UM ALGORITMO EVOLUTIVO HÍBRIDO APLICADO À SOLUÇÃO DO PROBLEMA DE CORTE BIDIMENSIONAL GUILHOTINADO

Mariana Silva Faleiro de Andrade, Sérgio Ricardo de Souza, Elias Carlos Corrêa Temponi
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas, 7675, 30.510-000, Belo Horizonte, MG, Brasil
marifaleiro@yahoo.com.br, sergio@dppg.cefetmg.br, elias@uai.com.br

Marcone Jamilson Freitas Souza
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário – 35.400-000 – Ouro Preto – MG – Brasil
marcone@iceb.ufop.br

RESUMO

Este trabalho trata do problema de corte com dimensão aberta guilhotinado. Dado um conjunto de peças menores (itens) e uma peça maior (objeto) com largura fixa e altura variável, o problema consiste em alocar ortogonalmente todos os itens no objeto, de maneira que a altura utilizada seja minimizada. Propõe-se, para sua resolução, um Algoritmo Evolutivo Híbrido. Este algoritmo utiliza a metaheurística *Iterated Local Search* para gerar sua população inicial, os algoritmos aproximados *First-Fit* e *Best-Fit*, além de um operador de busca local (Método Randômico de Descida). Experimentos computacionais realizados sobre um conjunto de problemas-teste da literatura indicam que o método proposto consegue alcançar o valor ótimo na maioria dos problemas de pequenas e médias dimensões (16 a 49 itens) e, em problemas de dimensões maiores (73 a 197 itens), obtém desvios baixos em relação à solução encontrada pelo método exato.

PALAVRAS CHAVE. Problema de corte bidimensional, Algoritmos Evolutivos, *Iterated Local Search*.

ABSTRACT

This paper deals with the guilhotined Open Dimensional Problem (ODP). Given a set of rectangular small items and a rectangular large object of fixed width and variable length, the problem consists of orthogonally placing all the items within the large object, such that the overall length of the layout is minimised. A Hybrid Evolutionary Algorithm is proposed to solve the problem. In the algorithm, we use an Iterated Local Search Algorithm to improve the quality of inicial population, as well the approximate algorithms *First-Fit* and *Best-Fit*, and a Local Search Operator (Random Descent Method) to solve it. Computational experiments realized with instances from the literature show that the method proposed is able to reach the optimal solution in all of the small and medium instances (16 to 49 items) and yield small gaps in instances with 73 to 197 items.

KEYWORDS. Bidimensional Cutting Problem, Evolutionary Algorithms, Iterated Local Search.

1. Introdução

Em diversas atividades industriais é necessário cortar peças menores (itens), a partir de peças maiores (objetos). Os objetos podem ser, por exemplo, barras de aço, bobinas de papel, alumínio e tecido, placas metálicas e de madeira, chapas de vidro e fibra de vidro, peças de couro, etc. Nessa atividade, o objetivo é atender a demanda de itens, de maneira a minimizar os custos envolvidos com a perda de material. A variedade de situações práticas envolvendo este processo e a necessidade de otimizá-lo fizeram surgir uma classe de problemas de otimização combinatória, denominada Problemas de Corte e Empacotamento (PCE). Os PCE consistem em determinar o melhor arranjo de um conjunto de itens sobre um objeto, que precisa ser cortado ou empacotado.

Neste trabalho, é estudado um caso particular de problema de corte, denominado Problema de Corte com Dimensão Aberta (ODP – *Open Dimensional Problem*), segundo a tipologia de Wäscher et al. (2006). Como o próprio nome sugere, trata-se de um problema que possui dimensões variáveis, sendo, portanto, definido para duas ou mais dimensões. O ODP tratado neste trabalho apresenta as seguintes características: trata de itens com duas dimensões (retangulares) e de diversos tipos (muito heterogêneos) e de um objeto com duas dimensões, sendo uma delas fixa.

O ODP bidimensional pode ser definido da seguinte forma: consiste em determinar o arranjo de um conjunto de itens sobre um objeto que possui largura (W) fixa e altura (H) variável, tendo como objetivo minimizar a altura utilizada do objeto. Esse problema ocorre em fábricas que realizam o corte de bobinas ou rolos, como, por exemplo, as fábricas de papel, aço e tecidos.

Por se tratar de um problema de corte bidimensional, são acrescidas outras classificações, devidas às restrições impostas pelos equipamentos e pelo processo de corte. Uma dessas restrições é com relação ao tipo de corte, que pode ser guilhotinado, quando o corte se estende de um lado ao outro do objeto, produzindo, a cada corte, dois retângulos, ou não-guilhotinado, se o corte acompanhar o contorno do item, sem descaracterizar o objeto. O problema também pode considerar ou não rotação ortogonal dos itens. Além disso, pode haver limitação no número de estágios de corte. Cada estágio indica uma mudança na direção do corte. No caso de ocorrer apenas uma mudança na direção, tem-se um corte do tipo 2-estágios.

Os PCE têm sido extensivamente tratados na literatura. Devido às diversas aplicações práticas do problema, e à sua complexidade (geralmente são NP-Difíceis) (Berkey e Wang, 1987; Coffman, Garey e Johnson, 1982), a maioria dos trabalhos encontrados na literatura apresenta abordagens heurísticas para sua resolução. Revisões sobre a utilização de heurísticas e metaheurísticas na resolução dos PCE bidimensionais são encontradas em Lodi, Martello e Monaci (2002), Hopper e Turton (2001) e Lodi, Martello e Vigo (1999).

No caso do ODP, vários autores já mostraram que se trata de um problema NP-Difícil (Hochbaum, 1985; Leung *et al.*, 1990). Devido a esse fato e também ao enorme número de aplicações práticas do problema, são encontrados vários trabalhos na literatura que têm seu foco em técnicas para a resolução do ODP.

Hifi (1998) apresenta um método exato para o ODP guilhotinado e com rotações ortogonais, baseado num procedimento *branch-and-bound*, que soluciona problemas de pequeno porte. Lesh *et al*. (2004) também apresenta um método baseado em *branch-and-bound* para o ODP sem perdas, para problemas com até 30 itens. No entanto, os métodos exatos não se mostraram capazes de solucionar problemas de grande porte, justificando, assim, a utilização de heurísticas e metaheurísticas.

Dentre as metaheurísticas utilizadas para solucionar o ODP, destaca-se o uso de métodos baseados em Algoritmos Genéticos (AG). Dentre eles podemos citar Yeung e Tang (2004), que tratam de um ODP não-guilhotinado através de uma heurística de encaixe de itens, que transforma o ODP num simples problema de permutação, que então é solucionado por um AG. Um método baseado em AG também é apresentado em Liu e Teng (1999). Uma revisão sobre a utilização de AG na resolução do OPD é feita em Kroger (1995).

Neste trabalho é utilizado um modelo de programação linear inteira proposto por Lodi *et al.* (2004) para representar o ODP guilhotinado. Para solucionar o problema, é proposto um

método heurístico híbrido baseado nas metaheurísticas AG e *Iterated Local Search* (ILS). Além disso, são comparados os desempenhos das técnicas de encaixe *First-Fit* e *Best-Fit,* utilizadas como sub-rotina do método.

Os resultados obtidos pelo método proposto são comparados com aqueles apresentados em Temponi (2007), obtidos pela resolução do modelo de Programação Linear Inteira pelo otimizador LINGO, versão 8.0.

Este trabalho está organizado como segue. Na seção 2 são descritas as características do problema estudado, enquanto na seção 3 é apresentado o modelo de programação linear inteira para representar o mesmo. Na seção 4 são apresentadas as técnicas de encaixe utilizadas no trabalho, enquanto na seção 5 é desenvolvida a metodologia heurística de resolução do ODP guilhotinado. A seção 6 resume a metodologia utilizada. Na seção 7 são apresentados e discutidos os resultados computacionais. A seção 8 conclui o trabalho.

2. Descrição do Problema Estudado

O problema estudado neste trabalho é o *Open Dimensional Problem* (ODP) guilhotinado. Esse problema de corte possui as seguintes características: o corte é irrestrito (não há limite para o número de vezes em que determinado tipo de item é cortado), os itens são alocados ortogonalmente nos objetos, não é permitida a rotação dos itens, o corte é do tipo guilhotinado de 2-estágios e os itens têm demanda unitária.

Assim sendo, o ODP pode ser descrito como segue. Seja S um objeto de largura W e altura grande o suficiente para alocar todos os itens ("altura infinita") e uma lista de itens retangulares $I = \{r_1,..., r_n\}$, onde cada item $r_i = (w_i, h_i)$, tal que $w_i \leq W$, para $i = 1,...,n$, sendo w_i a largura e h_i a altura dos itens. O objetivo é cortar os itens de I em S com a menor altura possível. Por exemplo, dada a solução $s = (5,2,4,1,3,6)$, esta pode ser representada conforme a Fig. 1.

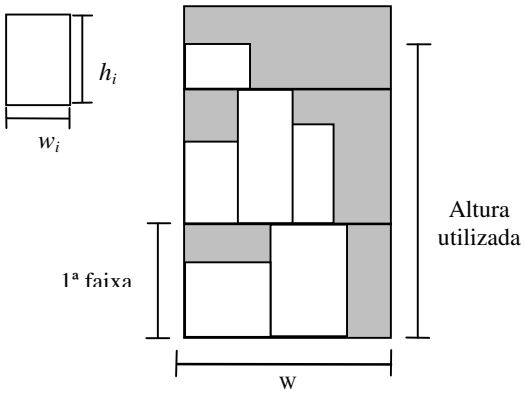

Figura 1 – *Open dimensional problem* guilhotinado 2-estágios

3. Modelagem Matemática

O modelo matemático apresentado a seguir para o ODP guilhotinado é o de Lodi *et al.* (2004). Antes de apresentá-lo, devem ser assumidas as seguintes observações, feitas sobre uma solução qualquer:

a) O primeiro item alocado em cada faixa (mais à esquerda) é o de maior altura;

b) A primeira faixa do objeto (mais baixa) é a mais alta;

c) Os itens são ordenados de forma decrescente em relação à altura, ou seja:

$$h_1 \geq h_2 \geq \ldots \geq h_n.$$

Como conseqüência de (c), tem-se que a altura de cada faixa corresponde à altura h_i do item i que a inicializa (primeiro item alocado). Assim, sendo n o número de faixas formadas para alocar todos os itens demandados, pode-se definir a seguinte variável de decisão:

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

Deve-se ressaltar ainda, que, devido às observações (a) e (c), somente os itens *j,* tal que $j > i$, podem ser alocados na faixa. Essa condição se deve ao fato de que, se um dado item *j*, tal que $j = i$, inicializa a faixa *i*, ele não pode ser atribuído novamente a essa faixa. Assim sendo, é definida a seguinte variável binária:

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

O modelo de PLI é apresentado a seguir:

Minimizar $$\sum_{i=1}^{n} h_i . y_i \qquad (3.1)$$

sujeito a: $$\sum_{i=1}^{j-1} x_{ij} + y_j = 1 \qquad j = 1,...,n; \qquad (3.2)$$

$$\sum_{j=i+1}^{n} w_j x_{ij} \le (W - w_i) y_i \qquad i = 1,...,(n-1); \qquad (3.3)$$

$$x_{ij} \in \{0,1\} \qquad i = 1,...,(n-1);\ j > i$$
$$y_i \in \{0,1\} \qquad i = 1,...,n; \qquad (3.4)$$

A função objetivo, representada pela equação (3.1), tem, como critério de otimização, a minimização da altura utilizada do objeto. A restrição (3.2) garante que cada item é alocado apenas uma vez, ou seja, ou o item inicializa a faixa ou está numa faixa inicializada por outro item. A restrição (3.3) garante que, em cada faixa, a largura do objeto não será ultrapassada. A restrição (3.4) define o tipo das variáveis do problema.

4. Técnicas de Encaixe

Neste trabalho, a solução do problema é representada por uma seqüência de itens, que devem ser encaixados segundo essa ordem. Dependendo da técnica de encaixe utilizada, é possível encontrar diferentes alturas para uma mesma solução. Desse modo, é necessário encontrar uma forma eficiente de encaixar os itens, de forma a obter a menor altura para a solução. Dentre os algoritmos aproximados desenvolvidos para tratar do encaixe de itens, em problemas com duas ou mais dimensões, existem os chamados algoritmos de níveis, nos quais os itens são empacotados em níveis (faixas), da esquerda para a direita, sendo que o início de um novo nível coincide com o topo do item mais alto do nível anterior. Estes métodos apresentam a vantagem de serem rápidos 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*), *Best Fit Decreasing Height* (*BFDH*). Neste trabalho foram utilizados e comparados, para o encaixe dos itens, os algoritmos *First-Fit* e *Best Fit,* os quais são descritos a seguir.

Nos dois algoritmos, a primeira faixa se forma com a inserção dos itens na ordem em que aparecem na solução, a partir do canto inferior esquerdo do objeto, até que a largura do objeto se torne insuficiente. No *First-Fit,* antes dos próximos itens serem inseridos, são pesquisadas as larguras disponíveis de cada faixa já formada, para avaliar a possibilidade de inseri-los. Caso não seja possível, é formada uma nova faixa. Assim, os itens são alocados na primeira faixa em que couberem, sem ultrapassar a largura *W* do objeto. Portanto, uma nova faixa somente é formada quando o item não couber em nenhuma outra faixa que está sendo utilizada (BLAZEWICZ *et al.*, 1989). No *Best-Fit*, antes do próximo item ser inserido, são verificadas as áreas restantes de cada faixa já formada, sendo o item inserido naquela onde o espaço for mais bem utilizado, ou seja, onde a área restante após sua inserção seja menor. Se não for possível encaixá-lo, é formada uma nova faixa (BLAZEWICZ *et al*., 1989). Os pseudocódigos dos algoritmos são apresentados a seguir.

```
Procedimento First-Fit ():
1  W ← largura do objeto;
2  h_i ← altura do item i, i = 1,...,n;
3  w_i ← largura do item i, i = 1,...,n;
4  Wdf_j ← W;
   {Largura disponível na faixa j, j =1,...,m}
5  j ← 1;
6  para i =1 até i = n faça
7     enquanto (w_i > Wdf_j) faça
8        j ← j + 1;
9     fim-enquanto
1     Wdf_j ← Wdf_j - w_i;{Insere o item i na faixa j}
11 fim-para;
```

Figura 2 – Algoritmo *First-Fit*

```
Procedimento Best-Fit ():
1  W ← largura do objeto;
2  h_i ← altura do item i, i = 1,...,n;
3  w_i ← largura do item i, i = 1,...,n;
4  hf_j ← 0 {Altura da faixa j, j = 1,...,m}
5  wdf_j ← W; {Largura disponível na faixa j, j = 1,...,m}
6  j ← 1;
7  para i =1 até i = n faça
8  melhor_rel ← ∞;  {Melhor razão entre as áreas}
9  A_i ← h_i * w_i;  {Área do item i}
10    para i =1 até i = n faça
11       Adf_j ← hf_j * wdf_j;  {Área disponível na faixa j}
12       rel ← ∞;  {Razão entre as áreas}
13       se (w_i≤wdf_j) e (h_i≤hf_j ) e (Rel<melhor_rel) então
14          melhor_rel ← Rel;
15          k ← j;
16       fim-se
17    fim-para
18    wdf_k ← Wdf_k - w_i;  {Insere o item i na faixa k}
19 fim-para;
fim Best-Fit
```

Figura 3 – Algoritmo *Best-Fit*

5. Algoritmo Evolutivo aplicado ao ODP guilhotinado

Os Algoritmos Genéticos (AG) foram propostos inicialmente por Holland (1975). Trata-se de uma metaheurística baseada nos mecanismos de seleção natural e da genética, na qual cada indivíduo representa uma solução factível do espaço de busca de soluções. O AG clássico inicia-se com uma população inicial de cromossomos gerada aleatoriamente. Durante o processo evolutivo, esta população é avaliada e cada cromossomo recebe uma aptidão, que reflete a qualidade da solução que ele representa. Em geral, os cromossomos mais aptos são selecionados e os menos aptos são descartados. Os membros selecionados podem sofrer modificações em suas características fundamentais, através dos operadores de *crossover* e mutação, gerando descendentes para a próxima geração. Este processo é repetido até que uma solução satisfatória seja encontrada.

O desempenho dos AG em sua forma original, no entanto, não tem se mostrado satisfatório para muitos problemas de otimização. Em razão disto, diversos pesquisadores têm proposto modificações nos AG convencionais, incorporando a estes técnicas de busca local, ou incluindo conceitos de outras metaheurísticas, tais como: *Simulated Annealing*, *Tabu Search (busca tabu)*, *Iterated Local Search*, entre outras, obtendo, assim, versões conhecidas como Algoritmos Meméticos (MOSCATO, 1989), ou simplesmente Algoritmos Evolutivos (AE). Com isso, procura-se um algoritmo evolutivo que possua as características básicas dos AG, mas que incorpore mecanismos adicionais, de forma a torná-lo mais especializado para determinadas aplicações. Neste trabalho propõe-se um Algoritmo Evolutivo Híbrido (AEH) que possui as seguintes particularidades em relação a um AG básico:

a) A população inicial não é gerada maneira aleatória. Os indivíduos são gerados pela metaheurística *Iterated Local Search* (ILS).
b) Em vez do operador de cruzamento (*crossover*) tradicional, utilizou-se o operador *Order Crossover* (*OX*).
c) Um operador de busca local é aplicado nos melhores indivíduos da população a cada geração, realizando, assim, uma busca intensiva na vizinhança das melhores soluções encontradas pelo AEH, com o objetivo de atingir um número maior de soluções com nível de aptidão mais alto.

A seguir são descritas as etapas do algoritmo evolutivo proposto.

5.1 Representação de uma solução

Neste problema, um cromossomo representa a seqüência em que os itens serão inseridos no objeto, ou seja, é um vetor de n valores inteiros, sendo n o número de itens da solução. Assim, cada gene representa um item, e sua posição no vetor indica a seqüência utilizada pelo algoritmo de encaixe.

5.2 Operadores genéticos

5.2.1 Operadores do Tipo *Crossover*

O cruzamento é o principal mecanismo do AG para explorar o espaço de busca, através da criação de novos pontos, obtidos pela troca de informação genética entre os indivíduos.

Neste trabalho, deve-se evitar que uma operação provoque a repetição de uma variável num mesmo individuo. Por isso, para realizar o cruzamento *(crossover*), foi utilizado o operador *Order Crossover* (OX), aplicado com uma dada probabilidade. Este operador gera filhos a partir da escolha de uma seqüência parcial dos genes de um dos pais, preservando a ordem relativa dos genes do outro pai, conforme descrito a seguir.

São escolhidos aleatoriamente dois indivíduos p_1 e p_2 (pais). Os pais são divididos em três partes, por dois pontos de corte escolhidos aleatoriamente. Os filhos f_1 e f_2 herdam a parte do meio de seus respectivos pais, p_1 e p_2. Em seguida, partindo do segundo corte de um pai (por exemplo, p_2), copiam-se, em f_1, os elementos de p_2, na ordem em que aparecem, sendo removidos os elementos contidos entre os dois cortes do outro pai (no caso, p_1), conforme a figura a seguir:

$p_1 = (1\ \ 6\ \ 8\ |\ 3\ \ 2\ \ 5\ |\ 4\ \ 7)$ → $f_1 = (7\ \ 1\ \ 4\ |\ 3\ \ 2\ \ 5\ |\ 8\ \ 6)$

$p_2 = (6\ \ 2\ \ \mathbf{7}\ |\ 1\ \ 4\ \ 3\ |\ 5\ \ 8)$ → $f_2 = (6\ \ 8\ \ 4\ |\ 1\ \ 4\ \ 3\ |\ 7\ \ 1)$

Figura 4 – *Crossover OX*

5.2.1. Operadores do Tipo Mutação

A mutação é um operador genético que tem a função de introduzir características na população, ou mesmo recuperar aquelas que se perderam em operações, como por exemplo, o *crossover*. Além disso, a mutação é usada para prevenir a convergência prematura para ótimos locais, através da amostragem de novos pontos no espaço de busca. Este operador é aplicado, numa dada probabilidade, a cada descendente individualmente, após a aplicação do operador de *crossover*. Neste trabalho, foram aplicados dois operadores de mutação, escolhidos aleatoriamente, em indivíduos também selecionados de forma aleatória. São eles:

- *Operador troca*: consiste em trocar a ordem de encaixe de dois itens, escolhidos aleatoriamente, conforme ilustrado na Fig. 5. Nesse operador, dado um conjunto de n itens, existem $n(n\text{-}1)/2$ indivíduos mutantes possíveis.

- *Operador realocação*: consiste em realocar um item da posição i para a posição j, sendo i e j escolhidos aleatoriamente. Nesse operador, dado um conjunto de n itens, existem $(n\text{-}1)^2$ indivíduos mutantes possíveis. Por exemplo, sendo $i = 2$ e $j = 6$ (realocar o item da segunda posição para a sexta posição), obtém-se, a partir do individuo s, o indivíduo mutante s' mostrado na Fig.6. Nessa estrutura são permitidos movimentos para posições sucessoras (como no exemplo acima) ou antecessoras à posição em que o item se encontra na seqüência do vetor.

$s = (1\ \ 6\ \ 8\ \ 3\ \ 2\ \ 5\ \ 4\ \ 7)$
$s' = (1\ \ 6\ \ 5\ \ 3\ \ 2\ \ 8\ \ 4\ \ 7)$

Figura 5 – Operador de mutação troca

$s = (1\ \ 6\ \ 8\ \ 3\ \ 2\ \ 5\ \ 4\ \ 7)$
$s' = (1\ \ 8\ \ 3\ \ 2\ \ 5\ \ 6\ \ 4\ \ 7)$

Figura 6 – Operador de mutação realocação

5.3 Função de custo dos cromossomos

A função de custo é a altura utilizada do objeto em determinada solução, obtida após a execução de um procedimento de encaixe dos itens, ou seja, a solução é avaliada pela própria função objetivo, dada pela expressão (3.1) do modelo de PLI.

5.4 Função de aptidão de um cromossomo

A função de aptidão avalia a possibilidade de sobrevivência de cada indivíduo gerado pelo AG. A partir desta avaliação, é possível descobrir o grau de adaptação dos indivíduos e eliminar aqueles menos adaptados. Devido à utilização do mecanismo da roleta, para selecionar os indivíduos da próxima geração, não foi possível usar a função de custo como função de aptidão, pois este trabalho trata de um problema de minimização.

Assim, para obter o valor da função de aptidão a partir da função objetivo, foi utilizado um método de escalamento linear, que utiliza uma função linear para essa conversão. Neste trabalho, a função utilizada é dada por uma equação que corresponde a equação da reta que passa pelos pontos (*fmin,* α) e (*fmax,*β). Considerando $\beta = 0$, temos:,

$$f_{aptidão}(s) = \alpha \times (fo_{max} - fo(s)) / (fo_{max} - fo_{min})$$

em que:

$fo(s)$ = valor da função objetivo da solução s;
$f_{aptidão}(s)$ = valor da função de aptidão da solução s;
fo_{max} = valor máximo de função objetivo encontrado na população;
fo_{min} = valor mínimo de função objetivo encontrado na população;
α = função de aptidão atribuída ao indivíduo com $fo_{min.}$;
β = função de aptidão atribuída ao indivíduo com fo_{max}.

5.5 Seleção dos indivíduos para a próxima geração

Um mecanismo de seleção para as próximas gerações, muito utilizado em AG clássicos, é o chamado *roulette wheel*, ou roleta (HOLLAND, 1975). O método da roleta atribui a cada indivíduo de uma população, uma probabilidade de passar para a próxima geração, proporcional a sua função de aptidão, calculada em relação à somatória das funções de aptidão de todos os indivíduos da população. Assim, quanto maior a função de aptidão de um indivíduo, maior a sua probabilidade de passar para a próxima geração.

Nesse método, todas as soluções têm chance de serem escolhidas, permitindo assim que o algoritmo escape de ótimos locais. No entanto, a roleta também pode fazer com que o melhor indivíduo da população seja perdido, ou seja, não passe para a próxima geração. Uma alternativa para contornar esse problema é manter sempre o melhor indivíduo da geração atual na geração seguinte, sendo essa estratégia conhecida como ***seleção elitista***. Neste trabalho, a estratégia elitista utilizada é a seguinte: as duas melhores soluções da população anterior são refinadas e sempre incluídas na nova população. A utilização do elitismo nos AG tem mostrado que a estratégia é usualmente vantajosa (LEVINE, 1993).

5.6 Geração da população inicial

Para gerar uma população inicial são utilizados vários procedimentos, que podem ser aleatórios ou heurísticos. Gerar soluções aleatórias diversifica o espaço de soluções, mas, geralmente, cria soluções de baixa qualidade, que exigem um tempo de processamento muito elevado para ser melhorada. Por outro lado, soluções gulosas, apesar de serem de boa qualidade, apresentam baixa diversidade. Como alternativa aos métodos totalmente aleatórios ou totalmente gulosos, vários autores sugerem a utilização de heurísticas para gerar a população inicial.

Neste trabalho, a população inicial é gerada pelo algoritmo *Iterated Local Search* (ILS). A metaheurística ILS é baseada na idéia de que se pode melhorar um procedimento de busca local com a geração de novas soluções de partida obtidas com a aplicação de perturbações sobre a solução ótima. É um método que não explora o espaço completo das soluções, mas um pequeno subespaço composto por soluções que são ótimos locais, ou ainda, soluções que sofreram uma busca local, mas sem a garantia de otimalidade de determinados procedimentos de otimização.

O algoritmo ILS inicia-se com a geração de uma solução aleatória, que em seguida sofre uma busca local. A seguir, é aplicada uma perturbação nessa solução. Tais perturbações são divididas em níveis. O primeiro nível de perturbação consiste em aplicar uma vez um determinado movimento sobre a solução. No segundo nível esse movimento é aplicado duas vezes sobre a solução, e assim sucessivamente. Após a perturbação, a solução passa por uma busca local, sendo avaliada em seguida. Se esta solução for pior que a anterior, o movimento é

desfeito e o nível de perturbação cresce. Caso contrário, a solução encontrada passa a ser a solução corrente, e o nível de perturbação atual é aplicado. A solução corrente é a melhor solução encontrada até então. O procedimento pára após um certo número de iterações.

Como se pode perceber, a perturbação deve ser forte o suficiente para garantir a exploração de diferentes soluções e fraca o bastante para que não ocorra um reinício aleatório. A eficiência do ILS é baseada no conjunto de amostragem de ótimos locais e na escolha do método de busca local, das perturbações do critério de aceitação. O pseudocódigo do método é apresentado na Fig.7. Os seguintes procedimentos são utilizados neste trabalho nas etapas que compõem o ILS:

a) Geração da solução inicial: solução inicial gerada aleatoriamente;
b) Busca Local: Método Randômico de Descida, descrito na próxima seção;
c) Perturbação: em cada nível, são escolhidos aleatoriamente e aplicados à solução os movimentos de troca ou realocação de itens, por *ILS_max* vezes sem melhora em um dado nível;
d) Critério de aceitação: valor da função de avaliação.

5.7 Método de Busca Local proposto

Neste trabalho, o procedimento de busca local incorporado ao AEH é o Método Randômico de Descida (MRD), que é uma variação do Método de Descida (*Best Improvement Method)*. Esse método parte de uma solução inicial qualquer e, a cada iteração, escolhe aleatoriamente uma estrutura de vizinhança, que é aplicada à solução corrente. Em seguida, analisa um vizinho qualquer, movendo-se somente para o vizinho que representar uma melhora estrita no valor atual da função de avaliação. O método é interrompido após um número fixo de iterações sem melhora no valor da melhor solução obtida até então, neste trabalho dado pelo valor *MRD_max*. As estruturas de vizinhança são os movimentos de troca e realocação de itens. A Fig. 8 mostra o pseudocódigo do algoritmo.

```
Procedimento ILS(·)
1  s_o ← ConstroiSolucaoAleatoria(·);
2  s ← Descida_Randomica(s_o);
3  nivel =1; iter =1;
4   enquanto (nivel  < n_niveis) faça
5        enquanto (iter < ILS_max ) faça
6              s' ← Perturbação(s, nivel);
7              s" ← Descida_Randomica(s');
8             se (f(s") < f(s) ) então
9                    s ← s" ;
10                   nivel ← 1;
11                   iter ← 0;
12            senão
13            iter ← iter +1;
14       fim-enquanto;
15       nivel ← nivel +1;
16       iter ← 0;
17  fim-enquanto;
18  Retorne s;
fim ILS();
```

Figura 7 - Pseudocódigo ILS

```
Procedimento Descida_Randomica (s, MRD_max):
1            Iter = 0
2            enquanto Iter < MRD_max faça
3                 Iter = Iter + 1
4                 s' ← GeraVizinhoAleatoriamente(s)
5                 se ( f(s') < f (s) ) então
6                       Iter = 0
7                       s ← s'
8                 fim-se;
9            fim-enquanto;
10           Retorne s
fim Descida_Randomica;
```

Figura 8 - Algoritmo de Busca Local

6. Algoritmo Evolutivo Híbrido aplicado ao ODP guilhotinado

O método proposto, conforme visto nas seções anteriores, incorpora a metaheurística *ILS* e a técnica de busca local MRD (Método Randômico de Descida) (Fig. 7 e 8), ao AG. A população inicial do AEH é gerada de acordo com a metodologia descrita na seção 5.6. A avaliação da população, feita após a aplicação das técnicas de encaixe, é feita por uma função escala, descrita na seção 5.4. O cruzamento utiliza o operador *OX* e a mutação utiliza dois

movimentos (troca e realocação). A estratégia de elitismo utilizada consiste em refinar (aplicar MRD) os dois melhores indivíduos da geração e mantê-los na próxima geração. Isto é feito mesmo na ausência de melhora em relação à geração anterior, porque não há garantia de que essas soluções sejam ótimos locais, já que nem toda a vizinhança é considerada no MRD. O critério de parada do algoritmo é o número de gerações. A Fig. 9 apresenta o pseudocódigo do método proposto.

```
procedimento AEH
1.    Inicializar a população ;
2.    t ← 0;   {Contador do número de gerações}
3.    Gerar uma população inicial P(t);
4.    Avaliar P(t);
5.    enquanto (t < Gerações_Max)faça
6.        Cruzar selecionados;
7.        Mutar resultantes;
8.        Avaliar P(t);
9.        Selecionar P(t +1) a partir de P(t), usando a estratégia de elitismo;
10.       t ← t + 1;
11.   fim-enquanto;
12.   Retorne o melhor indivíduo;
fim AEH;
```

Figura 9 - Método Evolutivo Híbrido proposto

7. Resultados e Análise

O Algoritmo Evolutivo Híbrido proposto foi implementado na linguagem C, e compilado em C++ Builder 5.0. Os testes computacionais foram feitos em um computador AMD Atlon XP 64 Bits 4000+ (aproximadamente 2.11 GHz), com 1 GB de memória RAM, sob plataforma Windows XP.

Para avaliar o AEH proposto, foram utilizados os problemas-teste de Hopper e Turton (2001), disponíveis na OR-Library (BEASLEY, 1990). Estes problemas-teste foram construídos de forma a não apresentarem perda quando solucionados de maneira não-guilhotinada e com a rotação dos itens permitida. Eles estão divididos em sete categorias, sendo que cada categoria contém três problemas. Cada categoria dos problemas-teste apresenta o número de itens, a largura do objeto e a altura ótima, conforme a Tabela 1. Para cada problema são dadas a largura e a altura dos itens.

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

Categoria	*Número de itens (n)*			*Largura do objeto*	*Altura ótima*
	P_1	P_2	P_3		
C1	16	17	16	20	20
C2	25	25	25	40	15
C3	28	29	28	60	30
C4	49	49	49	60	60
C5	73	73	73	60	90
C6	97	97	97	80	120
C7	196	197	196	160	240

A seguir, são apresentados os resultados obtidos, a partir dos dados dos problemas-teste, para o caso envolvendo o ODP guilhotinado, nos termos tratados no presente trabalho, aplicando-se o otimizador Lingo, versão 8.0, e o AE Híbrido proposto, em suas duas versões, que utilizam os dois algoritmos de encaixe (*First-Fit* e *Best-Fit*), aqui chamadas de AE-*FF* e AE-*BF*. É importante ressaltar que estes resultados, na medida em que tratam do ODP guilhotinado, têm soluções com alturas maiores que as alturas ótimas encontradas nos problemas-teste, solucionados de forma não-guilhotinada e com a rotação dos itens permitida, segundo sua formulação original.

Os resultados listados em Temponi (2007), obtidos pela resolução do modelo matemático apresentado na Seção 3, pelo *software* de otimização LINGO, versão 8.0, são apresentados na Tabela 2. Foram resolvidos os problemas-teste das sete categorias. Nesta tabela, a primeira coluna apresenta o problema-teste, a segunda coluna apresenta a solução encontrada e a última coluna apresenta o tempo computacional gasto para se chegar à solução. Considerou-se um tempo limite de 30 horas para obtenção de uma solução ótima.

Tabela 2: Resultados do Modelo Matemático

Problema-teste	*Itens*	*Solução*	*Tempo (seg)*	*Problema-teste*	*Itens*	*Solução*	*Tempo (seg)*
C1P1	16	27*	< 1	C5P1	73	102	30 hs
C1P2	17	29*	< 1	C5P2	73	110	
C1P3	16	23*	< 1	C5P3	73	106	
C2P1	25	20*	< 1	C6P1	97	145	
C2P2	25	34*	4	C6P2	97	153	
C2P3	25	23*	1	C6P3	97	145	
C3P1	28	40*	6	C7P1	196	300	
C3P2	29	42*	1	C7P2	197	312	
C3P3	28	43*	14	C7P3	196	326	
C4P1	49	74*	19420				
C4P2	49	75*	5736				
C4P3	49	81*	160				

* Solução ótima

Como pode ser observado na Tabela 2, o modelo de programação matemático obteve a solução ótima para todos os problemas-teste com até 49 itens. Acima de 73 itens, o otimizador não foi capaz de encontrar soluções ótimas dentro do tempo limite dado, sendo apresentadas, portanto, as soluções obtidas com 30 horas de execução.

Para testar o AEH, foram resolvidos problemas-teste com 16 a 196 itens, totalizando 21 problemas. Cada problema foi resolvido 10 vezes pelo método proposto. Os parâmetros do AEH, obtidos experimentalmente, são apresentados na Tabela 3.

Tabela 3: Parâmetros do AEH

Parâmetros			*Valores*
Número de indivíduos	*Categoria Problema*	C1, C2, C3, C4, C5	6
		C6, C7	10
Número de gerações		C1, C2, C3, C4, C5	50
		C6, C7	100
Parâmetro α da função de escalamento linear			100
Probabilidade de *crossover* (p_c)			0,85
Probabilidade de mutação (p_m)			0,05
Número máximo de iterações sem melhora do MRD (*MRD_max*)			200
Número máximo de iterações sem melhora do ILS (*ILS_max*)			20

A Tabela 4 apresenta os resultados dos experimentos com o AEH. Os resultados obtidos com os problemas-teste são comparados com os resultados ótimos obtidos pelo modelo matemático. O erro relativo (*%gap*), apresentado nas sétima e oitava colunas, é calculado pela expressão:

$$gap = ((r_{médio} - r_{melhor}) / r_{melhor}) \times 100$$

em que $r_{médio}$ é o resultado médio obtido pela aplicação do AEH e r_{melhor} é o resultado obtido pelo otimizador para cada problema.

Pela Tabela 4, observa-se que o desvio da solução média produzida pelo método AEH-FF nunca ultrapassou 7,55%. Os tempos de processamento variaram, para o AEH-*FF*, de 2,06 a 47,75 segundos, e, para o AEH-*BF,* de 2,09 a 743,98 segundos, em média. Observa-se também que os dois algoritmos sempre alcançaram o resultado ótimo conhecido para os problemas-teste de 16 a 49 itens. Além disso, comparando-se os tempos de processamento (Tabelas 2 e 4), nota-se

que os tempos gastos pelos métodos propostos foram, geralmente, muito inferiores àqueles demandados pelo otimizador aplicado ao modelo de PLI.

Destaca-se que o método AEH-*BF* proposto encontrou, em 9 dos 21 problemas-teste, resultados melhores do que os encontrados pelo método exato após 30 horas de execução, sendo que, para o problema C7P3, obteve-se uma melhoria média de 4,91%. O método AEH-*FF* encontrou, em 4 dos 21 problemas-teste, resultados melhores do que os encontrados pelo método exato após 30 horas de execução.

Tabela 4: Resultados do AEH

Problema Teste	*Nº de itens*	*Solução método exato*	*Melhor solução*		*Solução média*		*Média dos gap's (%) da solução média*		*Tempo Médio (s)*	
			AE– FF	*AE– BF*	*AE –FF*	*AE –BF*	*AE– FF*	*AE– BF*	*AE-FF*	*AE-BF*
C1P1	16	27*	27*	27*	27	27	0	0	2,06	2,09
C1P2	17	29*	29*	29*	29	29	0	0	2,25	2,14
C1P3	16	23*	23*	23*	23	23	0	0	2,78	2,13
C2P1	25	20*	20*	20*	20	20	0	0	2,96	3,45
C2P2	25	34*	34*	34*	34	34	0	0	3,28	3,49
C2P3	25	23*	23*	23*	23	23	0	0	3,46	3,44
C3P1	28	40*	40*	40*	40	40	0	0	4,29	6,85
C3P2	29	42*	42*	42*	42	42	0	0	4,03	5,50
C3P3	28	43*	43*	43*	43	43	0	0	3,95	6,04
C4P1	49	74*	74*	74*	76	74	2,70	0	7,21	20,86
C4P2	49	75*	75*	75*	76	75	1,33	0	7,00	18,21
C4P3	49	81*	81*	81*	82	81	1,23	0	6,53	18,56
C5P1	73	102♣	104	101**	106	102	3,92	0	11,79	34,66
C5P2	73	110♣	108**	106**	110	106	0	-3,64	11,35	38,75
C5P3	73	106♣	111	107**	114	106	7,55	0	11,89	36,50
C6P1	97	145♣	144**	137**	149	140	2,76	-3,45	17,40	116,25
C6P2	97	153♣	152**	147**	155	148	1,31	-3,27	15,41	117,62
C6P3	97	145♣	147	143**	151	144	4,14	-0,69	16,86	111,18
C7P1	196	300♣	311	285**	320	294	6,67	-2,00	43,50	690,24
C7P2	197	312♣	314	299**	326	303	4,49	-2,88	45,48	743,98
C7P3	196	326♣	317**	304**	332	310	1,84	-4,91	47,75	717,27

* Solução ótima.
** Solução melhor do que a encontrada pelo método exato após 30 horas de processamento.
♣ Melhor solução obtida após 30 horas de processamento.

8. Conclusões

O objetivo principal deste trabalho foi elaborar um Algoritmo Evolutivo Híbrido para a resolução do ODP guilhotinado e comparar o desempenho de dois algoritmos aproximados (*First-Fit* e *Best-Fit*) utilizados com sub-rotinas do método. Para tal, propôs-se incorporar a um Algoritmo Genético clássico, a metaheurística *Iterated Local Search,* para a construção da população de indivíduos, e um operador de busca local, aplicado aos melhores indivíduos de cada geração.

Nos experimentos computacionais realizados, mostrou-se a importância do uso de uma heurística que gere soluções de boa qualidade para inicializar um AG e também a necessidade de incorporar ao AG, um mecanismo eficiente de busca local. Nos experimentos realizados, observou-se que o trabalho computacional adicional exigido pela busca local pode ser compensado pela redução do número de indivíduos e do número de iterações do algoritmo.

O método heurístico proposto foi aplicado nos problemas-teste de Hopper e Turton (2001), que tratam o ODP de forma não-guilhotinada. Mostrou-se nos experimentos que o AEH-*FF* e o AEH-*BF* propostos alcançaram as soluções ótimas dos problemas com até 49 itens. Em nove problemas-teste, com mais de 49 itens, a melhor solução encontrada pelo AEH-*BF* superou as soluções obtidas pelo método exato. Em quatro problemas-teste, com mais de 49 itens, o AEH-

FF superou as soluções obtidas pelo método exato. O AEH-*FF* apresentou resultados bastante inferiores aos do AEH-*BF* nos problemas-teste com mais de 49 itens, sendo demonstrada, assim, a superioridade do algoritmo aproximado *Best-Fit* em relação ao *First-Fit*. O tempo de processamento dos dois métodos foi pequeno se comparado ao tempo gasto pelo método exato, sendo que a diferença entre os tempos médios dos métodos AEH-*FF* e AEH-*BF* se deve ao fato do segundo ser um algoritmo mais sofisticado. Estes resultados empíricos mostram o potencial da técnica proposta e a possibilidade da utilização desta em situações práticas.

Referências

Beasley, J.E. (1990), OR-library: distribution test problems by electronic mail, *Journal of the Operational Research Society*, 41, 1069-1072.

Berkey, J.O. e Wang, P.Y. (1987), Two dimensional finite bin packing algorithms, *Journal of the Operational Research Society*, 38, 423-429.

Blazewicz, J., Drozdowaki, M. e Soniewicki, B. (1989), Two-dimensional cutting problem basic complexity results and algorithms for irregular shapes, *Foundations of Control Engineering*, 14, 137-160.

Coffman, E. G., Garey, M. R. e Johnson, D. S., Approximation algorithm for bin packing: an updated survey, em Ausiello, G., Lucertini, M. e Serafini, P. (Eds.), *Algorithm Design for Computer Systems Design*, Springer-Verlag, New York, EUA, 49-106, 1984.

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

Hochbaum, D. S. e Wolfgang, M. (1985), Approximation schemes for covering and packing problems in image processing and VLSI, *Journal of the Association for Computing Machinery*, 32, 130-136.

Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, EUA, 1975.

Hopper, E. e Turton, B.C.H. (2001), An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, *European Journal of Operational Research*, 128, 34-57.

Kroger, B. (1995), Guillotineable binpacking: a genetic approach, *European Journal of Operational Research,* 84, 645-661.

Lesh, N., Marks, J., McMahon, A. e Mitzenmacher, M. (2004)**,** Exhaustive approaches to 2D rectangular perfect packings, *Information Processing Letters*, 90, 7-14.

Leung, J., Tam, T., Wong, C.S., Young, G. e Chin, F. (1990), Packing squares into square, *Journal of Parallel and Distributed Computing*, 10, 271-275.

Levine, D. M. (1993), A genetic algorithm for the set partitioning problem, *Proceedings of the 5th International Conference on GA's*, 481-487.

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

Lodi, A., Martello, S. e Vigo, D. (1999), Heuristic and metaheuristic approaches for a class of two-dimensional binpacking problems, *INFORMS Journal on Computing*, 11, 345-357.

Lodi, A., Martello, S. e Monaci, M. (2002), Two-dimensional packing problems: a survey, *European Journal of Operational Research*, 141, 241-252.

Lodi, A.; Martelo, S. e Vigo, D. (2004), Models and bounds for two-dimensional level packing problems, *Journal of Combinatorial Optimization*, 8, 363–379.

Moscato, P., On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms, *Technical Report Caltech Concurrent Computation Program*, Report 826, California Institute of Technology, 1989.

Temponi, E.C.C.,. Uma proposta de resolução do problema de corte bidimensional via abordagem metaheurística, *Dissertação de Mestrado,* Belo Horizonte, CEFET-MG, 2007.

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

Wäscher, G., Haußner, H. e Schuann, H. (2006), An improved typology of cutting and packing problems, *European Journal of Operations Research*, 4, 44-454.