UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

Uma abordagem multiobjetivo para o problema de planejamento operacional de lavra

Relatório Final, referente ao período março de 2011 a fevereiro de 2012, apresentado à Universidade Federal de Ouro Preto, como parte das exigências do programa de iniciação cientifica - PROBIC/FAPEMIG.

Equipe:
Marcone Jamilson Freitas Souza (DECOM/UFOP) - Orientador
Vitor Nazário Coelho (Bolsista de Iniciação Científica / FAPEMIG)
Igor Machado Coelho (Mestre em Algoritmos e Otimização / IC-UFF) - Co-orientador

Data de início do projeto: 01/03/2011

Data de fim do projeto: 29/02/2012

Ouro Preto - Minas Gerais - Brasil

Fevereiro de 2012

Resumo

Este trabalho tem seu foco no planejamento operacional de lavra em minas a céu aberto. O problema tratado consiste na mistura de minérios provenientes de várias frentes de lavra, para formar um produto, levando-se em consideração vários objetivos conflitantes, como: obtenção das metas de produção e qualidade para o produto formado, e minimização do número de veículos necessários ao processo produtivo. Considera-se o sistema de alocação dinâmica de caminhões, o que significa que, após as descargas nos pontos de basculamento, cada caminhão pode se dirigir a uma frente diferente para novo carregamento, aumentando a produtividade da frota. Na abordagem multiobjetivo não há uma única solução que satisfaça a todos os objetivos. O que se procura é um conjunto de soluções não-dominadas, também chamadas de soluções eficientes, ou Fronteira de Pareto, cabendo ao tomador de decisões a escolha da solução mais adequada. Neste trabalho foram desenvolvidos dois algoritmos heurísticos multiobjetivos baseados em busca local. O primeiro deles, denominado GRASP-NSGAII-PR, combina os procedimentos *Greedy Randomized Adaptive Search Procedures* (GRASP), *Nondominated Sorting Genetic Algorithm II* (NSGA-II) e o procedimento de Reconexão por Caminhões (PR, do inglês *Path Relinking*), como operador de cruzamento. O segundo algoritmo, denominado GRASP-MOVNS, combina os procedimentos GRASP e *Multiobjective Variable Neighborhood Search* (MOVNS). Ambos os algoritmos foram aplicados de forma a determinar um conjunto de soluções não-dominadas de boa qualidade e em um tempo computacional aceitável para a tomada de decisão em situações práticas. A fase de construção do procedimento GRASP é usada para gerar a população inicial dos algoritmos. As aproximações para a Fronteira de Pareto geradas pelos algoritmos desenvolvidos foram comparadas entre si tendo em vista as métricas de hipervolume, cobertura e espaçamento. Os resultados computacionais encontrados mostraram a superioridade do algoritmo GRASP-MOVNS, que foi capaz de encontrar Fronteiras de Pareto mais diversificadas e com uma melhor qualidade nas soluções.

Palavras-chave: Planejamento operacional de lavra, Metaheurísticas, GRASP, NSGA-II, Otimização Multiobjetivo

Vitor Nazário Coelho
Bolsista

Marcone Jamilson Freitas Souza
Orientador

Sumário

Lista de Figuras

Lista de Algoritmos

Lista de Tabelas

1 Introdução

1.1 O Problema de Planejamento Operacional de Lavra

As mineradoras realizam suas atividades em minas subterrâneas ou a céu aberto. Em minas a céu aberto as atividades de carregamento e transporte ocorrem da seguinte maneira: os caminhões se deslocam até a frente de lavra, que são os pontos da mina onde o minério e o estéril são retirados, são carregados pelos equipamentos de carga e em seguida se dirigem aos pontos de descarga, onde descarregam o minério e o estéril. Os pontos de descarga podem ser pilhas de estéril, material que não é aproveitado pelo processo; pilhas de homogeneização, quando é transportada uma quantidade de minério maior do que a usina pode beneficiar ou quando é necessário "misturar" os minérios antes de iniciar o beneficiamento, e usina de tratamento, onde se inicia o beneficiamento de minério.

Para fornecer minério de qualidade uniforme para o processo é necessário misturar minério de diferentes qualidades proveniente de várias partes da mina ou de diferentes minas com o objetivo de assegurar a uniformidade da alimentação, já que mudanças são usualmente acompanhadas de aumento do custo total da operação (CHANDA; DAGDELEN, 1995).

A atividade de transporte de material é um dos mais importantes aspectos na operação de minas a céu aberto (ALARIE; GAMACHE, 2002). Segundo Maran e Topuz (1988), sistemas de transporte nessas minas envolvem grande volume de capital e recursos. O objetivo do problema de transporte é mover o material retirado da mina para a usina de modo que o custo seja minimizado, uma vez que o custo associado influencia a escolha de onde retirar minério (GERSHON, 1982).

Minas a céu aberto utilizam dois critérios para o transporte de material por caminhões: alocação estática e alocação dinâmica. Na alocação estática, os caminhões seguem uma trajetória fixa entre um ponto de carga e outro de descarga, ou seja, os caminhões ficam fixos a esses dois pontos durante um determinado período de tempo. Já na alocação dinâmica, os caminhões não ficam vinculados a uma mesma rota; assim, a cada descarga, o caminhão pode ser direcionado

a um ponto de carga não necessariamente o mesmo da viagem anterior.

A alocação estática é o método mais utilizado nas minerações de pequeno e médio porte por não apresentar a obrigatoriedade de utilização de um sistema automático de alocação, conhecido como sistema de despacho. Esse método, entretanto, proporciona menor produtividade em função da possibilidade de formação de filas de caminhões e ociosidade dos equipamentos de carga (RODRIGUES, 2006).

A vantagem da alocação dinâmica de caminhões é que com essa estratégia há uma maior produtividade da frota. Esse aumento de produtividade pode refletir um aumento na produção da mina ou a redução do número de equipamentos necessários para manter o mesmo nível de produção. Um algoritmo eficiente para a alocação dinâmica de caminhões é importante porque ele integra um sistema de despacho computadorizado. Um sistema de despacho reúne, ainda, um algoritmo de sequenciamento de viagens, um sistema de comunicação entre os equipamentos de carga e caminhões e uma central de comandos. Segundo White e Olson (1986), para que o sistema de despacho de caminhões seja completo é importante que o sistema de monitoramento dos equipamentos seja preciso e confiável, de modo que as operações da mina possam ser otimizadas em tempo real.

O custo de instalação de sistemas de despacho depende do tamanho da mina e do tipo de operação. Esse custo inibia a sua utilização por mineradoras de pequeno e médio porte. A partir da década de 90, em consequência da evolução da informática, o custo desses sistemas foi consideravelmente reduzido. Essa redução no custo levou ao aumento no número de mineradoras e empreiteiras que utilizam esse tipo de sistema. Segundo Rodrigues (2006), atualmente cerca de 35 minas fazem uso desses sistemas no Brasil, com diferentes níveis de automação.

Ao contrário das abordagens anteriores, em que o POLAD era tratado como um problema de otimização mono-objetivo com soma ponderada de três objetivos, pretende-se no presente projeto fazer uma abordagem multiobjetivo ao mesmo. O projeto visa, assim, a estudar, desenvolver e implementar algoritmos de otimização multiobjetivo para o POLAD. Espera-se conceber um algoritmo que seja capaz de produzir soluções aproximadas para o conjunto Pareto-ótimo, deixando à escolha do tomador de decisão a solução mais atrativa para os interesses da empresa

No presente trabalho, tem-se como foco o problema de planejamento operacional de lavra, considerando alocação dinâmica de caminhões, doravante referenciado por POLAD. Tradicionalmente, o POLAD tem sido tratado como um problema de otimização mono-objetivo com soma ponderada de três objetivos: a minimização dos desvios de qualidade, a minimização dos desvios de produção e a minimização do número de caminhões necessários ao processo. Neste

trabalho propõe-se tratá-lo por uma abordagem multiobjetivo. Desta forma, o que se procura é um conjunto de soluções não-dominadas, também chamadas de soluções eficientes, ou Fronteira de Pareto, cabendo ao tomador de decisões a escolha da solução mais adequada às suas necessidades.

1.2 Objetivos do trabalho

Este trabalho teve como objetivo geral desenvolver um algoritmo multiobjetivo eficiente para resolver o problema de planejamento operacional de lavra com alocação dinâmica de caminhões (POLAD).

Os objetivos específicos estabelecidos foram os seguintes:

(a) Fazer uma revisão de literatura sobre os métodos utilizados para resolver o problema de planejamento de lavra em minas a céu aberto;

(b) Fazer uma revisão de literatura sobre técnicas de otimização discreta multiobjetivo;

(c) Desenvolver algoritmos de otimização multiobjetivo;

(d) Testar os métodos desenvolvidos, sempre que possível, em casos reais da indústria extrativa brasileira;

(e) Produzir um artigo que possa ser apresentado e publicado nos anais de um evento científico nacional;

(f) Contribuir com a divulgação de técnicas de otimização aplicadas à resolução do problema, possibilitando à indústria extrativa nacional melhorar sua produtividade e tornar-se mais competitiva;

(g) Contribuir com a formação de recursos humanos especializados nessa área do conhecimento;

(h) Contribuir para a consolidação das linhas de pesquisa “Otimização e simulação de operações de lavra em minas a céu aberto e subterrâneas” e “Otimização Combinatória” do grupo de Logística e Pesquisa Operacional da UFOP;

1.3 Estrutura do trabalho

Este trabalho está dividido em sete capítulos, incluindo esta introdução, onde o problema de planejamento operacional de lavra é contextualizado.

No Capítulo 2 é apresentada uma revisão bibliográfica sobre os diversos métodos utilizados na resolução do POLAD, bem como a forma com que diversos autores tratam esse problema.

No Capítulo 3 são descritos os principais conceitos e características dos Algoritmos Genéticos, explorando sua estrutura, os diversos operadores, os métodos de seleção dos indivíduos e os diversos parâmetros genéticos que podem ser configurados na solução do POLAD. Também são definidos alguns conceitos de algoritmos multiobjetivo, em especial, os métodos *Non-Sorting Genetic Algorithm - II* (NSGA-II) e *Multiobjective Variable Neighborhood Search* (MOVNS), adaptados neste trabalho para a resolução do POLAD. Por fim, é apresentada uma breve revisão sobre os métodos de comparação e validação, de forma a testar a eficiência dos algoritmos desenvolvidos.

No Capítulo 4 é apresentado o problema abordado em detalhes.

No Capítulo 5 são descritos os algoritmos desenvolvidos para resolver o POLAD.

No Capítulo 6 são apresentados os resultados da aplicação dos algoritmos desenvolvidos, no caso, o GRASP-NSGAII-PR e o GRASP-MOVNS, na solução do POLAD.

No Capítulo 7 são apresentadas as conclusões e apontados os trabalhos futuros.

2 *Revisão Bibliográfica*

White e Olson (1986) propuseram um algoritmo que é a base para o sistema *DISPATCH*, que vem operando em muitas minas em todo o mundo. Uma solução é obtida em duas etapas. Na primeira, baseada em programação linear, realiza-se uma otimização do problema da mistura de minérios tendo como objetivo a minimização de uma função de custo que considera o ritmo de lavra, a qualidade da mistura, o atendimento às taxas de alimentação da usina de beneficiamento e o remanuseio de material. As restrições do modelo estão relacionadas às capacidades de produção dos equipamentos de carga, à qualidade da mistura e às taxas de alimentação mínima requerida da usina de beneficiamento. A segunda etapa do algoritmo, a qual é resolvida por programação dinâmica, usa um modelo semelhante ao de White, Arnold e Clevenger (1982), diferenciando-se deste por utilizar como variável de decisão o volume de material transportado por hora em uma determinada rota, ao invés da taxa de caminhões por hora. É considerada, ainda, a presença de pilhas de estocagem. Nesta segunda etapa do algoritmo, o objetivo é minimizar a necessidade de transporte de material na mina.

Chanda e Dagdelen (1995) desenvolveram um modelo de programação linear por metas para resolver um problema de mistura de minérios no planejamento de curto prazo em uma mina de carvão. A função objetivo do modelo consistia na soma ponderada de três objetivos distintos: maximizar um critério econômico, minimizar os desvios de produção requeridos e minimizar os desvios de qualidade relativos aos valores desejados para os parâmetros de controle. Nenhuma alocação de equipamento de carga e transporte foi considerada nesse modelo.

Alvarenga (1997) desenvolveu um programa para o despacho ótimo de caminhões em uma mineração de ferro, a céu aberto, com o objetivo de minimizar o tempo de fila da frota de caminhões, aumentar a produtividade desta e melhorar a qualidade do minério lavrado. No trabalho desenvolvido, que é base do sistema SMART MINE, atualmente muito utilizado em várias minas brasileiras, foi aplicada uma técnica estocástica de otimização, o algoritmo genético com processamento paralelo.

Merschmann (2002) desenvolveu um sistema de otimização e simulação para análise de

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

Em Costa, Souza e Pinto (2004) e Costa, Souza e Pinto (2005) foram apresentados e modelados problemas relativos à mistura de minérios provenientes de várias frentes de lavra, levando-se em consideração metas de produção e qualidade, alocação dinâmica e estática de caminhões, restrições operacionais e alocação dos equipamentos de carga e transporte necessários ao processo. Os modelos considerados foram baseados em programação linear por metas e representaram um avanço em relação àqueles de Merschmann (2002), isto porque, além de contemplarem mais situações reais, reduziam significativamente o número de restrições do problema.

Como relatado nesses trabalhos, o POLAD é um problema da classe NP-difícil e, como tal, métodos exatos de solução têm aplicabilidade restrita. Desta forma, a abordagem mais comum a ser utilizada passou a ser feita por meio de procedimentos heurísticos, como relatado em Costa (2005), que desenvolveu um algoritmo heurístico baseado em *Greedy Randomized Adaptive Search Procedures* - GRASP (FEO; RESENDE, 1995; RESENDE; RIBEIRO, 2003, 2010) e VNS (MLADENOVIĆ; HANSEN, 1997; HANSEN; MLADENOVIĆ, 2001) para o POLAD usando seis tipos diferentes de movimentos para explorar o espaço de soluções. Foi feita uma comparação entre os resultados obtidos por esse algoritmo heurístico e os encontrados pelo otimizador LINGO, versão 7, aplicado a um modelo de programação matemática desenvolvida pelos autores, publicado em (COSTA; SOUZA; PINTO, 2004). Mostrou-se que o algoritmo heurístico desenvolvido foi capaz de encontrar soluções de melhor qualidade mais rapidamente.

Guimarães, Pantuza e Souza (2007) apresentaram um modelo de simulação computacional para validar resultados obtidos pela aplicação de um modelo de programação matemática na determinação do ritmo de lavra em minas a céu aberto. Dessa maneira, foi possível validar os resultados da otimização, já que na modelagem de otimização não é possível tratar a variabilidade nos tempos de ciclo e a ocorrência de fila.

Em Coelho, Ribas e Souza (2008), o POLAD é resolvido por um algoritmo heurístico, denominado GVILS, que combina os procedimentos heurísticos GRASP, VND e ILS (LOURENÇO; MARTIN; STÜTZLE, 2003). O algoritmo GVILS faz uso de oito movimentos para explorar o espaço de soluções. Além dos desvios de produção e qualidade, procurou-se minimizar, também, o número de veículos. Usando quatro problemas-teste da literatura, o GVILS foi comparado com o otimizador CPLEX 9.1 aplicado a um modelo de programação matemática. Foram realizados testes envolvendo 15 minutos de processamento. Em dois dos problemas, o algoritmo proposto mostrou-se bastante superior; enquanto nos dois outros ele foi competitivo com o CPLEX, produzindo soluções médias com valores até 0,08% piores, na média.

Souza et al. (2010) propuseram um algoritmo, denominado GGVNS, que combina as metaheurísticas *General Variable Neighborhood Search* - GVNS (HANSEN; MLADENOVIC; PÉREZ, 2008) e o procedimento GRASP. Do procedimento GRASP utilizou-se a fase de construção para produzir soluções viáveis e de boa qualidade rapidamente. O GVNS foi escolhido devido a sua simplicidade, eficiência e capacidade natural de sua busca local para lidar com diferentes vizinhanças. Os autores compararam os resultados gerados pelo GGVNS com aqueles alcançados pelo otimizador CPLEX 11.01, utilizando oito problemas-teste. Os experimentos computacionais mostraram que o algoritmo GVNS era competitivo e capaz de encontrar soluções próximas do ótimo (com um $gap < 1\%$) na maioria das instâncias, demandando um pequeno tempo computacional. Coelho et al. (2011b) apresentaram uma paralelização do algoritmo sequencial de Souza et al. (2010). Comparando a versão paralela e a versão sequencial, observou-se a supremacia da versão paralela, tanto em termos de qualidade da solução final quanto na variabilidade.

Coelho et al. (2011a) propõem um algoritmo evolutivo inspirado em Estratégias Evolutivas para resolver o POLAD mono-objetivo. O algoritmo desenvolvido utilizou o procedimento GRASP para gerar a população inicial entregue à Estratégia Evolutiva (ES) (BEYER; SCHWEFEL, 2002). Essa nova abordagem mostrou ser a mais eficiente até o momento, visto que os experimentos computacionais realizados mostraram a efetividade desse algoritmo quando comparado a outros da literatura.

Em termos de abordagem multiobjetivo para POLAD, o único trabalho encontrado na literatura foi o de Pantuza (2011). Este autor propôs um algoritmo genético multiobjetivo híbrido baseado no procedimento *Nondominated Sorting Genetic Algorithm II* (NSGA-II) (DEB et al., 2002). Na abordagem utilizada, foram considerados três objetivos conflitantes: minimizar o número de caminhões necessários para o processo de produção, minimizar os desvios em relação às metas dos teores dos parâmetros de qualidade e minimizar os desvios de produção de

minério. Os resultados do modelo de otimização foram validados pela simulação.

3 *Heurísticas*

As heurísticas são técnicas que visam a obtenção de soluções de boa qualidade em um tempo computacional aceitável. Essas técnicas, no entanto, não garantem a obtenção da solução ótima para o problema, nem são capazes de garantir o quão próximo a solução obtida está da ótima.

As heurísticas podem ser construtivas ou de refinamento. As construtivas têm por objetivo construir uma solução, usualmente, elemento a elemento. A escolha de cada elemento está, geralmente, relacionada a uma determinada função que o avalia de acordo com sua contribuição para a solução. Tal função é bastante relativa, pois varia conforme o tipo de problema abordado.

As heurísticas de refinamento, também chamadas de mecanismos de busca local, são técnicas baseadas na noção de vizinhança. Para definirmos o que é uma vizinhança, seja S o espaço de busca de um problema de otimização e f a função objetivo a minimizar. O conjunto $N(s) \subseteq S$, o qual depende da estrutura do problema tratado, reúne um número determinado de soluções s', denominado vizinhança de s. Cada solução $s' \in N(s)$ é chamada de vizinho de s e é obtida a partir de uma operação chamada de movimento.

Em linhas gerais, esses métodos partem de uma solução inicial s_0, percorrem o espaço de busca por meio de movimentos, passando de uma solução para outra que seja sua vizinha.

3.1 Metaheurísticas

Metaheurísticas são procedimentos destinados a resolver aproximadamente um problema de otimização, tendo a capacidade de escapar das armadilhas dos ótimos locais, ainda distantes de um ótimo global. Elas podem ser de busca local ou populacional. Na primeira, a exploração do espaço de soluções é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora em sua vizinhança. Já na segunda, trabalha-se com um conjunto de soluções, recombinando-as com o intuito de aprimorá-las.

3.2 Otimização Mono-objetivo

3.2.1 Introdução

Os algoritmos de otimização são estratégias inteligentes para solucionar problemas de minimização (ou de maximização) de funções[1], em um determinado domínio, definido pelo conjunto de restrições nas variáveis de decisão. Segundo Goldberg (1989), o problema de otimização mono-objetivo pode ser matematicamente formulado como:

minimize:

$$f(x) \tag{3.1}$$

s.a:

$$g(x) \leq 0 \tag{3.2}$$

$$h(x) = 0 \tag{3.3}$$

$$x \in X \subset \mathbb{R}^N$$

sendo x o vetor de variáveis de decisão ou de otimização com N elementos, X um sub-espaço de $\mathbb{R}^N$, $f(x) : \mathbb{R}^N \to \mathbb{R}$, $g(x) : \mathbb{R}^N \to \mathbb{R}^j$ e $h(x) : \mathbb{R}^N \to \mathbb{R}^k$ são, respectivamente, a função objetivo, o vetor de restrições de desigualdades e o vetor restrições de igualdade.

Para a solução dos problemas de otimização, dois grupos de métodos de otimização se destacam: *i*) métodos determinísticos e *ii*) métodos probabilísticos. No primeiro, os métodos são caracterizados por necessitarem de cálculos de derivadas das funções e fazem a busca da solução ótima gerando uma sequência de pontos segundo a expressão $x^{t+1} = x^t + \alpha d^t$, onde d^t é o vetor de busca, cuja expressão matemática contém informações de derivada das funções[2]

Os algoritmos determinísticos, por necessitarem de informações de derivadas da função objetivo (caso irrestrito), não garantem a convergência para a solução ótima global quando esta função é multimodal.

No segundo grupo, os métodos não necessitam do cálculo da derivada das funções; portanto a função não necessita ser contínua. Além disto, com estes métodos é possível encontrar a solução ótima global de funções multimodais, contudo, não é garantido que esta solução seja encontrada. A desvantagem dos métodos probabilísticos em relação aos determinísticos é que eles necessitam de maior número de cálculo de funções e são, portanto, mais custosos do ponto

[1] As funções podem ser mono ou multivariáveis, lineares ou não-lineares, contínuas ou descontínuas

[2] No caso do método do gradiente, a direção de busca para o problema de minimização irrestrita: minimize $f(x)$ é $d^t = - \bigtriangledown f(x)|_{x=x^t}$

de vista computacional.

Os métodos da computação evolucionária são probabilísticos. Segundo Back, Hammel e Schwefel (1997), a computação evolucionária teve origem na década de 50, contudo, sem grande desenvolvimento nas três primeiras décadas, principalmente devido à falta de computadores eficientes na época. Na década de 70, trabalhos de Rechenberg, Shwefel e Foger foram de grande importância para a mudança da imagem da computação evolucionária. Podem ser citados, em especial, a publicação do livro de John Holland - "Adaptation in Natural and Artificial Systems" em 1975. Nesse trabalho, Holland propõe um método de otimização baseado na seleção e genética natural, que, posteriormente, ficou conhecido como Algoritmos Genéticos (AGs). Os AGs têm sido utilizados na otimização de diversos problemas em várias áreas da ciência. Trabalhos importantes como otimização de dispositivos eletromagnéticos (VASCONCELOS, 1994), Otimização de Funções Matemáticas, Otimização Combinatória, Otimização de Planejamento, Problema do Caixeiro Viajante, Problema de Roteamento de Veículos, Otimização de *Layout* de Circuitos, Síntese de Circuitos Eletrônicos, citados em Michalewicz (1984), são exemplos de aplicações do uso de AGs.

A ideia dos algoritmos presentes na computação evolucionária é evoluir populações de indivíduos (soluções candidatas) na direção do ótimo. Segundo Back, Hammel e Schwefel (1997), dentre os algoritmos existentes na computação evolucionária, a principal diferença entre eles está na representação de seus indivíduos e nas operações realizadas para a geração de novos pontos no espaço de otimização. Esse tópico será melhor discutido na Seção 3.3.4.

Nos casos em que se utilizam métodos de otimização determinísticos, o resultado final do processo de otimização é uma única solução, a qual é aceita pelo tomador de decisões ou não. No caso de se utilizar algoritmos evolucionários, o resultado final é mais flexível, pois o tomador de decisões tem à sua disposição não só a melhor opção, mas também outras que podem ser tão interessantes quanto à melhor solução encontrada.

No contexto da otimização mono-objetivo, alguns algoritmos mono-objetivo serão apresentados a seguir.

3.2.2 Heurísticas Mono-objetivo

Dentre as várias heurísticas conhecidas na literatura, neste trabalho foram utilizadas três heurísticas mono-objetivo descritas nas seções a seguir.

***Greedy Randomized Adaptive Search Procedure* (GRASP)**

GRASP (FEO; RESENDE, 1995; RESENDE; RIBEIRO, 2003, 2010) é um método iterativo constituído basicamente de duas fases: uma fase de construção e uma fase de busca local, cujo objetivo é convergir à solução encontrada na fase de construção para um ótimo local. A Algoritmo 1 apresenta o pseudocódigo básico do método GRASP para um problema de minimização.

Algoritmo 1: Greedy Randomized Adaptive Search Procedure

Entrada: Inteiro *GRASPmax*, Função $f(.)$
Saída: Solução s^* melhor quanto à função f em *GRASPmax* iterações

```
1  f* ← ∞
2  para cada uma das GRASPmax iterações faça
3      Construa uma solução s_0 por uma heurística parcialmente gulosa
4      Submeta s_0 a um procedimento de busca local, retornando s
5      se f(s) < f* então
6          s* ← s
7          f* ← f(s)
8      fim
9  fim
10 retorna s*
```

A primeira fase do GRASP é a de construção, na qual uma solução viável é construída elemento a elemento. Cada elemento ainda não usado na solução é avaliado por uma função gulosa g e compõe uma lista, denominada de Lista de Candidatos (LC). Por meio de um fator $\alpha \in [0,1]$ é criada uma Lista Restrita de Candidatos (LRC), cujos elementos i são os melhores da LC segundo a função g e satisfazem a condição $g_i \leq g_{min} + \alpha \times (g_{max} - g_{min})$, sendo g_{min} o valor do elemento com a melhor avaliação segundo g e g_{max}, o de pior avaliação. Definida a LRC, seleciona-se, aleatoriamente, um candidato da LRC e, em seguida, atualizam-se as listas LC e LRC. O método pára quando LC = $\emptyset$.

De acordo com Feo e Resende (1995), o parâmetro α, que determina o tamanho da LRC, influencia significativamente a qualidade e diversidade das soluções geradas durante a fase de construção. Valores de α muito baixos (próximos de zero), ou seja, que determinam um tamanho muito limitado para a LRC, geram soluções próximas à solução puramente gulosa e implicam em uma baixa diversidade das soluções finais. Já uma escolha de α próxima da seleção puramente aleatória (valores de α próximos a 1) leva a uma grande diversidade de soluções con-

struídas mas, por outro lado, muitas das soluções construídas são de qualidade inferior, tornando mais lento o processo de busca local.

O pseudocódigo da fase de construção é apresentado na Figura 2:

Algoritmo 2: Construção GRASP

Entrada: Lista de elementos candidatos LC, Função $g(.)$

Saída: Solução s construída de forma parcialmente gulosa quanto à função g

```
1 s ← ∅
2 enquanto a solução s não estiver totalmente construída faça
3 |   Classifique os elementos de LC de acordo com a função g
4 |   Crie uma LRC, composta pelos melhores elementos da LC
5 |   Selecione aleatoriamente um elemento de LRC e inclua-o na solução s
6 |   Atualize as listas LC e LRC, eliminando o elemento candidato inserido em s
7 fim
```

A segunda fase do GRASP consiste em refinar a solução gerada pela fase de construção, aplicando um método de busca local. A velocidade de convergência para um ótimo local irá depender da qualidade da solução construída. Quanto melhor for a qualidade da solução gerada pela heurística de construção, maior será a velocidade de convergência desta solução para um ótimo local.

Path Relinking (Reconexão por Caminhos)

A técnica Reconexão por Caminhos ou Religação de Caminhos, conhecida na literatura como *Path Relinking*, foi proposta em (GLOVER, 1996) como uma estratégia de intensificação para explorar trajetórias que conectavam soluções elite obtidas pelo método Busca Tabu ou *Scatter Search* (GLOVER; LAGUNA, 1997).Inicialmente, essa técnica busca por soluções de melhor qualidade, gerando e explorando caminhos no espaço de soluções partindo de uma ou mais soluções elite e levando a outras soluções elite. Para tal finalidade, são selecionados movimentos que introduzem atributos das soluções guia na solução corrente. Desse modo, a Reconexão por Caminhos pode ser vista como uma estratégia que objetiva incorporar atributos de soluções de boa qualidade, favorecendo a seleção de movimentos que as contenham. Porém, Ribeiro e Resende (2012) apresentam a técnica de Reconexão por Caminhos como um operador avançado de recombinação ou cruzamento.

A Reconexão por Caminhos pode ser aplicada segundo duas estratégias básicas (ROSSETI, 2003):

- Reconexão por Caminhos aplicada como uma estratégia de pós-otimização entre todos os pares de soluções elite;

- Reconexão por Caminhos aplicada como uma estratégia de intensificação a cada ótimo local obtido após a fase de busca local.

A aplicação da técnica de Reconexão por Caminhos como um procedimento de intensificação a cada ótimo local é mais eficaz do que empregá-la como um procedimento de pós-otimização (ROSSETI, 2003). Neste caso, a Reconexão por Caminhos é aplicada a pares (s_1, s_2) de soluções, onde s_1 é a solução corrente obtida após o procedimento de busca local e s_2 é uma solução selecionada aleatoriamente de um conjunto formado por um número limitado, $TamConjElite$, de soluções elite encontradas durante a exploração do espaço de soluções. Este conjunto está, inicialmente, vazio. Cada solução obtida ao final de uma busca local é considerada como uma candidata a ser inserida no conjunto elite, desde que ela seja melhor que a solução de pior qualidade desse conjunto e apresente um percentual mínimo de diferença em relação a cada solução do conjunto elite ($PercDif$). Esta estratégia é adotada para evitar que o conjunto elite contenha soluções muito parecidas. Se o conjunto estiver vazio, a solução é simplesmente inserida no conjunto. Se o conjunto elite já possui $TamConjElite$ soluções e a solução corrente é candidata a ser inserida neste conjunto, então esta substitui a solução de pior qualidade.

O algoritmo inicia computando a diferença simétrica $\Delta(s_1, s_2)$ entre s_1 e s_2, resultando no conjunto de movimentos que deve ser aplicado a uma delas, dita solução inicial, para alcançar a outra, dita solução guia. A partir da solução inicial, o melhor movimento ainda não executado de $\Delta(s_1, s_2)$ é aplicado à solução corrente até que a solução guia seja atingida. A melhor solução encontrada ao longo desta trajetória é considerada como candidata à inserção no conjunto elite e a melhor solução já encontrada é atualizada. Diversas alternativas têm sido consideradas e combinadas em implementações recentes (ROSSETI, 2003):

- Não aplicar Reconexão por Caminhos a cada iteração, mas sim periodicamente, para não onerar o tempo final do algoritmo;

- Explorar duas trajetórias potencialmente diferentes, usando s_1 como solução inicial e depois s_2 no mesmo papel;

- Não percorrer a trajetória completa de s_1 até s_2, mas sim apenas parte dela (Reconexão por Caminhos Truncada).

Para a escolha da alternativa mais apropriada, deve-se ter um compromisso entre tempo de processamento e qualidade da solução. Em (RIBEIRO; UCHOA; WERNECK, 2002) foi observado que a exploração de duas trajetórias potencialmente diferentes para cada par (s_1, s_2) de soluções consome, aproximadamente, o dobro do tempo de processamento necessário para explorar apenas uma delas, com um ganho marginal muito pequeno em termos de qualidade de solução. Também foi observado pelos autores que, se apenas uma trajetória deve ser investigada, melhores soluções tendem a ser obtidas quando a Reconexão por Caminhos usa como solução inicial a melhor solução dentre s_1 e s_2 (Esta é a chamada Reconexão por Caminhos Regressiva - *Backward Path Relinking*). Como a vizinhança da solução inicial é explorada com muito mais profundidade do que aquela da solução guia, usar como solução inicial a melhor dentre s_1 e s_2 oferece mais chances ao algoritmo de investigar mais detalhadamente a vizinhança da solução mais promissora. Pela mesma razão, as melhores soluções são normalmente encontradas próximas da solução inicial, permitindo que o procedimento de Reconexão por Caminhos seja interrompido após algumas iterações, antes de a solução guia ser alcançada.

O Algoritmo 3 apresenta o pseudocódigo da Reconexão por Caminhos para um problema de minimização.

Algoritmo 3: Reconexão por Caminhos

Entrada:

1 $\bar{g} \leftarrow s$
2 Atribuir a g' a melhor solução entre s e g
3 Calcular o conjunto de movimentos possíveis $\Delta(s,g)$
4 **enquanto** $|\Delta(s,g)| \neq 0$ **faça**
5 Atribuir a g'' a melhor solução obtida aplicando o melhor movimento de $\Delta(s,g)$ a $\bar{g}$
6 Excluir de $\Delta(s,g)$ este movimento
7 $\bar{g} \leftarrow g''$
8 **se** $f(\bar{g}) < f(g')$ **então**
9 $g' \leftarrow \bar{g}$;
10 **fim**
11 **fim**
12 **retorna** g'

O Algoritmo 3 mostra que o algoritmo de Reconexão por Caminhos unidirecional inicia determinando o conjunto de movimentos $\Delta(s,g)$ que será aplicado a s (solução inicial) até chegar a g (solução guia) (linha 3). Cada iteração do procedimento de reconexão por caminhos unidirecional possui os quatro seguintes passos:

- aplicar à solução corrente $\bar{g}$ o melhor movimento do conjunto de movimentos (linha 5), obtendo a solução g'';
- excluir o melhor movimento do conjunto de movimentos ainda possível (linha 6);
- atualizar a solução corrente (linha 7); e
- testar se a solução corrente, $\bar{g}$, é melhor que a melhor solução, g', encontrada ao longo da trajetória aplicada a s para chegar a g. Em caso afirmativo, atribui-se $\bar{g}$ a g' (linha 9). No início do método de reconexão por caminhos, atribui-se a g' a melhor solução dentre s e g (linha 2).

Quando a solução corrente chega a g, o algoritmo de Reconexão por Caminhos pára (linha 4 do O Algoritmo 3) e retorna a solução g' (linha 12).

***Variable Neighborhood Descent* (VND)**

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto por Mladenović e Hansen (1997), é um método de refinamento que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O pseudocódigo desse algoritmo, em que se considera o refinamento de uma solução s utilizando uma função de avaliação f, a ser minimizada, e um conjunto de r diferentes vizinhanças $N = \left\{N^{(1)}; N^{(2)}; ...; N^{(r)}\right\}$, é apresentado no Algoritmo 4.

Algoritmo 4: Variable Neighborhood Descent

Entrada: Solução s, Vizinhanças $N^{(k)}(.)$, Função $f(.)$

Saída: Solução s^* de qualidade superior ou igual à s de acordo com a função f

```
1  Seja k_max o número de diferentes estruturas de vizinhança
2  k ← 1 {Tipo de estrutura de vizinhança corrente}
3  enquanto k < k_max faça
4      Encontre o melhor vizinho s' ∈ N^k(s)
5      se s' for melhor que s de acordo com a função f então
6          s ← s'
7          k ← 1
8      senão
9          k ← k + 1
10     fim
11 fim
12 s* ← s
13 retorna s*
```

Segundo os autores, o método VND baseia-se em três princípios básicos:

- Um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança;
- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
- Para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximas.

Ainda segundo os autores, o último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Esse é o caso em que os ótimos local e global compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

3.3 Otimização Multiobjetivo

3.3.1 Introdução

Um Problema de Otimização Multiobjetivo (MOOP, do inglês *Multi-Objective Optimization Problem*) possui um conjunto de funções objetivo a serem otimizadas (maximizadas ou minimizadas). Matematicamente, de acordo com Dias e Vasconcelos (2002), o MOOP pode ser formulado como:

minimize:

$$f(x) = \{f_1(x), f_2(x), ..., f_M(x)\} \tag{3.4}$$

s.a:

$$g(x) = \{g_1(x), g_2(x), ..., g_J(x)\} \leq 0 \tag{3.5}$$

$$h(x) = \{h_1(x), h_2(x), ..., h_K(x)\} = 0 \tag{3.6}$$

$$x = \{x_1, x_2, ..., x_N\} \in X \subset \mathbb{R}^N$$

$$y = \{y_1, y_2, ..., y_M\} \in Y$$

em que X é o espaço de decisão e Y o espaço dos objetivos.

Na solução de problemas multiobjetivo é gerado ao final um conjunto de soluções, conhecidas como soluções não-dominadas ou de soluções eficientes. Para compreender o que são estas soluções não-dominadas, é necessário apresentar algumas definições.

Definição 1 : Um vetor x^1 domina um vetor x^2 (matematicamente se escreve $x^1 \prec x^2$), quando a avaliação do primeiro não é pior do que a avaliação do segundo em nenhum dos objetivos e é melhor em pelo menos um. Matematicamente, pode-se dizer que $x^1 \prec x^2$ quando se verifica a seguinte relação matemática:

Se $\forall i \in \{1, ..., M\}, y(x^1) \leq y(x^2) \wedge \ni i \in \{1, ..., M\} \mid y(x^1) < y(x^2)$

Definição 2 :

Se um vetor x^1 não é dominado por nenhum vetor x^2 qualquer, em todo o espaço viável, diz-se que x^1 é uma solução eficiente, não-dominada ou solução Pareto-ótima.

Assim, utilizando estas definições, quando um conjunto de soluções finito P é encontrado, se torna possível realizar comparações das soluções duas a duas, dividindo estas soluções em um grupo chamado de soluções dominadas e de soluções não-dominadas P'. As soluções de

P' são não-dominadas por qualquer outra solução presente em P. Se o conjunto não-dominado P' abrange a totalidade do espaço de busca factível, ele é chamado de conjunto Pareto-ótimo global.

A Figura 3.1 ilustra os espaços das variáveis de decisão e dos objetivos. É também mostrado nesta figura a fronteira Pareto-ótima global. Nesta figura, há dois conjuntos Pareto-ótimos que são não-dominados localmente, mostrando a sua vizinhança no seu espaço de dois objetivos e no espaço de variáveis (à direita).

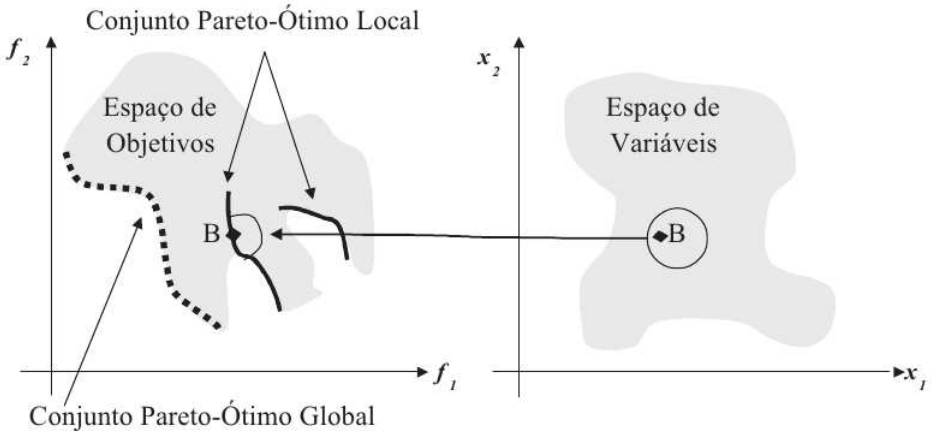

Figura 3.1: Soluções Pareto ótimo locais e globais (ALEXANDRE, 2010)

A fronteira Pareto ótimo, ilustrada na Figura 3.1, é formada por valores das funções objetivo $f(x) = (f_1(x), ..., f_m(x))$ correspondentes a cada solução no espaço de busca. Logo, para cada uma das soluções encontradas no espaço de variáveis, estas soluções são representadas no espaço dos objetivos, avaliando cada uma delas em cada um dos objetivos existentes.

Um dos objetivos principais de algoritmos que solucionam problemas com múltiplos objetivos é encontrar soluções o mais próximo possível da fronteira de Pareto, e, ainda no universo de soluções encontradas, buscarem uma maior diversidade possível.

3.3.2 Metas da Otimização Multiobjetivo

Se não existe nenhuma informação adicional sobre a importância de cada um dos objetivos, todas as soluções Pareto-ótimas são igualmente importantes. Na prática (em empresas, indústrias e em vários outros setores), o tomador de decisão (*decision maker*) define qual a melhor solução a ser utilizada no momento. Deb (2001) assinala três importantes metas em otimização multiobjetivo:

1. Encontrar um conjuntos de soluções que esteja o mais próximo possível da Fronteira de

Pareto;

2. Encontrar um conjuntos de soluções com a maior diversidade possível;

3. Realizar as duas metas anteriores com a maior eficiência computacional possível.

3.3.3 Diferenças com a otimização Mono-objetivo

Deb (2001) identifica três importantes aspectos que diferenciam a otimização multiobjetivo da otimização mono-objetivo

1. Em problemas de otimização com um único objetivo, a meta é encontrar uma solução ótima global. Se a função objetivo desses problemas for multimodal, poderia existir mais ótimo global. Neste caso, todos os ótimos são equivalentes. Por outro lado, em MOOP, determinar o conjunto de soluções da fronteira de Pareto é tão importante quanto preservar a diversidade neste conjunto. Um algoritmo eficiente para otimização multiobjetivo deve considerar ambos os aspectos;

2. Um MOOP trabalha com dois espaços (das variáveis e dos objetivos) ao invés de um. Problemas de objetivo simples trabalham unicamente no espaço de variáveis pois procuram apenas uma soluções no espaço o de objetivos. Manter a diversidade em ambos espaços complica mais o problema, dado que a proximidade de duas soluções no espaço de variáveis *não implica* proximidade no espaço de objetivos.

3. Os métodos tradicionais de otimização multiobjetivo reduzem o conjunto de funções objetivo a uma função simples a qual pondera cada objetivo. Estes métodos podem também tratar cada objetivo separadamente, utilizando os demais objetivos como restrições. Portanto, um MOOP pode ser convertido por meio de algumas técnicas, em um problema de otimização simples.

3.3.4 Heurísticas Multiobjetivo

Computação Evolucionária

Como relatado em (ALEXANDRE, 2010), a computação evolucionária é uma área de pesquisa que busca encontrar soluções eficientes para problemas de grande complexidade. As características principais dos algoritmos evolucionários são:

- São baseados na teoria da evolução de Darwin;

- Trabalham com populações de possíveis soluções;
- São aplicados em diversas áreas tais como otimização mono e multiobjetivo, classificação de padrões, diagnóstico de falhas incipientes, entre outras;
- São fáceis de serem adaptados a diferentes problemas da engenharia e não dependem de características específicas das funções envolvidas no modelo matemático dos problemas;
- São capazes de encontrar boas soluções para problemas com elevado grau de complexidade;
- São simples e fáceis de serem implementados.
- Não garante que seja encontrada a solução ótima para os problemas.
- Utiliza o tempo computacional para avaliação de cada solução gerada.

Os Algoritmos Genéticos (AGs), introduzidos por John Holland, na década de 70, fazem parte da área de Computação Evolutiva, que constitui uma família de métodos computacionais inspirados na evolução natural das espécies. Goldberg (1989) afirma em seu trabalho que o uso de AGs para a solução de problemas multiobjetivo teve inicio quando Schaffer (SCHAFFER, 1985) implementou a primeira versão de um AG multiobjetivo denominado VEGA (*Vector Evaluated Genetic Algorithm*). Esse algoritmo considera uma população de N indivíduos e M objetivos, dividida em m subpopulações com N/m indivíduos em cada uma delas. O operador de seleção dos AGs é aplicado separadamente para cada uma das subpopulações, isto é, para a subpopulação m considera-se apenas o m-ésimo objetivo para fins da seleção, e, posteriormente, une-se estas subpopulações e aplicam-se os outros operadores genéticos de cruzamento e mutação. Além disso, Goldberg (1989) propôs várias abordagens para estender as aplicações de AGs para problemas multiobjetivos. Uma delas propõe um procedimento para ordenação de soluções baseado no conceito de dominância de Pareto. Neste caso, o valor da aptidão de uma solução é proporcional ao número de soluções que ela domina.

O trabalho de Coello (2006) apresenta uma visão geral da história da otimização multiobjetivo. Ele divide os algoritmos até então existentes em duas gerações. A primeira delas envolve algoritmos que possuem como característica a ênfase maior na simplicidade. Entre esses algoritmos destacam-se o VEGA, já discutido anteriormente, o *Nondominated Sorting Genetic Algorithm* (NSGA) (SRINIVAS; DEB, 1994), o *Niched-Pareto Genetic Algorithm* (NPGA) (HORN; NAFPLIOTIS; GOLDBERG, 1994) e o *Multi-Objective Genetic Algorithm* (MOGA) (FONSECA; FLEMING, 1993). A segunda geração dos algoritmos dá maior ênfase à eficiência.

Entre os algoritmos classificados nessa segunda geração estão: *Strength Pareto Evolutionary Algorithm* (SPEA) e *Strength Pareto Evolutionary Algorithm II* (SPEA2) (ZITZLER; LAUMANNS; THIELE, 2001), *Pareto Archived Evolution Strategy* (PAES) (KNOWLES; CORNE, 1999) e o *Nondominated Sorting Genetic Algorithm II* (NSGA-II) (DEB et al., 2002).

Dentre os diversos métodos utilizados na literatura para se encontrar soluções não-dominadas, neste trabalho o foco será dado aos algoritmos NSGA-II e MOVNS, apresentados nas seções 3.3.4 e 3.3.4, respectivamente.

NSGA-II

O algoritmo NSGA II (*Non-dominated Sorting Genetic Algorithm II*), proposto por Deb et al. (2002), foi baseado em seu predecessor, o NSGA, que por sua vez, foi implementado a partir de uma ideia dada por Goldberg (1989). A ideia central do NSGA é classificar os indivíduos em fronteiras não-dominadas e aplicar um método de nicho para diversificar ao máximo possível as soluções.

O NSGA-II apresentou soluções para problemas encontrados no NSGA, como a alta complexidade do procedimento proposto para a ordenação de não-dominância, inexistência de elitismo e a necessidade de se definir a *priori* o raio do nicho no processo de cálculo para manter a diversidade da população. Para a solução dos problemas citados, o NSGA-II define um novo procedimento para a ordenação das soluções com base no critério de não-dominância e cria um novo conceito chamado de *crowding distance* que se torna responsável por manter a diversidade da população.

O Algoritmo 5 apresenta o método NSGA-II é com todos os detalhes do processo iterativo.

Algoritmo 5: Non-dominated Sorting Genetic Algorithm II

Entrada: População P_0; População de *offpring* Q_0; Tamanho fixo da população N;

```
1  g ← 0
2  enquanto critério de parada não satisfeito faça
3      R_g = P_g + Q_g
4      F ← fastNonDominatedSort(R_g)
5      P_{g+1} ← ∅
6      j ← 1
7      enquanto |P_{g+1} + F_j| ≤ N faça
8          F_j ← P_{g+1}
9          j ← j + 1
10     fim
11     dist_j = crowdingDistance(F_j)
12     Ordena F_j conforme as distâncias dist_j
13     Copiar as primeiras N − |P_{g+1}| soluções de F_j para P_{g+1}
14     P_{g+1} ← Torneio por multidão
15     Q_{g+1} ← Seleção, cruzamento e mutação em P_{g+1}
16     g ← g + 1
17 fim
```

No Algoritmo 5, as populações P_g e Q_g são unidas, gerando assim, uma população definida como $R_g = P_g \cup Q_g$. Esta população resultante possui tamanho $2N$.

Após a geração da população R_g, faz-se necessário uma ordenação por não dominância sobre toda a população R_g, obtendo assim, as fronteiras pareto $\mathcal{F}_1$, $\mathcal{F}_2$, e assim por diante. Contudo, apenas algumas soluções contidas nestes conjuntos são inseridas na população P_{g+1}. Desta forma, N soluções da população R_g são descartadas. Com isso, faz-se necessário um procedimento competitivo para realizar o preenchimento da população P_{g+1}. Para realizar este procedimento, são utilizados os dados das soluções encontradas nos conjuntos $\mathcal{F}_1$, $\mathcal{F}_2$, ... sendo as mesmas inseridas em sua totalidade em P_{g+1} enquanto $|P_{g+1} + \mathcal{F}_j| \leq N$. Ao encontrar um conjunto $\mathcal{F}_j$ onde esta condição seja falsa, ou seja, o algoritmo NSGA-II escolhe as soluções presentes na fronteira $\mathcal{F}_j$ que estejam mais bem espalhadas.

Cálculo de Dominância

O procedimento de ordenação utilizado , conhecido como *Fast Non Dominated Sort*, consiste em classificar os indivíduos S de uma população I em diversos níveis de dominância

$\mathcal{F}_1; \mathcal{F}_2; ...; \mathcal{F}_d$ de acordo com o grau de dominância de tais indivíduos, sendo d o número de níveis de dominância. Assim, o nível $\mathcal{F}_1$ contêm os indivíduos dominantes (não-dominados) de toda população I. Já o nível $\mathcal{F}_2$ possui os indivíduos dominantes (não-dominados) de $\mathcal{F}_1$, $\mathcal{F}_3$ contêm as soluções de $I - (\mathcal{F}_1 \cup \mathcal{F}_2)$ e assim sucessivamente.

O Algoritmo 6 exemplifica o procedimento *Fast Non Dominated Sort*. A cada indivíduo i da população P, são associados dois valores $\mathcal{N}_i$ e I_i, a seguir definidos:

- I_i, o conjunto de indivíduos que são dominados pelo indivíduo i;
- $\mathcal{N}_i$, o número de indivíduos que dominam o indivíduo i.

Algoritmo 6: Fast Non Dominated Sort

Entrada: População P

Saída: Fronteiras $\mathcal{F}$

```
1  para cada i ∈ P faça
2      I_i = ∅
3      N_i = 0
4      para cada j ∈ P faça
5          se i ≻ j então
6              I_i = I_i ∪ {j}
7          fim
8          se j ≻ i então
9              N_i = N_i + 1
10         fim
11     fim
12     se N_i = 0 então
13         F_1 = F_1 ∪ {i}
14     fim
15 fim
16 k ← 1
17 enquanto F_k ≠ ∅ faça
18     Q = ∅
19     para cada i ∈ F_i faça
20         para cada j ∈ I_i faça
21             N_j = N_j − 1
22             se N_j = 0 então
23                 Q = Q + {j}
24             fim
25         fim
26     fim
27     k ← k + 1
28     F_k = Q
29 fim
```

As linhas 1 a 15 do Algoritmo 6 fazem os cálculos do número de indivíduos que dominam cada indivíduo da população e criam o conjunto de indivíduos que são dominados por cada

um dos indivíduos. Desta forma, na linha 13 do algoritmo 6, os indivíduos que possuem $\mathcal{N}_i = 0$ (não-dominados) estão contidos no nível $\mathcal{F}_1$. Em seguida, as linhas 17 a 29 percorrem o conjunto de indivíduos dominados I_{d_i} para cada indivíduos i de $\mathcal{F}_i$. O contador $\mathcal{N}_j$ de cada indivíduo j em I_{d_j} é subtraído em 1 unidade. Se $\mathcal{N}_j = 0$, então a solução j pertence à fronteira $k+1$ corrente, logo, essa solução j é armazenada na população auxiliar Q. O procedimento é repetido até que todos os indivíduos estejam classificados em um nível de dominância $\mathcal{F}_d$.

Crowding Distance

Para garantir a diversidade das soluções presentes na fronteira Pareto, o NSGA-II emprega um método que possibilita o cálculo de uma estimativa de densidade das soluções que se encontram em torno de cada um dos indivíduos da população. Para isso, é necessário estimar a distância das duas soluções adjacentes de cada um dos indivíduos para todos os objetivos existentes. Deb et al. (2002) chamou este procedimento de *Crowding Distance*.

Esse procedimento consiste em calcular a média da distância de dois indivíduos adjacentes de cada indivíduo da população em cada um dos objetivos. Assim, os indivíduos são classificados quanto à sua distribuição no conjunto solução, sendo que os indivíduos mais espalhados são priorizados. Esta etapa garante que o conjunto de indivíduos encontrado seja um conjunto mais próximo do conjunto Pareto-ótimo (DEB et al., 2002).

A Fig. 3.2 exemplifica o cálculo da *Crowding Distance*.

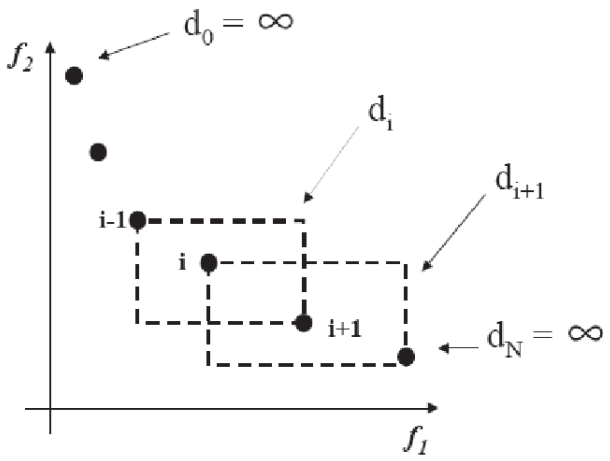

Figura 3.2: Cálculo da crowding distance (DEB, 2001)

Torneio de Multidão

Além da criação do método de Distância de Multidão, o NSGA-II faz uma proposta de alteração no método de seleção. Esse novo método utiliza o método do torneio como base para a seleção dos indivíduos. Contudo, utiliza a Distância de Multidão para compor o torneio. A

este método foi dado o nome de *crowded tournament selection operator*, cuja notação é $<_n$.

Sendo assim, assumindo que cada indivíduo i possui dois atributos:

1. nondomination rank(i_{rank});
2. crowding distance($i_{distance}$).

é definida a seguinte relação de ordem:

$i <_n j$ se ($i_{rank} < j_{rank}$) ou (($i_{rank} = j_{rank}$) e ($i_{distance} > j_{distance}$)).

Desta forma, entre duas soluções com diferentes *nondomination ranks*, escolhe-se a soluções com o menor *rank*. Caso contrário, se as duas soluções possuem estão na mesma fronteira, escolhe-se a solução que está em uma região mais distante, ou seja, possui a maior *crowding distance*.

Multiobjetive Variable Neighborhood Search (MOVNS)

Uma das primeiras aplicações da metaheurística VNS (MLADENOVIĆ; HANSEN, 1997; HANSEN; MLADENOVIĆ, 2001) para otimização multiobjetivo foi proposta por Geiger (2004). Esse autor aplica seu algoritmo, denominado *Multiobjective Variable Neighborhood Search* (MOVNS), para resolver problemas de programação de tarefas *flowshop* bi-objetivo. Esse algoritmo já foi aplicado em vários outros problemas, como o problema de programação de tarefas em máquinas paralelas com mais de um objetivo (LIANG; CHEN; TIEN, 2009) e ao problema de alocação de redundância bi-objetivo (LIANG; LO, 2010). Seu pseudocódigo básico é apresentado pelo Algoritmo 7:

Algoritmo 7: Multiobjective Variable Neighborhood Search

Entrada: Aproximação inicial de um conjunto eficiente Xe; Vizinhanças $N^{(k)}(.)$

Saída: Aproximação de um conjunto eficiente Xe

enquanto *Critério de parado não satisfeito* **faça**
  Seleciona uma solução não visitada $s \in Xe$
  Marque s como *visitada*
  Selecione aleatoriamente uma vizinhança $N^{(k)}(.)$
  $s' \leftarrow shaking(N^{(k)}(s))$
  para todo $s'' \in N_k(s')$ **faça**
    addSolution$(Xe, s'', f(s''))$
  fim
  se *todo* $s \in Xe$ *estão marcadas como visitadas* **então**
    Marque todos $s \in Xe$ como *não-visitado*
  fim
fim
retorna D

Algoritmo 8: addSolution

Entrada: População Xe potencialmente eficiente; Solução s, e sua avaliação $z(s)$

Saída: Xe; Added (opcional)

Added $\leftarrow$ verdadeiro
para todo $x \in Xe$ **faça**
  se $z(x) <= z(s)$ **então**
    Added $\leftarrow$ falso
    Break
  fim
  se $z(s) \prec z(x)$ **então**
    $Xe \leftarrow Xe \setminus x$
  fim
fim
se $Added = verdadeiro$ **então**
  $Xe \leftarrow Xe \cup s$
fim
retorna Xe

O funcionamento do procedimento MOVNS, apresentado no Algoritmo 7, pode ser descrito da seguinte forma:

- Escolhe-se aleatoriamente uma estrutura de vizinhança e uma solução base;
- A solução base é escolhida no conjunto *D* de soluções não-dominadas, cujo procedimento é apresentado no Algoritmo 8;
- Esta solução é marcada como visitada, para evitar que ela seja selecionada novamente;
- Se todas as soluções já foram selecionadas e a condição de parada não foi satisfeita, então todas as soluções são desmarcadas;
- A cada iteração o conjunto é atualizado.

3.3.5 Métricas de Avaliação de Desempenho

Métricas de avaliação de desempenho são bastante utilizadas a fim de mensurar características de algoritmos, ajudando a entender seu comportamento no domínio do problema e permitindo uma avaliação mais concreta do desempenho do algoritmo. Porém, comparar experimentalmente o desempenho de um ou vários algoritmo multiobjetivos não é uma tarefa trivial (DEB, 2001; ZITZLER; DEB; THIELE, 2000). As métricas também são um importante parâmetro de comparação entre algoritmos, uma vez que muitas vezes é difícil perceber qual algoritmo apresenta um melhor conjunto de soluções para o problema. As duas principais metas da otimização multiobjetivo são a convergência e a diversidade das soluções encontradas. A Figura 3.3 ilustra ambas as metas. É importante observar que a diversidade e a convergência são conflitantes entre si; logo, utilizar apenas uma métrica não avaliará por completo a Frente de Pareto analisada.

A Figura 3.4 mostra as soluções não-dominadas obtidas por dois algoritmos hipotéticos A e B, neste caso, o algoritmo A possui uma convergência, enquanto que o algoritmo B obteve uma Fronteira de Pareto bem diversificada. Já na Figura 3.5, é notório que a tarefa de comparar as Fronteiras de Pareto não é trivial, sendo difícil determinar qual algoritmo obteve o melhor desempenho.

Neste trabalho foram utilizadas três métricas de comparação: Hipervolume, Espaçamento e Cobertura. Uma breve revisão sobre essas três métricas é apresentada abaixo.

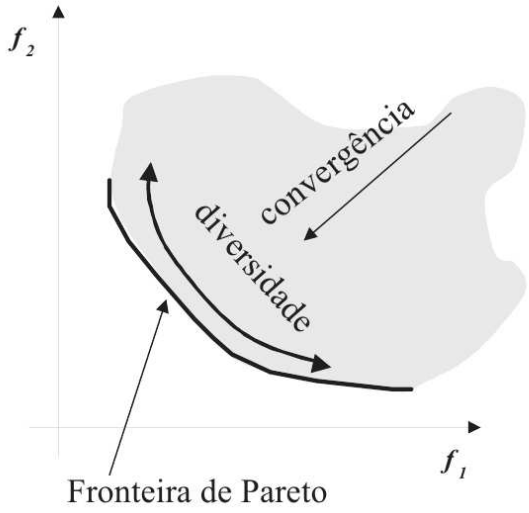

Figura 3.3: Metas da Otimização Multiobjetivo (DEB, 2001)

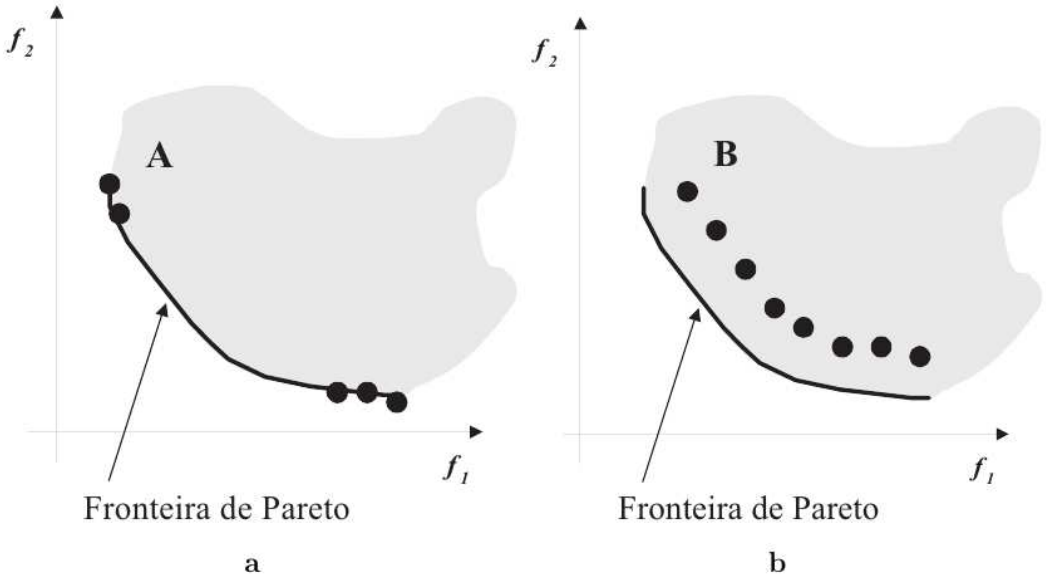

Figura 3.4: Distribuição $\times$ Convergência - 1 (DEB, 2001)

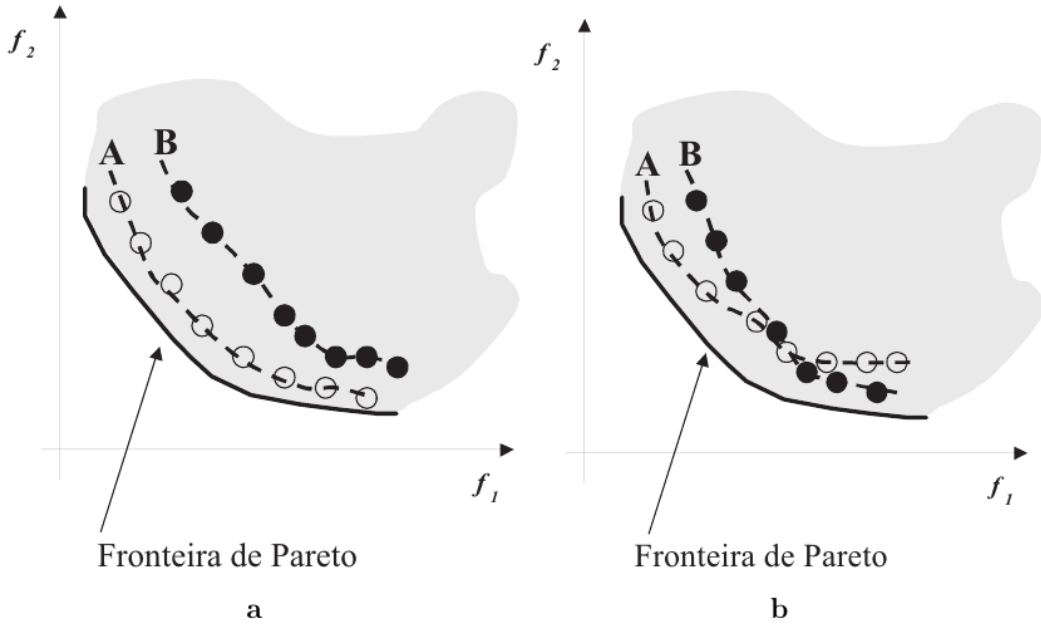

Figura 3.5: Distribuição × Convergência - 2 (DEB, 2001)

Hipervolume

O indicador de hipervolume (ZITZLER; THIELE, 1998) mensura o volume da região coberta entre os pontos das soluções do *front* encontrado e um ponto de referência. Matematicamente, para cada solução i pertencente à Fronteira de Pareto Q, é construído um hipercubo v_i de acordo com uma ponto de referência W_0. Uma maneira fácil de determinar este ponto é contruir um vetor com os piores valores de função objetivo. O resultado da métrica é a união de todos os hipercubos encontrados. Nessa métrica, um alto valor de hipervolume indica que houve um elevado espalhamento entre as soluções extremas do Pareto encontrado e indica, também, que houve uma maior convergência, pois a convergência aumenta o volume em relação ao ponto de referência. A figura Essa métrica apresenta um elevado custo computacional e quando se tem um número de objetivos maior que dois, o seu cálculo passa a não ser trivial. Fonseca, Paquete e Lopez-Ibanez (2006), Beume et al. (2009) propuseram uma ferramenta computacional eficiente para o cálculo do hipervolume, a qual foi utilizada neste trabalho.

A Figura 3.6 ilustra o hipervolume para um Fronteira de Pareto com dois objetivos.

Espaçamento

Schott (1995) propôs uma métrica de espaçamento que mensura as distribuições das soluções na Fronteira de Pareto. Essa métrica calcula a distância relativa entre soluções consecutivas na Fronteira. A Eq. (3.7) descreve o cálculo dessa métrica.

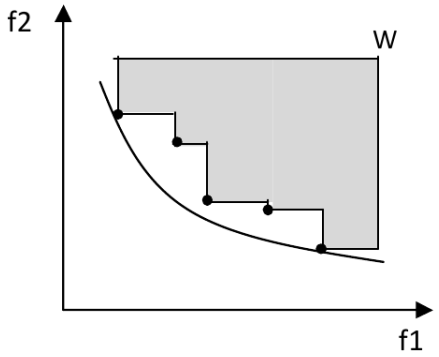

Figura 3.6: Hipervolume gerado pelas soluções não-dominadas de um Fronteira de Pareto hipotética

$$SS = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - \overline{d})^2} \tag{3.7}$$

Na Eq. (3.7), $|Q|$ é número de soluções da Frente de Pareto, $d_i = min_{j \in Q | i \neq j} \sum_{m=1}^{M} |f_m^i - f_m^j|$ e $\overline{d}$ é a média de todos os d_i, ou seja, $\overline{d} = \sum_{i=1}^{|Q|} \frac{d_i}{|Q|}$. O parâmetro M indica o número de objetivos do problema e, por fim, f_m^i e f_m^j indicam os valores do objetivo m para as soluções i e j, respectivamente. Como toda medida de variância, quanto menor for o valor do espaçamento, melhor será a distribuição das distâncias d_i. Consequentemente, as soluções da Frente de Pareto estarão separadas mais uniformemente. Um valor da métrica igual a zero significa que todas as soluções estão equidistantes na Frente de Pareto analisada.

Cobertura

A métrica de cobertura (ZITZLER; THIELE, 1998) é capaz de mensurar quanto um determinado conjunto de soluções domina outro conjunto. Para duas Frentes de Pareto A e B, a cobertura $C(A,B)$ é calculada de acordo com a equação 3.8 que indica a porcentagem de indivíduos em B que são fracamente dominados por indivíduos de A.

$$C(A,B) = \frac{|b \in B; \exists a \in A : a \succeq b|}{|B|} \tag{3.8}$$

Como se pode observar, o valor de $C(A,B)$ está dentro do intervalo $[0,1]$. O valor $C(A,B) = 1$ indica que todas as soluções em B são fracamente dominadas por A. Por outro lado, $C(A,B) = 0$ indica que nenhuma das soluções em B são fracamente dominadas por A. É notório que,

se $C(A,B) = X$ e $C(B,A) = Y \nRightarrow X + Y = 1$, ou seja, $C(A,B)$ não é, necessariamente, igual $1 - C(B,A)$.

4 *O Planejamento Operacional de Lavra Abordado*

4.1 Introdução

O Problema de Planejamento de Lavra a Céu Aberto em Mineração envolve a alocação de máquinas e caminhões às frentes de lavra. As Figuras 4.1 e 4.2 e ilustram, respectivamente, um equipamento de carga abastecendo um caminhão em uma frente e um caminhão depositando o minério (ou estéril) no britador.

Figura 4.1: Equipamentos de carga e transporte

Cada frente de lavra contém uma determinada quantidade de material (minério ou estéril), com características físicas, químicas e econômicas diferenciadas, denominadas parâmetros de controle. Como exemplo típico de parâmetros de controle, tem-se: Fe, SiO_2, H_2O, Mn, P, granulometria. Para satisfazer as especificações exigidas pelos clientes, é necessário selecionar as frentes a serem lavradas e seu ritmo de lavra, os quais devem ser determinados proporcional-

Figura 4.2: Britador

mente. Para a operação de minério e estéril, a mina conta com uma frota limitada de equipamentos de carga, os quais devem ser alocados às frentes de lavra e operarem em uma faixa de produtividade que torne viável sua utilização (COSTA, 2005).

Considera-se que o transporte do material retirado da frente de lavra é realizado por uma frota de caminhões com capacidades de carga diferentes, a Figura 4.3 ilustra um modelo de caminhão para transporte de estéril e minério. Esses caminhões são alocados às frentes de lavra dinamicamente, tentando-se evitar a formação de filas, ou seja, o caminhão é alocado a um ponto de carga ou basculamento que proporcione o menor tempo de fila possível.

O ritmo de lavra é determinado pelas capacidades de operação dos equipamentos de carga e transporte alocados às diversas frentes. A Figura 4.4 mostra um tipo de equipamento de carga utilizado para o carregamento de minério e estéril na frente de lavra.

Em minas a céu aberto, são utilizados dois critérios para a alocação de caminhões: alocação estática e alocação dinâmica. No sistema de alocação dinâmica, os caminhões não ficam fixos a uma determinada rota, como no sistema de alocação estática. Eles podem ser direcionados a diferentes frentes de lavra, onde esteja um equipamento de carga compatível. Esta estratégia faz aumentar a produtividade da frota e proporciona, segundo Costa (2005), um aumento na capacidade de produção da mina ou mesmo a redução do número de equipamentos necessários para manter o mesmo nível de produção.

Figura 4.3: Modelo de Caminhão (ARAÚJO, 2008)

Figura 4.4: Modelo de Carregadeira - L1850 (ARAÚJO, 2008)

4.2 Características do Problema de Alocação Abordado

O problema abordado neste trabalho é o de Planejamento Operacional de Lavra com alocação dinâmica de caminhões (POLAD), sendo estes de capacidades diferentes.

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

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

Considera-se o planejamento para uma hora de produção, sendo este aplicado até uma frente exaurir ou ocorrer uma quebra de equipamento, situação na qual deve ser feito outro planejamento.

Dado o elevado custo de uma carregadeira, é imposto um limite mínimo de produção para cada carregadeira para justificar economicamente sua utilização.

Finalmente, considera-se uma taxa de utilização máxima para os caminhões. Por exemplo, supondo uma taxa de utilização máxima de 85%, um caminhão l de 80 t de capacidade, deveria trabalhar 51 (= 0,85 $\times$ 60) minutos, no máximo, em uma hora. Isso é adotado para retratar uma situação mais real, uma vez que um caminhão não fica todo o tempo em atividade. Além disso, essa taxa de utilização máxima tem por objetivo, também, modelar a variabilidade nos tempos de ciclo dos caminhões.

Por fim, a Figura 4.5 ilustra, de uma forma geral, o problema de planejamento operacional de lavra com alocação dinâmica de caminhões. Nota-se que não temos nenhuma carregadeira alocada na frente com teor de 50% Fe, situação que ocorre na prática, pois nem sempre o número de carregadeiras é suficiente para alocar um equipamento em cada frente de lavra. A Figura 4.5 mostra, também, que a frota de caminhões e carregadeiras é heterogênea, ou seja, é comum encontrar equipamentos de transporte e carga com diferentes capacidade e compatibilidade, ou seja, nem todo caminhão é compatível com o equipamento de carga alocado na frente de lavra.

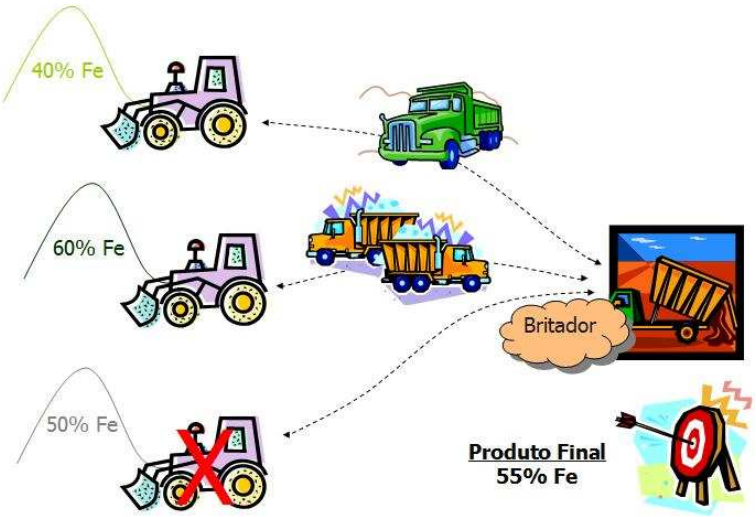

Figura 4.5: Exemplo de operação em uma mina a céu aberto

5 *Metodologia Heurística*

Neste capítulo são apresentados os métodos heurísticos propostos para resolver o POLAD multiobjetivo. A Seção 5.1 descreve como uma solução do POLAD é representada nos algoritmos desenvolvidos neste trabalho. A heurística utilizada para geração da solução inicial é apresentada na Seção 5.2. Na Seção 5.3 são apresentados os movimentos que constituem as estruturas de vizinhança utilizadas para resolução do problema. A Seção 5.4 mostra como uma solução é avaliada. A Seção 5.5 mostra os dois algoritmos desenvolvidos para resolver o MOPOLAD, o primeiro deles, denominado GRASP-MOVNS, combina os procedimentos *Greedy Randomized Adaptive Search Procedures* (GRASP) e *Multiobjective Variable Neighborhood Search* (MOVNS). O segundo algoritmo, denominado GRASP-NSGAII-PR, combina os procedimentos GRASP e *Nondominated Sorting Genetic Algorithm II* (NSGA-II), sendo que a fase de cruzamento do NSGA-II utiliza o procedimento de Reconexão por Caminhões (PR).

5.1 Representação de uma solução

Uma solução do POLAD é representada por uma matriz $R_{|F|\times(1+|V|)}$ de valores inteiros, sendo F o conjunto de frentes e V o conjunto de caminhões.

Para clareza de apresentação, a matriz $R_{|F|\times(1+|V|)}$ é decomposta em duas submatrizes Y e N, com $R = [Y|N]$, sendo $Y = (y_i)_{|F|\times 1}$ e $N = (n_{il})_{|F|\times|V|}$. A submatriz $Y_{|F|\times 1}$ representa a alocação dos equipamentos de carga ao conjunto F de frentes e o respectivo *status* de cada um desses equipamentos com relação ao fato de estarem ativos ou não. Em cada célula y_i da matriz $Y_{|F|\times 1}$ representa-se a carregadeira k alocada à frente i. Um valor D significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente i, a carregadeira k associada a tal frente é considerada *inativa* e não é penalizada por produção abaixo da mínima para este equipamento de carga, restrições da formulação de programação modelo matemática de Souza et al. (2010). A submatriz $N = (n_{il})_{|F|\times|V|}$ representa o número de viagens realizadas pelos caminhões l às frentes i. Um valor 0 (zero) significa que não há viagem para aquele caminhão, enquanto um valor X informa que há incompatibilidade entre o caminhão e a carregadeira

alocada àquela frente.

A Tabela 5.1 exemplifica uma solução para uma instância do problema. Nesta tabela, as linhas representam as frentes de lavra disponíveis no conjunto F, a coluna *CARGA* representa a alocação dos equipamentos de carga às frentes de lavra e as demais colunas indicam o número de viagens que serão realizadas pelo conjunto V de caminhões disponíveis.

Tabela 5.1: Exemplo de características de uma solução para o POLAD

	$Carga$	Cam_1	Cam_2	...	Cam_V
F_1	$< Car_1, 1 >$	8	X	...	X
F_2	$< D, 0 >$	0	0	...	0
F_3	$< Car_8, 0 >$	0	0	...	0
...	...	...	...	...	...
F_F	$< Car_5, 1 >$	0	9	...	3

Neste exemplo observa-se, na coluna CARGA, linha F_1, a dupla $\langle Car_1, 1\rangle$, indicando que o equipamento de carga Car_1 está alocado à frente F_1 e em operação. Na coluna CARGA, linha F_3, a dupla $\langle Car_8, 0\rangle$ indica que o equipamento de carga Car_8 está alocado à frente F_3, mas não está em operação. Observa-se, ainda, na coluna *CARGA*, linha F_2, o valor $\langle D, 0\rangle$ informando que não existe equipamento de carga alocado à frente F_2 e que, portanto, esta frente está disponível. As demais colunas representam o número de viagens a serem realizadas por um caminhão a uma frente, considerando a compatibilidade entre o caminhão e o equipamento de carga alocado à frente. As células com os valores X indicam incompatibilidade entre um caminhão e o respectivo equipamento de carga.

A partir de Y, N e dos tempos de ciclo dados na matriz $TC = (tc_{il})_{|F|\times|V|}$ são determinados o ritmo de lavra em cada frente e o somatório dos tempos de ciclo de cada caminhão.

5.2 Geração de uma solução inicial

Uma solução inicial para o problema é feita em duas etapas. Na primeira, realizamos a alocação das carregadeiras e a distribuição das viagens às frentes estéril, e na segunda, às frentes de minério. Esta estratégia é adotada tendo em vista que nas frentes de estéril o importante é atender à produção e não é necessário observar a qualidade.

Na primeira etapa utilizamos uma heurística gulosa, cujo pseudocódigo está descrito no Algoritmo 9.

Algoritmo 9: ConstróiSoluçãoEstéril

Entrada: T, S, W

Saída: Solução de estéril S_w

$T \leftarrow$ Conjunto de caminhões ordenados por suas capacidades (o primeiro é o de maior capacidade).

$S \leftarrow$ Conjunto de carregadeiras ordenadas pelas produtividades (a primeira é a de maior produtividade).

$W \leftarrow$ Conjunto de frentes de estéril ordenadas pelas massas (a primeira é a de maior massa).

enquanto *a produção de estéril for menor que a produção recomendada* ***e*** *existirem frentes de estéril não utilizadas* **faça**

  Selecione a primeira frente de estéril i do vetor W;

  se *não há carregadeira alocada à frente i* **então**

    se *Todas as carregadeiras estão alocadas* **então** Remova a frente i de W

    senão

      Atualize S_w alocando a maior carregadeira disponível à frente i ;

    fim

  fim

  se *A frente i não foi removida de W* **então**

    Encontre um caminhão $l \in T$ tal que: a) Seja compatível com a carregadeira alocada à frente i; b) Seja possível realizar mais uma viagem; c) Sua capacidade não viole a produção máxima da carregadeira;

    se *O caminhão l existe* **então** Atualize S_w, alocando a maior quantidade possível de viagens do caminhão l à frente de estéril i ;

    senão

      Remova a frente i de W ;

    fim

  fim

fim

Retorne S_w;

Na segunda etapa, utilizamos uma heurística que aplica *GRASPmax* vezes a fase de construção do procedimento *GRASP* e retorna a melhor das soluções construídas, desta feita incluindo-se as frentes de minério. A justificativa para esse procedimento é que a busca local de nosso algoritmo é muito custosa computacionalmente. Assim, a mesma requer uma boa solução inicial, o que, de acordo com (LOURENÇO; MARTIN; STÜTZLE, 2003), aceleraria a convergência para um ótimo local.

Para cada construção, utilizamos uma função guia g que relaciona os valores de desvio de qualidade em relação à meta. De acordo com esta função, é mais indicado selecionar uma frente

que minimize os desvio de qualidade dos parâmetros de controle.

Inicialmente, todas as frentes *i* candidatas são ordenadas de acordo com os valores de g_i e inseridas em uma lista de candidatos *LC*. De *LC* é extraída uma lista restrita de candidatos *LRC* contendo as frentes de minério mais bem qualificadas de acordo com a função guia. A cardinalidade desta lista, isto é, $\lceil \gamma \times |LC| \rceil$ é definida pelo parâmetro $\gamma \in [0,1]$. A estratégia utilizada para escolher uma frente *i* consiste em atribuir, primeiramente, uma classificação probabilística para cada frente candidata da *LRC*. A função $bias(r) = 1/(r)$ é associada à frente que está na *r*-ésima posição na classificação. Cada frente candidata é, então, escolhida com probabilidade $p(r) = bias(r)/\sum_{i=1,\cdots,|LRC|} bias(i)$. Em seguida, o algoritmo escolhe aleatoriamente uma frente de minério *i* de *LRC*, adicionando-a à solução parcial. O Algoritmo 10 descreve este procedimento de construção.

Algoritmo 10: ConstróiSoluçãoMinério

Entrada: S_w, γ, g, T, S

Saída: Solução S_0

$S_0 \leftarrow S_w$

$T \leftarrow$ Conjunto de caminhões ordenados pelas suas capacidades (o primeiro é o de menor capacidade).

$S \leftarrow$ Conjunto de carregadeiras ordenadas por suas produtividades (a primeira é a que de maior produtividade).

enquanto *a produção de minério for menor que a produção recomendada* **e** *existirem frentes de minério não utilizadas* **faça**

- $LC \leftarrow$ Conjunto de frentes de minério ordenadas de acordo com a função g ;
- $|LRC| = \lceil \gamma \times |LC| \rceil$;
- Selecione uma frente $i \in LRC$ de acordo com a função *bias*;
- **se** *não há carregadeira alocada à frente i* **então**
  - **se** *Todas as carregadeiras estão alocadas* **então** Remova a frente i de LC
  - **senão**
    - Atualize S_0 alocando a carregadeira de maior capacidade à frente i ;
  - **fim**
- **fim**
- **se** *A frente i não foi removida de LC* **então**
  - Encontre um caminhão $l \in T$ tal que: a) Seja compatível com a carregadeira alocada à frente i; b) Seja possível realizar mais uma viagem; c) Sua capacidade não viole a produção máxima da carregadeira;
  - **se** *O caminhão l existe* **então** Atualize S_0, alocando a maior quantidade possível de viagens do caminhão l à frente de estéril i ;
  - **senão**
    - Remova a frente i de W ;
  - **fim**
- **fim**

fim

Retorne S_0;

5.3 Estruturas de vizinhança

Para explorar o espaço de soluções do POLAD foram desenvolvidos 8 tipos diferentes de movimentos, apresentados a seguir, para definir oito estruturas de vizinhança $N^k(s)$. Os seis primeiros movimentos, e suas devidas estruturas de vizinhança, foram propostos em Costa (2005). Souza et al. (2010) propôs as demais vizinhanças e relatou a boa capacidade explo-

ratória desses movimentos.

Uma breve descrição dos movimentos segue abaixo:

Movimento Número de Viagens - $N^{NV}(s)$: Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

Movimento Carga - $N^{CG}(s)$: Consiste em trocar duas células distintas y_i e y_k da matriz Y, ou seja, trocar os equipamentos de carga que operam nas frentes i e k, caso as duas frentes possuam equipamentos de carga alocados. Havendo apenas uma frente com equipamento de carga, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas junto com as frentes escolhidas.

Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$: Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl}. Assim, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$: Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik}. Isto é, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Operação Frente - $N^{OF}(s)$: Operation consists of removing the loading equipment that is operating in front of i . The move removes all travel made on this front, leaving equipment inactive. The machine returns to operation so a new route is associated with it.

Movimento Operação Caminhão - $N^{OC}(s)$: Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, isto é, retirar de atividade um caminhão l que esteja operando em uma frente i.

Movimento Troca de Viagens - $N^{VT}(s)$: Duas células da matriz N são selecionadas e uma unidade de uma célula passa para a outra, isto é, uma viagem de um caminhão a uma frente passa para outro caminhão a outra frente.

Movimento Troca de Carregadeiras - $N^{CT}(s)$: Duas células distintas y_i e y_k da matriz Y tem

seus valores permutados, ou seja, os equipamentos de carga que operam nas frentes i e k são trocados. Analogamente ao movimento *CG*, os equipamentos de carga são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

A Figura 5.1 mostra uma possível solução para o problema.

$$s = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

Figura 5.1: Exemplo de Solução para o POLAD

Desta forma, a Figura 5.2 mostra como ficaria a solução da figura 5.1 após uma aplicação aleatória de cada um dos movimentos descritos.

$$s \oplus m^{NV} = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & {}^*4 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{CG} = \left[\begin{array}{r|rrrr} {}^*3 & {}^*1 & {}^*0 & {}^*3 & {}^*2 \\ -1 & 0 & 0 & 0 & 0 \\ {}^*1 & {}^*2 & {}^*4 & {}^*3 & {}^*0 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

$$s \oplus m^{VC} = \left[\begin{array}{r|rrrr} 1 & 2 & {}^*3 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & {}^*1 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{VF} = \left[\begin{array}{r|rrrr} 1 & {}^*1 & {}^*5 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

$$s \oplus m^{OF} = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & {}^*0 & {}^*0 & {}^*0 & {}^*0 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{OC} = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & {}^*0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

$$s \oplus m^{VT} = \left[\begin{array}{r|rrrr} 1 & {}^*1 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & {}^*4 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{CT} = \left[\begin{array}{r|rrrr} {}^*3 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ {}^*1 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

Figura 5.2: Exemplo de aplicação dos movimentos

5.4 Função de avaliação

Na abordagem multiobjetivo, temos os seguintes objetivos conflitantes:

- Minimizar o número de caminhões necessários para o processo de produção
- Minimizar os desvios de produção
- Minimizar os desvios dos parâmetros de qualidade

Na abordagem mono-objetivo do POLAD, a função objetivo é dada pela equação (5.1), que representa a soma ponderada desses três objetivos:

$$f^{MP}(s) = \sum_{j \in T} \lambda_j^- d_j^- + \sum_{j \in T} \lambda_j^+ d_j^+ + \alpha^- P_m^- + \alpha^+ P_m^+ + \beta^- P_e^- + \beta^+ P_e^+ + \sum_{l \in V} \omega_l U_l \tag{5.1}$$

Como uma solução s pode não respeitar todas as restrições, penalizamos uma solução de acordo com a equação (5.2):

$$P = f^p(s) + \sum_{j \in P} f_j^q(s) + \sum_{l \in T} f_l^u(s) + \sum_{k \in S} f_k^c(s) \tag{5.2}$$

em que:

$f^p(s)$ avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril;

$f_j^q(s)$ avalia s quanto à inviabilidade em relação ao j-ésimo parâmetro de controle;

$f_l^u(s)$ avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l-ésimo caminhão;

$f_k^c(s)$, que avalia s quanto ao desrespeito aos limites de produtividade da carregadeira k.

Logo, obtêm-se a função de avaliação dada pela Eq. (5.3):

$$f(s) = f^{MP}(s) + P \tag{5.3}$$

Na equação 5.3, a primeira parcela é a função objetivo propriamente dita, $f^{MP}(s)$ (equação do modelo de programação matemática de Souza et al. (2010)), e a segunda é composta pelas funções que penalizam a ocorrência de inviabilidade na solução corrente. Assim, a função f mensura o desvio dos objetivos considerados e penaliza o não atendimento às restrições do problema.

Decompondo a função (5.1) em três objetivos conflitantes, temos:

$$f^{MP}(s) = \underbrace{\sum_{j\in T} \lambda_j^- d_j^- + \sum_{j\in T} \lambda_j^+ d_j^+}_{z_1(.)} + \underbrace{\alpha^- P_m^- + \alpha^+ P_m^+ + \beta^- P_e^- + \beta^+ P_e^+}_{z_2(.)} + \underbrace{\sum_{l\in V} \omega_l U_l}_{z_3(.)} \tag{5.4}$$

sendo,

- Qualidade dos parâmetros de qualidade na mistura

$$z_1(s) = \sum_{j\in T} \lambda_j^- d_j^- + \sum_{j\in T} \lambda_j^+ d_j^+ \tag{5.5}$$

- Desvios de Produção

$$z_2(s) = \alpha^- P_m^- + \alpha^+ P_m^+ + \beta^- P_e^- + \beta^+ P_e^+ \tag{5.6}$$

- Utilização dos Caminhões

$$z_3(s) = \sum_{l\in V} \omega_l U_l \tag{5.7}$$

Por fim, segue a função de avaliação z, descrita pela equação (5.8), com os três objetivos a serem minimizados:

$$z(s) = (z_1(s) + P, z_2(s) + P, z_3(s) + P) \tag{5.8}$$

5.5 Algoritmos propostos

São propostos, neste trabalho, dois algoritmos. O primeiro deles, denominado GRASP-MOVNS, consiste na combinação de procedimentos heurísticos *Greedy Randomized Adaptative Search Procedure* - GRASP e *Multiobjective Variable Neighborhood Search* (MOVNS).

O segundo, denominado GRASP-NSGAII-PR, combina os procedimentos GRASP, *Nondominated Sorting Genetic Algorithm II* - NSGA-II e Reconexão por Caminhos (PR, do inglês *Path Relinking*).

O pseudocódigo do algoritmo GRASP-MOVNS está esquematizado no Algoritmo 11.

Algoritmo 11: GRASP-MOVNS

Entrada: Vizinhanças $N_k(x)$; $graspMax$; $levelsMax$

Saída: Aproximação de um conjunto eficiente Xe

1 **para** $i \leftarrow 1$ **até** $graspMax$ **faça**
2 $s_w \leftarrow$ ConstróiSoluçãoEstéril()
3 Gere um número aleatório $\gamma \in [0,1]$
4 $s_i \leftarrow$ ConstróiSoluçãoMinério(s_w, γ)
5 addSolution($Xe, s_i, f(s_i)$)
6 **fim**
7 $level \leftarrow 1$; $shaking \leftarrow 1$
8 **enquanto** *Critério de parado não satisfeito* **faça**
9 Seleciona uma solução não visitada $s \in Xe$
10 Marque s como *visitada*
11 $s' \leftarrow s$
12 **para** $i \leftarrow 1$ **até** $shaking$ **faça**
13 Selecione aleatoriamente uma vizinhança $N_k(.)$
14 $s' \leftarrow$ Perturbação(s', k)
15 **fim**
16 Seja $k_{ult} \leftarrow k$
17 $incrementa \leftarrow verdadeiro$
18 **para todo** $s'' \in N_{k_{ult}}(s')$ **faça**
19 addSolution($Xe, s'', f(s''), Added$)
20 **se** $Added$ = $verdadeiro$ **então**
21 $incrementa \leftarrow falso$;
22 **fim**
23 **fim**
24 **se** $incrementa = verdadeiro$ **então**
25 $level \leftarrow level + 1$
26 **senão**
27 $level \leftarrow 1$; $shaking \leftarrow 1$
28 **fim**
29 **se** $level >= levelMax$ **então**
30 $shaking \leftarrow shaking + 1$; $level \leftarrow 1$
31 **fim**
32 **se** *todo* $s \in Xe$ *estão malgMOVNSarcadas como visitadas* **então**
33 Marque todos $s \in Xe$ como *não-visitado*
34 **fim**
35 **fim**
36 **retorna** Xe

Uma solução inicial (linha 4 do Algoritmo 11) é gerada pelo procedimento parcialmente

guloso GRASP, conforme descrito na Seção 5.2. O procedimento addSolution (linha 5 do Algoritmo 11), já detalhado na Seção 3.3.4, adiciona as soluções criadas pelo procedimento GRASP na população X_e. As linhas 9 e 10 selecionam um indivíduo da população de soluções eficientes e marca este indivíduo como "visitado". Quando todos os indivíduos estão com este marcador, a linha 33 retira estes marcadores.

As variáveis *shaking* e *level*, linha 7 do Algoritmo 11, regulam a perturbação utilizada neste algoritmo. Essa versão do algoritmo MOVNS, proposta neste trabalho, possui um mecanismo que regula o nível de perturbação do algoritmo, ou seja, a variável *shaking* é incrementada quando o algoritmo passa um determinado tempo sem obter boas soluções. Da linha 12 até 15 do Algoritmo 11 ocorre o laço de perturbação do algoritmo. No mecanismo utilizado, quanto maior o valor da variável *shaking*, mais a solução será perturbada. Para cada unidade dessa variável seleciona-se, aleatoriamente, uma vizinhança entre as seis seguintes: N^{NV}, N^{CG}, N^{VC}, N^{VF}, N^{VT} e N^{CT}. Em seguida, aplica-se um movimento de perturbação. A linha 30 retorna os valores das variáveis *level* e *shaking* para uma unidade quando, pelo menos, uma solução é adicionada ao conjunto eficiente.

Por fim, o procedimento `addSolution` (linhas 5 e 19 do Algoritmo 11) já foi detalhado na Seção 3.3.4.

O pseudocódigo do algoritmo GRASP-NSGAII-PR está esquematizado no Algoritmo 12.

Algoritmo 12: GRASP-NSGAII-PR

Entrada: Tamanho da população N; Vizinhanças $N_k(x)$

Saída: Conjunto Eficiente Xe

1 População inicial P_0
2 **enquanto** $|P_0| < N$ **faça**
3 $\quad s_w \leftarrow$ ConstróiSoluçãoEstéril()
4 $\quad$ Gere um número aleatório $\gamma \in [0,1]$
5 $\quad s_i \leftarrow$ ConstróiSoluçãoMinério(s_w, γ)
6 $\quad$ addSolution($P_0, s_i, f(s_i)$)
7 **fim**
8 $Q_0 \leftarrow$ SelecaoCruzamentoPRMutacao(P_0, Vizinhanças $N^{(k)}(.)$)
9 $Xe \leftarrow$ Non-dominated Sorting Genetic Algorithm II(P_0, Q_0, N, SelecaoCruzamentoPRMutacao(...))
10 **retorna** Xe

Analogamente ao Algoritmo 11, a população inicial P_0 do Algoritmo 12 é iniciada adi-

cionando-se indivíduos gerados pelo procedimento GRASP (linha 5 do Algoritmo 12), porém, neste caso, o critério de parada do procedimento GRASP é o tamanho da população P_0 (linha 2 do Algoritmo 12).

A linha 8 do Algoritmo 12 aciona o procedimento `SelecaoCruzamentoPRMutacao`. Como citado na Seção 3.2.2, Ribeiro e Resende (2012) apresentam a técnica de Reconexão por Caminhos como um operador avançado de recombinação ou cruzamento. Neste trabalho, a Reconexão por Caminhos foi utilizada como um operador de cruzamento. O Algoritmo 13 exemplifica esse procedimento de seleção, cruzamento e mutação.

Por fim, a linha 9 do Algoritmo 12 ativa o procedimento *Non-dominated Sorting Genetic Algorithm II*, descrito no Algoritmo 5 e detalhado Seção 3.3.4. As etapas de "seleção, cruzamento e mutação" são substituídas pelo procedimento `SelecaoCruzamentoPRMutacao`.

Algoritmo 13: SelecaoCruzamentoPRMutacao

Entrada: População de pai P; Vizinhanças $N^{(k)}(.)$; Parâmetros da mutação $mutationRate$ e $localSearchRate$

Saída: População de *offprings* Q

```
1  enquanto |Q| < N faça
2      Selecione dois indivíduos aleatórios s1 e s2 ∈ P
3      s ← melhor indivíduo (s1, s2, ReconexãoPorCaminhos(s1,s2),
       ReconexãoPorCaminhos(s2,s1))
4      addSolution(Q, s, f(s))
5      Gere um número aleatório ap_mutation ∈ [0,1]
6      se ap_mutation < mutationRate então
7          Selecione aleatoriamente uma vizinhança N^(k)(.)
8          s' ← N^(k)(s)
9      senão
10         s' ← s
11     fim
12     Gere um número aleatório ap_localSearch ∈ [0,1]
13     se ap_localSearch < localSearchRate então
14         s'' ← VND(s')
15         addSolution(Q, s'', f(s''))
16     senão
17         addSolution(Q, s', f(s'))
18     fim
19 fim
20 retorna Q
```

O Algoritmo 13 realiza a aplicação dos operadores genéticos, ou seja, dada uma população de pais P, esse procedimento faz a seleção, cruzamento e mutação, de forma a obter uma população de *offprings* Q.

Na linha 3 do Algoritmo 13 o indivíduo s recebe o melhor indivíduo (considerando a função de avaliação mono-objetivo descrita na seção 5.4) entre os dois indivíduos selecionados aleatoriamente, s_1 e s_2, ou um dos dois indivíduos retornados pelo procedimento de Reconexão por Caminhos, ou seja, a Reconexão por Caminhos é acionada tanto com o indivíduo s_1 sendo a solução base quanto sendo a solução guia. A abordagem de Reconexão por Caminhos utilizada é baseada na fixação de atributos, sendo considerado como atributo a posição que uma car-

regadeira ocupa em uma solução. A cada passo do procedimento, verificamos se a carregadeira k alocada na frente de lavra i da solução guia é igual à carregadeira alocada na frente i da solução base. Caso a carregadeira seja diferente, move-se a carregadeira k da solução base para a frente i, permanecendo as viagens que eram associadas a esta carregadeira. Desta forma, são mantidos os critérios de compatibilidade entre as carregadeiras e os caminhões. Então, realiza-se uma busca local baseada no procedimento VND (descrito na Seção 3.2.2), utilizando apenas as estruturas que modificam o número de viagens dos caminhões, ou seja, as vizinhanças: N^{NV}, N^{VC} e N^{VF}, preservando assim as carregadeiras já fixadas. Para prosseguir para o passo seguinte, listamos todos estes movimentos e aplicamos aquele que possui melhor resultado considerando a função de avaliação mono-objetivo. Este procedimento se repete até que todos os atributos sejam fixados, ou seja, para cada frente i a carregadeira alocada na solução base seja igual a carregadeira alocada na solução guia.

Já a linha 8 do Algoritmo 13 aplica uma mutação no indivíduo s caso a variável aleatória $ap_{mutation}$ seja menor que a taxa de mutação $mutationRate$. O mesmo acontece na linha 14, aplica-se o procedimento VND caso uma condição análoga seja satisfeita. Finalmente, as linhas 4, 15 e 17 do Algoritmo 13 verificam se os indivíduos s, s' e s'' devem ser adicionadas à população de *offprings* Q.

6 *Resultados*

Descrevem-se, neste capítulo, os resultados dos algoritmos heurísticos GRASP-NSGAII-PR e GRASP-MOVNS propostos. Na Seção 6.1 são descritos os problemas-teste usados para comparar o desempenho dos algoritmos. Na Seção 6.2 são apresentados os valores dos parâmetros e pesos utilizados. Na Seção 6.3 é apresentado o ambiente de desenvolvimento dos algoritmos, bem como o de teste dos algoritmos. Na Seção 6.4 são mostrados os resultados computacionais dos algoritmos.

6.1 Descrição dos problemas-teste

Para testar os algoritmos desenvolvidos, foi usado um conjunto de 8 problemas-teste da literatura, disponíveis em http://www.iceb.ufop.br/decom/prof/marcone/projects/mining.html. Estes problemas-teste foram os mesmos utilizados em Souza et al. (2010) para validar o algoritmo *GGVNS*.

Os melhores resultados da literatura para os problemas-teste analisados são apresentados na Tabela 6.1. Na coluna "Opt." indicamos por "$\surd$" as instâncias nas quais o otimizador matemático CPLEX 11.02 obteve o valor ótimo da função.

Tabela 6.1: Melhores valores

Problema-Teste	Melhor da Literatura	Opt†
opm1	227.12	
opm3	256.37	
opm2	164,027.15	$\surd$
opm4	164,056.68	$\surd$
opm5	227.04	
opm6	236.58	
opm7	164,017.46	$\surd$
opm8	164,018.65	$\surd$

6.2 Pesos e parâmetros utilizados

Após uma bateria preliminar de testes, os seguintes valores para os parâmetros dos algoritmos desenvolvidos foram fixados. Para o Algoritmo GRASP-MOVNS (Algoritmo 11):

- *graspMax*: 300;
- *levelMax*: 10;
- *shaking*: 5.

Para o Algoritmo GRASP-NSGAII-PR (Algoritmo 12):

- *mutationRate*: 10%;
- *localSearchRate*: 40%.

Os pesos adotados na função de avaliação são apresentados na Tabela 6.2 e são os mesmos de Costa (2005).

Tabela 6.2: Pesos adotados

Pesos	**Descrição**	**Valor**
γ	Penalidade por tonelada abaixo dos limites inferiores ou acima dos limites superiores de produção (estéril/minério)	1000
α	Penalidade por tonelada abaixo ou acima da meta de produção (estéril/minério)	100
Φ_j	Penalidade por tonelada abaixo do limite mínimo de especificação, ou acima do limite de especificação para os parâmetros de controle j	100
λ	Penalidade por tonelada abaixo ou acima da meta de qualidade	1
ω	Penalidade pelo uso de um caminhão	1
Tx_l	Taxa máxima de utilização de um caminhão	85%
Ω^+	Penalidade por utilização acima da taxa máxima de utilização de um caminhão	1000
Ψ_k^-	Penalidade por tonelada abaixo do limite mínimo de produção, ou acima do limite de produção para a carregadeira k	1000

6.3 Ambiente de desenvolvimento

Os experimentos foram testados em um microcomputador DELL XPS 8300 Intel Core i7-2600, 8MB Cache, 3.4GHz, 16GB RAM, sob sistema operacional Ubuntu 10.10.

Os algoritmos foram implementados em C++ com auxílio do *framework* OptFrame [1] (COELHO

[1] disponível em http://sourceforge.net/projects/optframe/

et al., 2011, 2010). Essa escolha foi devido a seu arcabouço de fácil utilização, que inclui:

- Representações de soluções e populações;
- Arcabouços de métodos heurísticos e metaheurísticos;
- Inferface para análise dos resultados;
- Fórum de discussão com os colaboradores do projeto.

O *framework* OptFrame é, basicamente, uma estrutura computacional que agiliza o desenvolvimento de algoritmos heurísticos. O objetivo do *framework* é fornecer uma simples interface em C++ para componentes comuns de metaheurísticas baseadas em trajetória e população, aplicadas a problemas de otimização combinatória. Uma vez que muitos métodos são comuns na literatura, o OptFrame fornece um implementação eficiente para versões simples de algumas heurísticas e metaheurísticas, todavia, o usuário pode desenvolver versões "mais inteligentes"e mais robustas, aplicadas ao seu problema específico. Além disso, o OptFrame possui suporte ao desenvolvimento de sistemas em paralelo, tanto para memória compartilhada quanto para memória distribuída. OptFrame tem sido aplicado com sucesso para modelar e resolver vários problemas combinatórios, mostrando um bom equilíbrio entre a flexibilidade e eficiência.

A licença de uso do OptFrame é a **LGPLv3**.

A LGPL acrescenta restrições ao código fonte desenvolvido, mas não exige que seja aplicada a outros softwares que empreguem seu código, desde que este esteja disponível na forma de uma biblioteca.

A principal diferença entre a GPL e a LGPL é que esta permite também a associação com programas que não estejam sob as licenças GPL ou LGPL, incluindo Software proprietário.

Conforme descrito na *Wikipedia*: "*The main difference between the GPL and the LGPL is that the latter can be linked to (in the case of a library,* ***used by****) a non-(L)GPLed program, and regardless of whether it is free software or proprietary software. This non-(L)GPLed program can then be distributed under any chosen terms if it is not a derivative work*".

O código-fonte do OptFrame bem como uma documentação mais detalhada do *framework* encontram-se disponíveis em *http://sourceforge.net/projects/optframe*, sob licença **GNU LGPLv3**.

6.4 Resultados e análise

Primeiramente, foi realizada uma bateria de testes de forma a validar o algoritmo GRASP-NSGAII-PR. Três variantes desse algoritmo foram propostas, sendo que cada uma dessas variantes corresponde a diferentes tamanhos da população. Para a primeira variante, denominada GRASP-NSGAII-PR-20, o tamanho da população foi fixado em 20. Para a segunda variante, denominada GRASP-NSGAII-PR-35, o tamanho da população foi fixado em 35. Já a terceira variante, denominada GRASP-NSGAII-PR-65, teve o tamanho da população fixado em 65 indivíduos. Todos os oito problemas-testes foram executados 30 vezes pelos algoritmos, com um tempo computacional de 2 minutos (visto que este tempo de execução é o mais indicado para uma aplicação real).

As tabelas 6.3 e 6.4 apresentam os resultados obtidos pelos algoritmos GRASP-NSGAII-PR-20, GRASP-NSGAII-PR-35 e GRASP-NSGAII-PR-65 em relação à função de avaliação dada pela Equação (5.8), à página 54. Nesta primeira etapa de análise, apenas as métricas de espaçamento e hipervolume foram utilizadas, ambas descritas na Seção 3.3.5.

Para as tabelas 6.3 e 6.4, a coluna "Instância" indica o problema-teste utilizado. A coluna "Melhor" indica o melhor valor da métrica analisada obtido nas 30 execuções. As colunas "Média" e "Desv.Padrão" indicam a média e desvio-padrão da amostra.

A Tabela 6.5 apresenta a média das cardinalidades das frentes de pareto obtidas pelas variantes e pelo algoritmo GRASP-MOVNS, também desenvolvido neste trabalho. Desta forma, os valores indicados nesta tabela são a soma do número de soluções não-dominadas obtidas em cada execução dividido pelo número de execuções (no caso, 30).

A Tabela 6.3 apresenta os valores da métrica de hipervolume para as três variantes do algoritmo GRASP-NSGAII-PR. Analisando-a, percebe-se que para as instâncias opm1, opm2, opm3, opm5 e opm8 a variante GRASP-NSGAII-PR-20 obteve o maior número de melhores valores para a métrica em questão; porém, a variante GRASP-NSGAII-PR-35 apresentou o maior número de valores médios e os menores desvios padrão. A variante GRASP-NSGAII-PR-65 conseguiu obter a melhor média e o melhor valor para a instância opm6.

A Tabela 6.4 mostra os resultados da comparação entre as variantes utilizando a métrica de Espaçamento. Novamente, a variante GRASP-NSGAII-PR-35 obteve os melhores resultados na média e os menores desvios padrão. A variante GRASP-NSGAII-PR-65 obteve os melhores valores de espaçamento, como pode ser visto nas instâncias opm1, opm3, opm4, opm5, opm6 e opm8. Todavia, analisando a Tabela 6.5, percebemos que a variante GRASP-NSGAII-PR-65 obteve frentes de pareto com poucos indivíduos quando comparada com outros algoritmos de-

Tabela 6.3: GRASP-NSGAII-PR-20 × GRASP-NSGAII-PR-35 × GRASP-NSGAII-PR-65: Hipervolume

Instância	Tempo (min)	GRASP-NSGAII-PR-20			GRASP-NSGAII-PR-35			GRASP-NSGAII-PR-65		
		Melhor	Média	Desv. Padrão	Melhor	Média	Desv. Padrão	Melhor	Média	Desv. Padrão
opm1	2	**162466396,8**	129870849	12180709	155562826	**137868212**	11104475	151925082	130764556	**8236198**
opm2	2	**99860690,8**	78846453	8025046	96286396	85468260	6009733	95653888	**86279181**	**4626825**
opm3	2	**3476415800**	**2924383550**	352347583	3353658800	2887700217	**260685669**	2433281400	1729772773	267983635
opm4	2	3380314000	2636623070	438538452	**3389246700**	**2650979457**	383861371	2284418900	1814165213	**219706068**
opm5	2	**88496949,6**	70251866	7270402	81906283	**74301286**	**5152934**	94471010	73650530	6290286
opm6	2	111113096	94360509	8423115	114901451	99513155	**7310672**	**121875341**	**100057066**	8153750
opm7	2	837147300	741268390	**54043613**	**907517500**	**747806017**	113074359	648157300	518369350	79105846
opm8	2	**1219130100**	1063714943	106924891	1192355600	**1070222527**	**95202467**	898823200	729660120	106293138

Tabela 6.4: GRASP-NSGAII-PR-20 × GRASP-NSGAII-PR-35 × GRASP-NSGAII-PR-65: Espaçamento

Instância	Tempo (min)	GRASP-NSGAII-PR-20			GRASP-NSGAII-PR-35			GRASP-NSGAII-PR-65		
		Melhor	Média	Desv. Padrão	Melhor	Média	Desv. Padrão	Melhor	Média	Desv. Padrão
opm1	2	2510,81	11761,6	8210,3	2589,54	**6924,4**	**5007,2**	**1659,79**	7731,5	5872,2
opm2	2	2172,8	8190,8	6106,2	**1812,14**	6557,2	5030,7	2081,74	**5198,4**	**2674,6**
opm3	2	10626,5	16365,4	**3736,4**	4850,5	**14368,6**	5173,6	**0**	16547,9	12236,1
opm4	2	11550,5	**18200,8**	7931	4865,89	18295,3	**7046,6**	**0**	19781,9	28634,9
opm5	2	2751,91	11210,1	7047	2578,39	**7544,5**	**5529,4**	**1906,21**	8124,1	6340,2
opm6	2	2080,01	9494,4	8071,5	2218,05	5963,8	**4650,8**	**1878,61**	**5746,6**	5326,5
opm7	2	9395,22	18402,7	5493,7	**4924,61**	**14673,6**	**4646,0**	4995,21	16309,8	7041,8
opm8	2	9880,23	19584,6	**5308,2**	4974,85	**14874,7**	5897,7	**0**	16017,4	8068,0

Tabela 6.5: Média do número de soluções não-dominadas obtidas

Instância	GRASP-MOVNS	GRASP-NSGAII-PR-20	GRASP-NSGAII-PR-35	GRASP-NSGAII-PR-65
opm1	39,93	8,10	12,00	11,17
opm2	64,23	8,14	13,04	13,63
opm3	26,20	11,34	13,8	5,97
opm4	24,40	9,94	11,47	5,4
opm5	41,27	8,37	11,00	11,37
opm6	61,43	8,47	12,47	13,30
opm7	26,33	11,34	13,94	7,50
opm8	23,63	11,04	13,9	7,90

senvolvidos. Este resultado forçou uma análise mais aprofundada nas frentes de pareto geradas por essa variante. Desta forma, foi observado que as frentes de pareto onde o espaçamento obtido foi igual a 0, eram frentes compostas por apenas dois indivíduos.

Para a segunda análise de resultados, o algoritmo GRASP-MOVNS foi comparado com a variante do algoritmo GRASP-NSGAII-PR que obteve o melhor desempenho em termos de convergência e diversidade, ou seja, a variante GRASP-NSGAII-PR-35. As tabelas 6.6 e 6.7 indicam os valores obtidos pelos algoritmos utilizando as métricas de Hipervolume, Espaçamento e Cobertura.

Analisando-se as tabelas 6.6 e 6.7 percebe-se que o algoritmo GRASP-MOVNS obteve melhores resultados em todas as instâncias. Observando a Tabela 6.6, nota-se que o algoritmo GRASP-MOVNS foi capaz de obter um melhor convergência e diversidade, obtendo as melhores médias para as métricas de hipervolume e espaçamento. A Tabela 6.7 indica a hegemonia desse algoritmo em relação ao algoritmo GRASP-NSGAII-PR, apontando execuções com valores de cobertura iguais a 1 e médias acima ou iguais a 0,66.

A última bateria de testes buscou verificar se esta nova proposta multiobjetivo conseguiria uma boa solução mono-objetivo, de acordo com a função mono-objetivo descrita pela Equação (5.1). A Tabela 6.8 mostra as melhores soluções mono-objetivo obtidas pelo algoritmo GRASP-MOVNS comparadas com o algoritmo GGVNS, estado-da-arte, de Souza et al. (2010).

Analisando-se a Tabela 6.8 percebemos que o algoritmo GRASP-MOVNS mostrou-se competitivo com o algoritmo mono-objetivo da literatura GGVNS, obtendo melhores soluções que o algoritmo GGVNS em quatro instâncias.

Por fim, as Figuras 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7 e 6.8 mostram as frentes de pareto obtidas com a união de todas as soluções não-dominadas encontradas nos testes citados anteriormente.

Tabela 6.6: GRASP-MOVNS × GRASP-NSGAII-PR-35: Hipervolume e Espaçamento

Instância	Tempo (min)	Espaçamento					Hipervolume				
		GRASP-MOVNS			GRASP-NSGAII-PR-35		GRASP-MOVNS			GRASP-NSGAII-PR-35	
		Melhor	Média	Desv. Padrão	Média	Desv. Padrão	Melhor	Média	Desv. Padrão	Média	Desv. Padrão
opm1	2	1662,54	**2868,68**	**1099,72**	6924,42	5007,23	**209208790**	**181862424**	**10968178**	137868212	11104475
opm2	2	**1066,13**	**2041,59**	**970,75**	6557,2	5030,65	**139741477**	**120880574**	7250650	85468260	**6009733**
opm3	2	2250,96	**6188,99**	**3331,59**	14368,62	5173,6	**3919537275**	**3515593945**	287661442	2887700217	**260685669**
opm4	2	2411,89	**8232,88**	**3957,29**	18295,27	7046,64	**4075530050**	**3522173300**	**325184991**	2650979457	383861371
opm5	2	**1400,84**	**2563,72**	**1057,14**	7544,48	5529,35	**108083430**	**97329268**	5574432	74301286	**5152934**
opm6	2	**1087,95**	**2025,84**	**695,00**	5963,77	4650,8	**158114999**	**140950355**	8965616	99513155	**7310672**
opm7	2	**2917,66**	**8311,12**	**3437,74**	14673,57	4646,02	**1003006550**	**860023190**	**67612990**	747806017	113074359
opm8	2	2258,14	**8045,67**	**4684,92**	14874,73	5897,7	**1374062875**	**1148710870**	127612341	1070222527	**95202467**

Tabela 6.7: GRASP-MOVNS × GRASP-NSGAII-PR-35: Cobertura

Instância	Tempo (min)	Cobertura					
		C(GRASP-MOVNS,GRASP-NSGAII-PR-35)			C(GRASP-NSGAII-PR-35,GRASP-MOVNS)		
		Melhor	Média	Desv. Padrão	Melhor	Média	Desv. Padrão
opm1	2	**1,00**	**0,90**	0,12	0,92	0,04	0,17
opm2	2	**1,00**	**0,86**	0,09	0,10	0,01	0,02
opm3	2	**1,00**	**0,75**	0,19	0,21	0,04	0,07
opm4	2	**1,00**	**0,80**	0,15	0,12	0,02	0,04
opm5	2	**1,00**	**0,88**	0,11	0,070	0,01	0,02
opm6	2	**1,00**	**0,80**	0,16	0,08	0,02	0,03
opm7	2	**1,00**	**0,66**	0,25	0,40	0,08	0,11
opm8	2	**1,00**	**0,66**	0,19	0,79	0,09	0,16

Tabela 6.8: Comparação de resultados: *MOVNS* × *GGVNS*

Instância	Tempo	MOVNS	GGVNS
		Melhor	Melhor
opm1	2	**228,12**	230,12
opm2	2	257,758	**256,37**
opm3	2	164051	**164039,12**
opm4	2	164111	**164099,66**
opm5	2	**227,16**	228,09
opm6	2	239,708	**236,58**
opm7	2	**164019**	164021,28
opm8	2	**164022**	164023,73

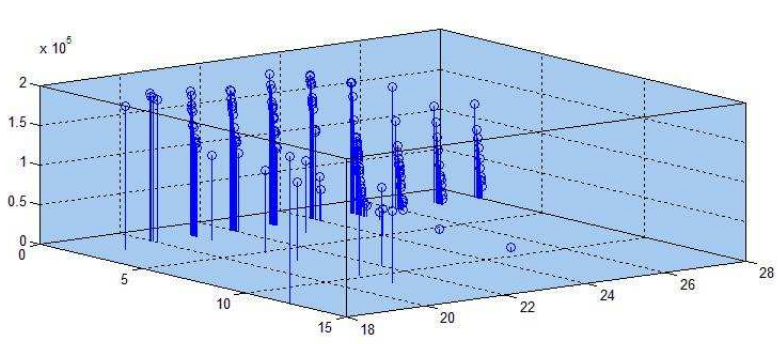

Figura 6.1: Frente de Pareto com 141 soluções não-dominadas - opm1

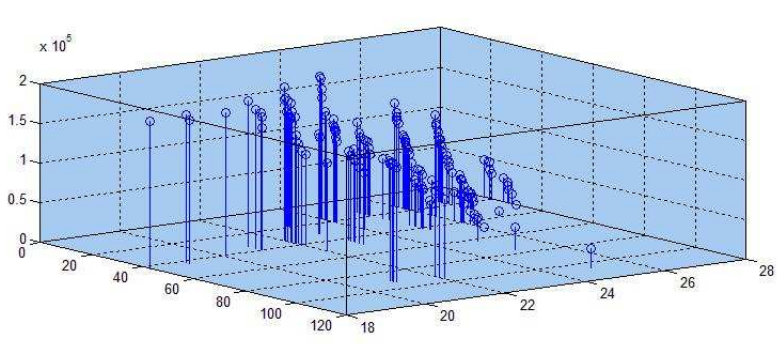

Figura 6.2: Frente de Pareto com 146 soluções não-dominadas - opm2

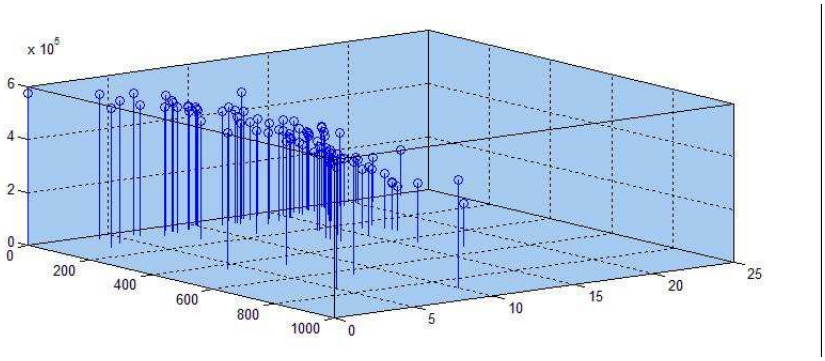

Figura 6.3: Frente de Pareto com 83 soluções não-dominadas - opm3

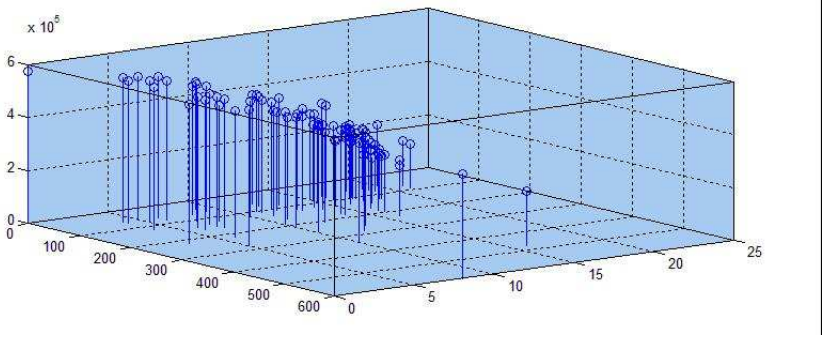

Figura 6.4: Frente de Pareto com 89 soluções não-dominadas - opm4

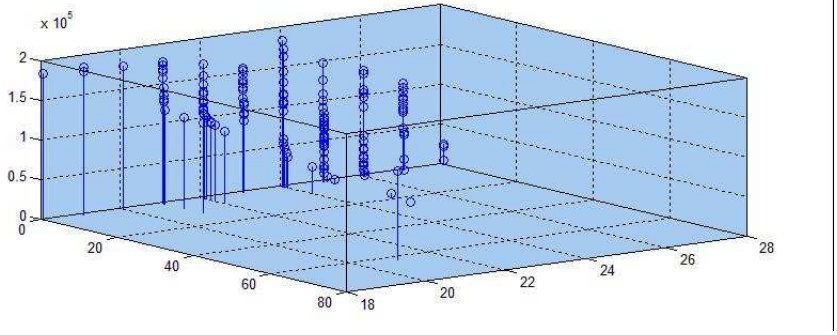

Figura 6.5: Frente de Pareto com 129 soluções não-dominadas - opm5

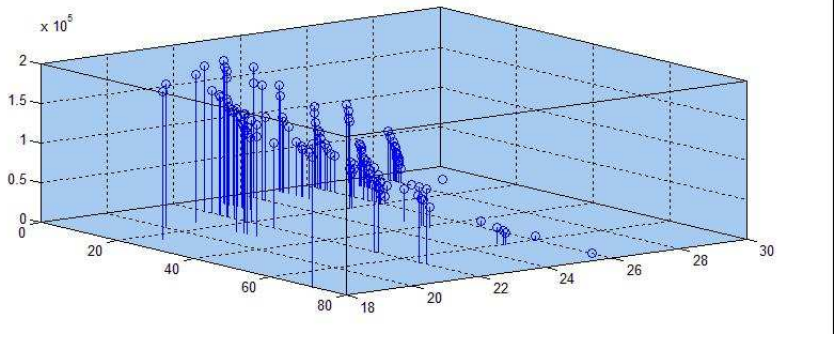

Figura 6.6: Frente de Pareto com 119 soluções não-dominadas - opm6

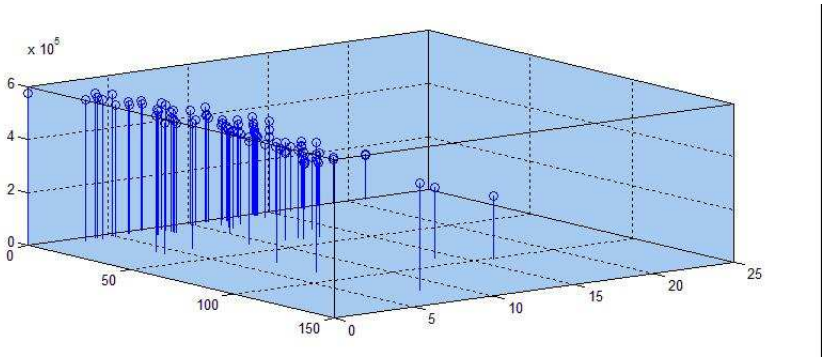

Figura 6.7: Frente de Pareto com 87 soluções não-dominadas - opm7

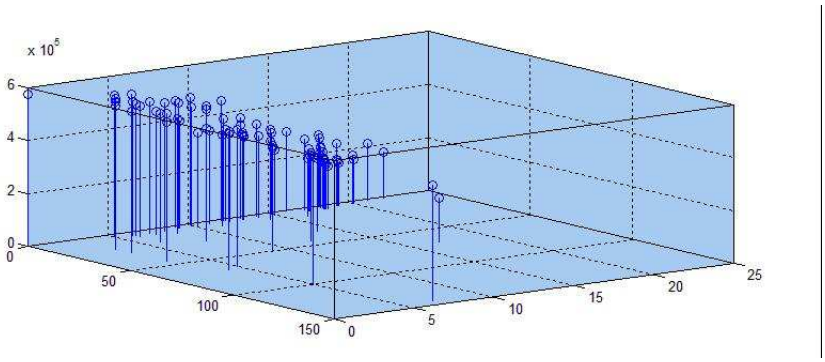

Figura 6.8: Frente de Pareto com 79 soluções não-dominadas - opm8

7 *Conclusões e Trabalhos Futuros*

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões, POLAD.

Neste trabalho o POLAD foi resolvido com abordagem multiobjetivo, levando-se em consideração vários objetivos conflitantes, como: obtenção das metas de produção e qualidade para o produto formado, e minimização do número de veículos necessários ao processo produtivo. Na abordagem multiobjetivo não há uma única solução que satisfaça a todos os objetivos. O que se procura é um conjunto de soluções não-dominadas, também chamadas de soluções eficientes, ou Fronteira de Pareto, cabendo ao tomador de decisões a escolha da solução mais adequada.

Em virtude da complexidade combinatória do problema, foram propostos dois algoritmos heurísticos multiobjetivo. O primeiro deles, denominado GRASP-NSGAII-PR, combina os procedimentos *Greedy Randomized Adaptive Search Procedures* (GRASP), *Nondominated Sorting Genetic Algorithm II* (NSGA-II) e o procedimento de Reconexão por Caminhões (PR, do inglês *Path Relinking*), como operador de cruzamento. O segundo algoritmo, denominado GRASP-MOVNS, combina os procedimentos GRASP e *Multiobjective Variable Neighborhood Search* (MOVNS).

Usando oito problemas-teste da literatura e três métricas comparação para algoritmo multiobjetivo, os algoritmos foram comparados entre si.

Primeiramente, foi feita uma bateria de testes com três variantes do algoritmo GRASP-NSGAII-PR. Esses testes tiveram como objetivo calibrar e validar o algoritmo em questão. Em seguida o algoritmo GRASP-MOVNS foi comparado ao algoritmo GRASP-NSGAII-PR, obtendo, em todos os problemas-testes utilizados, frentes de pareto mais diversificadas e com uma melhor convergência. Por fim, utilizando uma função mono-objetivo, as soluções obtidas pelo algoritmo GRASP-MOVNS foram comparadas a um algoritmo da literatura mono-objetivo, denominado GGVNS, de Souza et al. (2010). O algoritmo GRASP-MOVNS foi capaz de obter melhores soluções em quatro problemas-testes, mostrando assim o poderio deste algoritmo tanto para aplicações multiobjetivo quanto para aplicações mono-objetivo.

Dado que a tomada de decisão no problema em pauta tem que ser rápida, os resultados encontrados validam a utilização dos algoritmos propostos enquanto ferramenta de apoio à decisão.

Para trabalhos futuros, propõe-se a criação de mais um novo conjunto de problemas-teste, porém apenas com intuito acadêmico, visto que o conjunto atual é bastante restrito e não permite um número maior de comparações entre os métodos. Propõe-se também a execução de testes exaustivos, de forma a obter um conjunto de referência com maior convergência e diversidade. Para validação desses resultados, outras métricas de validação de desempenho de algoritmos multiobjetivo devem ser implementadas. É também proposto como trabalho futuro a implementação de novos métodos e estratégias de resolução para o MOPOLAD.

Referências Bibliográficas

ALARIE, S.; GAMACHE, M. Overview of solution strategies used in truck dispatching systems for open pit mines. *International Journal of Surface Mining, Reclamation and Environment*, v. 16, p. 59–76, 2002.

ALEXANDRE, R. F. *Modelagem, Simulação da Operação e Otimização Multiobjetivo Aplicada ao Problema de Despacho de Veículos em Minas a Céu Aberto*. Dissertação (Dissertação de Mestrado) — Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, Belo Horizonte, 2010.

ALVARENGA, G. B. *Despacho ótimo de caminhões numa mineração de ferro utilizando algoritmo genético com processamento paralelo*. Dissertação (Dissertação de Mestrado) — Programa de Pós-Graduação em Engenharia Elétrica/UFMG, Belo Horizonte, 1997.

ARAÚJO, F. C. R. *Planejamento Operacional de Lavra com Alocação Dinâmica de Caminhões: Abordagens Exata e Heurística*. Dissertação (Dissertação de Mestrado) — Programa de Pós-Graduação em Engenharia Mineral do Departamento de Engenharia de Minas da Escola de Minas da Universidade Federal de Ouro Preto, Ouro Preto, 2008.

BACK, T.; HAMMEL, U.; SCHWEFEL, H.-P. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, v. 1, p. 1097–1100, 1997.

BEUME, N. et al. On the complexity of computing the hypervolume indicator. *Evolutionary Computation, IEEE Transactions on*, v. 13, n. 5, p. 1075 –1082, 2009.

BEYER, H. G.; SCHWEFEL, H. P. Evolution strategies - a comprehensive introduction. *Natural Computing*, v. 1, p. 3–52, 2002.

CHANDA, E. K. C.; DAGDELEN, K. Optimal blending of mine production using goal programming and interactive graphics systems. *International Journal of Surface Mining, Reclamation and Environment*, v. 9, p. 203–208, 1995.

COELHO, I. M. et al. A computational framework for combinatorial optimization problems. In: *VII ALIO/EURO Workshop on Applied Combinatorial Optimization*. Porto: [s.n.], 2011. p. 51–54.

COELHO, I. M. et al. Optframe: a computational framework for combinatorial optimization problems. In: *XLII Simpósio Brasileiro de Pesquisa Operacional*. Bento Gonçalves, RS: [s.n.], 2010. p. 1–12.

COELHO, I. M.; RIBAS, S.; SOUZA, M. J. F. Um algoritmo baseado em grasp, vnd e iterated local search para a resolução do planejamento operacional de lavra. In: *XV Simpósio de Engenharia de Produção*. Bauru/SP: [s.n.], 2008.

COELHO, V. N. et al. Estratégias evolutivas aplicadas a um problema de programação inteira mista. In: A (Ed.). *Anais X Congresso Brasileiro de Inteligência Computaciona (CBIC)*. Fortaleza/CE: [s.n.], 2011. v. 1, p. 1–8.

COELHO, V. N. et al. Pggvns: Um algoritmo paralelo para o problema de planejamento operacional de lavra. In: *Anais XVIII Simpósio de Engenharia de Produção (SIMPEP)*. Bauru/SP: [s.n.], 2011. v. 1, p. 1–14.

COELLO, C. C. Evolutionary multi-objective optimization: a historical view of the field. *Computational Intelligence Magazine, IEEE*, v. 1, n. 1, p. 28–36, 2006. ISSN 1556-603X.

COSTA, F. P. *Aplicações de Técnicas de Otimização a Problemas de Planejamento Operacional de Lavra em Minas a Céu Aberto*. Dissertação (Dissertação de Mestrado) — Departamento de Engenharia de Minas/EM/UFOP, Ouro Preto, 2005.

COSTA, F. P.; SOUZA, M. J. F.; PINTO, L. R. Um modelo de alocação dinâmica de caminhões. *Revista Brasil Mineral*, v. 231, p. 26–31, 2004.

COSTA, F. P.; SOUZA, M. J. F.; PINTO, L. R. Um modelo de programação matemática para alocação estática de caminhões visando ao atendimento de metas de produção e qualidade. *Revista da Escola de Minas*, v. 58, p. 77–81, 2005.

DEB, K. *Multiobjective Optimization Using Evolutionary Algorithms*. Chichester, U.K.: John Wiley & Sons, 2001.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 2, p. 182–197, 2002. ISSN 1089-778X.

DIAS, A.; VASCONCELOS, J. A. Multiobjective genetic algorithms applied to solvoptimization problems. *IEEE Transactions on Magnetcs*, v. 38, p. 1133–1136, 2002.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.

FONSECA, C.; PAQUETE, L.; LOPEZ-IBANEZ, M. An improved dimension-sweep algorithm for the hypervolume indicator. In: *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. [S.l.: s.n.], 2006. p. 1157 – 1163.

FONSECA, C. M.; FLEMING, P. J. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: *Proceedings of the Fifth International Conference*. San Mateo, California: Morgan Kauffman Publishers, 1993.

GEIGER, M. J. Randomised variable neighbourhood search for multi objective optimisation. *In Proceedings of the 4th EUME Workshop Design and Evaluation of Advanced Hybrid MetaHeuristics*, November 4, n. Nottingham, United Kingdom,, p. 34–42, 2004. Disponível em: <http://arxiv.org/abs/0809.0271>.

GERSHON, M. A linear programming approach to mine scheduling optimization. In: *Proceedings of the 17th Application of computers and operations research in the mineral industry*. New York: [s.n.], 1982. p. 483–493.

GLOVER, F. Computing tools for modeling, optimization and simulation: Interfaces in computer science and operations research. In: ____. [S.l.]: Kluwer Academic Publishers, 1996. cap. Tabu search and adaptive memory programming - Advances, applications and challenges, p. 1–75.

GLOVER, F.; LAGUNA, M. *Tabu Search*. Boston: Kluwer Academic Publishers, 1997.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1. ed. Berkeley: Addison-Wesley Professional, 1989. Hardcover.

GUIMARÃES, I. F.; PANTUZA, G.; SOUZA, M. J. F. Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. In: *Anais do XIV Simpósio de Engenharia de Produção (SIMPEP)*. Bauru, CD-ROM: [s.n.], 2007. p. 11.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, p. 449–467, 2001.

HANSEN, P.; MLADENOVIC, N.; PÉREZ, J. A. M. Variable neighborhood search: methods and applications. *4OR: Quarterly journal of the Belgian, French and Italian operations research societies*, v. 6, p. 319–360, 2008.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. A niched pareto genetic algorithm for multiobjective optimization. In: *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. [S.l.: s.n.], 1994. p. 82–87.

KNOWLES, J.; CORNE, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. [S.l.: s.n.], 1999. v. 1, p. 98–105.

LIANG, Y.; CHEN, A.; TIEN, C. Variable neighborhood search for multi-objective parallel machine scheduling problems. In: *Proceedings of the Eighth International Conference on Information and Management Sciences*. [S.l.: s.n.], 2009.

LIANG, Y.; LO, M. Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm. *Journal of Heuristics*, Springer, v. 16, n. 3, p. 511–535, 2010.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers, 2003.

MARAN, J.; TOPUZ, E. Simulation of truck haulage systems in surface mines. *International Journal of Surface Mining*, v. 2, p. 43–49, 1988.

MERSCHMANN, L. H. C. *Desenvolvimento de um sistema de otimização e simulação para análise de cenários de produção em minas a céu aberto*. Dissertação (Dissertação de Mestrado) — Programa de Engenharia de Produção/COPPE/UFRJ, Rio de Janeiro, 2002.

MICHALEWICZ, Z. *Genetic Algorithms+Data Structures=Evolution Programs*. [S.l.]: Springer- Verlag, 1984.

MLADENOVIĆ, N.; HANSEN, P. A variable neighborhood search. *Computers and Operations Research*, v. 24, p. 1097–1100, 1997.

PANTUZA, G. *Métodos de Otimização Multiobjetivo e de Simulação aplicados ao Problema de Planejamento Operacional de Lavra em Minas a Céu Aberto*. Dissertação (Dissertação de Mestrado) — Programa de Pós-Graduação em Engenharia Mineral - PPGEM da Universidade Federal de Ouro Preto, Ouro Preto, 2011.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers, 2003. p. 219–242.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In: GENDREAU, M.; POTVIN, J. (Ed.). *Handbook of Metaheuristics*. 2. ed. New York: Springer, 2010. p. 283–319.

RIBEIRO, C.; RESENDE, M. Path-relinking intensification methods for stochastic local search algorithms. *Journal of Heuristics*, Springer Netherlands, p. 1–22, 2012. ISSN 1381-1231. 10.1007/s10732-011-9167-1. Disponível em: <http://dx.doi.org/10.1007/s10732-011-9167-1>.

RIBEIRO, C. C.; UCHOA, E.; WERNECK, R. F. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing*, Linthicum, v. 14, p. 228–246, 2002.

RODRIGUES, L. F. *Análise comparativa de metodologias utilizadas no despacho de caminhões em minas a céu aberto*. Dissertação (Mestrado) — Programa de Pós-Graduação em Engenharia de Produção, Escola de Engenharia, UFMG, Belo Horizonte, 2006.

ROSSETI, I. C. M. *Estratégias sequenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos*. Dissertação (Tese de Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2003.

SCHAFFER, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In: *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985. p. 93–100. Disponível em: <http://dl.acm.org/citation.cfm?id=645511.657079>.

SCHOTT, J. R. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Dissertação (Dissertação de Mestrado) — Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.

SOUZA, M. J. F. et al. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research, EJOR*, v. 207, p. 1041–1051, 2010.

SRINIVAS, N.; DEB, K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, v. 2, p. 221–248, 1994.

VASCONCELOS, J. A. *Optimization de Forme des Structures Életromagnétiques*. Dissertação (Tese de Doutorado) — Escole Centrale de Lyon, França, 1994.

WHITE, J. W.; ARNOLD, M. J.; CLEVENGER, J. G. Automated open-pit truck dispatching at Tyrone. *Engineering and Mining Journal*, v. 183, n. 6, p. 76–84, 1982.

WHITE, J. W.; OLSON, J. P. Computer-based dispatching in mines with concurrent operating objetives. *Mining Engineering*, v. 38, n. 11, p. 1045–1054, 1986.

ZITZLER, E.; DEB, K.; THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 8, p. 173–195, June 2000. ISSN 1063-6560.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Zurich, Switzerland, 2001.

ZITZLER, E.; THIELE, L. Multiobjective optimization using evolutionary algorithms - a comparative case study. In: EIBEN, A. et al. (Ed.). *Parallel Problem Solving from Nature - PPSN V*. Springer Berlin / Heidelberg, 1998, (Lecture Notes in Computer Science, v. 1498). p. 292–301. ISBN 978-3-540-65078-2. 10.1007/BFb0056872. Disponível em: <http://dx.doi.org/10.1007/BFb0056872>.

Anexos

Como produtos do desenvolvimento deste trabalho foram geradas as seguintes produções no ano de 2011:

- COELHO, V. N. ; SOUZA, M.J.F. ; COELHO, I. M. ; RIBAS, S. ; OLIVEIRA, T. A. . PGGVNS: UM ALGORITMO PARALELO PARA O PROBLEMA DE PLANEJAMENTO OPERACIONAL DE LAVRA. In: XVIII SIMPEP, 2011, Bauru/SP. XVIII Simpósio de Engenharia de Produção, 2011. v. 1. 10p.

- COELHO, V. N. ; SOUZA, M.J.F. ; COELHO, I. M. ; GUIMARAES, F. G. ; COELHO, B. N. . Um Algoritmo Baseado em Estratégias Evolutivas para o Problema de Planejamento Operacional de Lavra. In: 14^{o} Simpósio de Pesquisa Operacional & Logística da Marinha, 2011, Rio de Janeiro, RJ. Anais do 14^{0} SPOLM, 2011. v. 1. 12p.

- COELHO, V. N. ; SOUZA, M.J.F. ; COELHO, I. M. ; GUIMARAES, F. G. ; COELHO, B. N. . Estratégias Evolutivas aplicadas a um problema de Programação Inteira Mista. In: X Congresso Brasileiro de Inteligência Computacional, 2011, Fortaleza. Anais do X CBIC, 2011. v. 1. 8p.

- COELHO, V. N. ; SOUZA, M.J.F. ; COELHO, I. M. ; GUIMARAES, F. G. ; COELHO, B. N. . Um Algoritmo Baseado em Estratégias Evolutivas para o Problema de Planejamento Operacional de Lavra. In: XXXI ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCAO, 2011, Belo Horizonte. XXXI ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCAO, 2011. v. 1. 13p.

XVIII SIMPEP SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO
07 a 09 de novembro 2011
Sustentabilidade na Cadeia de Suprimentos

PGGVNS: UM ALGORITMO PARALELO PARA O PROBLEMA DE PLANEJAMENTO OPERACIONAL DE LAVRA

VITOR NAZÁRIO COELHO
vncoelho@gmail.com
UNIVERSIDADE FEDERAL DE OURO PRETO - UFOP

MARCONE JAMILSON FREITAS SOUZA
marcone@iceb.ufop.br
UNIVERSIDADE FEDERAL DE OURO PRETO - UFOP

IGOR MACHADO COELHO
igor.machado@gmail.com
UNIVERSIDADE FEDERAL FLUMINENSE - UFF

SABIR RIBAS
sabirribas@gmail.com
UNIVERSIDADE FEDERAL FLUMINENSE - UFF

THAYS APARECIDA DE OLIVEIRA
thaysoliveira7@gmail.com
UNIVERSIDADE FEDERAL DE OURO PRETO - UFOP

Resumo: ESTE TRABALHO APRESENTA UM APERFEIÇOAMENTO DE UM ALGORITMO HEURÍSTICO SEQUENCIAL BASEADO NAS METAHEURÍSTICAS GRASP E GENERAL VARIABLE NEIGHBORHOOD SEARCH (GVNS). A MELHORIA CONSISTE NA PARALELIZAÇÃO DESTE ALGORITMO, VISTO QUE ESTE É APLICADDO A UM PROBLEMA QUE REQUER DECISÕES RÁPIDAS, O PROBLEMA DE PLANEJAMENTO OPERACIONAL DE LAVRA EM MINAS A CÉU ABERTO COM ALOCAÇÃO DINÂMICA DE CAMINHÕES (POLAD). OS RESULTADOS PRODUZIDOS PELO ALGORITMO PGGVNS FORAM COMPARADOS COM AQUELES PRODUZIDOS PELA SUA VERSÃO SEQUENCIAL, DENOMINADA GGVNS, E POR UMA OUTRA VERSÃO, DENOMINADA GGVNSMIP-PR, QUE COMBINA OS MÉTODOS DO ALGORITMO GGVNS COM UM MÓDULO DE INTENSIFICAÇÃO INTEGRADO AO SOLVER GLPK, ALÉM DE POSSUIR UMA FASE DE PÓS-OTIMIZAÇÃO, BASEADA EM RECONEXÃO POR CAMINHOS. COMPARANDO-SE A VERSÃO PARALELA E AS DUAS VERSÕES SEQÜENCIAIS PROPOSTAS EM TRABALHOS ANTERIORES, OBSERVOU-SE A SUPREMACIA DA VERSÃO PARALELA, TANTO EM TERMOS DE QUALIDADE DA SOLUÇÃO FINAL QUANTO NA VARIABILIDADE.

Palavras-chaves: PLANEJAMENTO OPERACIONAL DE LAVRA; PROGRAMAÇÃO INTEIRA MISTA; PROGRAMAÇÃO PARALELA; GRASP; GENERAL VARIABLE NEIGHBORHOOD SEARCH

PGGVNS: A PARALLEL ALGORITHM FOR THE OPEN-PIT-MINING OPERATIONAL PLANNING PROBLEM

Abstract: THIS PAPER IMPROVES A SEQUENCIAL HEURISTIC ALGORITHM BASED ON THE GRASP AND GENERAL VARIABLE NEIGHBORHOOD SEARCH (GVNS) METAHEURISTICS. THE IMPROVEMENT CONSISTS IN PARALLELIZING THIS ALGORITHM, SINCE IT IS APPLIED TO A PROBLEM THAT REQUIRESS QUICK DECISIONS, THE OPEN-PIT-MINING OPERATIONAL PLANNING PROBLEM WITH DYNAMIC TRUCK ALLOCATION (OPMOP). THE RESULTS PRODUCED BY THE ALGORITHM PGGVNS WERE COMPARED WITH THOSE PRODUCED BY ITS SEQUENTIAL VERSION, CALLED GGVNS, AND ANOTHER VERSION, SO-CALLED GGVNSMIP-PR, WHICH COMBINES THE METHODS OF THE ALGORITHM GGVNS WITH AN INTENSIFICATION MODULE INTEGRATED WITH GLPK SOLVER AND A POST-OPTIMIZATION MODULE, BASED ON PATH RELINKING APPROACH. COMPARING THE PARALLEL VERSION AND THE TWO SEQUENTIAL VERSIONS, PROPOSED IN PREVIOUS STUDIES, WE OBSERVED THE SUPREMACY OF THE PARALLEL VERSION, BOTH IN TERMS OF QUALITY OF THE FINAL SOLUTION AND IN VARIABILITY.

Keyword: OPEN-PIT MINING; MIXED INTEGER PROGRAMMING; PARALLEL PROGRAMMING; GRASP; GENERAL VARIABLE NEIGHBORHOOD SEARCH

1. Introdução

Este trabalho trata do problema de planejamento operacional de lavra com alocação dinâmica de caminhões (POLAD). Neste problema, deve-se determinar a taxa de extração de minério e estéril nas frentes de lavra e associar a elas carregadeiras e caminhões de forma que as metas de produção e qualidade sejam satisfeitas, além disso, procura-se minimizar o número de caminhões necessários para a execução desta tarefa. Considera-se a alocação dinâmica de caminhões, isto é, a cada viagem realizada, o caminhão pode se direcionar a uma frente diferente. Essa forma de alocação contribui para o aumento da produtividade da frota e, consequentemente, para a redução do número de caminhões necessários ao processo produtivo.

O POLAD é um problema da classe NP-difícil, uma vez que tem como subproblema, o Problema da Mochila Múltipla (PMM). Essa analogia pode ser feita considerando cada carregadeira como uma mochila e as cargas (minério ou estéril) carregadas pelos caminhões como sendo os objetos. Nessa analogia, o objetivo é determinar quais são as cargas que possuem melhor benefício para serem colocadas na mochila, respeitando-se a produtividade de cada carregadeira. Como o PMM pertence à classe NP-difícil (Papadimitriou e Steiglitz, 1998), o POLAD também o é.

A literatura tem mostrado várias abordagens de resolução por meio de procedimentos heurísticos, visto que métodos exatos possuem uma aplicabilidade restrita. Costa (2005) desenvolveu um algoritmo heurístico baseado em Greedy Randomized Adaptive Search Procedures - GRASP (Feo e Resende, 1995) e VNS (Hansen e Mladenovic, 2001) para o POLAD usando seis tipos diferentes de movimentos para explorar o espaço de soluções. Foi feita uma comparação entre os resultados obtidos por esse algoritmo heurístico e os encontrados pelo otimizador LINGO, versão 7, aplicado a um modelo de programação matemática desenvolvida. Mostrou-se que o algoritmo heurístico foi capaz de encontrar soluções de melhor qualidade mais rapidamente.

Guimarães et al. (2007) apresentaram um modelo de simulação computacional para validar resultados obtidos pela aplicação de um modelo de programação matemática na determinação do ritmo de lavra em minas a céu aberto. Dessa maneira, foi possível validar os resultados da otimização, já que na modelagem de otimização não é possível tratar a variabilidade nos tempos de ciclo e a ocorrência de fila.

Em Coelho et al. (2008), o POLAD é resolvido por um algoritmo heurístico, denominado GVILS, que combina os procedimentos heurísticos GRASP, VND e ILS (Lourenço et al., 2003). O algoritmo GVILS faz uso de oito movimentos para explorar o espaço de soluções. Além dos desvios de produção e qualidade, procurou-se minimizar, também, o número de veículos. Usando quatro problemas-teste da literatura, o GVILS foi comparado com o otimizador CPLEX 9.1 aplicado a um modelo de programação matemática. Foram realizados testes envolvendo 15 minutos de processamento. Em dois dos problemas, o algoritmo proposto mostrou-se bastante superior; enquanto nos dois outros ele foi competitivo com o CPLEX, produzindo soluções médias com valores até 0,08% piores, na média.

Souza et al. (2010) desenvolveram uma versão aprimorada do algoritmo desenvolvido em Coelho et al. (2008), denominado GGVNS. Esse algoritmo combina os procedimentos GRASP (Feo e Resende, 1995; Resende e Ribeiro, 2008) e General Variable Neighborhood Search - GVNS (Hansen et al., 2008a; Hansen et al., 2008b; Hansen e Mladenovic, 2001; Mladenovic e Hansen, 1997) para resolução do POLAD. Adicionalmente foi validado um novo modelo de programação matemática utilizando o software comercial CPLEX 11. Resultados computacionais mostraram que o algoritmo proposto foi capaz de encontrar

soluções competitivas com o otimizador CPLEX em oito problemas-teste, encontrando soluções perto da otimalidade (a menos de 1%) na maioria das instâncias, e demandando um pequeno tempo computacional.

O presente trabalho contribui com a apresentação de uma versão paralela do algoritmo heurístico GGVNS, de Souza et al. (2010). O algoritmo proposto combina a flexibilidade das metaheurísticas com o poder das máquinas e clusters multi-core. Do procedimento GRASP utilizou-se a fase de construção para produzir soluções viáveis e de boa qualidade rapidamente. O GVNS foi escolhido devido a sua simplicidade, eficiência e capacidade natural de sua busca local (método VND) para lidar com diferentes vizinhanças. Além disso, este algoritmo foi implementado com base na última versão do framework OptFrame, http://sourceforge.net/projects/optframe/, estando, assim, em um patamar eficiente de otimização.

O restante deste trabalho está organizado como segue. A Seção 2 detalha o algoritmo proposto para resolver o POLAD. A Seção 3 mostra os resultados dos experimentos computacionais e a Seção 4 conclui o trabalho.

2. Metodologia

2.1 Representação de uma solução

Dado um conjunto de frentes de lavra F, um conjunto de caminhões V e um conjunto de carregadeiras K, uma solução para o POLAD é representada por uma matriz R = [Y | N], sendo Y a matriz $|F| \times 1$ e N uma matriz $|F| \times |V|$. Cada célula y_i da matriz $Y_{|F| \times 1}$ representa a carregadeira $k \in K$ alocada à frente $i \in F$. Um valor -1 significa que não existe carregadeira alocada à frente i. Se não houver viagens feitas a uma frente i, a carregadeira k associada a tal frente é considerada inativa e não é penalizada por produção abaixo da mínima para este equipamento de carga.

Na matriz $N_{|F| \times |V|}$, cada célula n_{il} representa o número de viagens do caminhão $l \in V$ a cada frente $i \in F$. Um valor 0 (zero) significa que não há viagem para aquele caminhão. O valor -1 indica incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

Na Tabela 1, tem-se um exemplo de uma possível solução para o POLAD, observa-se que na coluna CARGA, linha F_1, a dupla $\langle Car_1,1\rangle$, indicando que o equipamento de carga Car_1 está alocado à frente F_1 e em operação. Na coluna CARGA, linha F_3, a dupla $\langle Car_8,0\rangle$ indica que o equipamento de carga Car_8 está alocado à frente F_3, mas não está em operação. Observa-se, ainda, na coluna CARGA, linha F_2, o valor $\langle D,0\rangle$ informando que não existe equipamento de carga alocado à frente F_2 e que, portanto, esta frente está disponível. As demais colunas representam o número de viagens a serem realizadas por um caminhão a uma frente, considerando a compatibilidade entre o caminhão e o equipamento de carga alocado à frente. As células com os valores -1 indicam incompatibilidade entre um caminhão e o respectivo equipamento de carga.

Tabela 1 – Representação de uma solução

	Carga	**Cam_1**	**Cam_2**	**...**	**Cam_V**
F_1	<Car_1,1>	8	-1	...	-1
F_2	<D, 0>	0	0	...	0
F_3	<Car_8, 0>	0	0	...	0
...	...	...	...	...	...

F_F	<Car_5,1>	0	9	...	3

2.2 Estruturas de Vizinhança

Como forma de explorar o espaço de soluções, foram utilizados os oito movimentos a seguir, os quais possuem uma boa capacidade exploratória, como relatado em Souza et al. (2010).

Movimento Número de Viagens - $N^{NV}(s)$: Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

Movimento Carga - $N^{CG}(s)$: Consiste em trocar duas células distintas y_i e y_k da matriz Y, ou seja, trocar os equipamentos de carga que operam nas frentes i e k, caso as duas frentes possuam equipamentos de carga alocados. Havendo apenas uma frente com equipamento de carga, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas junto com as frentes escolhidas.

Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$: Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl}. Assim, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$: Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik}. Isto é, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Operação Frente - $N^{OF}(s)$: Consiste em retirar de operação o equipamento de carga que esteja em operação na frente i. O movimento retira todas as viagens feitas a esta frente, deixando o equipamento inativo. O equipamento retorna à operação assim que uma nova viagem é associada a ele.

Movimento Operação Caminhão - $N^{OC}(s)$: Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, isto é, retirar de atividade um caminhão l que esteja operando em uma frente i.

Movimento Troca de Viagens - $N^{VT}(s)$: Duas células da matriz N são selecionadas e uma unidade de uma célula passa para a outra, isto é, uma viagem de um caminhão associado a uma frente i passa para outro caminhão associado outra frente.

Movimento Troca de Carregadeiras - $N^{CT}(s)$: Duas células distintas y_i e y_k da matriz Y tem seus valores permutados, ou seja, os equipamentos de carga que operam nas frentes i e k são trocados. Analogamente ao movimento $N^{CG}(s)$, os equipamentos de carga são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

A Figura 1 mostra uma possível solução para o problema.

$$s = \left\langle \begin{array}{c|cccc} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right\rangle$$

Figura 1 – Exemplo de uma solução para o POLAD

A Figura 2 ilustra como ficaria a solução da Figura 1 após uma aplicação aleatória de cada um dos movimentos descritos.

$$s \oplus m^{NV} = \left[\begin{array}{c|cccc} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & *4 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right] \qquad s \oplus m^{CG} = \left[\begin{array}{c|cccc} *3 & *1 & *0 & *3 & *2 \\ -1 & 0 & 0 & 0 & 0 \\ *1 & *2 & *4 & *3 & *0 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right]$$

$$s \oplus m^{VC} = \left[\begin{array}{c|cccc} 1 & 2 & *3 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & *1 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right] \qquad s \oplus m^{VF} = \left[\begin{array}{c|cccc} 1 & *1 & *5 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right]$$

$$s \oplus m^{OF} = \left[\begin{array}{c|cccc} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & *0 & *0 & *0 & *0 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right] \qquad s \oplus m^{OC} = \left[\begin{array}{c|cccc} 1 & 2 & 4 & *0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right]$$

$$s \oplus m^{VT} = \left[\begin{array}{c|cccc} 1 & *1 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & *4 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right] \qquad s \oplus m^{CT} = \left[\begin{array}{c|cccc} *3 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ *1 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array} \right]$$

Figura 2 – Exemplo de aplicação dos movimentos

2.3 Avaliação de uma solução

Como os movimentos usados podem gerar soluções inviáveis, uma solução é avaliada por uma função f, a ser minimizada, composta por duas parcelas. A primeira delas é a função objetivo propriamente dita, f^{PM}, descrita em Souza et al. (2010), e a segunda é composta pelas funções que penalizam a ocorrência de inviabilidade na solução corrente. A função f, definida pela Eq. (1), mensura o desvio dos objetivos considerados e penaliza o não atendimento às restrições do problema.

$$f(s) = f^{PM}(s) + f^{p}(s) + \sum_{j \in T} f_j^{q}(s) + \sum_{l \in V} f_l^{u}(s) + \sum_{k \in C} f_k^{c}(s) \qquad (1)$$

Na Eq. (1), tem-se:

– $f^{PM}(s)$: função que avalia s quanto ao atendimento às metas de produção e qualidade, bem como o número de caminhões utilizados (mesma do modelo de programação

matemática);

- $f^{p}(s)$: função que avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril;
- $f_j^{q}(s)$: função que avalia s quanto à inviabilidade em relação ao j-ésimo parâmetro de controle;
- $f_l^{u}(s)$: função que avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l-ésimo caminhão;
- $f_k^{c}(s)$: função que avalia s quanto ao não atendimento aos limites de produtividade da carregadeira k.

2.4 Algoritmo Proposto

O algoritmo desenvolvido, denominado PGGVNS, é uma versão paralela do algoritmo heurístico GGVNS, proposto em Souza et al. (2010). Este algoritmo consiste na combinação dos procedimentos heurísticos GRASP e GVNS.

O algoritmo PGGVNS representa várias melhorias em relação à sua versão sequencial, tanto em termos de otimização de código, pois ele foi implementado utilizando a última versão do framework OptFrame, quanto em termos de implementação de novos métodos e estratégias.

Para cada núcleo de processamento disponível, o algoritmo PGGVNS aciona o procedimento GGVNS, retornando a melhor solução encontrada por todos os processos. Na estratégia de paralelização utilizada, os processos podem estar ligados a uma rede de clusters multi-core, ou seja, pode-se utilizar o paralelismo fill up, no qual todos os núcleos das máquinas da rede de clusters trabalham.

O pseudocódigo do algoritmo sequencial GGVNS está esquematizado na Figura 3. Nesta Figura, GRASPmax representa a quantidade de iterações em que a fase de construção GRASP é aplicada e IterMax indica o número máximo de iterações executadas em um dado nível de perturbação.

Entrada: Solução s, γ, *GRASPmax*, *IterMax*, Função $f(.)$
Saída: Solução s^* de qualidade possivelmente superior à s de acordo com a função f

1 $s_w \leftarrow$ ConstróiSoluçãoEstéril()
2 $s_0 \leftarrow$ Melhor Solução em $GRASPmax$ iterações do procedimento ConstróiSoluçãoMinério(s_w, γ)
3 $s^* \leftarrow$ VND(s_0, f)
4 $p \leftarrow 0$
5 **enquanto** *critério de parada não satisfeito* **faça**
6 $\quad iter \leftarrow 0$
7 $\quad$ **enquanto** $iter < IterMax$ *e critério de parada não satisfeito* **faça**
8 $\quad\quad s' \leftarrow$ Refinamento(s^*, p, f)
9 $\quad\quad$ **se** s' *for melhor que* s^* *de acordo com a função* f **então**
10 $\quad\quad\quad s^* \leftarrow s'$;
11 $\quad\quad\quad p \leftarrow 0$;
12 $\quad\quad\quad iter \leftarrow 0$
13 $\quad\quad$ **fim**
14 $\quad\quad$ **senão**
15 $\quad\quad\quad iter \leftarrow iter + 1$
16 $\quad\quad$ **fim**
17 $\quad$ **fim**
18 $\quad p \leftarrow p + 1$
19 **fim**
20 **retorna** s

Figura 3 – Algoritmo GGVNS

A solução inicial (linha 2 da Figura 3) é gerada aplicando-se a fase de construção GRASP. Os procedimentos ConstróiSoluçãoEstéril e ConstróiSoluçãoMinério (linhas 1 e 2 da Figura 3) são os mesmos de Souza et al. (2010). O valor do parâmetro γ define o tamanho da lista restrita de candidatos, conforme a ideia básica do procedimento GRASP.

A busca local (linha 3 da Figura 3) é feita pelo procedimento VND, descrito na Figura 4, usando-se um grupo restrito de quatro movimentos apresentados na seção 2.2, no caso, relativos às vizinhanças N^{CG}, N^{NV}, N^{VC} e N^{VF}.

Entrada: r vizinhanças na ordem aleatória: N^{VF}, N^{VC}, N^{NV} e N^{CG}
Entrada: Solução Inicial s e Funcão de Avaliação f
Saída: Solução s

1 $k \leftarrow 1$
2 **enquanto** $k \leq r$ **faça**
3 $\quad$ Encontre o melhor vizinho $s' \in N^{(k)}(s)$
4 $\quad$ **se** $f(s') < f(s)$ **então**
5 $\quad\quad s \leftarrow s'; k \leftarrow 1$
6 $\quad$ **fim**
7 $\quad$ **senão**
8 $\quad\quad k \leftarrow k + 1$
9 $\quad$ **fim**
10 **fim**
11 **retorna** s

Figura 4 – Algoritmo VND

O refinamento das soluções ótimas locais é feito pelo procedimento descrito na Figura 5. Nesse refinamento, é acionado o procedimento SelecionaVizinhança (linha 2 da Figura 5),

o qual faz uso de todas as oito vizinhanças descritas na seção 2.2. Em seguida, o ótimo local é perturbado pela aplicação de um movimento aleatório relativo à vizinhança k selecionada. Essa seleção de um movimento e uma perturbação é aplicada p + 2 vezes, gerando-se uma solução perturbada s'.

Entrada: r vizinhanças
Entrada: Solução Inicial s, Nível p e Funcão de Avaliação f
Saída: Solução s

1 **para** $i \leftarrow 1$ **até** $p + 2$ **faça**
2 $\quad k \leftarrow$ SelecionaVizinhança(r)
3 $\quad s' \leftarrow$ Perturbação(s, k)
4 **fim**
5 $s \leftarrow \text{VND}(s', f)$
6 **retorna** s

Figura 5 – Algoritmo de refinamento

3. Resultados Computacionais

O algoritmo proposto PGGVNS foi implementado em C++ usando o framework de otimização OptFrame e compilado pelo g++ 4.0. Os testes foram executados em um microcomputador Pentium Core 2 Quad(Q6600), 2.4 GHZ e 8 GB de RAM. Desta forma, o algoritmo PGGVNS contou com o auxílio de quatro núcleos para sua execução.

Para testá-lo, foi usado um conjunto de 8 problemas-teste da literatura, disponível em http://www.iceb.ufop.br/decom/prof/marcone/projects/mining.html.

Os melhores resultados da literatura para os problemas-teste analisados são apresentados na Tabela 2. Na coluna "Opt." indicamos por "√" as instâncias nas quais o otimizador matemático CPLEX 11.02 obteve o valor ótimo da função objetivo do problema.

Tabela 2 – Melhores valores de função objetivo para cada problema-teste

Problema-Teste	Melhor da Literatura	Opt.
opm1	227,12	
opm2	256,37	
opm3	164.027,15	√
opm4	164.056,68	√
opm5	227,04	
opm6	236,58	
opm7	164.017,46	√
opm8	164.018,65	√

Todos os experimentos consideraram os seguintes parâmetros: GRASPmax = 5000, IterMax = 50 e γ = 0,3. A função objetivo utilizada foi a mesma de Souza et al. (2010).

Primeiramente, realizou-se um experimento de probabilidade empírica, Time-to-target (TTT) plots, na forma indicada por Aiex et al. (2007), de forma a verificar a eficiência do algoritmo PGGVNS em relação ao algoritmo GGVNS, de Souza et al. (2010). Este teste mostra, no eixo das ordenadas, a probabilidade de um algoritmo em encontrar uma solução boa em um dado limite de tempo, eixo das abscissas. TTT plots já foi utilizado por vários autores, como Feo et al. (1994), e continua sendo defendido (Ribeiro e Resende, 2011) como

uma forma de caracterizar tempo de execução de algoritmos estocásticos aplicados a problemas de otimização combinatória.

Foram realizadas 120 execuções com cada algoritmo. Para as curvas da Figura 6, o problema-teste utilizado foi o opm1, tendo como alvo o valor 228,00 (menos de 1% do valor ótimo) e tempo limite de 120 segundos.

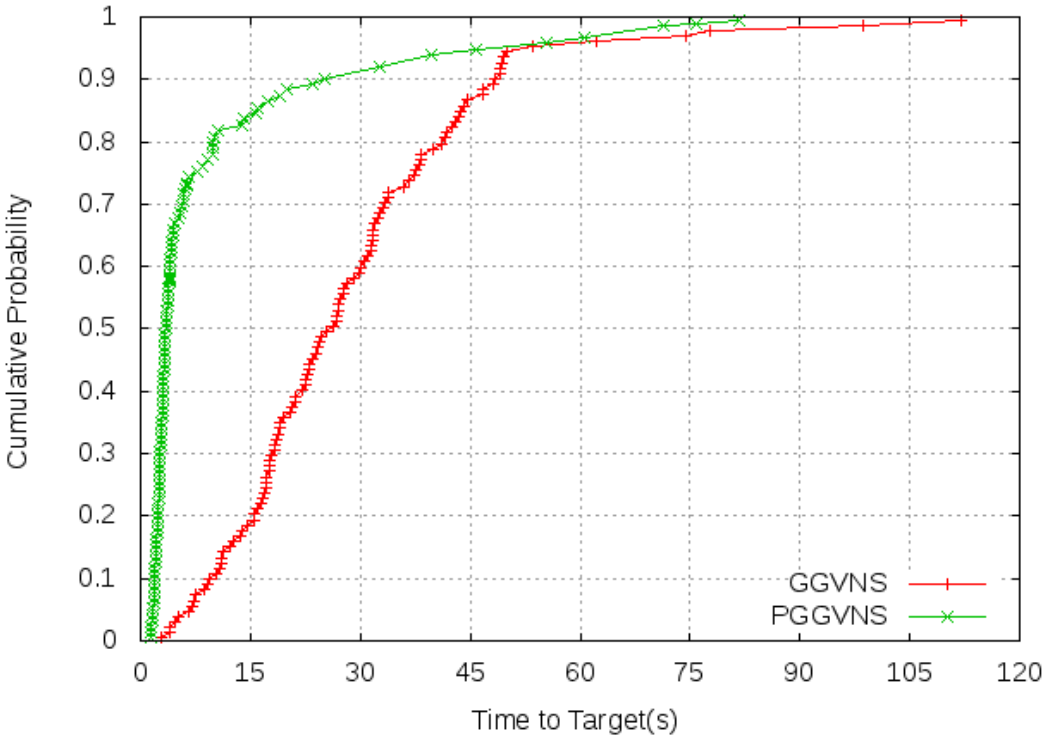

Figura 6 – Curva de probabilidade empírica GGVNS x PGGVNS – Instância opm1

Analisando-se as duas curvas de probabilidade empírica da Figura 6, verifica-se que a versão paralela PGGVNS foi capaz de gerar melhores soluções que a versão sequencial GGVNS desde os instantes iniciais da busca. Além disso, a curva mostra que o algoritmo PGGVNS teve uma probabilidade de quase 100% para encontrar o valor alvo desejado.

Tendo em vista a robustez do algoritmo PGGVNS, o mesmo foi comparado com dois algoritmos sequenciais da literatura. O primeiro deles, o algoritmo GGVNS, é o descrito em Souza et al. (2010). O segundo algoritmo, denominado GGVNSMIP-PR, foi proposto por Coelho et al. (2010) e combina os métodos do algoritmo GGVNS com um módulo de intensificação integrado ao solver GLPK, além de possuir uma fase de pós-otimização, baseada em Reconexão por Caminhos.

As Tabelas 3 e 4 comparam o algoritmo paralelo PGGVNS com as duas versões sequenciais, GGVNS e GGVNSMIP-PR.

A coluna “Instância” indica o problema-teste utilizado. Na Tabela 3, a coluna “IMPdesv” menciona o ganho relativo do algoritmo PGGVNS em relação aos algoritmos sequenciais GGVNS e GGVNSMIP-PR. Já a coluna “IMPbest” da Tabela 4 indica o percentual de melhora proporcionado pela versão paralela proposta neste trabalho em relação ao valor da melhor solução encontrada pelos outros dois algoritmos. As equações 2 e 3 mostram como esses valores são calculados.

$$IMPdesv_i = \frac{\bar{f}_i^{\,Alg\ Sequencial} - \bar{f}_i^{\,PGGVNS}}{\bar{f}_i^{\,Alg\ Sequencial}} \quad (2)$$

$$IMPbest_i = \frac{f_i^{Alg\ Sequencial^*} - f_i^{PGGVNS^*}}{f_i^{Alg\ Sequencial^*}} \quad (3)$$

Tabela 3 – Comparação de resultados IMPdesv: GGVNS x GGVNSMIP-PR x PGGVNS

Instância	Tempo (s)	GGVNS	GGVNSMIP-PR	PGGVNS	IMPdesv	
		Média	Média	Média	GGVNS	GGVNSMIP-PR
opm1	120	230,12	228,72	228,21	**0,83**	**0,22**
opm2	120	256,56	256,46	256,37	**0,07**	**0,04**
opm3	120	164064,68	164050,95	164035,29	**0,02**	**0,01**
opm4	120	164153,92	164099,3	164082,75	**0,04**	**0,01**
opm5	120	228,09	227,4	227,04	**0,46**	**0,16**
opm6	120	237,97	237,5	236,58	**0,58**	**0,39**
opm7	120	164021,89	164020,67	164018,94	0	0
opm8	120	164027,29	164022,38	164021,13	0	0

Tabela 4 – Comparação de resultados IMPbest: GGVNS x GGVNSMIP-PR x PGGVNS

Instância	Tempo (s)	GGVNS	GGVNSMIP-PR	PGGVNS	IMPbest	
		MELHOR	MELHOR	MELHOR	GGVNS	GGVNSMIP-PR
opm1	120	230,12	228,12	227,34	**1,21**	**0,34**
opm2	120	256,37	256,37	256,37	0	0
opm3	120	164039,12	164033,22	164029,15	**0,01**	0
opm4	120	164099,66	164066,24	164058,04	**0,03**	0
opm5	120	228,09	226,11	226,11	**0,87**	0
opm6	120	236,58	236,58	236,58	0	0
opm7	120	164021,38	164019,22	164017,73	0	0
opm8	120	164023,73	164020,84	164020,12	0	0

Como podemos observar na Tabela 3, o algoritmo paralelo PGGVNS foi capaz de gerar soluções de boa qualidade e com baixa variabilidade. De fato, o algoritmo conseguiu reduzir a variabilidade das soluções em até 0,83% quando comparado ao algoritmo GGVNS e em até 0,39% quando comparado ao algoritmo GGVNSMIP-PR. Analisando-se a Tabela 4 percebemos que para a instância opm1, o algoritmo proposto conseguiu melhorar a qualidade da solução final em 1,21% quando comparado ao algoritmo GGVNS e em 0,34% se comparado ao algoritmo GGVNSMIP-PR.

4. Conclusões

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões, POLAD.

Em virtude da complexidade combinatória do problema, foi proposto um algoritmo paralelo, denominado PGGVNS, que combina os procedimentos heurísticos Greedy Randomized Adaptive Search Procedure e General Variable Neighborhood Search em uma

arquitetura de programação paralela.

Usando-se problemas-teste da literatura, o algoritmo paralelo foi comparado com duas versões sequenciais propostas em trabalhos anteriores. O algoritmo proposto mostrou-se competitivo, sendo capaz de encontrar soluções de boa qualidade rapidamente e com baixa variabilidade das soluções finais.

Dado que a tomada de decisão no problema em pauta tem que ser rápida, os resultados encontrados validam a utilização do algoritmo PGGVNS enquanto ferramenta de apoio à decisão. Além disso, os resultados obtidos validam a proposta de paralelização utilizada, mostrando a robustez que podemos chegar utilizando o atual poder das máquinas multi-core.

Agradecimentos

Os autores agradecem às agências de fomento FAPEMIG e CNPq pelo apoio à execução do presente trabalho de pesquisa.

Referências

Aiex, Renata; Resende, Mauricio e Ribeiro, Celso. Tttplots: a perl program to create time-to-target plots. Optimization Letters, v.1, p.355-366, 2007.

Coelho, I. M.; Ribas, S. e Souza, M. J. F. Um algoritmo baseado em grasp, vnd e iterated local search para a resolução do planejamento operacional de lavra. XV Simpósio de Engenharia de Produção, Bauru/SP, 2008.

Coelho, V. N.; Souza, M. J. F.; Coelho, I. M. e Ribas, S. Busca geral em vizinhança variável com reconexão por caminhos para o planejamento operacional de lavra. XLII Simpósio Brasileiro de Pesquisa Operacional, p.1606-1617, Bento Gonçalves/RS, 2010.

Costa, F. P. Aplicações de técnicas de otimização a problemas de planejamento operacional de lavras em mina a céu aberto. Dissertação (Mestrado em Engenharia Mineral) – Escola de Minas, UFOP, Ouro Preto, 2005.

Feo, T. A. e Resende, M. G. C. Greedy randomized adaptive search procedures. Journal of Global Optimization, v.6, p.109-133, 1995.

Feo, T.A.; Resende, M.G.C. e Smith, S.H. A greedy randomized adaptive search procedure for maximum independent set. Operations Research, v.42, p.860-878, 1994.

Guimarães, I. F.; Pantuza, G. e Souza, M. J. F. Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. XIV Simpósio de Engenharia de Produção, 11 p., Bauru/SP, 2007.

Hansen, P. e Mladenovic, N. Variable neighborhood search: Principles and applications. European Journal of Operational Research, v.130, p.449-467, 2001.

Hansen, P.; Mladenovic, N. e Pérez, J. A. M. Variable neighborhood search. European Journal of Operational Research, v.191, p.593-595, 2008a.

Hansen, P.; Mladenovic, N. e Pérez, J. A. M. Variable neighborhood search: methods and applications. 4OR: Quarterly journal of the Belgian, French and Italian operations research societies, v.6, p.319-360, 2008b.

Lourenço, H. R.; Martin, O. C. e Stützle, T. Iterated local search. Glover, F. e Kochenberger, G., editors Handbook of Metaheuristics. Kluwer Academic Publishers, Boston, 2003.

Mladenovic, N. e Hansen, P. A variable neighborhood search. Computers and Operations Research, v.24, p.1097-1100, 1997.

Papadimitriou, C. H. e Steiglitz, K. Combinatorial Optimization: Algorithms and Complexity. Dover Publications, Inc., New York, 1998.

Resende, M. G. C. e Ribeiro, C. C. Greedy randomized adaptive search procedures: Advances and applications. Gendreau, M. e Potvin, J.Y., editors, Handbook of Metaheuristics. Springer, 2 edição, 2008.

Ribeiro, Celso e Resende, Mauricio. Path-relinking intensification methods for stochastic local search

algorithms. Journal of Heuristics, p.1-22, 2011.

Souza, M. J. F.; Coelho, I. M.; Ribas, S.; Santos, H. G. e Merschmann, L. H. C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. European Journal of Operational Research, v.207, p.1041-1051, 2010.

UM ALGORITMO BASEADO EM ESTRATÉGIAS EVOLUTIVAS PARA O PROBLEMA DE PLANEJAMENTO OPERACIONAL DE LAVRA

Vitor Nazário Coelho[1], Marcone Jamilson Freitas Souza[1], Igor Machado Coelho[2], Frederico Gadelha Guimarães[3], Bruno Nazário Coelho[1]

[1]Departamento de Ciência da Computação - Universidade Federal de Ouro Preto (UFOP)
Campus Universitário, Morro do Cruzeiro, CEP 35.400-000, Ouro Preto (MG), Brasil
vncoelho@gmail.com, marcone@iceb.ufop.br
brunonazario@gmail.com

[2]Departamento de Engenharia Elétrica - Universidade Federal de Minas Gerais (UFMG)
Campus Pampulha, Av. Antônio Carlos, 6627, CEP 31270-010, Belo Horizonte (MG), Brasil
frederico.g.guimaraes@gmail.com

[3]Instituto de Computação - Universidade Federal Fluminense (UFF)
Campus da Praia Vermelha, CEP 24210-240, Niterói (RJ), Brasil
imcoelho@gmail.com

Resumo

Este artigo apresenta um algoritmo evolutivo inspirado em Estratégias Evolutivas para resolução de um problema de programação inteira mista. O algoritmo proposto usa o procedimento (GRASP) para gerar a população inicial e é aplicado a um problema que requer decisões rápidas, o problema de Planejamento Operacional de Lavra com Alocação Dinâmica de Caminhões (POLAD). Para validá-lo, seus resultados são comparados com os produzidos por um algoritmo da literatura, denominado GGVNS, que não contempla o conceito de população. Resultados computacionais mostram a efetividade do algoritmo proposto.

Palavras-Chaves: Planejamento Operacional de Lavra, Programação Inteira Mista, Estratégias Evolutivas, GRASP, VND.

Abstract

This paper presents an evolutionary algorithm based on evolutionary strategies for solving a mixed integer programming problem. The proposed algorithm uses Greedy Randomized Adaptive Search Procedure (GRASP) to generate the initial population and it is applied to a problem that requires quick decisions, the Open-Pit-Mining Operational Planning problem with dynamic truck allocation (OPMOP). To validate the developed algorithm, its results are compared with those produced by a literature algorithm, called GGVNS, without the concept of population. Computational results show the effectiveness of the proposal.

Keywords: Open-pit mining, Mixed Integer Programming, Evolution strategies, Greedy Randomized Adaptive Search Procedure, Variable Neighborhood Descent.

1. Introdução

Este trabalho tem seu foco no planejamento operacional de lavra com alocação dinâmica de caminhões (POLAD). Esse problema envolve a alocação de carregadeiras às frentes de lavra (que podem ser de minério ou estéril), assim como a determinação do número de viagens que cada caminhão deve fazer a cada frente de forma que sejam atendidas tanto a meta de produção quanto a da composição mineral requerida para o minério. O objetivo é encontrar um ritmo de lavra em cada frente que minimize os desvios das metas de produção e qualidade, assim como o número de caminhões necessários ao processo produtivo.

Considera-se o sistema de alocação dinâmica de caminhões, isto é, a cada viagem realizada o caminhão pode se direcionar a uma frente diferente. Esse sistema de alocação contribui para o aumento da produtividade da frota e, consequentemente, para a redução do número de caminhões necessários ao processo produtivo.

O POLAD é um problema da classe NP-difícil e, como tal, métodos exatos de solução têm aplicabilidade restrita. A abordagem mais comum é por meio de procedimentos heurísticos.

Costa (2005) desenvolveu um algoritmo heurístico baseado em *Greedy Randomized Adaptive Search Procedures* - GRASP (FEO & RESENDE, 1995; RESENDE & RIBEIRO, 2008) e VNS (HANSEN & MLADENOVIC, 2001) para o POLAD usando seis tipos diferentes de movimentos para explorar o espaço de soluções. Foi feita uma comparação entre os resultados obtidos por esse algoritmo heurístico e os encontrados pelo otimizador LINGO, versão 7, aplicado ao modelo de programação matemática de Costa *et al*. (2004). Mostrou-se que o algoritmo heurístico foi capaz de encontrar soluções de melhor qualidade mais rapidamente.

Guimarães *et al*. (2007) apresentaram um modelo de simulação computacional para validar resultados obtidos pela aplicação de um modelo de programação matemática na determinação do ritmo de lavra em minas a céu aberto. Dessa maneira, foi possível validar os resultados da otimização, já que na modelagem de otimização não é possível tratar a variabilidade nos tempos de ciclo e a ocorrência de fila.

Em Coelho *et al*. (2008), o POLAD é resolvido por um algoritmo heurístico, denominado GVILS, que combina os procedimentos heurísticos GRASP, *Variable Neighborhood Descent* – VND (MLADENOVIC & HANSEN, 1997) e ILS (LOURENÇO *et al.*, 2003). O algoritmo GVILS faz uso de oito movimentos para explorar o espaço de soluções. Além dos desvios de produção e qualidade, procurou-se minimizar, também, o número de veículos. Usando quatro problemas-teste da literatura, o GVILS foi comparado com o otimizador CPLEX 9.1 aplicado a um modelo de programação matemática. Foram realizados testes envolvendo 15 minutos de processamento. Em dois dos problemas, o algoritmo proposto mostrou-se bastante superior; enquanto nos dois outros ele foi competitivo com o CPLEX, produzindo soluções médias com valores até 0,08% piores, na média.

Souza *et al*. (2010) propuseram um algoritmo, denominado GGVNS, que combina as metaheurísticas *General Variable Neighborhood Search* (GVNS) e o procedimento *Greedy Randomized Adaptative Search Procedure* (GRASP). Do procedimento GRASP utilizou-se a fase de construção para produzir soluções viáveis e de boa qualidade rapidamente. O GVNS foi escolhido devido a sua simplicidade, eficiência e capacidade natural de sua busca local para lidar com diferentes vizinhanças. Os autores compararam os resultados gerados pelo GGVNS com aqueles alcançados pelo otimizador CPLEX 11.01, utilizando oito problemas-teste. Os experimentos computacionais mostraram que o algoritmo proposto era competitivo com o CPLEX e capaz de encontrar soluções próximas do ótimo (com um *gap* < 1%) na maioria das instâncias, demandando um pequeno tempo computacional.

Por outro lado, a literatura tem mostrado que algoritmos evolutivos têm resolvido com eficiência vários problemas combinatórios (DE JONG *et al.*,1997; FREITAS e GUIMARÃES, 2011).

No presente trabalho investiga-se uma classe desses algoritmos, as chamadas Estratégias Evolutivas, denotadas por ES, do termo em inglês *Evolution Strategies* (BEYER & SCHWEFEL, 2002). Essas técnicas têm sido usadas na resolução de problemas inteiros ou mistos. Rajasekaran (2006) desenvolveu uma estratégia evolutiva combinada com redes neurais para resolução do problema de consistência do Concreto de Alta Performance (HPC). O algoritmo proposto foi comparado com um Algoritmo Genético – AG e com um outro baseado no procedimento *Simulated Annealing* – SA, obtendo, nos problemas-teste em questão, o melhor desempenho. Costa & Oliveira (2001) desenvolveram um algoritmo evolutivo baseado nos conceitos das ES para resolução de problemas não-lineares mistos, sendo que o algoritmo ES também foi comparado com um algoritmo AG clássico e com o procedimento SA, obtendo novamente o melhor desempenho nos problemas-teste analisados.

O algoritmo proposto, denominado GES, gera sua população inicial por meio de um procedimento parcialmente guloso e diversificado, baseado na metaheurística GRASP. Para mutação dos indivíduos, foram utilizadas as vizinhanças propostas em Souza *et al*. (2010), sendo a mutação o principal operador de busca no espaço de soluções. Para intensificar a busca, a cada geração uma pequena parte da população sofre uma busca local baseada no procedimento VND. O algoritmo proposto foi comparado ao GGVNS daqueles autores e se mostrou superior com relação à capacidade de encontrar melhores soluções mais rapidamente.

O restante deste trabalho está organizado como segue. A Seção 2 detalha o algoritmo proposto para resolver o POLAD. A Seção 3 mostra os resultados dos experimentos computacionais e a Seção 4 conclui o trabalho.

2. Metodologia

2.1. Modelo Exato

A formulação de programação matemática usada neste trabalho é a mesma de Coelho *et al.* (2008), em que se considera a função de avaliação mono-objetivo dada pela Eq. (1):

$$\min f^{PM}(s) = \sum_{j \in T} \lambda_j^- d_j^- + \sum_{j \in T} \lambda_j^+ d_j^+ + \alpha^- P_m^- + \alpha^+ P_m^+ + \beta^- P_e^- + \beta^+ P_e^+ + \sum_{l \in V} \omega_l U_l \qquad (1)$$

Na Eq. (1) busca-se minimizar os desvios positivos (d_j^+) e negativos (d_j^-) das metas de cada parâmetro de controle *j* da mistura, bem como minimizar os desvios positivos e negativos das metas de produção de minério e estéril, representados pelas variáveis de decisão P_m^+, P_m^-, P_e^+ e P_e^-, respectivamente. Nessa função também considera-se a minimização do número de veículos utilizados, representado pela variável binária U_l, que assume valor 1 se o veículo *l* for utilizado e 0, caso contrário.

As constantes λ_j^-, λ_j^+, α^-, α^+, β^-, β^+, e ω_l são pesos que refletem a importância de cada componente da função objetivo.

2.2. Modelo Heurístico

2.2.1. Representação de uma Solução

Dado um conjunto de frentes de lavra *F*, um conjunto de caminhões *V* e um conjunto de carregadeiras *K*, uma solução para o POLAD é representada por uma matriz $R = [Y \mid N]$, sendo *Y* a matriz $|F| \times 1$ e *N* uma matriz $|F| \times |V|$. Cada célula y_i da matriz $Y_{|F| \times 1}$ representa a carregadeira $k \in K$ alocada à frente $i \in F$. Um valor -1 significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente *i*, a carregadeira *k* associada a tal frente é considerada *inativa* e não é penalizada por produção abaixo da mínima para este equipamento de carga.

Na matriz $N_{|F| \times |V|}$, cada célula n_{il} representa o número de viagens do caminhão $l \in V$ à frente $i \in F$. Um valor 0 (zero) significa que não há viagem para aquele caminhão. O valor -1 informa a incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

Na Tabela 1, tem-se um exemplo de uma possível solução para o POLAD, observa-se que na coluna CARGA, linha F_1, a dupla $\langle Car_1, 1 \rangle$, indicando que o equipamento de carga Car_1 está alocado à frente F_1 e em operação. Na coluna CARGA, linha F_3, a dupla $\langle Car_8, 0 \rangle$ indica que o equipamento de carga Car_8 está alocado à frente F_3, mas não está em operação. Observa-se, ainda, na coluna CARGA, linha F_2, o valor $\langle D, 0 \rangle$ informando que não existe equipamento de carga alocado à frente F_2 e que, portanto, esta frente está disponível. As demais colunas representam o número de viagens a serem realizadas por um caminhão a uma frente, considerando a compatibilidade entre o caminhão e o equipamento de carga alocado à frente. As células com os valores X indicam incompatibilidade entre um caminhão e o respectivo equipamento de carga.

Tabela 1 – Representação de uma solução

	Carga	**Cam_1**	**Cam_2**	**...**	**Cam_V**
F_1	$<Car_1, 1>$	8	X	...	X
F_2	$<D, 0>$	0	0	...	0
F_3	$<Car_8, 0>$	0	0	...	0
...	...	...	...	...	...
F_F	$<Car_5, 1>$	0	9	...	3

2.2.2. Estruturas de Vizinhança

Como forma de explorar o espaço de soluções, foram utilizados os oito movimentos a seguir, os quais possuem uma boa capacidade exploratória, como relatado em Souza *et al*. (2010).

Movimento Número de Viagens - $N^{NV}(s)$: Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

Movimento Carga - $N^{CG}(s)$: Consiste em trocar duas células distintas y_i e y_k da matriz Y, ou seja, trocar os equipamentos de carga que operam nas frentes i e k, caso as duas frentes possuam equipamentos de carga alocados. Havendo apenas uma frente com equipamento de carga, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas junto com as frentes escolhidas.

Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$: Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl}. Assim, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$: Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik}. Isto é, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Operação Frente - $N^{OF}(s)$: Consiste em retirar de operação o equipamento de carga que esteja em operação na frente i. O movimento retira todas as viagens feitas a esta frente, deixando o equipamento *inativo*. O equipamento retorna à operação assim que uma nova viagem é associada a ele.

Movimento Operação Caminhão - N^{OC}(s): Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, isto é, retirar de atividade um caminhão l que esteja operando em uma frente i.

Movimento Troca de Viagens - N^{VT}(s): Duas células da matriz N são selecionadas e uma unidade de uma célula passa para a outra, isto é, uma viagem de um caminhão associado a uma frente i passa para outro caminhão associado outra frente.

Movimento Troca de Carregadeiras - N^{CT}(s): Duas células distintas y_i e y_k da matriz Y tem seus valores permutados, ou seja, os equipamentos de carga que operam nas frentes i e k são trocados. Analogamente ao movimento *CG*, os equipamentos de carga são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

2.2.3 Avaliação de uma Solução

Como os movimentos usados podem gerar soluções inviáveis, uma solução é avaliada por uma função f, a ser minimizada, composta por duas parcelas. A primeira delas é a função objetivo propriamente dita, f^{PM} , dada pela Eq. (1), e a segunda é composta pelas funções que penalizam a ocorrência de inviabilidade na solução corrente. Assim, a função f, definida pela Eq. (2), mensura o desvio dos objetivos considerados e penaliza o não atendimento às restrições do problema.

$$f(s) = f^{PM}(s) + f^{P}(s) + \sum_{j \in T} f_j^q(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \quad (2)$$

em que:

- f^{PM}(s) é uma função que avalia s quanto ao atendimento às metas de produção e qualidade, bem como o número de caminhões utilizados (mesma do modelo de programação matemática, Subseção 2.1);
- f^{P}(s) avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril;
- f_q^j (s) avalia s quanto à inviabilidade em relação ao j-ésimo parâmetro de controle;
- f_u^l (s) avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l-ésimo caminhão;
- f_c^k(s), que avalia s quanto ao desrespeito aos limites de produtividade da carregadeira k.

2.2.4 Representação de um Indivíduo

Um dado indivíduo possui, além da matriz de solução $R = [Y \mid N]$ definida na Subseção 2.2.1, dois vetores de parâmetros de mutação.

O primeiro vetor diz a probabilidade de aplicação de cada um dos movimentos descritos na Subseção 2.2.2; logo, ele é um vetor de números reais, em que cada posição i diz a probabilidade de aplicação de um dado movimento. Já o segundo vetor de parâmetros que compõe a representação de um indivíduo é um vetor de números inteiros e regula a intensidade da perturbação, ou seja, cada posição i deste vetor limita o número de aplicações de um dado movimento, caso ele venha a ser aplicado.

Desta forma, tem-se um vetor de probabilidades P, tal que:

$$P = [p_1, p_2, ..., p_i, ..., p_8] \quad (3)$$

sendo $p_i \in [0, 1], p_i \in \Re$.

Já para o vetor de aplicações, tem-se:

$$A = [a_1, a_2, ..., a_i, ..., a_8] \quad (4)$$

sendo $a_i \in [0, nap_i], a_i \in Z^+$, com nap_i representando o número máximo de aplicações para um dado movimento i.

2.3 Algoritmo Proposto

O algoritmo proposto neste trabalho, denominado *GES*, consiste na combinação dos procedimentos heurísticos *GRASP* e segue os passos de uma Estratégia Evolutiva. Seu pseudocódigo está esquematizado na Figura 1.

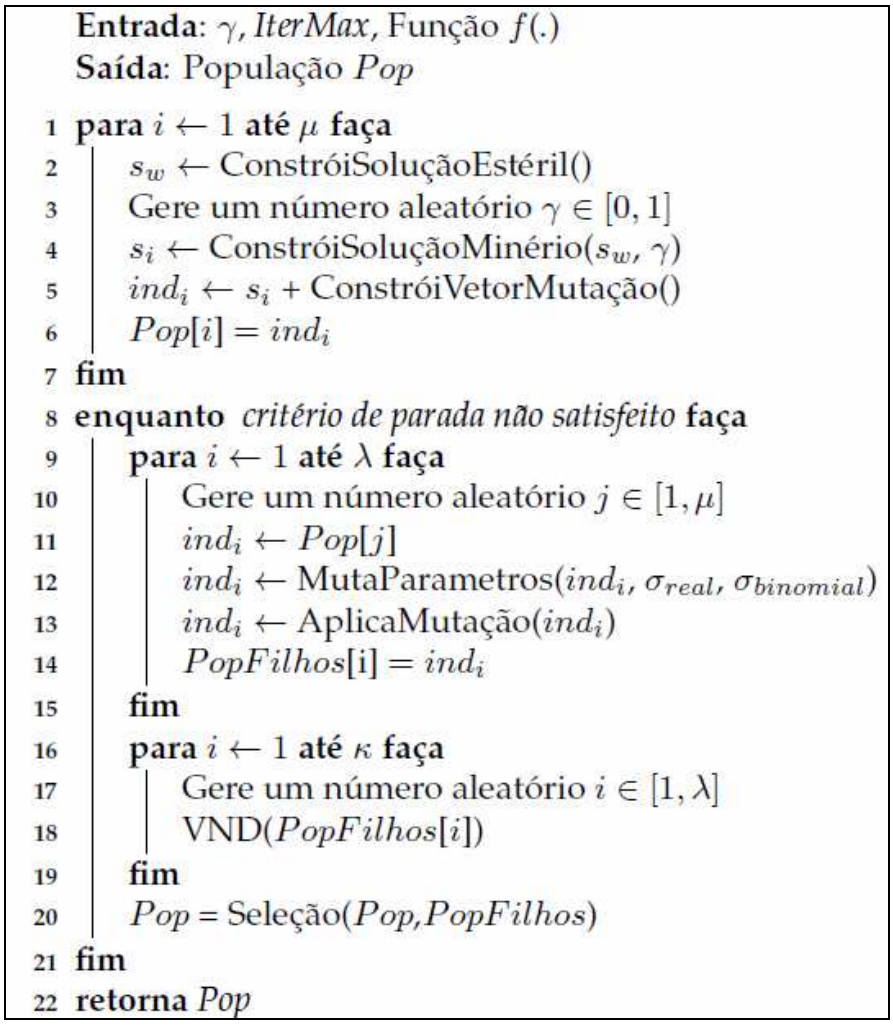

Figura 1 – Algoritmo *GES*

A população inicial do algoritmo (linhas 1 a 7 da Figura 1) é composta por μ indivíduos e é criada em duas etapas.

Na primeira, realiza-se a construção da matriz de solução $R = [Y \mid N]$ de cada indivíduo da população. Para esta etapa são utilizados os procedimentos *ConstróiSoluçãoEstéril* e *ConstróiSoluçãoMinério* descritos em Souza *et al*. (2010). A obtenção de uma população inicial diversificada é de extrema importância para a convergência do algoritmo; logo, o parâmetro γ que define o tamanho da lista restrita de candidatos varia em cada um dos indivíduos.

Na segunda etapa (linha 5 da Figura 1), é feita a construção dos vetores de mutação de cada um dos indivíduos. Para o vetor P de probabilidades utiliza-se uma Distribuição Normal. Já para o vetor de aplicações A utiliza-se uma Distribuição Binomial.

Na linha 12 da Figura 1 é acionado o procedimento de mutação dos parâmetros para um dado

indivíduo, cujo pseudocódigo está descrito na Figura 2.

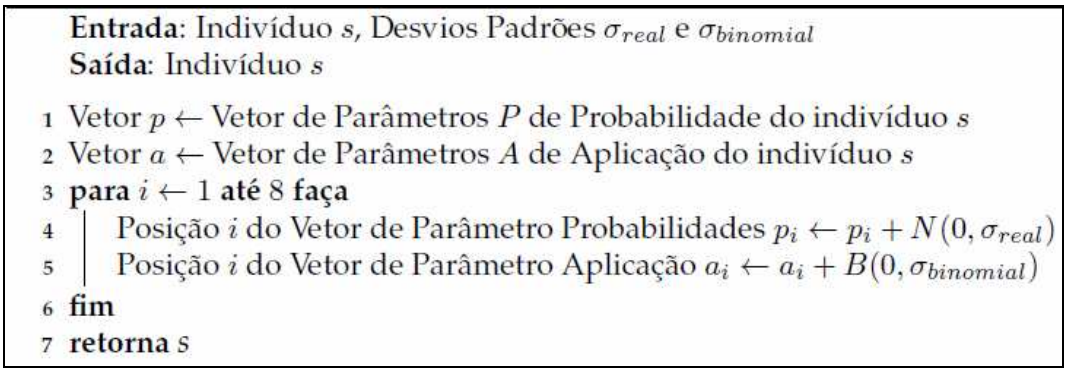

Figura 2 – Algoritmo MutaParâmetros

Para cada posição *i* dos vetores de parâmetros da Figura 2 aplica-se uma Distribuição Normal ou Binomial, ambas centradas com média zero e desvio-padrão σ_{real} e $\sigma_{binomial}$, respectivamente. Este procedimento pode ser visto nas linhas 4 e 5 desta figura. Para a mutação do vetor de probabilidades *P* aplica-se uma perturbação seguindo uma Distribuição Normal com desvio σ_{real}; já para a mutação do vetor aplicações *A* aplica-se uma perturbação seguindo uma Distribuição Binomial com desvio $\sigma_{binomial}$.

O procedimento AplicaMutação, linha 13 da Figura 1, está descrito na Figura 3.

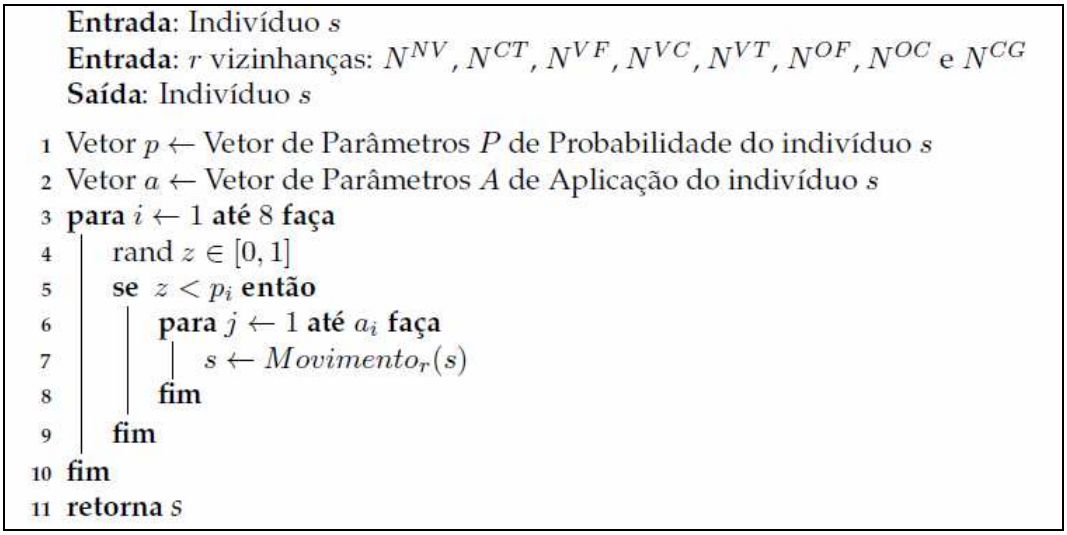

Figura 3 – Algoritmo AplicaMutação

Na linha 4 da Figura 3 é gerado um número aleatório $z \in [0,1]$ e, em seguida, é verificado se este número satisfaz a probabilidade p_i do vetor de probabilidades. Caso afirmativo, aplica-se a_i vezes um dos oito movimentos (seção 2.2.2) referentes à posição *i*. A ordem da vizinhança neste vetor de parâmetros é escolhida aleatoriamente.

O procedimento VND de busca local (linha 18 da Figura 1) é opcional, sendo aplicado apenas em algumas versões do algoritmo proposto. Quando este procedimento é acionado, realiza-se um busca local de acordo com o procedimento VND (descrito pela Figura 4) em κ filhos. Para esta busca local, usa-se, apenas, um grupo restrito dos movimentos descritos na seção 2.2.2, no caso, apenas aqueles que definem as vizinhanças N^{CG}, N^{NV}, N^{VC} e N^{VF}. A justificativa para essa restrição é que a busca local do algoritmo é muito custosa computacionalmente. Estrategicamente, a busca local opera nessas vizinhanças em uma ordem aleatória, definida a cada chamada. Esse resultado foi alcançado após uma bateria de testes preliminares, a qual

mostrou que não havia uma ordem de vizinhança que resultasse, sempre, na geração de soluções melhores. Na seção 3 o resultado da adição deste procedimento é mostrado.

Entrada: Indivíduo s e Funcão de Avaliação f
Entrada: r vizinhanças na ordem aleatória: N^{VF}, N^{VC}, N^{NV} e N^{CG}
Saída: Indivíduo s^* de qualidade possivelmente superior à s de acordo com a função f

```
1  k ← 1
2  enquanto k ≤ r faça
3  |  Encontre o melhor vizinho s' ∈ N^(k)(s)
4  |  se f(s') < f(s) então
5  |  |  s ← s'; k ← 1
6  |  fim
7  |  senão
8  |  |  k ← k + 1
9  |  fim
10 fim
11 retorna s
```

Figura 4 – Algoritmo *VND*

O procedimento de Seleção (linha 20 da Figura 1) pode ser qualquer estratégia de seleção desejada, desde que esta retorne uma população de tamanho μ. Foram utilizadas duas formas básicas de competição, ambas com a mesma notação de Beyer & Schwefel (2002). Na primeira delas, denotada por (μ + λ), ocorre uma competição entre pais e filhos. Nesta estratégia são selecionados os μ melhores indivíduos dentre pais e filhos. Já na segunda estratégia de seleção utilizada, denotada por (μ, λ), os indivíduos que sobrevivem para a próxima geração são os μ melhores filhos. É notório que utilizando a estratégia (μ, λ) como forma de seleção a população que sobrevive para próxima geração sofre uma considerável pressão seletiva; porém, esta pressão torna-se ainda maior quando a estratégia (μ + λ) é utilizada.

3. Experimentos Computacionais e Análises

O algoritmo *GES* proposto foi implementado em C++ usando o *framework* de otimização OptFrame, disponível em https://sourceforge.net/projects/optframe, e compilado pelo g++ 4.13, utilizando a IDE Eclipse 3.1. Os experimentos foram testados em um microcomputador Pentium Core 2 Quad(Q6600), com 8 GB de RAM, no sistema operacional Ubuntu 10.10. Para testá-lo, foi usado um conjunto de 8 problemas-teste da literatura, disponíveis em http://www.iceb.ufop.br/decom/prof/marcone/projects/mining.html. Estes problemas-teste foram os mesmos utilizados em Souza *et al*. (2010) para validar o algoritmo *GGVNS*.

Os melhores resultados da literatura para os problemas-teste analisados são apresentados na Tabela 2. Na coluna "Opt." Indica-se por "√" os problemas-teste nos quais o otimizador matemático CPLEX 11.02 obteve o valor ótimo da função.

Tabela 2 – Melhores valores

Problema-Teste	Melhor da Literatura	Opt
opm1	227,12	
opm2	256,37	
opm3	164.027,15	√
opm4	164.056,68	√
opm5	227,04	
opm6	236,58	
opm7	164.017,46	√
opm8	164.018,65	√

A Tabela 3 mostra seis variantes do algoritmo *GES*, criadas a partir de diferentes valores para seus parâmetros. As variantes *GES-VND1* e *GES-VND2* incluem o procedimento VND (descrito na Figura 4) como método de busca local. Nestas duas variantes uma parcela de κ=3 indivíduos da população de filhos sofre esse procedimento de busca local. O número de indivíduos que sofrem este procedimento de busca local é pequeno em relação à população de filhos visto que, como já citado anteriormente, a busca local utilizada é muito custosa computacionalmente.

Tabela 3 – Variantes do Algoritmo *GES*

Algoritmo	μ	λ	**Seleção**	***VND***
GES1	30	160	$(\mu; \lambda)$	
GES2	30	160	$(\mu + \lambda)$	
GES3	100	600	$(\mu; \lambda)$	
GES4	100	600	$(\mu + \lambda)$	
GES-VND1	30	160	$(\mu; \lambda)$	√
GES-VND2	30	160	$(\mu + \lambda)$	√

Primeiramente, foi realizado um experimento de probabilidade empírica, *Time-to-target* (TTT) *plots*, na forma indicada por Aiex *et al*. (2007), de forma a verificar a eficiência das variantes do algoritmo elencadas na Tabela 2. Este teste mostra, no eixo das ordenadas, a probabilidade de um algoritmo em encontrar uma solução boa em um dado limite de tempo, eixo das abscissas. TTT *plots* já foi utilizado por vários autores, como Hoos & Stutzle (1998), e continua sendo defendido (RIBEIRO, 2011) como uma forma de caracterizar tempo de execução de algoritmos estocásticos aplicados a problemas de otimização combinatória.

Foram realizadas 120 execuções com cada uma das seis variantes do algoritmo desenvolvido. O problema-teste utilizado foi o opm1, tendo os seguintes valores como alvo o valor 229,00 e tempo limite de 120 segundos. A Figura 5 mostra as curvas obtidas.

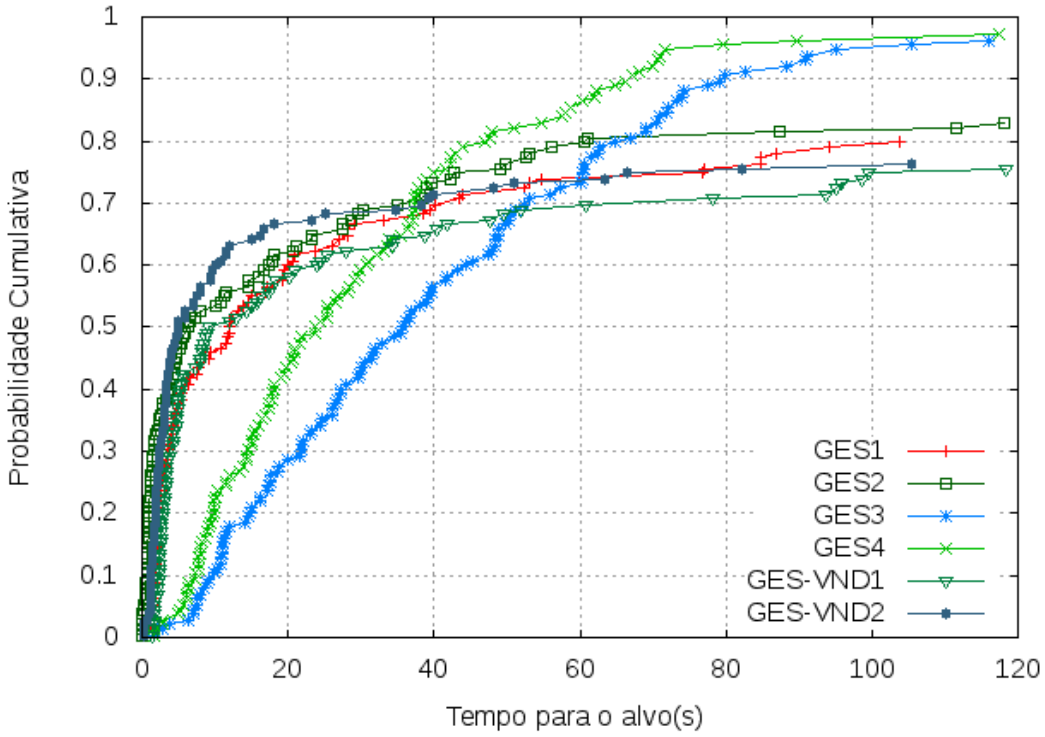

Figura 5 – Curva de Probabilidade Empírica - Instância opm1

Analisando as curvas de probabilidade empírica da Figura 5, percebe-se que as variantes que utilizaram a estratégia de seleção $(\mu + \lambda)$ prevaleceram em relação a suas versões com

estratégia (μ, λ). Este fato mostra que a competição entre pais e filhos fez com que indivíduos com um bom potencial de otimalidade permanecessem por mais gerações.

Nos instantes iniciais da busca, a variante *GES-VND2* foi capaz de gerar melhores soluções do que os outros algoritmos propostos. No entanto, a partir de 40 segundos esta variante perde seu desempenho, sendo superada pela variante *GES4*, a qual continua progredindo sistematicamente, sendo a primeira a alcançar o alvo desejado com uma probabilidade de aproximadamente 100%.

Pela Figura 5, verifica-se que as variantes que utilizam o método de busca local VND, *GES-VND1* e *GES-VND2*, têm ótimo desempenho no início da busca, mas convergem prematuramente. Isto é devido ao fato de que o VND contribui para gerar super-indivíduos que causam estagnação da população depois de alguns instantes de execução do algoritmo. Por outro lado, pode-se comprovar a força do mecanismo de mutação das Estratégias Evolutivas quando é analisada a convergência das variantes que não usam o procedimento de busca local.

Desta forma, tendo em vista a robustez da variante *GES4*, foi feita uma comparação entre os algoritmos *GES*4 e o algoritmo *GGVNS*, de Souza *et al.*, (2010).

A Tabela 3 mostra a comparação entre algoritmo *GES4* e o algoritmo *GGVNS*. Nessa tabela, a coluna "Instância" indica o problema-teste utilizado. A coluna "IMPdesv" menciona o ganho relativo do algoritmo *GES*4 em relação ao algoritmo *GGVNS*, ou seja:

$$IMPdesv_i = \frac{\bar{f}_i^{GGVNS} - \bar{f}_i^{GES4}}{\bar{f}_i^{GGVNS}} \quad (5)$$

Já a coluna "IMPbest" indica o percentual de melhora proporcionado pelo algoritmo *GES4* em relação ao valor da melhor solução encontrada pelo algoritmo *GGVNS*.

$$IMPbest_i = \frac{f_i^{*GGVNS} - f_i^{*GES4}}{f_i^{*GGVNS}} \quad (6)$$

Tabela 4 – Comparação de resultados: GGVNS x GES4

Instância	Tempo (s)	*GGVNS*		*GES4*		*IMPbest*	*IMPgap*
		MEDIA	MELHOR	MEDIA	MELHOR		
opm1	2	230,12	230,12	228,50	228,12	**0,87%**	**0,70%**
opm2	2	256,56	256,37	256,43	256,37	0,00%	**0,05%**
opm3	2	164064,68	164039,12	164044,68	164031,28	0,00%	**0,01%**
opm4	2	164153,92	164099,66	164097,61	164057,04	**0,03%**	**0,03%**
opm5	2	228,09	228,09	227,21	226,66	**0,63%**	**0,39%**
opm6	2	237,97	236,58	237,07	236,58	0,00%	**0,38%**
opm7	2	164021,89	164021,38	164020,24	164018,22	0,00%	0,00%
opm8	2	164027,29	164023,73	164022,38	164020,26	0,00%	0,00%

Analisando a Tabela 5 podemos verificar que o algoritmo *GES4* proposto foi capaz de gerar soluções de boa qualidade e baixa variabilidade em relação ao algoritmo *GGVNS*. Percebe-se que ele conseguiu melhorar a qualidade das soluções finais em até 0,87%, e reduzir a variabilidade dessas soluções em até 0,39%. Além disso, tendo em vista a Tabela 2, pode ser observado que para o problema-teste opm5 o *GES4* foi capaz de encontrar uma solução melhor que a da literatura.

4. Conclusões

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões (POLAD). Em virtude de sua dificuldade de solução, foi proposto um algoritmo populacional, denominado *GES*, que combina o poderio do GRASP com as Estratégias Evolutivas, percorrendo um espaço de busca de soluções inteiras.

Seis variantes desse algoritmo foram desenvolvidas e comparadas entre si. Dessas, quatro eram algoritmos baseados em Estratégia Evolutiva tendo como principal mecanismo de busca no espaço a mutação e diferiam entre si pelos parâmetros de tamanho da população e estratégia de seleção. As outras duas incluíam um procedimento de busca local, baseado em VND, na exploração do espaço de soluções. Verificou-se a convergência prematura das variantes que usam o procedimento de busca local e optou-se pela variante que mostrou ser mais eficiente em alcançar o valor alvo.

Usando problemas-teste da literatura, essa variante do algoritmo evolutivo, denotada por GES4, foi comparada com o algoritmo *GGVNS* de Souza *et al*. (2010). Os resultados mostraram que o algoritmo evolutivo proposto é competitivo, apresentando boas soluções com uma menor variabilidade em torno da média.

Para trabalhos futuros é proposto o uso de novas estratégias de perturbação ao vetor de parâmetros de cada indivíduo, assim como variar a etapa de seleção durante a execução do algoritmo, de forma que o algoritmo entre em maior harmonia no quesito *Exploration-Exploitation*. Além disso, propõe-se uma melhoria no mecanismo de auto-adaptação, bem como uma melhor calibragem dos parâmetros das distribuições utilizadas. A adição do módulo de busca local apenas em determinadas gerações é outra estratégia que pode ser testada. Finalmente, propõe-se a implementação de uma versão paralela do algoritmo *GES* visando tirar proveito da tecnologia *multi-core*, já presente nas máquinas atuais e de fácil abstração para algoritmos populacionais.

Agradecimentos

Os autores agradecem à FAPEMIG e ao CNPq pelo apoio à execução do presente trabalho de pesquisa.

Referências

Aiex R. M., Resende, M.G.C. & Ribeiro, C.C. TTTPLOTS: a perl program to create time-to-target plots. *Optimization Letters*, Vol. 1, n.4, p.355-366, 2007.

Beyer, H. G. & Schwefel, H. P. Evolution strategies - a comprehensive introduction. *Natural Computing*, Vol. p.3-52, 2002.

Coelho, I. M., Ribas, S., & Souza, M. J. F. Um algoritmo baseado em grasp, vnd e iterated local search para a resolução do planejamento operacional de lavra. In *XV Simpósio de Engenharia de Produção* - SIMPEP, Bauru/SP, 2008.

Coelho, I. M., Ribas, S., Souza, M. J. F., & Coelho, V. N. Um algoritmo heurístico híbrido para o planejamento operacional de lavra. In *Anais do IX Congresso Brasileiro de Redes Neurais* - CBRN, 7 p., Ouro Preto, MG, 2009.

Costa, F. P. *Aplicações de técnicas de otimização a problemas de planejamento operacional de lavra em minas a céu aberto*. Dissertação, Programa de Pós-Graduação em Engenharia Mineral, Escola de Minas, UFOP, Ouro Preto, 2005.

Costa, F. P., Souza, M. J. F., & Pinto, L. R. Um modelo de alocação dinâmica de caminhões. *Revista Brasil Mineral*, Vol. 231, p.26-31, 2004.

Costa, L. & Oliveira, P. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers & Chemical Engineering*, Vol. 25, n.10,

p.257-266, 2001.

De Jong, K., David, D., Fogel, B. & Schwefel, H.P. *A history of evolutionary computation.*In: Bäck T, Fogel DB and Michalewicz Z (eds). Handbook of Evolutionary Computation, Vol. A2, n.3, p.1-12. Oxford University Press, New York, and Institute of Physics Publishing,Bristol, 1997.

Feo, T. A. & Resende, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, Vol. 6, p.109-133, 1995.

Freitas, A. R. R.; Guimarães, F. G. Originality and Diversity in the Artificial Evolution of Melodies. In: Genetic and Evolutionary Computation Conference (GECCO-2011), 2011, Dublin. *Proceedings of the 13th annual Conference on Genetic and Evolutionary Computation*. New York: ACM Press, 2011.

Guimarães, I. F., Pantuza, G., & Souza, M. J. F. Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. In *Anais do XIV Simpósio de Engenharia de Produção* - SIMPEP, 11 p., Bauru, CD-ROM, 2007.

Hansen, P. & Mladenovic, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, Vol. 130, p.449-467, 2001.

Hoos, H. H. & Stutzle, T. On the empirical evaluation of Las Vegas algorithms - Position paper. *Technical report*, Computer Science Department, University of British Columbia, 1998.

Lourenço, H. R., Martin, O. C., & Stützle, T. *Iterated local search*. In Glover, F. and Kochenberger, G., editors, Handbook of Metaheuristics. Kluwer Academic Publishers, Boston, 2003.

Mladenovic, N. & Hansen, P. A variable neighborhood search. *Computers and Operations Research*, Vol. 24, p.1097-1100, 1997.

Rajasekaran, S. Optimal mix for high performance concrete by evolution strategies combined with neural networks. *Indian Journal of Engineering and Materials Sciences*, Vol. 13, n.11, p.7-17, 2006.

Resende, M. G. C. & Ribeiro, C. C. *Greedy randomized adaptive search procedures: Advances and applications*. In Gendreau, M. and Potvin, J., editors, Handbook of Metaheuristics. Springer, 2 edition. (to appear). Available at: http://www2.research.att.com/ mgcr/doc/sgrasp2008a.pdf, 2008.

Ribeiro, C.C. & Resende, M. G. C. Path-relinking intensification methods for stochastic local search algorithms. *Journal of Heuristics*, Springer Netherlands, p.1-22, 2011.

Souza, M. J. F., Coelho, I. M., Ribas, S., Santos, H. G., & Merschmann, L. H. C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, Vol. 207, n.2, p.1041-1051, 2010.

X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

ESTRATÉGIAS EVOLUTIVAS APLICADAS A UM PROBLEMA DE PROGRAMAÇÃO INTEIRA MISTA

V. N. Coelho[1], **M. F. J. Souza**[1], **I. M. Coelho**[2]
[1]Universidade Federal de Ouro Preto, [2]Universidade Federal Fluminense
vncoelho@gmail.com, marcone@iceb.com.br, imcoelho@ic.uff.br

F. G. Guimaraes[3], **B. N. Coelho**[1]
[3]Universidade Federal de Minas Gerais
fredericoguimaraes@ufmg.br, brunonazario@gmail.com

Resumo – Este artigo apresenta um algoritmo evolutivo inspirado em Estratégias Evolutivas para resolução de um problema de programação inteira mista. O algoritmo proposto usa o procedimento *Greedy Randomized Adaptive Search Procedure* (GRASP) para gerar a população inicial e é aplicado a um problema que requer decisões rápidas, o problema de Planejamento Operacional de Lavra com Alocação Dinâmica de Caminhões (POLAD). Para validá-lo, seus resultados são comparados com os produzidos por um algoritmo da literatura, denominado GGVNS, que não contempla o conceito de população. Resultados computacionais mostram a efetividade do algoritmo proposto.

Palavras-chave – Planejamento Operacional de Lavra, Programação Inteira Mista, Estratégias Evolutivas, GRASP, VND.

Abstract – This paper presents an evolutionary algorithm based on evolutionary strategies for solving a mixed integer programming problem. The proposed algorithm uses Greedy Randomized Adaptive Search Procedure (GRASP) to generate the initial population and it is applied to a problem that requires quick decisions, the Open-Pit-Mining Operational Planning problem with dynamic truck allocation (OPMOP). To validate the developed algorithm, its results are compared with those produced by a literature algorithm, called GGVNS, without the concept of population. Computational results show the effectiveness of the proposal.

Keywords – Open-pit mining, Mixed Integer Programming, Evolution strategies, Greedy Randomized Adaptive Search Procedure, Variable Neighborhood Descent.

1. INTRODUÇÃO

Este trabalho tem seu foco no planejamento operacional de lavra com alocação dinâmica de caminhões (POLAD). Esse problema envolve a alocação de carregadeiras às frentes de lavra (que podem ser de minério ou estéril), assim como a determinação do número de viagens que cada caminhão deve fazer a cada frente de forma que sejam atendidas tanto a meta de produção quanto a da composição mineral requerida para o minério. O objetivo é encontrar um ritmo de lavra em cada frente que minimize os desvios das metas de produção e qualidade, assim como o número de caminhões necessários ao processo produtivo.

Considera-se o sistema de alocação dinâmica de caminhões, isto é, a cada viagem realizada o caminhão pode se direcionar a uma frente diferente. Esse sistema de alocação contribui para o aumento da produtividade da frota e, consequentemente, para a redução do número de caminhões necessários ao processo produtivo.

O POLAD é um problema da classe NP-difícil e, como tal, métodos exatos de solução têm aplicabilidade restrita. A abordagem mais comum é por meio de procedimentos heurísticos. [1] desenvolveu um algoritmo heurístico baseado em *Greedy Randomized Adaptive Search Procedure* - GRASP [2–4] e VNS [5,6] para o POLAD usando seis tipos diferentes de movimentos para explorar o espaço de soluções. Foi feita uma comparação entre os resultados obtidos por esse algoritmo heurístico e os encontrados pelo otimizador LINGO, versão 7, aplicado a um modelo de programação matemática desenvolvida pelos autores, publicado em [7]. Mostrou-se que o algoritmo heurístico foi capaz de encontrar soluções de melhor qualidade mais rapidamente. [8] apresentaram um modelo de simulação computacional para validar resultados obtidos pela aplicação de um modelo de programação matemática na determinação do ritmo de lavra em minas a céu aberto. Dessa maneira, foi possível validar os resultados da otimização, já que na modelagem de otimização não é possível tratar a variabilidade nos tempos de ciclo e a ocorrência de fila. Em [9], o POLAD é resolvido por um algoritmo heurístico, denominado GVILS, que combina os procedimentos heurísticos GRASP, Variable Neighborhood Descent – VND [5] e ILS [10]. O algoritmo GVILS faz uso de oito movimentos para explorar o espaço de soluções. Além dos desvios de produção e qualidade, procurou-se minimizar, também, o número de veículos. Usando quatro problemas-teste da literatura, o GVILS foi comparado com o otimizador CPLEX 9.1 aplicado a um modelo de programação matemática. Foram realizados testes envolvendo 15 minutos de processamento. Em dois dos problemas, o algoritmo proposto mostrou-se bastante superior; enquanto nos dois outros ele foi competitivo com o CPLEX, produzindo soluções médias com valores até 0,08% piores, na média. [11] propuseram um algoritmo, denominado GGVNS, que combina as metaheurísticas *General Variable Neighborhood Search* - GVNS [12] e o procedimento GRASP. Do procedimento GRASP

utilizou-se a fase de construção para produzir soluções viáveis e de boa qualidade rapidamente. O GVNS foi escolhido devido a sua simplicidade, eficiência e capacidade natural de sua busca local para lidar com diferentes vizinhanças. Os autores compararam os resultados gerados pelo GGVNS com aqueles alcançados pelo otimizador CPLEX 11.01, utilizando oito problemas-teste. Os experimentos computacionais mostraram que o algoritmo proposto era competitivo com o CPLEX e capaz de encontrar soluções próximas do ótimo (com um $gap < 1\%$) na maioria das instâncias, demandando um pequeno tempo computacional.

Por outro lado, a literatura tem mostrado que algoritmos evolutivos têm resolvido com eficiência vários problemas combinatórios [13, 14]. No presente trabalho investiga-se uma classe desses algoritmos, as chamadas Estratégias Evolutivas, *Evolution Strategies* - ES, [15]. Essas técnicas têm sido usadas na resolução de problemas inteiros ou mistos. [16] desenvolveu uma estratégia evolutiva combinada com redes neurais para resolução do problema de consistência do Concreto de Alta Performance (HPC). A estretégia evolutiva foi comparada com um Algoritmo Genético (AG) e com o procedimento *Simulated Annealing* (SA), obtendo, nos problemas-teste em questão, o melhor desempenho. Já em [17], os autores desenvolveram um algoritmo evolutivo baseado nos conceitos das Estratégias Evolutivas (ES, dos termos em inglês *Evolutive Strategies*) para resolução de problemas não-lineares mistos, sendo que o algoritmo ES foi também comparado com um algoritmo GA clássico e com o procedimento SA, obtendo novamente o melhor desempenho nos problemas-teste analisados.

O algoritmo proposto, denominado GES, gera sua população inicial por meio de um procedimento parcialmente guloso diversificado, baseado na metaheurística GRASP. Para mutação dos indivíduos, foram utilizadas as vizinhanças propostas em [11], sendo a mutação o principal operador de busca no espaço de soluções. Para intensificar a busca, a cada geração uma pequena parte da população sofre uma busca local baseada no procedimento VND. O algoritmo proposto foi comparado ao GGVNS daqueles autores e se mostrou superior com relação à capacidade de encontrar melhores soluções mais rapidamente.

por isso as ES deixam de ser um grande instrumento para este tipo de aplição de Otimização, este trabalho mostra o poderio dessa classe algoritmo populacionais no presente momento.

A abstração e aplicação dos conceitos das Estratégias Evolutivas nesse problema não foi encontrada na literatura. Assim, este trabalho mostra como ele pode ser modelado por essa classe de algoritmos populacionais, assim como o poderio dela.

O restante deste trabalho está organizado como segue. A Seção 2 detalha o algoritmo proposto para resolver o POLAD. A Seção 3 mostra e analisa os resultados dos experimentos computacionais, enquanto a Seção 4 conclui o trabalho.

2. METODOLOGIA

2.1 MODELO MATEMÁTICO

A formulação de programação matemática usada neste trabalho é a mesma de [11]. Nesta formulação, considera-se a função de avaliação mono-objetivo dada pela Eq. (1):

$$\min f^{PM}(s) = \sum_{j \in T} \lambda_j^- d_j^- + \sum_{j \in T} \lambda_j^+ d_j^+ + \alpha^- P_m^- + \alpha^+ P_m^+ \beta^- P_e^- + \beta^+ P_e^+ + \sum_{l \in V} \omega_l U_l \quad (1)$$

Na Eq. (1), busca-se minimizar os desvios positivos (d_j^+) e negativos (d_j^-) das metas de cada parâmetro de controle j da mistura, bem como minimizar os desvios positivos e negativos das metas de produção de minério e estéril, representados pelas variáveis de decisão P_m^+, P_m^-, P_e^+ e P_e^-, respectivamente. Nessa função também considera-se a minimização do número de veículos utilizados, representado pela variável binária U_l, que vale 1 se o veículo l for utilizado e 0, caso contrário.

As constantes λ_j^-, λ_j^+, α^-, α^+, β^-, β^+ e ω_l são pesos que refletem a importância de cada componente da função objetivo.

2.2 MODELO HEURÍSTICO

2.2.1 REPRESENTAÇÃO DE UMA SOLUÇÃO

Uma solução é representada por uma matriz $R = [Y|N]$, sendo Y a matriz $|F| \times 1$ e N a matriz $|F| \times |V|$. Cada célula y_i da matriz $Y_{|F|\times 1}$ representa a carregadeira k alocada à frente i. O valor -1 significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente i, a carregadeira k associada a tal frente é considerada *inativa* e não é penalizada por produção abaixo da mínima para este equipamento de carga.

Na matriz $N_{|F|\times|V|}$, cada célula n_{il} representa o número de viagens do caminhão $l \in V$ a uma frente $i \in F$. O valor 0 (zero) significa que não há viagem para aquele caminhão. O valor -1 informa a incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

Tabela 1: Representação de uma Solução

	$Carga$	Cam_1	Cam_2	...	Cam_V
F_1	$< Car_1, 1 >$	8	X	...	X
F_2	$< D, 0 >$	0	0	...	0
F_3	$< Car_8, 0 >$	0	0	...	0
...	...	...	...	...	...
F_F	$< Car_5, 1 >$	0	9	...	3

Na Tabela 1 temos um exemplo de uma possível solução para o POLAD, observa-se que na coluna CARGA, linha F_1, a dupla $\langle Car_1, 1\rangle$, indicando que o equipamento de carga Car_1 está alocado à frente F_1 e em operação. Na coluna CARGA, linha

F_3, a dupla $\langle Car_8, 0\rangle$ indica que o equipamento de carga Car_8 está alocado à frente F_3, mas não está em operação. Observa-se, ainda, na coluna $CARGA$, linha F_2, o valor $\langle D, 0\rangle$ informando que não existe equipamento de carga alocado à frente F_2 e que, portanto, esta frente está disponível. As demais colunas representam o número de viagens a serem realizadas por um caminhão a uma frente, considerando a compatibilidade entre o caminhão e o equipamento de carga alocado à frente. As células com os valores X indicam incompatibilidade entre um caminhão e o respectivo equipamento de carga.

2.2.2 ESTRUTURAS DE VIZINHANÇA

Como forma de explorar o espaço de soluções, foram utilizados oito movimentos, os quais possuem uma boa capacidade exploratória, como relatado em [11].

Os movimentos são descritos abaixo:

Movimento Número de Viagens - $N^{NV}(s)$: Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

Movimento Carga - $N^{CG}(s)$: Consiste em trocar duas células distintas y_i e y_k da matriz Y, ou seja, trocar os equipamentos de carga que operam nas frentes i e k, caso as duas frentes possuam equipamentos de carga alocados. Havendo apenas uma frente com equipamento de carga, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas junto com as frentes escolhidas.

Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$: Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl}. Assim, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$: Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik}. Isto é, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Operação Frente - $N^{OF}(s)$: Consiste em retirar de operação o equipamento de carga que esteja em operação na frente i. O movimento retira todas as viagens feitas a esta frente, deixando o equipamento *inativo*. O equipamento retorna à operação assim que uma nova viagem é associada a ele.

Movimento Operação Caminhão - $N^{OC}(s)$: Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, isto é, retirar de atividade um caminhão l que esteja operando em uma frente i.

Movimento Troca de Viagens - $N^{VT}(s)$: Duas células da matriz N são selecionadas e uma unidade de uma célula passa para a outra, isto é, uma viagem de um caminhão a uma frente passa para outro caminhão a outra frente.

Movimento Troca de Carregadeiras - $N^{CT}(s)$: Duas células distintas y_i e y_k da matriz Y tem seus valores permutados, ou seja, os equipamentos de carga que operam nas frentes i e k são trocados. Analogamente ao movimento CG, os equipamentos de carga são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

2.2.3 AVALIAÇÃO DE UMA SOLUÇÃO

Como os movimentos usados podem gerar soluções inviáveis, uma solução é avaliada por uma função f, a ser minimizada, composta por duas parcelas. A primeira delas é a função objetivo propriamente dita, f^{PM}, dada pela Eq. (1), e a segunda é composta pelas funções que penalizam a ocorrência de inviabilidade na solução corrente. Assim, a função f, dada pela Eq. (2), mensura o desvio dos objetivos considerados e penaliza o não atendimento às restrições do problema.

$$f(s) = f^{PM}(s) + f^{p}(s) + \sum_{j \in T} f_j^q(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s) \tag{2}$$

em que:
$f^{PM}(s)$ é uma função que avalia s quanto ao atendimento às metas de produção e qualidade, bem como número de caminhões utilizados (mesma do modelo de programação matemática, Subseção 2.1);
$f^{p}(s)$ avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril;
$f_j^q(s)$ avalia s quanto à inviabilidade em relação ao j-ésimo parâmetro de controle;
$f_l^u(s)$ avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l-ésimo caminhão;
$f_k^c(s)$, avalia s quanto ao desrespeito aos limites de produtividade da carregadeira k.

2.2.4 REPRESENTAÇÃO DE UM INDIVÍDUO

Um dado indivíduo possui, além da matriz de solução $R = [Y|N]$ (Subseção 2.2.1), dois vetores de parâmetros de mutação. O primeiro vetor diz a probabilidade de aplicação de cada um dos movimentos descritos na Subseção 2.2.2, logo, este é um vetor de números reais, onde cada posição i diz a probabilidade de aplicação de um dado movimento. Já o segundo vetor de parâmetros

que compõe a representação de um indíviduo é um vetor números inteiros e regula a intensidade da perturbação, ou seja, cada posição i deste vetor limita o número de aplicações de um dado movimento, caso este venha a ser aplicado.

Desta forma, tem-se um vetor de probabilidades $P = [p_1, p_2, ..., p_i]$, com $p_i \in [0, 1], p_i \in \mathbb{R}$. Já para o vetor de aplicações A, tem-se $A = [a_1, a_2, ..., a_i]$, sendo $a_i \in [0, nap_i], a_i \in \mathbb{N}$, com nap_i representando o número máximo de aplicações para um dado movimento i.

um dado movimento i.

2.3 ALGORITMO PROPOSTO

O algoritmo proposto neste trabalho, denominado *GES*, consiste na combinação dos procedimentos heurísticos GRASP e Estratégia Evolutiva. Seu pseudocódigo está esquematizado no Algoritmo 1.

Algoritmo 1: *GES*

Entrada: γ, *IterMax*, Função $f(.)$
Saída: População Pop

1 **para** $i \leftarrow 1$ **até** μ **faça**
2 $s_w \leftarrow$ ConstróiSoluçãoEstéril()
3 Gere um número aleatório $\gamma \in [0, 1]$
4 $s_i \leftarrow$ ConstróiSoluçãoMinério(s_w, γ)
5 $ind_i \leftarrow s_i$ + ConstróiVetorMutação()
6 $Pop[i] = ind_i$
7 **fim**
8 **enquanto** *critério de parada não satisfeito* **faça**
9 **para** $i \leftarrow 1$ **até** λ **faça**
10 Gere um número aleatório $j \in [1, \mu]$
11 $ind_i \leftarrow Pop[j]$
12 $ind_i \leftarrow$ MutaParametros($ind_i, \sigma_{real}, \sigma_{binomial}$)
13 $ind_i \leftarrow$ AplicaMutação(ind_i)
14 $PopFilhos$[i] $= ind_i$
15 **fim**
16 **para** $i \leftarrow 1$ **até** κ **faça**
17 Gere um número aleatório $i \in [1, \lambda]$
18 VND($PopFilhos[i]$)
19 **fim**
20 Pop = Seleção(Pop,$PopFilhos$)
21 **fim**
22 **retorna** *Pop*

Algoritmo 2: MutaParametros

Entrada: Indivíduo s, Desvios Padrões σ_{real} e $\sigma_{binomial}$
Saída: Indivíduo s

1 Vetor $p \leftarrow$ Vetor de Parâmetros P de Probabilidade do indivíduo s
2 Vetor $a \leftarrow$ Vetor de Parâmetros A de Aplicação do indivíduo s
3 **para** $i \leftarrow 1$ **até** 8 **faça**
4 Posição i do Vetor de Parâmetro Probabilidades
$p_i \leftarrow p_i + N(0, \sigma_{real})$
5 Posição i do Vetor de Parâmetro Aplicação
$a_i \leftarrow a_i + B(0, \sigma_{binomial})$
6 **fim**
7 **retorna** s

A população inicial do algoritmo (linhas 1 a 7 do Algoritmo 1) é composta por μ indivíduos e é criada em duas etapas.

Na primeira, realiza-se a construção da matriz de solução $R = [Y|N]$ de cada indivíduo da população. Para esta etapa são utilizados os procedimentos *ConstróiSoluçãoEstéril* e *ConstróiSoluçãoMinério* descritos em [11]. A obtenção de uma população inicial diversificada é de extrema importância para a convergência do algoritmo; logo, o parâmetro γ define o tamanho da lista restrita de candidatos varia em cada um dos indivíduos.

Na segunda etapa (linha 5 do Algoritmo 1), é feita a construção dos vetores de mutação de cada um dos indivíduos. Para o vetor P de probabilidades utiliza-se uma Distribuição Normal. Já para o vetor de aplicações A utiliza-se uma Distribuição Binomial.

Na linha 12 do Algoritmo 1 é acionado o procedimento de mutação dos parâmetros para um dado indivíduo, sendo o pseudocódigo desse procedimento descrito no Algoritmo 2.

Algoritmo 3: AplicaMutação

Entrada: Indivíduo s
Entrada: r vizinhanças: N^{NV}, N^{CT}, N^{VF}, N^{VC}, N^{VT}, N^{OF}, N^{OC} e N^{CG}
Saída: Indivíduo s

1 Vetor $p \leftarrow$ Vetor de Parâmetros P de Probabilidade do indivíduo s
2 Vetor $a \leftarrow$ Vetor de Parâmetros A de Aplicação do indivíduo s
3 **para** $i \leftarrow 1$ **até** 8 **faça**
4 Gere um número aleatório $z \in [0, 1]$
5 **se** $z < p_i$ **então**
6 **para** $j \leftarrow 1$ **até** a_i **faça**
7 $s \leftarrow Movimento_r(s)$
8 **fim**
9 **fim**
10 **fim**
11 **retorna** s

Algoritmo 4: VND

Entrada: Indivíduo s e Funcão de Avaliação f
Entrada: r vizinhanças na ordem aleatória: N^{VF}, N^{VC}, N^{NV} e N^{CG}
Saída: Indivíduo s^* de qualidade possivelmente superior à s de acordo com a função f

1 $k \leftarrow 1$
2 **enquanto** $k \leq r$ **faça**
3 Encontre o melhor vizinho $s' \in N^{(k)}(s)$
4 **se** $f(s') < f(s)$ **então**
5 $s \leftarrow s'$; $k \leftarrow 1$
6 **fim**
7 **senão**
8 $k \leftarrow k + 1$
9 **fim**
10 **fim**
11 **retorna** s

Para cada posição i dos vetores de parâmetros do Algoritmo 2 aplica-se uma Distribuição Normal ou Binomial, ambas centradas com média zero e desvio-padrão σ_{real} e $\sigma_{binomial}$, respectivamente. Este procedimento pode ser visto na linha 4 e 5 do Algoritmo 2. Para a mutação do vetor de probabilidades P aplica-se uma perturbação seguindo uma Distruibuição Normal

com desvio σ_{real}. Já para a mutação do vetor aplicações A aplica-se uma perturbação seguindo uma Distribuição Binomial com desvio $\sigma_{binomial}$.

O procedimento AplicaMutação (linha 13 do Algoritmo 1) é descrito no Algoritmo 3.

Na linha 4 do Algoritmo 3 é gerado um número aleatório $z \in [0, 1]$ e, em seguida, é verificado se este número satisfaz a probabilidade p_i do vetor de probabilidades. Caso afirmativo, aplica-se a_i vezes um dos oito movimentos (seção 2.2.2) referente à posição i. A ordem da vizinhança neste vetor de parâmetros é escolhida aleatoriamente.

O procedimento VND de busca local (linha 18 do Algoritmo 1) é opcional, isto é, nem todas as versões desse algoritmo o usam. Quando este procedimento é acionado no algoritmo, realiza-se um busca local de acordo com o procedimento VND (descrito no Algoritmo 4) usando-se, apenas, um grupo restrito dos movimentos descritos na seção 2.2.2, no caso, apenas nas vizinhanças: N^{CG}, N^{NV}, N^{VC} e N^{VF}. A justificativa para essa restrição é que a busca local de nosso algoritmo é muito custosa computacionalmente. Estrategicamente, a busca local opera nessas vizinhanças em uma ordem aleatória, definida a cada chamada. Esse resultado foi alcançado após uma bateria de testes preliminares, a qual mostrou que não havia uma ordem de vizinhança que resultasse, sempre, na geração de soluções melhores. Na seção 3 o resultado da adição deste procedimento é mostrado.

O procedimento de Seleção (linha 20 do Algoritmo 1) pode ser qualquer estratégia de seleção desejada, desde que esta retorne uma população de tamanho μ. Foram utilizadas duas formas básicas de competição, ambas com a mesma notação de [15]. Na primeira delas, denotada por $(\mu + \lambda)$, ocorre uma competição entre pais e filhos. Nesta estratégia são selecionados os μ melhores indivíduos dentre pais e filhos. Já na segunda estratégia de seleção utilizada, denotada por (μ, λ), os indivíduos que sobrevivem para a próxima geração são os μ melhores filhos. É notório que utilizando a estratégia (μ, λ) como forma de seleção, a população que sobrevive para próxima geração sofre uma considerável pressão seletiva, porém, esta pressão torna-se ainda maior quando a estratégia $(\mu + \lambda)$ é utilizada.

3. EXPERIMENTOS COMPUTACIONAIS E ANÁLISES

O algoritmo *GES* proposto foi implementado em C++ usando o framework de otimização OptFrame[1], e compilado pelo g++ 4.13, utilizando a IDE Eclipse 3.1. Os experimentos foram testados em um microcomputador Pentium Core 2 Quad(Q6600), com 8 GB de RAM, no sistema operacional Ubuntu 10.10. Para testá-lo, foi usado um conjunto de 8 problemas-teste da literatura, disponíveis em http://www.iceb.ufop.br/decom/prof/marcone/projects/mining.html. Estes problemas-teste foram os mesmos utilizados em [11] para validar o algoritmo *GGVNS*.

Os melhores resultados da literatura para os problemas-teste analisados são apresentados na Tabela 2. Na coluna "Opt." são indicados por "$\surd$" os problemas-teste nos quais o otimizador matemático CPLEX 11.02 obteve o valor ótimo da função.

Tabela 2: Melhores Valores

Problema-Teste	Melhor da Literatura	Opt[†]
opm1	227.12	
opm3	256.37	
opm2	164,027.15	$\surd$
opm4	164,056.68	$\surd$
opm5	227.04	
opm6	236.58	
opm7	164,017.46	$\surd$
opm8	164,018.65	$\surd$

Tabela 3: Tabela de Algoritmos

Sigla	μ	λ	Seleção	*VND*
GES1	30	160	(μ, λ)	
GES2	30	160	$(\mu + \lambda)$	
GES3	100	600	(μ, λ)	
GES4	100	600	$(\mu + \lambda)$	
GES-VND1	30	160	(μ, λ)	$\surd$
GES-VND2	30	160	$(\mu + \lambda)$	$\surd$

A Tabela 3 mostra seis variantes do algoritmo *GES*, criadas a partir de diferentes valores para seus parâmetros. As variantes *GES-VND1* e *GES-VND2* incluem o procedimento VND (descrito no Algoritmo 4) como método de busca local. Nestas duas variantes uma parcela de $\kappa = 3$ indivíduos da população de filhos sofre esse procedimento de busca local. O número de indivíduos que sofrem este procedimento de busca local é pequeno em relação à população de filhos visto que, como já citado anteriormente, a busca local utilizada é muito custosa computacionalmente.

Primeiramente, foram realizados dois experimentos de probabilidade empírica, *time-to-target (TTT) plots*, de forma a verificar a eficiência das seis variantes propostas. [18] descrevem um programa na linguagem *Perl* para criar *time-to-target plots*. Este tipo de gráfico tem sido amplamente utilizado como ferramenta de desenvolvimento de algoritmos, mostrando-se muito útil na comparação de diferentes algoritmos ou estratégias aplicadas a um determinado problema. Este teste mostra, no eixo das ordenadas, a probabilidade de um algoritmo em encontrar uma solução boa em um dado limite de tempo, registrado no eixo das abscissas. TTTplots já foi utilizado por vários autores, como [19], e continua sendo defendido [20] como uma forma de caracterizar tempo de execução de algoritmos estocásticos aplicados a problemas de otimização combinatória.

Para uma melhor comparação entre as variantes, suas curvas de probabilidade empírica foram sobrepostas. Foram realizados 120 execuções com cada uma das seis variantes desenvolvidas. Para as curvas da Figura 1, o problema-teste utilizado foi o opm1, tendo como alvo o valor 230,00 (2% do valor ótimo) e tempo limite de 1800 segundos. Já no segundo experimento, Figura 2, utilizou-se a problema-teste opm8, tendo como alvo o valor 164024,00 (0,0033 % do valor ótimo) e tempo limite de 1800 segundos.

[1]OptFrame website: http://sourceforge.net/projects/optframe/

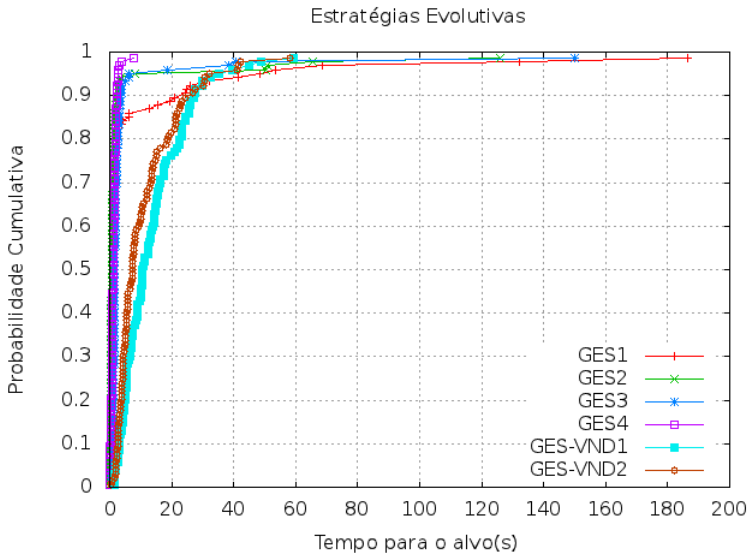

Figura 1: Curva de Probabilidade Empírica - Instância opm1

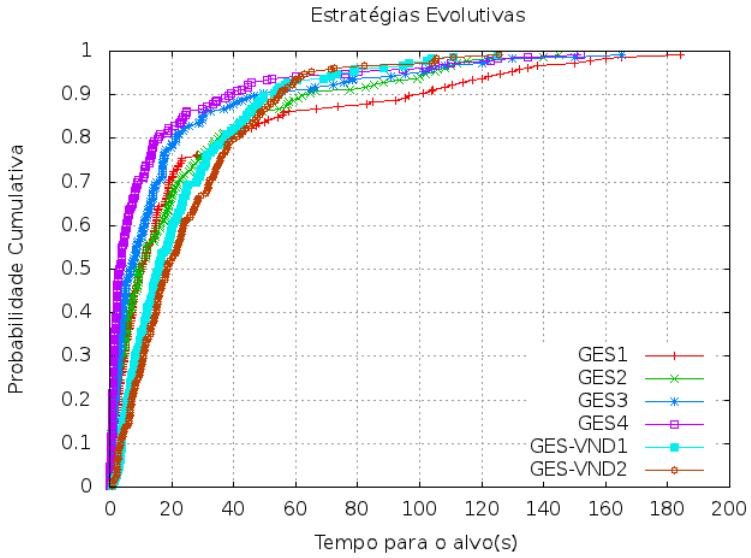

Figura 2: Curva de Probabilidade Empírica - Instância opm8

Analisando as curvas de probabilidade empírica (figuras 1 e 2), percebe-se que as variantes que utilizaram a estratégia de seleção $(\mu + \lambda)$ prevaleceram em relação às suas versões com estratégia (μ, λ). Este fato mostra que a competição entre pais e filhos fez com que indivíduos com um bom potencial de otimalidade permanecessem por mais gerações.

Desde os instantes iniciais da busca, a variante *GES4* foi capaz de gerar melhores soluções do que os outros algoritmos propostos. Na Figura 1 observa-se uma supremacia total da variante citada; no entanto, analisando as curvas da Figura 2, percebe-se que a partir de 60 segundos esta variante perde seu desempenho, sendo superada pela variante *GES-VND2*, a qual continua progredindo sistematicamente, sendo a primeira a alcançar o alvo desejado com uma probabilidade de aproximadamente 100%.

Dados dois algoritmos de busca estocástica A1 e A2 aplicados a um mesmo problema, denotamos por X1 e X2, respectivamente, a variável contínua que representa o tempo necessário ao algoritmo A1 (resp. A2) para encontrar a solução alvo de um dado problema-teste. Para lidar com a situação ilustrada na Figura 2, [21] desenvolveram uma ferramenta numérica para calcular a probabilidade PR(X1 $<=$ X2), ou seja, a probabilidade de o tempo de execução do algoritmo A1 ser menor ou igual à do A2. Utilizando esta ferramenta para as variantes da Figura 2 foi gerada a Tabela 4.

Analisando-se a Tabela 4, verifica-se que, apesar de a variante *GES-VND2* possuir uma maior probabilidade de alcançar o alvo em relação à variante *GES4* a partir dos 60 segundos, a variante *GES4* possui uma probabilidade 78,68% de ter o tempo de execução menor ou igual à variante *GES-VND2*. Além disso, nota-se que a variante *GES4* supera todas as outras variantes.

(μ, λ), a qual obteve o segundo melhor desempenho.

Desta forma, tendo em vista a robustez da variante *GES4*, foi feita uma comparação entre os algoritmos *GES4* e o algoritmo

Tabela 4: Estimativa de Convergência dos Algoritmos

PR($i <= j$)	*GES1*	*GES2*	*GES3*	*GES4*	*GES-VND1*	*GES-VND2*
GES1	-	48,23%	42,74%	32,17%	66,34%	63,13%
GES2	50,16%	-	43,77%	34,19%	65,58%	62,65%
GES3	56,05%	53,94%	-	**37,68%**	72,52%	69,91%
GES4	**65,68%**	**61,72%**	**59,25%**	-	**80,28%**	**78,68%**
GES-VND1	33,52%	34,15%	27,28%	19,36%	-	45,85%
GES-VND2	36,83%	37,27%	30,02%	21,20%	54,15%	-

GGVNS, de [11]. A Tabela 5 mostra a comparação entre os algoritmos *GES4* e *GGVNS*. Nessa tabela, a coluna "Instância" indica o problema-teste utilizado. A coluna "IMPdesv" menciona o ganho relativo do algoritmo *GES4* em relação ao algoritmo *GGVNS*, ou seja:

$$IMPdesv_i = \frac{\bar{f}_i^{GGVNS} - \bar{f}_i^{GES}}{\bar{f}_i^{GGVNS}} \qquad (3) \qquad\qquad IMPbest_i = \frac{f_i^{\star GGVNS} - f_i^{\star GES}}{f_i^{\star GGVNS}} \qquad (4)$$

A coluna "IMPbest" indica o percentual de melhora proporcionado pelo algoritmo *GES4* em relação ao valor da melhor solução encontrada pelo algoritmo *GGVNS*.

Tabela 5: Comparação de resultados: *GES4* × *GGVNS*

Instância	Tempo	*GGVNS*		*GES4*			
	(min)	Média	Melhor	Média	Melhor	IMPbest (%)	IMPdesv (%)
opm1	2	230,12	230,12	228,50	228,12	**0,87**	**0,70**
opm2	2	256,56	256,37	256,43	256,37	0,00	**0,05**
opm3	2	164064,68	164039,12	164044,68	164031,28	0,00	**0,01**
opm4	2	164153,92	164099,66	164097,61	164057,04	**0,03**	**0,03**
opm5	2	228,09	228,09	227,21	226,66	**0,63**	**0,39**
opm6	2	237,97	236,58	237,07	236,58	0,00	**0,38**
opm7	2	164021,89	164021,38	164020,24	164018,22	0,00	0,00
opm8	2	164027,29	164023,73	164022,38	164020,26	0,00	0,00

Analisando a Tabela 5 pode-se verificar que o algoritmo *GES4* proposto foi capaz de gerar soluções de boa qualidade e baixa variabilidade em relação ao algoritmo *GGVNS*. Percebe-se que ele conseguiu melhorar a qualidade das soluções finais em até 0,87%, e reduzir a variabilidade dessas soluções em até 0,70%. Além disso, tendo em vista a Tabela 2, pode ser observado que para o problema-teste opm5 o *GES4* foi capaz de encontrar uma solução melhor que a da literatura.

4. CONCLUSÕES

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões (POLAD). Em virtude de sua dificuldade de solução, foi proposto um algoritmo populacional, denominado *GES*, que combina o poderio do GRASP com as Estratégias Evolutivas, percorrendo um espaço de busca de soluções inteiras.

Seis variantes desse algoritmo foram desenvolvidas e comparadas entre si. Dessas, quatro eram algoritmos baseados em Estratégias Evolutivas tendo como principal mecanismo de busca no espaço a mutação e diferiam entre si pelos parâmetros de tamanho da população e estratégia de seleção. As outras duas incluíam um procedimento de busca local, baseado em VND, na exploração do espaço de soluções. Porém, devido ao desempenho das variantes sem o procedimento de busca local, optou-se pela variante *GES4*, a qual mostrou ser a mais eficiente em alcançar o valor alvo.

Usando problemas-teste da literatura, a variante *GES4* do algoritmo evolutivo foi comparada com um algoritmo da literatura, denominado GGVNS. Os resultados mostraram que o algoritmo evolutivo proposto é superior, apresentando boas soluções com uma menor variabilidade em torno da média.

Para trabalhos futuros propõe-se desenvolver novas estratégias de perturbação ao vetor de parâmetros de cada indivíduo, assim como variar a etapa de seleção durante a execução do algoritmo, de forma que o algoritmo entre em maior harmonia no quesito *Exploration-Exploitation*. Além disso, propõe-se uma melhoria no mecanismo de auto-adaptação, bem como uma melhor calibragem dos parâmetros das distribuições utilizadas. A adição do módulo de busca local apenas em determinadas gerações é outra estratégia que pode ser testada. Finalmente, propõe-se a implementação de uma versão paralela do algoritmo GES visando tirar proveito da tecnologia *multi-core*, já presente nas máquinas atuais e de fácil abstração para algoritmos populacionais.

5. AGRADECIMENTOS

Os autores agradecem às agências CNPq (processos 482765/2010-0 e 306458/2010-1) e FAPEMIG (processos CEX 00357/09 e CEX 01201/09) pelo suporte financeiro ao desenvolvimento desta pesquisa. Agradecem, também, à pesquisadora Isabel Rosseti, da Universidade Federal Fluminense, pela cessão dos códigos em *perl* dos aplicativos de comparação entre algoritmos, que permitiram a comparação das variantes desenvolvidas neste trabalho.

REFERÊNCIAS

[1] F. P. Costa. "Aplicações de técnicas de otimização a problemas de planejamento operacional de lavra em minas a céu aberto". Dissertação, Programa de Pós-Graduação em Engenharia Mineral, Escola de Minas, UFOP, Ouro Preto, 2005.

[2] T. A. Feo and M. G. C. Resende. "Greedy randomized adaptive search procedures". *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.

[3] M. G. C. Resende and C. C. Ribeiro. "Greedy randomized adaptive search procedures". In *Handbook of Metaheuristics*, edited by F. Glover and G. Kochenberger, pp. 219–242. Kluwer Academic Publishers, Boston, 2003.

[4] M. G. C. Resende and C. C. Ribeiro. "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications". In *Handbook of Metaheuristics*, edited by M. Gendreau and J. Potvin, pp. 283–319. Springer, New York, second edition, 2010.

[5] N. Mladenović and P. Hansen. "Variable Neighborhood Search". *Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[6] P. Hansen and N. Mladenović. "Variable neighborhood search: Principles and applications". *European Journal of Operational Research*, vol. 130, pp. 449–467, 2001.

[7] F. P. Costa, M. J. F. Souza and L. R. Pinto. "Um modelo de alocação dinâmica de caminhões". *Revista Brasil Mineral*, vol. 231, pp. 26–31, 2004.

[8] I. F. Guimarães, G. Pantuza and M. J. F. Souza. "Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto". In *Anais do XIV Simpósio de Engenharia de Produção (SIMPEP)*, p. 11, Bauru, CD-ROM, 2007.

[9] I. M. Coelho, S. Ribas and M. J. F. Souza. "Um algoritmo baseado em GRASP, VND e Iterated Local Search para a resolução do Planejamento Operacional de Lavra". In *XV Simpósio de Engenharia de Produção*, Bauru/SP, 2008.

[10] H. R. Lourenço, O. C. Martin and T. Stützle. "Iterated Local Search". In *Handbook of Metaheuristics*, edited by F. Glover and G. Kochenberger. Kluwer Academic Publishers, Boston, 2003.

[11] M. J. F. Souza, I. M. Coelho, S. Ribas, H. G. Santos and L. H. C. Merschmann. "A hybrid heuristic algorithm for the open-pit-mining operational planning problem". *European Journal of Operational Research*, vol. 207, no. 2, pp. 1041–1051, 2010.

[12] P. Hansen, N. Mladenovic and J. A. M. Pérez. "Variable neighborhood search: methods and applications". *4OR: Quarterly journal of the Belgian, French and Italian operations research societies*, vol. 6, pp. 319–360, 2008.

[13] K. D. Jong, D. David, B. Fogel and H. Schwefel. "A history of evolutionary computation". In *Handbook of Evolutionary Computation*, edited by F. Glover and G. Kochenberger, pp. 1–12. Oxford University Press and and Institute of Physics Publishing, New York/Bristol, third edition, 1997.

[14] A. Freitas and F. Guimarães. "Originality and Diversity in the Artificial Evolution of Melodies". In *Proceedings of the 13th annual Conference on Genetic and Evolutionary Computation*, New York, 2011.

[15] H. G. Beyer and H. P. Schwefel. "Evolution strategies - A comprehensive introduction". *Natural Computing*, vol. 1, pp. 3–52, 2002.

[16] S. Rajasekaran. "Optimal mix for high performance concrete by evolution strategies combined with neural networks". *Indian Journal of Engineering and Materials Sciences*, vol. 13, no. 11, pp. 7–17, 2006.

[17] L. Costa and P. Oliveira. "Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems". *Computers & Chemical Engineering*, vol. 25, no. 10, pp. 257–266, 2001.

[18] R. Aiex, M. Resende and C. Ribeiro. "TTTplots: a perl program to create time-to-target plots". *Optimization Letters*, vol. 1, pp. 355–366, 2007.

[19] T. Feo, M. Resende and S. Smith. "A greedy randomized adaptive search procedure for maximum independent set". *Operations Research*, vol. 42, pp. 860–878, 1994.

[20] C. Ribeiro and M. Resende. "Path-relinking intensification methods for stochastic local search algorithms". *Journal of Heuristics*, pp. 1–22, 2011.

[21] C. Ribeiro and I. Rosseti. "Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms". In *Proceedings of the VIII Metaheuristics International Conference*, Hamburg, 2009.

UM ALGORITMO BASEADO EM ESTRATÉGIAS EVOLUTIVAS PARA O PROBLEMA DE PLANEJAMENTO OPERACIONAL DE LAVRA

Vitor Nazario Coelho (UFOP)
vncoelho@gmail.com
Marcone Jamilson Freitas Souza (UFOP)
marcone.freitas@oi.com.br
Igor Machado Coelho (UFF)
igor.machado@gmail.com
Frederico Gadelha Guimaraes (UFMG)
frederico.g.guimaraes@gmail.com
Bruno Nazario Coelho (UFOP)
brunonazario@yahoo.com.br

Este artigo apresenta um algoritmo evolutivo inspirado em Estratégias Evolutivas com um procedimento GRASP para gerar a população inicial. O algoritmo proposto é aplicado a um problema que requer decisões rápidas, o problema de Planejamentoo Operacional de Lavra com Alocação Dinâmica de Caminhões (POLAD). Para validar o algoritmo desenvolvido, seus resultados são comparados com os produzidos por um algoritmo da literatura, denominado GGVNS, que não contempla o conceito de população. Resultados computacionais mostram a efetividade do algoritmo proposto.

Palavras-chaves: Planejamento Operacional de Lavra, Estratégias Evolutivas, Metaheurísticas, Computação Evolutiva, GRASP

1. Introdução

Este trabalho tem seu foco no planejamento operacional de lavra com alocação dinâmica de caminhões (POLAD). Esse problema envolve a alocação de carregadeiras às frentes de lavra (que podem ser de minério ou estéril), assim como a determinação do número de viagens que cada caminhão deve fazer a cada frente de forma que sejam atendidas tanto a meta de produção quanto a da composição mineral requerida para o minério. O objetivo é encontrar um ritmo de lavra em cada frente que minimize os desvios das metas de produção e qualidade, assim como o número de caminhões necessários ao processo produtivo.

Considera-se o sistema de alocação dinâmica de caminhões, isto é, a cada viagem realizada o caminhão pode se direcionar a uma frente diferente. Esse sistema de alocação contribui para o aumento da produtividade da frota e, consequentemente, para a redução do número de caminhões necessários ao processo produtivo.

O POLAD é um problema da classe NP-difícil e, como tal, métodos exatos de solução têm aplicabilidade restrita. A abordagem mais comum é por meio de procedimentos heurísticos. Costa (2005) desenvolveu um algoritmo heurístico baseado em Greedy Randomized Adaptive Search Procedures - GRASP (FEO & RESENDE, 1995; RESENDE & RIBEIRO, 2008) e VNS (HANSEN & MLADENOVIC, 2001) para o POLAD usando seis tipos diferentes de movimentos para explorar o espaço de soluções. Foi feita uma comparação entre os resultados obtidos por esse algoritmo heurístico e os encontrados pelo otimizador LINGO, versão 7, aplicado a um modelo de programação matemática desenvolvida pelos autores, publicado em Costa et al. (2004). Mostrou-se que o algoritmo heurístico foi capaz de encontrar soluções de melhor qualidade mais rapidamente. Guimarães et al. (2007) apresentaram um modelo de simulação computacional para validar resultados obtidos pela aplicação de um modelo de programação matemática na determinação do ritmo de lavra em minas a céu aberto. Dessa maneira, foi possível validar os resultados da otimização, já que na modelagem de otimização não é possível tratar a variabilidade nos tempos de ciclo e a ocorrência de fila. Em Coelho et al. (2008), o POLAD é resolvido por um algoritmo heurístico, denominado GVILS, que combina os procedimentos heurísticos GRASP, Variable Neighborhood Descent – VND (MLADENOVIC & HANSEN, 1997) e ILS (LOURENÇO ET AL., 2003). O algoritmo GVILS faz uso de oito movimentos para explorar o espaço de soluções. Além dos desvios de produção e qualidade, procurou-se minimizar, também, o número de veículos. Usando quatro problemas-teste da literatura, o GVILS foi comparado com o otimizador CPLEX 9.1 aplicado a um modelo de programação matemática. Foram realizados testes envolvendo 15 minutos de processamento. Em dois dos problemas, o algoritmo proposto mostrou-se bastante superior; enquanto nos dois outros ele foi competitivo com o CPLEX, produzindo soluções médias com valores até 0,08% piores, na média. Souza et al. (2010) propuseram um algoritmo, denominado GGVNS, que combina as metaheurísticas General Variable Neighborhood Search (GVNS) e o procedimento Greedy Randomized Adaptative Search Procedure (GRASP). Do procedimento GRASP utilizou-se a fase de construção para produzir soluções viáveis e de boa qualidade rapidamente. O GVNS foi escolhido devido a sua simplicidade, eficiência e capacidade natural de sua busca local para lidar com diferentes vizinhanças. Os autores compararam os resultados gerados pelo GGVNS com aqueles alcançados pelo otimizador CPLEX 11.01, utilizando oito problemas-teste. Os experimentos computacionais mostraram que o algoritmo proposto era competitivo com o CPLEX e capaz de encontrar soluções próximas do ótimo (com um gap < 1%) na maioria das instâncias, demandando um pequeno tempo computacional.

Por outro lado, a literatura tem mostrado que algoritmos evolutivos têm resolvido com eficiência vários problemas combinatórios (DE JONG ET AL.,1997).

No presente trabalho, como fruto da iniciação científica, investiga-se uma classe desses algoritmos, as chamadas Estratégias Evolutivas, Evolution Strategies - ES, (Beyer & Schwefel, 2002). Essas técnicas têm sido usadas na resolução de problemas inteiros ou mistos. Rajasekaran (2006) desenvolveu uma estratégia evolutiva combinada com redes neurais para resolução do problema de consistência do Concreto de Alta Performance (HPC), o algoritmo proposto foi comparado com um Algoritmo Genético (AG) e com o outro baseado no procedimento Simulated Annealing (SA), obtendo, nos problemas-teste em questão, o melhor desempenho. Costa & Oliveira (2001) desenvolveram um algoritmo evolutivo baseado nos conceitos das ES para resolução de problemas não-lineares mistos, sendo que o algoritmo ES também foi comparado com um algoritmo GA clássico e com o procedimento SA, obtendo novamente o melhor desempenho nos problemas-teste analisados.

O algoritmo proposto, denominado GES, gera sua população inicial por meio de um procedimento parcialmente guloso diversificado, baseado na metaheurística GRASP. Para mutação dos indivíduos, foram utilizadas as vizinhanças propostas em Souza et al. (2010), sendo a mutação o principal operador de busca no espaço de soluções. Para intensificar a busca, a cada geração uma pequena parte da população sofre uma busca local baseada no procedimento VND. O algoritmo proposto foi comparado ao GGVNS daqueles autores e se mostrou superior com relação à capacidade de encontrar melhores soluções mais rapidamente.

O restante deste trabalho está organizado como segue. A Seção 2 detalha o algoritmo proposto para resolver o POLAD. A Seção 3 mostra os resultados dos experimentos computacionais e a Seção 4 conclui o trabalho.

2. Metodologia

2.1. Modelo Exato

A formulação de programação matemática usada neste trabalho é a mesma de Coelho et al. (2008), em que se considera a função de avaliação mono-objetivo dada pela Eq. (1):

$$\min\ f^{PM}(s) = \sum_{j \in T} \lambda_j^- d_j^- + \sum_{j \in T} \lambda_j^+ d_j^+ + \alpha^- P_m^- + \alpha^+ P_m^+ + \beta^- P_e^- + \beta^+ P_e^+ + \sum_{l \in V} \omega_l U_l \qquad (1)$$

Na Eq. (1) busca-se minimizar os desvios positivos (d_j^+) e negativos (d_j^-) das metas de cada parâmetro de controle j da mistura, bem como minimizar os desvios positivos e negativos das metas de produção de minério e estéril, representados pelas variáveis de decisão P_m^+, P_m^-, P_e^+ e P_e^-, respectivamente. Nessa função também considera-se a minimização do número de veículos utilizados, representado pela variável binária U_l, que vale 1 se o veículo l for utilizado e 0, caso contrário.

As constantes λ_j^-, λ_j^+, α^-, α^+, β^-, β^+, e ω_l são pesos que refletem a importância de cada componente da função objetivo.

2.2. Modelo Heurístico

2.2.1. Representação de uma Solução

Uma solução é representada por uma matriz R = [Y|N], sendo Y a matriz $|F| \times 1$ e N a matriz $|F| \times |V|$. Cada célula y_i da matriz $Y_{|F| \times 1}$ representa a carregadeira k alocada à frente i. O valor -1 significa que não existe carregadeira alocada. Se não houver viagens feitas a uma frente i, a carregadeira k associada a tal frente é considerada inativa e não é penalizada por produção abaixo da mínima para este equipamento de carga.

Na matriz $N_{|F| \times |V|}$, cada célula n_{il} representa o número de viagens do caminhão $l \in V$ a uma frente $i \in F$. O valor 0 (zero) significa que não há viagem para aquele caminhão. O valor -1 informa a incompatibilidade entre o caminhão e a carregadeira alocada àquela frente.

Na Tabela 1, tem-se um exemplo de uma possível solução para o POLAD, observa-se que na coluna CARGA, linha F_1, a dupla $\langle Car_1,1\rangle$, indicando que o equipamento de carga Car_1 está alocado à frente F_1 e em operação. Na coluna CARGA, linha F_3, a dupla $\langle Car_8,0\rangle$ indica que o equipamento de carga Car_8 está alocado à frente F_3, mas não está em operação. Observa-se, ainda, na coluna CARGA, linha F_2, o valor $\langle D,0\rangle$ informando que não existe equipamento de carga alocado à frente F_2 e que, portanto, esta frente está disponível. As demais colunas representam o número de viagens a serem realizadas por um caminhão a uma frente, considerando a compatibilidade entre o caminhão e o equipamento de carga alocado à frente. As células com os valores X indicam incompatibilidade entre um caminhão e o respectivo equipamento de carga.

	Carga	**Cam_1**	**Cam_2**	**...**	**Cam_V**
F_1	<Car_1,1>	8	X	...	X
F_2	<D, 0>	0	0	...	0
F_3	<Car_8, 0>	0	0	...	0
...	...	...	...	...	...
F_F	<Car_5,1>	0	9	...	3

Tabela 1 – Representação de uma solução

2.2.2. Estruturas de Vizinhança

Como forma de explorar o espaço de soluções, foram utilizados os oito movimentos a seguir, os quais possuem uma boa capacidade exploratória, como relatado em Souza et al. (2010).

Os oito movimentos são descritos abaixo:

Movimento Número de Viagens - $N^{NV}(s)$: Este movimento consiste em aumentar ou diminuir o número de viagens de um caminhão l em uma frente i onde esteja operando um equipamento de carga compatível. Desta maneira, neste movimento uma célula n_{il} da matriz N tem seu valor acrescido ou decrescido de uma unidade.

Movimento Carga - $N^{CG}(s)$: Consiste em trocar duas células distintas y_i e y_k da matriz Y, ou seja, trocar os equipamentos de carga que operam nas frentes i e k, caso as duas frentes possuam equipamentos de carga alocados. Havendo apenas uma frente com equipamento de carga, esse movimento consistirá em realocar o equipamento de carga à frente disponível. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas às frentes são realocadas junto com as frentes escolhidas.

Movimento Realocar Viagem de um Caminhão - $N^{VC}(s)$: Consiste em selecionar duas células n_{il} e n_{kl} da matriz N e repassar uma unidade de n_{il} para n_{kl}. Assim, um caminhão l deixa de realizar uma viagem em uma frente i para realizá-la em outra frente k. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Realocar Viagem de uma Frente - $N^{VF}(s)$: Duas células n_{il} e n_{ik} da matriz N são selecionadas e uma unidade de n_{il} é realocada para n_{ik}. Isto é, esse movimento consiste em realocar uma viagem de um caminhão l para um caminhão k que esteja operando na frente i. Restrições de compatibilidade entre equipamentos são respeitadas, havendo realocação de viagens apenas quando houver compatibilidade entre eles.

Movimento Operação Frente - $N^{OF}(s)$: Consiste em retirar de operação o equipamento de carga que esteja em operação na frente i. O movimento retira todas as viagens feitas a esta frente, deixando o equipamento inativo. O equipamento retorna à operação assim que uma nova viagem é associada a ele.

Movimento Operação Caminhão - $N^{OC}(s)$: Consiste em selecionar uma célula n_{il} da matriz N e zerar seu conteúdo, isto é, retirar de atividade um caminhão l que esteja operando em uma frente i.

Movimento Troca de Viagens - $N^{VT}(s)$: Duas células da matriz N são selecionadas e uma unidade de uma célula passa para a outra, isto é, uma viagem de um caminhão associado a uma frente i passa para outro caminhão associado outra frente.

Movimento Troca de Carregadeiras - $N^{CT}(s)$: Duas células distintas y_i e y_k da matriz Y tem seus valores permutados, ou seja, os equipamentos de carga que operam nas frentes i e k são trocados. Analogamente ao movimento CG, os equipamentos de carga são trocados, mas as viagens feitas às frentes não são alteradas. Para manter a compatibilidade entre carregadeiras e caminhões, as viagens feitas a frentes com equipamentos de carga incompatíveis são removidas.

2.2.3 Avaliação de uma Solução

Como os movimentos usados podem gerar soluções inviáveis, uma solução é avaliada por uma função f, a ser minimizada, composta por duas parcelas. A primeira delas é a função objetivo propriamente dita, f^{PM} , dada pela Eq. (1), e a segunda é composta pelas funções que penalizam a ocorrência de inviabilidade na solução corrente. Assim, a função f mensura o desvio dos objetivos considerados e penaliza o não atendimento às restrições do problema. Ela está definida pela Eq. (2).

$$f(s) = f^{PM}(s) + f^{p}(s) + \sum_{j \in T} f_j^{q}(s) + \sum_{l \in V} f_l^{u}(s) + \sum_{k \in C} f_k^{c}(s) \qquad (2)$$

em que:

- $f^{PM}(s)$ é uma função que avalia s quanto ao atendimento às metas de produção e qualidade, bem como número de caminhões utilizados (mesma do modelo de programação matemática, Subseção 2.1);
- $f^{p}(s)$ avalia s quanto ao desrespeito aos limites de produção estabelecidos para a quantidade de minério e estéril;
- $f_q^{j}(s)$ avalia s quanto à inviabilidade em relação ao j-ésimo parâmetro de controle;
- $f_u^{l}(s)$ avalia s quanto ao desrespeito do atendimento da taxa de utilização máxima do l-ésimo caminhão;
- $f_c^{k}(s)$, que avalia s quanto ao desrespeito aos limites de produtividade da carregadeira k.

2.2.4 Representação de um Indivíduo

Um dado indivíduo possui, além da matriz de solução R = [Y | N] (Subseção 2.2.1), dois vetores de parâmetros de mutação.

O primeiro vetor diz a probabilidade de aplicação de cada um dos movimentos descritos na Subseção 2.2.2, logo, este é um vetor de números reais, onde cada posição i diz a probabilidade de aplicação de um dado movimento. Já o segundo vetor de parâmetros que compõe a representação de um indivíduo é um vetor números inteiros e regula a intensidade da perturbação, ou seja, cada posição i deste vetor limita o número de aplicações de um dado movimento, caso este venha a ser aplicado.

Desta forma, tem-se um vetor de probabilidades P, tal que:

$$P = [p_1, p_2, ..., p_i] \quad (3)$$

sendo $p_i \in [0,1]$, $p_i \in \Re$.

Já para o vetor de aplicações, tem-se:

$$A = [a_1, a_2, ..., a_i] \quad (4)$$

sendo $a_i \in [0, nap_i], a_i \in Z^+$, com nap_i representando o número máximo de aplicações para um dado movimento i.

2.3 Algoritmo Proposto

O algoritmo proposto neste trabalho, denominado GES, consiste na combinação dos procedimentos heurísticos GRASP e segue os passos de uma Estratégia Evolutiva. Seu pseudocódigo está esquematizado na Figura 1.

```
   Entrada: γ, IterMax, Função f(.)
   Saída: População Pop
 1 para i ← 1 até μ faça
 2     s_w ← ConstróiSoluçãoEstéril()
 3     Gere um número aleatório γ ∈ [0, 1]
 4     s_i ← ConstróiSoluçãoMinério(s_w, γ)
 5     ind_i ← s_i + ConstróiVetorMutação()
 6     Pop[i] = ind_i
 7 fim
 8 enquanto critério de parada não satisfeito faça
 9     para i ← 1 até λ faça
10         Gere um número aleatório j ∈ [1, μ]
11         ind_i ← Pop[j]
12         ind_i ← MutaParametros(ind_i, σ_real, σ_binomial)
13         ind_i ← AplicaMutação(ind_i)
14         PopFilhos[i] = ind_i
15     fim
16     para i ← 1 até κ faça
17         Gere um número aleatório i ∈ [1, λ]
18         VND(PopFilhos[i])
19     fim
20     Pop = Seleção(Pop,PopFilhos)
21 fim
22 retorna Pop
```

Figura 1 – GES

A população inicial do algoritmo (linhas 1 a 7 da Figura 1) é composta por μ indivíduos e é criada em duas etapas.

Na primeira, realiza-se a construção da matriz de solução R = [Y | N] de cada indivíduo da população. Para esta etapa são utilizados os procedimentos ConstróiSoluçãoEstéril e ConstróiSoluçãoMinério descritos em Souza et al. (2010). A obtenção de uma população inicial diversificada é de extrema importância para a convergência do algoritmo; logo, o parâmetro γ que define o tamanho da lista restrita de candidatos varia em cada um dos indivíduos.

Na segunda etapa (linha 5 da Figura 1), é feita a construção dos vetores de mutação de cada um dos indivíduos. Para o vetor P de probabilidades utiliza-se uma Distribuição Normal. Já para o vetor de aplicações A utiliza-se uma Distribuição Binomial.

Na linha 12 da Figura 1 é acionado o procedimento de mutação dos parâmetros para um dado indivíduo, cujo pseudocódigo está descrito na Figura 2.

Entrada: Indivíduo s, Desvios Padrões σ_{real} e $\sigma_{binomial}$
Saída: Indivíduo s

1 Vetor $p \leftarrow$ Vetor de Parâmetros P de Probabilidade do indivíduo s
2 Vetor $a \leftarrow$ Vetor de Parâmetros A de Aplicação do indivíduo s
3 **para** $i \leftarrow 1$ **até** 8 **faça**
4 | Posição i do Vetor de Parâmetro Probabilidades $p_i \leftarrow p_i + N(0, \sigma_{real})$
5 | Posição i do Vetor de Parâmetro Aplicação $a_i \leftarrow a_i + B(0, \sigma_{binomial})$
6 **fim**
7 **retorna** s

Figura 2 – MutaParâmetros

Para cada posição i dos vetores de parâmetros da Figura 2 aplica-se uma Distribuição Normal ou Binomial, ambas centradas com média zero e desvio-padrão σ_{real} e σ_{binomial}, respectivamente. Este procedimento pode ser visto na linha 4 e 5 desta figura. Para a mutação do vetor de probabilidades P aplica-se uma perturbação seguindo uma Distribuição Normal com desvio σ_{real}; já para a mutação do vetor aplicações A aplica-se uma perturbação seguindo uma Distribuição Binomial com desvio σ_{binomial}.

Já o procedimento AplicaMutação (linha 13 da Figura 1) está exemplificado na Figura 3.

Entrada: Indivíduo s
Entrada: r vizinhanças: $N^{NV}, N^{CT}, N^{VF}, N^{VC}, N^{VT}, N^{OF}, N^{OC}$ e N^{CG}
Saída: Indivíduo s

1 Vetor $p \leftarrow$ Vetor de Parâmetros P de Probabilidade do indivíduo s
2 Vetor $a \leftarrow$ Vetor de Parâmetros A de Aplicação do indivíduo s
3 **para** $i \leftarrow 1$ **até** 8 **faça**
4 | rand $z \in [0, 1]$
5 | **se** $z < p_i$ **então**
6 | | **para** $j \leftarrow 1$ **até** a_i **faça**
7 | | | $s \leftarrow Movimento_r(s)$
8 | | **fim**
9 | **fim**
10 **fim**
11 **retorna** s

Figura 3 – AplicaMutação

Na linha 4 da Figura 3 é gerado um número aleatório $z \in [0,1]$ e, em seguida, é verificado se este número satisfaz a probabilidade p_i do vetor de probabilidades. Caso afirmativo, aplica-se a_i vezes um dos oito movimentos (seção 2.2.2) referentes à posição i. A ordem da vizinhança neste vetor de parâmetros é escolhida aleatoriamente.

O procedimento VND de busca local (linha 18 da Figura 1) é opcional. Quando este procedimento é acionado no algoritmo, realiza-se um busca local de acordo com o procedimento VND (descrito pela Figura 4) usando-se, apenas, um grupo restrito dos movimentos descritos na seção 2.2.2, no caso, apenas nas vizinhanças: N^{CG}, N^{NV} , N^{VC} e N^{VF}. A justificativa para essa restrição é que a busca local do algoritmo é muito custosa computacionalmente. Estrategicamente, a busca local opera nessas vizinhanças em uma ordem aleatória, definida a cada chamada. Esse resultado foi alcançado após uma bateria de testes preliminares, a qual mostrou que não havia uma ordem de vizinhança que resultasse, sempre, na geração de soluções melhores. Na seção 3 o resultado da adição deste procedimento é mostrado.

Entrada: Indivíduo s e Funcão de Avaliação f
Entrada: r vizinhanças na ordem aleatória: N^{VF}, N^{VC}, N^{NV} e N^{CG}
Saída: Indivíduo s^* de qualidade possivelmente superior à s de acordo com a função f

```
1  k ← 1
2  enquanto k ≤ r faça
3  |  Encontre o melhor vizinho s' ∈ N^(k)(s)
4  |  se f(s') < f(s) então
5  |  |  s ← s'; k ← 1
6  |  fim
7  |  senão
8  |  |  k ← k + 1
9  |  fim
10 fim
11 retorna s
```

Figura 4 – VND

O procedimento de Seleção (linha 20 da Figura 1) pode ser qualquer estratégia de seleção desejada, desde que esta retorne uma população de tamanho μ. Foram utilizadas duas formas básicas de competição, ambas com a mesma notação de Beyer & Schwefel (2002). Na primeira delas, denotada por (μ + λ), ocorre uma competição entre pais e filhos. Nesta estratégia são selecionados os μ melhores indivíduos dentre pais e filhos. Já na segunda estratégia de seleção utilizada, denotada por (μ, λ), os indivíduos que sobrevivem para a próxima geração são os μ melhores filhos. É notório que utilizando a estratégia (μ, λ) como forma de seleção a população que sobrevive para próxima geração sofre uma considerável pressão seletiva; porém, esta pressão torna-se ainda maior quando a estratégia (μ + λ) é utilizada.

3. Experimentos Computacionais e Análises

O algoritmo GES proposto foi implementado em C++ usando o framework de otimização OptFrame, disponível em https://sourceforge.net/projects/optframe, e compilado pelo g++ 4.13, utilizando a IDE Eclipse 3.1. Os experimentos foram testados em um microcomputador Pentium Core 2 Quad(Q6600), com 8 GB de RAM, no sistema operacional Ubuntu 10.10. Para testá-lo, foi usado um conjunto de 8 problemas-teste da literatura, disponíveis em

http://www.iceb.ufop.br/decom/prof/marcone/projects/mining.html. Estes problemas-teste foram os mesmos utilizados em Souza et al. (2010) para validar o algoritmo GGVNS.

Os melhores resultados da literatura para os problemas-teste analisados são apresentados na Tabela 2. Na coluna "Opt." Indica-se por "√" os problemas-teste nos quais o otimizador matemático CPLEX 11.02 obteve o valor ótimo da função.

Problema-Teste	Melhor da Literatura	Opt
opm1	227.12	
opm2	256.37	
opm3	164,027.15	√
opm4	164,056.68	√
opm5	227.04	
opm6	236.58	
opm7	164,017.46	√
opm8	164,018.65	√

Tabela 2 – Melhores valores

A Tabela 3 mostra seis variantes do algoritmo GES, criadas a partir de diferentes valores para seus parâmetros. As variantes GES-VND1 e GES-VND2 incluem o procedimento VND (descrito na Figura 4) como método de busca local. Nestas duas variantes uma parcela de k=3 indivíduos da população de filhos sofre esse procedimento de busca local. O número de indivíduos que sofrem este procedimento de busca local é pequeno em relação à população de filhos visto que, como já citado anteriormente, a busca local utilizada é muito custosa computacionalmente.

Sigla	μ	λ	Seleção	VND
GES1	30	160	$(\mu; \lambda)$	
GES2	30	160	$(\mu + \lambda)$	
GES3	100	600	$(\mu; \lambda)$	
GES4	100	600	$(\mu + \lambda)$	
GES-VND1	30	160	$(\mu; \lambda)$	√
GES-VND2	30	160	$(\mu + \lambda)$	√

Tabela 3 – Variantes do Algoritmo GES

Primeiramente, foi realizado um experimento de probabilidade empírica, Time-to-target (TTT) plots, na forma indicada por Aiex et al. (2007), de forma a verificar a eficiência das variantes do algoritmo elencadas na Tabela 2. Este teste mostra, no eixo das ordenadas, a probabilidade de um algoritmo em encontrar uma solução boa em um dado limite de tempo, eixo das abscissas. TTT plots já foi utilizado por vários autores, como Hoos & Stutzle (1998), e continua sendo defendido (RIBEIRO, 2011) como uma forma de caracterizar tempo de execução de algoritmos estocásticos aplicados a problemas de otimização combinatória.

Foram realizadas 120 execuções com cada uma das seis variantes do algoritmo desenvolvido. O problema-teste utilizado foi o opm1, tendo os seguintes valores como alvo o valor 229,00 e tempo limite de 120 segundos. A Figura 5 mostra as curvas obtidas.

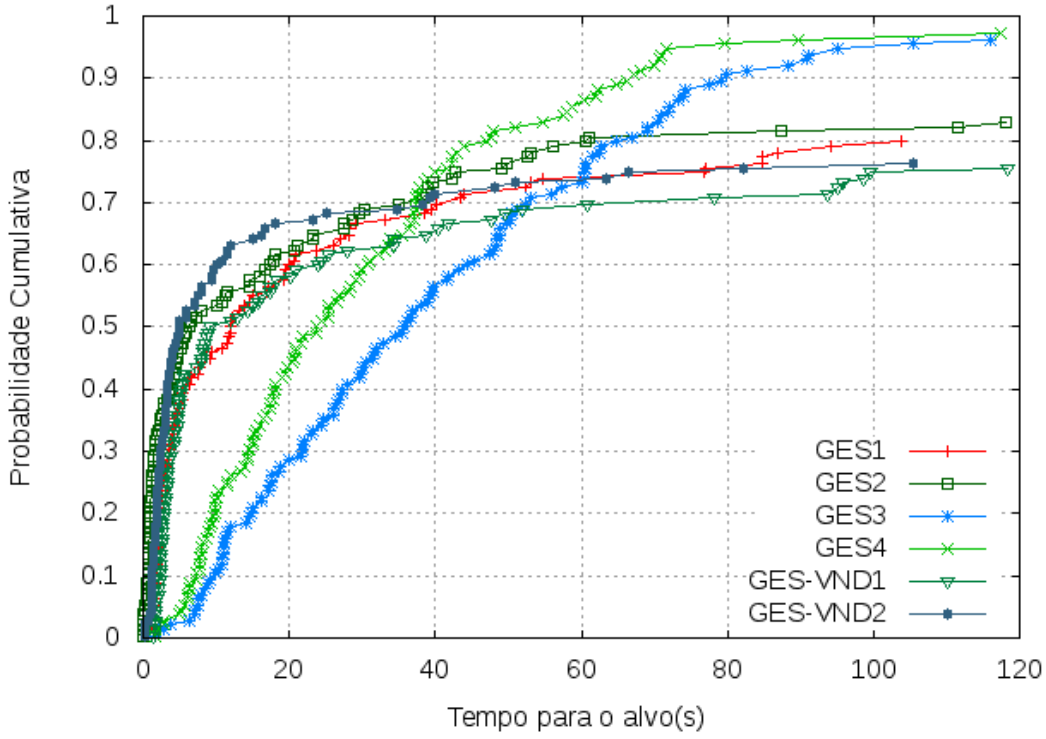

Figura 5 – Curva de Probabilidade Empírica - Instância opm1

Analisando as curvas de probabilidade empírica da Figura 5, percebe-se que as variantes que utilizaram a estratégia de seleção ($\mu + \lambda$) prevaleceram em relação a suas versões com estratégia (μ, λ). Este fato mostra que a competição entre pais e filhos fez com que indivíduos com um bom potencial de otimalidade permanecessem por mais gerações.

Nos instantes iniciais da busca, a variante GES-VND2 foi capaz de gerar melhores soluções do que os outros algoritmos propostos. No entanto, a partir de 40 segundos esta variante perde seu desempenho, sendo superada pela variante GES4, a qual continua progredindo sistematicamente, sendo a primeira a alcançar o alvo desejado com uma probabilidade de aproximadamente 100%.

Pela Figura 5, verifica-se que as variantes que utilizam o método de busca local VND, GES-VND1 e GES-VND2, têm ótimo desempenho no início da busca, mas convergem prematuramente. Isto é devido ao fato de que o VND contribui para gerar super-indivíduos que causam estagnação da população depois de alguns instantes de execução do algoritmo. Por outro lado, pode-se comprovar a força do mecanismo de mutação das Estratégias Evolutivas quando é analisada a convergência das variantes que não usam o procedimento de busca local.

Desta forma, tendo em vista a robustez da variante GES4, foi feita uma comparação entre os algoritmos GES4 e o algoritmo GGVNS, de Souza et al., (2010).

A Tabela 3 mostra a comparação entre algoritmo GES4 e o algoritmo GGVNS. Nessa tabela, a coluna "Instância" indica o problema-teste utilizado. A coluna "IMPdesv" menciona o ganho relativo do algoritmo GES4 em relação ao algoritmo GGVNS, ou seja:

$$\text{IMPdesv}_i = \frac{\bar{f}_i^{\,\text{GGVNS}} - \bar{f}_i^{\,\text{GES4}}}{\bar{f}_i^{\,\text{GGVNS}}} \qquad (5)$$

Já a coluna "IMPbest" indica o percentual de melhora proporcionado pelo algoritmo GES4 em relação ao valor da melhor solução encontrada pelo algoritmo GGVNS.

$$IMPbest_i = \frac{f_i^{*GGVNS} - f_i^{*GES4}}{f_i^{*GGVNS}} \quad (6)$$

Instância	Tempo (s)	GGVNS		GES4		IMPbest	IMPgap
		MEDIA	MELHOR	MEDIA	MELHOR		
opm1	2	230,12	230,12	228,50	228,12	**0,87%**	**0,70%**
opm2	2	256,56	256,37	256,43	256,37	0,00%	**0,05%**
opm3	2	164064,68	164039,12	164044,68	164031,28	0,00%	**0,01%**
opm4	2	164153,92	164099,66	164097,61	164057,04	**0,03%**	**0,03%**
opm5	2	228,09	228,09	227,21	226,66	**0,63%**	**0,39%**
opm6	2	237,97	236,58	237,07	236,58	0,00%	**0,38%**
opm7	2	164021,89	164021,38	164020,24	164018,22	0,00%	0,00%
opm8	2	164027,29	164023,73	164022,38	164020,26	0,00%	0,00%

Tabela 4 – Comparação de resultados: GGVNS x GES4

Analisando a Tabela 5 podemos verificar que o algoritmo GES4 proposto foi capaz de gerar soluções de boa qualidade e baixa variabilidade em relação ao algoritmo GGVNS. Percebe-se que ele conseguiu melhorar a qualidade das soluções finais em até 0,87%, e reduzir a variabilidade dessas soluções em até 0,39%. Além disso, tendo em vista a Tabela 2, pode ser observado que para o problema-teste opm5 o GES4 foi capaz de encontrar uma solução melhor que a da literatura.

4. Conclusões

Este trabalho teve seu foco no problema de planejamento operacional de lavra considerando alocação dinâmica de caminhões (POLAD). Em virtude de sua dificuldade de solução, foi proposto um algoritmo populacional, denominado GES, que combina o poderio do GRASP com as Estratégias Evolutivas, percorrendo um espaço de busca de soluções inteiras.

Seis variantes desse algoritmo foram desenvolvidas e comparadas entre si. Dessas, quatro eram algoritmos baseados em Estratégia Evolutiva tendo como principal mecanismo de busca no espaço a mutação e diferiam entre si pelos parâmetros de tamanho da população e estratégia de seleção. As outras duas incluiam um procedimento de busca local, baseado em VND, na exploração do espaço de soluções. Verificou-se a convergência prematura das variantes que usam o procedimento de busca local e optou-se pela variante que mostrou ser mais eficiente em alcançar o valor alvo.

Usando problemas-teste da literatura, essa variante do algoritmo evolutivo, denotada por GES4, foi comparada com um algoritmo da literatura, denominado GGVNS. Os resultados mostraram que o algoritmo evolutivo proposto é competitivo, apresentando boas soluções com uma menor variabilidade em torno da média.

Para trabalhos futuros é proposto o uso de novas estratégias de perturbação ao vetor de parâmetros de cada indivíduo, assim como variar a etapa de seleção durante a execução do algoritmo, de forma que o algoritmo entre em maior harmonia no quesito Exploration-Explotation. Além disso, propõe-se uma melhoria no mecanismo de auto-adaptação, bem como uma melhor calibragem dos parâmetros das distribuições utilizadas. A adição do

módulo de busca local apenas em determinadas gerações é outra estratégia que pode ser testada. Finalmente, propõe-se a implementação de uma versão paralela do algoritmo GES visando tirar proveito da tecnologia multi-core, já presente nas máquinas atuais e de fácil abstração para algoritmos populacionais.

Agradecimentos

Os autores agradecem à FAPEMIG pela bolsa de iniciação científica e recursos recebidos, assim como ao CNPq pelo apoio à execução do presente trabalho de pesquisa.

Referências

Aiex R. M., Resende, M.G.C. & Ribeiro, C.C. TTTPLOTS: a perl program to create time-to-target plots. Optimization Letters, Vol. 1, n.4, p.355-366, 2007.

Beyer, H. G. & Schwefel, H. P. Evolution strategies - a comprehensive introduction. Natural Computing, Vol. p.3-52, 2002.

Coelho, I. M., Ribas, S., & Souza, M. J. F. Um algoritmo baseado em grasp, vnd e iterated local search para a resolução do planejamento operacional de lavra. In XV Simpósio de Engenharia de Produção, Bauru/SP, 2008.

Coelho, I. M., Ribas, S., Souza, M. J. F., & Coelho, V. N. Um algoritmo heurístico híbrido para o planejamento operacional de lavra. In Anais do IX Congresso Brasileiro de Redes Neurais (CBRN), p.1-7, Ouro Preto, MG, 2009.

Costa, F. P. Aplicações de técnicas de otimização a problemas de planejamento operacional de lavra em minas a céu aberto. Dissertação, Programa de Pós-Graduação em Engenharia Mineral, Escola de Minas, UFOP, Ouro Preto, 2005.

Costa, F. P., Souza, M. J. F., & Pinto, L. R. Um modelo de alocação dinâmica de caminhões. Revista Brasil Mineral, Vol. 231, p.26-31, 2004.

Costa, L. & Oliveira, P. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. Computers & Chemical Engineering, Vol. 25, n.10, p.257-266, 2001.

De Jong, K., David, D., Fogel, B. & Schwefel, H.P. A history of evolutionary computation.In: Bäck T, Fogel DB and Michalewicz Z (eds) Handbook of Evolutionary Computation, Vol. A2, n.3, p.1-12. Oxford University Press, New York, and Institute of Physics Publishing,Bristol, 1997.

Feo, T. A. & Resende, M. G. C. Greedy randomized adaptive search procedures. Journal of Global Optimization, Vol. 6, p.109-133, 1995.

Guimarães, I. F., Pantuza, G., & Souza, M. J. F. Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. In Anais do XIV Simpósio de Engenharia de Produção (SIMPEP), p.11, Bauru, CD-ROM, 2007.

Hansen, P. & Mladenovic, N. Variable neighborhood search: Principles and applications. European Journal of Operational Research, Vol. 130, p.449-467, 2001.

Hoos, H. H. & Stutzle, T. On the empirical evaluation of Las Vegas algorithms - Position paper. Technical report, Computer Science Department, University of British Columbia, 1998.

Lourenço, H. R., Martin, O. C., & Stützle, T. Iterated local search. In Glover, F. and Kochenberger, G., editors, Handbook of Metaheuristics. Kluwer Academic Publishers, Boston, 2003.

Mladenovic, N. & Hansen, P. A variable neighborhood search. Computers and Operations Research, Vol. 24, p.1097-1100, 1997.

Rajasekaran, S. Optimal mix for high performance concrete by evolution strategies combined with neural networks. Indian Journal of Engineering and Materials Sciences, Vol. 13, n.11, p.7-17, 2006.

Resende, M. G. C. & Ribeiro, C. C. Greedy randomized adaptive search procedures: Advances and applications. In Gendreau, M. and Potvin, J., editors, Handbook of Metaheuristics. Springer, 2 edition. (to appear). Available at: http://www2.research.att.com/ mgcr/doc/sgrasp2008a.pdf, 2008.

Ribeiro, C.C. & Resende, M. G. C. Path-relinking intensification methods for stochastic local search algorithms. Journal of Heuristics, Springer Netherlands, p.1-22, 2011.

Souza, M. J. F., Coelho, I. M., Ribas, S., Santos, H. G., & Merschmann, L. H. C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. European Journal of Operational Research, Vol. 207, n.2, p.1041-1051, 2010.