UMA ANÁLISE DA APLICAÇÃO DA METAHEURÍSTICA GRASP AO PROBLEMA DE CORTE COM DIMENSÃO ABERTA GUILHOTINADO

Dayanne Gouveia Coelho[1], **Marcelus Xavier Oliveira**[1], **Elizabeth Fialho Wanner**[1], **Sérgio Ricardo de Souza**[1]

[1]Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675, CEP 30510-000, Belo Horizonte (MG), Brasil

dayagc@gmail.com, marcelusxavier@gmail.com,

efwanner@gmail.com, sergio@dppg.cefetmg.br

Resumo. *Este artigo apresenta uma aplicação da metaheurística GRASP para a solução do Problema de Corte com Dimensão Aberta (PCDA) Guilhotinado. O problema consiste em alocar um conjunto de peças menores (itens) dentro de uma peça maior (objeto), de forma que a altura utilizada deste objeto seja minimizada. Para resolver o problema, incorpora-se ao GRASP o Algoritmo de Nível Best-Fit Decreasing Height que realiza o encaixe dos itens no objeto. Esta metodologia foi testada em um conjunto de problemas-testes da literatura. A metodologia proposta foi capaz de encontrar valores ótimos para as instâncias com até 49 itens, e para as instâncias de maior dimensão, ou foi encontrado o valor ótimo ou foi determinado, a um baixo custo computacional, um limitante superior do valor da função de avaliação melhor que aquele determinado pelo uso do pacote computacional CPLEX. Nota-se que a fase de busca local, do Algoritmo GRASP, aumenta o custo computacional do algoritmo, sem necessariamente garantir a melhora da solução encontrada durante a fase de construção.*

PALAVRAS-CHAVE: *Problema de Corte com Dimensão Aberta, GRASP, Algoritmo de Nível*

Abstract. *This paper presents an application of GRASP metaheuristic for solving the Guillotine Open Dimensional Problem. The problem consists of placing a set of small items within a large object such that the overall height must be minimized. For solving the problem, the level algorithm known as Best Fit Decreasing Height is coupled with GRASP to perform the fitting of the items onto the object. This methodology was tested on a set of test problems found in the literature. The proposed methodology was able to find optimal values for the instances with up to 49 items, and for larger instances, the optimal value was found or it was obtained, at a low computational cost, a upper bound for the value of the objective function better than the one determined by using the computational package CPLEX. It is worthwhile to notice that the local search phase of GRASP increases the overall computational cost of the algorithm, without necessarily guarantee the improvement of the solution found during the construction phase.*

KEYWORDS: *Open Dimensional Problem, GRASP, Level algorithms*

1 Introdução

Os Problemas de Corte (*Cutting Stock Problem*) têm sido objeto de estudo de muitos pesquisadores, desde os trabalhos iniciais de Gilmore e Gomory (1961) e Gilmore e Gomory (1963), na década de 60. Estes problemas são essenciais para o planejamento da produção, auxiliando as indústrias de couro, madeira, vidro, papelão e ferro, dentre outras, a minimizar o desperdício da matéria-prima. A matéria-prima é produzida no primeiro momento em tamanhos grandes e padronizados. Conforme a necessidade, para atender as demandas internas ou externas das indústrias, ela será reduzida a itens de tamanhos menores, variados, e geralmente não padronizados. Planejar como será o corte não é uma tarefa fácil, visto que esta operação gera perdas da matéria-prima. Porém, um bom planejamento evita a necessidade de constantes preparações das máquinas para os tamanhos dos produtos requisitados, o que irá minimizar os efeitos negativos gerados pelo desperdício sobre os custos de produção.

Neste trabalho é estudado um caso particular dos Problemas de Corte (PC), o Problema de Corte com Dimensão Aberta (PCDA) - *Open Dimensional Problem* (ODP), que consiste em determinar o melhor arranjo de um conjunto de itens retangulares sobre um objeto maior, que possui largura fixa e altura variável. Esta variação do problema pode ser encontrada principalmente em indústrias que realizam o corte de bobinas ou rolos, com o objetivo de minimizar a altura utilizada desses objetos.

Este problema pode ser encontrado na literatura sob outras denominações. Nos trabalhos de Hopper e Turton (2001b), Martelo et al. (2003) e Yeung e Tang (2004), o PCDA é tratado com o nome de *(Rectangular) Strip Packing Problem* ou *Two-Dimensional Strip Packing Problem*. Hopper e Turton (2001a) e Martelo et al. (2003) intitulam o PCDA como *Orthogonal Rectangular Strip Packing Problem*. Já Lodi et al. (2004) denomina o PCDA como *Level Packing Problem* e propõem um modelo de programação linear inteira para o problema. A denominação aqui adotada, qual seja, *Open Dimensional Problem* (ODP), é introduzida pela tipologia de Wäscher et al. (2007).

Apesar de alguns trabalhos solucionarem o PCDA utilizando métodos exatos, como Hifi (1998) e Cui et al. (2008), que utilizam um método baseado em *branch-and-bound*, as principais pesquisas sobre o problema desenvolvem técnicas heurísticas. Tal escolha se deve ao fato do problema pertencer à classe de problemas NP-difíceis, veja Garey e Johnson (1979). Os trabalhos de Kroger (1995), Liu e Teng (1999), Yeung e Tang (2004) e de de Andrade (2009), por exemplo, apresentam métodos baseados em algoritmos evolutivos. No trabalho Alvarez-Valdez et al. (2008), é proposta uma metodologia para resolver o PCDA, na sua versão não-guilhotinada, baseada na metaheurística *Greedy randomized adaptive search procedure* (GRASP).

Neste trabalho, é proposta uma aplicação da metaheurística GRASP, em conjunto com a técnica de encaixe *Best-Fit Decreasing Height*, para solucionar o PCDA guilhotinado. Em especial, é proposto um estudo da relação da contribuição de cada fase (construção e busca local) desta metaheurística na geração da solução final do problema. O artigo está organizado como segue: a seção 2 descreve o PCDA e as definições de interesse do presente trabalho; a seção 3 apresenta a metodologia proposta; a seção 4 mostra os resultados computacionais obtidos, para cada fase do algoritmo proposto, para as instâncias da literatura, e, por fim, a seção 5, apresenta as conclusões a respeito do desenvolvimento realizado.

2 Problema de Corte com Dimensão Aberta

O Problema de Corte com Dimensão Aberta (PCDA) estuda o corte de um conjunto de itens retangulares, com largura e altura definidos, a partir de um único objeto, que possui largura fixa e altura variável. Este problema ainda pode ser definido de acordo com as seguintes restrições:

- Orientação: define se o item poderá ou não sofrer rotação em relação a alguns eixos.
- Tipo de corte: guilhotinado, se o corte se estende de um lado ao outro do objeto, produzindo, a cada corte, dois retângulos; ou não-guilhotinado, se o corte acompanha o contorno do item, sem descaracterizar o objeto.

O PCDA abordado neste trabalho será definido de acordo com as seguintes restrições:

i) Os itens são alocados de forma ortogonal no objeto;
ii) Os itens apresentam orientação fixa;
iii) O corte é do tipo guilhotinado;
iv) O corte é feito em 2-estágios.

O corte guilhotinado pode ser classificado de acordo com o número de estágios em que é feito o corte. Cada estágio representa uma mudança ortogonal na direção do corte. Assim, o corte em 2-estágios implica que o primeiro corte é realizado em uma direção (horizontal) e depois em outra direção (vertical), ocorrendo apenas uma mudança na direção do corte.

O problema consiste em um objeto S de largura W e altura grande o suficiente ("altura infinita") para alocar todos os itens de uma lista de itens retangulares $I = \{r_1, \cdots, r_n\}$, em que cada item $r_i = (w_i, h_i)$, tal que $w_i \leq W$, para $i = 1, \cdots, n$, possui largura w_i e altura h_i. O objetivo do problema é cortar os itens de I em S utilizando a menor altura possível do objeto, como mostra a Figura 1.

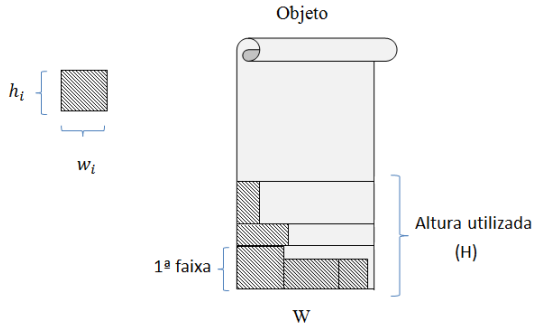

Figura 1. Solução para o PCDA guilhotinado envolvendo 5 itens.

2.1 Formulação Matemática

Para a formulação matemática do PCDA, considere as seguintes proposições, apresentadas por Lodi et al. (2004), para uma dada solução:

i) O primeiro item é alocado em cada faixa mais a esquerda, e possui maior altura;
ii) A primeira faixa do objeto (mais baixa) é a mais alta;
iii) Os itens são ordenados de forma decrescente em relação à altura ($h_1 \geq h_2 \geq \cdots \geq h_n$).

A altura de cada faixa corresponde à altura h_i do primeiro item i a ser alocado. Dessa forma, considerando n o número de possíveis faixas formadas para alocar os itens, então a variável de decisão y_i é definida como:

$$y_i = \begin{cases} 1, & \text{se o item } i \text{ inicializa a faixa } i;\ (i = 1, \cdots, n) \\ 0, & \text{caso contrário.} \end{cases}$$

Além disso, apenas os itens j tais que $j > i$ podem ser colocados na faixa. Desse modo, dado um item j tal que $j = i$ inicia a faixa i, ele não poderá ser atribuído à faixa novamente. Dessa forma, a variável binária é definida por:

$$x_{ij} = \begin{cases} 1, & \text{se o item } j \text{ está alocado na faixa } i,\ \ (i = 1, \cdots, n-1);\ j > i \\ 0, & \text{caso contrário} \end{cases}$$

Com bases nessas considerações, a formulação matemática do problema é dada por:

$$\min \quad \sum_{i=1}^{n} h_i.y_i \tag{1}$$

$$\text{sujeito a:} \quad \sum_{i=1}^{j-1} x_{ij} + y_j = 1, \quad j = 1, \cdots, n; \tag{2}$$

$$\sum_{j=i+1}^{n} w_j x_{ij} \leq (W - w_i) y_i, \ \ i = 1, \cdots, (n-1); \tag{3}$$

$$x_{ij} \in \{1, 0\} \qquad i = 1, \cdots, (n-1); j > i \tag{4}$$

$$y_i \in \{0, 1\} \qquad i = 1, \cdots, n;$$

A função objetivo (1) tem como critério minimizar a altura utilizada do objeto. A restrição (2) garante que o item só será alocado uma vez no objeto; a restrição (3) garante que a largura do objeto não será ultrapassada em nenhuma das faixas; e, por fim, as restrições (4) definem o tipo de variável do problema.

3 Metodologia Proposta

Nesta seção será descrita como uma solução para o PCDA é representada, a técnica de encaixe utilizada para alocar os itens dentro do objeto, a função de avaliação utilizada, a estrutura de vizinhança adotada, a metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) aplicada ao problema e uma descrição das fases (construção e busca local) desta metaheurística segundo a implementação realizada.

3.1 Representação de uma solução

Cada solução do PCDA é representada por uma sequência de números inteiros, em que cada posição da sequência representa a ordem na qual o item é encaixado. Para um problema com quatro itens, por exemplo, uma possível solução é $s = (2\ \ 4\ \ 3\ \ 1)$, de modo que o primeiro item a ser encaixado é o item 2 e o último é o item 1.

3.2 Técnica de Encaixe

Para determinar a menor altura utilizada do objeto, é necessário encontrar a melhor combinação do conjunto de itens, para encaixá-los no objeto, ao realizar o corte. As técnicas de encaixe (ou Algoritmo de Nível) são técnicas desenvolvidas para realizar o encaixe dos itens nos objetos. Nesses algoritmos, os itens são alocados em faixas, da esquerda para a direita, sendo que o início de uma nova faixa coincide com o topo do item mais alto da faixa anterior. A utilização destas técnicas, se deve ao fato delas serem mais rápidas e gerarem padrões guilhotináveis. Os algoritmos de níveis mais utilizados são: *Next-Fit* (NF); *First-Fit* (FF); *Best-Fit* (BF); *Next-Fit Decreasing Height* (NFDH); *First-Fit Decreasing Height* (FFDH); e *Best-Fit Decreasing Height* (BFDH). Para uma revisão a respeito destas técnicas, veja Ntene e van Vuuren (2009).

3.2.1 *Best-Fit Decreasing Height* (BFDH)

Neste trabalho, para realizar o encaixe dos itens, é utilizado o algoritmo de nível *Best-Fit Decreasing Height*. Nesta técnica, as faixas são preenchidas pelos itens na ordem em que eles aparecem na solução. Os itens são inseridos no canto inferior esquerdo do objeto até que a largura do objeto não seja mais suficiente.

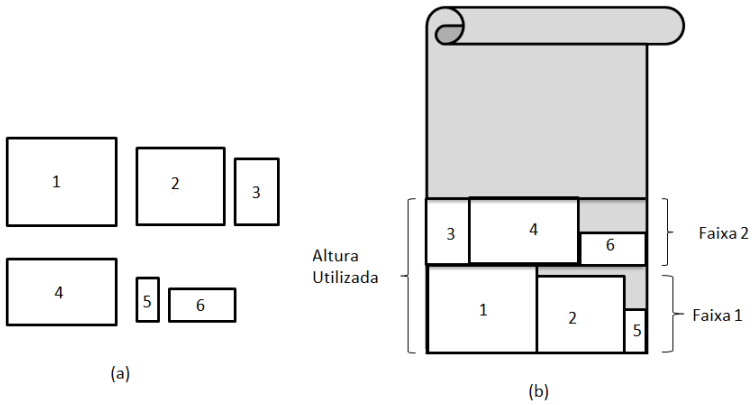

Figura 2. A figura mostra uma sequência de seis itens (a) encaixados em uma bobina (b) de acordo com o Algoritmo de Nível *Best-Fit Decreasing Height*.

No BFDH, após inserir o primeiro item (linha 4 do Algoritmo 1), é pesquisado a largura disponível das faixas existentes. O próximo item é inserido na faixa que resultar o menor espaço residual em relação à largura (linha 7). Uma nova faixa só é formada (linha 9) se o item não puder ser inserido em nenhuma das faixas existentes. Sendo assim, é permitido que as faixas formadas fiquem abertas, possibilitando que o item seja inserido apenas na faixa que resultar o melhor encaixe, como pode ser observado em Ortmann (2010). A figura 2 mostra como é feito, para esta técnica, o encaixe de uma sequência de seis itens. O pseudocódigo do BFDH é apresentado no Algoritmo 1.

3.3 Função de Avaliação

O objetivo do problema é minimizar a altura utilizada do objeto. Assim, a função de avaliação é dada pelo somatório da altura de cada faixa, na forma:

$$\sum_{i=1}^{n} h_i.y_i \quad (i = 1, \cdots, n)$$

Algoritmo 1: *Best-Fit Decreasing Height*

Entrada: Conjunto de itens (I), de dimensões (w_i, h_i), e a largura do objeto (W)
Saída: sequência do encaixe, altura utilizada

```
1  início
2      n = quantidade de itens;
3      faixa = 1, i = 1;
4      Insira o primeiro item na primeira faixa;
5      Altura(faixa) = h(I_i);
6      para i=2 até n faça
7          Insira o item na faixa que resultar o melhor encaixe;
8          se o item não puder ser inserido em nenhuma das faixas então
9              Forme uma nova faixa;
10             faixa = faixa + 1;
11         fim
12     fim
13 fim
```

em que n é a quantidade de itens que serão encaixados, h_i representa a altura do item i e y_i determina se o item i inicializa a faixa ou não.

3.4 Estrutura de Vizinhança

A vizinhança de uma solução s é o conjunto de soluções $N(s)$, em que cada solução $s' \in N(s)$ é obtida a partir de um movimento feito na solução corrente s.

Neste trabalho, para determinar a vizinhança de uma dada solução s, é aplicado o movimento de troca de dois itens dentro do objeto. A estrutura de vizinhança utilizada permuta cada item da solução com os demais itens dentro do objeto. Por exemplo, se a solução corrente é $s = (4\ 1\ 2\ 3)$, então $s' = (2\ 1\ 4\ 3)$ é um vizinho de s, obtido pela troca de posição dos itens 4 e 2.

3.5 *Greedy Randomized Adaptive Search Procedure* (GRASP)

Para resolver o PCDA, é implementado o algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*), proposto por Feo e Resende (1995), cujo pseudocódigo está ilustrado no Algoritmo 2. GRASP é um método constituído de duas fases, denominadas fase de construção e fase de refinamento, que são aplicadas repetidamente, retornando a melhor das soluções encontradas. As subseções seguintes mostram como esse algoritmo foi adaptado ao PCDA.

3.5.1 Fase de Construção

Na fase de construção do Algoritmo GRASP (linha 4 do Algoritmo 2), uma solução é construída pela inserção dos itens, sendo um de cada vez. O pseudocódigo dessa fase está apresentado no Algoritmo 3.

Este procedimento se inicia na linha 3 do Algoritmo 3, com a construção de uma lista de itens candidatos C, que é formada pelo conjunto dos itens I que serão inseridos. Os elementos (itens) dessa lista são ordenados de forma decrescente, de acordo com uma função $g(t)$, que avalia o benefício do candidato ser incluído na solução. A função $g(\cdot)$ retorna, para cada item t da lista C, uma nota, que corresponde a altura deste item. A cada iteração da fase de construção, os k melhores itens da lista C, ou seja, os k itens

Algoritmo 2: *GRASP*

```
   Entrada: f(.), g(.), N(.), GRASPmax,s
   Saída: s
1  início
2      f* ← ∞;
3      para Iter=1 até GRASPmax faça
4          Construção( g(.), k, s);
5          Buscalocal (f(.), N(.), s);
6          se (f(s) < f*) então
7              s* ← s;
8              f* ← f(s);
9          fim
10     fim
11     s ← s*;
12     Retorne s;
13 fim
```

mais altos, são colocados em uma Lista Restrita de Candidatos (LRC). Na implementação feita, adotou-se, para fins de análise, os valores $k = 2$ e $k = 5$. O componente aleatório do método consiste em, para cada inserção na solução, selecionar aleatoriamente um item dentre os k itens candidatos da LRC. Em seguida, atualizam-se as listas C e LRC e repete-se o processo, até que todos os itens tenham sido incluídos na solução.

O encaixe dos itens da solução inicial gerada é feita utilizando-se a técnica de encaixe *Best-Fit Decreasing Height*, conforme a linha 11 do Algoritmo 3. Em seguida, é determinado o valor da função de avaliação desta solução inicial.

Algoritmo 3: Construção

```
   Entrada: g(.), k,s
   Saída: s
1  início
2      s* ← ∅;
3      Inicia o conjunto C de itens candidatos;
4      Ordene o Conjunto C de acordo com g(.);
5      enquanto C ≠ ∅ faça
6          LRC = {conj. dos k melhores itens de C};
7          Selecione, aleatoriamente, um item t ∈ LRC;
8          s* ← s* ∪ {t};
9          Atualize o conjunto C de itens candidatos;
10     fim
11     s ← Best-Fit Decreasing Height de s*;
12     Retorne s;
13 fim
```

3.5.2 Fase de busca local

A fase de refinamento do algoritmo proposto (linha 5 do Algoritmo 2) consiste na aplicação do procedimento de busca local *Best Improvement* (BI). Neste método, parte-se de uma solução s e, a cada iteração, são analisados todos os possíveis vizinhos, movendo-se somente para aquele que tiver o valor mais favorável da função de avaliação, isto é, a melhor solução é aquela cujo somatório das alturas dos itens que estão na primeira posição de cada faixa é menor. Dada uma solução, aplica-se a técnica de encaixe BFDH, descrita

no Algoritmo 1, para determinar o valor da função de avaliação. O método para quando encontra um ótimo local. O Algoritmo 4 ilustra o pseudocódigo deste procedimento de busca local.

Algoritmo 4: *Best Improvement*

Entrada: $f(.), s$
Saída: s
início
  $V = \{s' \in N(s) \mid f(s') < f(s)\}$; (Avalia toda a vizinhança de s)
  enquanto $V \neq \emptyset$ **faça**
    Selecione $s' \in V$, sendo $s' = \arg\min \{f(s') | s' \in V\}$;
    se $(f(s') < f(s))$ **então**
      $s \leftarrow s'$;
      $f(s) \leftarrow f(s')$;
    fim
    $V = \{s' \in N(s) \mid f(s') < f(s)\}$;
  fim
  Retorne s;
fim

4 Resultados

O algoritmo GRASP foi desenvolvido utilizando a plataforma JAVA 5, e os testes foram feitos em um PC Intel Core 2 Duo 2.4 GHz, com 2 GB de memória RAM, em ambiente Windows 7.

Para realizar os testes computacionais, foram utilizados os problemas-teste de Hopper e Turton (2001a), que estão disponíveis na OR-Library Beasley (1990). Estes problemas estão divididos em sete categorias, sendo que cada categoria é sub-dividida em três problemas, em um total de 21 problemas, como mostra a Tabela 1. Cada problema possui o número de itens e a largura do objeto conforme apresentados nessa Tabela.

Tabela 1. Problemas-teste de Hopper e Turton (2001)

Categoria		C1	C2	C3	C4	C5	C6	C7
nº	P1	16	25	28	49	73	97	196
de	P2	17	25	29	49	73	97	197
itens	P3	16	25	28	49	73	97	196
Largura	do objeto	20	40	60	60	60	80	160

Nesta seção são apresentados os resultados obtidos da solução do PCDA guilhotinado dos problemas-testes da tabela 1 para 30 execuções do algoritmo proposto. Os parâmetros utilizados foram: o número de iterações realizadas igual a 100 ($GRASPmax = 100$) e o valor de k, que determina o tamanho da LRC, igual a 2 ou 5.

Nas tabelas 2 e 3, são apresentados os resultados obtidos em cada fase da metaheurística estudada utilizando a técnica de encaixe *Best-Fit Decreasing Height*, com o tamanho da LRC iguais a $k = 2$ e $k = 5$, respectivamente. Em ambas as tabelas, a terceira coluna mostra os resultados listados em de Andrade (2009) gerados pela resolução do modelo matemático do problema utilizando o *software* de otimização CPLEX, versão 11.0, limitado a 1 hora de processamento. Na tabela, os resultados marcados com "$*$" representam as soluções ótimas do problema encontradas pelo CPLEX em menos de 1 hora de processamento.

Tabela 2. Resultados obtidos pela metodologia proposta em cada uma das fases, para $k = 2$. "*": Solução ótima.

Problemas testes	Nº de itens	método Exato	Fase de Construção			Fase de Busca Local			Gap(%) da solução	
			Melhor	Média	Tempo	Melhor	Média	Tempo	Melhor	Média
C1P1	16	27*	27*	27,13	0,0006	27*	27*	0,0130	0	0
C1P2	17	29*	29*	29,63	0,0007	29*	29,033	0,0150	0	2
C1P3	16	23*	23*	23*	0,0006	23*	23*	0,0130	0	0
C2P1	25	20*	20*	20*	0,0008	20*	20*	0,0380	0	0
C2P2	25	34*	34*	34*	0,0006	34*	34*	0,0380	0	0
C2P3	25	23*	23*	23*	0,0008	23*	23*	0,0400	0	0
C3P1	28	40*	40*	40,06	0,0008	40*	40*	0,0540	0	0
C3P2	29	42*	42*	42,36	0,0007	42*	42*	0,0600	0	1
C3P3	28	43*	43*	43*	0,0008	43*	43*	0,0550	0	0
C4P1	49	74*	74*	74,50	0,0010	74*	74,1	0,3050	0	1
C4P2	49	74*	74*	75,40	0,0011	74*	74,233	0,3090	0	2
C4P3	49	80*	80*	80,36	0,0010	80*	80*	0,3000	0	0
C5P1	73	101	98	98,66	0,0013	98	98,266	1,0800	0	0
C5P2	73	106*	106*	106,03	0,0014	106*	106,03	1,1040	0	0
C5P3	73	106	106	107,40	0,0013	106	106,633	1,2610	0	1
C6P1	97	136	136	136,36	0,0015	136	136	2,5690	0	0
C6P2	97	145	142	142,2	0,0015	142	142	2,4140	0	0
C6P3	97	139	139	139,40	0,0016	139	139,1	2,5120	0	0
C7P1	196	263	261	261,43	0,0027	261	261,05	21,878	0	0
C7P2	197	283	282	283,30	0,0030	282	282,8	24,315	0	0
C7P3	196	283	273	274,60	0,0027	272	273,06	21,940	0,36	1

Tabela 3. Resultados obtidos pela metodologia proposta em cada uma das fases, para $k = 5$. "*": Solução ótima.

Problemas testes	Nº de itens	método Exato	Fase de Construção			Fase de Busca Local			Gap(%) da solução	
			Melhor	Média	Tempo	Melhor	Média	Tempo	Melhor	Média
C1P1	16	27*	27*	29	0,001	27*	27,53	0,016	0	5
C1P2	17	29*	31	32,06	0,001	29*	30,13	0,016	6,4	6
C1P3	16	23*	25	26,86	0,001	23*	23,73	0,015	8	12
C2P1	25	20*	20*	23,06	0,001	20*	20,13	0,045	0	13
C2P2	25	34*	34*	35,73	0,001	34*	34	0,043	0	5
C2P3	25	23*	23*	24,86	0,001	23*	23*	0,050	0	7
C3P1	28	40*	41	44,53	0,001	40*	41,26	0,060	2,4	7
C3P2	29	42*	46	50,53	0,001	42*	45,60	0,066	8,6	10
C3P3	28	43*	46	48	0,001	43*	44,60	0,063	6,5	7
C4P1	49	74*	78	79,80	0,001	77	78	0,331	1,2	2
C4P2	49	74*	78	88,80	0,001	77	81,10	0,335	1,2	9
C4P3	49	80*	80*	88	0,001	80*	83,76	0,301	0	5
C5P1	73	101	102	104,86	0,002	101	103,10	1,163	0,98	2
C5P2	73	106*	112	116,73	0,001	108	112,40	1,197	3,57	4
C5P3	73	106	111	117,93	0,002	109	114,10	1,218	1,80	3
C6P1	97	136	143	145,40	0,002	140	142,20	2,781	2,09	2
C6P2	97	145	146	152,46	0,002	145	148,10	2,527	0,68	3
C6P3	97	139	146	155,73	0,001	146	150,50	2,602	0	4
C7P1	196	263	268	276,50	0,003	266	270,80	22,809	0,74	2
C7P2	197	283	293	305,26	0,003	285	295,10	25,093	2,73	3
C7P3	196	283	286	292,86	0,003	284	287,80	23,619	0,699	2

As colunas 4 e 5 das Tabelas 2 e 3 apresentam o melhor resultado e o resultado médio, respectivamente, alcançados durante a fase de construção. Já as colunas 7 e 8 mostram o melhor resultado e o resultado médio da fase de busca local. Os tempos médios dos problemas-testes de cada fase são representados pelas colunas 6 e 9, e as colunas 10 e 11, reproduzem nesta ordem, os valores referentes ao erro relativo (*Gap*) da melhor solução

e da solução média dos resultados. O *Gap* da coluna 10 pode ser calculado pela seguinte fórmula:

$$gap = \frac{C_{melhor} - BL_{melhor}}{C_{melhor}}$$

em que C_{melhor} é o resultado da melhor solução encontrada entre as 30 iterações da fase de construção e BL_{melhor} é o resultado da melhor solução encontrada entre as 30 iterações da fase de busca local.

O *Gap* da solução média é calculado usando método semelhante, em que:

$$gap = \frac{C_{media} - BL_{media}}{C_{media}}$$

em que C_{medio} é o resultado da solução média das 30 execuções da fase de construção e BL_{melhor} é o resultado da solução média das 30 execuções da fase de busca local.

Na Tabela 2, em que o tamanho da LRC é $k = 2$, foram encontrados as soluções ótimas, em ambas as fases da metaheurística, para as instâncias com até 49 itens e para a instância C5P2. Para as instâncias C5P1, C5P3, C6P1 e C6P3, os mesmos resultados do método exato foram encontrados. Para os problemas-testes C6P2, C7P1 ao C7P3, os resultados obtidos foram melhores que os obtidos pelo CPLEX.

Durante as 30 execuções do algoritmo, o erro relativo da melhor solução encontrada foi de 0% na maioria dos problemas-testes, apresentando uma pequena variação de $0,36\%$ para o problema C7P3. Já o *gap* da solução média foi de 10% para as instâncias C3P2, C4P1, C5P3 e C7P3 e de 20% para as instâncias C1P2 e C4P2.

Na tabela 3, em que o tamanho da LRC é $k = 5$, foram encontrados as soluções ótimas para as instâncias C1P1, C2P1, C2P2, C2P3 e C4P3 em ambas as fases do GRASP, e para as instâncias C1P2, C1P3, C3P1, C3P2 e C3P3 apenas na fase de busca local. Exceto nas instâncias C5P1 e C6P2, que obtiveram resultados iguais ao do método exato na fase de busca local, para as instâncias com 73 itens ou mais foram encontradas, durante as duas fases, soluções piores.

Para as 30 execuções do algoritmo com LRC de tamanho $k = 5$, o erro relativo da melhor solução encontrada foi de 0% apenas nos problemas-testes C1P1, C2P1, C2P2, C2P3, C4P3 e C6P3. Para os demais instâncias, o *gap* se portou variando até 8%, sendo destaque para as instâncias C6P2 e C6P7, que obtiveram o menor *gap* de $0,6\%$, e para as instâncias C1P3 e C3P2 que obtiveram o maior *gap* de 8%. Já o erro relativo da solução média variou entre 1% a 13%, com média de 5%.

5 Conclusão

Este artigo apresentou um estudo da relação da contribuição de cada fase (construção e busca local) da metaheurística GRASP, em conjunto com a técnica de encaixe *Best-Fit Decreasing Height*, na geração da solução final do Problema de Corte de Dimensão Aberta Guilhotinado.

Os resultados se mostraram promissores para as instâncias de Hopper e Turton (2001a) com até 49 itens, pois foram encontrados para estas instâncias as soluções ótimas para os valores de k testados. Para as instâncias com maior quantidade de itens, foram

encontrados resultados melhores, para $k = 2$, que os resultados obtidos pelo método exato, em um tempo computacional bem mais baixo.

Tomando como base o tamanho da lista restrita de candidatos (LRC) igual a $k = 2$, utilizado na metaheurística GRASP e exemplificado pela Tabela 2, todos os resultados obtidos, tanto para a fase de construção quanto para a fase de busca local, foram iguais ou superiores aos relatados pelo método exato. Em apenas um dos problemas-testes - C7P3, o resultado da fase de busca local superou o resultado da fase de construção. A média dos resultados alcançados durante a fase de busca local é melhor que os da fase de construção; contudo, o tempo médio computacional é, por exemplo para a instância C7P3, 8 mil vezes maior. Para tanto, a utilização da busca local como refinamento para alcançar melhores resultados se mostrou desnecessária, visto que somente em um dos casos o resultado final foi superior. Em outras palavras, a busca local agrega um alto custo computacional, sem garantias de melhoras no resultado.

Nos resultados da Tabela 3, para o tamanho de LRC igual a $K = 5$, os resultados obtidos após o refinamento são melhores que os resultados encontrados durante a fase de construção em no máximo 8% para duas instâncias (C1P3 e C3P2); para as demais instâncias, essa melhora é pequena. Porém, o tempo computacional durante a fase de busca local é quase 350 vezes maior para instâncias com 49 itens, como, por exemplo, para C4P1, ao da fase de construção, e quase 8 mil vezes superior, para as instâncias maiores, como a instância C7P2. Dessa forma, para este caso, apesar do resultado da busca local ser melhor que o resultado da fase de construção para a maioria dos problemas, essa melhora não é significante, considerando o tempo computacional gasto durante o processo. Sendo assim, a utilização da busca local, neste caso, também se mostra desnecessária.

Em relação ao valor da função objetivo para o tamanho da LRC igual a $k = 5$, como mostra a Tabela 3, o algoritmo encontrou resultados inferiores aos obtidos para o tamanho da LRC igual a $k = 2$ na tabela 2. Isso mostra que quanto maior o tamanho da LRC, piores serão os resultados encontrados.

Por fim, a qualidade dos resultados alcançados durante a fase de construção, quando comparados com os obtidos pela fase de busca local, foram poucos significantes se tomarmos como base o esforço computacional que a busca local exerce sobre o algoritmo. Para tanto, a qualidade dos resultados não necessariamente estão relacionadas ao refinamento da solução gerada pela fase de construção. Futuramente, pretende-se testar outras estruturas de vizinhanças dentro da metodologia proposta e verificar se podem podem contribuir com a melhora da solução.

Agradecimentos

Agradecemos ao CNPq, ao CEFET-MG e à FAPEMIG pelo apoio ao desenvolvimento do trabalho, assim como ao Prof. Marcone Jamilson Freitas Souza, da Universidade Federal de Ouro Preto, pela co-orientação no desenvolvimento do mesmo.

Referências

Alvarez-Valdez, R.; Parreno, F. e J. M. Tamarit, J. M. (2008). Reactive grasp for the strip-packing problem. *Computers & Operations Research*, v. 35, p. 1065–1083.

Beasley, John E. (1990). Or - library: Distributing test problems by electronic mail. *Journal of the Operation Research Society*, v. 41, p. 1069 – 1072.

Cui, Yaodong; Yang, Yuli; Cheng, Xian e Song, Peihua. (2008). A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem. *Computers & Operations Research*, v. 35, p. 1281–1291.

deAndrade, Mariana Silva Faleiro. (2009). Algoritmos evolutivos mono e multiobjetivos para problemas bidimensionais de corte. Master's thesis, Centro Federal de Educação Tecnológica de Minas Gerais.

Feo, Thomas A. e Resende, Maurício G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 9, p. 849–859.

Garey, Michael R. e Johnson, David S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, Freeman, San Francisco.

Gilmore, P. C. e Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, v. 9, p. 849–859.

Gilmore, P. C. e Gomory, R. E. (1963). A linear programming approach to the cutting stock problem - part ii. *Operations Research*, v. 11, p. 863 – 888.

Hifi, Mhand. (1998). Exact algorithms for the guillotine strip cutting/packing problem. *Computers & Operations Research*, v. 25, p. 925–940.

Hopper, E. e Turton, B. C. (2001)a. An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operations Research*, v. 16, p. 257–300.

Hopper, E. e Turton, B. C. (2001)b. A review of the application of meta-heuristic algorithm to 2d strip packing problems. *Artificial Intelligence Review*, v. 16, p. 257–300.

Kroger, B. (1995). Guillotineable binpacking: A genetic approach. *European Journal of Operations Research*, v. 84, p. 645–661.

Liu, D. e Teng, H. (1999). An improved bl-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operations Research*, v. 112, p. 413–420.

Lodi, Andrea; Martello, Silvano e Vigo, Daniele. (2004). Models and bounds for two-dimensional level packing problems. *Journal of Combinatorial Optimization*, v. 8, p. 363–379.

Martelo, S.; Monaci, M. e Vigo, D. (2003). An exact approach to strip-packing problem. *INFORMS Journal on Computing*, v. 15, p. 310–319.

Ntene, N. e vanVuuren, J. H. (2009). A survey and comparison of guillotine heuristics for the 2d oriented offline strip packing problem. *Discrete Optimization*, v. 6, p. 174–188.

Ortmann, Frank Gerald. *Heuristics for Offline Rectangular Packing Problems*. PhD thesis, Department of Logistics, Stellenbosch University, (2010).

Wäscher, Gerhard; HauBner, Heike e Schumann, Holger. (2007). An improved typology of cutting and packing problems. *European Journal of Operations Research*, v. 183, p. 1109–1130.

Yeung, L. H. W. e Tang, W. K. S. (2004). Strip-packing using hybrid genetic approach. *Engineering Applications of Artificial Intelligence*, v. 17, p. 169–177.