Diferentes abordagens para problemas de empacotamento

IME - Instituto de Matemática e Estatística

Aluno: Rafael Durbano Lobato (lobato@ime.usp.br)
Orientador: Ernesto G. Birgin (egbirgin@ime.usp.br)

Departamento de Ciência da Computação
Instituto de Matemática e Estatística – Universidade de São Paulo

FAPESP

1. Introdução

O problema de carregamento de paletes (PCP) consiste em carregar caixas retangulares sobre um palete retangular. Supõe-se que as caixas, disponíveis em grandes quantidades, devem ser arranjadas ortogonalmente, isto é, com um de seus lados ortogonal a um dos lados do palete, e em camadas nas quais a orientação vertical das caixas é fixada. O PCP tem muitas aplicações práticas e aparece com freqüência na logística de armazenamento e transporte de produtos. Um pequeno aumento no número de caixas transportadas pode representar uma redução significativa de custos.

2. O Problema

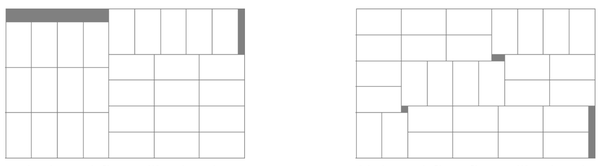
Neste estudo abordamos o PCP do produtor. Esse problema consiste em arranjar, sem sobreposição, caixas retangulares idênticas num palete retangular. As caixas devem ser colocadas ortogonalmente e podem sofrer rotações de 90°. O objetivo é determinar um arranjo com a maior quantidade possível de caixas. Denotaremos por L e W o comprimento e a largura do palete e por l e w o comprimento e a largura das caixas, respectivamente. Consideramos L, W, l, w inteiros e cada problema é determinado pela quádrupla (L, W, l, w).

3. Algoritmos

3.1 Algoritmo de cortes não-guilhotinados de primeira ordem

Um corte é do tipo guilhotinado se, quando aplicado em um retângulo, produz dois novos retângulos. Caso contrário, o corte é do tipo não-guilhotinado. Um padrão é do tipo guilhotinado se é obtido por sucessivos cortes do tipo guilhotinado. A Figura 1(a) ilustra um padrão guilhotinado. O padrão é do tipo não-guilhotinado se é obtido por sucessivos cortes guilhotinados e/ou não-guilhotinados.

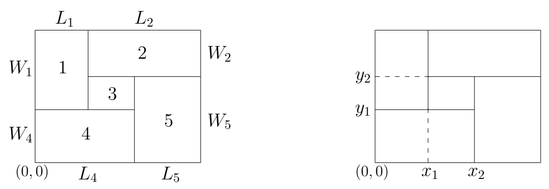
Um corte é do tipo não-guilhotinado de primeira ordem se, quando aplicado em um retângulo, produz cinco novos retângulos arranjados de modo a não formarem um padrão guilhotinado. Um padrão é do tipo não-guilhotinado de primeira ordem se é obtido por sucessivos cortes guilhotinados e/ou não-guilhotinados de primeira ordem. A Figura 1(b) ilustra um padrão não-guilhotinado de primeira ordem.



(a) Padrão guilhotinado. (b) Padrão não-guilhotinado.

Figura 1: *Padrões de corte. Exemplos com* $(L, W, l, w) = (37, 23, 7, 4)$.

Morabito e Morales [3] apresentaram um procedimento recursivo para resolver o problema. Basicamente, ele funciona da seguinte forma: são calculados os limitantes inferior e superior para o número de caixas que podem ser colocadas no retângulo. Se eles forem iguais, a solução ótima foi encontrada e o método devolve o valor obtido. Senão, o retângulo é dividido em no máximo cinco novos retângulos, como ilustra a Figura 2, e o algoritmo é aplicado recursivamente em cada um deles.



(a) Tamanho dos retângulos. (b) Pontos de divisão.

Figura 2: *Divisão do retângulo em cinco partes, através de um corte não-guilhotinado de primeira ordem.*

Os pontos x_1 e x_2 podem ficar restritos ao conjunto S_L e os pontos y_1 e y_2 ao conjunto S_W definidos por

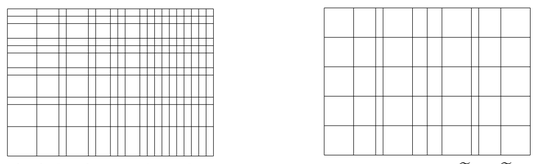
$$S_L = \{x \mid x = rl + sw,\ x \leq L,\ r, s \in \mathbb{Z}_+\} \text{ e} \quad (1)$$
$$S_W = \{y \mid y = tw + ul,\ y \leq W,\ t, u \in \mathbb{Z}_+\} \quad (2)$$

sem perda de generalidade. Porém, esse conjuntos podem ser reduzidos aos conjuntos de *raster points* [1]

$$\widetilde{S}_L = \{\langle L - x\rangle_{S_L} \mid x \in S_L\} \text{ e} \quad (3)$$
$$\widetilde{S}_W = \{\langle L - y\rangle_{S_W} \mid y \in S_W\}, \quad (4)$$

onde

$$\langle x'\rangle_{S_L} = \max\{x \in S_L \mid x \leq x'\} \text{ e}$$
$$\langle y'\rangle_{S_W} = \max\{y \in S_W \mid y \leq y'\}.$$



(a) Conjuntos S_L e S_W. (b) Conjuntos $\widetilde{S}_L$ e $\widetilde{S}_W$.

Figura 3: *Grade de pontos para* $(L, W, l, w) = (28, 20, 7, 4)$.

3.2 Algoritmo-L

Lins, Lins e Morabito [2] propuseram particionar o retângulo não apenas em regiões retangulares, mas também em regiões em forma de L. Uma peça em L é determinada por quatro parâmetros e é denotada por $L(X, Y, x, y)$. Se $X = x$ ou $Y = y$, então a peça é um retângulo e é denotada por $R(X, Y)$.

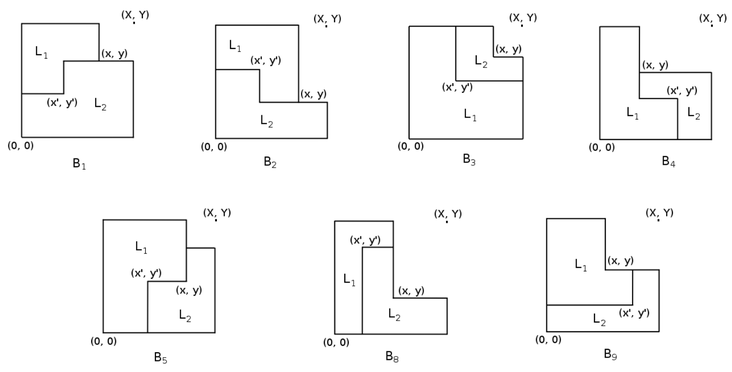


Figura 4: *Subdivisões* $B_1, B_2, B_3, B_4, B_5, B_8$ *e* B_9 *de um* L *em dois* $L's$.

Figura 5: *Subdivisões* B_6 *e* B_7 *de um* R *em dois* $L's$.

Cada subdivisão B_k, $k \in \{1, 2, 3, 4, 5, 8, 9\}$, de um $L(X, Y, x, y)$ em dois novos $L's$, é determinada por dois parâmetros internos, (x', y'), que indicam o local onde a divisão é realizada. O conjunto de tais parâmetros é denotado por $P'_k(L)$.

$$P'_k(L) = \{p_k = (x', y') \mid 0 \leq x' \leq x,\ 0 \leq y' \leq y\},\ k \in \{1, 3, 5\},$$
$$P'_k(L) = \{p_k = (x', y') \mid 0 \leq x' \leq x,\ y \leq y' \leq Y\},\ k \in \{2, 8\},$$
$$P'_k(L) = \{p_k = (x', y') \mid x \leq x' \leq X,\ 0 \leq y' \leq y\},\ k \in \{4, 9\}.$$

As subdivisões B_6 e B_7 de um $R(X, Y)$ em dois novos $L's$ são determinadas por três parâmetros internos e $P_6(R)$ e $P_7(R)$ são dados por:

$$P'_6(R) = \{p_6 = (x', x'', y') \mid 0 \leq x' \leq x'' \leq X,\ 0 \leq y' \leq Y\}$$
$$P'_7(R) = \{p_7 = (x', y', y'') \mid 0 \leq x' \leq X,\ 0 \leq y' \leq y'' \leq Y\}.$$

O algoritmo é baseado em programação dinâmica e possui a seguinte fórmula derecorrência:

$$v^\star(L) = \max_{1 \leq k \leq 9}\{\max_{p_k \in P_k(L)}\{v^\star(\mathscr{L}_1(L, k, p_k)) + v^\star(\mathscr{L}_2(L, k, p_k))\}\}.$$

onde $\mathscr{L}_i(L, k, p_k)$ é a função que devolve a peça L_i de acordo com a subdivisão B_k. Os autores conjecturam que o Algoritmo-L sempre encontra a solução ótima de empacotamentos de retângulos em retângulo.

4. Implementações e experimentos

Implementamos de forma eficiente o Algoritmo 1 com os conjuntos de *raster points*, modificamos o modo de gerar os padrões de corte de forma a gerar apenas os padrões não simétricos entre si e introduzimos o limitante superior de Barnes. A Tabela 1 mostra os tempos de execução da nossa implementação e da implementação original para diferentes níveis de profundidade máxima de recursão.

N	Tempo (segundos)		A / B
	Implementação original (A)	Nossa implementação (B)	
3	5238.67	73.08	71.68
4	6665.58	75.73	88.01
5	7814.75	78.54	99.50
6	8619.68	80.38	107.24
7	9144.11	81.29	112.49
8	9449.20	81.56	115.86
9	9659.74	82.55	117.02
10	9711.25	82.91	117.13
∞	9845.40	71.16	138.36

Tabela 1: *Tempo gasto, em segundos, para resolver 16938 problemas selecionados de Cover II. Apenas 16 problemas não foram resolvidos de forma ótima.*

Implementamos o Algoritmo-L incorporando de forma eficiente os conjuntos de *raster points* e introduzimos as subdivisões $B8$ e $B9$. A Tabela 2 apresenta os resultados alcançados pela implementação original ao resolver problemas selecionados de conjuntos bem conhecidos na literatura, chamados *Cover I* e *Cover II*. A Tabela 3 mostra os resultados obtidos por nossa implementação.

Conjunto de problemas	N° de problemas selecionados	Tempo (segundos)				
		Total	Média	Desvio padrão	Mín.	Máx.
Cover I	3179	3187.41	1.00	1.66	0.00	27.36
Cover II	16938	944231.50	55.74	106.00	0.00	1479.46

Tabela 2: *Implementação proposta em [2] com os conjuntos (1) e (2).*

Conjunto de problemas	N° de problemas selecionados	Tempo (segundos)				
		Total	Média	Desvio padrão	Mín.	Máx.
Cover I	3179	1650.24	0.51	0.33	0.00	2.48
Cover II	16938	83298.61	4.91	6.90	0.00	56.91

Tabela 3: *Nossa implementação com raster points.*

4.1 Memória

O Algoritmo-L utiliza uma matriz de quatro dimensões de tamanho $|\widetilde{S}_L|^2|\widetilde{S}_W|^2$ para armazenar as informações dos subproblemas. No entanto, para problemas grandes, não é possível alocar tal quantidade de memória. A solução encontrada foi trocar essa estrutura de dados por uma que combina uma matriz multidimensional e uma árvore binária de busca e que se adapta ao problema. Essa nova estrutura de dados, apesar de menos eficiente do que a primeira, permite resolver problemas maiores.

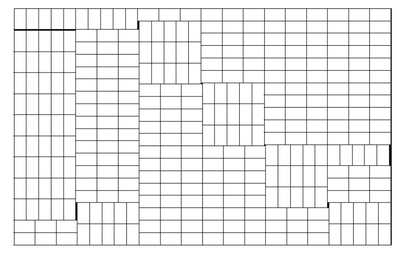
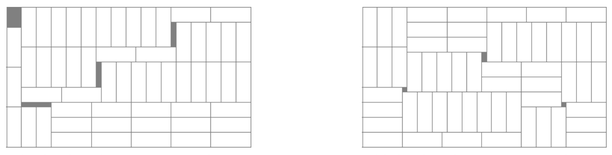


Figura 6: *Problema (2560, 1610, 143, 84). Solução com 341 caixas.*

4.2 Combinação dos algoritmos

A grande vantagem do Algoritmo 1 é a sua velocidade. Ele levou apenas 71 segundos para resolver 16938 problemas selecionados de *Cover II*, enquanto que o Algoritmo-L demorou 23 horas. No entanto, o Algoritmo-L encontrou a solução ótima de todos os problemas, enquanto que o Algoritmo 1 falhou em 16 problemas, encontrando padrões com uma caixa a menos. A Figura 7 mostra um desses problemas.



(a) Algoritmo 1: 56 caixas. (b) Algoritmo-L: 57 caixas.

Figura 7: *Instância* $(49, 28, 8, 3)$.

A fim de unir a velocidade do Algoritmo 1 e a robustez do Algoritmo-L, implementamos uma versão que combina os dois algoritmos da seguinte forma: primeiramente, aplicamos o Algoritmo 1 ao problema que desejamos resolver. Se a otimalidade da solução encontrada não puder ser comprovada, executamos o Algoritmo-L. Além disso, todas as informações obtidas na execução do primeiro algoritmo são aproveitadas pelo segundo.

Referências

[1] G. Scheithauer e J. Terno. The G4-Heuristic for the Pallet Loading Problem. *Journal of the Operational Research Society*, 47(4):511–522, 1996.

[2] L. Lins, S. Lins e R. Morabito. An L-approach for packing (ℓ, w)-rectangles into rectangular and L-shaped pieces. *Journal of the Operational Research Society*, 54(7):777–789, 2003.

[3] R. Morabito e S. Morales. A simple and effective recursive procedure for the manufacturer's pallet loading problem. *Journal of the Operational Research Society*, 49(8):819–828, 1998.