Universidade Federal de Ouro Preto - UFOP
Escola de Minas
Departamento de Engenharia de Produção,
Administração e Economia – DEPRO
Campus Morro do Cruzeiro
Ouro Preto – Minas Gerais - Brasil

Redução de custos da programação diária de tripulações de ônibus urbano via metaheurísticas

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Sílvia Maria Santana Mapa

Ouro Preto
Dezembro de 2004

Sílvia Maria Santana Mapa

Redução de custos da programação diária de tripulações de ônibus urbano via metaheurísticas

Monografia apresentada ao Curso de Engenharia de Produção da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção de Grau de Engenheira de Produção

Orientador: Marcone Jamilson Freitas Souza
Co-orientador: Gustavo Peixoto Silva

Ouro Preto
Dezembro de 2004

Monografia defendida e aprovada, em 20 de Dezembro de 2004, pela comissão avaliadora constituída pelos professores:

_____

Prof. Gustavo Peixoto Silva, Dr.

_____

Prof. Marcone Jamilson Freitas Souza, Dr.

_____

Prof. Alexandre Xavier Martins, Msc.

"O homem se torna muitas vezes o que ele próprio acredita que é.
Se eu insisto em repetir para mim mesmo que não sou capaz de realizar
alguma coisa, é possível que realmente seja incapaz de fazê-la.
Ao contrário, se tenho a convicção de que posso fazê-la, certamente adquirirei
capacidade de realizá-la, mesmo que não a tenha no começo".

(Gandhi)

Aos meus pais, Luiz e Sônia, pelo exemplo de vida e apoio na realização de meus sonhos.

Ao Albano, pelo amor, carinho e compreensão nos momentos de ausência.

Aos meus irmãos, Michelle e Danilo, que torcem pelo meu sucesso.

Meus sinceros agradecimentos:

A Deus, por estar sempre presente ao meu lado.

Aos mestres Marcone e Gustavo, por tudo que me ensinaram e pela confiança depositada, o que fez brotar a segurança de que eu seria capaz.

Aos meus companheiros de projeto de pesquisa, Margarida e Leonardo XT, pela oportunidade de compartilhamento de conhecimentos e experiências e pelo ambiente agradável e descontraído proporcionado, fundamental para que esta pesquisa se tornasse realidade.

A todos meus amigos, pelos bons momentos compartilhados.

À direção da UFOP e aos funcionários.

A todos aqueles que, direta ou indiretamente, contribuíram para a concretização deste trabalho.

RESUMO

Este trabalho aborda o Problema de Programação de Tripulações (PPT) no Sistema de Transporte Público. Tal problema consiste em atribuir um conjunto de tarefas aos tripulantes de uma dada empresa de forma que todas as viagens das linhas sob responsabilidade desta sejam executadas com o menor custo possível. A solução do PPT é um conjunto de jornadas diárias de trabalho de tripulantes. Neste trabalho, o PPT foi abordado utilizando as metaheurísticas Método de Pesquisa em Vizinhança Variável (VNS), *Simulated Annealing* (SA) e Busca Tabu (BT). Esses métodos exploram o espaço de soluções utilizando diferentes estruturas de vizinhança, as quais modificam as jornadas de trabalho através de operações de troca realizadas com suas tarefas. Cada solução gerada pelos métodos é avaliada por uma função baseada em penalidades que visa atender a legislação trabalhista, as regras operacionais da empresa, assim como melhorar o aproveitamento da mão-de-obra operacional. Os algoritmos foram testados com dados reais de uma empresa que opera na cidade de Belo Horizonte.

Palavras-chave: Programação de Tripulações, Método de Pesquisa em Vizinhança Variável, *Simulated Annealing*, Busca Tabu

ABSTRACT

This work deals with the Bus Crew Scheduling Problem (BCSP) related to a company operating in a public mass transit. Such problem consists in assigning the set of all vehicle trips of a given company to a set of drivers with minimal cost. The solution of BCSP is a set of driver duties. In this work the BCSP was solved through the Variable Neighborhood Search, Simulated Annealing and Tabu Search metaheuristics. These methods explore the solution space using different neighborhoods, which modify the driver duties through tasks' movements. Each schedule is evaluated by a function based on penalties that has as goal to satisfy the labor agreement rules, the operational rules of the company, as well as to optimize the use of the crew at work. The algorithms were tested with real data provided by a company operating in Belo Horizonte city.

Key-words: Bus Crew Scheduling, Variable Neighborhood Search, Simulated Annealing, Tabu Search

SUMÁRIO

LISTA DE SIGLAS

BHTRANS:	Empresa de Transporte e Trânsito de Belo Horizonte S/A	
BT:	Busca Tabu	
GRASP:	Procedimento de busca adaptativa gulosa e randomizada	*Greedy Randomized Adaptive Search Procedure*
NP-difícil:	Não polinomial difícil	
OT:	Oportunidade de Troca	
PPT:	Problema da Programação de Tripulações	
SA:	Recozimento Simulado	*Simulated Annealing*
SETRABH:	Sindicato das Empresas de Transporte de Passageiros de Belo Horizonte	
STTRBH:	Sindicato dos Trabalhadores em Transporte Rodoviário de Belo Horizonte	
TS:	Busca Tabu	*Tabu Search*
UFOP:	Universidade Federal de Ouro Preto	
VND:	Descida em Vizinhança Variável	*Variable Neighborhood Descent*
VNS:	Pesquisa em Vizinhança Variável	*Variable Neighborhood Search*

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

O problema do planejamento de uma rede de transportes, devido à sua grande complexidade, é normalmente decomposto em inúmeros subproblemas, conforme esquematizado na Figura 1.1 a seguir:

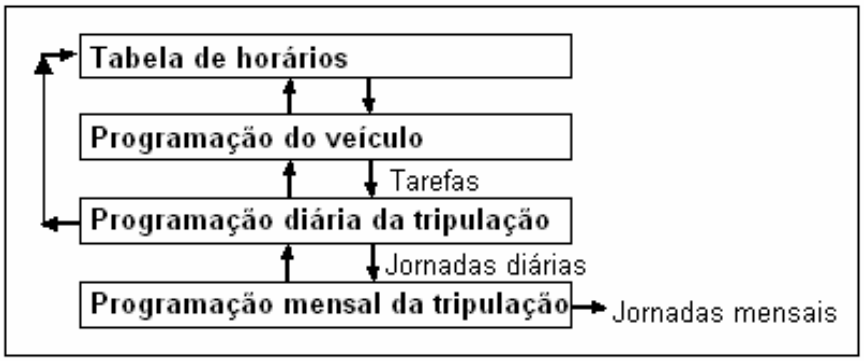


Figura 1.1 - Planejamento de Transporte

As diferentes etapas no processo de planejamento no transporte público são totalmente interdependentes. Assim, modificações em partes do sistema alteram significativamente o sistema global.

Na cidade de Belo Horizonte, a BHTRANS é o órgão responsável pelo planejamento do transporte público coletivo. Compete a ele a análise da demanda e a elaboração da tabela de ônibus, ou seja, o quadro de horários. A partir daí, empresas privadas filiadas à BHTRANS são os órgãos responsáveis pela prestação do serviço. Tais empresas, ao receberem o quadro de horário, fazem a alocação dos veículos, bem como da mão-de-obra disponível, os tripulantes, aqui definidos como motoristas e trocadores.

Na realidade brasileira, o que se observa é uma grande dificuldade na geração das escalas de trabalho, devido às inúmeras restrições existentes, sejam elas trabalhistas ou operacionais, e uma carência de ferramentas computacionais no setor.

A etapa na qual o presente trabalho se enquadra é a programação diária de tripulações, apresentando como solução as jornadas diárias de cada tripulação, que estão submetidas a um conjunto de leis específicas de cada organização. Estas são leis trabalhistas nacionais, regionais ou mesmo internas de organizações que acabam por gerar

restrições à programação. A natureza da lei tem um profundo efeito computacional, portanto estas estão submetidas a diferentes pesos, o que permite a distinção entre fatores proibitivos e outros indesejáveis.

O Problema de Programação de Tripulações de uma empresa do Sistema de Transporte Público consiste em gerar uma escala de trabalho, isto é, um conjunto de jornadas de trabalho para as tripulações que conduzirão a frota de ônibus em operação. Tais jornadas devem contemplar todas as viagens sob responsabilidade da empresa, satisfazer a um conjunto de leis trabalhistas e regras operacionais desta, com o menor custo possível. Este problema tem como dados de entrada a programação dos veículos. Portanto, considera-se que já estejam definidos os blocos de viagens de cada veículo, isto é, a seqüência das viagens de cada ônibus. O objetivo do PPT é determinar o número mínimo necessário de tripulações tal que a programação dos veículos seja realizada com sucesso.

A solução deste problema é uma tarefa complexa por envolver um grande número de possibilidades de solução. As escalas geradas manualmente têm como principal objetivo satisfazer as leis e regras operacionais, sem, no entanto, procurar reduzir os custos envolvidos. Esta é a situação em que se encontra a maioria das empresas nacionais, o que ressalta a importância de desenvolver técnicas de otimização voltadas para a resolução deste problema.

Na literatura, o PPT é classificado como um problema Não Polinomial-Difícil (NP_Difícil), o que torna inviável sua solução por métodos exatos, pois não existem algoritmos que o resolva em tempo polinomial, constituindo um problema de grande porte. Dessa forma, o PPT é normalmente abordado através de técnicas heurísticas. Dentre essas técnicas, destacam-se as metaheurísticas, as quais, ao contrário das heurísticas convencionais, são providas de mecanismos para escapar de ótimos locais, ou seja, de soluções que ainda podem ser melhoradas.

Este trabalho apresenta a formulação e implementação de algoritmos baseados nas técnicas *Simulated Annealing* (SA – Recozimento Simulado), *Variable Neighborhood Search* (VNS - Método de Pesquisa em Vizinhança Variável), Tabu Search (BT - Busca Tabu) e Greedy Randomized and Adaptative Search Procedure (GRASP - Procedimento de busca adaptativa gulosa e randomizada).

1.1. ORIGEM DO TRABALHO

O presente trabalho tem sua origem no campo de atuação da Engenharia de Produção, mais especificamente na área de Pesquisa Operacional, que dispõe de ferramentas que facilitam na tomada de decisões gerenciais. O tema do trabalho foi proposto ao longo do curso, a partir da necessidade, observada por professores envolvidos na área, de implementação de um sistema computacional para resolução do problema da programação de tripulações de ônibus no sistema de transporte público. Esta necessidade despertou o interesse do presente pesquisador, surgindo a oportunidade de participar do Programa Institucional de Bolsas de Iniciação Científica (PIBIC/UFOP), apoiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), com o objetivo de conhecer e desenvolver uma proposta de solução para o problema em questão.

1.2. IMPORTÂNCIA DO TRABALHO

O principal benefício que o presente trabalho vem a oferecer, no âmbito da Engenharia de Produção, é a partir do momento em que, no ato do planejamento gerencial, surgem dúvidas do tipo: Como agendar as escalas de trabalho diária de motoristas e cobradores de ônibus? Como garantir que todas as escalas atendam às normas operacionais e trabalhistas vigentes na empresa? Como programar, obedecendo às regras, ao menor custo possível?

É a partir de questões como estas que se justifica a importância do referido tema, aplicado às empresas do Sistema de Transporte Público. Estas empresas têm encontrado, cada vez mais, desafios na área de planejamento. Tais desafios são causados principalmente pelo crescimento da população urbana, alterações nas políticas de gestão de transporte e leis que regem os acordos entre empresas, governos e empregados. Além disso, há uma exigência constante na qualidade do serviço a ser prestado.

Várias técnicas podem ser utilizadas na elaboração de um plano que define as escalas de trabalho dos tripulantes. Algumas delas procuram soluções otimizadas, como é o caso do presente trabalho; e outras que se aproveitam da experiência e do bom senso

dos planejadores, situação na qual se enquadra a empresa de transportes da cidade de Belo Horizonte, objeto alvo desta pesquisa. Nos dois casos, o objetivo é gerar uma solução que atenda às estratégias da empresa empregando menor custo com a mão-de-obra, principal gasto de toda a operação.

Pode-se ver que, na prática brasileira, as técnicas informais de tentativa e erro são as mais empregadas, sendo que as empresas encontram grande dificuldade na elaboração manual das escalas de trabalho dos tripulantes, exigindo dias de trabalho. Talvez a razão deste procedimento seja a falta de conhecimento de técnicas matemáticas que empregam modelos matemáticos capazes de administrar um grande número de variáveis e tratar adequadamente questões político-estratégicas da empresa. É dentro deste contexto que o presente trabalho se dispõe a apresentar um modelo matemático baseado em metaheurísticas, ou seja, um procedimento modelado de forma específica para resolução do problema em questão. Por sua vez, metaheurísticas são técnicas computacionais que buscam alcançar uma solução de boa qualidade, sem estar capacitada a garantir a otimalidade dessa solução, requerendo para tanto, um esforço computacional considerado razoável. As metaheurísticas vão sendo construídas baseadas na estrutura do problema, de forma a guiar a uma busca eficiente, sendo capazes de gerar boas soluções em um tempo de processamento relativamente curto.

Por envolver um grande número de possibilidades de solução, o Problema da Programação de Tripulações, ou simplesmente PPT, é um problema complexo, impossível de ser resolvido por modelos exatos para sistemas de grande porte. Tal fato justifica sua abordagem através de técnicas heurísticas.

1.3. OBJETIVOS

Neste trabalho serão apresentadas várias propostas de soluções para o Problema de Programação de Tripulações de ônibus urbano, ou seja, diversas escalas diárias de motoristas. Dentre as escalas geradas, será escolhida aquela que apresentar menores custos operacionais para a empresa, além de estar em conformidade com as leis e regras operacionais vigentes, ou seja, ser uma escala viável. Os custos envolvidos estão associados, principalmente, ao número de tripulações necessárias para operar a frota de

veículos e respectiva utilização de horas extras, que deverão ser minimizadas. A ferramenta usada para a resolução desse problema está no campo de atuação da pesquisa operacional, possibilitando o uso de várias heurísticas para geração das melhores escalas diárias de motoristas e cobradores de ônibus.

1.3.1. OBJETIVO GERAL

O objetivo geral é o desenvolvimento de metaheurísticas eficientes capazes de gerar soluções de boa qualidade e com menor custo para o Problema da Programação Diária de Tripulações de ônibus urbano de uma empresa de transporte público que opera na cidade de Belo Horizonte.

1.3.2. OBJETIVOS ESPECÍFICOS

A partir de um levantamento bibliográfico sobre o tema, será abordada uma fundamentação teórica para tratar o problema, identificando as lacunas existentes e os principais entraves metodológicos. O objetivo é a determinação do estado da arte, ou seja, procurar mostrar através de literatura já publicada, o que já se sabe sobre o tema e quais técnicas estão sendo utilizadas atualmente.

Na seqüência, será apresenta a metodologia do problema. Primeiramente será feita a descrição do problema abordado e posteriormente será apresentada sua modelagem. Na descrição do problema de programação diária de tripulações o objetivo é descrever as principais características da empresa, bem como suas regras operacionais, que acabarão por constituir restrições para modelagem do problema.

O resultado final que o presente trabalho visa alcançar é, após ter sido desenvolvido e implementado um modelo heurístico, realizar os testes de validação da solução proposta. Uma vez validadas, basta interpretar os resultados obtidos e compará-los

com a solução em uso pela empresa. O objetivo principal é constatar que a solução obtida com o uso de metaheurísticas é mais viável economicamente e ainda produzida em tempo relativamente inferior se comparada àquela gerada pela empresa.

1.4. LIMITAÇÕES DO TRABALHO

Como o próprio título do trabalho sugere, far-se-á a programação de tripulações de ônibus diária, ficando como proposta para trabalhos futuros a implementação da programação mensal e, com maior audácia, a integração com a programação de veículos.

Além disso, o presente trabalho retrata um estudo de caso de uma empresa de Belo Horizonte. Logo, as regras às quais o problema será submetido são específicas desta organização. Acrescentando-se às regras, tem-se também a legislação trabalhista, que sofre constantes alterações. Caso a parte, o problema foi tratado segundo a Convenção Coletiva de Trabalho 2002/2003.

Outro entrave relativo ao trabalho está no estudo de caso proposto. A empresa de ônibus que opera no sistema de transporte público em Belo Horizonte não permitiu a divulgação de seu nome, que será omitido através de uma descrição genérica para que o sigilo seja mantido. Este fato impossibilitou uma descrição mais detalhada da empresa.

Outra limitação a ser citada é referente à implantação do sistema desenvolvido na empresa, que então se responsabilizará pela análise e posterior decisão. Para tanto, faz-se necessário o desenvolvimento de uma interface computacional de alta qualidade para garantir a valorização do sistema e a possibilidade de seu eventual uso comercial.

1.5. ESTRUTURA DO TRABALHO

Este trabalho será disposto segundo capítulos. No presente capítulo é feita uma introdução ao trabalho, relatando suas origens e importância, assim como objetivos e limitações, justificando-se o uso de metaheurísticas para abordagem do mesmo.

No segundo capítulo, será feita uma revisão bibliográfica, descrevendo-se o estado da arte e caracterizando o presente trabalho como uma continuação de uma linha de pesquisa já em andamento. Apontam-se quais sistemas típicos já foram aplicados ao PPT em um cenário mundial. Dentre eles, destaca-se as heurísticas e respectivos autores que as propuseram.

No terceiro capítulo será descrita a metodologia empregada para solucionar o problema. O PPT será descrito, modelado e formulado, apresentando-se as restrições operacionais e trabalhistas às quais será submetido e suas respectivas penalizações, visando futura construção de uma função de avaliação, a qual medirá o grau de qualidade da solução gerada. Este capítulo apresentará também as técnicas utilizadas para geração de uma solução inicial, a partir da qual serão submetidas heurísticas para posterior refinamento.

Durante o quarto capítulo será destacada as metaheurísticas aplicadas ao problema de programação das tripulações, com as respectivas estruturas de vizinhança utilizadas, bem como alguns parâmetros de entrada utilizados.

O quinto capítulo será a aplicação do modelo proposto para o estudo de caso, levantando-se uma breve descrição da realidade da empresa. Serão apresentadas as características das escalas de trabalho diárias, geradas pelas heurísticas, e posterior comparação com a solução em uso na empresa. Neste capítulo estarão dispostos também os parâmetros utilizados na programação e os resultados computacionais gerados.

No sexto capítulo serão feitas as conclusões e recomendações para o presente trabalho e perspectivas para trabalhos futuros. Este capítulo destacará os resultados esperados partindo-se dos objetivos esclarecidos no primeiro capítulo e as prováveis contribuições do trabalho.

O sétimo capítulo lista alguns artigos oriundos deste trabalho. O próximo e último passo é a listagem das referências bibliográficas utilizadas no corpo do projeto, dispostas segundo as normas da Associação Brasileira de Normas Técnicas (ABNT).

2. REVISÃO BIBLIOGRÁFICA

2.1. INTRODUÇÃO

Apesar do problema ser conhecido desde a década de 60, o PPT não podia ser formulado em termos matemáticos como modelos de programação linear, já que até então não se havia tecnologia disponível para solucionar esses modelos de médio e grande porte. Sendo assim, até a década de 70 as pesquisas se concentraram na elaboração de métodos heurísticos. Estas metodologias eram algumas vezes tentativas de simulação dos programadores manuais, mas na maioria das vezes acabaram por representar uma linha de partida para o desenvolvimento de novas soluções heurísticas. Os sistemas heurísticos foram os primeiros a serem utilizados na resolução do PPT (ELIAS 1964). Os primeiros sistemas desenvolvidos consistiam, no entanto, apenas na automação do trabalho antes realizado manualmente. A principal desvantagem de tais sistemas é a sua incapacidade de detectar possibilidades de otimização.

Os modelos de fluxo em rede, embora menos eficazes na representação do PPT, também têm sido explorados para resolvê-lo. Carraresi & Gallo (1984) descrevem formulações para o problema que se baseiam no modelo de emparelhamento. Este modelo pode ser usado iterativamente, resolvendo-se o problema diário, semanal ou mensal (SIQUEIRA 1999). Ball & Benoit-Thompson (1988) abordam o PPT encontrando um conjunto de caminhos que representam as jornadas de trabalho em um grafo. Neste grafo, os nós são as oportunidades de troca e os arcos são de dois tipos: pedaços de jornadas e conexões factíveis entre tarefas. Esta abordagem adota a relaxação lagrangeana como forma de incluir restrições adicionais ao modelo de fluxo em rede.

O Problema de Programação da Tripulação tem sido largamente estudado e aplicado nos países mais desenvolvidos. Uma abordagem de programação matemática clássica para o PPT é aquela que o formula como um problema de recobrimento ou de particionamento e utiliza a técnica de geração de colunas para resolvê-lo (SMITH & WREN 1988, DESROCHERS & SOUMIS 1989). A variedade de trabalhos que utiliza tais abordagens deriva das diferentes possibilidades de encontrar uma solução inteira a partir da solução do problema linear. As técnicas mais exploradas são as de *branch-and-bound*,

branch-and-price e a relaxação lagrangeana, que permitem formas alternativas de implementação, tendo em vista as características do problema (WREN & ROUSSEAU 1995).

Existe uma ampla bibliografia sobre construções de jornada de trabalho para funcionários de empresas de transporte, com diversas técnicas distintas. Dentre esses trabalhos pode-se citar: Wren & Wren (1995), Clement & Wren (1995), Kwan et al. (1999), Lourenço et al. (2001) e Shen & Kwan (2000, 2001). Os três primeiros aplicam algoritmos genéticos, enquanto o quarto, além de Algoritmos Genéticos, aplica também Busca Tabu. Em todas essas quatro primeiras modelagens, o PPT é formulado como um problema de recobrimento de conjuntos. Esses conjuntos representam possíveis jornadas de trabalho factíveis, isto é, em cada jornada de trabalho há respeito às regras operacionais e trabalhistas. O que diferencia cada um dos trabalhos citados é a forma de gerar o conjunto de jornadas de trabalho a ser submetido ao modelo de programação matemática baseado em recobrimento. O último artigo mencionado trabalha numa linha diferente, não vinculada ao problema de recobrimento. Nesse trabalho, as jornadas são construídas a partir de movimentos feitos sobre suas tarefas, isto é, uma jornada pode ser não factível. Os autores aplicam Busca Tabu em cada uma das várias estruturas de vizinhança desenvolvidas.

Com o surgimento das metaheurísticas tais como, Algoritmos Genéticos (Goldberg 1989), Busca Tabu (Glover 1989,1990), *Simulated Annealing* (Kirkpatrick 1983) entre outras, abriu-se um novo horizonte na resolução de problemas NP-Difíceis como o PPT. Embora tais métodos não garantam a obtenção do ótimo global, eles permitem incluir com facilidade qualquer tipo de restrição. Além de motoristas e cobradores de ônibus, algumas publicações mostram problemas semelhantes como a construção de escalas de trabalho para tripulação de trens e de aviões.

A presente iniciativa constitui uma continuidade dos trabalhos desenvolvidos pelo grupo de pesquisa em transportes da UFOP, que resultaram na publicação dos artigos Silva et al (2002a, 2002b) e Souza et al (2003a, 2003b, 2003c), que resolvem o PPT considerando as características da realidade brasileira.

2.2. ESCALAS DE TRABALHO PARA MOTORISTAS E COBRADORES DE ÔNIBUS

2.2.1. MÉTODO HASTUS

Existe um método de elaboração de escalas de trabalho de motoristas e cobradores de ônibus conhecido na literatura como método Hastus. Neste método, as escalas são construídas em três fases.

Na primeira fase, uma escala aproximada é construída usando-se um problema de Programação Linear, relaxando-se as restrições de integralidade e de factibilidade com o objetivo de reduzir o tempo computacional. Estas restrições referem-se às possibilidades de formação e de combinação de escalas, máxima duração de uma parte de escala e sobre os locais de possíveis trocas de funcionários.

Deste modo, obtém-se as primeiras escalas, que são denominadas "pedaços" de trabalho. O conjunto formado por estas escalas é denominado escala aproximada. Em uma etapa seguinte, as jornadas são divididas novamente, em partes semelhantes às primeiras escalas geradas, com o objetivo de melhorar a escala aproximada. Estas melhorias são feitas, reduzindo-se o número de motoristas e a quantidade de horas-extras que são formadas com as primeiras divisões.

A segunda fase consiste na combinação destas escalas divididas com a utilização do algoritmo do *matching* com atribuição de pesos às combinações factíveis de escalas de trabalho. A solução ótima da combinação é o *matching* de peso máximo.

Na última fase, as divisões das tabelas são reconsideradas e algumas melhorias são feitas modificando-se estas divisões visando, novamente, a redução de número de funcionários e a quantidade de horas-extras. A descrição completa deste método, os diferentes modos de aplicação e de elaboração podem ser encontrados em publicações, tais como Blais et al (1976), Lessard et al (1981) e Rousseau et al (1985).

2.2.2. OS MÉTODOS HEURÍSTICOS DE DIVISÕES SUCESSIVAS

As heurísticas de divisões sucessivas constroem as escalas de trabalho em duas fases.

Na primeira fase as divisões das escalas são feitas e combinadas, formando-se uma jornada diária de trabalho completa para a tripulação. Estas escalas combinadas devem formar dias de trabalho factíveis, com a mesma idéia utilizada no primeiro estágio do método Hastus.

Na segunda fase, as divisões das escalas são modificadas várias vezes, e o algoritmo heurístico procura formar jornadas de trabalho factíveis, melhorando as escalas de trabalho obtidas na primeira fase, com relação às quantidades de motoristas e de horas extras formadas. Algumas publicações que mostraram estes métodos heurísticos e suas aplicações com mais detalhes são: Manington & Wren (1975), Wilhelm (1975), Wren et al (1985).

2.2.3. O MÉTODO DE COBERTURA DE CONJUNTOS

Um método bastante utilizado para construir escalas de trabalho é o método de geração de colunas, usado para resolver problemas de Programação Linear que envolvem muitas variáveis. No artigo de Desrochers & Soumis (1989) o problema de cobertura de conjuntos é utilizado para elaborar as escalas de trabalho de motoristas e cobradores de ônibus. O objetivo deste método é cobrir todas as escalas com um custo mínimo. Em sua formulação, a função de avaliação minimiza os custos dos dias de trabalho, e cada variável representa um dia de trabalho factível.

2.3. ESCALAS DE TRABALHO PARA TRIPULAÇÃO FERROVIÁRIA

Semelhante ao problema das escalas de motoristas de ônibus, o problema das escalas de trabalho para a tripulação ferroviária consiste em construir as escalas de trabalho e designá-las aos funcionários de tal maneira que a tabela de demandas seja

cumprida. Maiores detalhes deste problema podem ser encontrados em Caprara et al (1997).

Inicialmente, toma-se uma tabela de serviços de trem, a ser cumprida diariamente em certos períodos de tempo. Alguns segmentos de jornada são definidos para cada funcionário. Estes segmentos definem os tempos inicial e final de uma jornada e as estações de entrada e de saída para um funcionário.

Quando um funcionário termina sua jornada, o mesmo pode retornar à estação base (início da jornada), ou deslocar-se a uma outra estação para iniciar outra jornada. Este deslocamento não é considerado como uma jornada de trabalho, e o funcionário tem um intervalo de pelo menos um dia para fazer este deslocamento e iniciar sua próxima jornada de trabalho em uma estação.

O problema consiste em encontrar um conjunto de escalas, que cubra toda a tripulação e satisfaça a demanda, com um custo mínimo. Para resolver este problema, considera-se um grafo direcionado, onde o arco (i, j) existe se, e somente se, a seqüência de trabalho (i, j) existir, ou seja, o funcionário j pode substituir o funcionário i em uma determinada estação.

Utiliza-se o modelo clássico de cobertura de vértices, com a Relaxação Lagrangeana para modelos de Programação Linear Inteira (Fisher, 1981). Este trabalho foi implementado em uma companhia férrea italiana, e os resultados desta aproximação Lagrangeana foram melhores do que os resultados heurísticos, utilizados pela companhia tratada.

2.4. ESCALA DE TRABALHO PARA TRIPULAÇÃO AÉREA

Este problema é semelhante ao de construção de escalas para tripulação ferroviária, e está apresentado com mais detalhes em Gamache et al (1999).

A resolução deste problema é feita em duas fases. Na primeira fase, formam-se pares de segmentos de vôo em dois dias consecutivos, onde a tripulação viaja e retorna à cidade base. Alguns dias isolados, que não formam pares, são separados dos demais períodos. O problema de tripulação em pares consiste em encontrar o conjunto de

pares que cobre todos os segmentos de vôo, com custo mínimo. O custo de um par de vôos é dado pela duração total deste par.

Na segunda fase, toma-se a listagem da tripulação, construindo-se a escala mensal personalizada para cada funcionário, de acordo com sua atividade. O método de geração de colunas pode ser utilizado para resolver este problema, gerando vários subproblemas NP-difíceis.

Este procedimento de solução foi testado na companhia AIR FRANCE, onde o problema pode ser dimensionado do seguinte modo: milhares de restrições, centenas de subproblemas e centenas de milhares de arcos.

Alguns testes com o programa CADET mostram que a aplicação do método de geração de colunas é mais eficiente computacionalmente para o problema da companhia AIR FRANCE, e os resultados são melhores do que os métodos heurísticos utilizados pela companhia.

2.5. DESIGNAÇÃO DE ESCALAS DE TRABALHOS PARA MOTORISTAS E COBRADORES DE ÔNIBUS.

Uma das últimas fases da construção de jornadas de trabalho de motoristas e cobradores de ônibus é a designação da jornada semanal de trabalho para um funcionário. Esta designação envolve um grande número de restrições, impostas com o objetivo de satisfazer os critérios estabelecidos pelas empresas de transporte coletivo. Por exemplo, alguns funcionários não podem ser escalados para alguns horários, como no início da manhã ou no final da noite por problemas de deslocamento da sua residência até o local de trabalho ou vice-versa.

Uma aproximação utilizada para a designação de motoristas de ônibus para as jornadas é denominada *Multi-Level bottleneck* (ou "engarrafamento" em vários níveis). Nesta aproximação, considera-se um grafo bipartido, onde um conjunto de vértices representa as jornadas, e o outro conjunto de vértices representa os funcionários.

Um peso $p(i,j)$ é atribuído para cada designação de uma escala i para um funcionário j, onde este peso representa o custo total desta designação. A duração e os horários de início e de término de cada escala devem ser considerados, pois os custos da

empresa com os funcionários dependem destes fatores. O objetivo é encontrar um peso total mínimo das designações de escalas para motoristas.

O algoritmo inicia com uma designação inicial. A partir desta designação, escolhe-se um vértice sem designação, criando-se uma cadeia alternante, cujos vértices são designados e não designados. Quando uma cadeia possui o peso total dos arcos sem designação menor do que o peso total dos arcos já designados, trocam-se as funções da designação, ou seja, os arcos que não tinham designação passam a ter, e os demais deixam de ter designação. A descrição completa deste método pode ser encontrada em Carraresi & Gallo (1984).

Este algoritmo é uma simplificação do algoritmo de emparelhamento de peso mínimo, usado somente para grafos bipartidos (Christofides, 1975).

Os resultados encontrados demonstram que este algoritmo é mais eficiente do que o algoritmo proposto por Derigs & Zimmermann (1978), mais conhecido como o método DZ. O método DZ começa com uma solução inicial, em geral não factível e, a cada iteração, o algoritmo procura reduzir as inviabilidades.

2.6. MÉTODOS DE BUSCA LOCAL

Os Métodos de busca local em problemas de otimização constituem um conjunto de técnicas baseadas na noção de vizinhança. Mais especificamente, seja S o espaço de pesquisa de um problema de otimização e f a função objetivo a minimizar. A função N, a qual depende da estrutura do problema tratado, associa a cada solução viável $s \in S$, sua vizinhança $N(S) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de s. Denomina-se movimento a modificação m que transforma uma solução s em outra, s', que esteja em sua vizinhança. Essa operação é representada por $s' \leftarrow s \oplus m$.

Em linhas gerais, uma técnica de busca local, começando de uma solução inicial s_0 (a qual pode ser obtida por alguma outra técnica ou gerada de forma aleatória), navega pelo espaço de pesquisa através de movimentos, passando iterativamente, de uma solução para outra que seja sua vizinha.

2.6.1. MÉTODO DE DESCIDA

É um método de busca local que se caracteriza por analisar todos os possíveis vizinhos de uma solução *s* em sua vizinhança *N(s)*, escolhendo, a cada passo, aquele que tem o menor valor para a função de avaliação. Nesse método, o vizinho candidato somente é aceito se ele melhorar estritamente o valor da melhor solução até então obtida. Dessa forma, o método pára tão logo um mínimo local seja encontrado.

2.6.2. MÉTODO RANDÔMICO DE DESCIDA

O método de descida requer a exploração de toda a vizinhança. Um método alternativo, que evita essa pesquisa exaustiva é o método randômico de descida. Ele consiste em analisar um vizinho qualquer escolhido aleatoriamente 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. No entanto, o método randômico de descida também fica preso no primeiro mínimo local encontrado.

2.6.3. MÉTODO RANDÔMICO NÃO ASCENDENTE (RNA)

O método randômico não ascendente (RNA) é uma variante do método randômico de descida, diferindo dele por aceitar o vizinho gerado aleatoriamente se ele for melhor ou igual à solução corrente. Esse método pára, também, após um número fixo de iterações sem melhora no valor da melhor solução produzida até então.

Por esse método, entretanto, é possível navegar pelo espaço de pesquisa por movimentos laterais (Selman, Levesque & Mitchell, 1992). Dessa forma, ele tem condições de percorrer caminhos de descida que passam por regiões planas, ou seja, se a pesquisa chega em uma dessas regiões, o método tem condições de mover-se por ela e sair, obtendo uma nova solução diferente daquela inicial da região plana.

O método RNA é, portanto, um procedimento que executa a pesquisa no espaço de soluções combinando movimentos de descida com movimentos laterais, aumentando as chances de se escapar de um mínimo local.

2.7. METAHEURÍSTICAS

As metaheurísticas são métodos de busca local, destinadas 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 (Ribeiro, 1996).

Contrariamente às heurísticas convencionais, as metaheurísticas são de caráter geral e têm condições de escapar de ótimos locais.

As metaheurísticas, assim como os métodos de busca local tradicionais, diferenciam-se entre si basicamente pelas seguintes características:

- critério de escolha de uma solução inicial;
- definição da vizinhança *N(s)* de uma solução *s*;
- critério de seleção de uma solução vizinha dentro de *N(s)*;
- critério de término;

A seguir, serão apresentadas as principais metaheurísticas referenciadas ao longo deste trabalho.

2.7.1. GRASP (GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE)

GRASP (Procedimento de busca adaptativa gulosa e randomizada) é um método iterativo, proposto por Feo & Resende (1995), que consiste de duas fases: uma fase de construção, na qual uma solução é gerada, elemento a elemento; e de 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 pela Figura 2.1 ilustra o procedimento GRASP.

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.

```
Procedimento GRASP (f(.), g(.), N(.), GRASPmax, s)
f* ← ∞;
para (Iter = 1,2,...,Graspmax) faça
   Construção (g(.),α,s);
   BuscaLocal(f(.), N(.), s);
   se (f(s) < f*) então
      s* ← s;
      f* ← f(s);
   fim-se;
fim-para;
s* ← s;
Retorne s;
fim GRASP;
```

Figura 2.1 - Algoritmo GRASP

Esse processo de seleção é baseado em uma função adaptativa gulosa $g : C \rightarrow \Re$, 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 (LCR). Esta técnica de escolha permite que diferentes soluções sejam geradas em cada iteração GRASP. Seja $\alpha \in [0, 1]$ um dado parâmetro. O pseudocódigo representado pela Figura 2.2 descreve a fase de construção GRASP.

```
procedimento Construção (g(.), α, s);
    s ← ∅;
    Inicialize o conjunto C de candidatos;
    enquanto (C ≠ ∅) faça
       t min   = min{g(t) | t ∈ C};
       t max   = max{g(t) | t ∈ C};
       LCR = { t ∈ C | g(t) ≤  tmin + α( tmax - tmin) };
       Selecione, aleatoriamente, um elemento t ∈ LRC;
       s ← s ∩ {t};
       Atualize o conjunto C de candidatos;
    fim-enquanto;
    Retorne s;
fim Construção;
```

Figura 2.2 - Fase de construção do algoritmo GRASP

Observamos que o parâmetro α controla o nível de gulosidade e aleatoriedade do procedimento Construção. 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 melhorar a solução construída. A Figura 2.3 descreve o pseudocódigo de um procedimento básico de busca local com respeito a uma certa vizinhança *N(.)* de *s*.

```
procedimento BuscaLocal(f(.),N(.), s);
1.      s*← s; {Melhor solução encontrada }
2.      V = {s' ∈ N(s) |  f(s') < f(s) };
3.      enquanto (|V| > 0) faça
4.              Selecione s ∈ V ;
5.              se (f(s) < f(s*)) então s* ←s ;
6.              V = { s'∈ N(s) |  f(s*) < f(s) } ;
7.      fim-enquanto;
8.      s ←  s*;
9.       Retorne s;
fim BuscaLocal;
```

Figura 2.3 - Fase de Busca Local de um algoritmo GRASP

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 & 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, valor de α próximo da escolha gulosa) 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 procedimento 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 & RIBEIRO 1998, 1999a, 1999b).

2.7.2. TS (TABU SEARCH)

Descrevem-se a seguir os princípios básicos da Busca Tabu (BT), técnica originada nos trabalhos de Fred Glover (GLOVER, 1986) e Pierre Hansen (HANSEN, 1986). Para um melhor detalhamento do método, far-se-á referência a Glover (1986, 1989,

1990), Glover & Laguna (1993, 1997), Glover, Taillard & Werra (1993), Hertz & Werra (1990).

A Busca Tabu é um procedimento adaptativo que utiliza uma estrutura de memória para guiar um método de descida a continuar a exploração do espaço de soluções, mesmo na ausência de movimentos de melhora, evitando que haja a formação de ciclos, isto é, o retorno a um ótimo local previamente visitado.

Mais especificamente, começando com uma solução inicial s_0, um algoritmo BT explora, a cada iteração, um subconjunto V da vizinhança $N(s)$ da solução corrente s. O membro s' de V com menor valor nessa região segundo a função $f(.)$ torna-se a nova solução corrente mesmo que s' seja pior que s, isto é, que $f(s') > f(s)$.

O critério de escolha do melhor vizinho é utilizado para escapar de um mínimo local. Esta estratégia, entretanto, pode fazer com que o algoritmo cicle, isto é, que retorne a uma solução já gerada anteriormente. De forma a evitar que isto ocorra, existe uma lista tabu T, a qual é uma lista de movimentos proibidos. A lista tabu clássica contém os movimentos reversos aos últimos $|T|$ movimentos realizados (onde $|T|$ é um parâmetro do método) e funciona como uma fila de tamanho fixo, isto é, quando um novo movimento é adicionado à lista, o mais antigo sai. Assim, na exploração do subconjunto V da vizinhança $N(s)$ da solução corrente s, ficam excluídos da busca os vizinhos s' que são obtidos de s por movimentos m que constam na lista tabu.

A lista tabu se, por um lado, reduz o risco de ciclagem (uma vez que ela garante o não retorno, por $|T|$ iterações, a uma solução já visitada anteriormente); por outro, também pode proibir movimentos para soluções que ainda não foram visitadas. Assim, existe também uma função de aspiração, que é um mecanismo que retira, sob certas circunstâncias, o status tabu de um movimento. Mais precisamente, para cada possível valor v da função objetivo existe um nível de aspiração $A(v)$: uma solução s' em V pode ser gerada se $f(s') \leq A(f(s))$, mesmo que o movimento m esteja na lista tabu. A função de aspiração A é tal que, para cada valor v da função objetivo, retorna outro valor $A(v)$, que representa o valor que o algoritmo aspira ao chegar de v. Considerando uma função objetivo de valores inteiros, um exemplo simples de aplicação desta idéia é considerar $A(f(s)) = f(s^*) - 1$ onde s^* é a melhor solução encontrada até então. Neste caso, aceita-se um movimento tabu somente se ele conduzir a um vizinho melhor que s^*.

Duas regras são normalmente utilizadas de forma a interromper o procedimento. Pela primeira, pára-se quando é atingido um certo número máximo de

iterações sem melhora no valor da melhor solução. Pela segunda, quando o valor da melhor solução chega a um limite inferior conhecido (ou próximo dele). Esse segundo critério evita a execução desnecessária do algoritmo quando uma solução ótima é encontrada ou quando uma solução é julgada suficientemente boa.

Os parâmetros principais de controle do método de Busca Tabu são a cardinalidade $|T|$ da lista tabu, a função de aspiração A, a cardinalidade do conjunto V de soluções vizinhas testadas em cada iteração e *BTmax*, ou seja, o número máximo de iterações sem melhora no valor da melhor solução.

Apresenta-se, pela Figura 2.4, o pseudocódigo de um algoritmo de Busca Tabu básico. Neste procedimento *fmin* é o valor mínimo conhecido da função f, informação essa que em alguns casos está disponível.

```
procedimento BT
1. Seja s0 solução inicial;
2. s* ← s;                {Melhor solução obtida até então}
3. Iter ← 0;              {Contador do número de iterações}
4. MelhorIter ← 0;        {Iteração mais recente que forneceu s*}
5. Seja BTmax o número máximo de iterações sem melhora em s*;
6. T ← ∅;                 {Lista Tabu}
7. Inicialize a função de aspiração A;
8. enquanto (Iter – MelhorIter ≤ BTmax) faça
9.      Iter ← Iter + 1;
10.       Seja s' ← s ⊕ m o melhor elemento de V ⊆ N (s) tal que o movimento m não
              seja tabu (m ∉ T)  ou  s' atenda a condição de aspiração ( f(s') < A(f(s)));
11.     T ← T – {movimento mais antigo} + {movimento que gerou s'};
12.     s ← s';
13.     se f(s) < f(s*) então
14.           s* ← s;
15.           MelhorIter ← Iter ;
16.     fim-se;
17.     Atualize a função de aspiração A;
18. fim-enquanto;
19. Retorne s*;
fim BT;
```

Figura 2.4 - Algoritmo de Busca Tabu

É comum em métodos de Busca Tabu incluir estratégias de intensificação, as quais têm por objetivo concentrar a pesquisa em determinadas regiões consideradas promissoras. Uma estratégia típica é retornar às soluções já visitadas para explorar sua

vizinhança de forma mais efetiva. Outra estratégia consiste em incorporar atributos das melhores soluções já encontradas durante o progresso da pesquisa e estimular componentes dessas soluções a tornar parte da solução corrente. Nesse caso, são consideradas livres no procedimento de busca local apenas as componentes não associadas às boas soluções, permanecendo as demais componentes fixas. Um critério de término, tal como um número fixo de iterações, é utilizado para encerrar o período de intensificação.

Métodos baseados em Busca Tabu incluem, também, estratégias de diversificação. O objetivo dessas estratégias, que tipicamente utilizam uma memória de longo prazo, é redirecionar a pesquisa para regiões ainda não suficientemente exploradas do espaço de soluções. Estas estratégias procuram, ao contrário das estratégias de intensificação, gerar soluções que têm atributos significativamente diferentes daqueles encontrados nas melhores soluções obtidas. A diversificação, em geral, é utilizada somente em determinadas situações, como, por exemplo, quando dada uma solução *s*, não existem movimentos *m* de melhora para ela, indicando que o algoritmo já exauriu a análise naquela região. Para escapar dessa região, a idéia é estabelecer uma penalidade *w(s; m)* para uso desses movimentos. Um número fixo de iterações sem melhora no valor da solução ótima corrente é, em geral, utilizado para acionar essas estratégias.

Métodos de Busca Tabu incluem também listas tabu dinâmicas (DAMMEYER & VOŸ 1993, SCHAEFER 1996), muitas das quais atualizadas de acordo com o progresso da pesquisa (BATTITI 1994,1996, BASTOS & RIBEIRO 1999). A grande vantagem de se usar uma lista tabu de tamanho dinâmico é que se minimiza a possibilidade de ocorrência de ciclagem.

2.7.3. SA (*SIMULATED ANNEALING*)

Simulated Annealing é uma classe de metaheurística proposta originalmente por Kirkpatrick et al (1983), sendo uma técnica de busca local probabilística, que se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos.

Esta técnica começa sua busca a partir de uma solução inicial qualquer, normalmente gerada aleatoriamente, pois este método não depende de soluções iniciais de boa qualidade. O procedimento principal consiste em um loop que gera aleatoriamente, em cada iteração, um único vizinho *s'* da solução corrente *s*.

A cada geração de um vizinho *s'* de *s*, é testada a variação Δ do valor da função objetivo, isto é, $\Delta = f(s') - f(s)$. Sendo o problema de minimização, se $\Delta < 0$, o método aceita a solução e *s'* passa a ser a nova solução corrente. Caso $\Delta \geq 0$ a solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde *T* é um parâmetro do método, chamado de temperatura, e que regula a probabilidade de aceitação de soluções com pior custo.

A temperatura *T* assume, inicialmente, um valor elevado T_0. Após um número fixo de iterações (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura), a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_n \leftarrow \alpha * T_{n-1}$, sendo $0 < \alpha < 1$. Com esse procedimento, dá-se, no início uma chance maior para escapar de mínimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de piora ($T \rightarrow 0 \Rightarrow e^{-\Delta/T} \rightarrow 0$).

O procedimento pára quando a temperatura chega a um valor próximo de zero e nenhuma solução que piore o valor da melhor solução é mais aceita, isto é, quando o sistema está estável. A solução obtida quando o sistema encontra-se nesta situação evidencia o encontro de um mínimo local. Os parâmetros de controle do procedimento são a razão de resfriamento α, o número de iterações para cada temperatura (*SAmax*) e a temperatura inicial T_0.

O pseudocódigo do algoritmo é apresentado pela Figura 2.5. Detalhes adicionais desse algoritmo podem ser encontrados em Dowsland (1993).

```
procedimento SA
 1. Seja s0 uma solução inicial, T0 a temperatura inicial, α a taxa de resfriamento
    e SAmax o número máximo de iterações para se atingir o equilíbrio térmico;
 2. s ← s0;              {Solução corrente}
 3. s' ← s;              {Melhor solução obtida até então}
 4. T ← T0;              {Temperatura corrente}
 5. IterT ← 0;           {Número de iterações na temperatura T}
 6. enquanto (T > 0) faça
 7.    enquanto (IterT < SAmax) faça
 8.       IterT ← IterT + 1;
 9.       Gere um vizinho qualquer s' ∈ N (s);
10.       Δ = f(s') – f(s);
11.       se (Δ < 0) então
12.             s ← s';
13.             se f(s') < f(s*) então s* ← s';
14.             senão
15.                   Tome x ∈ [0,1];
16.              se x < e^(-Δ/T) então s ← s';
17.       fim-se;
18.    fim-enquanto;
19.    T ← α ×T;
20.    IterT ← 0;
21. fim-enquanto;
22. Retorne s*;
fim SA;
```

Figura 2.5 - Algoritmo de Simulated Annealing

2.7.4. VNS (VARIABLE NEIGHBORHOOD SEARCH)

O VNS (Método de Pesquisa em Vizinhança Variável) é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Contrariamente a outras metaheurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno de uma nova solução se, e somente se, um movimento de melhora é realizado. O método inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança. O pseudocódigo do algoritmo é apresentado pela Figura 2.6. Detalhes adicionais desse algoritmo podem ser encontrados em Mladenovic and Hansen (1997).

```
procedimento VNS
1.    Seja s0 uma solução inicial e r o número de estruturas de vizinhança;
2.    s ← s0;          {Solução corrente}
3.    enquanto (Critério de parada não satisfeito) faça
4.       k ← 1;        {Tipo de estrutura de vizinhança}
5.       enquanto (k ≤ r) faça
6.          Gere um vizinho qualquer s' ∈ N(k)(s);
7.          s'' ← BuscaLocal(s');
8.          se f(s'') < f(s)
9.             então s ← s'';
10.                   k ← 1;
11.            senão k ← k + 1;
12.         fim-se;
13.      fim-enquanto;
14.   fim-enquanto;
15.   Retorne s;
fim VNS
```

Figura 2.6 - Algoritmo VNS

Nesse algoritmo, parte-se de uma solução inicial qualquer e a cada iteração seleciona-se aleatoriamente um vizinho *s'* dentro da vizinhança $N^{(k)}(s)$ da solução *s* corrente. Esse vizinho é então submetido a um procedimento de busca local. Se a solução ótima local, *s''*, for melhor que a solução *s* corrente, a busca continua de *s''* recomeçando da primeira estrutura de vizinhança $N^{(1)}(s)$. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança $N^{(k+1)}(s)$. Este procedimento é encerrado quando uma condição de parada for atingida, tal como o tempo máximo permitido de CPU, o número máximo de iterações ou número máximo de iterações consecutivas entre dois melhoramentos. A solução *s'* é gerada aleatoriamente no passo 6 da figura acima, de forma a evitar ciclagem, situação que pode ocorrer se alguma regra determinística for usada.

2.7.5. VND (VARIABLE NEIGHBORHOOD DESCENT)

O VND, ou Método de Descida em Vizinhança Variável, proposto em Mladenovic and 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. O pseudocódigo desse algoritmo é apresentado pela Figura 2.7.

```
procedimento VND
    1.  Seja s0 uma solução inicial e r o número de estruturas de vizinhança;
    2.  s ← s0;              {Solução corrente}
    3.  k ← 1;               {Tipo de estrutura de vizinhança}
    4.  enquanto (k ≤ r) faça
    5.      Encontre o melhor vizinho s' ∈ N(k)(s);
    6.      se  f(s') < f(s)
    7.              então  s ← s';
    8.                     k ← 1;
    9.              senão k ← k + 1;
    10.     fim-se;
    11. fim-enquanto;
    12. Retorne s;
fim VND;
```

Figura 2.7 - Algoritmo VND

3. METODOLOGIA

3.1. DESCRIÇÃO DO PROBLEMA DE PROGRAMAÇÃO DIÁRIA ABORDADO

O problema considerado para análise é o de uma empresa de transporte público, situada em Belo Horizonte. Durante o ano de 2002 a empresa estava responsável por 11 linhas de ônibus, com frota empenhada de 101 veículos. Para operar esta frota a empresa conta com 219 tripulações.

No transporte público, usualmente a programação da tripulação é feita após a programação dos veículos. Nesta, as viagens são reunidas em blocos. Um bloco apresenta a seqüência de viagens que um determinado veículo tem que realizar em um dia, começando e terminando na garagem. Cada bloco mostra também as Oportunidades de Troca (OT). Uma OT é um intervalo de tempo suficiente, em um ponto apropriado, para haver a troca das tripulações.

A Figura 3.1 ilustra a seqüência de viagens (bloco) para o Carro 2.

<Linha 1170> <Carro 2>

Parte da GARAGEM às 06:05

Viagem	<---Partida--->		<---Chegada--->		V.Vazia	Terminal	Linha
00	0	06:05	4	06:20	15	00	1170.00
01	4	06:20	4	07:10	00	05	1170.00
02	4	07:15	4	08:15	00	15	1170.00
03	4	08:30	4	09:38	00	12	1170.00
04	4	09:50	4	11:00	00	05	1170.00
05	4	11:05	4	12:05	00	20	1170.00
06	4	12:25	4	13:25	00	25	1170.00
07	4	13:50	4	15:10	00	20	1170.00
08	4	15:30	4	16:46	00	14	1170.00
09	4	17:00	4	18:00	00	25	1170.00
10	4	18:25	4	19:25	00	25	1170.00
11	4	19:50	4	20:40	00	10	1170.00
12	4	20:50	4	21:40	00	00	1170.00
00	4	21:40	0	21:55	15	00	1170.00

Retorna à GARAGEM às 21:55 - Viagem vazia: 15 min

Figura 3.1 - Programação dos veículos

Conforme a figura, em Partida temos o ponto inicial e o horário de início de cada viagem; já em Chegada, temos o ponto final e horário de fim de cada viagem. Quando o número que representa o ponto inicial ou final for igual a zero, significa que o veículo está saindo ou retornando à garagem, respectivamente. Na programação dos veículos há também as viagens vazias, que é o tempo necessário ao veículo deslocar-se à garagem; o Tempo de Terminal, o qual definirá as oportunidades de troca entre tripulações; e a Linha à qual o veículo pertence.

A partir do bloco de um veículo são criadas as tarefas. Cada tarefa é um conjunto de viagens compreendidas entre duas OT´s: uma no início e outra no final da tarefa. Assim, durante a realização da tarefa, não é possível que haja troca de tripulação. A seguir, temos a Figura 3.2 que esquematiza a programação das tripulações:

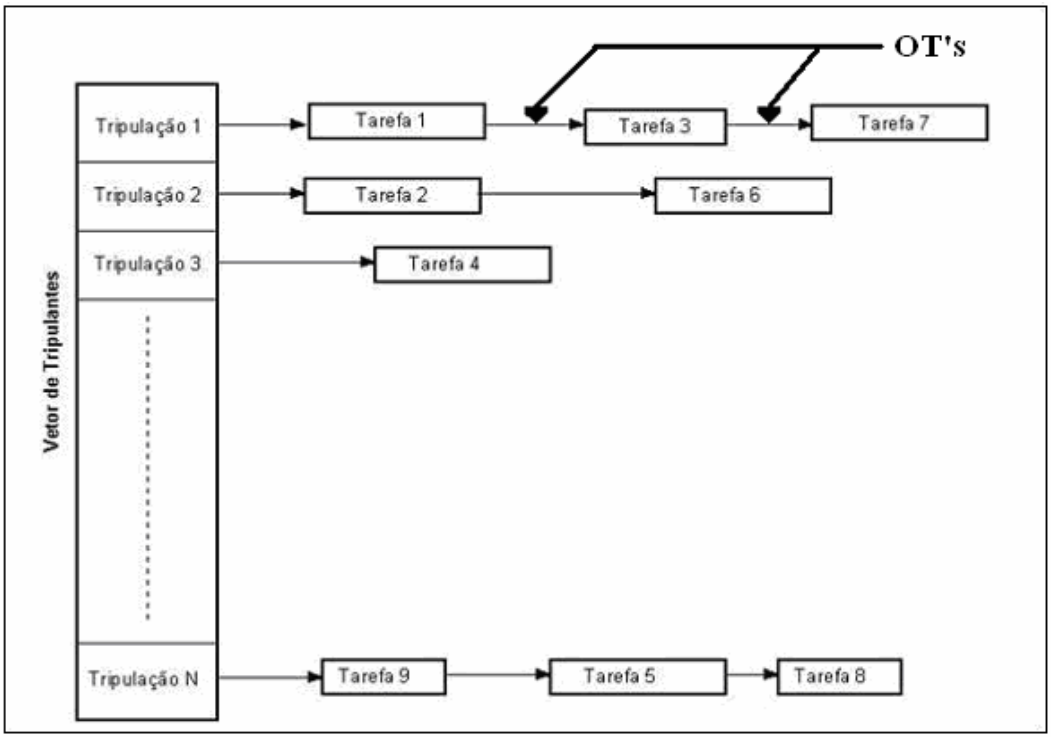


Figura 3.2 - Programação das tripulações

A programação de uma tripulação é formada por um conjunto de tarefas, chamado jornada. As jornadas podem ser divididas em dois tipos: Pegada Simples ou Dupla Pegada, conforme o tempo existente entre as tarefas. No primeiro tipo, as tarefas são realizadas de uma única vez e os intervalos de tempo entre as tarefas são inferiores a duas horas. Caso ocorra um intervalo igual ou superior a duas horas, a jornada é classificada como do tipo dupla pegada. Este intervalo não é contabilizado na remuneração da tripulação, que fica liberada durante este tempo. Após o término do intervalo, o tripulante deve retornar ao trabalho para realizar as tarefas remanescentes.

Na Tabela 3.1 a seguir, temos a representação das tarefas constituindo a jornada de duas tripulações, TRIP 1 e TRIP2, a primeira jornada do tipo Pegada Simples e a segunda do tipo Dupla Pegada, uma vez que entre o término da tarefa 3 e o início da tarefa 4 do segundo tripulante há mais de duas horas de intervalo.

	Tarefa	Carro	Hora de Partida	Hora de Chegada	Ponto Inicial	Ponto Final	Linha
TRIP 1	1	1	04:50	05:45	0	1	101
	2	1	06:48	07:18	1	1	101
	3	1	07:24	07:44	1	1	101
	4	1	07:50	08:28	1	1	101
	5	1	08:30	08:58	1	1	101
	6	1	09:00	09:28	1	1	101
	7	1	09:30	09:58	1	1	101
	8	1	10:00	10:40	1	1	101
TRIP 2	1	3	06:24	07:19	0	2	201
	2	3	07:25	08:10	2	2	201
	3	3	08:16	09:11	2	0	201
	4	3	15:37	16:32	0	2	201
	5	3	16:38	17:23	2	2	201
	6	3	17:29	18:15	2	2	201
	7	3	18:20	19:15	2	0	201

Tabela 3.1 - Representação dos tripulantes e suas respectivas tarefas

Ao se reunir as tarefas formando as jornadas, deve-se levar em conta inúmeras restrições operacionais e trabalhistas. Restrições trabalhistas são impostas pelo governo e sindicatos, já as operacionais representam a gestão da empresa e acordo entre ela e seus funcionários. Essas restrições podem ser classificadas em dois tipos: restrições essenciais, ou seja, aquelas de caráter obrigatório e que devem ser satisfeitas para gerar uma escala viável; e restrições não essenciais, aquelas cujo atendimento melhoram a qualidade da escala gerada, mas que, se não satisfeitas, não geram escalas inviáveis. Foram consideradas as seguintes restrições essenciais:

- Uma tripulação não pode realizar mais de uma tarefa ao mesmo tempo, ou seja, não pode haver sobreposição de tarefas;
- As trocas das tripulações só podem ocorrer nas Oportunidades de Trocas;
- As trocas das tripulações só podem ocorrer entre grupos de linhas predeterminadas, ou seja, grupos de linhas com as mesmas características;
- Uma tripulação não pode realizar trocas de pontos fora do horário de pegada dupla;
- Uma tripulação tem direito a 30 minutos de descanso/alimentação durante sua jornada diária de trabalho, podendo este período ser fracionado em intervalos menores, desde que um deles seja maior ou igual a 15 minutos. Numa jornada com dupla pegada, este intervalo é desconsiderado;
- O número de tripulações com pegada dupla deve estar limitado a um certo valor;

- A jornada máxima de trabalho diário é de 7:10 horas para as tripulações com pegada simples e 6:40 horas para aquelas com dupla pegada, acrescidas de até duas horas extras;
- O tempo entre o final de uma jornada diária de trabalho e o seu início no dia seguinte deve ser de, no mínimo, 11 horas.

Foram consideradas as seguintes restrições não essenciais:

- O tempo ocioso de uma tripulação deve ser o menor possível;
- O número de horas extras deve ser minimizado;
- O número de tripulações deve ser mínimo;
- O número de vezes que uma tripulação troca de veículo deve ser reduzido;
- O número de vezes que uma tripulação com dupla pegada finaliza a primeira parte da jornada em um ponto e inicia a segunda parte em um outro ponto deve ser reduzido;
- O número de vezes que uma tripulação troca de linhas que pertencem ao mesmo grupo, ou seja, possuem as mesmas características, deve ser reduzido.

Obs: Essas restrições constam na Convenção Coletiva de Trabalho 2002/2003 celebrada entre o Sindicato das Empresas de Transporte de Passageiros de Belo Horizonte (SETRABH) e o Sindicato dos Trabalhadores em Transporte Rodoviário de Belo Horizonte (STTRBH).

Tendo em vista o PPT com estas características, foi desenvolvida uma modelagem heurística para implementar os métodos propostos.

3.2. MODELAGEM DO PROBLEMA

A modelagem e a implementação do PPT é fundamentada basicamente em três entidades: Tarefas, Tripulações e Escala diária.

Tarefas: Essa entidade tem a finalidade de representar as tarefas das tripulações.

Tarefa	
N° do veículo	Folga acumulada
Horário inicial	Horário final
Ponto inicial	Ponto final
Linha inicial	Linha final

Tabela 3.2 - Representação de uma tarefa

- Número do veículo: número do veículo que realiza a tarefa;
- Folga acumulada: intervalos obtidos durante a realização da tarefa;
- Horário inicial: horário de início da tarefa;
- Horário final: horário de término da tarefa;
- Ponto inicial: ponto no qual o veículo inicia a tarefa;
- Ponto final: ponto no qual o veículo termina a tarefa;
- Linha inicial: linha na qual o veículo inicia a tarefa;
- Linha final: linha na qual o veículo termina a tarefa.

Tripulações: Essa entidade tem a finalidade de representar as tripulações. Existe, portanto, uma lista de tripulações, e a cada uma está associada sua respectiva jornada de trabalho.

Tripulação	
Nº de tarefas	Tempo de sobreposição
Folga acumulada total	Dupla Pegada
Tempo ocioso	Tempo entre jornadas
Hora extra	Tempo total de trabalho
Nº de trocas de veículos	Função objetivo
Nº trocas de ponto permitidas	Nº trocas de ponto proibidas
Nº trocas de linha permitidas	Nº trocas de linha proibidas
Lista de tarefas	

Tabela 3.3 - Representação de uma tripulação.

- Número de tarefas: é a quantidade de tarefas da tripulação;
- Tempo de sobreposição: tempo total em que uma tripulação realiza mais de uma tarefa ao mesmo tempo;
- Folga acumulada total: folga acumulada de todas as tarefas da tripulação;
- Dupla pegada: caso a jornada possua um intervalo superior a 2 horas entre duas tarefas, ela é classificada como dupla pegada;

- Tempo ocioso: tempo total entre as tarefas, mais a folga acumulada total e mais o tempo que faltar para completar as 7:10 horas (caso pegada simples) ou 6:40 horas (caso seja dupla pegada);
- Tempo entre jornadas: tempo entre o horário de fim e o horário de início de jornadas;
- Hora extra: tempo superior a 7:10 horas, caso não haja dupla pegada, ou superior a 6:40, caso haja;
- Tempo total de trabalho: tempo total trabalhado menos o tempo da dupla pegada (caso haja);
- Número de trocas de veículos: número de vezes em que a tripulação troca de veículos entre duas tarefas;
- Função objetivo: função que avalia a jornada de cada tripulação;
- Número de trocas de pontos permitidas: número de vezes em que o ponto final de uma tarefa é diferente do ponto inicial da tarefa seguinte e o intervalo entre essas tarefas é o tempo da dupla pegada (caso haja);
- Número de trocas de pontos proibidas: número de vezes em que o ponto final de uma tarefa é diferente do ponto inicial da tarefa seguinte e o intervalo entre essas tarefas não é o tempo da dupla pegada;
- Número de trocas de linhas permitidas: número de vezes em que a linha final de uma tarefa é diferente da linha inicial da tarefa seguinte e estas duas linhas pertencem a um mesmo grupo de linhas, definido pela empresa;
- Número de trocas de linhas proibidas: número de vezes em que a linha final de uma tarefa é diferente da linha inicial da tarefa seguinte e estas duas linhas não pertencem ao mesmo grupo de linhas.

As linhas e seus respectivos grupos estão apresentados na tabela a seguir:

Linha	101	201	321	1170	2104	2152	4150	5201	8207	8208	9206
Grupo	1	1	1	2	2	2	2	2	3	3	3

Tabela 3.4 - Linhas de ônibus e respectivos grupos.

- Lista de tarefas: é uma lista duplamente encadeada que contém as tarefas da tripulação. Uma lista de tarefas associada a um tripulante i qualquer é representada pela Figura 3.3:

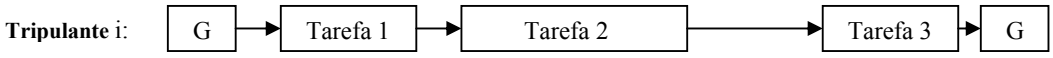


Figura 3.3 - Exemplo de uma alocação

Esta figura mostra, por exemplo, que o tripulante *i* sairá da garagem no início de sua jornada, e a ela retornará ao seu final, após executar as Tarefas 1,2 e 3.

Escala diária: Essa entidade tem a finalidade de representar as escalas diárias com suas respectivas tripulações e tarefas, mostrada na Tabela 3.5:

Escala Diária	
Nº de tripulações com tarefa	Nº de tripulações com dupla pegada
Função objetivo de cada tripulante	Função objetivo total das tripulações
Lista de tripulações	

Tabela 3.5 - Representação de uma escala diária

- Número de tripulações com tarefa: número de tripulações que possuem pelo menos uma tarefa;
- Número de tripulações com dupla pegada: número de tripulações cuja jornada é classificada como do tipo dupla pegada;
- Função objetivo de cada tripulante: função que avalia a escala diária da programação de cada tripulação;
- Função objetivo total das tripulações: somatório das funções objetivos de todas as tripulações;
- Lista de tripulações: lista que contêm todas tripulações e suas respectivas tarefas, como representado na Figura 3.4.

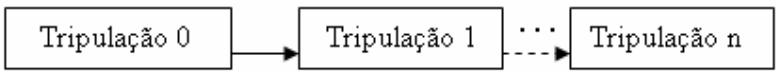


Figura 3.4 - Lista de tripulações

3.3. ESTRUTURAS DE VIZINHANÇA

Dada uma solução *s*, para atingir uma solução *s'*, onde *s'* é dito vizinho de *s*, são usados três tipos de movimentos: Realocação, Troca e Link, para definir, respectivamente, três estruturas diferentes de vizinhança, a saber: $N^{(R)}(s)$, $N^{(T)}(s)$ e $N^{(L)}(s)$.

O movimento de realocação consiste em realocar a tarefa de um tripulante qualquer a outro. Este tipo de movimento é ilustrado na Figura 3.5.

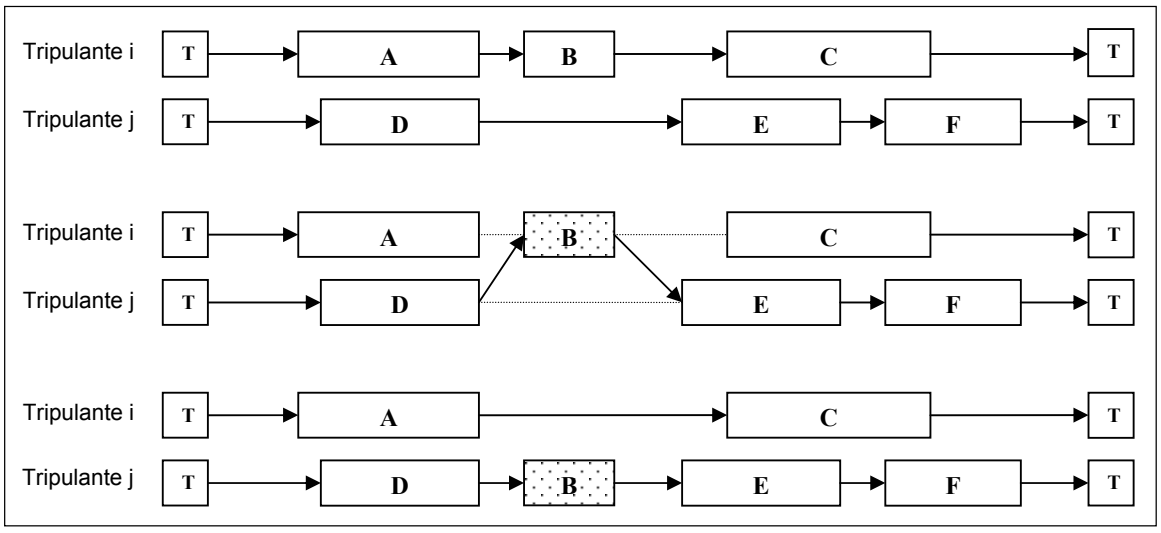


Figura 3.5 - Movimento de Realocação $N^{(R)}(s)$

Nesta figura, onde T significa terminal de troca, a tarefa B, pertencente ao tripulante *i*, é realocada ao tripulante *j*. O conjunto de todos os vizinhos de *s* gerados através de movimentos de realocação define a estrutura de vizinhança $N^{(R)}(s)$.

Já um movimento de troca consiste na permuta de tarefas entre dois tripulantes. Este movimento encontra-se ilustrado na Figura 3.6, onde as tarefas escolhidas, A e D, são trocadas entre as tripulações *i* e *j*.

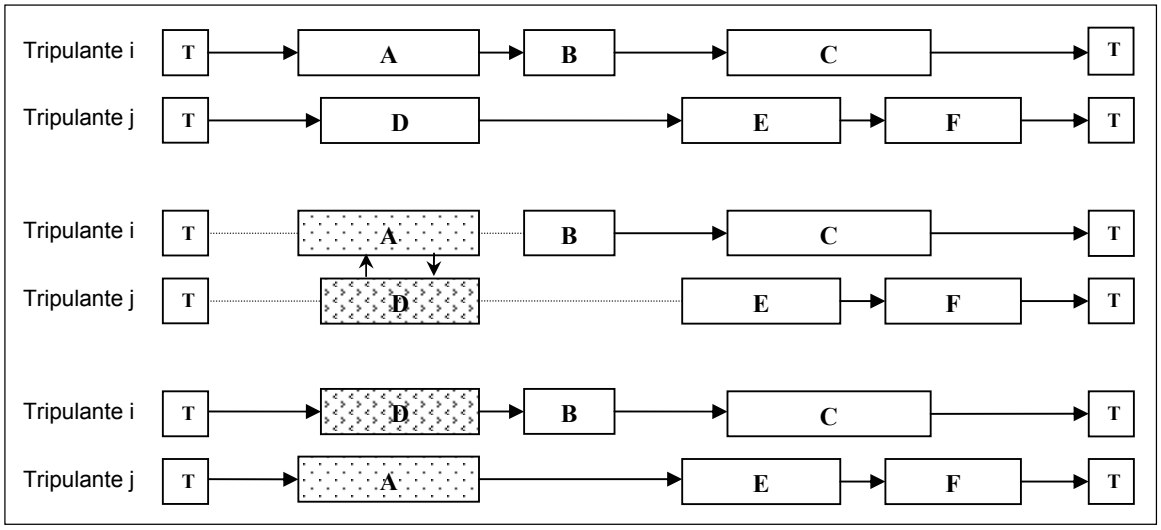


Figura 3.6 - Movimento de Troca $N^{(T)}(s)$

O conjunto de todos os vizinhos de uma escala *s* gerados a partir de movimentos de troca define a estrutura de vizinhança $N^{(T)}(s)$.

Finalmente, o movimento Link consiste na troca de um conjunto de tarefas entre dois tripulantes, conforme ilustrado na Figura 3.7:

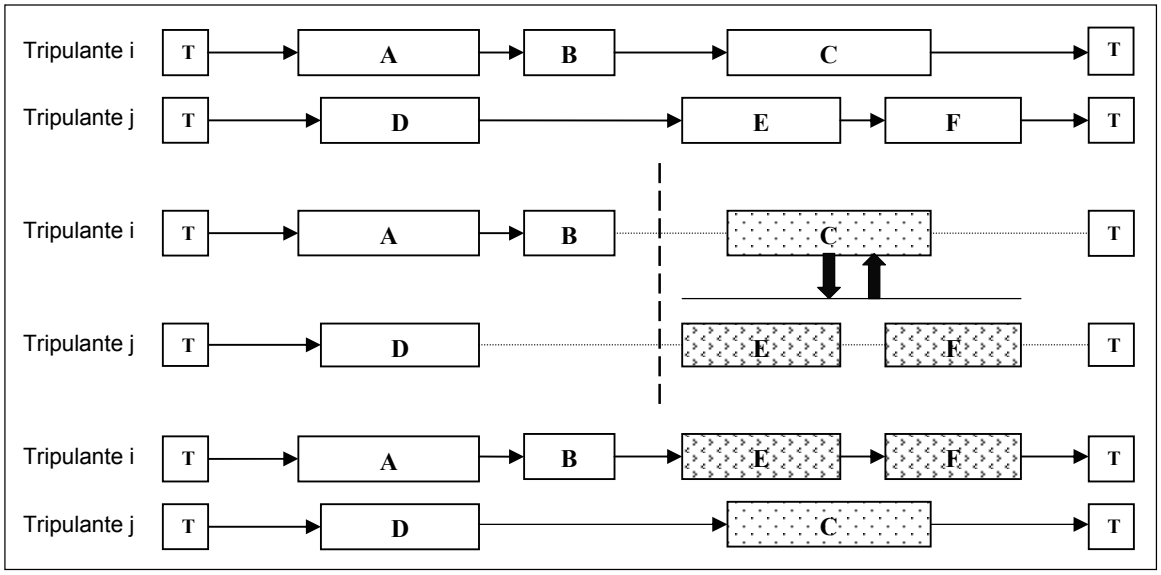


Figura 3.7 - Movimento Link $N^{(L)}(s)$

A partir de uma solução *s* define-se um ponto de corte, local a partir do qual as tarefas serão trocadas entre as tripulações *i* e *j*. A partir de então, vemos as tripulações *i* e *j* constituindo a nova solução *s'* vizinha de *s*. O ponto de corte foi definido como sendo o

tempo correspondente a seis horas após o horário inicial da primeira tarefa do tripulante. A partir deste ponto de corte é escolhida uma tarefa após este, na figura anterior representada pela tarefa C, cujo horário de início definirá uma tarefa E do outro tripulante. Em seguida, o pedaço de jornada constituída pela tarefa C e suas subseqüentes são transferidas para o segundo tripulante. Por sua vez, o primeiro tripulante recebe o pedaço de jornada do segundo tripulante que se inicia com a tarefa E e suas subseqüentes. O critério de escolha dos dois tripulantes envolvidos no movimento é definido a partir de duas listas, L1 e L2, sendo que L1 contêm 10% dos tripulantes com as maiores jornadas, e L2 com 10% daqueles com menor jornada de trabalho. Assim, é considerado um tripulante da lista L1 e outro tripulante da lista L2, de forma a minimizar a ociosidade ou sobrecarga na jornada de trabalho. O conjunto de todos os vizinhos de *s* gerados a partir de movimentos do tipo Link define a estrutura de vizinhança $N^{(L)}(s)$.

3.4. FUNÇÃO DE AVALIAÇÃO

A função objetivo (ou função de avaliação) adotada se baseia na penalização de cada um dos requisitos essenciais e não-essenciais não atendidos.

- Requisitos essenciais: são aqueles que se não forem satisfeitos gerarão uma alocação inviável, como, por exemplo, horário de trabalho superior a 9:10, o que contradiz a lei no qual o trabalhador pode trabalhar no máximo duas horas-extras diária;
- Requisitos não-essenciais: são aqueles cujo atendimento é desejável mas que, se não satisfeitos, não geram alocações inviáveis, como por exemplo, a ociosidade de uma tripulação.

Desse modo, uma escala (ou solução) *s* pode ser medida com base em duas componentes, uma de inviabilidade *g(s)*, a qual mede o não atendimento aos requisitos essenciais, e outra de qualidade *h(s)*, a qual mede o não atendimento aos requisitos considerados não-essenciais. Assim, a função de avaliação de uma solução *s*, *f(s)*, que deve ser minimizada, pode ser calculada na forma:

$$f(s) = g(s) + h(s)$$

A parcela *g(s)*, que mensura o nível de inviabilidade de uma solução *s*, é avaliada com base na expressão:

$$g(s) = \sum_{k=1}^{K} \alpha_k I_k$$

Nesta, K representa o número de medidas de inviabilidade, I_k o valor da k-ésima medida de inviabilidade e α_k como sendo o peso associado à k-ésima medida.

A parcela h(s), que mensura a qualidade de uma solução *s*, é avaliada com base na seguinte função:

$$h(s) = \sum_{l=1}^{L} \beta_l Q_l$$

Nesta, L representa o número de medidas de qualidade, Q_1 o valor da l-ésima medida de qualidade e β_1 o peso associado a essa l-ésima medida.

Deve ser observado que uma solução *s* é viável se, e somente se, *g(s)* = 0. Nas componentes da função *f(s),* os pesos (penalizações) atribuídos às diversas medidas refletem a importância relativa de cada uma delas e, sendo assim, deve-se tomar um valor bem mais elevado para os requisitos essenciais, de forma a privilegiar a eliminação das soluções inviáveis.

3.5. CONTEMPLAÇÃO DO INTERVALO DE REPOUSO/ALIMENTAÇÃO

O presente trabalho foi formulado para atender leis trabalhistas, na qual se exige que haja trinta minutos para repouso/alimentação cumpridos entre o horário inicial da primeira tarefa do tripulante até as suas seis primeiras horas de trabalho, podendo haver fracionamento do período de descanso em intervalos menores, desde que um deles seja maior ou igual a 15 minutos.

Para garantir a contemplação do referente intervalo em tripulações do tipo pegada simples, já que para os tripulantes com dupla pegada esse período de descanso não é exigido, é acrescentado, ao final da jornada da tripulação, caso necessário, uma tarefa

virtual correspondente ao tempo que faltar para tal. A Figura 3.8 abaixo retrata esta situação:

Figura 3.8 - Contemplação do intervalo de repouso/alimentação

Em (a), a jornada do tripulante não está em conformidade a Convenção Coletiva de Trabalho 2002/2003, não possuindo este o intervalo obrigatório de 30 (trinta) minutos. A fim de solucionar este problema, foi acrescentada ao final da jornada do tripulante uma tarefa virtual, *V*, com o intervalo que falta para cumprir este requisito, como mostrado na situação descrita em (b).

3.6. GERAÇÃO DA SOLUÇÃO INICIAL

Para resolver o Problema de Programação das Tripulações faz-se necessária a construção de uma solução inicial, que é então submetida a refinamento pelas metaheurísticas desenvolvidas. Algumas destas metaheurísticas, como o VNS, dependem de uma solução inicial de boa qualidade, garantindo melhor desempenho do método. Sendo assim, foram desenvolvidas algumas formas de geração da solução inicial, como a solução baseada na filosofia da empresa ou usando-se o método de construção da metaheurística GRASP, apresentadas a seguir.

3.6.1. BASEADA NA FILOSOFIA DA EMPRESA

A solução inicial pode ser gerada por várias metodologias. Optou-se, neste trabalho, por gerar uma solução inicial seguindo a filosofia da empresa, a qual baseia-se na repartição dos blocos dos veículos. Cada bloco de um veículo contém um conjunto de

tarefas a serem executadas pelo veículo em um dia. Este bloco será dividido seguindo-se dois critérios: sua duração e o fato de o veículo retornar ou não à garagem durante sua jornada diária. Caso o veículo não retorne à garagem, a divisão do bloco é feita de acordo com a sua duração. O bloco é dividido ao meio caso cada metade não ultrapasse 9:10 horas de trabalho, que é o tempo máximo de trabalho permitido por lei, incluindo as 2:00 horas extras. Caso contrário, o bloco é dividido em três partes, divididas proporcionalmente para os três tripulantes. Cada parte de um bloco é alocada a um tripulante diferente, que se encontra inicialmente desprovido de tarefas.

Caso o veículo retorne à garagem durante sua jornada, a repartição do bloco também se dará conforme a duração da jornada, porém a divisão é feita no intervalo de tempo em que a duração da jornada, excluindo o tempo em que o veículo ficou na garagem, for inferior a 8:40 horas. O tempo em que o veículo permanece na garagem é excluído porque o tripulante que receber essa jornada fará dupla pegada e, portanto, não necessitará dos 30 minutos de intervalo para repouso e/ou alimentação.

Uma das vantagens de gerar a solução inicial seguindo a filosofia da empresa é que esta não contém sobreposições de tarefas nem trocas de pontos ou linhas proibidas. Entretanto, pode ocorrer excesso nas jornadas diárias de trabalho, além do que a solução será sempre idêntica se os dados de entrada (programação dos veículos) forem os mesmos.

3.6.2. BASEADA NA FILOSOFIA DA EMPRESA COM FILTRO

Outra metodologia abordada para o PPT, com capacidade de gerar soluções distintas a cada iteração, é apresentada a seguir. Esta nova metodologia baseia-se também na filosofia da empresa e na repartição dos blocos dos veículos que serão divididos segundo dois critérios: se for um bloco do tipo Pegada Simples ou se for um bloco do tipo Dupla Pegada. A duração do bloco de viagens também será um fator importante para determinação da solução final, pois diminuirá a possibilidade de haver sobrecarga de trabalho na jornada dos tripulantes.

Inicialmente verifica-se o tipo de bloco. Se este for do tipo Pegada Simples, obtém-se então a duração do bloco do veículo, ou seja, o intervalo de tempo entre o horário inicial da primeira tarefa e o horário final da última tarefa a serem realizadas por um único

veículo, em um determinado dia. Se esta duração for inferior a 18:20 minutos, que é o equivalente à soma da duração máxima permitida de jornadas de dois tripulantes com respectivas horas extras, o bloco do veículo será dividido em duas partes distintas, segundo um ponto de corte inicial. Caso contrário, será dividido em três partes, seguindo o mesmo procedimento adotado para o evento anterior.

Se o bloco do veículo for dividido em duas partes, o ponto de corte inicial é definido justamente na metade da duração do bloco. Em seguida, será criada uma variável aleatória para marcar um corte secundário a 60 minutos antes ou após o corte inicial, que dividirá o bloco do veículo em dois grupos distintos de tarefas. A aleatoriedade desta variável está em decidir se o corte secundário será acima ou abaixo do corte inicial. Uma vez obtidos os grupos de tarefas, estes serão atribuídos a dois tripulantes inicialmente desprovidos de tarefas. A seguir, ilustra-se um exemplo demonstrativo:

Tarefa	Carro	Partida (minutos)	Chegada (minutos)
1	1	290	345
2	1	360	405
3	1	411	456
4	1	462	507
5	1	513	600
6	1	605	649
7	1	655	699
8	1	725	749
9	1	755	799
10	1	805	849
11	1	855	899
12	1	905	950
13	1	964	1009
14	1	1015	1060
15	1	1066	1111
16	1	1117	1172

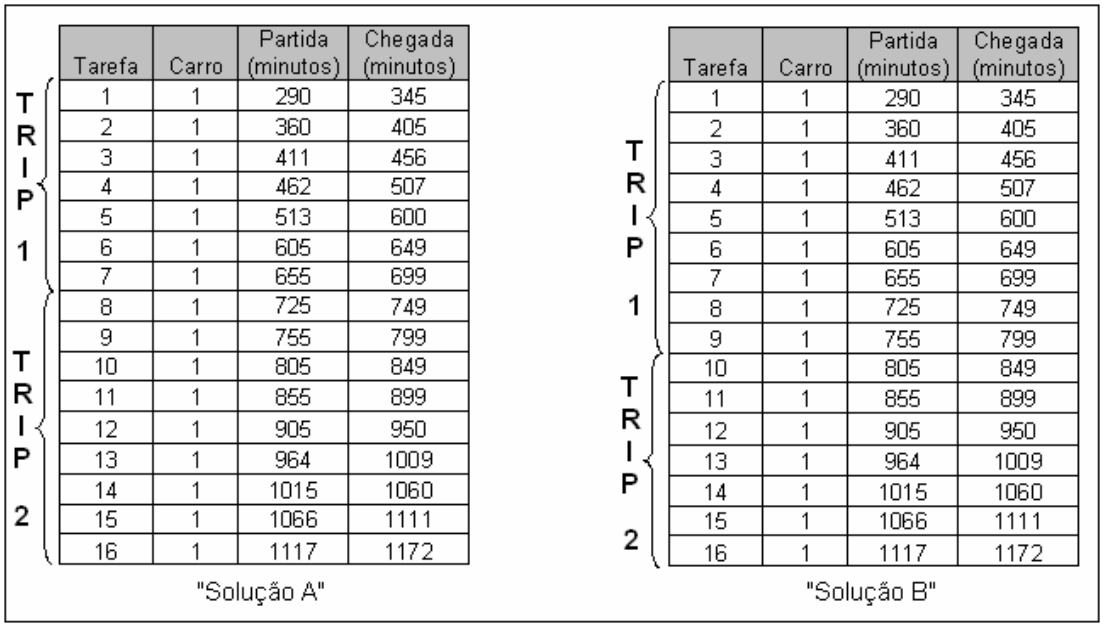
Ponto de corte inicial (731)

Ponto de corte secundário (671)

Ponto de corte secundário (791)

Figura 3.9 - Bloco do veículo e seus respectivos cortes

A figura 3.9 apresenta o bloco do veículo 1, com suas respectivas tarefas. A partir deste bloco, foram definidos os pontos de corte, conforme explicado no parágrafo anterior. Se o ponto de corte secundário sorteado for aquele relativo ao minuto 671, a solução formada estará representada na figura 3.10 como “Solução A”, caso contrário, se o ponto de corte secundário sorteado for aquele que indica o momento 791, a solução será “Solução B” da figura.



"Solução A"

	Tarefa	Carro	Partida (minutos)	Chegada (minutos)
TRIP 1	1	1	290	345
	2	1	360	405
	3	1	411	456
	4	1	462	507
	5	1	513	600
	6	1	605	649
	7	1	655	699
TRIP 2	8	1	725	749
	9	1	755	799
	10	1	805	849
	11	1	855	899
	12	1	905	950
	13	1	964	1009
	14	1	1015	1060
	15	1	1066	1111
	16	1	1117	1172

"Solução B"

	Tarefa	Carro	Partida (minutos)	Chegada (minutos)
TRIP 1	1	1	290	345
	2	1	360	405
	3	1	411	456
	4	1	462	507
	5	1	513	600
	6	1	605	649
	7	1	655	699
TRIP 2	8	1	725	749
	9	1	755	799
	10	1	805	849
	11	1	855	899
	12	1	905	950
	13	1	964	1009
	14	1	1015	1060
	15	1	1066	1111
	16	1	1117	1172

Figura 3.10 - Possíveis soluções após o corte

Outra situação que poderá ocorrer é constatar que o veículo é do tipo Pegada Dupla. Neste caso, o procedimento será verificar o número de tripulantes que já possuem jornada do tipo Pegada Dupla. Se este número for inferior ao número máximo permitido pela empresa, o bloco do veículo será dividido em intervalos com duração de até 8:40 minutos, para que não haja sobrecarga na jornada dos tripulantes envolvidos, e possibilitando que alguns deles sejam do tipo pegada dupla. Caso contrário, o bloco do veículo será dividido justamente no momento da dupla pegada, ou seja, quando o veículo retornar à garagem, evitando assim que o número de tripulantes classificados como de pegada dupla extrapole o máximo permitido pela empresa.

Uma vez esclarecida a fase de construção, esta heurística será executada por um dado número parâmetro do método e retornará a melhor solução encontrada até então. Esta repetição caracteriza uma heurística com filtro, ou seja, estabelecido um número máximo de vezes que será executada, retorna-se aquela solução que possui menor função objetivo.

Uma das vantagens de gerar a solução inicial seguindo esta metodologia é a possibilidade de gerar diferentes soluções a cada iteração, além do que esta será desprovida de sobreposições de tarefas, trocas de pontos ou linhas proibidas ou excesso de tripulantes

do tipo pegada dupla. Entretanto, pode ocorrer excesso nas jornadas diárias de trabalho e não cumprimento do intervalo para repouso e/ou alimentação.

3.6.3. METAHEURÍSTICA GRASP

Outra solução inicial para o PPT pode ser gerada por um procedimento construtivo que segue as idéias da fase de construção do método GRASP (*Greedy Randomized and Adaptative Search Procedure* - Feo and Resende 1995), na qual uma solução é gerada. A melhor solução encontrada ao longo de todas as iterações GRASP realizadas é retornada como resultado.

Inicialmente, toma-se a lista de tarefas que será ordenada pela duração das tarefas (ordem decrescente) e então é submetida a um fator Beta, que determina o tamanho da uma lista restrita das tarefas candidatas (LRTC). O parâmetro *Beta* define o nível de aleatoriedade ou gulosidade da construção, ou seja, a partir do momento que se trabalha com valores de *Beta* próximos a zero, apenas uma pequena percentagem das tarefas farão parte da lista restrita das tarefas candidatas, fazendo gerar soluções gulosas, enquanto *Beta* próximo à unidade faz produzir soluções praticamente aleatórias. Este procedimento foi adotado para que as tarefas de maior duração, que são as mais difíceis de serem alocadas, sejam as primeiras a faz parte da jornada dos tripulantes.

Analogamente, aplica-se o mesmo procedimento para a construção da lista restrita de candidatos (LRC) dos tripulantes, porém a lista dos tripulantes é ordenada segundo sua respectiva função objetivo (em ordem crescente), conforme parâmetro *Alfa*.

Alfa e *Beta* são os únicos parâmetros a serem ajustados nessa implementação do procedimento GRASP.

4. APLICAÇÃO DO MODELO PROPOSTO / ESTUDO DE CASO

4.1. VNS E VND APLICADOS AO PPT

O método VNS explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno de uma nova solução se, e somente se, um movimento de melhora é realizado. O método inclui também um procedimento de busca local, no caso o VND, a ser aplicado sobre a solução corrente.

Para resolver o PPT, foram utilizadas as três estruturas de vizinhança $N^{(R)}(s)$, $N^{(T)}(s)$ e $N^{(L)}(s)$, definidas na seção 3.3, e combinações entre estas. Com estas três estruturas de vizinhança é possível gerar diferentes ordens de exploração. Aquela seqüência que minimize o esforço computacional na exploração das vizinhanças, constituindo a de menor complexidade, será a mais viável para resolução do problema. Para minimizar ainda mais esse esforço e evitar a análise completa de uma dada vizinhança em cada iteração, interrompe-se a busca na vizinhança corrente sempre que ocorrer uma solução de melhora. Outra forma de minimizar este esforço é analisar apenas parte dos tripulantes, usando para isto um parâmetro de entrada que define uma certa porcentagem de tripulantes para realização dos movimentos. No caso do PPT, foi-se analisado 70% da vizinhança no método VNS e 100% da vizinhança do método VND.

O procedimento de busca local implementado (linha 7 da Figura 2.6) foi o Método de Descida em Vizinhança Variável, conhecido como VND (Variable Neighborhood Descent), descrito na seção 2.7.5. Como um exemplo explicativo, se a heurística está definida para seguir a seqüência $N^{(R)}(s)$, $N^{(T)}(s)$ e $N^{(L)}(s)$, começa-se analisando a estrutura de Realocação, fazendo movimentos até que não haja mais melhora nesta estrutura; em seguida parte-se para a estrutura de Troca e caso haja melhora, retorna-se à estrutura de Realocação, dando continuidade ao método. Caso não haja movimento de melhora na estrutura de Troca, segue-se para a o movimento Link. Se houver melhora na função objetivo nesta estrutura, retorna-se aos movimentos de Realocação, repetindo-se o processo. Caso contrário, o método pára, ou seja, foi atingida a última estrutura de vizinhança e nenhuma melhora na solução corrente foi possível. Ao contrário do método VND, o método VNS tem condições de prosseguir a busca quando essa última situação ocorre, uma vez que é promovido o retorno à primeira estrutura de vizinhança e seleciona-

se um outro vizinho qualquer até que uma determinada condição de parada seja satisfeita. A condição de parada adotada foi o tempo total de processamento.

4.2. SIMULATED ANNEALING APLICADO AO PPT

O valor da temperatura inicial do *Simulated Annealing* (T_0) foi calculado por uma função baseada em dois métodos, um de simulação e outro de prospecção, retornando o maior valor entre eles.

No método da simulação, começa-se definindo uma temperatura $T_{inicial}$ com valor mínimo. A partir de então, realizam-se movimentos de realocação, sorteando-se aleatoriamente dois tripulantes e uma tarefa qualquer. O método de simulação utiliza-se de um contador para registrar todos os movimentos de melhora realizados. Caso não seja um movimento de melhora, aceita-o com certa probabilidade, definida por $e^{-\Delta/Tinicial}$, como no método SA. Se o movimento for aceito com esta probabilidade, incrementa-se também o contador. Este ciclo repete-se por um número fixo de iterações *Niter*, definido como sendo o número de tripulantes que a empresa possui multiplicado pelo número total de tarefas que devem ser executadas. O objetivo do método é obter pelo menos 95% dos movimentos aceitos, ou seja, o valor armazenado no contador deve ser igual ou superior a 95% do número de iterações realizadas *Niter*. Enquanto esta condição não é satisfeita, incrementa-se o valor da temperatura $T_{inicial}$ e repete-se o ciclo. Nesta nova etapa, permite-se aumentar a probabilidade de aceitar movimentos de piora, uma vez que está em função de $T_{inicial}$. Quando são atingidos os 95% dos movimentos aceitos, grava-se então o valor de $T_{inicial}$ para posterior comparação com o método da prospecção.

No método de prospecção, o objetivo é obter o valor do maior pico encontrado em um mesmo número de iterações definido no primeiro método, *Niter*. Este pico representa o maior valor da função objetivo, obtido também com movimentos de realocação, que constituirá o valor da temperatura final do método da prospecção. Uma vez calculados o valor da temperatura pelos dois métodos, a função retornará o maior valor entre eles, constituindo então o valor da temperatura inicial T_0, parâmetro de entrada do *Simulated Annealing*.

Uma vez gerada a solução inicial *s* conforme seção 3.6.1, a metaheurística *Simulated Annealing* é composta por quatro opções para resolução do PPT, cada qual

utilizando-se de diferentes combinações de estruturas de vizinhança, conforme a seqüência de tópicos a seguir.

4.2.1. SIMULATED ANNEALING COM MOVIMENTOS DE REALOCAÇÃO

Nesta opção, dois tripulantes são sorteados e é gerado um vizinho qualquer s' de acordo com a estrutura de vizinhança considerada, $N^{(R)}(s)$. Se o valor da função de avaliação f(s') do vizinho for menor que o da solução corrente f(s) e gerar uma variação de energia $\Delta = f(s') - f(s) < 0$, este vizinho é aceito e passa a ser a nova solução corrente. Caso contrário, ou seja, se ocorrer um movimento de piora, a solução é aceita com uma certa probabilidade definida por $e^{-\Delta/T}$, sendo T um parâmetro do método, conhecido como temperatura. Sua função é regular a probabilidade de se aceitar soluções de piora. A temperatura T é inicializada com um valor T_0 elevado, conforme detalhado no tópico anterior. Após um número fixo de iterações SA_{max} (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura), a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_n \leftarrow \alpha\ T_{n-1}$, sendo $0 < \alpha < 1$. Com esse procedimento dá-se, no início, uma chance maior para escapar de mínimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, diminuindo a probabilidade de se aceitar movimentos de piora ($T \rightarrow 0 \Rightarrow e^{-\Delta/T} \rightarrow 0$). O método SA é interrompido quando a temperatura T for menor que um dado valor positivo T_{final}, nomeado temperatura de congelamento, ou quando atinge-se o tempo de processamento estabelecido.

4.2.2. SIMULATED ANNEALING COM MOVIMENTOS DE TROCA

Nesta metodologia, dois tripulantes são sorteados e para cada um deles, escolhe-se uma tarefa aleatoriamente. Uma vez estabelecidos os tripulantes e suas respectivas tarefas, simula-se um movimento de troca, conforme definido em 3.3. Se a solução gerada for de melhora, o movimento é aceito; caso contrário, a solução é aceita com uma certa probabilidade, tal como definido em 4.2.

O que se pode notar neste método, definido apenas por vizinhança $N^{(T)}(s)$, é que ele não é tão eficaz se comparado à opção de se realizar apenas movimentos de realocação. A razão de sua ineficiência é estabelecida pela situação na qual o número inicial de tarefas do tripulante, gerado pela solução inicial definida em 3.6.1, será mantida constante, pois se realizam apenas movimentos de troca de tarefas entre dois tripulantes. Como resultado, esta opção não gera soluções de boa qualidade final se comparado a outros métodos, devendo assim ser descartado seu uso como única estrutura de vizinhança adotada para a resolução do problema.

4.2.3. SIMULATED ANNEALING COM MOVIMENTOS LINK

O movimento Link consiste na troca de um conjunto de tarefas, definidas a partir de um ponto de corte, entre dois tripulantes, sorteados aleatoriamente, gerando-se um vizinho na estrutura de vizinhança $N^{(L)}(s)$, conforme descrito na seção 3.3. Uma vez realizado o movimento, a metaheurística *Simulated Annealing* prossegue conforme já descrito (seção 4.2), até ser interrompido.

Por ser um tipo de movimento que requer maior esforço computacional, a estrutura LINK exige, para que se torne competitivo, que o número máximo de iterações em uma dada temperatura (SA_{max_LINK}) seja menor se comparado às outras duas estruturas de vizinhança. Com o intuito de resolver esta questão, desenvolveu-se uma função que calcula o número máximo de iterações para a estrutura LINK, de forma que seja compatível em tempo computacional com a estrutura $N^{(R)}(s)$, ou seja, consumirá o mesmo intervalo de tempo para se realizar SA_{max} iterações em uma mesma temperatura. Nesta nova função, calcula-se o tempo (em segundos) necessário para realizar o mesmo número de iterações dentro da estrutura de vizinhança $N^{(R)}(s)$ e, posteriormente, dentro da estrutura LINK. Com o propósito de se obter SA_{max_LINK}, relaciona-se a razão entre SA_{max} e a fração entre os tempos determinados.

4.2.4. SIMULATED ANNEALING COM TODOS OS MOVIMENTOS SEGUNDO CERTA PROBABILIDADE

Nesta opção, cada um dos tipos de estrutura de vizinhança definidos para resolução do problema terá uma certa probabilidade de ocorrência. Esta probabilidade está definida pela complexidade de cada estrutura, ou seja, dá-se uma maior chance para movimentos de realocação, seguido por movimentos de troca e movimentos LINK. Para isso, é definida uma variável aleatória z, tal que $0 < z < 1$, que concede 80% dos movimentos para a estrutura $N^{(R)}(s)$ e o restante é dividido igualmente entre as outras duas estruturas de vizinhança.

Uma vez escolhido aleatoriamente a estrutura e realizado o movimento, o *Simulated Annealing* prossegue conforme já descrito na seção 4.2, até atingir o congelamento ou o tempo máximo de processamento.

4.3. BUSCA TABU APLICADO AO PPT

Como dito na seção 2.7.2, a Busca Tabu (BT) é uma classe de metaheurística proposta por Glover (1989,1990) para se resolver problemas de otimização altamente combinatoriais. A idéia básica desse método é de se evitar ficar preso em ótimos locais aceitando-se movimentos de piora.

A seguir temos a adaptação do método BT para resolver o problema de programação de tripulações, utilizando-se duas estruturas de vizinhança: $N^{(R)}(s)$ e $N^{(T)}(s)$.

4.3.1. A LISTA TABU

Como a metaheurística Busca Tabu foi implementada utilizando-se duas estruturas de vizinhança, usou-se também duas listas tabu: uma para armazenar movimentos $N^{(R)}(s)$ e outra para armazenar movimentos $N^{(T)}(s)$.

As Listas Tabu (T^R e T^T), usadas para evitar ciclagem, ou seja, que tem por objetivo impedir o retorno a soluções previamente geradas, foi implementada como uma fila de movimentos reversos àqueles obtidos ao se passar de uma solução *s* para seu vizinho *s'*. Estas listas tabu são de tamanho |T| dinâmico ajustado a cada *MAXITER* (número máximo de iterações) iterações. Dessa forma, um movimento permanece tabu durante |T| iterações. A lista foi considerada de tamanho dinâmico para minimizar a

possibilidade de ciclagem, situação que pode ocorrer quando utiliza-se uma lista tabu de tamanho fixo.

Cada movimento M pertencente a $N^{(R)}(s)$ inserido na Lista Tabu T^R é uma dupla $< T_{fonte}, T >$, onde T_{fonte} é a tripulação de onde foi retirada a tarefa, e T a tarefa realocada. Esse movimento permanecerá tabu por |T| iterações, ou seja, não será permitido realocar a tarefa T à tripulação T_{fonte} ,enquanto este movimento pertencer à lista tabu. Já para um movimento M pertencente a $N^{(T)}(s)$ inserido na Lista Tabu T^T serão consideradas duas duplas, $<T_{fonte}, T_{destino}, T_A, T_B>$, onde T_{fonte} é o tripulante que possui a tarefa T_A, $T_{destino}$ é o tripulante que possui a tarefa T_B e que, após o movimento, estas tarefas serão trocadas entre os tripulantes. Este movimento também permanecerá tabu por |T| iterações.

4.3.2. ANÁLISE DA VIZINHANÇA

Como a análise completa da vizinhança do PPT, tanto na estrutura de realocação $N^{(R)}(s)$ quanto na de troca $N^{(T)}(s)$, é uma operação custosa computacionalmente, optou-se por analisar somente um percentual da vizinhança a cada iteração tabu.

4.3.3. CRITÉRIOS DE ASPIRAÇÃO E DE PARADA

O critério de aspiração implementado foi por objetivo, isto é, um movimento, mesmo tabu, é aceito se melhorar o valor da melhor solução obtida até então.

Com relação aos critérios de parada, foram implementados dois:

- pelo número total de iterações sem melhora no valor da melhor solução encontrada até o momento;
- pelo tempo máximo de processamento.

5. RESULTADOS

5.1. VALORES DOS PESOS UTILIZADOS NAS RESTRIÇÕES

Vários testes foram executados com o objetivo de calibrar os pesos de todas restrições, essenciais e não-essenciais, de forma a gerar soluções viáveis. A seguir, temos os valores destes pesos.

RESTRIÇÃO	PESO
Número de trocas de ponto proibidas	13.000
Número de trocas de linhas proibidas	13.000
Número excessivo de pegadas duplas	9.000
Tempo que excede duração da jornada	5.000
Tempo que falta entre jornadas	5.000
Tempo de sobreposição nas jornadas	5.000
Número de trocas de veículos	5.000
Número de tripulantes	1.000
Número de trocas de ponto permitidas	300
Número de trocas de linhas permitidas	300
Tempo de hora-extra	60
Tempo ocioso	40

Tabela 5.1 - Pesos das restrições

Para garantir a contemplação do intervalo de trinta minutos para repouso e/ou alimentação não é necessário criar um peso, pois o tempo que faltar para tal é acrescentado ao final da jornada da tripulação, como uma tarefa virtual, conforme já descrito em 3.5.

5.2. PARÂMETROS UTILIZADOS NA PROGRAMAÇÃO

Alguns parâmetros utilizados na programação das tripulações estão apresentados na tabela a seguir.

Parâmetros da programação da tripulação	
Número de tripulações	219
Intervalo de folga obrigatória contínua	15 min
Folga total obrigatória	30 min
Número máximo de tripulações com dupla pegada	21
Número total de tarefas	733
Tempo total de processamento para cada teste	60 min

Tabela 5.2 - Parâmetros do PPT utilizados nos testes

O número de tripulações consideradas nesta programação corresponde ao mesmo número de tripulações com o qual a empresa conta para operar a frota. Desta forma, pode-se estabelecer uma comparação entre a escala gerada pelas heurísticas e aquela em uso na empresa. Com relação ao intervalo de folga total obrigatória, equivalente a trinta minutos, corresponde à soma dos intervalos contínuo e que pode estar presente entre as tarefas da jornada diária. O número máximo de jornadas do tipo pegada dupla também é o mesmo adotado pela empresa, que equivale a 10% do número total de tripulações disponíveis.

O critério de parada estipulado para todos os testes foi o tempo total de execução, restrito a 60 minutos.

5.3. METAHEURÍSTICAS APRESENTADAS

As metaheurísticas utilizadas para apresentação dos resultados do PPT são:

- *Simulated Annealing* com três estruturas de vizinhança, $N^{(R)}(s)$, $N^{(T)}(s)$ e $N^{(L)}(s)$, segundo certa probabilidade, conforme seção 4.2;
- Busca Tabu (BT) com duas estruturas de vizinhança, $N^{(R)}(s)$ e $N^{(T)}(s)$, conforme descrito na seção 4.3;
- VNS utilizando as três estruturas de vizinhança e combinações entre estas. Com essas três estruturas é possível gerar diferentes ordens de exploração. Assim sendo, serão apresentadas soluções com VNS_RTL, cuja seqüência de exploração é $N^{(R)}(s)$, $N^{(T)}(s)$ e $N^{(L)}(s)$; VNS_RLT, com a seqüência $N^{(R)}(s)$, $N^{(L)}(s)$ e $N^{(T)}(s)$; VNS_LRT, cuja seqüência é $N^{(L)}(s)$, $N^{(R)}(s)$ e $N^{(T)}(s)$; e VNS_LTRT, cuja ordem de exploração é $N^{(L)}(s)$, $N^{(T)}(s)$, $N^{(R)}(s)$ e $N^{(T)}(s)$.

O que se observa é que a estrutura de vizinhança $N^{(T)}(s)$ faz apenas um refinamento na solução anterior, pois com este tipo de movimento não é possível alterar o tamanho das escalas geradas, ou seja o número de tarefas que compõe cada jornada da tripulação. Logo, não se justifica iniciar as seqüências definidas para o VNS com este tipo de estrutura.

5.4. RESULTADOS COMPUTACIONAIS

Os algoritmos foram desenvolvidos em linguagem C++ usando o compilador Borland C++ Builder 6.0. Os testes foram desenvolvidos em um microcomputador com processador ATLHON XP 1.700, com 512 MB de memória RAM.

A solução inicial para todos os métodos foi gerada conforme a filosofia da empresa (seção 3.6.1). Vale observar que uma das características apresentada por uma solução gerada conforme esta metodologia é a ausência de infrações a restrições tais como: excesso de tripulantes com pegadas duplas; trocas de pontos e linhas proibidas; trocas de veículos; tempo de sobreposição de tarefas; e tempo que falta entre jornadas. Porém, mesmo desprovidas destas infrações, percebe-se que esta solução ainda é inviável e o método pouco contribui para minimização de horas-extras e tempo ocioso, justificando assim a aplicação de heurísticas.

Outro método que poderia ter sido utilizado é a filosofia da empresa com filtro, não tendo sido apresentado aqui porque o objetivo dos testes realizados é fazer uma comparação entre os métodos, fazendo-se necessário que a solução inicial tenha como partida o mesmo valor, ou seja, que utilize a escala inicial com as mesmas características. A solução inicial gerada com a metodologia GRASP não produziu soluções finais de boa qualidade se comparado com a solução gerada conforme os outros métodos, não sendo então apresentado seu resultado neste trabalho.

Para cada metaheurística relacionada na seção 5.3 foram executados dez testes. Inicialmente faz-se necessária apresentação dos valores das funções objetivo dos dez testes realizados para cada metaheurística, conforme Tabela 10.5.

Testes	*BT*	*SA*	*VNS-RTL*	*VNS-RLT*	*VNS-LRT*	*VNS-LTRT*
1	1307800	1358340	1178120	1212980	1275200	1411460
2	1293240	1323300	1271860	1238480	1246760	1310580
3	1260580	1315860	1234040	1240660	1229080	1373820
4	1286320	1325320	1240940	1248060	1372260	1418960
5	1313320	1234900	1207160	1189360	1267120	1304360
6	1259700	1285300	1223160	1257360	1258260	1349320
7	1310360	1322680	1240460	1252780	1295600	1340640
8	1267980	1302700	1262820	1287500	1246720	1337120
9	1332660	1312080	1235420	1220000	1261280	1377780
10	1263620	1301500	1252400	1252200	1307940	1391340

Tabela 5.3 - Valores das funções de avaliação

Para que seja feita uma análise mais criteriosa a respeito dos resultados da tabela anterior, desenvolveu-se a Tabela 5.4 a seguir, na qual serão apresentados os melhores valores obtidos para as funções de avaliação, os valores médios, bem como os desvios em relação ao melhor valor encontrado entre os seis métodos, dado por:

Desvio = ((Valor Médio - Melhor Valor)×100) / (Melhor Valor)

Método	Melhor FO	Média	Desvio
BT	1.259.700	1.289.558	9,46 %
SA	1.234.900	1.308.198	11,04 %
VNS-RTL	1.178.120	1.234.638	4,80 %
VNS-RLT	1.189.360	1.239.938	5,25 %
VNS-LRT	1.229.080	1.276.022	8,31 %
VNS-LTRT	1.304.360	1.361.538	15,57 %

Tabela 5.4 - Desempenho dos algoritmos

As características das melhores escalas geradas por cada método estão apresentadas na Tabela a seguir. Vale observar que todas soluções são viáveis.

Método	Número de Tripulantes	Horas extras (min)	Trocas de Veículos	Trocas Linha permitidas	Trocas Ponto permitidas
BT	218	5.191	43	22	10
SA	218	5.275	40	20	8
VNS-RTL	219	5.229	26	13	12
VNS-RLT	219	5.054	32	17	5
VNS-LRT	219	4.944	38	23	11
VNS-LTRT	219	5.136	47	24	12

Tabela 5.5 - Características das melhores soluções

Para efeito de comparação da solução em uso na empresa de transporte publico que opera em Belo Horizonte, segue a Tabela 5.6:

	Empresa
Nº de tripulações	219
Hora extra (minutos)	6960
Troca de veículos	0
Troca de linhas	0

Tabela 5.6 - Características da solução da empresa

Verifica-se que os métodos Busca Tabu e *Simulated Annealing* foram capazes de minimizar em uma unidade o número de tripulações, assim como o tempo total

de horas extras diárias, principais itens na planilha de custos operacionais da empresa. Esta redução é devida a uma flexibilização na forma de tratar o problema, permitindo-se que tripulações troquem de veículos, pontos e linhas permitidas. Alem disto, pode-se constatar que todos os métodos foram capazes de minimizar o tempo total de horas extras pagas, se comparado à solução em uso na empresa em questão.

6. CONCLUSÃO E RECOMENDAÇÕES

Um dos objetivos deste trabalho é escolher o melhor método para resolução do problema proposto. A seguir será apresentada a figura que ilustra o desempenho da metaheurística VNS, utilizando as três estruturas de vizinhança, com as respectivas ordens de exploração: VNS_RTL, VNS_RLT, VNS_LRT e VNS_LTRT. A finalidade é demonstrar qual destas é a melhor seqüência de estruturas de vizinhança para ser submetida ao PPT.

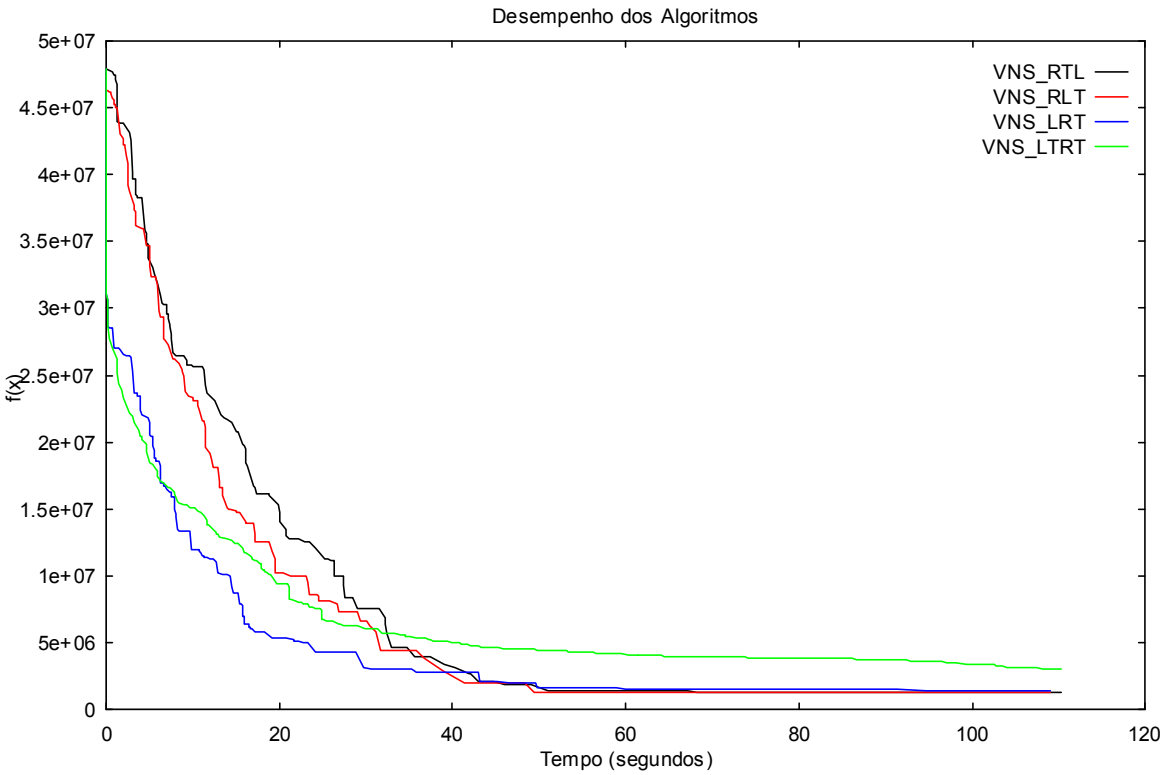


Figura 6.1 - Evolução do método VNS com diferentes ordens de estruturas

Conforme ilustra a Figura 6.1, o método VNS_LRT foi o mais eficiente, visto que alcançou menores valores de função objetivo em menor tempo de processamento. Porém, quando observada a Tabela 5.4, vemos que o VNS_RTL apresenta menor desvio (4,80%) se comparado com o VNS_LRT (desvio = 8,31%) e também menor valor da função objetivo final. Este fato mostra a competitividade entre as diferentes seqüências de estruturas de vizinhança dentro da heurística VNS.

A fim de comparar as metaheurísticas entre elas mesmas, a figura a seguir vem ilustrar a evolução de desempenho entre o VNS_LRT, o BT e o SA.

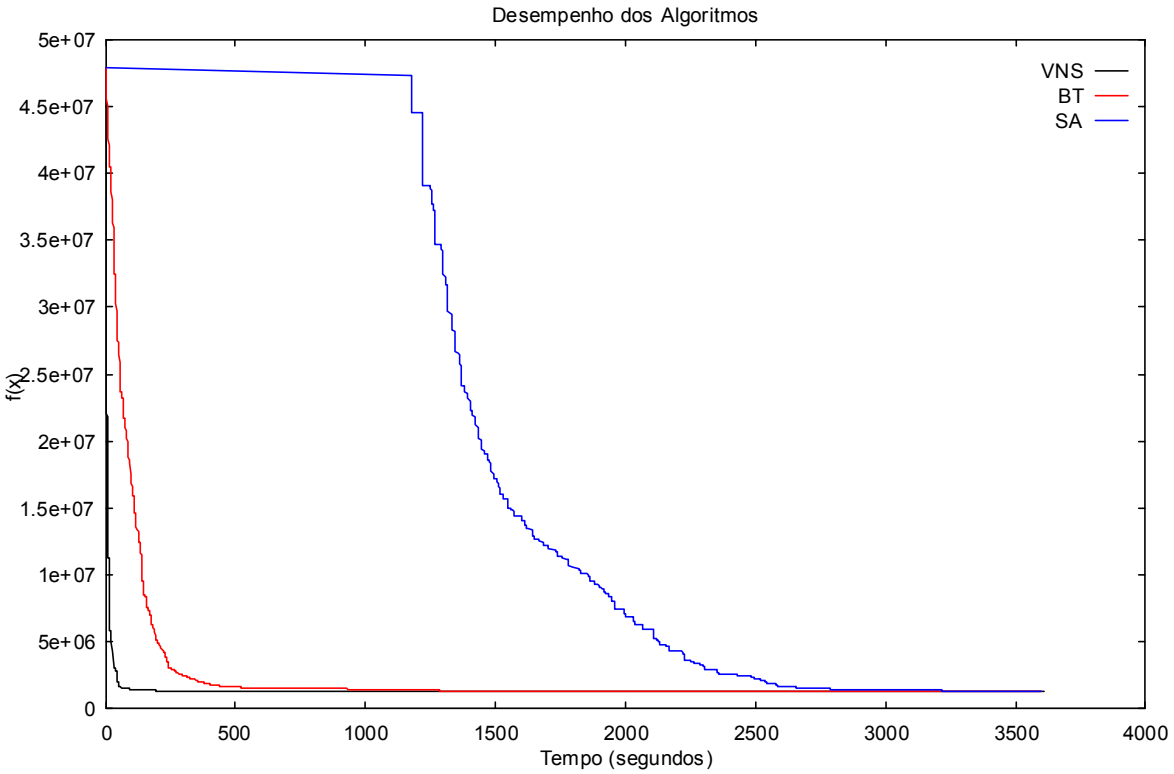


Figura 6.2 - Comparação da evolução entre metaheurísticas

Pela análise da Figura 6.2, percebe-se que o método VNS é mais eficiente que o BT, que ainda é mais eficiente que o SA, conforme se pode perceber também pela análise da Tabela 5.4 e respectivos desvios com relação ao menor valor de função objetivo encontrada, refletindo a robustez dos métodos em gerar soluções de boa qualidade. Pela análise da Tabela 5.5, na qual estão grafadas as características das melhores soluções geradas por cada método, vê-se que os algoritmos de Busca Tabu e *Simulated Annealing*

foram capazes de diminuir em uma unidade o número de tripulações necessárias para operar a frota. Embora a heurística VNS não ter sido capaz de reduzir esse número, esta gerou soluções com menores tempos de horas-extras nas seqüências VNS_RLT e VNS_LRT, compensando o acréscimo de uma tripulação na escala.

O presente trabalho, dando continuidade a uma bem sucedida abordagem heurística do problema, foi capaz de aperfeiçoar o sistema desenvolvido para atender as necessidades de um caso brasileiro, bem como implementar e testar outras metodologias heurísticas a partir das estruturas de vizinhança já criadas, implementadas também em diferentes seqüências. O objetivo pretendido foi possibilitar a redução dos custos operacionais de transporte público, através do desenvolvimento de técnicas heurísticas de otimização eficientes na resolução de um problema real de programação de tripulações de uma empresa do Sistema de Transporte Público, tendo sido então alcançado.

Como recomendações, fica como proposta para trabalhos futuros a implementação da programação mensal e, com maior audácia, a integração com a programação de veículos e posterior implantação do sistema desenvolvido na empresa em questão. Para tanto, faz-se necessário o desenvolvimento de uma interface computacional de alta qualidade para garantir a valorização do sistema e a possibilidade de seu eventual uso comercial.

Vale observar que não foram desenvolvidas metodologias híbridas neste trabalho, pois se apostou em explorar melhor cada metaheurística em sua implementação mais básica, não sendo injusto com nenhuma delas.

7. PRODUTOS GERADOS

Este trabalho deu origem a diversas publicações em revistas científicas, em anais de eventos e participação em fóruns de cunho científico e tecnológico, tais como:

- Título: "Metaheurísticas Aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público". Autores: M.J.F. Souza, G.P. Silva, S.M.S. Mapa, M.M.S. Rodrigues, L.X.T. Cardoso. Trabalho apresentado no XXVI Congresso Nacional de Matemática Aplicada e Computacional, realizado em São José do Rio Preto, SP, de 08 a 11 de setembro de 2003, com resumo publicado nos "Anais do XXVI CNMAC", p. 576.
- Título: "Metaheurísticas Aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público". Autores: M.J.F. Souza, G.P. Silva, S.M.S. Mapa, M.M.S. Rodrigues, L.X.T. Cardoso. Versão completa publicada na revista TEMA - Tendências em Matemática Aplicada e Computacional, em outubro de 2003.
- Título: "Um estudo das heurísticas *Simulated Annealing* e VNS aplicadas ao Problema de Programação de Tripulações". Autores: M.J.F. Souza, G.P. Silva, S.M.S. Mapa, M.M.S. Rodrigues. Trabalho apresentado no "XXIII Encontro Nacional de Engenharia de Produção", realizado em Ouro Preto, MG, de 22 a 24 de outubro de 2003. Versão completa publicada nos "Anais do XXIII ENEGEP".
- Título: "Métodos de Pesquisa em Vizinhança Variável Aplicados à Resolução do Problema de Programação Diária de Tripulações de Ônibus Urbano". Autores: M.J.F. Souza, G.P. Silva, S.M.S. Mapa. Trabalho aceito para apresentação e publicação nos anais do XVIII ANPET - Congresso de Pesquisa e Ensino em Transportes, realizado em Florianópolis, Santa Catarina, de 08 a 12 de novembro de 2004.

8. REFERÊNCIAS BIBLIOGRÁFICAS

BALL, M. & BENOIT-THOMPSON, H. (1988) - *A lagrangean relaxation based heuristic for the urban transit crew scheduling problem*. In: Daduna, J. R.; Wren, A. (eds), p. 54-67.

BASTOS, M.P. & RIBEIRO, C.C. (1999) - *Reactive Tabu Search with Path Relinking for the Steiner Problem in Graphs*. In Proceedings of the Third Metaheuristics International Conference, pages 31_36, Angra dos Reis, Brazil.

BATTITI, R. (1996) - *Reactive Search: Toward Self-Tuning Heuristics*. In V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors, Modern Heuristic Search Methods, chapter 4, pages 61_83. John Wiley & Sons, New York.

BATTITI, R. & TECCHIOLLI, G. (1994) - *The Reactive Tabu Search*. ORSA Journal of Computing, 6:126_140.

BLAIS, J. Y., LLAPORTE, G., LESSARD, R., ROUSSEAU, J. M. & SOUMIS, F. (1976) - *The problem of assigning drivers to bus routes in a urban transit system*, Report, n. 44, Centre de recherche sur les Transports, Université de Montréal.

CAPRARA, A., FISCHETTI, M., TOTH, P., VIGO, D. & GUIDA, P. L. (1997) - *Algorithms for railway crew management*, Mathematical programming, n, 79, p.125-141.

CARRARESI, P. & GALLO, G. (1984) - *Network models for vehicle and crew scheduling*. European Journal of Operational Research,v.16, 139-151.

CHRISTOFIDES,N., (1975) - *Graph theory, an algorithmic approach*, Academic Press,New York.

CLEMENT & WREN (1995) - *Greedy genetic algorithms, optimizing mutantis and bus driver scheduling*. In: Daduna, J. R.; Branco, I.; Paixão, J. M. P. (eds.), p. 213-235.

DAMMEYER, F. & VOŸ, S. (1993) - *Dynamic tabu list management using the reverse elimination method*. In P.L. Hammer, editor, Tabu Search, volume 41 of Annals of Operations Research, pages 31_46. Baltzer Science Publishers, Amsterdan.

DERIGS, U. & ZIMMERMANN, U., (1978) - *An augmenting path method solving linear bottleneck assignment problems*, Computing, n. 19, p. 285-295.

DESROCHERS, M & SOUMIS, F. (1989) - *A column generation approach to the urban transit crew scheduling problem*. Transportation Science, v.23, n.1, p.1-13

DOWSLAND, K. A. (1993) - *Simulated Annealing*. In C.R. Reeves, editor, Modern Heuristic Techniques for Combinatorial Problems, Advanced Topics in Computer Science Series, chapter 2, p. 20-69. Blackwell Scientific Publications, London.

ELIAS, S. E. G. (1964) - *The use of digital computers in the economic scheduling for both man and machine in public transport*. Technical Report 49, Kansas State University Bulletin, Kansas, EUA.

FEO, T.A & RESENDE, M.G.C (1995) - *Greedy randomized adaptive search procedures*, Journal of Global Optimization, 6:691-703.

FISHER, M.L., (1981) - *The Lagrangian relaxation method for solving integer programming problems*, Management Science, n. 27, p. 1-18.

GAMACHE, M., SOUMIS, F., MARQUIS, G. & DESROSIERS, J., (1999) - *A column generation approach for large-scale aircrew rostering problems*. Operations Research, v. 47, n.2, p. 247-263.

GLOVER, F. (1986) - *Future paths for Integer Programming and links to Artificial Intelligence*. Computers and Operations Research, 5:553_549.

GLOVER, F. (1989) - *Tabu Seach: Part I.* ORSA Journal on Computing, 1, p.190-206.

GLOVER, F. (1990) - *Tabu Search: Part II*. ORSA Journal of Computing, 2:4_32.

GLOVER, F. & LAGUNA, M. (1993) - *Tabu Search*. In C.R. Reeves, editor, Modern Heuristic Techniques for Combinatorial Problems, Advanced Topics in Computer Science Series, chapter 3, pages 70_150. Blackwell Scienti_c Publications, London.

GLOVER, F. & LAGUNA, M.(1997) - *Tabu Search*. Kluwer Academic Publishers, Boston.

GLOVER, F. TAILLARD, E. & WERRA, D. (1993) - *A user's guide to tabu search*. In P.L. Hammer, editor, Tabu Search, volume 41 of Annals of Operations Research, pages 3_28. Baltzer Science Publishers, Amsterdan

GOLDBERG, D. E. (1989) - *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Berkeley.

HANSEN, P. (1986) - *The steepest ascent mildest descent heuristic for combinatorial programming.* In Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

HERTZ, A. & WERRA, D. (1990) - *The tabu search metaheuristic: how we used it*. Annals of Mathematics and Arti_cial Intelligence, 1:111_121.

KIRKPATRICK, S.; GELLAT, D.C. & VECCHI, M.P. (1983) - *Optimization by Simulated Annealing*. Science, p. 671-680.

KWAN, A. S. K.; KWAN, R. K.; WREN, A. (1999) - *Driver scheduling using genetic algorithms with embedded combinatorial traits.* In: Wilson, N. H. M. (ed.), Springer-Verlag, Berlin, p. 81-102.

LESSARD, R., ROSSEAU, J. M. & DUPUIS, D., (1981) - *Hastus I: A mathematical programming approach to the bus driver scheduling problem*. Elsevier, Amsterdan.

LOURENÇO, H.R.; PAIXÃO, J.P. & PORTUGAL, R. (2001) - *Multiobjective Metaheuristcs for the Bus-Driver Scheduling Problem*, Transportation Science 35, 331-343.

MANINGTON, B. & WREN, A. (1975) - *A general computer method for bus crew scheduling*. Preprints of the Workshop on Automated Techniques for Scheduling of Vehicle operators for Urban Public Transportation Services, Chicago.

MLADENOVIC, N. & HANSEN, P. (1997) - *Variable neighborhood Search*. Computers and Operations Research, v. 24, p. 1097 – 1100.

PRAIS, M. & RIBEIRO, C.C. (1998) - *Reactive GRASP: An application to a matrix decomposition problem in TDMA trafic assignment*. INFORMS Journal on Computing.

PRAIS, M. & RIBEIRO, C.C. (1999a) - *Parameter Variation in GRASP Implementations*. In Proceedings of the Third Metaheuristics International Conference, pages 375_380, Angra dos Reis, Brazil.

PRAIS, M. & RIBEIRO, C.C. (1999b) - *Variação de Parâmetros em Procedimentos GRASP*. Investigación Operativa.

RIBEIRO, C.C. (1996) - *Metaheuristics and Applications*. In Advanced School on Artificial Intelligence, Estoril, Portugal.

ROSSEAU, J., LESSARD, R., & BLAIS, J.Y. (1985) - *Enhacement to the Hastus crew scheduling algorithm*, Elsevier, Amsterdan.

SCHAEFER, A. (1996) - *Tabu search techniques for large high-school timetabling problems*. In Proceedings of the 30th National Conference on Artificial Intelligence, pages 363_368.

SELMAN, B. LEVESQUE, H. & MITCHELL, D. (1992) - *A new method for solving hard satisfability problems*. In Proceedings of the 10th National Conference on Artificial Intelligence, pages 440_446.

SETRABH – Sindicato de empresas de transporte de passageiros de Belo Horizonte – PRES.OF.Nº 110/2002.

SHEN, Y & KWAN, R.S.K. (2000) - *Tabu Search for Time Windowed Public Transport Driver Scheduling*, In "Research Report Series" , Report 2000.14, School of Computer Studies, University of Leeds, Springer-Verlag.

SHEN, Y.& KWAN, R. S. K. (2001) - *Tabu Search for driver scheduling*. Berlin, In Voß, S; Daduna, J. (eds.) p. 121-135.

SIQUEIRA, P. H. (1999) - Aplicação *do algoritmo do matching no problema da construção de escalas de motoristas e cobradores de ônibus*. Dissertação de mestrado, Setor de Tecnologia e de Ciências Exatas, Universidade Federal do Paraná.

SILVA, G. P.; SOUZA, M. J. F. & ALVES, J. M. C. B. (2002a) - *Resolução do Problema de Programação Diária da Tripulação de Ônibus Urbano via Simulated Annealing*. In XVI Congresso de Pesquisa e Ensino em Transportes. Panorama Nacional de Pesquisa em Transportes. Rio de Janeiro: ANPET, 2002, v. 2, p. 95-104.

SILVA, G. P.; SOUZA, M. J. F. & ALVES, J. M. C. B. (2002b) - *Simulated Annealing Approach to Solve the Daily Crew Scheduling Problem*. In Proceedings of the IV ALIO/EURO Workshop on Applied Combinatorial Optimization. Pucon: Universidad de Concepción, 2002, v.1, p. 127-129.

SMITH, B. M. & WREN, A. (1988) - *A bus crew scheduling system using a set covering formulation*. Transportation Research, v. 22A, p. 97-108.

SOUZA, M.J.F.; CARDOSO, L.X.T.; RODRIGUES, M.M.S.; MAPA, S.M.S & SILVA, G.P. (2003a) - *Metaheurísticas aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público*. In XXVI Congresso Nacional de Matemática Aplicada e Computacional. São José do Rio Preto, setembro de 2003.

SOUZA, M.J.F.; RODRIGUES, M.M.S.; MAPA, S.M.S & SILVA, G.P. (2003b) - *Um estudo das heurísticas Simulated Annealing e VNS aplicadas ao Problema de Programação de Tripulações*. In XXIII Encontro Nacional de Engenharia de Produção. Ouro Preto, outubro de 2003.

SOUZA, M.J.F.; CARDOSO, L.X.T. & SILVA, G.P. (2003c) - *Programação de Ônibus Urbano: uma abordagem heurística*. In XXXV Simpósio Brasileiro de Pesquisa Operacional. Natal, novembro de 2003.

WILHELM, E. B. (1975) - *Overview of the Rucus package driver run cutting program* (RUNS), Chicago.

WREN, A. & ROUSSEAU, J. M. (1995) - *Bus driver scheduling – An overview*. In: Daduna, J. R.; Branco, I.; Paixão, J. M. P. (eds.), p. 173-187.

WREN, A., SMITH, B. M. & MILLER, A. J. (1985) - *Complementary approaches to crew scheduling*, Amsterdam.

WREN, A. & WREN, D. O. (1995) - A *genetic algorithm for public transport driver scheduling.* Computers and Operations Research, v.22, n.1, p.101-110.