Heurísticas para empacotamento em containers

Clovis Perin
Valéria de Podestá Gomes
Antonio Carlos Moretti
Departamento de Matemática Aplicada
Universidade Estadual de Campinas

Resumo

Carregamento de caixas de embalagens em containers e em paletes constituem importantes aplicações de modelos matemáticos de corte e empacotamento no planejamento da produção em indústrias de papel, aço, madeira, vidro, etc. Em geral, o problema real é apresentado de tal forma que para construir um modelo de programação matemática que o represente é necessário gerar um conjunto de padrões de empacotamento. Trata-se de um problema *NP-Hard*. Neste trabalho, são estudadas estratégias heurísticas alternativas para a geração de empacotamentos. Testes computacionais foram realizados para comparar tempos de processamento e qualidade dos empacotamentos obtidos para exemplares de problemas de empacotamento tridimensionais disponíveis na internet.

1 Introdução

Carregamento de caixas de embalagens em containers e em paletes é uma importante atividade de manipulação em indústrias de fabricação e distribuição. Produtos são embalados em caixas retangulares para proteção e manuseio, sendo estas caixas empilhadas e posicionadas no interior de containers para transporte e armazenagem. Este processo pode ser efetuado de modo manual ou automático. Este tipo de problema tem aplicações importantes no planejamento de transporte rodoviário, aéreo, marítimo, etc.

Empacotamento em containers é classificado como um problema de empacotamento retangular tridimensional e é relacionado com o problema tridimensional de corte de estoque.

No problema de corte de estoque, a ênfase é no corte de um bloco retangular de estoque em diversas peças menores de dimensões conhecidas. Este problema aparece em indústrias de manufatura no corte de espuma, madeira, etc. O objetivo usual é minimizar as perdas. Neste caso, os cortes são do tipo guilhotina, que partem de uma faceta da peça de estoque e correm paralelos a quatro arestas de outras facetas.

No problema de empacotamento tridimensional são utilizados cortes não-guilhotinados. Estes problemas podem ser de dois tipos [12]: do distribuidor e

do fabricante. O primeiro está relacionado com o empacotamento dos pedidos dos clientes em caixas de diferentes dimensões que são colocadas em containers e tem por objetivo maximizar o volume empacotado. O outro consiste em produzir e empacotar os produtos em caixas e empacotá-las em containers idênticos.

Um grande número de estratégias de modelagem (orientadas para peças, orientadas para cortes) e algoritmos já foram propostos. Neste trabalho vamos estudar problemas de empacotamento tridimensional do tipo $3/V/I/C$, conforme tipologia proposta em [6, 7], que classifica os problemas por dimensionalidade, tipo de associação entre peças e objetos, variedade de objetos e variedade de peças. Convém ressaltar que estamos considerando apenas empacotamentos ortogonais, onde as arestas das peças são mantidas paralelas às dos objetos. Caso contrário, pode haver empacotamento melhor, como apontado em [8].

De modo geral, são problemas de difícil resolução, que demandam tempo e memória computacional. Tais problemas são classificados como *NP-Hard*. Vamos concentrar o nosso estudo em heurísticas, comparando-as com o auxílio de exemplares de problemas de empacotamento disponíveis na literatura.

O objetivo deste trabalho é testar e comparar várias heurísticas para resolver problemas de empacotamento em containers.

Na Seção 2 deste artigo apresentamos uma descrição de modelos de empacotamento, na Seção 3, as heurísticas desenvolvidas neste trabalho e na Seção 4, os testes computacionais realizados com exemplares disponíveis na internet.

2 Modelos de Empacotamento

Considere o problema de corte tridimensional não-guilhotinado, definido por uma carteira com m pedidos de caixas retangulares com comprimentos c_i, larguras ℓ_i e alturas h_i, em quantidades b_i, a serem empacotadas em containers retangulares com comprimento C, largura L e altura H. Suponha que há n padrões de empacotamento especificados por uma $m\times n$-matriz A, isto é, o elemento $a_{ij} \in \mathcal{N}$ da matriz especifica o número de vezes que a caixa i é considerada no padrão de empacotamento j. Isto implica que $\sum_i a_{ij}c_i\ell_i h_i \leq CLH$. Deseja-se determinar um n-vetor de quantidades $w = (w_j)$, solução ótima do programa matemático:

$$max\{ \sum_j f_j w_j : Aw \simeq b,\ w \in \mathcal{W} \} \tag{1}$$

Neste modelo, tanto é possível minimizar o número total de containers utilizados ($f_j = -1$), quanto maximizar o volume empacotado ($f_j = \sum_i a_{ij}c_i\ell_i h_i$), etc. Além disto, o símbolo $\simeq$ deve ser substituído por um dos símbolos $=$ ou $\geq$, dependendo de como a carteira de pedidos deve ser atendida. O conjunto $\mathcal{W}$ pode tanto representar o n-produto cartesiano dos reais não-negativos $\Re_+^n$ como dos naturais $\mathcal{N}^n$ associando, desta forma, um problema linear contínuo ou inteiro.

A resolução de um problema de empacotamento, em geral, envolve a geração de padrões de empacotamento. Utilizando ternos de variáveis de decisão (x_i, y_i, z_i) para denotar as coordenadas do canto inferior esquerdo frontal da caixa i no

padrão de empacotamento, a matriz de restrições A deve ser construída de tal modo que cada caixa $c_i \times \ell_i \times h_i$ seja alocada inteiramente no objeto $C \times L \times H$ e deve satisfazer as condições

$$x_i + c_i \leq C, \qquad y_i + \ell_i \leq L, \qquad z_i + h_i \leq H, \qquad x_i, y_i, z_i \geq 0$$

Além disto, não pode haver sobreposição para nenhum par i, j de caixas e isto é construído com um grupo das seguintes restrições alternativas

$$\begin{aligned} x_i + c_i &\leq x_j \quad \text{ou} \\ x_j + c_j &\leq x_i \quad \text{ou} \\ y_i + \ell_i &\leq y_j \quad \text{ou} \\ y_j + \ell_j &\leq y_i \quad \text{ou} \\ z_i + h_i &\leq z_j \quad \text{ou} \\ z_j + h_j &\leq z_i \end{aligned}$$

cuja implementação, em modelos de programação matemática, é construída com a inclusão de variáveis binárias.

A literatura científica registra diversos modelos de programação inteira mista 0-1 para empacotamento em paletes e em containers que eventualmente consideram um certo grau variável de rotação das caixas que compõem a carteira de pedidos.

Uma estratégia eficiente e muito empregada na resolução de problemas de corte e empacotamento modelados como programação linear (e também fracionária) foi proposta por Gilmore e Gomory [9, 10]. Ao invés de gerar todos os padrões de empacotamento para construir explicitamente a matriz de restrições do programa matemático, um procedimento iterativo, utilizando as variáveis duais obtidas diretamente das equações básicas do método simplex, ele resolve um problema da mochila cuja solução é um novo padrão de corte (i.e., uma nova coluna), que caso seja lucrativo será adicionado ao problema de corte atual. Desta maneira, o número de padrões gerados tende a ser menor do que o total necessário para o modelo completo.

Um modelo adequado para ser utilizado com a estratégia iterativa de Gilmore e Gomory [9, 10] é proposto em [5]. Todas as caixas são alocadas em um enorme objeto e apenas aquelas alocadas em uma região especial (de mesmo tamanho dos objetos) são selecionadas para compor um padrão de empacotamento. O posicionamento das caixas é feito para otimizar o valor das caixas selecionadas. O modelo utiliza variáveis para fixar o canto inferior esquerdo frontal de cada caixa (variáveis (x_i, y_i, z_i) no caso tridimensional).

Um outro modelo com restrições sobre o número de caixas de um mesmo tipo na solução é proposto em [16]. O modelo tem por objetivo utilizar o menor número total de objetos e utiliza as mesmas variáveis (x_i, y_i, z_i) para fixar o canto inferior esquerdo frontal de cada caixa i selecionada.

Modelos de busca em grafo do tipo e/ou são apresentados em [1, 13]. Nesta abordagem, o espaço de soluções é reduzido ao considerar apenas soluções

com padrões de empacotamento construídos com cortes guilhotinados e não-guilhotinados simples. Tais padrões de empacotamento correspondem a caminhos completos no grafo e/ou e é utilizado um método *branch and bound* para gerar o melhor destes padrões. Deve-se ressaltar que esta abordagem não gera todos os padrões de empacotamento possíveis. Também são estudadas abordagens heurísticas.

Uma abordagem híbrida é estudada em [12]. Ela considera uma combinação de programação dinâmica e heurísticas, numa situação que envolve algumas caixas com conteúdo volátil e que devem ser posicionadas nas periferias dos paletes para facilitar o acesso.

Um modelo em grafo para representar um padrão qualquer de empacotamento é apresentado em [2]. Infelizmente, esta abordagem não permite gerar os padrões de empacotamento, mas tão somente representá-los por meio de um grafo.

3 Heurísticas

Uma heurística que preenche o container em estágios é proposta em [11]. Ela constrói camadas ao longo do comprimento procurando, desta forma, manter homogeneidade. A escolha das caixas é feita com prioridade para os tipos de caixas abertas em oposição às caixas do tipo fechadas, que ainda não foram selecionadas no empacotamento. A primeira caixa de cada camada é crucial, pois ela define a profundidade da camada. As outras caixas são utilizadas para preencher os espaços vazios restantes.

O preenchimento por camada ao longo do comprimento é proposto em [3]. Neste caso, cada camada é construída com apenas um único tipo de caixa e o arranjo destas caixas é determinado com um procedimento bidimensional que procura maximizar a área utilizada. Quando não é possível construir uma camada com um único tipo de caixa, o procedimento de [11] é utilizado.

Uma outra heurística composta pelas heurísticas em [11] e [3] é proposta em [4]. É realizada uma comparação entre diversas heurísticas.

Uma estrutura de dados eficiente para representar e armazenar um empacotamento é apresentada em [14]. Ela utiliza uma técnica de representação espacial que não fica restrita à seqüência de empacotamento utilizada, em geral, de trás para frente, de baixo para cima, etc. Esta estrutura de dados é muito eficaz para determinar espaços vazios no empacotamento corrente. Além disto, ela é construída na seqüência em que as caixas são inseridas no container, e isto permite implementar a solução com facilidade, tanto para empacotar como para desempacotar o container.

Neste trabalho, implementamos em Matlab e comparamos 4 heurísticas para empacotamentos em containers.

Estas heurísticas foram implementadas com uma estrutura de dados sugerida em [14]. Utilizamos três vetores `com`, `lar` e `alt` e uma matriz tridimensional `ind` que particionam o container em células com a seguinte notação: $\texttt{ind}(i, j, k) = p$ indica que a caixa p ocupa a célula que tem canto inferior es-

querdo frontal em $(x, y, z) = (\texttt{com}(i), \texttt{lar}(j), \texttt{alt}(k))$ e canto superior direito posterior em $(x, y, z) = (\texttt{com}(i+1), \texttt{lar}(j+1), \texttt{alt}(k+1))$. Uma mesma caixa pode ocupar mais de uma célula e se $p = 0$ a célula está vazia. O número de elementos destes três vetores tende a crescer a cada inserção de caixa no container, da mesma forma que a matriz de índices de caixas. A Figura 1 mostra os coeficientes da matriz `ind` para a altura 10 com três caixas identificadas pelos números 1, 2, 3 com dimensões horizontais 35×25, 35×40, 30×20, respectivamente. Convém salientar que o exemplo não é preciso, pois o número de elementos de cada vetor deve ser mantido como o menor possível; assim, $\texttt{com} = (00, 30, 35, 75)$, $\texttt{lar} = (00, 25, 40, 45, 50)$.

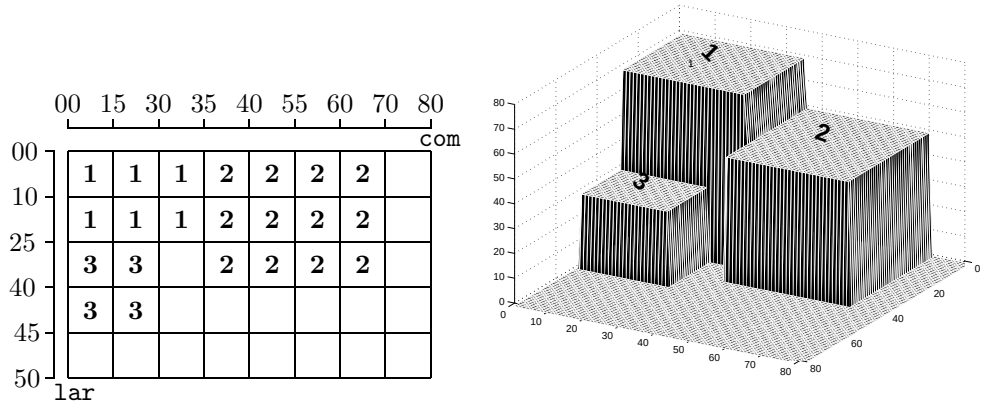

Figura 1: Matriz `ind` com 3 caixas em uma dada altura

A primeira heurística, que denominaremos `D3a`, é proposta em [14] e é aplicável para problemas de empacotamento com muitas caixas distintas. Ela trabalha com a inserção de uma caixa em um espaço vazio por iteração, selecionadas através de um sistema de ranqueamento baseado na igualdade de dimensões de cada caixa com cada espaço vazio do container. Na proposta original, a altura é considerada uma direção predominante, pois a igualdade da caixa com o vazio na altura é considerada mais importante do que a igualdade da dimensão ao longo do comprimento ou da largura. Esta heurística executa três passos por iteração:

1. identificação dos volumes vazios no empacotamento atual;
2. ranqueamento de cada par de caixa e espaço vazio;
3. alocação da caixa no espaço vazio.

Os índices do ranqueamento são:

1 caixa e vazio têm as três dimensões iguais

2 caixa e vazio têm duas dimensões iguais sendo a altura uma delas

3 caixa e vazio têm comprimento e largura iguais

4 caixa e vazio têm altura igual

5 caixa e vazio têm comprimento ou largura igual

6 caixa e vazio não tem nenhuma dimensão igual

Como critério de desempate no índice de ranqueamento, considera-se a maior proporção de volume ocupado pela caixa no espaço vazio.

Assim, temos a heurística `D3a3` considerando a altura (código 3) como direção predominante. Estamos estendendo o conceito de ranqueamento para criar as heurísticas `D3a2` e `D3a1` que têm como direção predominante a largura (código 2) e o comprimento (código 1), respectivamente, em lugar da altura. A heurística `D3a0` não tem direção predominante, de modo que os seus índices de ranqueamento são 1, 3, 5, 6.

A segunda heurística, que estamos propondo, considera para cada espaço vazio e para cada tipo de caixa, um bloco formado com o maior número de caixas que podem ser colocadas no espaço; não há rotação de caixas na construção de um bloco. O critério de ranqueamento é aplicado para estes blocos e espaços. Como critério de desempate também é utilizada a maior proporção de volume ocupado pelo bloco no espaço. Assim, temos as heurísticas `D3d3`, `D3d2`, `D3d1` considerando a altura, largura e comprimento como direção predominante, respectivamente, e a heurística `D3d0` que não tem direção predominante.

A terceira heurística, `D3b`, é proposta em [11] e é aplicável para problemas de empacotamento com poucas caixas distintas. Foi implementada uma versão com duas fases: na primeira, são construídas camadas ao longo de uma direção principal pré-definida que são formadas por um único tipo de caixa sem possibilidade de rotação. Na segunda, são inseridos blocos de um mesmo tipo de caixa em espaços vazios do empacotamento. A escolha do tipo de caixa a ser utilizada em cada camada é feita com base na proporção de preenchimento da camada: volume ocupado pelas caixas dividido pelo volume da camada. A escolha do tipo de caixa a ser utilizada em cada bloco é feita com base na proporção de preenchimento do vazio: volume ocupado pelas caixas dividido pelo volume do vazio. Assim, temos as heurísticas `D3b3`, `D3b2`, `D3b1` considerando a altura, largura e comprimento como direção de construção das camadas, respectivamente, e a heurística `D3b0` que não constrói camadas na primeira fase, apenas insere blocos na segunda fase.

A quarta heurística, `D3c`, que estamos propondo, também é aplicável para problemas de empacotamento com poucas caixas distintas. Ela trabalha em duas fases. Na primeira, a cada iteração, ela constrói uma camada de um único tipo caixa ao longo da melhor direção. Para cada direção (comprimento, largura e altura) e para cada tipo de caixa, ela determina o maior coeficiente de preenchimento de camada: proporção do volume ocupado pelas caixas divido pelo volume da camada. Na segunda fase, um bloco de caixas de um mesmo tipo é inserido em um espaço vazio. A escolha do tipo de caixa a ser utilizada em cada bloco é feita com base na proporção de preenchimento do vazio: volume

ocupado pelas caixas dividido pelo volume do vazio. Assim, temos a heurística `D3c0` que combina a construção de camadas na melhor direção possível.

Em todas as quatro implementações utilizamos a estrutura de dados apresentada em [14] para representação espacial das caixas alocadas no container. Além disto, estamos considerando a possibilidade de rotação das caixas. A rotação de 90^o ao longo do eixo vertical sempre é possível, mas a rotação ao longo de comprimento ou da largura somente é realizada quando é permitida explicitamente nos dados do tipo da caixa.

4 Testes computacionais

Os testes computacionais foram conduzidos em ambiente Matlab/Windows instalado em um microcomputador pentium IV com processador INTEL 1.7GHz e 392MB de memória RAM. Neste trabalho estamos relatando os testes computacionais efetuados com exemplares da literatura e procurando minimizar tanto o volume das caixas não-empacotadas quanto o volume não-empacotado do container.

As heurísticas testadas foram apresentadas na seção anterior e estão identificadas por `D3a0 ... D3c3`. Cada heurística produz um empacotamento e calcula os valores `pnf, pvf, pvo, pvs, cpu` que denotam, respectivamente, a proporção de caixas empacotadas (em quantidade), a proporção de caixas empacotadas (em volume), a proporção de caixas empacotadas em relação ao volume do container, a proporção de volume do container preenchido e o tempo de processamento (em segundos).

Os exemplares considerados nos testes foram extraídos do portal do ESICUP em `http://www.apdio.pt/esicup/`, EURO Special Interest Group on Cutting and Packing, sob a denominação de thpack.

Os resultados obtidos com os 100 exemplares do arquivo de dados `thpack1` estão apresentados na Tabela 1. Para este conjunto de dados, cada exemplar trabalha com 3 tipos de caixas e um container de dimensão $587 \times 233 \times 220$. Por causa da pequena variabilidade dos dados com respeito à direção principal utilizada nas heurísticas, somente são colocados nas figuras os valores correspondentes às heurísticas trabalhando sem direção principal (parâmetro 0 para `D3a, D3b, D3d`) na Figura 2. Os coeficientes `pnf, pvf, pvo, pvs` apresentaram médias similares para uma mesma heurística e o melhor resultado foi obtido com `D3c0`. Por outro lado, `D3b0` apresenta uma queda no valor médio de `pnf` em comparação com as outras medidas. Isto parece indicar que esta heurística não consegue manter bons resultados para a proporção de caixas empacotadas. Além disto, o valor médio de `pnf` apresenta uma variabilidade maior entre `D3b0, D3b1, D3b2, D3b3`; quase sempre o pior valor é de `D3b0` que não realiza a construção de camadas na primeira fase.

O mesmo procedimento foi adotado para os conjuntos de dados `thpack2`, `thpack3` e `thpack4`. Cabe ressaltar que cada conjunto de dados trabalha com a mesma dimensão para o container ($587 \times 233 \times 220$) enquanto que os número de tipos de caixas variam, a saber, 3 tipos de caixas para `thpack1`, 5 tipos

heurística	pnf	pvf	pvo	pvs	cpu
D3a0	0,85	0,86	0,86	0,86	11,54
D3a1	0,85	0,86	0,86	0,86	10,99
D3a2	0,85	0,86	0,86	0,86	10,77
D3a3	0,85	0,86	0,86	0,86	11,27
D3d0	0,84	0,85	0,85	0,84	8,15
D3d1	0,84	0,84	0,84	0,83	8,01
D3d2	0,84	0,85	0,85	0,84	8,07
D3d3	0,84	0,84	0,84	0,84	8,47
D3b0	0,79	0,84	0,84	0,84	0,34
D3b1	0,85	0,84	0,84	0,84	0,29
D3b2	0,81	0,82	0,82	0,82	0,35
D3b3	0,82	0,82	0,82	0,81	0,37
D3c0	0,88	0,88	0,88	0,87	0,40

Tabela 1: Valores médios para 100 exemplares do conjunto de dados `thpack1`

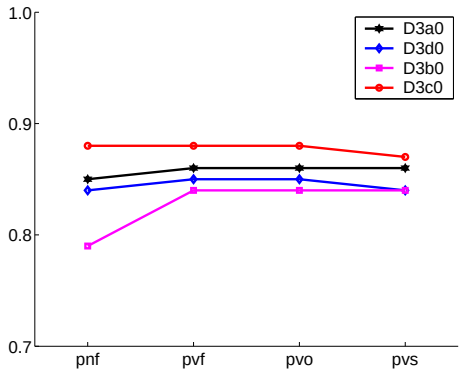

Figura 2: Resultados para `thpack1`

heurística	pnf	pvf	pvo	pvs	cpu
D3a0	0,84	0,87	0,87	0,87	12,30
D3a1	0,84	0,87	0,87	0,87	12,56
D3a2	0,84	0,87	0,87	0,87	12,46
D3a3	0,84	0,87	0,87	0,87	12,55
D3d0	0,84	0,84	0,84	0,83	10,07
D3d1	0,84	0,83	0,83	0,83	10,20
D3d2	0,84	0,83	0,84	0,83	10,59
D3d3	0,84	0,83	0,83	0,83	10,66
D3b0	0,79	0,85	0,85	0,84	0,75
D3b1	0,86	0,85	0,85	0,85	0,68
D3b2	0,80	0,82	0,83	0,82	0,78
D3b3	0,80	0,82	0,82	0,82	0,73
D3c0	0,89	0,89	0,89	0,88	0,58

Tabela 2: Valores médios para 100 exemplares do conjunto de dados `thpack2`

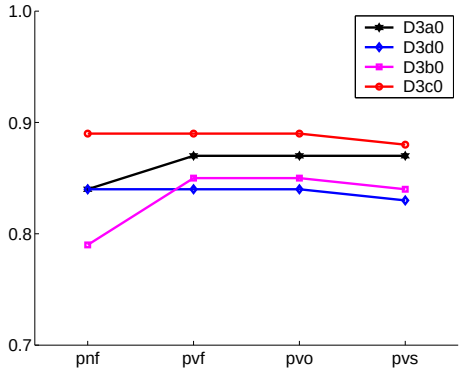

Figura 3: Resultados para `thpack2`

diferentes para `thpack2`, 8 para `thpack3` e, finalmente, 10 tipos distintos para `thpack4`.

Finalmente, a Tabela 5 e a Figura 6 apresentam os tempos de cpu médios das diversas heurísticas para resolver os problemas de `thpack1`, `thpack2`, `thpack3` e `thpack4`. Convém lembrar que o número de diferentes tipos de caixa para estes conjuntos são 3, 5, 8 e 10, respectivamente. Pode-se observar que `D3c` apresenta os menores tempos de cpu seguida de `D3b`, indicando que a construção de camadas é interessante para problemas com este tipo de estrutura. Note que `D3c` foi a mais rápida nos dados de `thpack2, thpack3, thpack4` mas não para `thpack1`.

Observações a respeito do número máximo de elementos dos vetores utilizados para marcar os comprimentos, larguras e alturas das células resultaram em: `com` ≤ 56, `lar` ≤ 40, `alt` ≤ 46. Mas em nenhum exemplar a matriz `ind`

heurística	pnf	pvf	pvo	pvs	cpu
D3a0	0,84	0,87	0,87	0,87	18,07
D3a1	0,83	0,87	0,87	0,87	18,10
D3a2	0,84	0,87	0,87	0,87	17,99
D3a3	0,84	0,87	0,87	0,87	18,35
D3d0	0,84	0,83	0,83	0,82	15,02
D3d1	0,83	0,81	0,81	0,81	14,47
D3d2	0,84	0,83	0,83	0,82	14,47
D3d3	0,84	0,83	0,83	0,83	15,33
D3b0	0,79	0,85	0,85	0,84	1,57
D3b1	0,86	0,85	0,85	0,85	1,61
D3b2	0,81	0,84	0,84	0,84	1,95
D3b3	0,77	0,80	0,80	0,79	1,54
D3c0	0,89	0,90	0,90	0,89	1,02

Tabela 3: Valores médios para 100 exemplares do conjunto de dados `thpack3`

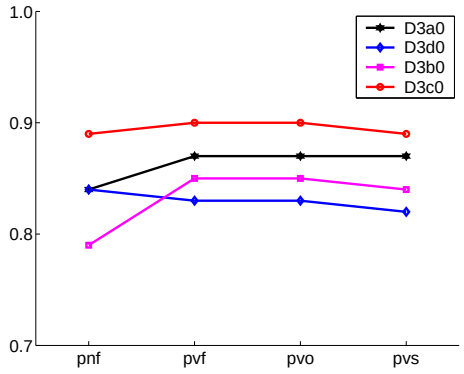

Figura 4: Resultados para `thpack3`

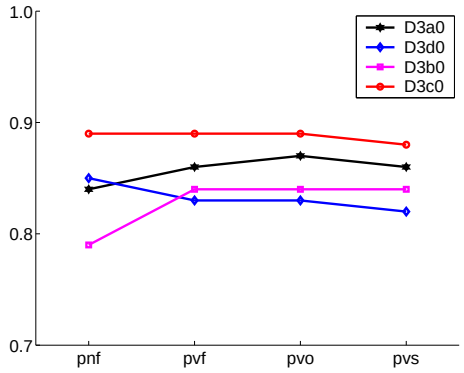

Figura 5: Resultados para `thpack4`

heurística	pnf	pvf	pvo	pvs	cpu
D3a0	0,84	0,86	0,87	0,86	20,47
D3a1	0,83	0,86	0,86	0,86	20,56
D3a2	0,84	0,86	0,87	0,86	20,60
D3a3	0,84	0,86	0,86	0,86	21,11
D3d0	0,85	0,83	0,83	0,82	17,31
D3d1	0,84	0,80	0,81	0,80	17,22
D3d2	0,85	0,82	0,82	0,82	16,60
D3d3	0,84	0,82	0,82	0,81	16,98
D3b0	0,79	0,84	0,84	0,84	2,35
D3b1	0,86	0,85	0,85	0,85	2,62
D3b2	0,81	0,84	0,84	0,84	2,74
D3b3	0,77	0,80	0,80	0,80	2,20
D3c0	0,89	0,89	0,89	0,88	1,33

Tabela 4: Valores médios para 100 exemplares do conjunto de dados `thpack4`

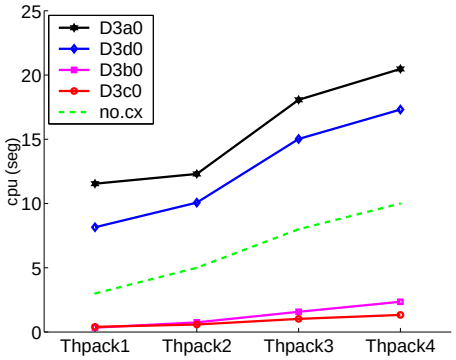

Figura 6: Tempos médios de cpu

heurística	thpack1(3)	thpack2(5)	thpack3(8)	thpack4(10)
D3a0	11,54	12,30	18,07	20,47
D3a1	10,99	12,56	18,10	20,56
D3a2	10,77	12,46	17,99	20,60
D3a3	11,27	12,55	18,35	21,11
D3d0	8,15	10,07	15,02	17,31
D3d1	8,01	10,20	14,47	17,22
D3d2	8,07	10,59	14,47	16,60
D3d3	8,47	10,66	15,33	16,98
D3b0	0,34	0,75	1,57	2,35
D3b1	0,29	0,68	1,61	2,62
D3b2	0,35	0,78	1,95	2,74
D3b3	0,37	0,73	1,54	2,20
D3c0	0,40	0,58	1,02	1,33

Tabela 5: Tempos médios de cpu

chegou a este número total de células.

Uma modificação interessante pode ser feita considerando a possibilidade de construção de camadas, ainda com um único tipo de caixa, mas permitindo a rotação de algumas caixas, de tal forma que a espessura da camada seja mantida. Isto pode aumentar a proporção de volume preenchido de cada camada em detrimento do tempo computacional.

Referências

[1] M. Arenales e R. Morabito, An and/or-graph approach to the solution of two-dimensional non-guillotine cutting problems, *European Journal of Operational Research*, **84** (1995) 599-617.

[2] M. Biró e E. Boros, Network Flows and non-guillotine cutting patterns, *European Journal of Operational Research*, **16** (1984) 215-221.

[3] E.E. Bishoff e W.B. Downsland, An application of the micro to product design adn distribution, *J. Opl.. Res. Soc.*, **33** (1982) 271-280.

[4] E.E. Bishoff e M.D. Marriott, A comparative evaluation of heuristics for container loading, *European Journal of Operational Research*, **44** (1990) 267-276.

[5] C.S. Chen, S.M. Lee e Q.S. Chen, An analytical model for the container loading problem, *European Journal of Operational Research*, **80** (1995) 68-76.

[6] H. Dyckhoff, A typology of cutting and packing problems, *European Journal of Operational Research*, **44** (1990) 145-159.

[7] H. Dyckhoff e U. Finke, *Cutting and Packing in Production and Distribuition.* Springer-Verlag Co., Heidelberg, Germany (1992).

[8] P. Erdös e R.L.Graham, On packing squares with equal squares, *J. Combinatorial Theory Series A*, **19** (1975) 119-123.

[9] P.C. Gilmore e R.E. Gomory, A linear programming approach to the cutting stock problem, *Operations Research*, **9** (1961), 849-859.

[10] P.C. Gilmore e R.E. Gomory, A linear programming approach to the cutting stock problem – part ii, *Operations Research*, **11** (1963), 863-888.

[11] J.A. George e D.F. Robinson, A heuristic for packing boxes into a container, *Comput & Ops. Res.*, **7** (1980) 147-156.

[12] T.J. Hodgson, A combined approach to the pallet loading problem, IIE Transactions, **14** (1982) 175-182.

[13] R. Morabito e M. Arenales, An and/or-graph approach to the container loading problem, *International Transactions on Operational Research*, **1** (1994), 59-73.

[14] B.K.A. Ngoi, M.L. Tay e E.S. Chua, Applying spatial representatation techniques to the container paking problem, *Int. J. Prod. Res*, **32** (1994), 111-123.

[15] D. Pisinger, Heuristics for the container loading problem, *European Journal of Operational Research*, **141** (2002) 382-392.

[16] R.D. Tsai, E.M. Malstrom e W. Kuo, Three dimensional palletization of mixed box sizes, *IIE Transaction*, **25** (1993) 64-75.