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/

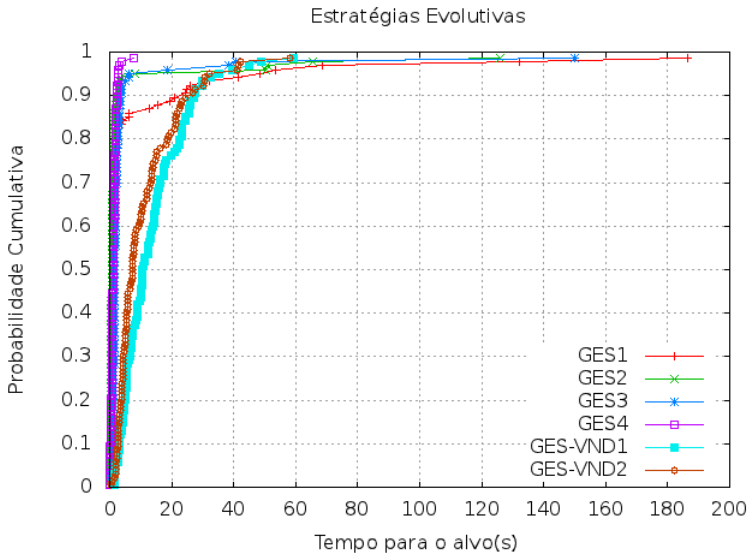

Figura 1: Curva de Probabilidade Empírica - Instância opm1

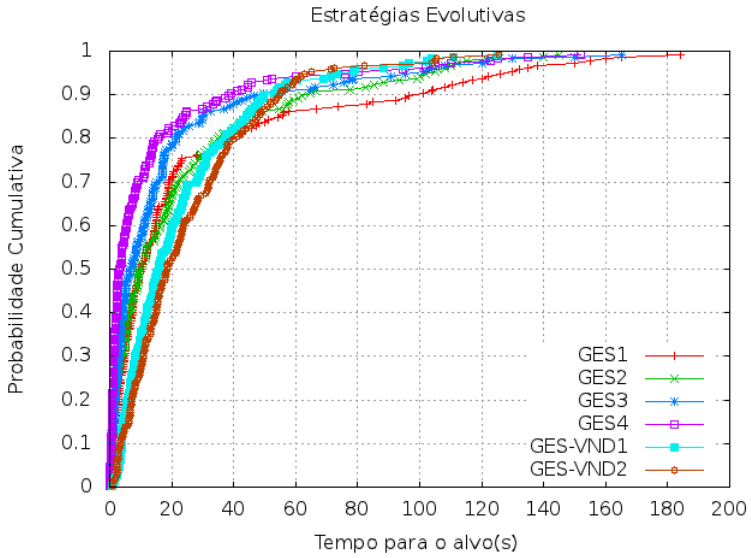

Figura 2: Curva de Probabilidade Empírica - Instância opm8

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.