Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional (IX CBRN)
Ouro Preto 25-28 de Outubro de 2009

UM ALGORITMO *SIMULATED ANNEALING* COM REAQUECIMENTO PARA A RESOLUÇÃO DE UM PROBLEMA DE CORTE BIDIMENSIONAL COM ROTAÇÃO DE ITENS

Daniel Góis de Oliveira Gomes*, Adriano do Carmo Oliveira*, Marcone Jamilson Freitas Souza*

*Departamento de Computação
Universidade Federal de Ouro Preto - UFOP
Ouro Preto - Minas Gerais - Brasil

Emails: danielgois.oliveira@gmail.com, adrianooliveira@iceb.ufop.br, marcone@iceb.ufop.br

Abstract— This paper has its focus on the two-dimensional strip packing problem, considering rotation of the items and strip cutting. The objective of the problem consists in allocate all items such that the strip height of the object is minimized. Given its combinatorial complexity, a Simulated Annealing based algorithm was developed with a reheating mechanism. To validate the algorithm, its results were compared to LINGO solver. The computational experiments showed the superiority of the proposed algorithm in solving instances of larger dimensions.

Keywords— Two-dimensional strip packing problem, Simulated Annealing, Mathematical Programming

Resumo— Este trabalho tem seu foco no problema de corte bidimensional, considerando a possibilidade de rotação dos itens e o corte na forma guilhotinada. O objetivo do problema consiste determinar a alocação dos itens de modo a minimizar a altura utilizada do objeto. Dada sua complexidade combinatória, foi desenvolvido um algoritmo Simulated Annealing com um mecanismo de reaquecimento. Para validar o algoritmo, os resultados obtidos pelo mesmo foram comparados com aqueles produzidos pelo otimizador LINGO. Os experimentos computacionais mostraram a superioridade do algoritmo proposto na resolução de problemas-teste de maiores dimensões.

Palavras-chave— Corte bidimensional guilhotinado, *Simulated Annealing*, Programação Matemática

1 Introdução

A alocação de itens retangulares em planos está presente em várias atividades industriais. Na indústria de vidros, assim como na indústria de chapas metálicas, componentes retangulares precisam ser cortados de largas chapas de material. Na paginação de um jornal, artigos e propagandas precisam ser arranjados entre as páginas. Em todas estas aplicações, os planos que conterão os itens têm a forma retangular e um objetivo comum é alocar todos os itens, minimizando o desperdício com o processo.

O problema de otimização resultante é conhecido na literatura por *two-dimensional bin packing problems* (2BP) e está contido na classe de problemas de corte e empacotamento. Em uma das variações do 2BP, tem-se um único plano (um rolo de material), e o objetivo é cortar os itens usando o menor comprimento do rolo. Esse problema é referenciado como *two-dimensional strip packing problem* (2SP) e aparece, por exemplo, na indústria de tecidos e papel.

O 2SP ainda pode ser classificado quanto ao processo de corte, que pode ser guilhotinado – quando o corte se estende de um lado ao outro do rolo, produzindo, a cada corte, dois retângulos – ou não-guilhotinado, se o corte acompanhar o contorno do item, sem descaracterizar o objeto.

Grande parte da literatura sobre os problemas de corte trata do caso onde os itens que serão alocados possuem uma orientação fixa em relação à(s) peça(s) que os conterão, isto é, não é permitido rotacionar os itens. Essa restrição aparece, por exemplo, na paginação de jornais e no corte de tecidos com decoração. Um trabalho com essa configuração é o de Andrade et al. (2008). No entanto, a rotação dos itens, geralmente por 90º, não é uma restrição em muitos contextos, como no corte de vidros, de itens de plásticos, de tecidos sem decoração, de chapas de metais, dentre outros casos. Neste caso, há redução do desperdício no processo se os itens forem dispostos de forma conveniente no objeto. Esse trabalho trata, então, do *two-dimensional strip packing problem* com corte guilhotinado e possibilidade de rotação dos itens em 90º, o qual será denotado por 2SP-GR.

Por se tratar de um problema pertencente à classe NP-Difícil, existem várias abordagens heurísticas para sua resolução. Lesh et al. (2004) apresentam um método exato para o 2SP, baseado num procedimento *branch-and-bound*, que soluciona problemas de pequeno porte (até 30 itens). No entanto, os métodos exatos geralmente não são capazes de solucionar problemas de grande porte em tempo computacional aceitável, justificando, assim, a utilização de técnicas heurísticas para resolução.

Para a resolução do 2SP-GR, implementou-se, neste

trabalho, um algoritmo *Simulated Annealing* com um mecanismo de requecimento automático. Tal escolha se baseou no sucesso alcançado pelos algoritmos de busca local na resolução de problemas altamente combinatórios.

Na resolução do 2SP-GR ainda foram implementadas as técnicas de encaixe *First-Fit* e *Best-Fit*, usadas como sub-rotinas do algoritmo proposto. Para validar os resultados obtidos, adaptou-se o modelo de programação linear inteira (PLI) proposto por Lodi et al. (2004), considerando-se a rotação dos itens. Os resultados obtidos pelo algoritmo proposto são comparados com aqueles gerados pela aplicação do otimizador LINGO, versão 10.0, ao modelo de PLI.

Um exame geral dos problemas de corte e empacotamento pode ser encontrado em Dyckhoff and Finke (1992), Dowsland and Dowsland (1992) e Dyckhoff et al. (1997), entre outros.

O restante deste artigo 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 e o método *Simulated Annealing* proposto para resolução do 2SP-GR, enquanto na Seção 5 são mostrados e discutidos os resultados computacionais. A Seção 6 conclui o trabalho.

2 Descrição do problema

Considere um objeto retangular de largura W e altura suficientemente grande. Considere, também, a necessidade de se obter, a partir desse objeto, um conjunto de itens diversos $i = 1, \cdots, n$ de dimensões (h_i, w_i), onde h_i é a altura do item i, e $w_i \leq W$ a largura (vide Figura 1).

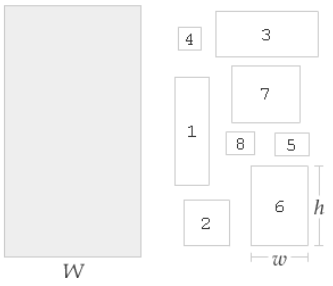

Figura 1: Objeto e itens a serem produzidos

O *two-dimensional strip packing problems* com corte guilhotinado e possibilidade de rotação dos itens (2SP-GR) consiste em determinar a melhor forma de cortar o objeto para atender ao pedido, de modo a minimizar a altura do objeto utilizado. Assume-se, ainda, que cada um dos itens pode ser rotacionado em 90^o.

A Figura 2 mostra uma solução para o exemplo da Figura 1. Nesta solução, o comprimento do objeto é dado pela soma das larguras dos itens 1, 6 e 3, mais a altura do item 7, isto é, o comprimento utilizado h_{tot} é $w_1 + w_6 + w_3 + h_7$.

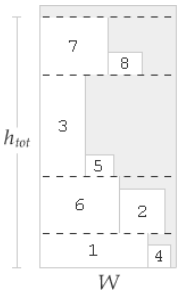

Figura 2: Exemplo de uma solução

No caso de itens retangulares, pode-se cortar um objeto de duas formas: guilhotinada e não-guilhotinada. Na forma guilhotinada, o corte se estende de um lado ao outro do objeto. Já na forma não guilhotinada, o corte acompanha o contorno dos itens. O modelo deste trabalho considera apenas a forma guilhotinada.

3 Modelagem matemática

A formulação de programação matemática apresentada a seguir para o 2SP-GR foi baseada no modelo proposto por Lodi et al. (2004) e possui as seguintes características: (i) os itens são alocados formando-se faixas e (ii) o objetivo consiste em minimizar a soma das alturas das faixas.

Assim, definem-se as seguintes variáveis de decisão, referentes à alocação sem rotação dos itens:

$$y_j = \begin{cases} 1 & \text{se o item } j \text{ inicializa a faixa } j \\ 0 & \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{se o item } i \text{ está alocado na faixa } j \\ 0 & \text{caso contrário} \end{cases}$$

De modo complementar, definem-se as variáveis referentes à alocação com rotação dos itens:

$$y'_j = \begin{cases} 1 & \text{se o item } j, \text{ rotacionado, inicializa a faixa } j \\ 0 & \text{caso contrário} \end{cases}$$

$$x'_{ij} = \begin{cases} 1 & \text{se o item } i, \text{ rotacionado, está alocado na faixa } j \\ 0 & \text{caso contrário} \end{cases}$$

Sendo n o número de itens demandados, o 2SP-GR pode ser modelado pelas Equações (1) a (8) a seguir:

$$\min \quad \sum_{i=1}^{n} (h_i y_j + w_i y'_j) \tag{1}$$

$$\sum_{i=1}^{n} (x_{ij} + x'_{ij}) + y_j + y'_j \;=\; 1 \qquad \forall j = 1, \cdots, n \tag{2}$$

$$\sum_{i=1}^{n} (w_i x_{ij} + h_i x'_{ij}) \;\leq\; (W - w_j) y_j + (W - h_j) y'_j \qquad \forall j = 1, \cdots, n \tag{3}$$

$$h_i x_{ij} + w_i x'_{ij} \;\leq\; h_j y_j + w_j y'_j \qquad \forall i = 1, \cdots, n \quad \forall j = 1, \cdots, n \tag{4}$$

$$y_i \in \{0, 1\} \qquad \forall i = 1, \cdots, n \tag{5}$$

$$y'_i \in \{0, 1\} \qquad \forall i = 1, \cdots, n \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1, \cdots, n \quad \forall j = 1, \cdots, n \tag{7}$$

$$x'_{ij} \in \{0, 1\} \qquad \forall i = 1, \cdots, n \quad \forall j = 1, \cdots, n \tag{8}$$

As equações (2) garantem que cada item será alocado uma única vez. As equações (3) asseguram que o somatório da largura dos itens alocados em cada faixa não ultrapassará a largura do objeto. As equações (4) forçam as faixas a serem inicializadas pelos itens com maior altura. As demais equações definem as variáveis de decisão como binárias.

4 Implementação

Neste trabalho, a solução do problema é representada por um vetor contendo a faixa em que os itens devem ser alocados, sendo essa solução avaliada pela altura encontrada para essa disposição. Outro vetor também é usado para determinar a rotação dos itens. Uma solução é construída por uma técnica de encaixe (Seção 4.1) e refinada por um algoritmo baseado em *Simulated Annealing* (4.2).

4.1 Técnicas de Encaixe

Dentre os algoritmos aproximados desenvolvidos para tratar do encaixe de itens em problemas com duas ou mais dimensões, os mais utilizados são: *Next Fit*, *First Fit*, *Best Fit*, *Next Fit Decreasing Height*, *First Fit Decreasing Height*, *Best Fit Decreasing Height*. 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, ordenam-se os itens em ordem não-decrescente de altura. 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 de os 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 já esteja sendo utilizada (Blazewicz et al., 1989).

```
Algoritmo First Fit()
 1  W ← inicialize largura do objeto
 2  h ← inicialize altura dos itens
 3  w ← inicialize largura dos itens
 4  wf ← inicialize largura das faixas
 5  para i = 1 até n faça
 6      para j = 1 até n faça
 7          d ← W − (wf_j + w_i)
 8          se (d ≥ 0) então
 9              interromper
10          fim-se
11      fim-para
12      s_i ← j
13      wf_j ← wf_j + w_i
14  fim-para
fim First Fit
```

Figura 3: Algoritmo *First Fit*

No *Best Fit*, antes de o 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 for menor. Não sendo possível encaixá-lo, é formada uma nova faixa (Blazewicz et al., 1989). Os pseudocódigos destes algoritmos são apresentados nas figuras 3 e 4.

```
Algoritmo Best Fit()
 1  W ← inicialize largura do objeto
 2  h ← inicialize altura dos itens
 3  w ← inicialize largura dos itens
 4  wf ← inicialize largura das faixas
 5  para i = 1 até n faça
 6      para j = 1 até n faça
 7          d ← W − (wf_j + w_i)
 8          se (d ≥ 0 e d < d_melhor) então
 9              j_melhor ← j
 0              d_melhor ← d
11          fim-se
12      fim-para
13      s_i ← j_melhor
14      wf_j ← wf_j + w_i
15  fim-para
fim Best Fit
```

Figura 4: Algoritmo *Best Fit*

4.2 Algoritmo proposto

O algoritmo proposto para resolver o problema é baseado na metaheurística *Simulated Annealing*. Trata-se de uma técnica de busca local probabilística, proposta em Kirkpathick et al. (1983), que se fundamenta em uma analogia com a termodinâ-

mica, ao simular o resfriamento de um conjunto de átomos aquecidos.

Esta técnica começa sua busca a partir de uma solução inicial qualquer; no nosso caso, formada pela técnica de encaixe *Best Fit*, que mostrou produzir soluções melhores em uma bateria preliminar de testes. O procedimento principal consiste em um *loop* que gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s. Esse vizinho é gerado pela realocação de um item qualquer para uma outra faixa, inclusive para uma nova, e com uma probabilidade de 50% de ser rotacionado. No caso de a nova faixa exceder a largura do objeto, esse excesso é penalizado na função de avaliação por um fator igual a 10 vezes a largura excedida.

Considerando um problema de minimização, seja Δ a variação de valor da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s') - f(s)$. O método aceita o movimento e a solução vizinha passa a ser a nova solução corrente se $\Delta < 0$. Caso $\Delta \geq 0$ a solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde T é um parâmetro do método, chamado de *temperatura* e que regula a probabilidade de se aceitar soluções de pior custo.

A temperatura T assume, inicialmente, um valor elevado T_{inicio}. Após um número fixo de iterações, dado por $SAmax$, o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura, a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_k \leftarrow \alpha \times T_{k-1}$, sendo $0 < \alpha < 1$.

Para tornar o método mais robusto, um mecanismo de reaquecimento é proposto. Quando a temperatura se reduz até o valor mínimo t_{fim}, o processo é continuado elevando-se a temperatura t para $t_0 \times (\beta - i)/\beta$. O parâmetro β corresponde ao número de reaquecimentos que será realizado até que se interrompa completamente o processo de busca local, e i indica o número de reaquecimentos já realizados.

Com esse mecanismo temos um reaquecimento "em escada", sendo o processo restaurado com uma energia um pouco abaixo da temperatura inicial anterior.

5 Resultados

O algoritmo *Simulated Annealing* com reaquecimento proposto, denominado SA-R, foi implementado na linguagem C usando o compilador g++ 3.4, enquanto o modelo de programação matemático foi modelado e resolvido pelo otimizador LINGO, versão 10.0. Tanto o método heurístico quanto o modelo matemático foram testado em um microcomputador com processador Intel Core 2 Quad Q6600, 2.4 GHz e 3,24 GB de RAM, rodando Windows XP Professional.

Para avaliar o algoritmo proposto foram utilizados os problemas-teste de Hopper and Turton (2001), disponíveis na OR-Library. Tais problemas 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 uma delas contém três problemas, totalizando 21 problemas-teste. Para cada problema-teste são dadas a largura e a altura dos itens, bem como a largura do objeto.

Tabela 1: Resultados encontrados

Prob.	LINGO		SA-R			Gap
-teste	Valor	Tempo	Melhor	Média	Tempo	(%)
C1P1 (16)	22,0*	6	22,0*	22,00	43,01	0,00
C1P2 (17)	22,0*	6	22,0*	22,00	46,51	0,00
C1P3 (16)	22,0*	5	22,0*	22,00	43,04	0,00
C2P1 (25)	16,0*	9	16,0*	16,14	72,98	0,88
C2P2 (25)	16,0*	8	17,0	17,00	72,69	6,25
C2P3 (25)	16,0*	9	16,0*	16,73	73,03	4,56
C3P1 (28)	33,0*	605	33,0*	34,26	87,15	3,82
C3P2 (29)	32,0*	603	32,0*	32,53	91,57	1,66
C3P3 (28)	32,0*	606	33,0	33,01	87,19	3,16
C4P1 (49)	65,0	14400	64,0	65,38	188,86	0,58
C4P2 (49)	66,0	14400	63,0	65,50	187,23	-0,76
C4P3 (49)	64,0	14400	64,0	66,50	188,54	3,91
C5P1 (73)	103,0	14400	97,0	99,32	331,34	-3,57
C5P2 (73)	104,0	14400	99,0	102,10	332,81	-1,83
C5P3 (73)	102,0	14400	96,0	99,44	330,75	-2,51
C6P1 (97)	158,0	14400	131,0	135,84	255,22	-14,03
C6P2 (97)	152,0	14400	133,0	136,38	252,60	-10,28
C6P3 (97)	167,0	14400	131,0	136,64	256,72	-18,18
C7P1 (196)	484,0	14400	261,0	261,00	802,39	-46,07
C7P2 (197)	390,0	14400	274,0	281,37	805,00	-27,85
C7P3 (196)	562,0	14400	265,0	272,60	802,58	-51,49

A Tabela 1 apresenta os resultados dos experimentos computacionais. A primeira coluna apresenta o problema-teste, seguido do número de itens correspondente. A segunda coluna mostra os resultados obtidos pela aplicação do *software* LINGO, versão 10.0, ao modelo matemático descrito na Seção 3, limitados a quatro horas de execução. Um número associado com um asterisco indica que o valor correspondente é ótimo. A terceira coluna apresenta o tempo computacional demandado pelo LINGO, em segundos. A quarta e quinta colunas mostram, respectivamente, o melhor resultado e o resultado médio em 100 execuções, considerando-se o *Best Fit* como técnica de encaixe. A sexta coluna mostra o tempo médio despendido pelo algoritmo SA-R, em segundos. A sétima coluna indica o *gap* das soluções médias do SA-R em relação ao valor da solução alcançada pelo LINGO em cada problema-teste.

Para os problemas-teste C1 a C6, os parâmetros adotados no algoritmo SA-R foram os seguintes: Taxa de resfriamento (α) = 0,99; Número de iterações em uma mesma temperatura ($SAmax$) = $5002n$; Número de reaquecimentos (β) = $80n$; Tem-

peratura inicial (t_{inicio}) = 1000; Temperatura final (t_{fim}) = 0,01. Apenas para os problemas-teste da classe C7 houve mudança nos seguintes parâmetros: $\alpha = 0{,}98$; $\beta = 70n$. Em todos os casos, n representa o número de itens.

Como pode ser observado na Tabela 1, o LINGO obteve a solução ótima para todos os problemas-teste com até 29 itens. Para o restante dos problemas, este otimizador não foi capaz de encontrar soluções ótimas no tempo estabelecido, sendo apresentadas, portanto, as soluções obtidas até esse tempo de processamento. O algoritmo SA-R foi capaz de superar o LINGO em 11 problemas-teste com relação à melhor solução, bem como em 10 problemas tendo em vista as soluções médias. Nos problemas da classe C1, bem como em dois problemas da classe C2, o algoritmo SA-R sempre foi capaz de gerar a solução ótima em todas as execuções. Já nos problemas das classes C2 e C3, o SA-R não conseguiu gerar a solução ótima em dois problemas, o C2P2 e o C3P3, ocorrendo, além disso, uma variabilidade média nas soluções finais de até 6,25%. Nos problemas-teste de maiores dimensões (49 a 197 itens), o SA-R teve um comportamento muito melhor que o LINGO. De fato, o SA-R foi capaz de produzir, nesses casos, soluções finais médias melhores em até 51,49% do que aquelas geradas pelo LINGO. Com relação ao tempo computacional, observa-se que o algoritmo SA-R produz soluções de qualidade em um tempo muito menor que o LINGO. A exceção ocorre, apenas, nos problemas-teste das classes C1 e C2; contudo, nesses casos, o tempo gasto pelo SA-R é muito pequeno.

6 Conclusões

Este trabalho teve seu foco no *two-dimensional strip packing problems* com corte guilhotinado, considerando possibilidade de rotação dos itens em 90º. Para resolvê-lo, foi proposta uma formulação de programação linear inteira para o mesmo, bem como um algoritmo heurístico *Simulated Annealing* com um mecanismo de reaquecimento.

Usando problemas-teste da literatura, o algoritmo foi comparado com o otimizador LINGO aplicado ao modelo de programação matemática. Verificou-se que o algoritmo proposto é competitivo com o otimizador nos problemas-teste de menores dimensões, pois apenas em dois problemas-teste não foi capaz de encontrar a solução ótima. Para problemas de maiores dimensões, o SA-R encontrou soluções muito melhores que as do LINGO em um tempo de processamento muito menor. Dado que a tomada de decisão neste problema é fator determinante em vários processos industriais, os resultados encontrados validam a utilização do algoritmo proposto, enquanto ferramenta de apoio à decisão.

Como trabalhos futuros, aponta-se a necessidade de aperfeiçoar o algoritmo com vistas à geração de soluções finais com uma menor variabilidade, principalmente nos problemas das classes C2 e C3. Uma possibilidade seria criar um conjunto elite durante a busca e aplicar o mecanismo de Reconexão por Caminhos como ferramenta de pós-otimização.

Agradecimentos

Os autores agradecem à UFOP e ao CNPq pelo apoio ao desenvolvimento da presente pesquisa.

Referências

Andrade, M. S. F., Souza, S. R. and Souza, M. J. F. (2008). Um algoritmo evolutivo híbrido aplicado à solução do problema de corte bidimensional guilhotinado, *Anais do XXVIII Encontro Nacional de Engenharia de Produção*, Rio de Janeiro, CD-ROM, 14 p.

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

Dowsland, K. A. and Dowsland, W. B. (1992). Packing problems, *European Journal of Operational Research* **56**: 2–14.

Dyckhoff, H. and Finke, U. (1992). Cutting and packing in production and distribution, *Physica Verlag* .

Dyckhoff, H., Scheithauer, G. and Terno, J. (1997). Cutting and packing (cp), *in* S. M. M. Dell'Amico, F.Maffioli (ed.), *Annotated Bibliographies in Combinatorial Optimization*, Wiley, Chichester, pp. 393–413.

Hopper, E. and 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.

Kirkpathick, S., Gellat, D. C. and Vecchi, E. M. P. (1983). Optimization by simulated annealing, *Science* **220**: 671–680.

Lesh, N., Mark, J., Mcmahons, A. and Mitzenmacher, M. (2004). Exhaustive approaches to 2d rectangular perfect packings, *Information Processing Letters* **90**: 7–14.

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