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

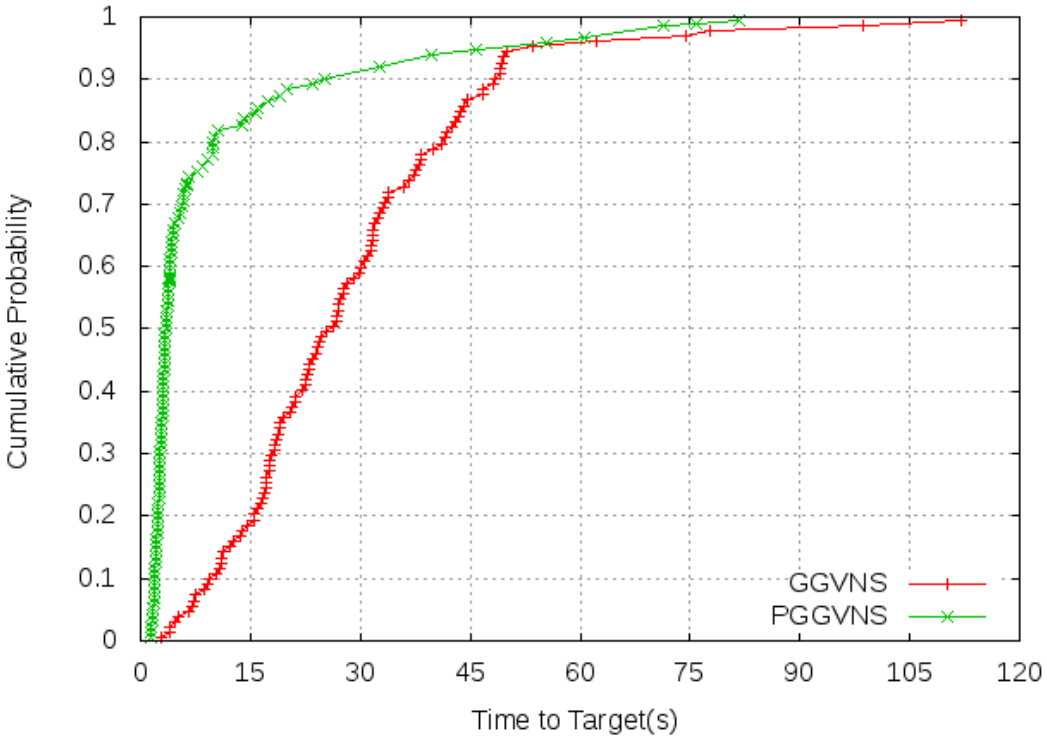

Figura 6 – Curva de probabilidade empírica *GGVNS* x *PGGVNS* – Instância opm1

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.