Diferentes abordagens para problemas de empacotamento

Rafael Durbano Lobato
Ernesto G. Birgin (orientador)

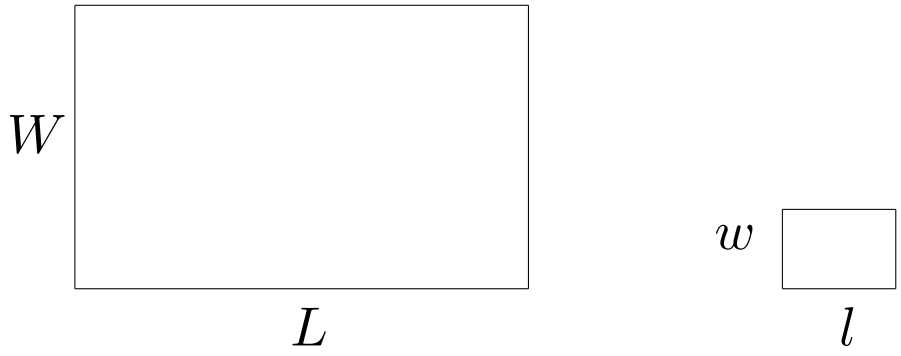
Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

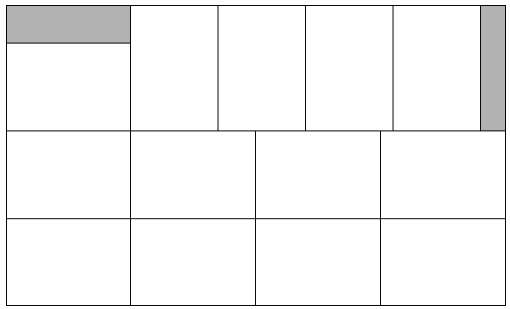
Apoio financeiro: FAPESP

17/11/2006

O Problema

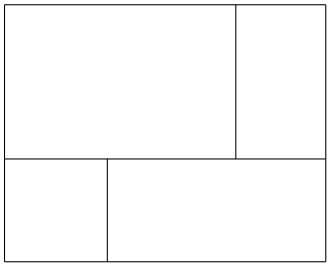
Dados um palete de comprimento L e largura W e caixas de comprimento l e largura w ($L, W, l, w \in \mathbb{Z}$), determinar um empacotamento ortogonal com a maior quantidade possível de caixas, sem que elas se sobreponham.





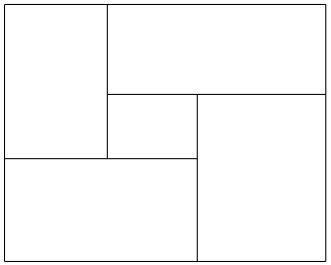
Definições

Corte guilhotinado: produz dois novos retângulos quando aplicado em um retângulo.



Padrão guilhotinado

Corte não-guilhotinado de primeira ordem: quando aplicado em um retângulo, produz cinco novos retângulos arranjados de modo a não formar um padrão guilhotinado.



Padrão não-guilhotinado de primeira ordem

Algoritmo de cortes não-guilhotinados de primeira ordem

Primeiramente, são calculados os limitantes inferior (z_{lb}) e superior (z_{ub}) para o número de caixas que podem ser colocadas no palete.

$$z_{lb} = \max\left\{\left\lfloor\frac{L}{l}\right\rfloor\left\lfloor\frac{W}{w}\right\rfloor, \left\lfloor\frac{L}{w}\right\rfloor\left\lfloor\frac{W}{l}\right\rfloor\right\}$$

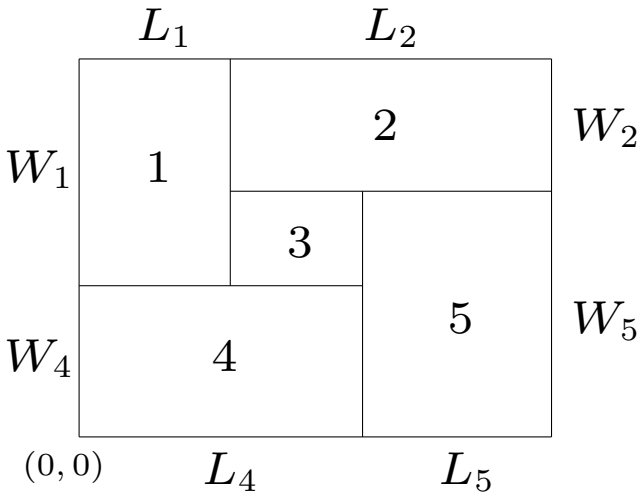
$$z_{ub} = \left\lfloor\frac{LW}{lw}\right\rfloor$$

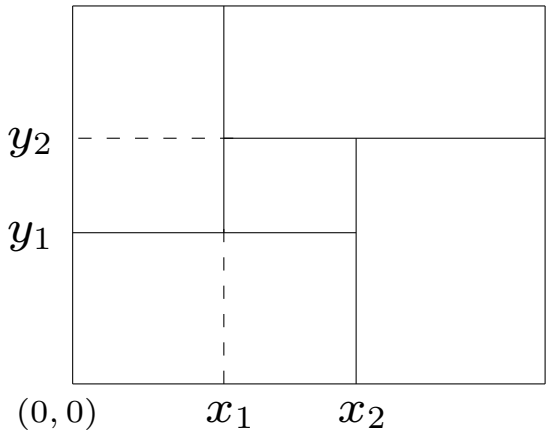
O limitante inferior é dado pelo melhor empacotamento homogêneo.

Empacotamentos homogêneos vertical e horizontal

Algoritmo de cortes não-guilhotinados de primeira ordem

Se $z_{lb} = z_{ub}$, a solução ótima foi encontrada e o método devolve z_{lb}. Senão, o palete (L, W) é dividido em, no máximo, cinco regiões, através de cortes guilhotinados e não-guilhotinados de primeira ordem, e o algoritmo é aplicado recursivamente em cada um deles.





Algoritmo de cortes não-guilhotinados de primeira ordem

Em quais pontos devemos realizar as divisões do palete?

Sejam (x, y) as coordenadas de algum vértice de uma caixa colocada no palete. Então x e y podem tomar valores nos conjuntos $\{x \mid 0 \leq x \leq L\}$ e $\{y \mid 0 \leq y \leq W\}$, respectivamente.

Algoritmo de cortes não-guilhotinados de primeira ordem

Em quais pontos devemos realizar as divisões do palete?

Sejam (x, y) as coordenadas de algum vértice de uma caixa colocada no palete. Então x e y podem tomar valores nos conjuntos $\{x \mid 0 \leq x \leq L\}$ e $\{y \mid 0 \leq y \leq W\}$, respectivamente.

Porém, eles podem ser reduzidos aos seguintes conjuntos:

$$
\begin{aligned}
S_L &= \{x \mid x = rl + sw,\ x \leq L,\ r, s \in \mathbb{Z}_+\} \\
S_W &= \{y \mid y = tw + ul,\ y \leq W,\ t, u \in \mathbb{Z}_+\}
\end{aligned}
$$

Algoritmo de cortes não-guilhotinados de primeira ordem

É possível reduzi-los ainda mais, considerando os conjuntos de *raster points*

$$\begin{aligned} \widetilde{S}_L &= \{\langle L - x\rangle_{S_L} \mid x \in S_L\} \text{ e} \\ \widetilde{S}_W &= \{\langle L - y\rangle_{S_W} \mid y \in S_W\}, \end{aligned}$$

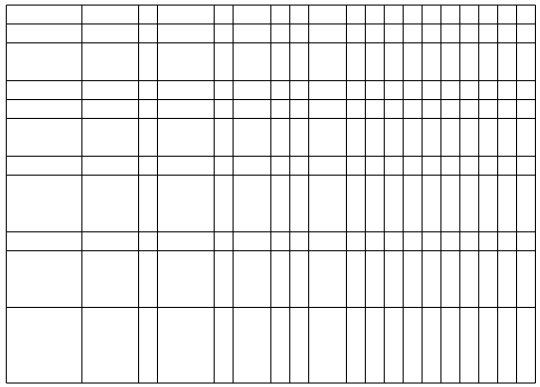
onde,

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

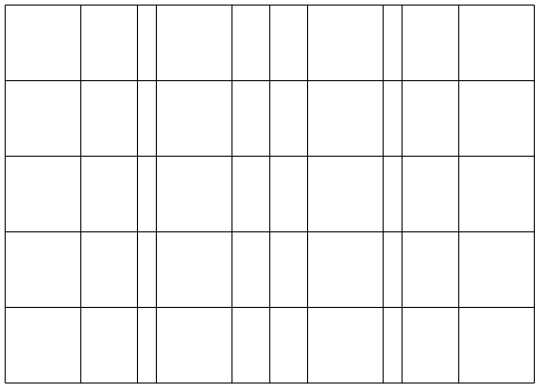
Algoritmo de cortes não-guilhotinados de primeira ordem

Para o problema $(L, W, l, w) = (28, 20, 7, 4)$, temos:

$$
\begin{aligned}
S_L &= \{0, 4, 7, 8, 11, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28\}, \\
\widetilde{S}_L &= \{0, 4, 7, 8, 12, 14, 16, 20, 21, 24, 28\}, \\
S_W &= \{0, 4, 7, 8, 11, 12, 14, 15, 16, 18, 19, 20\} \text{ e} \\
\widetilde{S}_W &= \{0, 4, 8, 12, 16, 20\}.
\end{aligned}
$$



$S_L \times S_W$



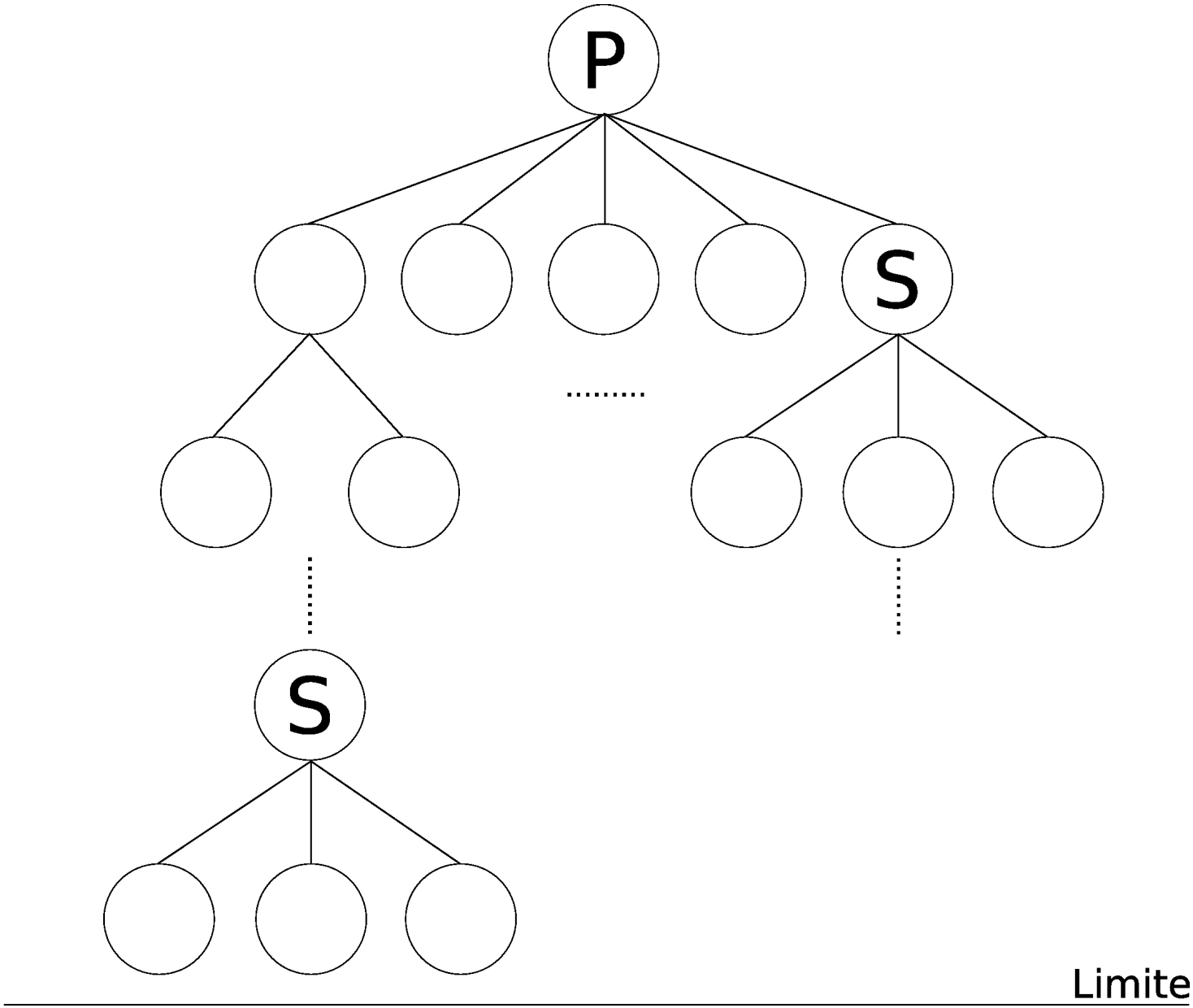
$\widetilde{S}_L \times \widetilde{S}_W$

Algoritmo de cortes não-guilhotinados de primeira ordem

Detalhes de implementação:

- Incorporação dos *raster points*;
- Geração apenas dos padrões não simétricos entre si;
- Inclusão do limitante superior de Barnes;
- Tratamento de subproblemas repetidos.

Algoritmo de cortes não-guilhotinados de primeira ordem



Experimentos

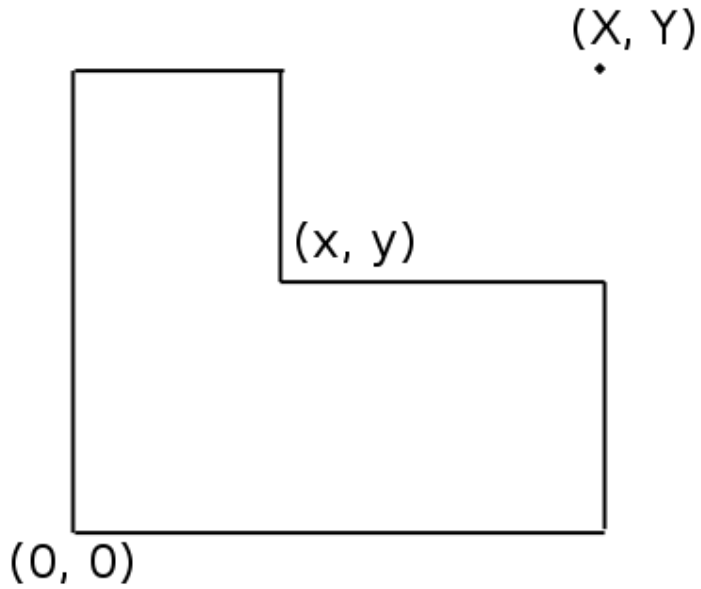
N	Tempo (segundos)		A / B
	Implementação original (A)	Nossa implementação (B)	
3	5238.67	73.08	71.68
4	6665.58	75.73	88.01
5	7814.75	78.54	99.50
6	8619.68	80.38	107.24
7	9144.11	81.29	112.49
8	9449.20	81.56	115.86
9	9659.74	82.55	117.02
10	9711.25	82.91	117.13
∞	9845.40	71.16	138.36

Tabela 1: Tempo gasto, em segundos, para resolver 16938 problemas de *Cover II*. Apenas 16 problemas não foram resolvidos de forma ótima. (AMD Athlon 64 3200+ 2.2 GHz, 1 GB de memória RAM)

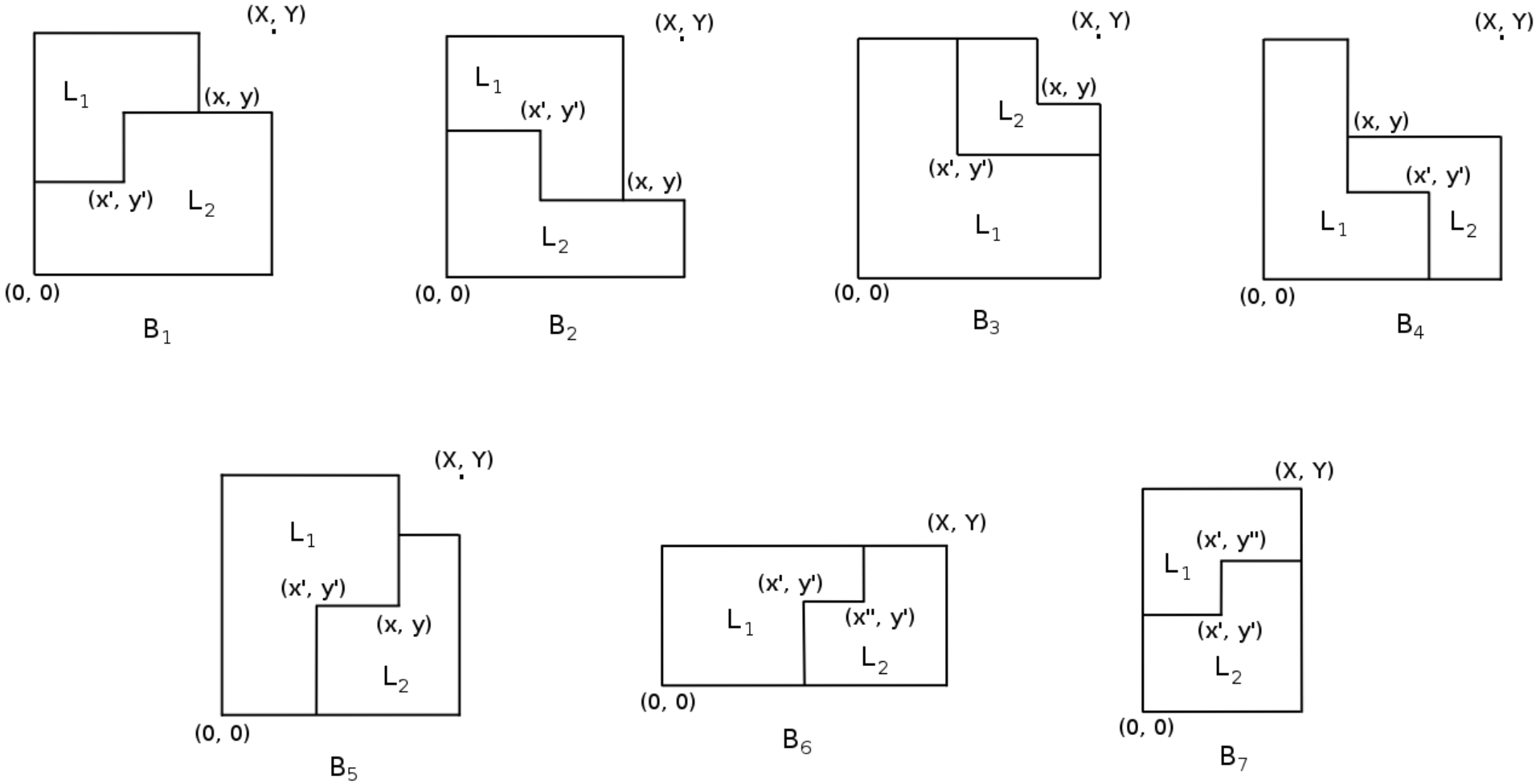
Algoritmo-L

A proposta foi de substituir o algoritmo de particionamento recursivo em cinco regiões por um particionamento recursivo de um retângulo (que chamaremos de R) ou de uma peça em forma de L (que chamaremos de L) em duas peças, cada uma delas sendo um R ou um L.

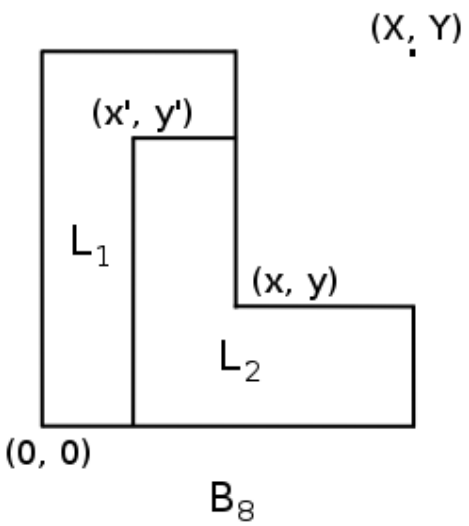
Um L é determinado por quatro inteiros (X, Y, x, y) e seu posicionamento padrão é definido como o fechamento topológico do retângulo cuja diagonal vai de $(0, 0)$ até (X, Y) menos o retângulo que vai de (x, y) até (X, Y), com $X \geq x$ e $Y \geq y$.

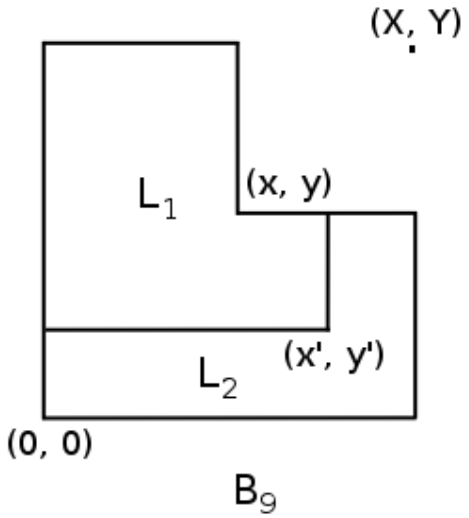


Divisões de L em dois L's (B_1, B_2, B_3, B_4 e B_5) e de R em dois L's (B_6 e B_7):



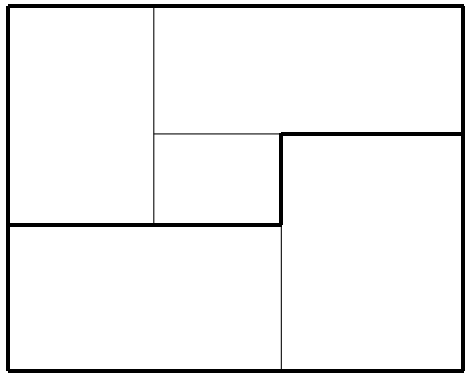
Duas novas divisões (B_8 e B_9):





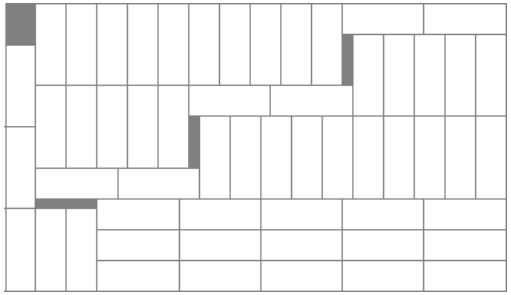
Algoritmo-L

O Algoritmo-L é mais geral!

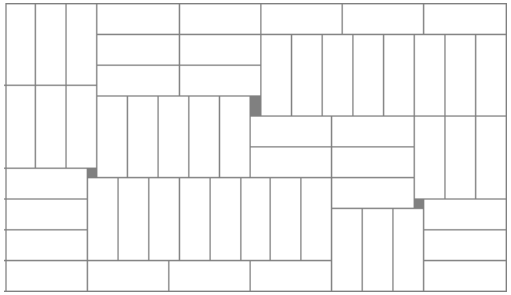


Também produz padrões não-guilhotinados de primeira ordem

Exemplo: $(L, W, l, w) = (49, 28, 8, 3)$.



Algoritmo 1: 56 caixas

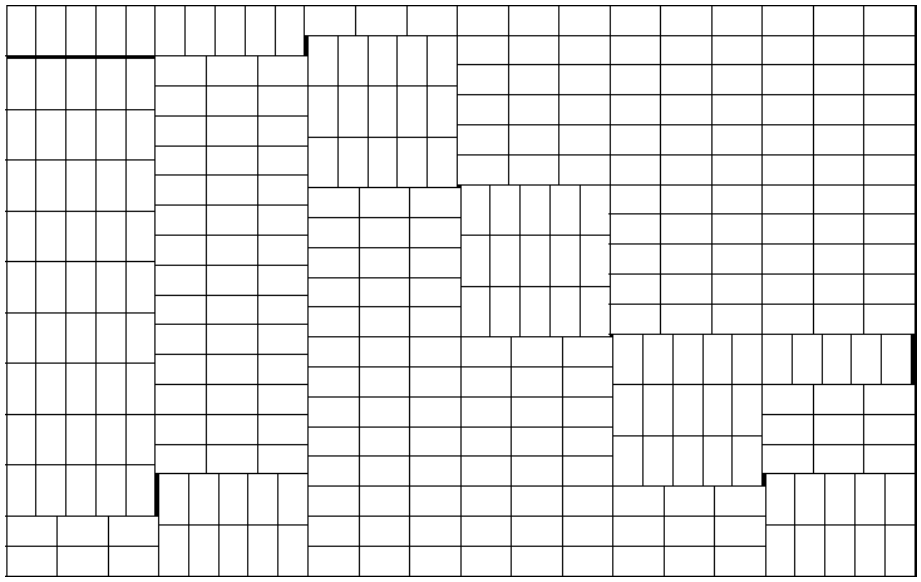


Algoritmo-L: 57 caixas
(solução ótima)

Algoritmo-L

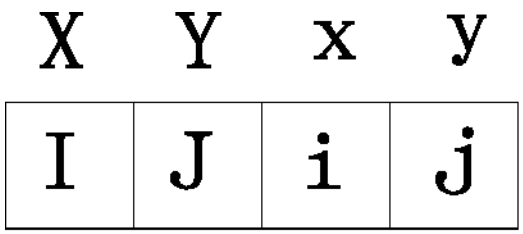
Detalhes de implementação:

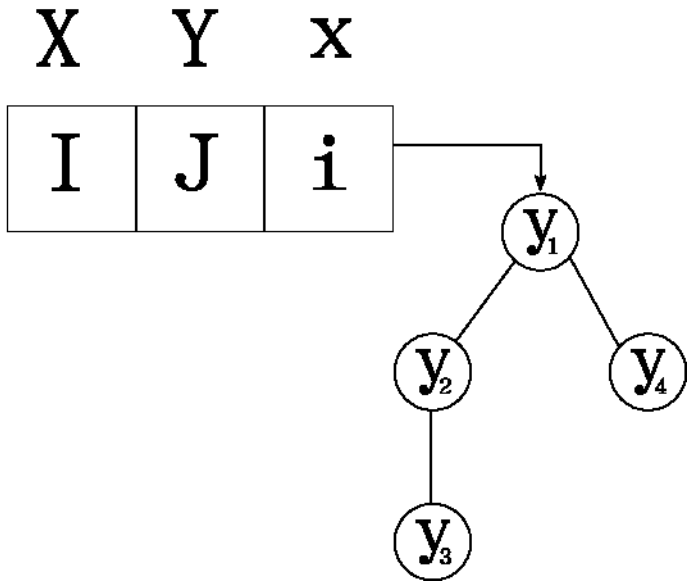
- Introdução de duas novas formas de dividir um L em dois;
- Utilização dos $raster\ points$;
- Controle de memória.



Instância $(2560, 1610, 143, 84)$. Solução com 341 caixas.

Controle de memória: alocamos a maior quantidade possível de memória para armazenar as informações dos subproblemas.





Experimentos

Conjunto de problemas	Número de problemas	Tempo (segundos)				
		Total	Média	Desvio padrão	Mín.	Máx.
16 hard	16	133.59	8.34	5.61	0.49	19.63
Cover II	16938	83298.61	4.91	6.90	0.00	56.91

Tabela 2: Implementação com *raster points*.

Conjunto de problemas	Número de problemas	Tempo (segundos)				
		Total	Média	Desvio padrão	Mín.	Máx.
16 hard	16	695.34	43.45	41.23	1.20	155.02
Cover II	16938	944231.50	55.74	106.00	0.00	1479.46

Tabela 3: Implementação sem *raster points*.

Algoritmo 1 $\times$ Algoritmo-L

Algoritmo 1

- Muito rápido;
- Não encontra a solução ótima de todos os problemas.

Algoritmo-L

- Mais lento;
- Encontra soluções ótimas de alguns problemas que o Algoritmo 1 não é capaz de encontrar.

Algoritmo 1 $\times$ Algoritmo-L

Algoritmo 1

- Muito rápido;
- Não encontra a solução ótima de todos os problemas.

Algoritmo-L

- Mais lento;
- Encontra soluções ótimas de alguns problemas que o Algoritmo 1 não é capaz de encontrar.

Combinação dos dois algoritmos: Executamos, primeiramente, o Algoritmo 1. Se não for possível comprovar a otimalidade da solução encontrada, rodamos o Algoritmo-L. Além disso, todas as informações dos subproblemas obtidas durante a execução do primeiro algoritmo são aproveitadas pelo segundo.

Perguntas?

Mais informações

Mais informações podem (poderão) ser encontradas em:

`www.linux.ime.usp.br/~lobato/mac499/`

Referências

[1] E. G. Birgin, R. Morabito and F. H. Nishihara. A note on an L-approach for solving the manufacturer's pallet loading problem. *Journal of the Operational Research Society*, 56:1448–1451, 2005.

[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.

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