SPOLM 2008

ISSN 2175-6295 Rio de Janeiro- Brasil, 05 e 06 de agosto de 2008.

UM ALGORITMO MEMÉTICO APLICADO À SOLUÇÃO DO PROBLEMA DE CORTE BIDIMENSIONAL GUILHOTINADO

Mariana Silva Faleiro de Andrade
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas 7675 – Nova Gameleira – 30.510-000 – Belo Horizonte – MG – Brasil
marifaleiro@yahoo.com.br

Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas 7675 – Nova Gameleira – 30.510-000 – Belo Horizonte – MG – Brasil
sergio@dppg.cefetmg.br

Elias Carlos Corrêa Temponi
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas 7675 – Nova Gameleira – 30.510-000 – Belo Horizonte – MG – Brasil
elias@uai.com.br

RESUMO

Este trabalho trata do problema de corte com dimensão aberta guilhotinado, ou seja, um problema de corte bidimensional em que uma das dimensões do objeto é variável. Propõe-se, para sua resolução, o algoritmo aproximado *Best-Fit* e a metaheurística Algoritmo Memético (AM). O AM utiliza, como operadores de busca local, as técnicas conhecidas como Método da Descida (MD) e o Método Randômico de Descida (MRD). Experimentos computacionais realizados sobre um conjunto de problemas-teste da literatura indicam que o método proposto consegue alcançar o valor ótimo de todos os 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-Chaves: Problema de corte bidimensional; Algoritmos Meméticos; Busca Local.

Abstract

This paper deals with the guilhotined Open Dimensional Problem (ODP). Given a set of restangular 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 Memetic Algorithm is proposed to solve the problem. In the algorithm, we use the local search operators Descent Method and Random Descent Method, as well the approximate algorithm *Best-Fit* 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 instances (16 to 49 items) and yield small gaps in instances with 73 to 197 items.
Keywords: Bidimensional Cutting Problem; Memetic Algorithms; 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 fez surgir uma classe de problemas, denominada Problemas de Corte e Empacotamento (PCE).

Os PCE são problemas de otimização combinatória que envolvem corte e empacotamento de itens, sendo que estas atividades, para fins de formulação e resolução, podem ser consideradas análogas. O PCE consiste em determinar o melhor arranjo, sobre um objeto, de um conjunto de itens, com diversas dimensões, 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 [24]. Como o próprio nome sugere, trata-se de um problema que possui dimensões variáveis, sendo, portanto, definido para duas ou mais dimensões. Este problema, usualmente, trata de um único objeto. Além da classificação quanto a dimensionalidade do problema, também são definidos pela tipologia de [24] o tipo e a forma dos itens (idênticos, pouco heterogêneos, muito heterogêneos/ regulares ou irregulares). O problema tratado neste trabalho tem as seguintes características: trata de itens regulares (retangulares) e muito heterogêneos e de um objeto com duas dimensões, sendo uma delas fixa.

Assim, para o caso bidimensional, o ODP 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 é encontrado, por exemplo, em fábricas de papel, aço e tecido, além de outras que realizam o corte de bobinas ou rolos.

Por se tratar de um problema de corte bidimensional, são acrescidas outras classificações, devidas a restrições impostas pelos equipamentos e pelo processo de corte. Uma dessas restrições é com relação ao tipo de corte, que pode ser guilhotinado, quando se estender de um lado ao outro do retângulo, 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, normalmente, uma mudança na direção do corte. No caso, por exemplo, de ocorrer apenas uma mudança na direção, o corte é dito ser 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) [2], 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 [9,16,17].

No caso do ODP, vários autores já mostraram que se trata de um problema NP-Difícil ([7,13]). Devido a esse fato, e também ao enorme número de aplicações práticas do problema, como, por exemplo, no corte de bobinas de papel, de tecido, de bobinas de alumínio, entre outras, são encontrados vários trabalhos na literatura que têm seu foco em técnicas para a resolução do ODP.

Em [6] é apresentado um método exato para o ODP guilhotinado e com rotações ortogonais, baseado num procedimento *branch-and-bound*, que soluciona problemas de pequeno porte. Em [12] os autores também apresentam 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 algoritmos populacionais baseados em Algoritmos Genéticos (AG). Dentre eles, podemos citar [23], que tratam de um ODP não-guilhotinado através de uma heurística de encaixe de

itens, que transforma o ODP em um simples problema de permutação, que então é solucionado pelo AG. Métodos baseados em AG também são apresentados em [11, 15]. Uma revisão sobre a utilização de AG na resolução do OPD é feita em [11].

No presente trabalho é utilizado um modelo de programação linear inteira proposto por [18] para representar o ODP guilhotinado. Para solucionar o problema, é proposto um Algoritmo Memético (AM). O AM é um algoritmo populacional que surgiu do desenvolvimento de algoritmos genéticos híbridos. Além disso, é utilizado um algoritmo aproximado específico (Algoritmo *Best-Fit*) para o problema de corte e empacotamento, como sub-rotina do método.

Os resultados obtidos pelo método proposto são comparados com aqueles obtidos pela resolução do modelo de PLI usando os resultados obtidos pelo otimizador LINGO, versão 8.0, apresentados em [22].

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 é apresentado o algoritmo aproximado utilizado no trabalho, enquanto na seção 5 é desenvolvida a metodologia heurística de resolução do ODP guilhotinado. Na seção 6 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, que 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.

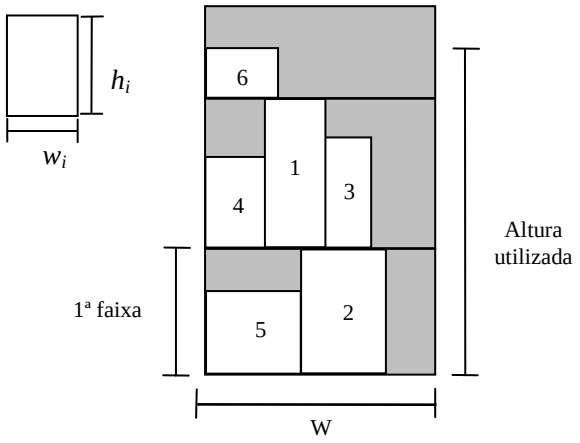

Figura 1 – *Open dimensional problem* guilhotinado 2-estágios.

3. MODELAGEM MATEMÁTICA

O modelo matemático para o ODP guilhotinado, apresentado a seguir, foi proposto por [18]. No presente trabalho, são feitas as seguintes observações 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, $h_1 \geq h_2 \geq ... \geq h_n$

Como conseqüência de (*c*), temos 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:

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

Deve-se observar ainda que, devido à (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 item *j*, tal que $j = i$, inicializa a faixa *i*, não pode ser atribuído novamente a essa faixa. Assim sendo, é definida a seguinte variável binária:

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

O modelo de PLI é apresentado a seguir:

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

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

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

$x_{ij} \in \{0,1\}$ $\quad i = 1,...,(n – 1);\ j > i$

$y_i \in \{0,1\}$ $\quad i = 1,...,n;$ (3.4)

A função objetivo, representada pela expressão (3.1), tem, como critério de otimização, a minimização da altura utilizado do objeto. A restrição (3.2) garante que cada item é alocado apenas uma vez, ou seja, 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 de acordo com 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 encaixe dos 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.

Dentre os algoritmos de níveis mais utilizados, podemos citar os seguintes: *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, foi utilizado o algoritmo *Best-Fit*, descrito a seguir.

Neste algoritmo, 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 mesmo se torne insuficiente. A partir deste momento, 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 0. O pseudocódigo do algoritmo é apresentado na Fig. 2.

```
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, j = 1,...,m}
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 2 – Algoritmo *Best-Fit.*

5. ALGORITMO MEMÉTICO APLICADO AO ODP GUILHOTINADO

Os Algoritmos Genéticos (AG), propostos inicialmente por [8], tornaram-se muito populares para solucionar uma enorme variedade de problemas, em diversas áreas. Segundo [5], um importante desenvolvimento deste método, bastante utilizada na área de otimização combinatória, são os AG's híbridos, que utilizam operadores de busca local para fazer com que indivíduos da população evoluam autonomamente (ou seja, sem sofrer mutações ou recombinações com outros indivíduos) para ótimos locais. Uma generalização dessa idéia foi apresentada por [21], recebendo a denominação de Algoritmo Memético (AM). Neste algoritmo, espera-se que indivíduos possam evoluir autonomamente, através da adição de unidades de informação cultural (memes). Várias aplicações de AM têm demonstrado que, apesar do maior esforço computacional despendido pelos operadores de busca local, o uso desses operadores fornecem resultados melhores quando comparados àqueles produzidos com os AG's.

Os Algoritmos Meméticos foram descritos como uma nova classe de algoritmos evolutivos por [21]. Assim como nos AG's, os AM's também se baseiam nos princípios biológicos de seleção, reprodução (recombinação) e mutação. O que os diferencia dos anteriores é a utilização da evolução cultural, que é um processo no qual a informação cultural é transmitida pela comunicação entre os indivíduos e não pelo mecanismo de recombinação. No AM, uma parte da população é formada por indivíduos chamados de agentes, que realizam explorações independentes nos espaço de busca. A busca no espaço de soluções, portanto, é feita com várias soluções (indivíduos/agentes), como em qualquer algoritmo populacional, constituindo, assim, um processo inerentemente paralelo [5]. A implementação adotada é descrita por [4]. O pseudocódigo simplificado do método é ilustrado a seguir.

Procedimento *Algoritmo Memético*
1. Inicialize a população Pop;
2. Realize um procedimento de otimização em cada indivíduo de Pop;
3. Faça (laço das gerações):
4. Estabeleça um número de recombinações;
5. Para cada recombinação faça:
6. Selecione indivíduos para recombinação;
7. Gere um novo indivíduo a partir dos pais selecionados no passo 6;
8. Realize mutação no novo indivíduo;
9. Realize um procedimento de otimização no novo indivíduo;
10. Tente adicionar o novo indivíduo na população;
Fim;
Até que o critério de parada seja satisfeito;
11. Fim.

Fonte: Adaptado de [5]
Figura 3 – Pseudocódigo simplificado do Algoritmo Memético

A seguir são descritas as etapas do AM proposto. São elas: representação do indivíduo, operadores cruzamento e mutação, função de aptidão de um cromossomo, estratégia de seleção dos indivíduos reprodutores, geração da população inicial e procedimento de busca local.

5.1 REPRESENTAÇÃO DE UMA SOLUÇÃO

Uma solução para o problema é representada pela seqüência em que os itens que serão inseridos, ou seja, é um vetor de *n* valores inteiros, sendo *n* o número de itens do problema (cada elemento do vetor representa um item e sua posição indica a seqüência de inserção).

5.2 OPERADORES DO ALGORITMO MEMÉTICO

5.2.1 Operador de recombinação

O cruzamento é o mecanismo responsável pela cooperação entre os indivíduos/agentes. Neste trabalho, deve-se evitar que uma operação provoque a repetição de uma variável num mesmo individuo. Por isso, para realizar a recombinação *(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:

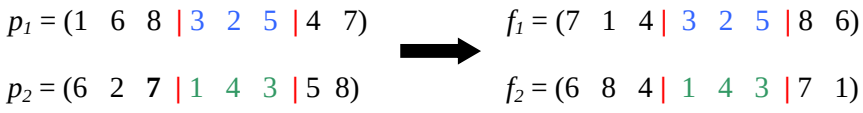

Figura 4 – *Crossover OX*

A probabilidade de *crossover* (p_c) é um valor fixo, escolhido experimentalmente, de forma a aproveitar as informações existentes e, ao mesmo tempo, evitar a convergência prematura (homogeneização rápida da população).

5.2.1. Operador de mutação

A mutação é o operador que tem a função de introduzir características na população, ou mesmo recuperar aquelas que se perderam em operações, como por exemplo, o cruzamento. 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 cruzamento. 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 no exemplo a seguir. Nesse operador, dado um conjunto de n itens, existem $n(n\text{-}1)/2$ indivíduos mutantes possíveis.

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

Figura 5 – Operador de mutação troca

- *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':

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

Figura 6 – Operador de mutação realocação

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. Neste trabalho, a probabilidade de mutação (p_m) é valor fixo escolhido experimentalmente, sendo suficiente para assegurar a diversidade dos cromossomos na população.

5.3 FUNÇÃO DE CUSTO

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, por se tratar 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 à equação da reta (Fig.7) que passa pelos pontos ($fmin$, α) e ($fmax$,β). Considerando $\beta = 0$, temos que:

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

sendo:
$fo(s)$ = 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}.

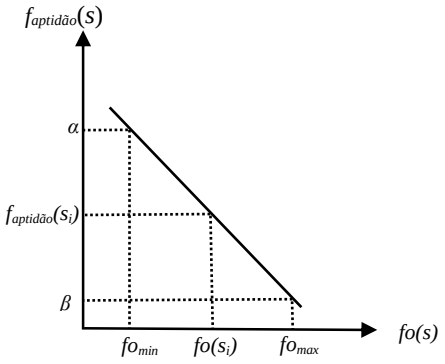

Figura 7 – Esquema do método de escalonamento

5.5 SELEÇÃO DOS INDIVÍDUOS PARA A PRÓXIMA GERAÇÃO

A seleção para as próximas gerações, neste trabalho, é realizada pelo método *roulette wheel*, ou roleta [8]. 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.

Procedimento *Método_da_roleta*;
1 $total \leftarrow \sum_{i=1}^{N} f_i$ {soma das aptidões de todos os cromossomos da população}
2 $rand \leftarrow$ *randômico(0, total)*
3 $totalparcial \leftarrow 0$
4 $i \leftarrow 0$
5 **repetir**
6 $i \leftarrow i + 1$
7 $totalparcial \leftarrow totalparcial + f_i$
8 **até** $totalparcial \geq rand$
9 **retornar** *o cromossomo* $\mathbf{s}_i$
fim *Método_da_roleta*;

Figura 8 – Método da roleta

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 sempre são refinadas e incluídas na nova população. A utilização do elitismo nos AG's tem mostrado que a estratégia é usualmente vantajosa [14].

5.6 GERAÇÃO DA POPULAÇÃO INICIAL

Para gerar a população inicial, optou-se pela utilização de um método construtivo, no qual cada elemento é escolhido aleatoriamente. A população do AM tem tamanho fixo, isto é, um número fixo de indivíduos em todas as gerações.

5.7 OPERADORES DE BUSCA LOCAL

Algoritmos de busca local podem ser considerados operadores do AM, e podem ser empregados em diversas fases do algoritmo. Neste trabalho, foram utilizadas as seguintes técnicas de busca local, nas etapas do algoritmo descritas a seguir:

- *Método da Descida - MD* (*Descent Method*), aplicada nos indivíduos da população depois da execução dos operadores de recombinação e mutação, em cada geração. Este método parte de uma solução inicial *s* qualquer e, a cada passo, analisa todos os seus possíveis vizinhos (considerando uma vizinhança *N(.)* qualquer), movendo-se somente para aquele que apresentar uma melhora no valor atual da função de avaliação. Assim, o método pára quando um ótimo local é encontrado. A Fig. 9(a) mostra o pseudocódigo do algoritmo.

- *Método Randômico de Descida - MRD* (*Random Descent Method*), aplicado em todos os indivíduos da população inicial e nos indivíduos selecionados para a recombinação (pais). Trata-se de uma variação do Método de Descida. Este método parte de uma solução inicial qualquer e, a cada iteração, escolhe aleatoriamente uma vizinhança, que é aplicada à solução corrente. Em seguida, analisa um vizinho qualquer nessa vizinhança, movendo-se somente para o vizinho que representar uma melhora estrita no valor atual da função de avaliação. O método é interrompido após *MRD_max* iterações sem melhora no valor da melhor solução obtida até então. O pseudocódigo do algoritmo é apresentado na Fig. 9(b).

Neste trabalho, para o MD, é considerada uma estrutura de vizinhança, que consiste em movimentos de troca de itens. No MRD, são consideradas duas vizinhanças, sendo que uma envolve movimentos de troca e outra, de realocação de itens.

```
Procedimento Descida (s):
1    V = {s' ∈ N(s)| f(s') < f(s)};
2    enquanto ( |V| > 0) faça
3        Selecione s' ∈ V, onde s'= arg min{f(s') | s' ∈ V};
4        s ← s' ;
5        V = {s' ∈ N(s)| f(s') < f(s)};
6    fim-enquanto;
11   Retorne s;
Fim Descida;
```

(a) Método da Descida

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

(a) Método Randômico de Descida

Figura 9 - Procedimentos de Busca Local.

6. ALGORITMO MEMÉTICO APLICADO AO ODP GUILHOTINADO

O método proposto para solucionar o ODP Guilhotinado, conforme visto nas seções anteriores, é a metaheurística Algoritmo Memético (AM), que neste trabalho consiste basicamente em incorporar, à estrutura de um Algoritmo Génetico, operadores de busca local (Método da Descida e Método Randômico de Descida). A população inicial do AM é gerada de forma aleatória, conforme descrito na seção 5.5, e em seguida é otimizada. A avaliação da população, feita após a aplicação da técnica de encaixe (*Best-Fit*), é feita por uma função escala, descrita na seção 5.3. O individuos selecionados são otimizados antes do cruzamento, que utiliza o operador *OX*, e, em seguida, sofrem mutação. Os individuos, após o cruzamento

e mutação são otimizados. A estratégia de elitismo utilizada consiste em refinar dois melhores indivíduos da geração e mantê-los na próxima geração. O critério de parada do algoritmo é o número de gerações. A Fig. 10 apresenta o pseudocódigo do método proposto.

```
Procedimento AM
1.    Inicializar a população;
2.    t ← 0;
3.    Gerar uma população inicial através de Construção aleatória ();
4.    Otimizar a população inicial com o MRD;
5.    Avaliar P(t);
6.       enquanto (Geracoes < Gerações_Max)faça
7.       Otimizar selecionados para o cruzamento (MRD);
8.       Cruzar selecionados;
9.       Mutar resultantes;
10.      Otimizar novos indivíduos usando MD;
11.      Avaliar P(t);
12.      Selecionar P(t +1) a partir de P(t), usando a estratégia de elitismo;
13.       t ← t + 1;
14.      fim-enquanto;
15.      Retorne o melhor indivíduo;
fim AM
```

Figura 10 – AM proposto

7. RESULTADOS E ANÁLISE

O Algoritmo Memético 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 AM proposto, foram utilizados os problemas-teste de [9], disponíveis na OR-Library. 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

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

É importante ressaltar que os resultados obtidos pelo otimizador e pelo AM proposto, apresentados a seguir, tratam do ODP guilhotinado, sendo, portanto, compreensível que estas soluções apresentem alturas maiores do que as alturas ótimas encontradas nos problemas-teste, solucionados de forma não-guilhotinada, conforme o exemplo da figura a seguir.

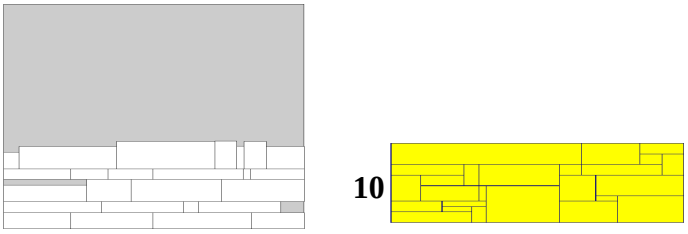

Figura 11 - (a) Exemplo de solução para a metodologia proposta;
(b) Exemplo de solução ótima de um problema-teste resolvido de forma não-guilhotinada.

Os resultados do AM foram comparados com os resultados listados em [22], obtidos pela resolução do modelo matemático apresentado na Seção 3, pelo *software* de otimização LINGO, versão 8.0. Foram resolvidos os problemas-teste das sete categorias, sendo os resultados apresentados na Tabela 2. 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)*
C1P1	16	27*	< 1
C1P2	17	29*	< 1
C1P3	16	23*	< 1
C2P1	25	20*	< 1
C2P2	25	34*	4
C2P3	25	23*	1
C3P1	28	40*	6
C3P2	29	42*	1
C3P3	28	43*	14
C4P1	49	74*	19420
C4P2	49	75*	5736
C4P3	49	81*	160

Problema-teste	*Itens*	*Solução*	*Tempo*
C5P1	73	102	30 hs
C5P2	73	110	
C5P3	73	106	
C6P1	97	145	
C6P2	97	153	
C6P3	97	145	
C7P1	196	300	
C7P2	197	312	
C7P3	196	326	

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

Tabela 3: Parâmetros do Algoritmo Memético

Parâmetros	*Valores*
Número de indivíduos (*nind*)	30
Número de gerações	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*)	*n*

n: número de itens do problema-teste

A Tabela 4 apresenta os resultados dos experimentos. 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:

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

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

Pela Tabela 4, observa-se que os desvios da solução média produzida pelo AM nunca ultrapassaram 3,77 %. Os tempos de processamento variaram de 1,34 a 1234,62 segundos, em média. Observa-se também que o AM sempre alcançou 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 pelo método proposto foram, geralmente, muito inferiores àqueles demandados pelo otimizador aplicado ao modelo de PLI.

Destaca-se que o método proposto encontrou resultados melhores em 7 dos 9 problemas-teste cuja solução ótima não foi encontrada pelo método exato após 30 horas de execução, sendo que, para o problema C7P3, obteve-se uma melhoria média de 3,68 %.

Tabela 4: Resultados do AM.

Problema-Teste	*Nº de itens*	*Solução método exato*	$N_{ótimo}$	*Melhor solução AM*	*Solução média AM*	*Média dos gap's (%) da solução média*	*Tempo Médio (s)*
C1P1	16	27*	10	27*	27	0	1,34
C1P2	17	29*	10	29*	29	0	1,52
C1P3	16	23*	10	23*	23	0	1,38
C2P1	25	20*	10	20*	20	0	4,04
C2P2	25	34*	10	34*	34	0	4,06
C2P3	25	23*	10	23*	23	0	4,10
C3P1	28	40*	10	40*	40	0	6,35
C3P2	29	42*	10	42*	42	0	6,39
C3P3	28	43*	10	43*	43	0	5,51
C4P1	49	74*	8	74*	74	0	25,10
C4P2	49	75*	7	75*	75	0	22,39
C4P3	49	81*	6	81*	81	0	23,56
C5P1	73	102♣	-	102	104	1,96	28,66
C5P2	73	110♣	-	107**	108	-1,82	27,39
C5P3	73	106♣	-	108	110	3,77	26,25
C6P1	97	145♣	-	138**	144	-0,69	106,05
C6P2	97	153♣	-	148**	150	-1,96	105,40
C6P3	97	145♣	-	142**	149	2,76	102,02
C7P1	196	300♣	-	286**	300	0	1178,12
C7P2	197	312♣	-	304**	310	-2,56	1234,65
C7P3	196	326♣	-	307**	314	-3,68	1217,24

* 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 apresentar um Algoritmo Memético para a resolução do ODP guilhotinado. O AM proposto possui a estrutura de um AG tradicional, com a incorporação de operadores de busca local, em diferentes etapas do processo evolutivo. Além disso, é utilizado, como sub-rotina do método, o algoritmo aproximado *Best-Fit.*

O método heurístico proposto foi aplicado nos problemas-teste de [9], que tratam o ODP de forma não-guilhotinada. Mostrou-se nos experimentos que o AM proposto alcançou as soluções ótimas de todos os problemas com até 49 itens. Em sete problemas-teste, com mais de 73 itens, a melhor solução encontrada pelo AM superou as soluções obtidas pelo método exato. Para o restante dos problemas de dimensões maiores (73 a 197 itens), o AM apresentou baixos desvios em relação à melhor solução. O tempo de processamento do método foi pequeno se comparado ao tempo gasto pelo método exato.

Com relação à fixação dos parâmetros do AM, neste trabalho, o número de indivíduos, o número de gerações, a probabilidade de *crossover* e a probabilidade de mutação não variaram em função do problema-teste. Tal característica é desejável num algoritmo, pois apesar de um ajuste conforme o problema-teste produzir melhores resultados, isso dificulta sua análise. Além disso, num sistema real, sendo desconhecidas as características e a solução ótima do problema, não seria razoável fazer estes ajustes.

Os experimentos ainda mostraram que os operadores de busca local permitem a utilização de uma população pequena (30 indivíduos). Como o tempo gasto em cada geração é proporcional ao tamanho da população, com sua diminuição é possível que mais gerações

sejam executadas num mesmo espaço de tempo. Estes resultados empíricos mostram o potencial da técnica proposta e a possibilidade da utilização desta em situações práticas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] **BLAZEWICZ, J. DROZDOWAKI, M. & SONIEWICKI B.** Two-dimensional cutting problem basic complexity results and algorithms for irregular shapes. *Foundations of Control Engineering*, v.14, 1989.

[2] **COFFMAN, S. E. G., GAREY M. R., JOHNSON, D. S., & TARJAN, R. E.** Performance bounds for leve-loriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9, 808-826, 1980.

[3] **COFFMAN, S. E. G., GAREY M. R., JOHNSON, D. S.** Approximation algorithm for bin packing – an updated survey, in algorithms design for computer design system, *Springer-Verlag*, New York, pp. 49-106, 1984.

[4] **FRANÇA, P.M., GUPTA, J.D., MENDES, A.S., MOSCATO, P., VELTINK, K.J.,** Metaheuristic Approaches th International Workshop On Project for the Pure Flowshop Manufacturing Cell Problem, 7 Management and Scheduling, Osnabrück, Germany, 2000.

[5] **GARCIA, V., MENDES, A., FRANÇA, P. & MOSCATO P.** Algoritmo Memético Paralelo Aplicado a Problemas de Sequenciamento em Máquina Simples, in proceedings of *XXXIII SOBRAPO - Simpósio Brasileiro de Pesquisa Operacional*, pp. 971-981, Campos do Jordão, SP, November, 2001.

[6] **HIFI, M.** Exact algorithms for the guillotine strip cutting/packing problem. *Computers & Operations Research*, 25:925-40, 1998.

[7] **HOCHBAUM DS, WOLFGANG M.** Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the Association for Computing Machinery*, 32:130-6, 1985.

[8] **HOLLAND, J.H.** Adaptation in natural and artificial systems. Univ. Of Michugan Press, Ann Arbor, Mich., 1975.

[9] **HOPPER, E., TURTON, B.C.H.** An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research* 128, 34-57, 2001.

[10] **JACOBS, S.** On genetic algorithms for the packing of polygons. *European Journal of Operational Research* 88, 165-181, 1996.

[11] **KROGER, B.** Guillotineable binpacking: A genetic approach. *European Journal of Operational Research*, 84, 645-661, 1995.

[12] **LESH N., MARKS J., MCMAHON A., MITZENMACHER M.** Exhaustive approaches to 2D rectangular perfect packings. *Information Processing Letters*, 90:7-14, 2004.

[13] **LEUNG J., TAM T., WONG C.S., YOUNG G., CHIN F.** Packing squares into square. *Journal of Parallel and Distributed Computing*, 10:271-5, 1990.

[14] **LEVINE, D. M.** A genetic algorithm for the set partitioning problem. *Proceedings of the 5th International Conference on GA's*, 481-487, 1993.

[15] **LIU D, TENG H.** An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112:413-20, 1999.

[16] **LODI A., MARTELLO S., MONACI M.** Two-dimensional packing problems: a survey. *European Journal of Operational Research*, 141:241-52, 2002.

[17] **LODI A., MARTELLO S.,VIGO D.** Recent advances on two-dimensional bin packing problems. Discrete Applied Mathematics, 123(1-3): 379–396, 2002.

[18] **LODI, A.; MARTELO, S. E VIGO, D.** Models and bounds for two-dimensional level packing problems. Journal of Combinatorial Optimization, v.8, p.363–379, 2004.

[19] **MARTELLO, S., MONACI, M., VIGO, D.** An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15, 310-319, 2003.

[20] **MICHALEWICZ, Z.** "Genetic algorithms + Data Structures = Evolution Programs", 3rd Edition, *Springer-Verlag*, 1996.

[21] **MOSCATO, P**., On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report, Caltech Concurrent Computation Program, C3P Report 826, 1989.

[22] **TEMPONI, E.C.C.,** Uma proposta de resolução do problema de corte bidimensional via abordagem metaheurística. Dissertação de Mestrado, CEFET-MG, 2007.

[23] **YEUNG, L.H.W., TANG W.K.S.** Strip-packing using hybrid genetic approach. *Engineering Applications of Articial Intelligence*, 17:169-77, 2004.

[24] **WÄSCHER, G., HAUBNER, H., SCHUANN, H.** An improved typology of cutting and packing problems. *European Journal of Operations Research*, v.4, p.44–454, 2006.