmo de lavra recomendado relativo a minério (t/h);

Pl : Ritmo de lavra mínimo relativo a minério (t/h);
 Pu : Ritmo de lavra máximo relativo a minério (t/h);
 Er : Ritmo de lavra recomendado relativo a estéril (t/h);
 a^- : Penalidade por desvio negativo da produção de minério;
 a^+ : Penalidade por desvio positivo da produção de minério;
 γ^- : Penalidade por desvio negativo da produção de estéril;
 γ^+ : Penalidade por desvio positivo da produção de estéril;
 t_{ij} : Valor do parâmetro j na frente i (%);
 tr_j : Valor recomendado para o parâmetro j na mistura (%);
 tl_j : Valor mínimo admissível para o parâmetro j na mistura (%);
 tu_j : Valor máximo admissível para o parâmetro j na mistura (%);
 β^- : Penalidade por desvio negativo para o parâmetro j na mistura;
 β^+ : Penalidade por desvio positivo para o parâmetro j na mistura;
 Qu_i : Ritmo de lavra máximo para a frente i (t/h);
 Cl_k : Produção mínima do equipamento de carga k (t/h);
 Cu_k : Produção máxima do equipamento de carga k (t/h);
 cap_l : Capacidade do caminhão l (t);
 T_{il} : Tempo total de ciclo do caminhão l na frente i (min);
 ω_l : Peso associado à importância da utilização do caminhão l ;
 $TxMax_l$: Taxa máxima de utilização admitida para o caminhão l ;
 g_{lk} : parâmetro que assume o valor 1 se o caminhão l é compatível com a carregadeira k e 0, caso contrário;

e as seguintes variáveis de decisão:

x_i : Ritmo de lavra da frente i (t/h);
 y_{ik} : variável que assume o valor 1 se a carregadeira k opera na frente i e 0, caso contrário;
 n_{il} : Número de viagens que um caminhão l realiza na frente i em uma hora;
 d_j^- : Desvio negativo do parâmetro j na mistura (t/h);
 d_j^+ : Desvio positivo do parâmetro j na mistura (t/h);
 P^- : Desvio negativo do ritmo de lavra de minério em relação ao recomendado (t/h);
 P^+ : Desvio positivo do ritmo de lavra de minério em relação ao recomendado (t/h);
 Pe^- : Desvio negativo do ritmo de lavra de estéril em relação ao recomendado (t/h);
 Pe^+ : Desvio positivo do ritmo de lavra de estéril em relação ao recomendado (t/h);
 $usou_l$: Variável binária que assume valor 1 se o caminhão l é usado e 0, caso contrário.

A seguir, é apresentado pelas equações (1)-(24) o modelo de programação matemática relativo à alocação dinâmica de uma frota heterogênea de caminhões e equipamentos de carga, levando-se em consideração metas de produção e qualidade de minério, bem redução no número de caminhões usados.

$$\min \sum_{j \in S} \alpha_j^- d_j^- + \sum_{j \in S} \alpha_j^+ d_j^+ + \beta^- P^- + \beta^+ P^+ + \gamma^- Pe^- + \gamma^+ Pe^+ + \sum_{l \in V} \omega_l usou_l \quad (1)$$

s.a:

$$\sum_{i \in M} (t_{ij} - tu_j) x_i \leq 0 \quad \forall j \in S \quad (2)$$

$$\sum_{i \in M} (t_{ij} - tl_j) x_i \geq 0 \quad \forall j \in S \quad (3)$$

$$\sum_{i \in M} (t_{ij} - tr_j) x_i + d_j^- - d_j^+ = 0 \quad \forall j \in S \quad (4)$$

$$\sum_{i \in M} x_i \leq Pu \quad (5)$$

$$\sum_{i \in M} x_i \geq Pl \quad (6)$$

$$\sum_{i \in M} x_i + P^- - P^+ = Pr \quad (7)$$

$$\sum_{i \in E} x_i + Pe^- - Pe^+ = Er \quad (8)$$

$$x_i \leq Qu_i \quad \forall i \in F \quad (9)$$

$$x_i \geq 0 \quad \forall i \in F \quad (10)$$

$$d_j^+, d_j^- \geq 0 \quad \forall j \in S \quad (11)$$

$$P^+, P^- \geq 0 \quad (12)$$

$$Pe^+, Pe^- \geq 0 \quad (13)$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad (14)$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad (15)$$

$$y_{ijk} \in \{0,1\} \quad \forall i \in F, \forall k \in C \quad (16)$$

$$x_i - \sum_{k \in C} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad (17)$$

$$x_i - \sum_{k \in C} Cl_k y_{ik} \geq 0 \quad \forall i \in F \quad (18)$$

$$n_{il} T_{il} - 60 \sum_{k \in C, g_{ik} \neq 0} y_{ik} \leq 0 \quad \forall i \in F, \forall l \in C \quad (19)$$

$$\frac{1}{60} \sum_{i \in F} n_{il} T_{il} \leq TxMax_l \quad \forall l \in V \quad (20)$$

$$x_i - \sum_{l \in V} n_{il} cap_l = 0 \quad \forall i \in F \quad (21)$$

$$usou_l \geq \frac{1}{60} \sum_{i \in F} n_{il} T_{il} \quad \forall l \in V \quad (22)$$

$$n_{il} \in Z^+ \quad \forall i \in F, \forall l \in V \quad (23)$$

$$usou_l \in \{0,1\} \quad \forall l \in V \quad (24)$$

O modelo de mistura de minérios com metas é formado pelas restrições (2)-(13) e pela função objetivo (1). As restrições (7) e (8) dizem respeito, respectivamente, aos atendimentos das metas de produção de minério e estéril. As demais restrições que complementam o modelo podem ser divididas em dois grupos. O primeiro diz respeito à alocação de equipamentos de carga e a faixa de produtividade que torne viável a utilização desses equipamentos. O segundo grupo de restrições está relacionado ao transporte de material na mina e a alocação e utilização dos caminhões.

Em relação ao primeiro grupo de restrições, o conjunto (14) define que cada frente possui um único equipamento de carga, enquanto que as restrições (15) definem que cada equipamento de carga opera em uma única frente. As restrições (16) definem se um equipamento de carga deve ou não ser alocado a uma determinada frente de lavra. As restrições (17) e (18) limitam, respectivamente, o ritmo de lavra máximo e mínimo, definido

pela carregadeira alocada à frente. Já as restrições (9) limitam o ritmo de lavra máximo definido pelo usuário.

No segundo grupo de restrições, que está relacionado ao transporte de material na mina e a alocação e utilização dos caminhões, cada restrição (19) faz com que um caminhão somente realize viagens à uma frente onde esteja alocado um equipamento de carga compatível. As restrições (20) definem que cada caminhão opere no máximo $TxMax_i\%$ em uma hora. As restrições (21) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados à frente. As restrições (22), juntamente com a função objetivo, forçam com que os caminhões usados sejam penalizados. As restrições (23) determinam que o número de viagens que um caminhão faz à uma frente é um valor inteiro positivo. As restrições (24) indicam que as variáveis $usou_i$ são binárias.

Modelo Heurístico

Iterated Local Search (ILS)

Iterated Local Search (ILS) é uma metaheurística de busca local e como tal, baseado na noção de vizinhança. A exploração do espaço de soluções é feita por meio de movimentos, os quais consistem em modificações feitas na solução corrente, guiando a soluções vizinhas.

O método *Iterated Local Search* (ILS) é baseado na idéia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local (GLOVER e KOCHENBERGER, 2003). A perturbação precisa ser suficientemente forte para permitir que a busca local explore diferentes soluções, mas também fraca o suficiente para evitar um reinício aleatório. Para aplicar um algoritmo ILS, quatro componentes têm que ser especificadas: (a) Procedimento GeraSolucaoInicial(), que gera uma solução inicial s' para o problema; (b) Procedimento BuscaLocal, que retorna uma solução melhorada s'' ; (c) Procedimento Perturbacao, que modifica a solução corrente s guiando a uma solução intermediária s' e (d) Procedimento CritérioAceitação, que decide de qual solução a próxima perturbação será aplicada. Na adaptação proposta para o ILS, o método parte de uma solução inicial gerada conforme descrito mais adiante, em subseção específica. A seguir, é feita uma busca local, de acordo com a subseção seguinte. A partir do ótimo local encontrado, são feitas, em cada iteração, perturbações nessa solução ótima local, tal como especificado mais à frente, seguida de nova aplicação de busca local. Em seguida, verifica-se se houve melhora no valor da função de avaliação (critério de aceitação). Em havendo, guarda-se esta solução como a melhor até o presente momento e volta-se ao menor nível de perturbação. Não havendo, a solução não é armazenada e, neste caso, aplica-se a perturbação no nível corrente à solução atual até que não haja melhora no valor da função de avaliação (no caso, 10% do produto do número de frentes pelo número de caminhões). Decorrido esse número de iterações sem melhora, o nível de perturbação é incrementado. As perturbações são dispostas de forma que as de menores índices são as mais fracas no sentido de complexidade computacional. O procedimento pára quando um tempo limite de processamento é alcançado ou quando o número de iterações sem melhora atingir um valor limiar.

Busca Local

A Busca Local é feita pelo Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto originalmente em *Mladenovic e Hansen (1997)*. Trata-se de um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

Cada vizinhança da solução corrente é explorada por meio de movimentos que serão descritos a seguir. Em cada vizinhança, caminha-se de um vizinho para outro, parando-se ao encontrar um ótimo local.

Movimentos e Estruturas de Vizinhança

Para explorar o espaço de soluções do problema são utilizados os seguintes movimentos, cada qual definindo uma dada vizinhança, aplicáveis tanto para as frentes de minério quanto para as de estéril: (1) Retirar uma viagem de um caminhão em uma frente qualquer; (2) Retirar duas viagens de um caminhão em uma frente qualquer; (3) Realocar uma viagem a uma frente diferente, mantendo o caminhão; (4) Realocar uma viagem a um caminhão diferente, mantendo a frente; (5) Inserir uma viagem de um caminhão a uma frente; (6) Retirar todas as viagens de um caminhão em todas as frentes; (7) Desativar uma carregadeira; (8) Realocar uma carregadeira.

Dados de Entrada

Sejam: F o conjunto de frentes, V o conjunto de veículos, C o conjunto de equipamentos de carga, T o conjunto dos parâmetros de controle (teores de Fe , Al_2O_3 e etc e granulometria) presentes nas frentes, q_i a quantidade máxima de massa lavrável em cada frente i , tc_{il} o tempo de ciclo de cada veículo l a cada frente i , t_{ij} o percentual do parâmetro de controle j presente na frente i , $Compatib_{|C| \times |V|}$ a matriz que representa a compatibilidade entre os equipamentos de carga k e os caminhões l , cap_l a capacidade de cada caminhão l , cl_k a capacidade mínima de cada carregadeira k e cu_k a capacidade máxima de cada carregadeira k e tuc_l a taxa de utilização mínima desejável para o l -ésimo caminhão.

Representação de uma solução

Uma solução é representada por uma matriz $R_{|F| \times (1+V)}$ formada pelas matrizes $Y_{|F| \times 1}$ e $N_{|F| \times |V|}$, em que:

y_{ik} : representa a alocação do equipamento de carga k à frente i , observando-se o estado ativo ou inativo de cada equipamento;

n_{il} : representa o número de viagens do caminhão l à frente i .

$$R_{|F| \times (1+V)} = [Y_{|F| \times 1} \mid N_{|F| \times |V|}] \quad (25)$$

A partir de Y , N e os tempos de ciclo da matriz $Tc_{|F| \times |V|}$ são determinados $S_{|F| \times 1}$ e $Tcv_{1 \times |V|}$, os quais representam, respectivamente, a quantidade de massa lavrada em cada frente e o somatório dos tempos de ciclo de cada veículo.

Geração de uma solução inicial

Para se gerar a solução inicial é necessário se construir as matrizes $Y_{|F| \times 1}$, $N_{|F| \times |V|}$, $S_{|F| \times 1}$ e $Tcv_{1 \times |V|}$. Para se gerar a matriz $Y_{|F| \times 1}$, inicialmente, atribui-se o valor -1 para todas as carregadeiras, ou seja, considera-se que não existe carregadeira alocada a nenhuma frente. Atribui-se, também, o valor 0 (zero) para o *status* de cada carregadeira informando a inatividade de cada uma delas em cada frente. Em seguida, as carregadeiras são alocadas aleatoriamente a cada uma das frentes disponíveis. Para estas carregadeiras, atribui-se o *status* ativo (1), inicialmente.

O número de viagens em cada célula da matriz $N_{|F| \times |V|}$ é gerado aleatoriamente. Para cada célula n_{il} é verificado inicialmente o número máximo (nv_max) de viagens, por hora, em função do tempo de ciclo (Tc) de cada caminhão l às diversas frentes i . É gerado, então, de forma aleatória, um número compreendido entre 0 e o número máximo de viagens (nv_max).

Posteriormente, é verificado se existe carregadeira alocada a essa frente. Se existir, verifica-se se há compatibilidade entre a carregadeira k e o caminhão alocado l . Caso afirmativo, verifica-se se a carregadeira k alocada à frente i encontra-se ativa ou não. Estando ativa, considera-se o número aleatório calculado anteriormente como sendo o número de viagens do caminhão l à frente i . O procedimento é repetido para todas as células da matriz $N_{|F| \times |V|}$.

A quantidade de massa lavrável (s_i) em cada frente é gerada da seguinte forma: para cada frente i , é gerado s_i como sendo o somatório dos produtos das matrizes $N_{|F| \times |V|}$ e $Capcam_{|V|}$, ou seja, entre o número de viagens (n_{il}) e a capacidade de cada caminhão (Cap_l).

Para cada caminhão, é calculado também o somatório do produto entre o número de viagens (n_{il}) e o tempo de ciclo Tc_{il} nas diversas frentes, gerando a matriz $Tcv_{1 \times |V|}$ (somatório dos tempos de ciclo por caminhão).

Mesmo com a geração de todas essas matrizes, é conveniente verificar se existem inviabilidades na solução inicial gerada. A correção das mesmas propicia uma solução inicial de melhor qualidade diminuindo o esforço computacional exigido no refinamento dessa solução. O procedimento verifica a existência de duas inviabilidades.

A primeira diz respeito ao somatório dos tempos de ciclo por caminhão. O tempo de ciclo acumulado de cada caminhão não poderá exceder a 60 minutos. Como o objetivo é fornecer uma solução em uma hora, há a necessidade de se fazer uma correção na solução. Para cada caminhão l , é verificado se o tempo de ciclo acumulado (tcv_l) é superior a 60 minutos. Caso seja, é gerada uma frente i , aleatoriamente, e retirada uma viagem do caminhão l , caso exista. Em seguida, subtrai-se o tempo de ciclo ($tc_{i,l}$) correspondente. O procedimento é repetido, para cada caminhão, até que o tempo acumulado seja inferior ou igual a 60 minutos. Após aplicar este reparador de inviabilidade, é necessário reconstruir a matriz $S_{|F| \times 1}$ (massa lavrável em cada frente).

A segunda inviabilidade diz respeito à quantidade de massa lavrável, por frente. Mesmo com a correção da primeira inviabilidade, não há garantias de que a massa lavrável calculada seja inferior à quantidade máxima lavrável (Q_u) existente em cada frente. Sendo superior, há a necessidade de se fazer a correção. Essa correção também deverá ser realizada caso a quantidade de massa lavrável calculada seja maior que capacidade máxima de produção da carregadeira em uma hora. Portanto, para cada frente, é verificado se a quantidade de massa lavrável calculada inicialmente é superior à quantidade máxima (Q_u) existente na frente i e superior à capacidade máxima de produção da carregadeira alocada à frente i , por hora. Se pelo menos uma das duas situações acontecer, é gerado, aleatoriamente, um caminhão l . Caso haja viagens desse caminhão l à frente i , é retirada uma viagem e recalculados o tempo total (tcv) do caminhão l e a quantidade de massa lavrável s_i . É atualizada também a variável que corresponde à somatória de todas as massas lavráveis, seja de minério (t), seja de estéril (est). Esse procedimento é repetido para todas as frentes até que não haja mais nenhuma inviabilidade.

Esse processo de geração de uma solução inicial é repetido *IterConstr* (99) vezes e, em cada vez, aplica-se o método da descida em vizinhança variável (VND), descrito anteriormente (Busca Local), e a melhor solução obtida é utilizada como solução inicial para o ILS. Este procedimento tem a vantagem de produzir uma solução inicial de boa qualidade rapidamente, já que a busca local exige pouco esforço computacional.

Avaliação de uma solução

Uma solução s é avaliada em função da produção, da qualidade do produto final e da alocação dos equipamentos utilizados (carga e transporte), como mostra a equação a seguir:

$$f(s) = f^p(s) + \sum_{j \in S} f_j^q(s) + f^l(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \quad (26)$$

em que:

- $f^p(s)$: função que avalia s quanto a produção;
- $f_j^q(s)$: função que avalia s quanto à qualidade do j -ésimo parâmetro de controle;
- $f_l^u(s)$: função que avalia s quanto ao atendimento da taxa de utilização máxima do l -ésimo caminhão.
- $f^l(s)$: função que avalia s quanto ao número de caminhões utilizados.
- $f_k^c(s)$: função que avalia s quanto à produtividade da carregadeira k .

As duas últimas parcelas da função de avaliação são usadas apenas para guiar a busca no espaço de soluções e são desconsideradas na apresentação dos resultados, de forma a permitir uma comparação com a função objetivo do modelo de programação matemática.

A seguir, é mostrada a avaliação de cada um dos componentes da função $f(i)$:

- **Produção de Minério e Estéril**

A produção de minério e estéril é avaliada pela equação (27).

$$f^p(s) = \beta \times (|Pr - P| + \gamma \times |Pre - Pe|) \quad (27)$$

onde P é a produção de minério (t); Pr é a meta de Produção de Minério (t); Pe é a produção de Estéril (t); Pre é a meta de Produção de Estéril (t); β é a penalidade por não atendimento à meta de minério e γ é a penalidade por não atendimento à meta de estéril.

A produção P é calculada segundo a equação (28), ou seja, é obtida pela multiplicação do número de viagens dos caminhões às frentes, pela capacidade de carga de cada caminhão.

$$P = \sum_{i \in M} \sum_{l \in V} n_{il} cap_l \quad (28)$$

onde M representa o conjunto de frentes de minério; V é o conjunto de equipamentos de transporte; n_{il} indica o número de viagens de um caminhão l à frente i e cap_l é a capacidade do caminhão l (t);

Para o cálculo de Pe , substitui-se o conjunto das frentes de minério M pelo conjunto E das frentes de estéril e aplica-se uma equação similar à (28).

- **Qualidade da Mistura**

A qualidade da mistura é avaliada segundo a equação (29), em que tm_j é o valor percentual encontrado para o parâmetro j (%) na mistura; tr_j é a meta de qualidade para o parâmetro j (%); P indica a produção de minério (t) e α_j^q é a penalidade por não atendimento à meta de qualidade do parâmetro de controle j ;

$$f_j^q(s) = \alpha_j^q \times P \times \frac{|tr_j - tm_j|}{100} \quad \forall j \in S \quad (29)$$

O valor tm_j é calculado através da média ponderada entre t_{ij} e a produção (em uma hora) de cada frente i de minério, conforme mostrado na equação (30).

$$tm_j = \frac{\sum_{i \in M} t_{ij} x_i}{\sum_{i \in M} x_i} \quad \forall j \in S \quad (30)$$

onde o ritmo de lavra em cada frente i é calculado com base na equação a seguir:

$$x_i = \sum_{l \in V} n_{il} cap_l \quad (31)$$

- **Taxa de utilização dos caminhões**

A utilização dos caminhões é avaliada pela equação (32).

$$f_l^u(s) = \mu \times |TxUtil_l - TxMaxUtil_l| \quad \forall l \in V \quad (32)$$

onde μ é a penalidade por se ultrapassar a taxa máxima de utilização dos caminhões; $TxMaxUtil_l$ é a taxa máxima permitida de utilização do caminhão l e $TxUtil_l$ representa a taxa de utilização do caminhão l , dada por:

$$TxUtil_l = \sum_{i \in F} n_{il} tc_{il} \quad \forall l \in V \quad (33)$$

sendo n_{il} o número de viagens realizadas pelo caminhão l à frente i e tc_{il} o tempo de ciclo do caminhão l à frente i .

São utilizados dois valores para o peso μ na expressão (32). Um valor alto (μ_A) é usado caso a quantidade transportada esteja acima da máxima permitida e outro valor, mais baixo (μ_B), caso a quantidade transportada esteja abaixo da máxima permitida.

- **Número de caminhões utilizados**

A avaliação do número de caminhões utilizados é feita com base na equação (34).

$$f^l(s) = \omega \times \sum_{l \in V} \lceil TxUtil_l \rceil \quad (34)$$

onde ω é a penalidade pela utilização de caminhões e $\lceil TxUtil_l \rceil$ é uma função que assume valor igual a 1 se o caminhão l for utilizado;

- **Avaliação da produtividade das carregadeiras**

A produtividade das carregadeiras é avaliada pela equação (35).

$$f_k^c(s) = \begin{cases} \varphi \times (Cu_k - prod_k) & \text{se } Cl_k \leq prod_k \leq Cu_k \\ \lambda \times (Cl_k - prod_k) & \text{se } prod_k < Cl_k \end{cases} \quad \forall k \in C \quad (35)$$

onde φ é a penalidade por não se atingir a produtividade máxima da carregadeira k e λ a penalidade por utilizar uma carregadeira k com uma produtividade inferior à mínima;

Perturbações

Para escapar de ótimos locais, o método *Iterated Local Search* utiliza-se de mecanismos de perturbações, que são modificações realizadas nos ótimos locais encontrados. As perturbações funcionam como um mecanismo de diversificação do algoritmo. As seguintes perturbações são usadas: (1) Retirar uma viagem, (2) Retirar duas viagens, (3) Realocar viagem mantendo o caminhão, (4) Realocar viagem mantendo a frente, (5) Retirar viagem duas vezes, (6) Retirar duas viagens duas vezes, (7) Realocar viagem mantendo o caminhão duas vezes, (8) Realocar viagem mantendo a frente duas vezes, (9) Realocar

carregadeira, (10) Desativar carregadeira, (11) Ativar carregadeira, (12) Trocar carregadeira, (13) Tirar todas as viagens de um caminhão, (14) Inserir uma viagem.

RESULTADOS COMPUTACIONAIS

O algoritmo heurístico ILS proposto foi desenvolvido na linguagem visual Delphi, versão 7 e o modelo de programação matemática foi implementado no modelador e otimizador LINGO, versão 7.0. Ambos foram testados em um Notebook HP, AMD Turion64 Mobile, Technology MK-36, 2,01 GHz, com 1 GB de RAM.

Apresentam-se, a seguir, os resultados do uso dessas metodologias em um problema-teste envolvendo 5 parâmetros de controle, 15 frentes de lavra, sendo 11 de minério e 4 de estéril, com 30 caminhões e 8 carregadeiras disponíveis.

Os dados desse cenário foram submetidos a dez execuções do algoritmo proposto. De forma a comparar o desempenho do ILS com o método de programação matemática executou-se o otimizador LINGO, interrompendo-se a execução em duas situações: (1) decorrido o tempo gasto pelo ILS para gerar a melhor solução em cada cenário e (2) decorrido uma hora de processamento. Os resultados obtidos são apresentados na Tabela 1.

Tabela 1: Comparação ILS × LINGO

Item	Meta	ILS	LINGO	LINGO
%Fe	47,000	47,9825	47,9937	47,9937
%Al ₂ O ₃	0,3200	0,3175	0,3166	0,3166
%P	0,0400	0,0419	0,0419	0,0419
%PPC	2,35	2,9137	2,9197	2,9197
%He	40,00	37,95	37,87	37,87
Fo	-	30.583,83	30.812,70	30.811,74
Carregadeiras	-	6	7	6
%Tx. util. car.	-	87,04	73,35	82,75
Caminhões	-	20	18	17
Viagens	-	92	80	71
%Tx. util. cam.	85	76,98	74,90	72,89
Tempo (seg.)	-	161	161	3600

Como se observa na Tabela 1, o ILS foi capaz de gerar soluções melhores que as do LINGO em vários aspectos. Por exemplo, a solução do ILS é a que mais se aproximou das metas de qualidade. Com relação à função objetivo, observa-se que o ILS em 161 segundos produz uma solução melhor que o LINGO, tanto em 161 segundos, quanto em uma hora de processamento. Com relação à taxa de utilização das carregadeiras, observa-se que o ILS produz soluções com melhor aproveitamento desses equipamentos. Já com relação ao número de veículos e viagens, as soluções LINGO são mais eficientes; entretanto, esses aspectos têm prioridade inferior à da qualidade dos parâmetros de controle.

Em relação aos valores médios, o algoritmo ILS apresentou desvio de 2,13% quando comparado com o valor da função objetivo da melhor solução conhecida para esse problema-teste.

CONCLUSÕES

Este trabalho tratou o problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões. Dada sua dificuldade de resolução na otimalidade, foi desenvolvido um algoritmo heurístico baseado na metaheurística *Iterated Local Search* (ILS). O algoritmo foi testado em um cenário relativo a um problema real de uma mineração de ferro, comparando-se as soluções produzidas pelo ILS com aquelas produzidas por um otimizador aplicado a um modelo de programação matemática desenvolvido. Os resultados obtidos mostraram que as melhores soluções produzidas pelo ILS são de melhor qualidade que aquelas produzidas pelo otimizador LINGO, mesmo considerando as dimensões relativamente modestas do problema-teste tratado. Adicionalmente, verificou-se que as soluções produzidas pelo ILS foram obtidas sem muito esforço computacional, ao contrário daquelas geradas pelo otimizador. Este último demandou tempo proibitivo para gerar soluções próximas àsquelas geradas pelo ILS em poucos minutos, fato que valida a utilização do algoritmo proposto, enquanto ferramenta de apoio à tomada de decisão. Com relação às características das soluções produzidas pelo algoritmo proposto, pode-se verificar que os valores dos parâmetros de controle ficaram bem próximos das metas estabelecidas. Em relação à frota de veículos de transporte e de equipamentos de carga ficou constatado o bom aproveitamento dos mesmos. No caso dos equipamentos de carga conseguiu-se reduzir a quantidade desses, bem como utilizá-los próximos de suas capacidades máximas de produção. Já no caso dos veículos de transporte, houve um melhor aproveitamento desses, indicado pelas taxas de utilização próximas à meta. Esses resultados comprovam, portanto, a eficiência do método heurístico proposto para a otimização do planejamento operacional de lavra. Destaca-se, finalmente, que o algoritmo ILS é flexível, no sentido de que outras restrições podem ser facilmente incorporadas ao algoritmo.

AGRADECIMENTOS

Os autores agradecem à FAPEMIG, processo CEX APQ-2991-6.01/07, e ao CNPq, processo 474831/2007-8, pelo apoio ao desenvolvimento deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

- ALVARENGA, G. B. (1997). *Despacho ótimo de caminhões numa mineração de ferro utilizando algoritmo genético com processamento paralelo*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais.
- CHANDA, E. K. C.; DAGDELEN, K. (1995). Optimal blending of mine production using goal programming and interactive graphics systems, *International Journal of Surface Mining, Reclamation and Enviroment*, v. 9, p. 203-208.
- COSTA, F. P. (2005). *Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, 141 p.
- COSTA, F. P.; SOUZA, M. J. F. e PINTO, L. R. (2004). Um modelo de alocação dinâmica de caminhões, *Revista Brasil Mineral*, v. 231, p. 26-31.

- COSTA, F. P.; SOUZA, M. J. F. e PINTO, L. R. (2005). Um modelo de programação matemática para alocação estática de caminhões visando ao atendimento de metas de produção e qualidade, *Revista da Escola de Minas*, v. 58, p. 77-81.
- GLOVER, F.; KOCHENBERGER, G. (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston.
- MERSCHMANN, L. H. C. (2002). *Desenvolvimento de um Sistema de Otimização e Simulação para Cenários de Produção em Minas a Céu Aberto*, Dissertação de mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.
- MLADENOVIC, N. & HANSEN, P. (1997). A Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097-1100.
- PINTO, L.R.; MERSCHMANN, L. H. C. (2001). Planejamento operacional da lavra de mina usando modelos matemáticos. *Revista Escola de Minas*, v. 54, n. 3, p. 211-214.
- PINTO, L. R.; BIAJOLI, F. L. e MINE, O. M. (2003). *Uso de otimizador em planilhas eletrônicas para auxílio ao planejamento de lavra*. Relatório técnico, Programa de Pós-graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, Ouro Preto, Minas Gerais.
- SOUZA, M. J. F. (2007). Notas de aula da disciplina Inteligência Computacional para Otimização. Departamento de Computação, Universidade Federal de Ouro Preto, disponível em [http://www.decom.ufop.br/prof/marcone/disciplinas/Inteligência Computacional/ Inteligência Computacional.pdf](http://www.decom.ufop.br/prof/marcone/disciplinas/InteligenciaComputacional/InteligenciaComputacional.pdf). Acesso em 30/04/2008.

Francisco César Rodrigues de Araújo
Programa de Pós-Graduação em Engenharia Mineral
Escola de Minas – Universidade Federal de Ouro Preto
Campus Universitário
35.400-000 Ouro Preto (MG) – Brasil
E-mail: fcesaraujo@yahoo.com.br

This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.



Um algoritmo baseado em GRASP, VND e *Iterated Local Search* para a resolução do problema de planejamento operacional de lavra

Igor Machado Coelho (UFOP) igor.machado@yahoo.com.br

Sabir Ribas (UFOP) sabirribas@yahoo.com.br

Marccone Jamilson Freitas Souza (UFOP) marcone@iceb.ufop.br

Resumo: Este trabalho trata do problema de planejamento operacional de lavra em minas a céu aberto, considerando alocação dinâmica de caminhões. O objetivo é determinar o ritmo de lavra em cada frente, seja ela de minério ou de estéril, de forma que as metas requeridas de produção e qualidade sejam satisfeitas. Dada a complexidade combinatória do problema, foi desenvolvido um algoritmo heurístico híbrido, o qual combina características dos procedimentos heurísticos GRASP, Variable Neighborhood Descent e Iterated Local Search. Para validar o algoritmo proposto, os resultados obtidos pelo mesmo foram comparados com aqueles produzidos pelo otimizador CPLEX, aplicado a um modelo de programação matemática encontrado na literatura. Os resultados obtidos mostraram que o algoritmo proposto é competitivo, sendo capaz de gerar soluções de boa qualidade rapidamente, com baixa variabilidade das soluções finais.

Palavras-chave: Planejamento operacional de lavra; Metaheurísticas; GRASP; Variable Neighborhood Descent; Iterated Local Search.

1. Introdução

Este trabalho trata do problema de planejamento operacional de lavra com alocação dinâmica de caminhões. O objetivo é determinar o ritmo de lavra em cada frente, seja ela de minério ou de estéril, de forma que as metas requeridas de produção e qualidade sejam satisfeitas e o número de caminhões necessários ao processo seja reduzido.

Supõe-se a existência de dois pontos de descarga: um para minério e outro para estéril. Apesar de se considerar um único ponto de descarga para o minério, que normalmente acontece no britador, admite-se que a descarga possa ser feita em um ponto próximo a ele. Isto é, considera-se que a distância das frentes ao britador e seus pontos próximos seja a mesma. Este procedimento é normalmente admitido nas minerações para evitar fila na descarga junto ao britador. Ao encontrar fila no britador, o caminhão descarrega seu material em uma pilha em um ponto próximo a ele e, posteriormente, esse material é retomado para descarga no britador.

Considera-se que o transporte do material às frentes e ao britador é feito somente por caminhões. Admite-se que há um conjunto de carregadeiras de diferentes produtividades, sendo este conjunto menor que o conjunto das frentes às quais elas serão alocadas. Dado o elevado custo de uma carregadeira, também se impõe uma quantidade mínima a ser extraída para justificar economicamente sua utilização.

Para cada caminhão, é imposta uma taxa máxima de utilização, para impedir que o mesmo fique em operação todo o tempo. Isto é necessário porque, na prática, há variação nos tempos de ciclo e, além disso, o ser humano não consegue trabalhar 100% de seu tempo. Para exemplificar o que isto representa na produção, uma taxa máxima de utilização de 85% para



um caminhão representa 51 ($0,85 \times 60$) minutos de operação, no máximo, em uma hora. Supondo que o tempo médio de ciclo (TMC) deste caminhão às diversas frentes seja de 7 minutos por viagem, o número de viagens que poderia ser feita 51 minutos seria 7. Assim, um caminhão com 80 t de capacidade poderia transportar até 560 t em uma hora (7×80), caso fosse limitado a essa taxa de utilização.

No sistema de alocação dinâmica de caminhões, os caminhões não ficam fixos a uma determinada frente de lavra. Eles podem ser direcionados a diferentes frentes de lavra, o que faz aumentar a produtividade da frota e proporciona, segundo Costa (2005), um aumento na capacidade de produção da mina ou a redução do número de equipamentos necessários para manter um determinado nível de produção. Nesse sistema é também possível diminuir o tempo de fila, uma vez que o caminhão pode ser alocado a pontos de carga diferentes. Como desvantagens da alocação dinâmica, apontam-se exigência de um maior número de operações e de um sistema de despacho de caminhões computadorizado.

Considera-se, também, o planejamento para uma hora de produção, o qual é aplicado até uma frente exaurir ou ocorrer uma parada de equipamento, seja por quebra ou manutenção, situação na qual deve ser feito outro planejamento.

De acordo com Costa (2005), este problema pertence à classe NP-difícil, uma vez que tem como subproblema, o Problema da Mochila Múltipla, o qual é NP-difícil. Em vista da dificuldade de resolvê-lo na otimalidade, propõe-se um algoritmo heurístico para encontrar soluções sub-ótimas. O algoritmo heurístico é baseado nos procedimentos GRASP (Feo e Resende, 1995), *Variable Neighborhood Descent* (Mladenovic e Hansen, 1997) e *Iterated Local Search* (Lourenço *et al.*, 2003) e representa uma evolução em relação ao de Costa (2005), por explorar mais vizinhanças que este último.

Este trabalho está organizado como segue. Na seção 2 são apresentados os trabalhos correlatos. Na seção 3 é descrito o problema abordado, enquanto na seção 4 são apresentadas duas modelagens para o mesmo, sendo uma baseada em programação matemática e outra, heurística. A seção 5 descreve os cenários utilizados para teste e a seção seguinte apresenta os experimentos computacionais. A seção 7 conclui o trabalho.

2. Trabalhos correlatos

White e Olson (1986) propuseram um algoritmo que é a base para o sistema *DISPATCH*, que vem operando em muitas minas em todo o mundo (RODRIGUES, 2005). O modelo utiliza programação linear e dinâmica. A programação linear é dividida em duas partes. Na primeira parte do modelo, realiza-se a otimização do problema da mistura conseguindo, através disso, a minimização do remanuseio, forçando uma solução próxima das restrições críticas de mistura e satisfazendo a necessidade de alimentação da planta. A segunda parte do modelo otimiza o transporte na mina por meio da minimização da necessidade de transporte, gerando o fluxo ótimo de transporte no caminho. A programação dinâmica usa o fluxo ótimo no caminho obtido na programação linear, uma lista formada pelos caminhões a serem alocados, os tempos de viagem e as distâncias entre os pontos de carga e descarga e é executada a cada necessidade de alocação de caminhão.

Chanda e Dagdelen (1995) apresentaram um modelo de programação linear por metas para resolver um problema de mistura de minérios no planejamento de curto prazo em uma mina de carvão. O objetivo foi maximizar um critério econômico e reduzir a soma dos desvios dos parâmetros individuais de qualidade e da tonelagem em relação aos valores desejados.

Merschmann (2002) desenvolveu um sistema de otimização e simulação para análise de cenário de produção em minas a céu aberto. O sistema, denominado OTISIMIN



(Otimizador e Simulador para Mineração), foi desenvolvido em dois módulos. O primeiro corresponde ao módulo de otimização onde um modelo de programação linear foi construído e resolvido e o segundo a um módulo de simulação que permite ao usuário utilizar os resultados obtidos na resolução do modelo de programação linear como dados de entrada para a simulação. O módulo de otimização foi elaborado com o objetivo de otimizar o processo de mistura de minérios oriundos das várias frentes de lavra de forma a atender as especificações de qualidade impostas pela usina de tratamento e realizar a alocação de equipamentos (caminhões, carregadeiras e/ou escavadeiras) às frentes de lavra, considerando tanto alocação dinâmica quanto estática dos caminhões.

Os modelos de otimização de Merschmann (2002) foram aperfeiçoados por Costa *et al.* (2004, 2005). Nesses trabalhos, foram apresentadas modelagens de programação matemática que reduziram significativamente o número de restrições do modelo. Além disso, os modelos incluíam o atendimento às metas de produção e qualidade por meio da minimização, na função objetivo, dos desvios de atendimento às restrições de meta.

Ezawa e Silva (1995) desenvolveram um sistema de alocação dinâmica de caminhões às frentes de lavra com o objetivo de reduzir a variabilidade dos teores do minério e aumentar a produtividade do sistema de transporte. O sistema utilizava uma heurística para seqüenciar os caminhões de forma a minimizar a variação dos teores. Para validar o modelo, os autores utilizaram simulação e para o modelamento matemático da mina, a teoria dos grafos. Com a implantação do sistema, foi obtido um aumento de 8% na produtividade do sistema de transporte e o gerenciamento passou a ter em mãos dados mais precisos e em tempo real.

Alvarenga (1997) desenvolveu um programa para o despacho ótimo de caminhões em uma mineração de ferro, a céu aberto, com o objetivo de minimizar o tempo de fila da frota de caminhões, aumentar a produtividade desta e melhorar a qualidade do minério lavrado. No trabalho desenvolvido, que é base do sistema SMART MINE, atualmente muito utilizado em várias minas brasileiras, foi aplicada uma técnica estocástica de otimização, o algoritmo genético com processamento paralelo. Basicamente, o problema consiste em indicar qual o ponto de basculamento ou carregamento ideal e a trajetória para o deslocamento, no momento em que há uma situação de escolha a ser realizada. O autor apontou ganhos de 5% a 15% de produtividade, comprovando a validade da proposta.

Costa (2005) abordou problemas de planejamento operacional de lavra em minas a céu aberto. Foram apresentados e modelados problemas relativos à mistura de minérios provenientes de várias frentes de lavra, levando-se em consideração metas de produção e qualidade, restrições operacionais e a alocação dos equipamentos de carga e transporte necessários ao processo. O autor tratou problemas de mistura de minérios com alocação dinâmica e estática de caminhões. Cada problema foi resolvido por duas metodologias, uma baseada em técnicas de programação matemática e outra em técnicas heurísticas. Dada a NP-completude desses problemas, foi utilizada a metaheurística *Variable Neighborhood Search* (VNS), a qual permite escapar de ótimos locais ainda distantes do ótimo global. Para explorar o espaço de soluções, o autor utilizou seis tipos de movimentos, os quais estão descritos nas subseções 3.2.2.1 a 3.2.2.6 do presente trabalho. Pelos experimentos realizados, o modelo heurístico foi capaz de gerar soluções de melhor qualidade em um menor tempo de processamento que aquelas obtidas pelo modelo exato.

Rodrigues (2006) fez uma análise comparativa de várias metodologias utilizadas para o despacho de caminhões em minas a céu aberto. As metodologias testadas, baseadas em Programação Linear, Programação Dinâmica e em Heurísticas, são aquelas consideradas as bases de algoritmos utilizados em sistemas de despacho comercializados no Brasil. As



soluções finais geradas por essas metodologias foram simuladas usando-se o ambiente ARENA, com o objetivo de reproduzir o comportamento das operações de lavra. Os resultados obtidos mostraram o desempenho dos algoritmos utilizando as metodologias sob diferentes condições em minas a céu aberto. A autora, porém, afirma que não se pode, através dos resultados obtidos, concluir pela superioridade de uma delas.

Guimarães *et al.* (2007) apresentaram um modelo de simulação computacional para validar resultados obtidos pela aplicação de um modelo de programação matemática na determinação do ritmo de lavra em minas a céu aberto. Os autores modelaram o problema usando o otimizador LINGO, versão 7.0, e validaram a solução usando o ARENA, versão 7.0. Foi concluído que, ao contrário do procedimento comumente adotado nas empresas mineradoras, o cumprimento da meta de produção não pode ser atingido simplesmente aumentando-se o número de veículos no sistema produtivo. Esta conclusão foi obtida porque a partir de um determinado número de veículos no sistema, a produção não era alcançada e até pelo contrário, diminuída, devido ao aumento no tempo de fila.

3. Métodos de análise

3.1 Programação matemática

O modelo proposto de alocação dinâmica de caminhões é uma adaptação daquele de Costa *et al.* (2004). Especificamente, a equação referente à relação estéril/minério é substituída por uma equação de meta de produção de estéril. Esta substituição é feita porque pode ser conveniente estabelecer limites inferiores ou superiores para a produção de estéril que não sigam a mesma proporção da meta de minério. Além disso, é mais lógico trabalhar com as metas de estéril, do que com proporção entre a meta de estéril e a de minério. Assim, são criadas variáveis de desvio relativas ao não atendimento desta meta. Adicionalmente, tal como em Guimarães *et al.* (2007), são incluídas restrições relativas às taxas de utilização dos caminhões, bem como uma componente de avaliação do número de veículos usados.

O planejamento da produção é feito para uma hora, sendo replicado enquanto não houver frente exaurida e as condições operacionais forem as mesmas. O objetivo do modelo é minimizar os desvios das metas de produção e qualidade, bem como reduzir o número de veículos necessários à operação. Para sua apresentação, sejam os seguintes parâmetros:

M : Conjunto de frentes de minério;

E : Conjunto de frentes de estéril;

F : Conjunto de frentes formado por $M \cup E$;

T : Conjunto de parâmetros de controle analisados no minério;

C : Conjunto de equipamentos de carga;

V : Conjunto de equipamentos de transporte;

Pr : Ritmo de lavra recomendado relativo a minério (t/h);

Pl : Ritmo de lavra mínimo relativo a minério (t/h);

Pu : Ritmo de lavra máximo relativo a minério (t/h);

Er : Ritmo de lavra recomendado relativo a estéril (t/h);

α^- : Penalidade por desvio negativo da produção de minério;

α^+ : Penalidade por desvio positivo da produção de minério;

γ^- : Penalidade por desvio negativo da produção de estéril;

γ^+ : Penalidade por desvio positivo da produção de estéril;

t_{ij} : Valor do parâmetro j na frente i (%);

tr_j : Valor recomendado para o parâmetro de controle j na mistura (%);

tl_j : Valor mínimo admissível para o parâmetro de controle j na mistura (%);



tu_j : Valor máximo admissível para o parâmetro de controle j na mistura (%);

β_j^- : Penalidade por desvio negativo para o parâmetro de controle j na mistura;

β_j^+ : Penalidade por desvio positivo para o parâmetro de controle j na mistura;

Qu_i : Ritmo de lavra máximo para a frente i (t/h);

Cl_k : Produção mínima do equipamento de carga k (t/h);

Cu_k : Produção máxima do equipamento de carga k (t/h);

cap_l : Capacidade do caminhão l (t);

tc_{il} : Tempo total de ciclo do caminhão l na frente i (min);

ω_l : Peso associado à utilização do caminhão l ;

$TxMax_l$: Taxa máxima de utilização admitida para o caminhão l ;

g_{lk} : Parâmetro que assume valor 1 se o caminhão l é compatível com a carregadeira k e 0, caso contrário.

e as seguintes variáveis de decisão:

x_i : Ritmo de lavra da frente i (t/h);

y_{ik} : Variável que assume valor 1 se a carregadeira k opera na frente i e 0, caso contrário.

n_{il} : Número de viagens que um caminhão l realiza à frente i em uma hora;

d_j^- : Desvio negativo do parâmetro de controle j na mistura (t/h);

d_j^+ : Desvio positivo do parâmetro de controle j na mistura (t/h);

P^- : Desvio negativo do ritmo de lavra de minério em relação ao recomendado (t/h);

P^+ : Desvio positivo do ritmo de lavra de minério em relação ao recomendado (t/h).

Pe^- : Desvio negativo do ritmo de lavra de estéril em relação ao recomendado (t/h);

Pe^+ : Desvio positivo do ritmo de lavra de estéril em relação ao recomendado (t/h).

$usou_l$: Variável binária que assume valor 1 se o caminhão l é usado e 0, caso contrário.

A seguir, é apresentado pelas equações (1)-(24) o modelo de programação matemática relativo à alocação dinâmica de uma frota heterogênea de caminhões e equipamentos de carga, tendo-se como objetivo alcançar as metas de produção e qualidade de minério e reduzir o número de caminhões necessários.

$$\min \sum_{j \in T} \beta_j^- d_j^- + \sum_{j \in T} \beta_j^+ d_j^+ + \alpha^- P^- + \alpha^+ P^+ + \gamma^- Pe^- + \gamma^+ Pe^+ + \sum_{l \in V} \omega_l usou_l \quad (1)$$

$$\sum_{i \in M} (t_{ij} - tu_j) x_i \leq 0 \quad \forall j \in T \quad (2)$$

$$\sum_{i \in M} (t_{ij} - tl_j) x_i \geq 0 \quad \forall j \in T \quad (3)$$

$$\sum_{i \in M} (t_{ij} - tr_j) x_i + d_j^- - d_j^+ = 0 \quad \forall j \in T \quad (4)$$

$$\sum_{i \in M} x_i \leq Pu \quad (5)$$

$$\sum_{i \in M} x_i \geq Pl \quad (6)$$

$$\sum_{i \in M} x_i + P^- - P^+ = Pr \quad (7)$$

$$\sum_{i \in E} x_i + Pe^- - Pe^+ = Er \quad (8)$$

$$x_i \leq Qu_i \quad \forall i \in F \quad (9)$$

$$x_i \geq 0 \quad \forall i \in F \quad (10)$$



$$d_j^+, d_j^- \geq 0 \quad \forall j \in T \quad (11)$$

$$p^+, p^- \geq 0 \quad (12)$$

$$Pe^+, Pe^- \geq 0 \quad (13)$$

$$\sum_{k \in C} y_{ik} \leq 1 \quad \forall i \in F \quad (14)$$

$$\sum_{i \in F} y_{ik} \leq 1 \quad \forall k \in C \quad (15)$$

$$y_{ik} \in \{0,1\} \quad \forall i \in F, k \in C \quad (16)$$

$$x_i - \sum_{k \in C} Cu_k y_{ik} \leq 0 \quad \forall i \in F \quad (17)$$

$$x_i - \sum_{k \in C} Cl_k y_{ik} \geq 0 \quad \forall i \in F \quad (18)$$

$$n_{il} tc_{il} - 60 \sum_{k \in C, g_{ik} \neq 0} y_{ik} \leq 0 \quad \forall i \in F, l \in V \quad (19)$$

$$\frac{1}{60} \sum_{i \in F} n_{il} tc_{il} \leq TxMax_l \quad \forall l \in V \quad (20)$$

$$x_i - \sum_{l \in V} n_{il} cap_l = 0 \quad \forall i \in F \quad (21)$$

$$usou_l \geq \frac{1}{60} \sum_{i \in F} n_{il} tc_{il} \quad \forall l \in V \quad (22)$$

$$n_{il} \in \mathbb{Z}^+ \quad \forall i \in F, l \in V \quad (23)$$

$$usou_l \in \{0,1\} \quad l \in V \quad (24)$$

O modelo de mistura de minérios com metas é formado pelas restrições (2)-(13) e pela função objetivo (1). As restrições (7) e (8) dizem respeito, respectivamente, aos atendimentos das metas de produção de minério e estéril. As demais restrições que complementam o modelo podem ser divididas em dois grupos. O primeiro diz respeito à alocação de equipamentos de carga e a faixa de produtividade que torne viável a utilização desses equipamentos. O segundo grupo de restrições está relacionado ao transporte de material na mina e a alocação e utilização dos caminhões.

Em relação ao primeiro grupo de restrições, o conjunto (14) define que em cada frente pode ser alocado, no máximo, um único equipamento de carga, enquanto que as restrições (15) definem que cada equipamento de carga pode operar, no máximo, em uma única frente. As restrições (16) definem que as variáveis y_{ik} são binárias. As restrições (17) e (18) limitam, respectivamente, o ritmo de lavra máximo e mínimo, definido pela carregadeira alocada à frente. Já as restrições (9) limitam o ritmo de lavra máximo definido pelo usuário.

No segundo grupo de restrições, que está relacionado ao transporte de material na mina e a alocação e utilização dos caminhões, cada restrição (19) faz com que um caminhão somente realize viagens a uma frente onde esteja alocado um equipamento de carga compatível. As restrições (20) definem que cada caminhão opere no máximo $TxMax_l\%$ em uma hora. As restrições (21) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados à frente. As restrições (22), juntamente com a função objetivo, forçam com que os caminhões usados sejam penalizados. As restrições (23) forçam que seja inteiro positivo o número de viagens que um caminhão faz a uma frente. As restrições (24) indicam que as variáveis $usou_l$ são binárias.

3.2 Metaheurística

3.2.1 Representação de uma solução

Uma solução é representada por uma matriz $R_{|F| \times (1+V)}$ formada pelas sub-matrizes $Y_{|F| \times 1}$ e $N_{|F| \times |V|}$, tal como indicado na equação (25).

$$R_{|F| \times (1+V)} = [Y_{|F| \times 1} \mid N_{|F| \times |V|}] \quad (25)$$

Cada célula da matriz $Y_{|F| \times 1}$ representa a carregadeira k alocada à frente i . O valor -1 significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente i , a carregadeira k associada a tal frente é considerada *inativa* e não é penalizada por produção abaixo da mínima para este equipamento de carga (restrição (18) do modelo matemático).

Na matriz $N_{|F| \times |V|}$, cada célula n_{il} representa o número de viagens do caminhão $l \in V$ a uma frente $i \in F$. O valor 0 (zero) significa que não há viagem para aquele caminhão. O valor -1 informa a incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

A partir de Y , N e os tempos de ciclo da matriz $T_{C|F| \times |V|}$ são determinados $X_{|F| \times 1}$ e $Tcv_{1 \times |V|}$, os quais representam, respectivamente, a quantidade de massa lavrada em cada frente e o somatório dos tempos de ciclo de cada caminhão.

3.2.2 Vizinhaça

Para explorar o espaço de soluções do problema foram desenvolvidos 8 movimentos, apresentados a seguir, sendo os seis primeiros propostos em Costa (2005).

3.2.2.1 Movimento Número de Viagens - $N^{NV}(s)$

Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i , onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

3.2.2.2 Movimento Carga - $N^{CG}(s)$

Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, trocar os equipamentos de carga que operam nas frentes i e k , caso as duas frentes possuam equipamentos de carga alocados. No caso de apenas uma das frentes possuir equipamento de carga e a outra estiver disponível, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas juntamente com as frentes escolhidas.

3.2.2.3 Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$

Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl} . Assim, neste movimento, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

3.2.2.4 Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$

Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik} . Portanto, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i . Restrições de compatibilidade entre equipamentos são respeitadas neste movimento, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

**3.2.2.5 Movimento Operação Frente - $N^{OF}(s)$**

Consiste em retirar de operação o equipamento de carga que esteja em operação na frente i . O movimento retira todas as viagens feitas a esta frente, deixando o equipamento *inativo*. O equipamento retorna à operação assim que uma nova viagem é associada a ele.

3.2.2.6 Movimento Operação Caminhão - $N^{OC}(s)$

Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, significando retirar de atividade um caminhão l que esteja operando em uma frente i .

3.2.2.7 Movimento Troca de Viagens - $N^{VT}(s)$

Duas células da matriz N são selecionadas e uma viagem é realocada entre elas. Tal movimento pode ocorrer entre quaisquer células da matriz N , respeitando-se as restrições de compatibilidade entre equipamentos.

3.2.2.8 Movimento Troca de Carregadeiras - $N^{CT}(s)$

Consiste em trocar duas células distintas y_i e y_k da matriz Y , ou seja, trocar os equipamentos de carga que operam nas frentes i e k . Analogamente ao movimento CG , em um movimento CT os equipamentos de carga das frentes são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

3.2.3 Avaliação de uma solução

Como os movimentos desenvolvidos podem gerar soluções inviáveis, uma solução é avaliada por uma função f , a ser minimizada, composta por duas parcelas. A primeira parcela é a função objetivo propriamente dita (equação (1) do modelo de programação matemática) e a segunda é uma função que penaliza a ocorrência de inviabilidade na solução corrente.

$$f(s) = f^{PM}(s) + f^p(s) + \sum_{j \in T} f_j^q(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \quad (26)$$

em que: $f^{PM}(s)$ é uma função que avalia s quanto ao atendimento às metas de produção e qualidade, bem como número de caminhões utilizados (mesma do modelo de programação matemática); $f^p(s)$ avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril; $f_j^q(s)$ avalia s quanto à inviabilidade em relação ao j -ésimo parâmetro de controle; $f_l^u(s)$ avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l -ésimo caminhão e $f_k^c(s)$, que avalia s quanto ao desrespeito aos limites de produtividade da carregadeira k .

3.2.4 Geração de uma solução inicial

Uma solução inicial para o problema é obtida por um procedimento construtivo parcialmente guloso, semelhante à da fase de construção GRASP (FEO e RESENDE, 1995). A construção é feita em duas etapas. As alocações das carregadeiras e a distribuição das viagens às frentes são feitas, na primeira etapa, às frentes de estéril, e na segunda, às frentes de minério. Esta estratégia é adotada tendo em vista que nas frentes de estéril o importante é atender à produção e não é necessário observar a qualidade.

Nas figuras 1 e 2 são mostrados os pseudocódigos dos procedimentos de construção mencionados. Na classificação dos elementos candidatos a ser inseridos na solução considera-se que para as frentes de estéril, a melhor frente é a que possui a maior massa, a melhor carregadeira é a que oferece a maior produção e o melhor caminhão é o de maior capacidade. Já para as frentes de minério considera-se que a melhor frente é a que possui o menor desvio dos teores em relação às metas, a melhor carregadeira é a que oferece a maior produção e o

melhor caminho é o de menor capacidade.

```
Algoritmo Constrói_Solução_Estéril()
1   Seja s uma solução vazia
2   enquanto (a produção de estéril for menor que a produção recomendada) e (existirem frentes de
    estéril não utilizadas) faça
3     frente_atual ← melhor frente de estéril ainda não utilizada
4     se não há carregadeira na frente frente_atual então
5       Aloque a melhor carregadeira ainda não alocada
6     se todas as carregadeiras já foram alocadas então Retorne s
7     para cada caminho l faça
8       se o caminho l for compatível com a carregadeira da frente frente_atual então
9         enquanto (a produção de estéril for menor que a produção recomendada)
10          e (o caminho l pode fazer mais uma viagem) faça
11            Aloque uma viagem ao caminho l para a frente frente_atual
12    Retorne s;
fim ConstróiEstéril;
```

FIGURA 1 – Construção de uma solução para o carregamento de estéril

```
Algoritmo Constrói_Solução_Minério()
1   enquanto (a produção de minério for menor que a produção recomendada) e (existirem frentes de
    minério não utilizadas) faça
2     Ordene as frentes pelos desvios de meta (ordem crescente)
3     frente_atual ← Escolha uma das frentes de minério que podem ser utilizadas
4     se não há carregadeira na frente frente_atual então
5       Aloque a melhor carregadeira ainda não alocada
6     se todas as carregadeiras já foram alocadas então Retorne s
7     para cada caminho l faça
8       se o caminho l for compatível com a carregadeira da frente frente_atual então
9         enquanto (a produção de minério for menor que a produção recomendada)
10          e (o caminho l pode fazer mais uma viagem) faça
11            Aloque uma viagem ao caminho l para a frente frente_atual
12    Retorne s;
fim ConstróiMinério;
```

FIGURA 2 – Construção de uma solução para o carregamento de minério

A escolha da frente *frente_atual* na segunda etapa da construção (Figura 2) é feita por uma função guia, tal como em Bresina (1996). Primeiramente, todas as frentes candidatas são ordenadas tendo em vista os valores dos desvios das metas de qualidade. À *r*-ésima frente melhor classificada, é associada uma função guia $bias(r) = 1/(2r)$. A seguir, cada frente candidata é escolhida com probabilidade $p(r) = bias(r) / \sum bias(r)$. Nesta estratégia, dá-se maior probabilidade de escolha à frente que mais contribui para minimizar os desvios das metas.

3.2.5 Algoritmo proposto

O algoritmo proposto, denominado GVILS, combina os procedimentos GRASP (FEO e RESENDE, 1995), *Variable Neighborhood Descent* – VND (MLADENOVIC e HANSEN, 1997) e *Iterated Local Search* – ILS (LOURENÇO *et al.*, 2003). Seu pseudocódigo está esquematizado na Figura 3. A construção de uma solução inicial (linhas 1 e 2 da Figura 3) é feita pelos procedimentos descritos na subseção anterior. A busca local é feita pelo procedimento VND usando-se os movimentos descritos na subseção 3.2.2. Estrategicamente, a busca local opera nas vizinhanças em uma ordem pré-definida, começando das que exigem menor esforço computacional para aquelas que exigem maior esforço. Assim, o VND segue a seguinte ordem de exploração: N^{CG} , N^{NV} , N^{VC} , N^{VF} . Os movimentos relativos às vizinhanças N^{CT} , N^{OC} , N^{OF} e N^{VT} não foram utilizados na busca local, mas apenas como perturbação. Esta estratégia foi adotada devido ao fato de estes últimos movimentos requererem maior esforço computacional na busca local.

```
Algoritmo GVILS(s)
1  s ← Constrói_Solução-Estéril();
2  s ← Constrói_Solução_Minério();
3  s ← VND(v');
4  enquanto (Critério de parada não satisfeito) faça
5      s' ← Perturbação(v, nível);
6      s'' ← VND(s');
7      se f(s'') < f(s) então
8          s ← s';
9      fim-se;
10 fim-enquanto;
11 s ← s*;
12 Retorne s;
fim GVILS;
```

FIGURA 3 – Algoritmo GVILS proposto

O objetivo da perturbação é diversificar a busca, gerando uma solução diferente e cada vez mais “distante” da região atual de exploração no espaço de busca. Para cumprir esta missão, são estabelecidos vários níveis de perturbação. Para cada nível n , são aplicados à solução corrente $n+2$ movimentos, escolhidos aleatoriamente dentre os 8 descritos na subseção 3.2.2. A essa solução perturbada é aplicada busca local, baseada no procedimento VND (linha 6 da Figura 3). Após *IterMax* iterações sem melhora em um dado nível, este é aumentado. No caso de se encontrar uma solução de melhora, o nível de perturbação volta ao seu nível mais baixo.

4. Descrição dos cenários

Os cenários utilizados para teste foram os mesmos de Costa (2005). Tais cenários se referem a dados do planejamento operacional de empresas mineradoras do quadrilátero ferrífero, situado na região central do Estado de Minas Gerais. Os parâmetros de controle são os teores químicos (Fe, SiO₂, Mn, P, H₂O etc) e granulometrias especificadas para o minério. A diferença reside apenas na função de avaliação que, ao contrário deste autor, considera a penalização pela utilização de veículos, bem como na inclusão de uma nova restrição que impede que um caminhão opere mais que uma determinada taxa de utilização, no caso, 85% do tempo.

A Tabela 1 apresenta as principais características dos problemas-teste utilizados.

TABELA 1 – Problemas-teste utilizados.

Problema-teste	Número de frentes de lavra	Número de parâmetros de controle	Número de carregadeiras	Número de caminhões
PADC01	17	10	8	30
PADC01	17	10	8	30
PADC01	32	10	7	30
PADC01	32	10	7	30

Fonte: Costa (2005).

Para os problemas-teste PADC01 e PADC02, os caminhões de 1 a 15 têm capacidade 50 t e compatíveis com as carregadeiras Car01 a Car04; enquanto os demais caminhões têm capacidade 80 t e compatíveis com as carregadeiras Car05 a Car08. Já para os problemas-teste PADC03 e PADC04, todos os caminhões são de 50 t e compatíveis com todas as carregadeiras. Os dois últimos problemas diferem apenas na composição química e granulométrica dos minérios presentes nas frentes de lavra.

O único parâmetro do método, *IterMax*, que indica o número de iterações sem melhora em um dado nível de perturbação, foi fixado em 60, após uma bateria preliminar de testes.

Os pesos adotados na função de avaliação são apresentados na Tabela 2. Tais pesos são os mesmos de Costa (2005).

TABELA 2 – Pesos adotados para os parâmetros de controle.

Pesos	Descrição	Valor
$\alpha^- = \alpha^+ = \gamma^- = \gamma^+$	Penalidade por tonelada abaixo ou acima da meta de produção (estéril/minério)	100
$\beta_j^- = \beta_j^+ \quad \forall j \in T$	Penalidade por tonelada abaixo ou acima da meta de qualidade	1
$\omega_l \quad \forall l \in V$	Penalidade pelo uso de um caminhão	1
$TxMax_l \quad \forall l \in V$	Taxa máxima de utilização de um caminhão	85%

Fonte: Costa (2005).

5. Resultados computacionais e análises

O algoritmo proposto (GVILS) foi desenvolvido em C++ usando o compilador g++ 4.0 e o IDE Eclipse 3.1. O modelo de programação matemático foi modelado em AMPL e resolvido pelo otimizador CPLEX, versão 9.1. Os dois modelos, heurístico e de programação matemático, foram testados em um microcomputador com processador Pentium IV, 3.0 GHZ e 2 GB de RAM, rodando Windows XP Professional.

Cada problema-teste foi executado 10 vezes pelo algoritmo GVILS, considerando-se como critério de parada o tempo de 15 minutos. Já para o otimizador CPLEX foram considerados três critérios de parada: 15, 60 e 120 minutos de processamento.

A Tabela 3 apresenta os resultados da função objetivo obtidos pelo algoritmo proposto e pelo otimizador CPLEX. A coluna “Melhora” indica o quanto o algoritmo GVILS melhorou a solução produzida pelo CPLEX-120 (solução obtida pelo CPLEX em duas horas de processamento). Já a coluna “Desvio” mostra o desvio dos valores médios de GVILS em relação à melhor solução conhecida em cada problema-teste.

TABELA 3 – Comparação de resultados: CPLEX $\times$ GVILS.

Problema-teste	CPLEX-15	CPLEX-60	CPLEX-120	Melhor GVILS	Média GVILS	Melhora (%)	Desvio (%)
PADC01	227,32	227,32	227,32	227,12	227,22	0,09	0,04
PADC02	257,30	254,06	254,06	252,41	254,38	0,65	0,78
PADC03	164.040	164.034	164.034	164.070	164.089	-0,02	0,03
PADC04	164.054	164.054	164.054	164.140	164.187	-0,05	0,08

Como pode ser observado pela Tabela 3, o algoritmo GVILS foi capaz de, em 15 minutos de processamento, gerar soluções de melhor qualidade que o CPLEX em duas horas de processamento nos problemas-teste PADC01 e PADC02. Já nos dois últimos problemas-teste, o algoritmo proposto apresentou soluções piores, mas apenas a um máximo de 0,05% da solução do CPLEX. A variabilidade das soluções finais também é baixa, variando de 0,03% a 0,78%, no máximo.

O algoritmo GVILS não foi comparado com o de Costa (2005) visto que a função custo utilizada por este era diferente e, além disso, o modelo desse autor não contemplava restrições relativas à taxa de utilização de caminhões.

6. Conclusões

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões.

Em virtude da complexidade combinatória do problema, foi proposto um algoritmo



heurístico híbrido, que combina os procedimentos heurísticos GRASP, *Variable Neighborhood Descent* e *Iterated Local Search*, para resolvê-lo.

Usando problemas-teste da literatura, o algoritmo heurístico foi comparado com o otimizador CPLEX aplicado a um modelo de programação matemática. Verificou-se que o algoritmo proposto é competitivo com este, pois foi capaz de encontrar soluções de boa qualidade rapidamente e com baixa variabilidade das soluções finais. Dado que a tomada de decisão no problema em pauta tem que ser rápida, os resultados encontrados validam a utilização do algoritmo proposto, enquanto ferramenta de apoio à decisão.

Agradecimentos

Os autores agradecem ao CNPq, processo 474831/2007-8, e à FAPEMIG, processo CEX 2991-06.1/07, pelo apoio ao desenvolvimento da presente pesquisa.

Referências

ALVARENGA, G. B. *Despacho ótimo de caminhões numa mineração de ferro utilizando algoritmo genético com processamento paralelo*. Dissertação (Mestrado em Engenharia Elétrica), Programa de Pós-Graduação em Engenharia Elétrica, Escola de Engenharia, Universidade Federal de Minas Gerais - UFMG, 1997.

BRESINA, J. L. Heuristic-biased stochastic sampling. In *Proceedings of the 13th National Conference on Artificial Intelligence*, AAAI Press, Portland, p. 271—278, 1996.

CHANDA, E. K. C.; DAGDELEN, K. Optimal blending of mine production using goal programming and interactive graphics systems, *International Journal of Surface Mining, Reclamation and Environment*, v. 9, p. 203-208, 1995.

COSTA, F. P. *Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto*. 141 p. Dissertação (Mestrado em Engenharia Mineral) - Programa de Pós-Graduação em Engenharia Mineral, Escola de Minas, Universidade Federal de Ouro Preto - UFOP, Ouro Preto, 2005.

COSTA, F. P.; SOUZA, M. J. F. e PINTO, L. R. Um modelo de alocação dinâmica de caminhões. *Revista Brasil Mineral*, v. 231, p. 26-31, 2004.

COSTA, F. P.; SOUZA, M. J. F. e PINTO, L. R. Um modelo de programação matemática para alocação estática de caminhões visando ao atendimento de metas de produção e qualidade. *Revista da Escola de Minas*, v. 58, p. 77-81, 2005.

EZAWA, L. e SILVA, K. S. Alocação dinâmica de caminhões visando qualidade. In: *Anais do VI Congresso Brasileiro de Mineração*, p. 15–19, Salvador, Bahia, 1995.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.

GUIMARÃES, I. F.; PANTUZA, G. e SOUZA, M. J. F. Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. In: *Anais do XIV Simpósio de Engenharia de Produção (SIMPEP)*, Bauru, CD-ROM, 11 p., 2004.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated Local Search. In: GLOVER, F. e KOCHENBERGER, G. (Eds). *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, 2003.

MERSCHMANN, L.H.C. *Desenvolvimento de um sistema de otimização e simulação para análise de cenários de produção em minas a céu aberto*. Dissertação (Mestrado em Engenharia de Produção) - Programa de Engenharia de Produção/COPPE, Universidade Federal do Rio de Janeiro - UFRJ, Rio de Janeiro, 2002.

MLADENOVIC, N.; HANSEN, P. A Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097-1100, 1997.

RODRIGUES, L. F. *Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto*. 86 p. Dissertação (Mestrado em Engenharia de Produção), Programa de Pós-Graduação em Engenharia de Produção, Escola de Engenharia, Universidade Federal de Minas Gerais - UFMG, 2006.

WHITE, J. W. e OLSON, J. P. Computer-based dispatching in mines with concurrent operating objectives. *Mining Engineering*, v. 38, n. 11, p. 1045–1054, 1986.

UM ALGORITMO HEURÍSTICO HÍBRIDO PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO INTEGRADA DE VEÍCULOS E TRIPULAÇÕES

Marcone Jamilson Freitas Souza

Departamento de Computação, Programa de Pós-Graduação em Engenharia Mineral
Universidade Federal de Ouro Preto
35.400-000 Ouro Preto (MG)
marcone@iceb.ufop.br

Sabir Ribas, Igor Machado Coelho

Universidade Federal de Ouro Preto
35.400-000 Ouro Preto (MG)
{sibirribas, igor.machado}@yahoo.com.br

RESUMO

Este trabalho trata da programação integrada de veículos e tripulações. Dada a natureza combinatória do problema, a abordagem utilizada é heurística. O algoritmo proposto é baseado nos procedimentos *Iterated Local Search*, *Variable Neighborhood Descent* e Busca Tabu. Para explorar o espaço de busca são usados movimentos envolvendo realocações e trocas de viagens nos veículos e tarefas nas tripulações. Para testar o algoritmo foram utilizados dados reais da cidade de Belo Horizonte, Brasil. Os experimentos computacionais realizados mostraram que o algoritmo proposto foi capaz de produzir resultados superiores àqueles alcançados por um método clássico de solução dessas duas etapas do planejamento de transporte público.

PALAVARAS CHAVE. Programação integrada de veículos e tripulações, Transporte Público, Metaheurísticas.

Área principal: Logística e Transportes (LGT)

ABSTRACT

This work deals with the integrated vehicle and crew scheduling. Due to the combinatorial complexity of the problem, the approach to solve it is heuristic. A hybrid algorithm, based on *Iterated Local Search*, *Variable Neighborhood Descent* and *Tabu Search*, is proposed. In order to explore the solution space, some moves based on reallocation and swap of both, trips of the vehicles schedule and tasks of the duties schedule are used. The algorithm was tested with real data of the city of Belo Horizonte, Brazil. Computational results show that the proposed algorithm was able to produce better solutions than a traditional method applied to solve these two stages of the public transport planning.

KEYWORDS. Integrated Vehicle and Crew Scheduling, Public Transport, Metaheuristics.

Main area: Logistic and Transports

1. Introdução

Dada a complexidade do planejamento do transporte no Sistema de Transporte Público, ele é geralmente dividido em cinco etapas: planejamento das rotas, definição da tabela de horários, programação dos veículos, programação das tripulações e rodízio das tripulações. Normalmente, a saída de uma etapa corresponde à entrada da etapa subsequente. Neste trabalho, trata-se da resolução de duas delas, a programação de veículos e a de tripulações.

Segundo Freling *et al* (2003), existem três diferentes abordagens para resolver o problema de programação de veículos e de tripulações: Seqüencial, Independente e Integrada.

Na abordagem seqüencial os problemas de programação de veículos e tripulações são resolvidos separadamente, sendo a solução de um utilizada como entrada para a resolução do outro. A abordagem seqüencial, por sua vez, é dividida em tradicional e inversa.

A abordagem seqüencial tradicional foi a primeira a ser estudada e é a mais adotada por empresas de transporte. Nesta estratégia, é resolvido primeiramente o problema de programação de veículos (PPV) e, a partir dessa solução, é resolvido o problema de programação das tripulações (PPT). O inconveniente dessa abordagem é que a solução dos veículos guia a programação das tripulações (FRELING *et al.*, 2003), a qual tem o custo mais relevante entre os custos do sistema (BOUZADA, 2003). Uma alternativa para resolver esse problema é aplicar a abordagem tradicional, mas considerando algumas características dos tripulantes na programação dos veículos, deixando outras características para serem tratadas posteriormente, durante a resolução do PPT. Ainda que isto implique em acréscimos no custo da programação de veículos, essa alternativa facilita a resolução do PPT e geralmente produz soluções melhores que as obtidas pela abordagem tradicional (FRELING *et al.*, 2003). Scott (1985) e Darby-Dowman (1988), *apud* Freling *et al.* (2003), e Reis (2007) utilizam esta última estratégia.

Na abordagem seqüencial inversa, o PPT é resolvido primeiro, diretamente da tabela de horários e, a seguir, é resolvido o PPV. Esta estratégia é justificada por dois motivos principais: 1) pelo fato de os custos com as tripulações serem mais elevados que os com os veículos e 2) o conjunto de restrições de veículos é muito menor que o de tripulações, o que sugere resolver o problema mais restrito primeiro, que tem menos alternativas de solução. Bassi *et al* (2007) propõem uma heurística baseada em GRASP (Feo e Resende, 1995) para resolver o problema de programação de tripulações, em que as tripulações são criadas a partir da tabela de horários, independentemente da solução dos veículos. Em um segundo momento, o problema de programação de veículos é resolvido tendo como entrada a solução das tripulações, definida anteriormente. Um mecanismo de Reconexão por Caminhos foi implementado para melhorar a solução. A heurística foi testada para várias empresas de transporte e os resultados comparados com aqueles obtidos pela abordagem tradicional. De acordo com os autores, o procedimento proposto é capaz de produzir soluções com custos menores que os obtidos pelo procedimento seqüencial tradicional.

Na abordagem independente tanto a programação dos veículos quanto a dos tripulantes é feita diretamente a partir da tabela de horários. Geralmente, os resultados obtidos por essa abordagem não são viáveis na prática, pois alguns veículos operam sem um tripulante, e alguns tripulantes operam em veículos inexistentes, que não pertencem à programação dos veículos. Entretanto, essa abordagem permite determinar os limitantes inferiores para a solução ótima do problema de programação de veículos e tripulações.

Na abordagem integrada os dois problemas são resolvidos simultaneamente. Obviamente, a dificuldade de se resolver o problema de programação integrada de veículos e tripulações (PPVT) é maior por que o espaço de soluções é imenso. O PPVT contém todas as combinações possíveis de tripulações para cada uma das possíveis combinações de veículos. A natureza combinatória desse problema é ressaltada em Borndröfer *et al.* (2006) e em Freling *et al.* (2003). Entretanto, com o aumento na velocidade das CPUs, combinado com o progresso dos algoritmos, pesquisadores vêm dando mais ênfase a esse tópico recentemente. São exemplos de trabalhos utilizando esse tipo de abordagem: Freling *et al.* (1995), Haase e Friberg (1999) e Freling *et al.* (1999).

Segundo Freling *et al.* (2003), a abordagem integrada do PPVT sempre permitirá obter resultados melhores ou iguais aos resultados da abordagem seqüencial, e resultados piores ou iguais aos da abordagem independente.

Este trabalho apresenta um novo algoritmo para resolver o PPVT. O algoritmo combina os procedimentos heurísticos *Iterated Local Search*, *Variable Neighborhood Descent* e Busca Tabu com Relaxação Adaptativa. O uso de métodos heurísticos é justificado pela natureza combinatória do PPVT. A utilização das técnicas propostas é justificada tendo em vista os resultados obtidos para cada problema isolado (Vide Souza *et al.* (2007) para o problema de programação de veículos e Marinho *et al.* (2004) para a programação de tripulações).

O restante deste artigo é organizado como segue. Na Seção 2 descreve-se o problema tratado. O algoritmo proposto é apresentado na Seção 3. Os resultados computacionais são apresentados e analisados na Seção 4 enquanto a última seção conclui o artigo.

2. Definição do Problema

De acordo com Freling *et al.* (2003), o problema de programação veículos e tripulações pode ser definido como segue: dada uma tabela de horários de viagens a serem realizadas, encontrar a programação de custo mínimo para os veículos e para as tripulações tal que ambas sejam factíveis e mutuamente compatíveis. Cada viagem possui horários de início e fim e o tempo de viagem entre todos os pontos de ônibus são conhecidos. A programação dos veículos é factível se (1) cada viagem é atribuída a um único veículo, e (2) cada veículo possui uma seqüência factível de viagens, isto é, todo veículo deve ser capaz de executar consecutivamente as viagens a ele atribuídas. Determinadas viagens não pertencem a nenhuma linha e, assim, são denominadas viagens mortas. O tempo de viagem morta é o tempo em que um veículo viaja entre dois locais (terminais ou garagens) sem passageiros. Tal procedimento é necessário para reposicionar um veículo do ponto final de uma viagem ao ponto inicial da próxima viagem. As viagens atribuídas a um veículo definem sua jornada diária de trabalho, sendo essa jornada conhecida como bloco do veículo. Cada bloco é dividido em oportunidades de troca, definidos por um horário e um local, onde e quando pode haver uma troca de tripulação. Uma tarefa é definida por um conjunto de viagens entre duas oportunidades de troca consecutivas e representa a mínima porção de trabalho que pode ser associada a uma tripulação. O conjunto de tripulações deve executar todas as tarefas assim como o de veículos deve executar todas as viagens. As tarefas que são associadas a uma mesma tripulação definem sua jornada de trabalho. A programação das tripulações é factível se (1) cada tarefa é associada a um único tripulante, e (2) a seqüência de tarefas de cada um dos tripulantes pode ser realizada na prática por apenas um tripulante. Em particular, cada tripulante deve satisfazer certos quesitos. Um exemplo típico é a restrição de tempo máximo de trabalho. O custo de um tripulante é uma combinação de custos fixos e custos variáveis. A Seção 3.4 detalha os custos considerados neste trabalho.

As empresas analisadas neste trabalho estimam que em horários de pico, períodos em que há um número maior de passageiros em trânsito, o tempo necessário para a ocorrência de uma troca de tripulações é de dois minutos, e fora desses horários esse tempo é de um minuto. Segundo elas, existem dois horários de pico ao longo do dia: das 5:00 às 7:59 e das 16:00 às 18:59. Em função disso, são considerados dois tipos de jornada de trabalho: jornada simples e jornada com dupla pegada. Este último tipo de jornada existe em função da necessidade de um número maior de tripulações nos horários de pico. Assim, em geral, nas jornadas com pegada dupla, as tripulações trabalham durante o primeiro pico e retornam depois para atender viagens do segundo pico. Para caracterizar uma jornada com dupla pegada entre o final da primeira parte da jornada e o início da subsequente deve haver um intervalo de pelo menos duas horas.

Neste trabalho considera-se que os veículos estão sempre disponíveis ao longo do dia e que há apenas uma garagem.

3. Metodologia Proposta

3.1. Representação de uma solução

Uma solução para o PPVT consiste em um par (s_v, s_T) , em que s_v representa uma

solução para os veículos e s_T representa uma solução para as tripulações.

A solução s_V consiste em um conjunto de veículos, em que a cada um é associada uma lista de viagens a serem feitas ao longo do dia. A solução s_T consiste em um conjunto de jornadas de trabalho para as tripulações, em que a cada qual é associada uma lista de tarefas a serem feitas diariamente.

3.2. Construção de uma solução inicial

Uma solução inicial para o PPVT é feita primeiramente construindo uma solução inicial para os veículos e depois, a partir desta, é gerada uma solução para as tripulações. As duas soluções são construídas por procedimentos gulosos.

Para a construção de uma solução inicial do PPV é aplicado um procedimento que, a cada iteração, insere uma nova viagem a um veículo, de acordo a função de avaliação relativa aos veículos (veja a equação (2) da Seção 3.4). A escolha do veículo que receberá a viagem é totalmente gulosa, o que significa que o veículo que apresentar o menor custo relativo à alocação é escolhido. O procedimento é finalizado quando todas as viagens forem alocadas. O conjunto de viagens alocadas a cada veículo é chamado de bloco do veículo.

A partir do bloco de viagens de cada veículo é construída uma solução para o PPT. Inicialmente, cada veículo tem suas viagens agrupadas em tarefas. A seguir, as tarefas são ordenadas pelo horário de início. A cada passo há sempre uma jornada sem tarefas, além do conjunto de jornadas não vazias. O procedimento funciona como segue. A primeira tarefa é associada à jornada do primeiro tripulante. Depois, para cada tarefa da seqüência, é avaliada a inserção na jornada de cada tripulante do conjunto e mais na jornada do tripulante sem tarefas. A avaliação das jornadas é feita de acordo com a equação (8) da Seção 3.4, relativa à programação das tripulações. A tarefa é associada à jornada que apresentar menor custo de alocação. O procedimento pára quando todas as tarefas tiverem sido alocadas.

3.3. Estruturas de vizinhança

Para explorar o espaço de busca foram definidos seis tipos de movimentos. As vizinhanças N_V^R , N_V^S , N_C^R e N_C^S usam movimentos baseados em realocação (*Reallocation*) e troca (*Swap*), envolvendo viagens de veículos (*Vehicle*) e tarefas de tripulantes (*Crew*), respectivamente. As vizinhanças N_V^{RP} e N_V^{SP} consistem em realocação e troca de viagens em veículos, respectivamente, seguidas de uma reconstrução da programação das tripulações afetadas pelos movimentos de acordo com a seção anterior.

3.4. Função de avaliação

O custo de uma solução para o PPVT é calculado pela equação (1):

$$f(s) = \alpha \times f_V(s) + \beta \times f_T(s) \quad (1)$$

em que $f_V(s)$ representa a componente de f que avalia s quanto à programação dos veículos (equação 2) e $f_T(s)$ avalia s quanto à programação das tripulações (equação 8) e α e β são pesos. As funções $f_V(s)$ e $f_T(s)$ foram propostas em Souza *et al.* (2007) e Marinho *et al.* (2004), respectivamente, e modelaram satisfatoriamente problemas reais de Transporte Público.

$$f_V(s) = \sum_{k \in F} f_V^k(s) + \omega_{TL} \times \text{numTrLinha} + \omega_{EDP} \times \text{excessoDuplaPegV} \quad (2)$$

Na equação (2), $f_V^k(s)$ é o custo do veículo k , calculado pela equação (3); numTrLinha é o número de trocas de linhas realizadas por todos os veículos e ω_{TL} é a penalidade associada; excessoDuplaPegV representa o excesso de vezes em que os veículos, durante suas jornadas, retornam à garagem e permanecem lá por duas horas ou mais e ω_{EDP} é a penalidade para esse quesito. Por restrições operacionais, o número de duplas pegadas na programação dos veículos deve ser inferior a 60% do tamanho da frota.

$$f_V^k(s) = \sum_{(i,j) \in T} C_{ij} + CD_{g1} + CD_{ng} \quad (3)$$

Na equação (3), T é o conjunto das n viagens do veículo k , C_{ij} é o custo para realizar as viagens i e j consecutivas, calculado pela equação (4), CD_{g1} é o custo da viagem do veículo k da

garagem ao ponto de início da primeira viagem e CD_{ng} é o custo da viagem do veículo k do ponto final de sua última viagem à garagem.

$$C_{ij} = \begin{cases} CD_{ij} + CC_{ij} & \text{se } ocioso_{ij} < 0 \\ CT_{ij} + CD_{ij} & \text{se } 0 \leq ocioso_{ij} \leq TempoMaxTerm \\ CD_{ig} + CD_{gj} & \text{se } ocioso_{ij} > TempoMaxTerm \end{cases} \quad (4)$$

Na equação (4), CT_{ij} , CD_{ij} e CC_{ij} representam, respectivamente, o tempo em que o veículo permanece no terminal entre as viagens i e j , o tempo de viagem morta entre i e j , e o tempo em minutos em que a viagem i é conflitante com a viagem j . $TempoMaxTerm$ é o tempo máximo que um veículo pode permanecer no terminal, estabelecido por políticas operacionais do Sistema de Transporte Público e $ocioso_{ij}$ corresponde ao tempo entre o fim da viagem i e o início da viagem j menos o tempo de viagem morta, se for necessário o reposicionamento do veículo.

Nas equações (5) e (6), em que são determinados os custos CD_{ij} e CT_{ij} , tem-se que: tvm_{ij} é o tempo de viagem morta entre as viagens i e j ; $TempoDeslocLim$ é o tempo limite de viagem morta, w_{TD} é o peso se o tempo de viagem morta for maior que $TempoDeslocLim$, w_{TT} é o peso se o tempo de permanência no terminal ultrapassar o $TempoTermLim$, α_1 e β_1 são pesos, no intervalo $[0, 1]$, usados para ajustar as funções custo (5) e (6), respectivamente.

$$CD_{ij} = \begin{cases} (\alpha_1 \times tv_{m_{ij}})^{\beta_1} & \text{se } tv_{m_{ij}} \leq TempoDeslocLim \\ tv_{m_{ij}} \times w_{TD} & \text{se } tv_{m_{ij}} > TempoDeslocLim \end{cases} \quad (5)$$

$$CT_{ij} = \begin{cases} (\alpha_2 \times ocioso_{ij})^{\beta_2} & \text{se } ocioso_{ij} \leq TempoTermLim \\ ocioso_{ij} \times w_{TT} & \text{se } ocioso_{ij} > TempoTermLim \end{cases} \quad (6)$$

O conflito CC_{ij} entre duas viagens i e j é avaliado pela equação (7):

$$CC_{ij} = - (ocioso_{ij}) \times w_{TC} \quad (7)$$

em que w_{TC} é o peso associado a esse quesito.

Seja $R = \{r_1, r_2, \dots, r_{|R|}\}$ o conjunto de linhas a cargo da empresa e $S_1, S_2, \dots, S_{|S|}$ subconjuntos de R , chamados grupos de linhas, com $R = S_1 \cup \dots \cup S_{|S|}$ e $S_1 \cap S_2 \cap \dots \cap S_{|S|} = \emptyset$.

A avaliação das tripulações é feita pela equação (8) a seguir:

$$\begin{aligned} f_T(s) = & \sum_{i \in Trip} (penalidadeTempoOcioso_i + penalidadeHoraExtra_i + \\ & + \mu_3 \times numTrVPerm_i + \mu_4 \times numTrLPerm_i + \mu_5 \times numTrPPerm_i + \\ & + \lambda_1 \times tempoSobreP_i + \lambda_2 \times tempoExcTrab_i + \lambda_3 \times tempoInsufEJorn_i + \\ & + \lambda_4 \times numTrocaVPr oib + \lambda_5 \times numTrLPr oib_i + \lambda_6 \times numTrPPr oib_i) + \\ & + \lambda_7 \times excessoDuplaPeg + \delta \times numTripulantes \end{aligned} \quad (8)$$

em que:

- (i) $Trip$ é o conjunto de tripulantes da componente s_T ;
- (ii) μ_i e λ_i são as penalidades aplicadas às restrições e objetivos, respectivamente;
- (iii) $penalidadeTempoOcioso_i$ assume o valor μ_1 se ociosidade na jornada i for maior que duas horas e o valor $0.347 \times TempoOcioso_i^{0.8}$ caso contrário, onde $TempoOcioso_i$ corresponde ao tempo em que o tripulante fica ocioso ao executar a jornada i ;
- (iv) $penalidadeHoraExtra_i$ assume μ_2 se o tempo de hora extra for maior que duas horas, caso contrário assume valor $1.357 \times HoraExtra_i^{0.6}$, onde $HoraExtra_i$ indica o tempo, em minutos, em que o tripulante que executa a jornada i faz hora extra;
- (v) $numTrVeiculosPerm_i$ é o número de vezes em que ocorre troca de veículos permitida durante a jornada i . Uma troca de veículos é permitida quando há tempo para se efetuar a

- troca de tripulações;
- (vi) $numTrLPerm_i$ é o número de vezes em o tripulante associado à jornada i na rota r_j , é realocado a uma outra rota r_k , com as duas pertencentes ao mesmo grupo, isto é, $r_j \in S_m \subset R$ e $r_k \in S_m$;
 - (vii) $numTrPPerm_i$ indica o número de vezes em que um tripulante é realocado a um ponto (terminal) diferente durante a jornada i . Uma troca de terminal é permitida se: (1) os terminais envolvidos na troca pertencerem ao mesmo grupo terminais; (2) as tarefas pertencerem ao mesmo veículo; (3) houver mais que duas horas de ociosidade entre duas tarefas consecutivas da jornada i , isto é, em uma pegada dupla, a realocação de um tripulante entre diferentes terminais é permitida. Uma troca de ponto (terminal) é proibida caso não satisfaça a nenhum dos três quesitos anteriores;
 - (viii) $tempoSobrep_i$ é a quantidade de tempo, em minutos, em que duas tarefas são realizadas simultaneamente na jornada i , isto é, em duas tarefas consecutivas quando a segunda inicia antes do termino da primeira;
 - (ix) $tempoExcTrab_i$ representa o tempo, em minutos, da jornada i que ultrapassa nove horas de trabalho. O tempo trabalhado em cada jornada i é calculado como: $TempoTrab_i = (\text{horário de término da ultima tarefa da jornada } i) - (\text{horário inicial da primeira tarefa}) - (\text{duração do intervalo entre as tarefas que caracterizam pegada dupla, se a jornada possuir pegada dupla})$;
 - (x) $tempoInsufEJorn_i$ é a quantidade de tempo, em minutos, que falta para que a diferença entre o horário final da jornada i em um dia e o horário inicial no dia subsequente atinja o tempo mínimo entre jornadas de 11 (onze) horas. Esse valor é calculado como: $tempoInsufEJorn_i = \max \{0, 11 \times 60 - (1440 - \text{horário de término da jornada } i + \text{horário inicial da jornada } i)\}$;
 - (xi) $numTrVProib_i$ é o número de vezes em que ocorreu troca proibida de veículos durante a jornada i . Uma troca de veículos é proibida quando não há tempo para se efetuar a troca;
 - (xii) $numTrLProib_i$ é o número de vezes em que ocorreu troca proibida de linhas durante a jornada i . Uma troca de linha é proibida se as linhas envolvidas pertencerem a grupos diferentes, isto é, se dadas duas linhas r_j e r_k tivermos $r_j \in S_m \subset R$ e $r_k \notin S_m$;
 - (xiii) $numTrPProib_i$ é o número de vezes em que ocorreu troca proibida de pontos (terminais) durante a jornada i (veja o item (vii));
 - (xiv) $excessoDuplaPeg$ representa o número de jornadas classificadas como de pegada dupla que excede $1,25 \times$ (tamanho da frota de veículos). Esse valor é uma estimativa da empresa, que assume que no máximo 50% dos tripulantes podem fazer pegadas duplas;
 - (xv) $numTripulantes$ é o número de tripulantes escalados para executar o conjunto de jornadas e δ é a penalização correspondente.

Nos itens (iii) e (iv) são usadas funções cujos valores estão no intervalo $[0, 16]$ e $[0, 24]$, respectivamente. Nestas funções, de domínio $[0, 120]$, quanto maior for o tempo de ociosidade ou hora-extra, mais acentuada é a penalidade. Elas procuram, também, levar em consideração o fato de que a hora-extra é 50% mais cara que a hora de trabalho normal. Maiores detalhes sobre a relação dos pesos utilizados podem ser vistos em Marinho (2005).

3.5. Uma nova heurística integrada de refinamento

Foram concebidos dois procedimentos de busca local para refinar soluções do PPVT. O primeiro, chamado de Descida Randômica de Viagens em Veículos com Propagação (DRVVP), tem por objetivo melhorar estritamente a componente s_v , relativa à programação dos veículos, mas mantendo-se coerente a componente s_t , relativa à programação das tripulações. Para isso, exploram-se as vizinhanças Realocação de Viagens de Veículos (N_V^R) e Realocação Propagada de Viagens de Veículos (N_V^{RP}). Nessa versão obtém-se, a cada iteração, um vizinho qualquer da solução corrente por meio de um movimento m de acordo com a vizinhança N_V^R . Caso o vizinho v'_1 gerado proporcione melhora no valor da função de avaliação, a nova solução corrente será o vizinho v'_2 obtido por meio de um movimento que possua as mesmas características de m .

(veículos e viagens; tripulantes e tarefas), mas agora, de acordo com a vizinhança N_V^{RP} ; caso contrário, um novo vizinho é gerado. Esse procedimento se repete até que se exceda *DescidaMax* iterações sem melhora na solução. Para verificar se um vizinho proporciona melhora, esse método considera apenas o custo referente à programação dos veículos.

O segundo procedimento é a Descida Randômica de Viagens em Veículos Integrada (DRVVI). O objetivo, nesse caso, é refinar uma solução s utilizando-se vizinhanças que explorem a integração dos dois problemas, no caso, N_V^{RP} , que faz a realocação propagada de viagens em veículos e N_V^{SP} , que faz a troca propagada de viagens em veículos. Essas são estruturas de vizinhança que exigem mais processamento que as demais e, por isso, a navegação no espaço de busca é geralmente mais lenta. As duas estruturas consistem em uma alteração simples, realocação ou troca, na programação de veículos e em uma adição gulosa de tarefas à programação de tripulações. A adição gulosa é a parte mais custosa do processo, por isso deve restringir-se a momentos propícios. No caso, apenas para as soluções pouco “distantes” da solução corrente, com respeito ao custo da programação dos veículos, são aplicados os movimentos integrados. Mais especificamente, nesta busca local gera-se, a cada iteração, um vizinho qualquer da solução corrente por meio de um movimento m de acordo com a vizinhança integrada escolhida. Se a estrutura escolhida for a N_V^{RP} , o movimento m será obtido pela vizinhança N_V^R , já se a estrutura escolhida for a N_V^{SP} , o movimento m será obtido pela vizinhança N_V^S . Caso o vizinho v'_1 gerado possua custo $f_V(v'_1)$ inferior ao valor obtido pela multiplicação de uma constante ρ ($\rho > 1$) pelo custo $f_V(v)$ dos veículos na solução corrente v , um novo vizinho v'_2 é gerado por meio de um movimento m' pertencente à vizinhança escolhida e que utiliza os mesmos veículos e viagens do movimento m . Se o custo global $f(v'_2)$, relativo a veículos e tripulações do vizinho v'_2 , for inferior ao custo global da solução corrente $f(v)$, a nova solução corrente será v'_2 . Assim como no método DRVVP, esse procedimento é repetido até que se exceda *DescidaMax* iterações sem melhora na solução global.

3.6. O algoritmo ILS_VND_BTRA

Descreve-se, a seguir, o algoritmo ILS-VND-BTRA para resolver o PPVT. Este algoritmo combina características dos procedimentos *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e Busca Tabu com Relaxação Adaptativa (BTRA).

Para fundamentá-lo, seja N^k ($k=1, \dots, k_{max}$) um conjunto finito de estruturas de vizinhança pré-selecionadas, e $N^k(s)$ o conjunto de soluções da k -ésima vizinhança de s . Chama-se uma solução s' de ótimo local com respeito a $N^k(s)$, se não existe uma solução $s \in N^k(s')$ tal que $f(s) < f(s')$. O procedimento heurístico VND (Hansen e Mladenovic, 2003) é baseado em três princípios básicos: (1) um ótimo local para uma vizinhança pode não necessariamente ser para outra; (2) um ótimo global é um ótimo local para todas as possíveis vizinhanças; e (3) em vários problemas, ótimos locais de uma vizinhança são relativamente próximos dos ótimos de outras.

O procedimento VND_2N, proposto para fazer uma busca local no espaço de soluções do PPVT, está descrito na Figura 1. Ele inspira-se na idéia da heurística VND no sentido de que não somente as vizinhanças são alteradas, mas também os procedimentos de busca local usados. No passo 1 aplica-se a uma solução s_0 o procedimento DRVVP utilizando a vizinhança N_V^{RP} resultando uma solução s_1 . Como DRVVP é um procedimento de descida, a solução s_1 ou é melhor que a solução s_0 ou é a própria solução s_0 . Assim, a solução s_1 é atribuída à solução s_0 . O algoritmo continua com a chamada ao procedimento DRVVP usando a vizinhança N_V^{SP} , gerando a solução s_2 . Se o custo da solução s_2 for inferior ao custo de s_0 , então se inicia novamente a execução do passo 1. O Passo 1 termina quando os procedimentos DRVVP não produzirem melhoria nas soluções. As execuções dos Passos 2 e 3 são semelhantes à do Passo 1, sendo que no Passo 2 o refinamento de uma solução é feito por meio dos procedimentos BTRA_R e BTRA_S e, no Passo 3, o refinamento é feito pelo procedimento DRVVI usando a vizinhança N_V^{RP} e depois a N_V^{SP} . Os procedimentos BTRA_R e BTRA_S são baseados na metaheurística Busca Tabu (Glover, 1997) com oscilação estratégica (ou Relaxação Adaptativa) usando as vizinhanças N_C^R e N_C^S , respectivamente. Estes procedimentos consistem em mudar periodicamente o valor dos pesos durante a busca, promovendo a geração de inviabilidades para

explorar regiões ainda não visitadas do espaço de soluções. A descrição deste procedimento, especializado para resolver o PPT, está descrito em Marinho *et al.* (2004).

Proc.	VND_2N
Início	Seja s_0 uma solução inicial
Passo 1	$s_1 \leftarrow \text{DRVVP}(s_0, N_V^{RP})$ $s_2 \leftarrow \text{DRVVP}(s_1, N_V^{SP})$ Se s_2 for melhor que s_0 , então faça $s_0 \leftarrow s_2$ e volte ao início do Passo 1
Passo 2	$s_3 \leftarrow \text{BTRA\_R}(s_2)$ $s_4 \leftarrow \text{BTRA\_S}(s_3)$ Se s_4 for melhor que s_2 , então faça $s_2 \leftarrow s_4$ e volte ao início do Passo 2
Passo 3	$s_5 \leftarrow \text{DRVVI}(s_4, N_V^{RP})$ $s_6 \leftarrow \text{DRVVI}(s_5, N_V^{SP})$ Se s_6 for melhor que s_4 , faça $s_4 \leftarrow s_6$ e volte ao início do Passo 3 Se houve alguma melhora no Passo 3, volte ao Passo 2
Passo 4	Retorne s_6

Figura 1 – Procedimento VND_2N aplicado ao PPVT

Iterated Local Search – ILS (Lourenço *et al.* 2003) é uma metaheurística com quatro componentes básicas: “Geração de solução inicial”, “Busca Local”, “Perturbação” e “Critério de aceitação”. O primeiro módulo tem por objetivo gerar uma boa solução inicial para o problema. “Busca local” é o módulo que retorna um ótimo local para a busca. Já “Perturbação” consiste em realizar modificações nos ótimos locais s de forma a gerar uma nova solução s' próxima de s . O módulo “Critério de aceitação” consiste em determinar se o resultado s'' da aplicação da busca local à solução s' vai ser aceita ou não como nova solução corrente. Neste trabalho, considera-se que a solução s'' é aceita, e escreve-se $s \leftarrow s''$, se ela for melhor que o ótimo local anterior, isto é, se $f(s'') < f(s)$. As perturbações no PPVT consistem em realizar movimentos aleatórios na vizinhança N_V^{RP} . O número de movimentos aleatórios é mensurado pela variável *nível*, de forma que quanto maior esse número, maior é a “distância” da solução perturbada ao ótimo local corrente. Uma perturbação de nível i consiste em realizar $i + 2$ movimentos aleatórios. O número de modificações no ótimo local, isto é, o nível de perturbação, é aumentado sempre que as buscas locais não forem bem sucedidas em *ILSmax* vezes. O algoritmo é encerrado quando um número máximo de modificações, dado por *NivelMax*, ou um tempo limite é alcançado. Na Figura 2 apresenta-se o pseudocódigo do algoritmo ILS-VND-BTRA proposto para resolver o PPVT.

Algoritmo ILS-VND-BTRA

```

1:  Entrada:  $s, f(\cdot), \text{NivelMax}, \text{ILSmax}, \text{TempoLimite}$ 
2:   $s^* \leftarrow s; \quad \text{nível} \leftarrow 0;$ 
3:  enquanto  $\text{nível} < \text{NivelMax}$  e  $\text{tempo\_de\_processamento} < \text{TempoLimite}$  faça
4:     $\text{iter} \leftarrow 0;$ 
5:    enquanto  $\text{iter} < \text{ILSmax}$  e  $\text{tempo\_de\_processamento} < \text{TempoLimite}$  faça
6:       $s \leftarrow \text{Perturbação}(s, \text{nível});$ 
7:       $s \leftarrow \text{VND\_2N}(s);$ 
8:      se  $f(s) < f(s^*)$  então
9:         $s^* \leftarrow s; \quad \text{nível} \leftarrow 0; \quad \text{iter} \leftarrow 0;$ 
10:     senão
11:        $s \leftarrow s^*; \quad \text{iter} \leftarrow \text{iter} + 1;$ 
12:     fim se
13:   fim enquanto
14:    $\text{nível} \leftarrow \text{nível} + 1;$ 
15: fim enquanto
16: retorne  $s^*$ 

```

Figura 2 – Algoritmo ILS-VND-BTRA aplicado ao PPVT

4. Resultados computacionais

O algoritmo descrito foi implementado na linguagem C++ usando o ambiente de desenvolvimento e compilador Borland C++ Builder 6.0 e testado em um microcomputador com processador Intel® Pentium®, 3.00 GHz, com 1 GB de memória RAM e sistema operacional Windows XP Professional.

Para validar o algoritmo foram disponibilizadas pela BHTRANS, empresa responsável pelo gerenciamento do Sistema de Transporte Público da cidade de Belo Horizonte, problemas-teste reais pertencentes a uma Bacia de Linhas de tal cidade.

Os parâmetros do método ILS_VND_BTRA foram empiricamente ajustados em uma bateria preliminar de testes. O número de iterações sem melhora do ILS (*ILSmax*) foi fixado em 50, enquanto *BTmax* (número de iterações sem melhora no método BTRA) foi fixado em 50. O valor escolhido para *NivelMax* foi 5 e o número de iterações sem melhora (*DescidaMax*) nas chamadas dos procedimentos DRVVP e DRVVI foi dado por *número de viagens* $\times$ *tamanho máximo da frota de veículos*. O valor do parâmetro ρ do procedimento DRVVI foi fixado em 1,2. O número máximo de duplas pegadas na programação dos veículos foi fixado em 60% do tamanho da frota referente à solução produzida manualmente pelas empresas analisadas. Já o número máximo de duplas pegadas na programação das tripulações foi fixado em 1,25 vezes o tamanho da frota. Esta última relação foi adotada considerando-se que, na prática, a cada veículo estão associados 2,5 tripulantes e que não é desejável se ter mais que 50% das tripulações fazendo dupla pegada.

A primeira fase da abordagem seqüencial tradicional usa o algoritmo ILS descrito em Souza *et al.* (2007) e foi interrompido com 15 minutos de processamento. Já a segunda fase usa o algoritmo BTRA descrito em Marinho *et al.* (2004). O tempo total de processamento foi limitado em meia hora. Portanto, na metodologia seqüencial tradicional foi destinado 15 minutos para resolver o problema de programação de veículos e 15 minutos para resolver o problema de programação de tripulações.

Nas tabelas 1 e 2 são apresentados os valores dos parâmetros utilizados para o cálculo da função de avaliação apresentada na seção 5.3. Na Tabela 2 os valores *TempoDeslocLim* e *TempoTermLim* são vinculados a cada problema-teste e representam os maiores tempos de viagem morta e terminal, nessa ordem, em minutos, encontrados nas soluções utilizadas por duas empresas do Sistema. Tais parâmetros são os mesmos utilizados pelos algoritmos que resolvem o PPV e o PPT de forma seqüencial. Os pesos $\alpha = 1$ e $\beta = 2,5$ relativos à programação de veículos e tripulações, respectivamente, foram utilizados com o objetivo de se dar uma importância maior à programação das tripulações.

Tabela 1 - Parâmetros utilizados na função de avaliação do PPVT

Parâmetro	Valor	Parâmetro	Valor
<i>TempoMaxTerm</i>	119 minutos	μ_1	18
α_1	2	μ_2	40
β_1	$\ln(16) / \ln(\alpha_1 \times \text{TempoDeslocLim})$	μ_3	19
α_2	1	μ_4	5
β_2	$\ln(16) / \ln(\alpha_2 \times \text{TempoTermLim})$	μ_5	5
ω_{TD}	50	λ_1	40
ω_{TT}	40	λ_2	40
ω_{TC}	80	λ_3	40
ω_{EDP}	80	λ_4	45
ω_{TL}	5	λ_5	35
δ	100	λ_6	35
α	1	λ_7	45
β	2.5		

A Tabela 3 contém os resultados encontrados, relativos à função de avaliação, para as

duas metodologias de solução do PPVT em 10 execuções para cada problema teste. Pode-se observar que há redução tanto nos melhores valores quanto nos valores médios da função de avaliação em todos os casos analisados. O desvio do algoritmo proposto é pequeno, provando sua robustez, pois há pouca variabilidade nos valores das soluções finais. As melhores soluções produzidas pelo algoritmo proposto têm valores da função de avaliação até 6,43% melhores que aqueles relativos à metodologia seqüencial tradicional.

Tabela 2 – Problemas-teste e seus respectivos tempos limites relativos ao PPV

Probl.-teste	TempoDeslocLim	TempoTermLim	Probl.-teste	TempoDeslocLim	TempoTermLim
G02_DOM	18	73	G27_DOM	23	110
G02_SEG	18	118	G27_SEG	23	47
G02_SEX	18	113	G27_SEX	23	42
G02_SAB	18	62	G27_SAB	23	57

Tabela 3 - Comparação de resultados (Seqüencial Tradicional × Integrada)

Problema-teste	Método Seqüencial Tradicional			Método Integrado			
	Melhor Valor	Valor Médio	Desvio	Melhor Valor	Valor Médio	Desvio	Melhora
G02_DOM	7036	7160,35	6,47%	6725,5	6841,5	1,72%	4,41%
G02_SEG	22507,5	22828,55	8,26%	21087,5	21297,7	1,00%	6,31%
G02_SEX	23045,5	23354,1	7,50%	21724,5	21910,1	0,85%	5,73%
G02_SAB	12938,5	13260,5	8,51%	12220	12468,05	2,03%	5,55%
G27_DOM	3497,5	3550,75	4,51%	3397,5	3442,3	1,32%	2,86%
G27_SEG	8299	8463,25	7,30%	7887,5	7992,55	1,33%	4,96%
G27_SEX	8429,5	8486,25	7,59%	7887,5	8015,55	1,62%	6,43%
G27_SAB	5193,5	5346,65	5,73%	5057	5176,35	2,36%	2,63%

As tabelas 4 e 5 comparam, para cada metodologia, algumas características das melhores soluções encontradas em quatro dias da semana das empresas G02 e G27.

Tabela 4 – Características das soluções de G02: Método Seq. Tradicional × Método Integrado

Empresa G02	Método Seqüencial Tradicional				Método Integrado			
	DOM	SEG	SEX	SAB	DOM	SEG	SEX	SAB
Número de viagens	90	260	260	172	90	260	260	172
Número de veículos	11	40	40	23	11	40	39	23
Tempo de terminal (hh:mm)	16:13	22:26	5:56	21:02	13:44	8:12	22:14	21:02
Tempo de viagem morta (hh:mm)	8:24	12:50	12:14	16:48	9:50	15:06	14:58	18:36
Número de tripulantes	21	62	63	37	19	62	63	34
Tempo ocioso (hh:mm)	15:13	15:04	9:23	15:44	11:07	7:52	6:28	15:12
Tempo de hora-extra (hh:mm)	5:04	6:44	7:34	20:08	14:58	8:28	12:31	19:44
Trocas de linhas	1	5	7	1	4	4	6	2
Trocas de pontos	2	18	19	9	3	11	8	10
Trocas de veículos	6	62	66	11	2	34	34	15
f_v	841	2510	2553	1491	893	2740	2887	1585
f_T	2478	7999	8197	4579	2333	7339	7535	4254
$f = f_v + 2.5 \times f_T$	7036	22507.5	23045.5	12938.5	6725.5	21087.5	21724.5	12220

Apesar de a função de avaliação dos veículos não visar a redução do número de veículos, pelo menos diretamente, percebe-se pelas Tabelas 4 e 5 que o método integrado produz soluções com número igual ou menor de veículos que o método tradicional. Comparando outras características das soluções produzidas, observam-se as seguintes vantagens do método integrado sobre o método seqüencial tradicional: redução no número de tripulações (de 21 para 19 aos domingos e de 37 para 34 aos sábados na empresa G02, de 24 para 23 às segundas e de 25 para 23 às sextas na empresa G27), redução no tempo de ociosidade das tripulações, redução no tempo de espera em terminal (à exceção das sextas-feiras nas duas empresas). O aumento no número de horas-extras é recompensado pela melhoria em alguma outra característica da solução como a

diminuição do número de tripulações ou do número de trocas de veículos. Por exemplo, na empresa G02 há às sextas um aumento na quantidade de horas-extras de quase 5 horas; no entanto, há uma redução significativa no número de trocas de veículos (de 66 para 34). Já na empresa G27, há às sextas um aumento de mais de 7 horas-extras; no entanto, este aumento é acompanhado de uma redução de duas tripulações, bem como de 7 trocas de veículos (o que representa mais da metade do número de trocas de veículos da solução produzida pelo método tradicional). A despeito do aumento no tempo de espera em terminal às sextas-feiras nas duas empresas, valem as mesmas observações anteriores, acrescidas ao fato de que, para a empresa G02, este aumento foi acompanhado da redução de um veículo.

Tabela 5 – Características das soluções de G27: Método Seq. Tradicional × Método Integrado

Empresa G27	Método Seq. Tradicional				Método Integrado			
	DOM	SEG	SEX	SAB	DOM	SEG	SEX	SAB
Número de viagens	52	98	98	69	52	98	98	69
Número de veículos	5	14	13	7	5	14	13	7
Tempo de terminal (hh:mm)	12:57	10:25	8:45	11:12	10:47	10:22	11:08	10:54
Tempo de viagem morta (hh:mm)	3:50	16:06	15:20	8:26	4:36	16:52	15:20	9:12
Número de tripulantes	10	24	25	15	10	23	23	14
Tempo ocioso (hh:mm)	6:30	10:52	13:44	10:14	5:06	10:06	8:57	8:54
Tempo de hora-extra (hh:mm)	6:53	5:58	6:29	7:33	6:35	11:38	14:42	13:19
Trocas de linhas	0	0	0	0	0	0	0	0
Trocas de pontos	0	3	2	1	0	3	3	0
Trocas de veículos	4	14	12	4	2	6	5	3
f_v	370	1019	952	661	385	1050	1005	682
f_T	1251	2912	2991	1813	1205	2735	2753	1750
$f = f_v + 2.5 \times f_T$	3497.5	8299	8429.5	5193.5	3397.5	7887.5	7887.5	5057

Pelas tabelas 4 e 5 verifica-se, também, que o algoritmo proposto foi capaz de produzir soluções melhores, sob o ponto de vista da função de avaliação, para todos os dias considerados. Observa-se que, quanto à integração, há uma piora no custo dos veículos e uma redução significativa no custo dos tripulantes, produzindo assim soluções melhores para o conjunto. A aplicação da metodologia desenvolvida, com os pesos adotados neste trabalho, é vantajosa em situações nas quais, para a empresa, os custos envolvidos são maiores com as tripulações.

5. Conclusões

Este artigo apresentou um algoritmo, baseado nas heurísticas *Iterated Local Search*, *Variable Neighborhood Descent* e Busca Tabu com Relaxação Adaptativa, para resolver o problema de programação integrada de veículos e tripulantes. Os resultados obtidos pelo algoritmo proposto foram comparados com aqueles obtidos por um método tradicional seqüencial, que resolve primeiramente o problema de programação de veículos e depois o problema de programação de tripulantes. Para testá-lo foram utilizados dados reais de empresas do Sistema de Transporte Público da cidade de Belo Horizonte (MG).

Os experimentos computacionais realizados mostraram que o algoritmo proposto foi capaz de produzir resultados superiores àqueles alcançados pela clássica metodologia seqüencial. Foram produzidas soluções piores para os veículos, mas melhores para as tripulações, gerando, no entanto, soluções melhores para o conjunto. A melhora global proporcionada pelo algoritmo foi de até 6,43%, o que mostra que é promissor explorar a interação desses problemas. Adicionalmente, a variabilidade das soluções finais foi relativamente baixa com relação ao valor da função de avaliação, no caso, inferior a 2,36%. Como trabalho futuro, faz-se necessário fazer uma análise de sensibilidade do algoritmo com respeito à adoção de diferentes pesos para a função de avaliação.

Agradecimentos

Os autores agradecem à FAPEMIG (processo TEC 679/06), ao CNPq (processo 474831/2007-8) e à BHTRANS pelo apoio ao desenvolvimento deste trabalho.

Referências

- Bassi, H. V., Silva, G. P., Marinho, E. H., Souza, M. J. F.** (2007), GRASP and Path-relinking Applied to the Independent Crew Scheduling Problem, In: International Conference on Computers in Urban Planning and Urban Management, 2007, Foz do Iguaçu, Reviewed papers in the Conference Proceedings of CUPUM 2007, v.1, p.1 – 11.
- Borndröfer, R.; Grötschel, M. and Pfetsch, M.** (2006), Public transport to the fORe!, *ORMS Today*, v. 33(2), p. 30-40.
- Bouzada, C. F.** (2003), *Custo do Transporte Coletivo por Ônibus*, Editora C/Arte, Belo Horizonte.
- Darby-Dowman, K., Jachnik, J. K.; Lewis, R. L. e Mitra, G.** (1988), Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience, in J. R. Daduna e A. Wren (eds.), *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop*, Springer Verlag, Berlin, p. 226-239.
- Feo, T.A., Resende, M.G.C.** (1995), Greedy randomized adaptive search procedures, *Journal of Global Optimization*, v. 6, p. 109-133.
- Freling, R. C.; Boender, G. E. e Paixão, J. M. P.** (1995), An integrated approach to vehicle and crew scheduling, Technical Report 9503/A, Econometric Institute, Erasmus University Rotterdam, Rotterdam.
- Freling, R., Huisman, D., Wagelmans, A.** (2003), Models and Algorithms for Integration of Vehicle and Crew Scheduling, *Journal of Scheduling*, v. 6(1), p. 63–85.
- Freling, R.; Wagelmans, A. P. M. e Paixão, J. M. P.** (1999), An overview of models and techniques for integrating vehicle and crew scheduling, in N. H. M. Wilson (ed.), *Computer-Aided Transit Scheduling*, Springer Verlag, Berlin, p. 441-460.
- Garey, M. R., Johnson, D. S.** (1979), Computers and Intractability, In W. H. Freeman (ed.), *A Guide to the Theory of NP-Completeness*, New York.
- Glover, F., Laguna, M.** (1997), *Tabu Search*, Kluwer Academic Publishers.
- Haase, K. e Friberg, C.** (1999), An exact branch and cut algorithm for the vehicle and crew scheduling problem, in N. H. M. Wilson (ed.), *Computer-Aided Transit Scheduling*, Springer Verlag, Berlin, p. 63-80.
- Hansen, P., Mladenovic, N.** (2003), Variable neighborhood search: Principles and applications, *European Journal of Operations Research*, v. 130, p. 449-467.
- Lourenço, H. R.; Martin, O.; Stützle, T.** (2003), Iterated Local Search, In F. Glover and G. Kochenberger (ed.), *Handbook of Metaheuristics*, Kluwer Academic Publishers.
- Marinho, E. H.** (2005), Heurísticas Busca Tabu para o Problema de Programação de Tripulações de Ônibus Urbano, Dissertação de mestrado, Niterói: Universidade Federal Fluminense, 89 p.
- Marinho, E. H., Ochi, L. S., Drummond, L. M. A., Souza, M. J. F., Silva, G. P.** (2004), Busca Tabu aplicada ao Problema de Programação de Tripulações de Ônibus Urbano, Simpósio Brasileiro de Pesquisa Operacional, 2004, São João Del Rey, Anais do XXXVI SBPO. Rio de Janeiro: SOBRAPO, v.1. p.1471 – 1482.
- Reis, J. v. A.** (2007), Programação integrada de veículos e tripulações de ônibus urbano, Exame de qualificação de mestrado, Programa de pós-graduação em Engenharia de Transportes, São Paulo: Escola Politécnica da Universidade de São Paulo.
- Scott, D.** (1985), A large linear programming approach to the public transport scheduling and cost model, in J. M. Rousseau (ed.), *Computer Scheduling of Public Transport 2*, North Holland, Amsterdam, p. 473-491.
- Souza, M. J. F., Silva, G. P., Simões, E. M. L.** (2007), Programação de ônibus urbano: uma abordagem heurística In: *Transporte em transformação XI: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2006*. Brasília: Positiva, 2007, v.1, p. 39-57.

UM ALGORITMO EVOLUTIVO HÍBRIDO PARA O PROBLEMA DE RECOBRIMENTO DE ROTAS COM COLETA DE PRÊMIOS

MATHEUS S. A. SILVA, MARCIO T. MINE, LUIZ S. OCHI,

*Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria 156 - Bloco E - 3º andar - São Domingos. CEP.: 24210-240
Niterói, RJ, Brasil*

MARCONE J. F. SOUZA

*Departamento de Computação - Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro. CEP.: 35400-000
Ouro Preto, MG, Brasil*

E-mails: {msalves;mmine;satoru}@ic.uff.br, marcone@iceb.ufop.br

Abstract— This paper proposes a hybrid evolutionary algorithm for getting an approximate solution of a generalization of the Traveling Salesman Problem (TSP) known in literature as the Prize Collecting Covering Tour Problem (PCCTP). Computational results on a set of instances illustrate the effectiveness and the robustness of the proposed heuristic.

Keywords— Computational Intelligence, Metaheuristics, Evolutionary Algorithm.

Resumo— Este artigo propõe um algoritmo evolutivo híbrido para obter soluções aproximadas para uma generalização do Problema do Caixeiro Viajante (PCV), conhecido na literatura como Problema de Recobrimento de Rotas com Coleta de Prêmios (PRRCP). Resultados computacionais para um conjunto de instâncias mostram a eficiência e a robustez da heurística proposta.

Palavras-chave— Inteligência Computacional, Metaheurísticas, Algoritmo Evolutivo.

1 Introdução

Com o aumento da complexidade dos processos produtivos, há uma necessidade cada vez maior da utilização de sistemas inteligentes, que auxiliem o processo de tomada de decisão da melhor maneira possível. Métodos clássicos de otimização têm encontrado dificuldade para obter a melhor solução, dita ótima, mesmo quando alguns deles possuem teoricamente a garantia de atingi-la. A elevada complexidade dos problemas de otimização encontrados em diferentes áreas tem provocado a necessidade de desenvolvimento de novos métodos mais eficientes na prática para solucionar tais problemas. Esses métodos são usualmente o resultado da adaptação de conceitos de várias áreas. Um exemplo bem sucedido são as metaheurísticas, ou heurísticas inteligentes. A principal característica desta categoria de métodos é a possibilidade de encontrar diferentes ótimos locais durante a busca pela melhor solução. Entre esses métodos, destacam-se os Algoritmos Evolutivos, os quais têm se mostrado eficientes na resolução de vários problemas combinatórios. Neste trabalho, desenvolvemos um Algoritmo Evolutivo Híbrido (AEH), que inclui fases de busca local por meio da metaheurística *Iterated Local Search* (Mladenovic e Hansen, 1997) e a heurística GENIUS (Gendreau *et al.*, 1992). Esse algoritmo é aplicado ao Problema de Recobrimento de Rotas com Coleta de Prêmios (PRRCP).

O PRRCP, referido na literatura como *Prize Collecting Covering Tour Problem* (PCCTP) é uma

variação do Problema de Recobrimento de Rotas (PRR) (*Covering Tour Problem* - CTP), que por sua vez é uma generalização do Problema do Caixeiro Viajante (PCV) (*Traveling Salesman Problem* - TSP).

O PRR pode ser definido em um grafo não-direcionado $G = (N, E)$, onde $N = V \cup W$ representa o conjunto dos vértices que podem fazer parte da solução (V), em união com os vértices que precisam ser cobertos (W). Os vértices do conjunto W são ditos cobertos quando existe na solução pelo menos um vértice de V a uma distância igual ou inferior a D de cada um desses vértices, sendo D um parâmetro do problema. O conjunto E é formado pelas arestas que conectam os vértices do conjunto V . Esse último conjunto é subdividido em dois outros conjuntos, T e $V \setminus T$, sendo o conjunto T formado pelos vértices obrigatórios, isto é, que *devem* fazer parte da rota e $V \setminus T$, o conjunto dos vértices *opcionais*, que não necessariamente precisam fazer parte da rota.

A partir destas considerações, tem-se que o PRR consiste em determinar uma rota de comprimento mínimo sobre um subconjunto de V , contendo todos os vértices obrigatórios T e cobrindo todos os vértices de W . Uma vez que o PRR pode ser reduzido ao PCV fazendo-se $D = 0$, $W = \emptyset$ e $N = T$, e este se enquadra na classe de problemas *NP-difíceis*, então o PRR também o é.

O PRRCP, além das restrições comuns ao PRR, possui as seguintes particularidades: (i) a cada vértice i de V está associado um prêmio não-negativo p_i ; (ii) a quantidade dos prêmios coletados nos vértices

presentes na solução tem que ser maior ou igual à quantidade mínima pré-estabelecida, dada por *PRIZE*. O objetivo do PRRCP, assim como no PRR, é encontrar uma rota de comprimento mínimo em um subconjunto de V , satisfazendo as restrições de pertinência de todos os vértices de T à solução, a cobertura de todos os vértices de W e por fim, a coleta da quantidade mínima de prêmios (*PRIZE*). Analogamente ao PRR, o PRRCP também é considerado um problema *NP-Difícil*, uma vez que pode ser reduzido ao PCV ($D = 0$, $N = T$, $PRIZE = 0$ e $W = \emptyset$).

2 Trabalhos Relacionados

O PRR foi formulado inicialmente por Current e Schilling (1989). Nesse trabalho, os autores abordam o PRR através de uma heurística de duas etapas, sendo que a primeira minimiza o comprimento da rota e a segunda maximiza o número de vértices cobertos pela rota gerada na primeira etapa. Posteriormente, (Gendreau *et al.*, 1995) propuseram um procedimento baseado na heurística GENIUS (Gendreau *et al.*, 1992) e no Algoritmo PRIMAL *Set Covering* (Balas and Ho, 1980) para resolução do PRR.

Além desse procedimento, nesse trabalho também foram propostas quatro regras de redução, um algoritmo exato baseado na técnica *Branch-and-Cut* e uma formulação matemática. Maniezzo *et al.* (2005) apresentaram uma formulação matemática para o PRR. Além disso, os autores também apresentam heurísticas *Scatter Search* para o PRR. Em Motta *et al.* (2001) é proposto uma nova formulação matemática e uma heurística GRASP para resolver o PRR.

Por ser um problema relativamente novo, existem poucos trabalhos na literatura relacionados ao PRRCP. Em Lyra (2004) é proposto uma formulação matemática, uma regra de redução dos vértices do problema e uma heurística GRASP (Feo e Resende, 1995) para o PRRCP. Na dissertação de Silva (2009) são propostas três novas regras de redução para o PRRCP, uma nova formulação matemática e duas abordagens heurísticas baseadas na metaheurística *Iterated Local Search* para resolução do PRRCP.

3 Metodologia proposta

Para gerar uma solução inicial são utilizados cinco algoritmos construtivos. Os algoritmos ADD e DROP foram utilizados por Lyra (2004) para o PRRCP. Os algoritmos de Inserção Mais Barata e do Vizinho Mais Próximo foram propostos para PCV e adaptados para o PRRCP. Por fim, utiliza-se uma adaptação da heurística GENIUS (Gendreau *et al.*, 1992).

Para explorar o espaço de soluções, utilizam-se cinco movimentos clássicos usados para o PCV, os quais são apresentados a seguir: (a) *Shift* - consiste em remover um vértice e reinseri-lo em outra posi-

ção; (b) *Swap* - troca dois vértices de suas posições; (c) *Or-Opt* - extensão do *shift*, consistindo em remover um conjunto de vértices adjacentes e os reinseri-los em outra posição; (d) *2-Opt* - remove duas arestas não adjacentes e insere duas novas arestas (e) *3-Opt* - remove três arestas não adjacentes e insere três novas arestas. Além desses movimentos, são usados quatro movimentos baseados nas fases de inserção e remoção da heurística GENIUS e na inserção do algoritmo de Inserção Mais Barata. Para avaliar uma solução, utiliza-se uma função $f(s)$, baseada em penalidade, que determina os custos totais de deslocamento. Essa função consiste em somar todos os custos (distâncias) de todas as arestas presentes na solução e penalizar o não atendimento das restrições do problema.

4 Algoritmo Evolutivo Híbrido (AEH) para o PRRCP

O termo Algoritmo Evolutivo (AE) compreende uma família de resolvidores de problemas com base em princípios que podem ser encontrados na evolução biológica (Eiben and Rudolph, 1999).

As três linhas principais de estudo nessa área são o *Algoritmo Genético* (AG), *Estratégias de Evolução* (EE) e *Programação Evolutiva* (PE).

Um AE típico funciona da seguinte forma: a cada geração constrói-se um conjunto de indivíduos, representando as soluções de um problema de otimização. Esses indivíduos, juntamente com a população vinda de gerações anteriores, são combinados numa fase de cooperação para formarem novos indivíduos. Esses novos indivíduos passam, então, por uma fase de adaptação antes de se decidir quais serão incluídos na população que irá para a próxima geração. O algoritmo finaliza ao se atingir um número máximo de gerações (Hertz and Kobler, 2000).

No Algoritmo Evolutivo proposto para resolver o PRRCP, utiliza-se o conceito de classes para diferenciar os indivíduos. A classe *A*, representada pelo *pool* de soluções elite, contém os melhores indivíduos em termos de custo da solução, encontrados durante as gerações. Esses indivíduos são substituídos somente se melhores indivíduos forem criados. A classe *B* é representada por uma parcela da população que também só é substituída caso indivíduos de melhor custo sejam gerados.

Por fim, a classe *C* é representada pela outra parcela restante da população com indivíduos de custos mais elevados, substituídos em todas as gerações. Para gerar a população inicial, utilizam-se versões randomizadas dos cinco métodos construtivos apresentados na Seção 3.

Como os métodos possuem idéias diferentes, diversificando-os na geração da população inicial, diversificam-se os indivíduos, promovendo assim uma miscigenação entre os indivíduos.

O Algoritmo 1 apresenta o pseudocódigo do Algoritmo Evolutivo Híbrido proposto.

```

procedimento Algoritmo Evolutivo Híbrido
  enquanto (!completo (populacao, pool) )
     $s_0 \leftarrow \text{SolucaoInicial}$ 
     $s \leftarrow \text{MRD}(s_0)$ 
     $\text{pool} \leftarrow s$ 
    se !poolAtualizado então
       $\text{populacao} \leftarrow s$ 
    fim se
  fim enquanto
  enquanto ( $\text{ger} < \text{ger}_{\text{max}}$ )
     $s_{\text{base}} \leftarrow s \in \text{populacao}$ 
     $s_{\text{guia}} \leftarrow s \in \text{pool}$ 
     $s_{\text{RC}} \leftarrow \text{Reconexão por Caminhos}(s_{\text{base}}, s_{\text{guia}})$ 
     $s \leftarrow \text{ILS-VNRD}(s_{\text{RC}})$ 
     $\text{pool} \leftarrow s$ 
  para ( $n_{\text{ind}} < \text{classeC}$ )
     $s_0 \leftarrow \text{SolucaoInicial}$ 
     $s \leftarrow \text{MRD}(s_0)$ 
     $\text{populacao} \leftarrow s$ 
  fim para
   $\text{ger} \leftarrow \text{ger} + 1;$ 
fim enquanto
 $s \leftarrow s^* \in \text{pool}$ 
retorne  $s$ 

```

Algoritmo 1: Algoritmo Evolutivo Híbrido

Neste algoritmo, inicialmente são criados vários indivíduos para compor uma *populacao* e o *pool* de soluções elite. Cada indivíduo é então submetido a uma busca local realizada pelo Método Randômico de Descida - MRD (descrito na Subseção 4.1). Se esse indivíduo s for melhor que o indivíduo que possui o pior custo dentre todas as soluções do *pool* e possuir um percentual mínimo de diferença para todas as soluções desse conjunto elite, então atualiza-se o *pool* com a inclusão de s . Caso esse indivíduo não satisfaça os requisitos para entrar no *pool* (*pool*Atualizado), verifica-se se s pode fazer parte da *populacao*. Para isso é preciso que ele seja melhor que a pior solução desse conjunto, não precisando ter um percentual mínimo de diferença em relação aos demais indivíduos. Criada a *populacao* inicial, enquanto o número máximo de geração (ger_{max}) não for alcançado, aleatoriamente escolhe-se uma solução da *populacao* para ser a solução base do procedimento *Reconexão por Caminhos* (vide Subseção 4.2). A solução guia é selecionada do *pool* de soluções elite de acordo com o grau de diferença em relação à solução base. A solução que possuir o maior grau de diferença será escolhida como guia. A melhor solução encontrada durante a *Reconexão por Caminhos* é então submetida a uma busca local realizada pelo procedimento *ILS-VNRD* descrito na Subseção 4.3. Caso a solução s retornada pelo *ILS-VNRD* seja melhor que a solução de maior custo do *pool* e se s possuir uma porcentagem mínima de diferença para as demais soluções do conjunto elite, atualiza-se o *pool* com s . Ao final de cada geração reconstrói-se a clas-

se C da *populacao*, substituindo-se seus indivíduos por novos indivíduos. Como a *populacao* é ordenada pelo custo dos indivíduos, pode acontecer de um novo indivíduo criado para a classe C possuir um custo menor que um indivíduo da classe B . Quando isto ocorre, realiza-se a inversão de classes entre os indivíduos, isto é, automaticamente o novo indivíduo passa a ser da classe B e o outro indivíduo que pertencia a esta classe passa agora a pertencer à classe C . Ao final do algoritmo, retorna-se o melhor indivíduo (solução) do *pool*.

4.1 Método Randômico de Descida – MRD

O Método Randômico de Descida, ou MRD, é uma heurística de refinamento que consiste em analisar um vizinho qualquer de uma dada solução e o aceitar somente se ele for estritamente melhor que a solução corrente. Caso esse vizinho não seja melhor, a solução corrente permanece inalterada e outro vizinho é gerado. O procedimento é finalizado quando se atinge um número máximo de iterações sem que haja melhorias no valor da melhor solução obtida.

4.2 Reconexão por Caminhos – RC

O procedimento de Reconexão de Caminhos (*Path Relinking*) RC foi utilizado como fase de intensificação para o AEH proposto. Este algoritmo inicia calculando a diferença simétrica de todas as soluções do *pool* de soluções elite em relação à solução base, s_{base} , proveniente da *populacao*. A diferença simétrica pode ser entendida como a quantidade de passos que são necessários para sair de uma solução base e chegar à solução guia. Sair de uma solução base e chegar a uma solução guia significa que a cada iteração, um atributo que exista na solução guia, mas que não pertença à solução base é inserido na solução base com o objetivo de que ao final do procedimento a solução base coincida com a solução guia.

Definida a solução base, s_{base} , enquanto ela possuir alguma diferença para s_{guia} , o procedimento aplica um movimento que consiste em inserir na s_{base} um atributo de s_{guia} . O movimento escolhido é o melhor dentre os possíveis movimentos que podem ser aplicados, isto é, o que trouxer o maior benefício para a solução. A essa solução intermediária, também chamada de candidata (s_{cand}), aplica-se uma busca local por meio do MRD. Se essa solução s for melhor que a pior solução do *pool* e atender ao critério de diversificação de soluções, então atualiza-se o *pool* com s . O procedimento termina quando ocorre a convergência das soluções, isto é, as soluções base e guia possuem os mesmos vértices e a ordem de visita dos vértices em ambas as soluções é a mesma.

4.3 ILS-VNRD

O procedimento ILS-VNRD tem como solução inicial a melhor solução encontrada durante a Reconexão por Caminhos (Subseção 4.2). Em seguida, aplica-se um mecanismo de busca local, o VNRD (Subseção

4.4). A cada iteração do procedimento ILS-VNRD, gera-se uma perturbação na solução corrente. Essa perturbação consiste em realizar k movimentos Aleatórios dentre aqueles descritos na Seção 3, sendo k um número arbitrário compreendido entre 1 e um valor parametrizado, denominado kp .

A essa solução perturbada, aplica-se o VNRD, tentando com isso gerar uma solução melhorada. Se essa for melhor que a solução corrente, então a solução melhorada é aceita como a nova solução corrente e reinicia-se o valor de kp . Essa repetição é executada até que um número máximo de iterações sem melhora seja realizado. Quando isso ocorre, incrementa-se o grau de perturbação, kp , em um fator δ , até que kp atinja seu valor máximo (kp_{max}). A variação do grau de perturbação é uma estratégia importante, pois permite a intensificação da busca e diversificação das soluções geradas. A busca é intensificada quando o grau de perturbação é baixo e a diversificação ocorre quando esse grau atinge um valor relativamente alto.

4.4 Método de Descida Randômica em Vizinhos Variável – MDRVV

O Método de Descida Randômica em Vizinhos Variável (*Variable Neighborhood Random Descent*, VNRD), é uma variante do Método de Descida em Vizinhos Variável (*Variable Neighborhood Descent*, VND), proposto por Nenad Mladenovic e Pierre Hansen em 1997. O VND é um método de refinamento que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança (descritas na Seção 3), aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada. A grande diferença do VNRD para o VND é que a exploração de cada vizinhança não é feita por completa, apenas um percentual desta é explorada. O valor desse percentual é um parâmetro passado para o método. Este método finaliza quando se aplica todas as estruturas de vizinhança e não se encontra nenhuma solução de melhora.

5 Resultados e Conclusões

O Algoritmo Evolutivo Híbrido foi implementado na linguagem C++ utilizando o ambiente Microsoft Visual Studio 2008 e testado em um computador Intel Core 2 Duo, com 2.2 GHz e 2.5 GB de memória principal, rodando o sistema operacional Windows Vista. Para validá-lo, foram utilizados problemas-teste adaptados para o PRRCP a partir do TSPLIB, proposto em Silva (2009). A justificativa para não se ter utilizado os problemas-teste do trabalho de Lyra (2009) se deve à não disponibilidade de tais problemas na literatura. Em Silva (2009) foram criados problemas-teste com coleta mínima de 25%, 50% e 75% do prêmio total dos vértices de V . Com isso, em grande parte dos problemas, alguns vértices do con-

junto VT devem estar obrigatoriamente presentes na solução, o que dificulta a resolução do problema, pois o número de combinações aumenta exponencialmente.

Foram utilizados dois grupos de problemas para validar a metodologia proposta. Esses grupos, chamados de Grupo 1 e Grupo 2 respectivamente, possuem problemas com o número de vértices variando entre 50 e 200 para o Grupo 1 e de 201 a 400 para o Grupo 2. Para cada problema foram realizadas 10 execuções, cada qual partindo de uma semente diferente de números aleatórios. Os parâmetros adotados no Algoritmo Evolutivo foram: $maxGer = 7$, $populationSize = 5$, $classBSize = 3$, $classCSize = 2$, $eliteSetSize = 5$, $percDiffSolution = 0,15$, $iterMaxMRD = 300$, $iterMaxILS = 100$, $iterMaxVNRD = 100$, $kpMinILSVNRD = 5$, $kpMaxILSVNRD = 7$ e $\delta = 2$. Tais valores foram definidos através de testes preliminares.

Para legitimar o desempenho do AEH proposto, este foi comparado com as duas melhores versões, descritas em Silva (2009), de um algoritmo de busca local baseado na metaheurística *Iterated Local Search*, tendo como método de busca local o MRD. Para o Grupo 1, o AEH foi comparado com a versão ILS-MRD_AD, a qual tem o procedimento ADD como método de geração de solução inicial. Para o Grupo 2, o AEH foi comparado com a versão ILS-MRD_IB, a qual usa o procedimento da Inserção Mais Barata para construir uma solução.

Esses algoritmos foram comparados pelos seguintes critérios: (a) *custo médio de todos os problemas do grupo*; (b) *desvio médio das soluções em relação ao melhor custo conhecido*; (c) *taxa de sucesso de cada algoritmo em alcançar o melhor custo em pelo menos uma das dez execuções*; (d) *porcentagem de problemas que cada algoritmo encontrou o menor custo sozinho* e (e) *tempo médio gasto em cada execução do algoritmo*. O desvio médio é realizado pela equação $100(Média-MelhorValor)/MelhorValor$.

De acordo com os critérios considerados, observou-se que, para os problemas do Grupo 1, o AEH obteve um desempenho superior ao ILS-MRD_AD, já que o custo médio das soluções obtidas pelo AE foi de 20073 enquanto o ILS-MRD_AD obteve um custo médio de 20080. O desvio médio das soluções do AE foi inferior ao do ILS-MRD_AD, no caso, de 0,02% contra 0,06% do outro algoritmo. Além disso, o AEH encontrou o melhor valor conhecido em 99,1% dos casos do Grupo 1 contra os 96,4% obtidos pelo ILS-MRD_AD, e em 1,8% dos problemas-teste, o AE encontrou esse melhor valor sozinho enquanto o outro algoritmo não obteve sucesso nesse critério. Em relação aos tempos de execução, verifica-se que o AEH demandou mais tempo (1538 segundos contra 1256 segundos do ILS-MRD_AD).

Em relação aos problemas do Grupo 2, observa-se que a diferença do desempenho do AEH em relação ao ILS-MRD_IB, avaliado pelos critérios propostos, foi mais significativa. De fato, o AEH obteve um custo médio das soluções de 33641, enquanto o ILS-MRD_IB obteve um custo médio de 33752. O desvio médio das soluções do AE também foi inferi-

or ao do ILS-MRD_IB, no caso, de 0,14% contra 0,34% do outro algoritmo. Além disso, o AEH encontrou o melhor valor conhecido em mais de 90% dos casos do Grupo 1 contra os 76,7% obtidos pelo ILS-MRD_IB, e em 16,7% dos problemas-teste, o AEH superou a melhor solução conhecida nos problemas-teste desse grupo. Ainda em relação as instâncias do Grupo 1, foi mostrado em Silva (2009) a solução ótima de parte dos problemas-teste deste grupo utilizando uma formulação matemática e o *software* CPLEX 11; e tanto o ILS-MRD_IB como o AEH alcançaram o valor ótimo sempre que este é conhecido.

Em relação aos tempos de execução, usando como critério de parada um número fixo de iterações, verifica-se que o AEH gastou cerca do dobro do tempo do ILS-MRD_IB (9294 segundos do AEH contra 4897 segundos do outro algoritmo). Este resultado já era esperado, pois uma iteração (geração do AEH é muito mais trabalhoso que uma iteração do ILS-MRD_IB. Neste contexto, novos testes computacionais foram efetuados usando agora como critério de parada, um valor alvo a ser atingido. Foi verificado que o AEH possui uma convergência empírica melhor quando se deseja encontrar uma solução de boa qualidade, ou seja, se colocarmos um valor alvo *difícil* (solução ótima ou sub-ótima) o AEH tende a ser mais rápido do que o algoritmo ILS-MRD_IB.

Estes últimos testes mostram que apesar do AEH possuir um tempo computacional maior por iteração, este tende a requerer na média menos iterações para obter um valor alvo de boa qualidade.

De acordo com os resultados apresentados, o AEH foi o que gerou as melhores soluções para a maioria dos problemas-teste. Esse mérito se deve à característica do algoritmo em trabalhar com os dois tipos de buscas, local e populacional, mesclando soluções de boa qualidade com soluções de qualidade inferior, fazendo com que a busca local tenha mais chances de melhorar a solução. O algoritmo se mostrou robusto por apresentar soluções finais com baixa variabilidade, no caso, sempre inferior a 0,14%.

5 Conclusões

Este trabalho abordou o Problema de Recobrimento de Rotas com Coleta de Prêmios (PRRCP). Para resolvê-lo, foi proposto um Algoritmo Evolutivo Híbrido, que mescla busca local com busca populacional. Para gerar os indivíduos da população foram adaptados cinco métodos de geração de solução inicial do PCV para serem utilizados no PRRCP. Para combinar diferentes indivíduos, foi utilizado o procedimento Reconexão por Caminhos e para refinar esses indivíduos, utilizou-se a metaheurística *Iterated Local Search* (ILS), tendo o *Variable Neighborhood Random Descent* (VNRD) como método de busca local. O VNRD explora a vizinhança de

uma solução por meio de várias estratégias apresentadas na Seção 3. De acordo com os resultados obtidos, verifica-se que o algoritmo proposto é eficiente e robusto para resolver o PRRCP, sendo capaz de produzir soluções de boa qualidade.

Agradecimentos

Os autores agradecem a CAPES, CNPq, FAPERJ e FAPEMIG pelo apoio ao desenvolvimento deste trabalho.

Referências Bibliográficas

- Balas, E. and Ho, A. (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimizations: A computational study. *Mathematical Programming*, **12**:37-70.
- Current, J. R. and Schilling, D. A. (1989). The covering salesman problem, *Transportation Science* **23**: 208-213.
- Eiben, A. E. and Rudolph, G. (1999). Theory of evolutionary algorithms: A birds eye view. In *Theoretical Computer Science*. 229: 3-9.
- Feo, T. and Resende, M. (1995). Greedy randomized search procedure, *Journal of Global Optimization* **6**: 109-133.
- Gendreau, M., Hertz, A. and Laporte, G. (1992). New insertion and post optimization procedures for the traveling salesman problem, *Oper. Res.***40**: 1086-1094.
- Gendreau, M., Laporte, G. and Semet, F. (1995). The Covering tour problem, *Oper. Res.* **45**: 568-576.
- Hertz, A. and Kobler, D. (2000). A framework for the description of evolutionary algorithms. *European Journal of Oper. Res.*, **126**:1-12.
- Lyra, A. R. de. (2004). *O problema de recobrimento de rotas com coleta de prêmios: Regras de redução, formulação matemática e heurísticas*, Master's thesis, IC/UFF.
- Maniezzo, V., Baldacci, R. and Zamboni, M. (2005). Scatter search methods for the covering tour problem, In *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, vol 30: 59-91. Kluwer.
- Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search, *Computers and Operations Research* **24**: 1097-1100.
- Silva, M. de S. A. (2009). *Problema de Recobrimento de Rotas com Coleta de Prêmios*, Master's thesis, IC/UFF.
- Motta, L. C. S., Ochi, L. S. and Martinhon, C. A. (2001). Reduction rules for the covering tour problems, *Electronic Notes In Discrete Applied Mathematics* **7**: 168-171.

O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA: UMA ABORDAGEM VIA ITERATED LOCAL SEARCH E GENIUS¹

Marcio Tadayuki Mine
Matheus de Souza Alves Silva
Luiz Satoru Ochi

Universidade Federal Fluminense
Instituto de Computação

Marccone Jamilson Freitas Souza

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas

RESUMO

Este trabalho considera o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). O PRVCES é um problema básico na área da logística reversa, a qual visa planejar o transporte de produtos aos clientes, bem como o retorno de resíduos ou produtos utilizados por esses para a reciclagem ou depósitos especializados. O PRVCES pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao Problema de Roteamento de Veículos clássico quando nenhum cliente necessita de serviço de coleta. Para resolvê-lo, propõe-se um algoritmo heurístico híbrido, denominado GENILS, baseado nas técnicas *Iterated Local Search*, *Variable Neighborhood Descent* e GENIUS. O algoritmo proposto foi testado em três conjuntos consagrados de problemas-teste da literatura e se mostrou competitivo com os melhores algoritmos existentes.

ABSTRACT

This work deals with the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). The VRPSPD is a common problem in the area of reverse logistics, which aims to plan the transportation of products to customers, as well as the return of leavings or products used by them for recycling or to special depots. The VRPSPD is NP-hard, since it can be reduced to the classical Vehicle Routing Problem when no client needs the pickup service. To solve it, we propose a hybrid heuristic algorithm, called GENILS, based on *Iterated Local Search*, *Variable Neighborhood Descent* and GENIUS. The proposed algorithm was tested on three well-known sets of instances found in literature and it was competitive with the best existing approaches.

1. INTRODUÇÃO

O Problema de Roteamento de Veículos (PRV), conhecido na literatura como *Vehicle Routing Problem* (VRP), foi originalmente proposto por Dantzig e Ramser (1959) e pode ser definido da seguinte forma: Dado um conjunto N de clientes, cada qual com uma demanda d_i e uma frota de veículos homogênea com capacidade Q , tem-se como objetivo, estabelecer os trajetos de custo mínimo a serem percorridos pelos veículos, de forma a atender completamente a demanda dos clientes numa única visita.

Em 1989, Min propôs uma importante variante do PRV: o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), em que os serviços de entrega e coleta devem ser realizados simultaneamente. Este modelo é um problema básico na área da logística reversa, a qual visa planejar o transporte de produtos aos clientes, bem como o retorno de resíduos ou produtos utilizados por esses para a reciclagem ou depósitos especializados. A logística reversa pode ser observada, por exemplo, na logística postal ou no planejamento da distribuição de indústria de bebidas.

¹ Trabalho vencedor do prêmio de produção acadêmica conferido pela Associação Nacional de Pesquisa e Ensino em Transportes (ANPET) e Confederação Nacional do Transporte (CNT). Será publicado em capítulo de livro da série Transporte em transformação IX: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2009

O PRVCES pertence à classe de problemas NP-difíceis, uma vez que ele pode ser reduzido ao PRV clássico quando nenhum cliente necessita de serviço de coleta. Dessa forma, diversos trabalhos na literatura o tratam de forma heurística.

Min (1989) propôs um método de três fases para resolver o planejamento de distribuição de materiais para bibliotecas públicas. A primeira fase do método consiste em agrupar os clientes em *clusters* por meio do Método de Ligação por Médias (*Average Linkage Method*) (Anderberg, 2007). A segunda fase associa os veículos às respectivas rotas e a terceira consiste em resolver cada *cluster* por meio de uma heurística para o Problema do Caixeiro Viajante. Essa heurística atribui, iterativamente, uma penalidade aos arcos em que a carga do veículo foi excedida, procurando, dessa forma, gerar uma solução viável.

Halse (1992) aborda o PRVCES por meio de uma heurística que consiste em, primeiramente, associar os clientes aos veículos e, em seguida, gerar as rotas através de um procedimento baseado no método *3-optimal*.

Dethloff (2001) desenvolveu uma adaptação do método da Inserção Mais Barata, em que os clientes são adicionados às rotas seguindo três critérios: (i) distância; (ii) capacidade residual e (iii) distância do cliente ao depósito. Nesse trabalho não foi aplicado nenhum método de refinamento da solução.

Vural (2003) desenvolveu duas versões do Algoritmo Genético (Goldberg, 1989). A primeira faz a codificação dos indivíduos através de chaves aleatórias (*Random Keys*) (Bean, 1994) e a segunda foi implementada como uma heurística de refinamento, baseada na estrutura do AG desenvolvido por Topcuoglu e Sevilmis (2002).

Gökçe (2004) trata o PRVCES com a metaheurística Colônia de Formigas (Dorigo *et al.*, 1996) e utiliza o método *2-optimal* como um procedimento de pós-otimização.

Nagy e Salhi (2005) desenvolveram uma metodologia para a resolução do PRVCES com a restrição de limite de tempo para percorrer cada rota. Essa metodologia reúne diferentes heurísticas para resolver o PRV clássico, tais como, *2-optimal*, *3-optimal*, realocação, troca e, além disso, procedimentos para viabilizar a solução.

Dell'Amico *et al.* (2006) utilizaram a técnica *branch-and-price* por meio de duas abordagens: programação dinâmica e relaxação do espaço de estados (*state space relaxation*).

Crispim e Brandão (2005) propõem uma técnica híbrida, combinando as metaheurísticas Busca Tabu (Glover e Laguna, 1997) e *Variable Neighborhood Descent* – VND (Hansen e Mladenović, 2001). Para gerar uma solução foi utilizado o método da varredura (*sweep method*) e, para refiná-la, um procedimento de busca local que explora o espaço de soluções com movimentos de realocação e troca.

Röpke e Pisinge (2006) desenvolveram uma heurística baseada na *Large Neighborhood Search* – LNS (Shaw, 1998). O LNS é uma busca local baseada em duas idéias para definir e explorar estruturas de vizinhança de alta complexidade. A primeira idéia é fixar uma parte da solução e assim definir o espaço de soluções. A segunda consiste em realizar a busca por meio

de programação por restrições, programação inteira, técnicas *branch-and-cut*, entre outras. Montané e Galvão (2006) utilizaram a metaheurística Busca Tabu considerando quatro tipos de estruturas de vizinhança. Essas estruturas utilizam os movimentos de realocação, troca e *crossover*. Para gerar uma solução vizinha foram desenvolvidas duas estratégias, sendo que uma considera o primeiro movimento viável e a outra, o melhor movimento viável. Chen (2006) trata o problema por meio de uma técnica baseada nas metaheurísticas *Simulated Annealing* (SA) (Kirkpatrick *et al.*, 1983) e Busca Tabu, enquanto Chen e Wu (2006) desenvolveram uma metodologia baseada na heurística *record-to-record travel* (Dueck, 1993), a qual é uma variação do SA.

Algoritmos construtivos, heurísticas de refinamento e técnicas baseadas na metaheurística Busca Tabu são apresentados em Bianchessi e Righini (2007). Essas técnicas utilizam movimentos de troca de nós (*node-exchange-based*) e troca de arcos (*arc-exchange-based*).

Wassan *et al.* (2007) propõem uma versão reativa da metaheurística Busca Tabu. Para gerar uma solução inicial, foi utilizado o método da varredura (*sweep method*) e para explorar o espaço de soluções, movimentos de realocação, de troca e de inversão do sentido da rota.

Em Subramanian *et al.* (2008) foi desenvolvido um algoritmo baseado em *Iterated Local Search* (ILS), tendo como busca local o procedimento *Variable Neighborhood Descent* (VND). Para gerar uma solução inicial foi utilizada uma adaptação da heurística de inserção de Dethloff (2001), porém sem considerar a capacidade residual do veículo. O VND explora o espaço de soluções usando movimentos baseados em realocação, troca e *crossover*. O VND realiza, a cada melhora na solução corrente, uma intensificação nas rotas alteradas, por meio dos procedimentos de busca local *Or-opt*, *2-opt*, *exchange* e *reverse*. O procedimento *Or-opt* que foi implementado consiste em permutar um, dois ou três clientes consecutivos em uma rota. O *2-opt* e o *exchange* realizam a permutação de um par de arcos e dois clientes, respectivamente. O movimento *reverse* consiste em inverter o sentido da rota, caso haja redução na carga do veículo nos arcos. Os mecanismos de perturbação aplicados no ILS foram o *ejection chain*, o *double swap* e o *double bridge*. O *ejection chain* consiste em transferir um cliente de cada rota a outra adjacente. O *double swap* consiste em realizar duas trocas sucessivas e o *double bridge* consiste em remover quatro arcos e inserir quatro novos arcos. Uma descrição detalhada desse algoritmo, bem como uma nova formulação de programação matemática para o PRVCES pode ser encontrada em Subramanian (2008).

Zachariadis *et al.* (2009) abordada o PRVCES com uma técnica híbrida, combinando as metaheurísticas Busca Tabu e *Guided Local Search* (Voudouris e Tsang, 1996).

Uma boa revisão do PRVCES pode ser encontrada em Parragh *et al.* (2008a, 2008b).

Para comparar as abordagens da literatura, Dethloff (2001) propôs um conjunto com 40 problemas envolvendo 50 clientes cada. Salhi e Nagy (1999) apresentaram 28 problemas-teste com 50 a 199 clientes, sendo que a metade desses têm restrições de limite de tempo. Por fim, Montané e Galvão (2006) adaptaram 18 problemas-teste de Solomon *et al.* (2005) e Gehring e Homberger (1999), envolvendo 100, 200 e 400 clientes.

Até o momento, os melhores resultados encontrados na literatura para esses problemas-teste pertencem a:

- Chen e Wu (2006): um problema-teste de Salhi e Nagy (1999);
- Röpke e Pisinger (2006): 26 problemas-teste de Dethloff (2001);
- Wassan *et al.* (2007): 6 problemas-teste de Salhi e Nagy (1999);
- Zachariadis *et al.* (2009): 6 problemas de Salhi e Nagy (1999) e 27 de Dethloff (2001);
- Subramanian (2008) e Subramanian *et al.* (2008): todos os problemas-teste de Dethloff (2001) e Montané e Galvão (2006) e 17 de Salhi e Nagy (1999).

Neste trabalho é apresentado um novo algoritmo heurístico para resolver o PRVCES. O algoritmo proposto, denominado GENILS, combina as técnicas *Iterated Local Search*, *Variable Neighborhood Descent* e uma adaptação da heurística GENIUS. Ele difere do de Subramanian *et al.* (2008) basicamente por incluir a heurística GENIUS e os procedimentos *3-optimal* e *4-optimal* na exploração do espaço de busca. Conforme mostram os resultados, estas estratégias se mostraram eficientes na resolução do problema.

O restante deste trabalho está organizado como segue. Na Seção 2 descreve-se o problema abordado. A metodologia proposta é apresentada na Seção 3. Na Seção 4 são apresentados e analisados os resultados encontrados, enquanto na Seção 5 conclui-se o trabalho, apontando trabalhos futuros.

2. DESCRIÇÃO DO PROBLEMA

O Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), ou *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD), é uma variante do PRV clássico. Neste problema existe um depósito com uma frota ilimitada de veículos de capacidade Q e um conjunto N de clientes espalhados geograficamente. Cada cliente $i \in N$ está associado a duas quantidades d_i e p_i , que representam a demanda por um determinado produto e a coleta no cliente i , respectivamente. O objetivo do problema é definir as rotas necessárias para atender a todos os clientes, de forma a minimizar os custos referentes ao deslocamento dos veículos e satisfazer as seguintes restrições: (a) cada rota deve iniciar e finalizar no depósito; (b) todos os clientes devem ser visitados uma única vez e por um único veículo; (c) as demandas por coleta e entrega de cada cliente devem ser completamente atendidas; (d) a carga do veículo, em qualquer momento, não pode superar a capacidade do veículo. Em algumas variantes desse problema, considera-se também a necessidade de cada veículo não percorrer mais que um determinado limite de distância (tempo). A Figura 1 ilustra um exemplo do PRVCES.

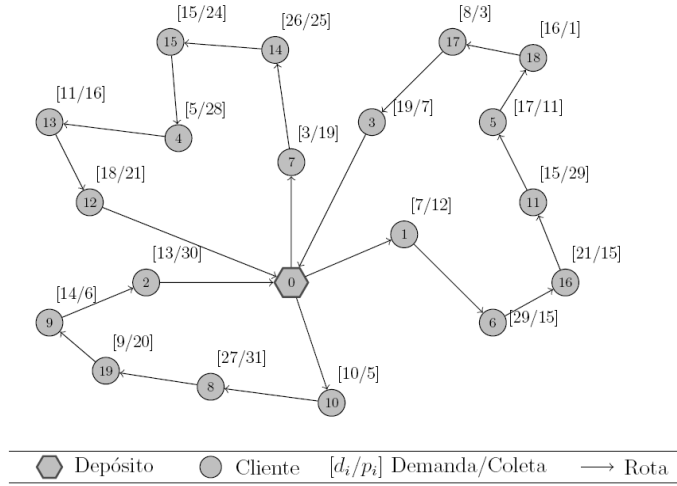


Figura 1: Exemplo do PRVCES

Na Figura 1, os clientes são representados pelos números 1 a $|N|$ e o depósito pelo número 0 (zero). Cada par $[d_i / p_i]$ denota a demanda e coleta em um cliente i , respectivamente. Nesta figura, há três rotas a serem executadas por veículos de capacidade $Q = 150$. Em uma delas, o veículo sai do depósito e atende aos clientes 10, 8, 19, 9 e 2, retornando ao depósito no final. No primeiro cliente atendido nessa rota, é feita uma entrega de 10 unidades do produto e recolhida outras 5 unidades. A última visita do veículo ocorre no cliente 2, o qual demanda 13 unidades do produto e necessita que sejam coletadas 30 unidades.

3. METODOLOGIA

Apresenta-se, nesta Seção, a metodologia desenvolvida para resolver o PRVCES. Na Subseção 3.1 mostra-se como gerar uma solução inicial. Na Subseção 3.2 são mostrados os movimentos usados para explorar o espaço de busca, enquanto em 3.3 é mostrado como uma solução é avaliada. Na Subseção 3.4 é apresentado o algoritmo GENILS desenvolvido para resolver o problema.

3.1. Geração de uma solução inicial

Para gerar uma solução inicial são utilizadas três heurísticas baseadas em inserção. A primeira, denominada IMB-1R, é uma adaptação da heurística de Inserção Mais Barata, a qual constrói uma solução rota a rota. A segunda, denominada IMB-NR, foi proposta por Subramanian *et al.* (2008) e baseia-se na heurística de inserção de Dethloff (2001). A última, VRGENIUS, proposta neste trabalho, é uma adaptação da heurística GENIUS (Gendreau *et al.*, 1992), desenvolvida para o Problema do Caixeiro Viajante, e possui duas fases: uma construtiva (VRGENI) e outra de refinamento (VRUS). A fase VRGENI é um método de inserção, cuja característica fundamental é que a inclusão de um cliente não é realizada necessariamente entre dois outros clientes consecutivos. No entanto, esses dois clientes tornam-se adjacentes após a inserção. Já a fase VRUS consiste em, a cada iteração, remover um cliente da solução e reinseri-lo em outra posição que vise melhorar a solução corrente. Caso essa melhora não seja possível com nenhum cliente, esta fase é finalizada. Vale ressaltar que, em ambas as fases, tanto a remoção, quanto a inserção de um cliente é realizada por procedimentos que analisam um espaço reduzido da vizinhança explorada pelas buscas locais 3-optimal e 4-optimal. A eficiência desses procedimentos encontra-se no fato de que o espaço analisado é restrito ao número de vizinhos de cada cliente, sendo, no máximo, igual a um

parâmetro p .

3.2. Estruturas de vizinhança

Para explorar o espaço de soluções do problema, aplicam-se, neste trabalho, sete tipos diferentes de movimentos, a saber: (a) *Shift*: movimento de realocação que consiste em transferir um cliente de uma rota para outra; (b) *Shift(2,0)*: movimento semelhante ao *Shift*, porém realocando dois clientes consecutivos de uma rota para outra; (c) *Swap*: consiste em trocar um cliente i de uma rota r_1 com um outro cliente j de uma rota r_2 ; (d) *Swap(2,1)*: é análogo ao *Swap*, porém trocando dois clientes consecutivos de uma rota com um cliente de outra rota; (e) *Swap(2,2)*: consiste em realizar a troca de dois clientes consecutivos de uma rota com dois outros clientes consecutivos de outra rota; (f) *M2-Opt*: consiste em remover dois arcos e inserir dois novos arcos; (g) *kOr-Opt*: consiste em remover k clientes consecutivos de uma rota r e, em seguida, reinseri-los em uma outra posição nessa mesma rota. O valor de k é um parâmetro. Esse movimento é uma generalização do *Or-Opt* proposto por Or (1976), em que é realizada a remoção de no máximo três clientes consecutivos. Destaca-se que não são permitidos movimentos que conduzam a soluções inviáveis.

3.3. Função de avaliação

Uma solução s é avaliada pela função f apresentada pela Equação (1), que determina o custo total de deslocamento.

$$f(s) = \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (1)$$

em que N : conjunto dos clientes, incluindo o depósito;
 A : conjunto dos arcos (i, j) , com $i, j \in N$;
 c_{ij} : custo de deslocamento ou distância de um cliente i a um cliente j ;
 x_{ij} : indica se o arco $(i, j) \in A$ é utilizado ($x_{ij} = 1$) ou não ($x_{ij} = 0$) na solução.

3.4. Algoritmo GENILS

Para resolver o PRVCES propõe-se um algoritmo híbrido, denominado GENILS. Este algoritmo utiliza o método de geração de solução inicial descrito na Subseção 3.2, e combina os procedimentos heurísticos *Iterated Local Search* – ILS (Stützle e Hoos, 1999), *Variable Neighborhood Descent* – VND (Hansen e Mladenović, 2001) e uma adaptação da heurística GENIUS (Gendreau *et al.*, 1992). Seu pseudocódigo é apresentado pelo Algoritmo 1.

```

Algoritmo GENILS
 $s^A \leftarrow$  construa uma solução com a IMB-1R
 $s^B \leftarrow$  construa uma solução com a IMB-NR
 $s^C \leftarrow$  construa uma solução com a VRGENIUS
 $s^A \leftarrow$  VND( $s^A$ )
 $s^B \leftarrow$  VND( $s^B$ )
 $s^C \leftarrow$  VND( $s^C$ )
 $s \leftarrow s' \mid f(s') = \min\{f(s^A), f(s^B), f(s^C)\}$ 
 $iter \leftarrow 0$ 
enquanto ( $iter < iter_{max}$ )
     $iter \leftarrow iter + 1$ 
     $s' \leftarrow$  perturbação( $s$ )
     $s'' \leftarrow$  VND( $s$ )
    se ( $f(s'') < f(s)$ ) faça
         $s \leftarrow s''$ 

```

```

iter ← 0
fim-se
fim-enquanto
retorne s

```

Algoritmo 1: GENILS

O algoritmo GENILS inicia gerando três soluções iniciais s^A , s^B , e s^C , cada qual por um dos métodos descritos na Subseção 3.4. Essas soluções são, a seguir, refinadas pelo VND e a melhor solução obtida é usada como a solução inicial s . Para escapar do ótimo local s , é feita uma perturbação, gerando uma nova solução s' . Em seguida, essa solução perturbada é refinada pela busca local VND, obtendo-se um novo ótimo local s'' . Esta solução torna-se a nova solução corrente caso s'' seja melhor que s ; caso contrário, ela é descartada e nova perturbação é feita a partir da solução s . Esse procedimento é repetido até o número máximo de iterações sem melhora na solução corrente ($iter_{max}$) seja atingido.

As perturbações são realizadas por um dos três mecanismos descritos a seguir, escolhidos aleatoriamente:

- *Múltiplos Shift*: Consiste em realizar k movimentos *Shift* (descrito na Subseção 3.2) sucessivamente. O valor de k é definido aleatoriamente entre 1, 2 ou 3;
- *Múltiplos Swap*: Segue a mesma idéia da perturbação com múltiplos *Shift*, porém utilizando movimentos *Swap*;
- *Ejection chain*: Essa perturbação foi proposta por Rego e Roucairol (1996). Inicialmente, seleciona-se um subconjunto de m rotas $R = \{r_1, r_2, \dots, r_m\}$ de forma arbitrária. Em seguida, transfere-se um cliente da rota r_1 para a rota r_2 , um cliente de r_2 para r_3 e assim sucessivamente até que um cliente seja transferido da rota r_m para a primeira rota r_1 . Nesse movimento, os clientes são escolhidos de forma aleatória.

O VND explora o espaço de soluções por meio dos movimentos descritos na Subseção 3.2. Esse método possui duas estratégias: a primeira consiste em determinar aleatoriamente a ordem das vizinhanças a serem pesquisadas, enquanto que a segunda é a intensificação da busca nas rotas modificadas em cada iteração do método. Essa intensificação é realizada por meio de buscas locais baseadas nos movimentos *Shift*, *Shift(2,0)*, *Swap*, *2-Opt*, *Swap(2,1)*, *Swap(2,2)*, *kOr-Opt* com $k = 3, 4, 5$, apresentados na Subseção 3.2. Além dessas buscas locais, a intensificação é realizada por dois outros procedimentos inspirados na heurística GENIUS, denominados *G3-Opt* e *G4-Opt*, que representam adaptações das buscas locais *3-optimal* e *4-optimal*. A adaptação consiste em analisar a inserção de um arco $(v_i; v_j)$ somente se os clientes v_i e v_j estiverem relativamente próximos. Para isso, define-se $N_p(v)$ como o conjunto dos p vizinhos mais próximos ao cliente v em uma rota r da solução s , sendo p um parâmetro. Além disso, considere as seguintes definições: N^r , conjunto dos clientes pertencentes à rota r ; v_i : cliente $v_i \in N^r$; v_{h+1} ; v_{h-1} : clientes, pertencentes à rota r , sucessor e antecessor ao cliente $v_h \in N^r$, respectivamente; v_j : cliente $v_j \in N_p(v_i)$; v_k : cliente $v_k \in N_p(v_{i+1})$

no caminho de v_j para v_i ; v_l : cliente $v_l \in Np(v_{j+1})$ no caminho de v_i para v_j . O procedimento *G3-Opt* funciona da seguinte forma: a cada passo, é feita a remoção dos arcos $(v_i; v_{i+1})$, $(v_j; v_{j+1})$ e $(v_k; v_{k+1})$ e a inserção dos arcos $(v_i; v_j)$, $(v_{i+1}; v_k)$ e $(v_{j+1}; v_{k+1})$ na rota r , de forma a melhorar a solução s e tal que o custo seja o menor possível. Ressalta-se que ambos os sentidos da rota r são analisados. Este procedimento é repetido até que não seja possível melhorar a solução s . Já o procedimento *G4-Opt* é semelhante ao *G3-Opt*, com a diferença de que, a cada iteração, são removidos os arcos $(v_i; v_{i+1})$, $(v_{l-1}; v_l)$, $(v_j; v_{j+1})$ e $(v_{k-1}; v_k)$ e adicionados os arcos $(v_i; v_j)$, $(v_l; v_{j+1})$, $(v_{k-1}; v_{l-1})$ e $(v_{i+1}; v_k)$. Por fim, é realizado o movimento *Reverse* nas rotas modificadas, que consiste em inverter o sentido da rota, caso haja redução na carga máxima.

4. RESULTADOS COMPUTACIONAIS

Apresentam-se, nesta Seção, os resultados computacionais obtidos pelo algoritmo heurístico híbrido proposto para resolver o PRVCEs. O sistema foi desenvolvido na linguagem C++, utilizando o ambiente Microsoft Visual C++, versão 2005. Foi utilizado um computador Intel Core 2 Duo com 1,66 GHz e 2 GB de memória RAM e sistema operacional Windows Vista Home Premium de 32 bits.

Para validar o algoritmo, foram utilizados três conjuntos de problemas-teste apresentados na Seção 1, a saber, os de Salhi e Nagy (1999), Dethloff (2001) e Montané e Galvão (2006). No conjunto de Salhi e Nagy (1999), não foram tratados os problemas que possuem restrições de limite de tempo. O número máximo de iterações do GENILS utilizado foi 10.000.

As Tabelas 1, 2 e 3 comparam o desempenho do GENILS com diferentes algoritmos da literatura. Nessas tabelas, a coluna *Problema* indica o problema-teste considerado, *Melhor* é o melhor valor encontrado pelo algoritmo do respectivo autor e *Tempo* é o tempo, em segundos, de processamento do algoritmo. Na coluna *Gap*, mostra-se o desvio percentual das soluções médias do GENILS em relação aos melhores resultados existentes. O *Gap* é calculado pela expressão $Gap = 100 \times (Média - MelhorValor) / MelhorValor$ (coluna *Gap*).

Tabela 1: Resultados obtidos pelo GENILS nos problemas-teste de Dethloff (2001)

Problema	Röpke e Pisinger		Zachariadis <i>et al.</i>		Subramanian <i>et al.</i>		GENILS		
	Melhor	Tempo ⁽¹⁾ (s)	Melhor	Tempo ⁽²⁾ (s)	Melhor	Tempo ⁽³⁾ (s)	Melhor	Tempo ⁽⁴⁾ (s)	Gap (%)
SCA3-0	636,10	232,00	636,06	2,83	635,62	0,90	635,62	6,77	0,00
SCA3-1	697,80	170,00	697,84	2,12	697,84	1,12	697,84	8,49	0,00
SCA3-2	659,30	160,00	659,34	2,58	659,34	1,19	659,34	8,13	0,00
SCA3-3	680,60	182,00	680,04	3,13	680,04	1,13	680,04	8,45	0,00
SCA3-4	690,50	160,00	690,50	2,68	690,50	1,32	690,50	8,09	0,00
SCA3-5	659,90	178,00	659,90	2,56	659,90	1,17	659,90	8,19	0,00
SCA3-6	651,10	171,00	651,09	4,40	651,09	1,23	651,09	8,21	0,00
SCA3-7	666,10	162,00	659,17	2,98	659,17	1,69	659,17	6,76	0,00
SCA3-8	719,50	157,00	719,47	3,98	719,47	1,08	719,48	8,85	0,00
SCA3-9	681,00	167,00	681,00	3,86	681,00	1,03	681,00	8,63	0,00
SCA8-0	975,10	98,00	961,50	3,21	961,50	2,52	961,50	5,65	0,00
SCA8-1	1052,40	95,00	1050,20	3,55	1049,65	2,98	1049,65	5,67	0,00
SCA8-2	1039,60	83,00	1039,64	4,67	1039,64	3,42	1039,64	5,92	0,00
SCA8-3	991,10	94,00	983,34	3,29	983,34	3,44	983,34	4,58	0,00
SCA8-4	1065,50	84,00	1065,49	2,68	1065,49	2,74	1065,49	5,98	0,00

SCA8-5	1027,10	96,00	1027,08	4,50	1027,08	3,44	1027,08	6,62	0,00
SCA8-6	972,50	93,00	971,82	2,67	971,82	2,48	971,82	6,57	0,00
SCA8-7	1061,00	92,00	1052,17	4,32	1051,28	5,39	1051,28	5,56	0,00
SCA8-8	1071,20	85,00	1071,18	3,43	1071,18	2,05	1071,18	5,57	0,00
SCA8-9	1060,50	86,00	1060,50	4,12	1060,50	3,10	1060,50	5,62	0,00
CON3-0	616,50	171,00	616,52	3,89	616,52	2,02	616,52	6,77	0,00
CON3-1	554,50	190,00	554,47	2,97	554,47	1,83	554,47	7,76	0,00
CON3-2	521,40	176,00	519,26	3,32	518,00	2,10	518,01	9,28	0,00
CON3-3	591,20	177,00	591,19	2,78	591,19	1,34	591,19	9,18	0,00
CON3-4	588,80	173,00	589,32	3,12	588,79	1,79	588,79	6,29	0,00
CON3-5	563,70	179,00	563,70	3,45	563,70	1,71	563,70	9,16	0,00
CON3-6	499,10	195,00	500,80	2,98	499,05	1,93	499,05	7,33	0,00
CON3-7	576,50	226,00	576,48	2,40	576,48	1,52	576,48	6,96	0,00
CON3-8	523,10	174,00	523,05	5,02	523,05	1,51	523,05	8,75	0,00
CON3-9	578,20	163,00	580,05	3,14	578,24	1,58	578,25	6,87	0,00
CON8-0	857,20	86,00	857,17	3,40	857,17	3,74	857,17	6,36	0,00
CON8-1	740,90	81,00	740,85	3,73	740,85	2,82	740,85	4,88	0,00
CON8-2	716,00	84,00	713,14	2,87	712,89	2,46	712,89	6,95	0,00
CON8-3	811,10	91,00	811,07	3,82	811,07	2,82	811,07	5,87	0,00
CON8-4	772,30	87,00	772,25	2,98	772,25	3,37	772,25	5,01	0,00
CON8-5	755,70	94,00	756,91	5,76	754,88	3,30	754,88	5,82	0,00
CON8-6	693,10	96,00	678,92	4,00	678,92	3,04	678,92	5,67	0,00
CON8-7	814,80	94,00	811,96	2,46	811,96	2,73	811,96	4,71	0,00
CON8-8	774,00	94,00	767,53	4,21	767,53	3,42	767,53	5,23	0,00
CON8-9	809,30	92,00	809,00	3,87	809,00	3,60	809,00	5,86	0,00

⁽¹⁾ Tempo de CPU em um computador Pentium IV 1.5 GHz.

⁽²⁾ Tempo de CPU em um computador Pentium IV 2.4 GHz.

⁽³⁾ Tempo de CPU em um computador Intel Core 2 Quad 2.5 GHz.

⁽⁴⁾ Tempo de CPU em um computador Intel Core 2 Duo 1,6 GHz.

Tabela 2: Resultados obtidos pelo GENILS nos problemas-teste de Salhi e Nagy (1999)

Problema	Wassan <i>et al.</i>		Zachariadis <i>et al.</i>		Subramanian <i>et al.</i>		GENILS		
	Melhor	Tempo ⁽¹⁾ (s)	Melhor	Tempo ⁽²⁾ (s)	Melhor	Tempo ⁽³⁾ (s)	Melhor	Tempo ⁽⁴⁾ (s)	Gap (%)
CMT1X	468,30	48	469,80	2,89	466,77	1,10	466,77	7,82	0,00
CMT1Y	458,96	69	469,80	3,85	466,77	1,08	466,77	7,61	1,68
CMT2X	668,77	94	684,21	7,42	684,21	6,99	684,21	17,62	2,31
CMT2Y	663,25	102	684,21	8,02	684,21	5,84	684,21	20,10	3,16
CMX3X	729,63	294	721,27	11,62	721,40	6,80	721,40	59,61	0,02
CMT3Y	745,46	285	721,27	13,53	721,40	7,37	721,27	58,72	0,00
CMT12X	644,70	242	662,22	11,80	662,22	8,02	662,22	22,89	2,72
CMT12Y	659,52	254	662,22	7,59	662,22	7,32	663,50	22,33	0,60
CMT11X	861,97	504	838,66	17,78	839,39	12,58	846,23	48,85	0,90
CMT11Y	830,39	325	837,08	14,26	841,88	14,80	836,04	287,30	0,68
CMT4X	876,50	558	852,46	27,75	852,46	50,72	852,46	134,26	0,00
CMT4Y	870,44	405	852,46 ^a	31,20	852,46	46,06	862,28	266,76	1,17 ^b
CMT5X	1044,51	483	1030,55	51,67	1030,55	53,51	1033,51	768,94	0,29
CMT5Y	1054,46	533	1030,55	58,81	1031,17	58,74	1036,14	398,75	0,54

⁽¹⁾ Tempo de CPU em um computador Sun-Fire-V440 com um processador UltraSPARC-IIIi 1062 MHz.;

⁽²⁾ Tempo de CPU em um computador Pentium IV 2,4 GHz.;

⁽³⁾ Tempo de CPU em um computador Intel Core 2 Quad 2,5 GHz.;

⁽⁴⁾ Tempo de CPU em um computador Intel Core 2 Duo 1,6 GHz.

^a Um resultado melhor de valor 852,35 foi obtido por Chen e Wu (2006).

^b Gap em relação ao valor encontrado por Chen e Wu (2006).

Tabela 3: Resultados obtidos pelo GENILS nos problemas-teste de Montané e Galvão (2006)

Problema	Montané e Galvão		Zachariadis <i>et al.</i>		Subramanian <i>et al.</i>		GENILS		
	Melhor	Tempo ⁽¹⁾ (s)	Melhor	Tempo ⁽²⁾ (s)	Melhor	Tempo ⁽³⁾ (s)	Melhor	Tempo ⁽⁴⁾ (s)	Gap (%)

r101	1042,62	12,20	1019,48	10,50	1010,90	10,51	1009,95	35,65	-0,09
r201	671,03	12,02	666,20	8,70	666,20	6,24	666,20	39,62	0,00
c101	1259,79	12,07	1220,99	10,20	1220,26	12,73	1220,18	18,34	-0,01
c201	666,01	12,40	662,07	5,70	662,07	4,18	662,07	16,62	0,00
rc101	1094,15	12,30	1059,32	12,90	1059,32	9,48	1059,32	12,79	0,00
rc201	674,46	12,07	672,92	10,50	672,92	4,21	672,92	24,03	0,00
r1_2_1	3447,20	55,56	3393,31	61,80	3371,29	95,79	3357,64	175,81	-0,40
r2_2_1	1690,67	50,95	1673,65	47,40	1665,58	24,13	1665,58	103,44	0,00
c1_2_1	3792,62	52,21	3652,76	66,30	3640,20	95,17	3636,74	117,62	-0,10
c2_2_1	1767,58	65,79	1753,68	60,90	1728,14	41,94	1726,59	127,81	-0,09
rc1_2_1	3427,19	58,39	3341,25	45,30	3327,98	76,30	3312,92	299,30	-0,45
rc2_2_1	1645,94	52,93	1562,34	62,40	1560,00	34,28	1560,00	77,48	0,00
r1_4_1	10027,81	330,42	9758,77	315,30	9695,77	546,39	9627,43	2928,31	-0,71
r2_4_1	3685,26	324,44	3606,72	273,60	3574,86	231,73	3582,08	768,60	0,20
c1_4_1	11676,27	287,12	11207,37	283,50	11124,30	524,35	11098,21	1510,44	-0,23
c2_4_1	3732,00	330,20	3630,72	336,00	3575,63	293,18	3596,37	569,01	0,58
rc1_4_1	9883,31	286,66	9697,65	145,80	9602,53	550,90	9535,46	2244,18	-0,70
rc2_4_1	3603,53	328,16	3498,30	345,00	3416,61	291,15	3422,11	3306,84	0,16

⁽¹⁾ Tempo de CPU em um computador Athlon XP 2,0 GHz.

⁽²⁾ Tempo de CPU em um computador Pentium IV 2,4 GHz.

⁽³⁾ Tempo de CPU em um computador Intel Core 2 Quad 2,5 GHz.

⁽⁴⁾ Tempo de CPU em um computador Intel Core 2 Duo 1,6 GHz.

Em relação aos problemas-teste propostos por Dethloff (2001), o GENILS foi capaz de alcançar todas as melhores soluções da literatura. Dos 14 problemas-teste de Salhi e Nagy (1999), o algoritmo proposto encontrou quatro melhores soluções da literatura, enquanto que nos demais problemas, o *gap* máximo foi de 3,16%. É importante ressaltar que neste conjunto nenhum algoritmo vence na maioria das instâncias, ou seja, não existe uma dominância clara de nenhum método. O melhor desempenho do GENILS se deu nos problemas-teste de Montané e Galvão (2006), em que, dos 18 problemas desse conjunto, em 9 foram geradas novas melhores soluções, em 6 foram encontrados os melhores resultados da literatura e nos 3 restantes, o *gap* foi inferior a 0,58%.

Comparando o GENILS com outros algoritmos, verifica-se que o mesmo tem desempenho bastante próximo ao de Subramanian *et al.* (2008). De fato, tanto nos problemas-teste de Dethloff (2001) quanto nos de Montané e Galvão (2006), são esses os únicos algoritmos que têm todos os melhores resultados da literatura. Nesse segundo conjunto de problemas-teste, o GENILS foi superior ao de Subramanian *et al.* (2008) em 9 problemas e inferior em 3. Já no conjunto de problemas de Salhi e Nagy (1999), o GENILS superou o algoritmo de Subramanian *et al.* (2008) em 2 casos e teve desempenho pior em 5.

Uma comparação em termos de tempos computacionais não foi feita porque os resultados dos outros algoritmos da literatura foram obtidos em máquinas distintas.

5. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho abordou o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Para resolvê-lo, foi proposto um algoritmo heurístico híbrido, denominado GENILS. Para gerar uma solução inicial foram utilizadas adaptações do método da Inserção Mais Barata e da heurística GENIUS. Para refinar essa solução, foi utilizada a metaheurística *Iterated Local Search* (ILS), tendo o *Variable Neighborhood Descent* (VND) como método de busca local. O VND explora a vizinhança de uma solução utilizando os movimentos *Shift*, *Shift(2,0)*, *Swap*, *Swap(2,1)*, *Swap(2,2)*, *M2-Opt* e *kOr-Opt*, apresentados

na Subseção 3.2. Além disso, ele realiza uma intensificação da busca sempre que ocorre uma melhora na solução corrente. Essa intensificação é feita somente nas rotas modificadas e é realizada pelas buscas locais apresentadas na Seção 3.2, pelos procedimentos *G3-opt* e *G4-opt* (os quais são baseados na heurística GENIUS) e pelo movimento *Reverse*.

De acordo com os resultados obtidos, verifica-se que o algoritmo proposto é competitivo com as melhores abordagens da literatura, sendo capaz de produzir soluções de qualidade. De fato, em um conjunto consagrado de problemas-teste, foram alcançados todos os melhores resultados da literatura; em outro, foram gerados nove melhores resultados e seis resultados iguais aos melhores da literatura; e no terceiro, foram igualados três resultados da literatura, tendo-se um *gap* máximo igual a 3,16% para os demais problemas desse conjunto. Além disso, o GENILS obteve soluções com variabilidade inferior a 1% em 67 dos 72 problemas-teste, o que corresponde a 93% dos casos. Um comportamento interessante do algoritmo GENILS é o fato deste obter o melhor desempenho nos problemas de maior porte, os de Montané e Galvão (2006), o que mostra o seu potencial em resolver aplicações reais, onde geralmente defrontamos com problemas de elevadas dimensões.

Como trabalho futuro, pretende-se aprimorar os procedimentos *G3-opt* e *G4-opt*, baseados na heurística GENIUS, de forma a considerar a recombinação de múltiplas rotas. Além disso, é estratégico combinar o algoritmo GENILS com a metaheurística Busca Tabu, sendo esta acionada em substituição ao VND, por exemplo, após certo número de iterações do ILS. Isso se deve ao fato de que a Busca Tabu é o algoritmo base de Wassan *et al.* (2007) e Zachariadis *et al.* (2009), os quais têm a maioria dos melhores resultados dos problemas-teste de Salhi e Nagy (1999), conjunto em que o GENILS teve o pior desempenho.

Agradecimentos

Os autores agradecem a CAPES, CNPq e FAPERJ pelo apoio parcial ao desenvolvimento deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

- Anderberg, M. R. (2007) *Cluster analysis for applications*. Monographs and Textbooks on Probability and Mathematical Statistics. Academic Press, Inc., New York.
- Bean, J. C. (1994) Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154-160.
- Bianchessi, N.; Righini, G. (2007) Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2):578-594.
- Chen, J. F. (2006) Approaches for the vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Chinese Institute of Industrial Engineers*, 23(2):141-150.
- Chen, J. F.; Wu, T. H. (2006) Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society*, 57(5):579-587.
- Crispim, J.; Brandão, J. (2005) Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society*, 56(7):1296-1302.
- Dantzig, G. B.; Ramser, J. H. (1959) The truck dispatching problem. *Management Science*, 6:80-91.
- Dell'Amico, M.; Righini, G.; Salani, M. (2006) A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235-247.
- Dethloff, J. (2001) Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, 23:79-96.
- Dorigo, M.; Maniezzo, V.; Colnari, A. (1996) The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, v 26, p 29-41.
- Dueck, G. (1993) New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86-92.
- Gehring, H.; Homberger, J. (1999) A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen K, Mäkelä M, Toivanen J, editors. *Proceedings of EUROGEN99*, v A2(S), Springer, Berlin, p 57-64.
- Gendreau, M.; Hertz, A.; Laporte, G. (1992) New insertion and post optimization procedures for the traveling

- salesman problem. *Operations Research*, 40:1086-1094.
- Glover, F.; Laguna, M. (1992) *Tabu Search*, Kluwer Academic Publishers, Boston.
- Gökçe, E. I. (2004) A revised ant colony system approach to vehicle routing problems. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University.
- Goldberg, D. E. (1989) Genetic Algorithms in Search. *Optimization and Machine Learning*. Addison-Wesley, Berkeley.
- Halse, K. (1992) *Modeling and solving complex vehicle routing problems*. PhD thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Denmark.
- Hansen, P.; Mladenović, N. (2003) Variable neighborhood search: Principles and applications, *European Journal of Operations Research*, 130:449-467.
- Kirkpatrick, S.; Gellat, D.C.; Vecchi, M. P. (1983) Optimization by Simulated Annealing, *Science*, 220:671-680.
- Min, H. (1989) The multiple vehicle routing problems with simultaneous delivery and pick-up points. *Transportation Research A*, 23(5):377-386.
- Mladenović, N.; Hansen, P. (1997) Variable neighborhood search. *Computers and Operations Research*, 24:1097-1100.
- Montané, F. A. T.; Galvão, R. D. (2006) A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research*, 33(3):595-619.
- Nagy, G.; Salhi, S. (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162:126-141.
- Or, I. (1976) *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking*. PhD thesis, Northwestern University, USA.
- Parragh, S.; Doerner, K.; Hartl, R. (2008a) A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21-51.
- Parragh, S.; Doerner, K.; Hartl, R. (2008b) A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81-117.
- Rego, C.; Roucairol, C. (1996) *Meta-Heuristics Theory and Applications*, chapter A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem, p 661-675. Kluwer Academic Publishers, Boston.
- Röpke, S.; Pisinger, D. (2006) A unified heuristic for a large class of vehicle routing problems with backhauls. Technical Report 2004/14, University of Copenhagen.
- Salhi, S.; Nagy, G. (1999) A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50:1034-1042.
- Shaw, P. (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: *CP-98 Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, p 417-431, London.
- Solomon, M. M.; Joachim, I.; Desrosiers, J.; Dumas, Y.; Villeneuve, D. (1995) A request clustering algorithm for door-to-door handicapped transportation, *Transportation Science*, 29:63-78.
- Stützel, T.; Hoos, H. H. (1999) Analyzing the run-time behaviour of iterated local search for the tsp. In *Proceedings of the Third Metaheuristics International Conference*, p 449-453, Angra dos Reis, Rio de Janeiro.
- Subramanian, A. (2008) *Metaheurística Iterated Local Search aplicada ao problema de roteamento de veículos com coleta e entrega simultânea*. Dissertação de mestrado, Universidade Federal da Paraíba, João Pessoa.
- Subramanian, A.; Cabral, L. A. F.; Ochi, L. S. (2008) An efficient ILS heuristic for the vehicle routing problem with simultaneous pickup and delivery. Relatório Técnico, Universidade Federal Fluminense, disponível em <http://www.ic.uff.br/~satoru/index.php?id=2>.
- Topcuoglu, H. and Sevilmiş, C. (2002) Task scheduling with conflicting objectives. In Yakhno, T. M., editor, ADVIS, volume 2457 of *Lecture Notes in Computer Science*, p 346-355. Springer.
- Voudouris, C.; Tsang, E. (1996) Partial constraint satisfaction problems and guided local search. In *In The Second Int. Conference on the Practical Application of Constraint Technology (PACT'96)*, p 337-356.
- Vural, A. V. A (2003) GA based meta-heuristic for capacited vehicle routing problem with simultaneous pick-up and deliveries. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University.
- Wassan, N. A.; Wassan, A. H.; Nagy, G. (2007) A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 15(4):368-386.
- Zachariadis, E. E.; Tarantilis, C. D.; Kiranoudis, C. T. (2009) A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, 36(2):1070-1081.

Marcio Tadayuki Mine (mmine@ic.uff.br)
Matheus de Souza Alves Silva (msalves@ic.uff.br)
Luiz Satoru Ochi (satoru@ic.uff.br)
Instituto de Computação, Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 3º Andar, CEP 24210-240 – Niterói, Rio de Janeiro, Brasil

Marcone Jamilson Freitas Souza (marcone@iceb.ufop.br)
Departamento de Computação, Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro, CEP 35400-000 – Ouro Preto, Minas Gerais, Brasil