Uma Estatégia de Geração de Colunas para o Problema de Empacotamento Bidimensional

Clovis Perin
Valéria de Podestá Gomes
Departamento de Matemática Aplicada,
Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas
Campinas, SP, Brasil

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. Neste trabalho, foi utilizado um ambiente computacional de modelagem de problemas de programação matemática, OPL-Studio (*Optimization Programming Language*), que é composto por um pacote para enumerar os padrões de empacotamento (*solver*) e outro para resolver programas matemáticos (*cplex*). São estudados modelos alternativos para a geração de conjugações necessárias para a definição explícita do problema. Em geral, o problema real não é modelado diretamente e é necessário gerar um conjunto de padrões de empacotamento para construir um modelo de programação matemática. Testes computacionais foram realizados para comparar tempos de processamento e o número de padrões de empacotamento obtidos para exemplares de problemas de empacotamento bidimensionais gerados aleatoriamente.

Abstract. Loading boxes into pallets and containers constitute important applications of mathematical models of cutting and packing problems on production planning in industries of paper, steel, wood, glass, etc. In this work, we use a computational environment for mathematical programming modelling, OPL-Studio (*Optimization Programming Language*), which is composed by a software to enumerate the packing patterns (*solver*) and another one to solve the mathematical programs (*cplex*). In general, the real problem is not formulated directly and it is necessary to generate the set of packing patterns to construct the mathematical programming model. Alternate models were considered for packing patterns generation, a necessary step to explicitly formulate the problem. Computational tests were performed to compare the processing time and the number of packing patterns obtained with instances of two-dimensional packing problems randomly generated.

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 manipulação, 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 paletes é classificado como um problema de empacotamento retangular bidimensional e é relacionado com o problema bidimensional de corte de estoque.

No problema de corte de estoque, a ênfase é no corte de uma placa retangular plana de estoque em diversas peças menores de dimensões conhecidas. Este problema aparece em indústrias de manufatura no corte de tecido, vidro, papel, madeira, etc.. O objetivo usual é minimizar as perdas. Neste caso, os cortes são do tipo guilhotina, que partem de uma aresta da peça de estoque e correm paralelos a outras duas arestas.

No problema de empacotamento bidimensional são utilizados cortes não-guilhotinados. Estes problemas podem ser de dois tipos [9]: 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 paletes e tem por objetivo maximizar o volume empacotado. O outro consiste em produzir e empacotar os produtos em caixas idênticas e empacotá-las em paletes 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 bidimensional do tipo $2/V/I/C$, conforme tipologia proposta em [4, 5], 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 [6].

Estes problemas costumam ser definidos implicitamente, pois é difícil construir um arquivo em formato MPS (Mathematical Programming System) com todos os coeficientes do programa matemático. As colunas destes programas matemáticos costumam ser construídas com um trabalho combinatorial extra para que o exemplar fique caracterizado por completo. Assim, uma solução fica bem definida por um vetor de pares representando a especificação do padrão de corte (número de vezes de cada largura no padrão) e a quantidade a ser cortada deste padrão de corte.

De modo geral, são problemas de difícil resolução, que demandam tempo e memória computacional, principalmente quando o número de padrões de empacotamento necessários para implementar a solução é crucial para a carga que o equipamento de corte deve operar.

Neste trabalho, consideramos o problema de empacotamento que determina o número ótimo de paletes para empacotar um dado conjunto de peças com

dimensões distintas. O objetivo é minimizar o espaço total não utilizado. É feito um estudo computacional utilizando a linguagem de modelagem de problemas de programação matemática OPL *(Optimization Programming Language)*. Esta linguagem integra um ambiente computacional que incorpora um pacote de *constraint programming* para enumeração dos padrões de empacotamento (denominado *Solver*) e um pacote de resolução de programas matemáticos (CPLEX).

Na literatura científica, há duas estratégias básicas para resolver problemas de corte e empacotamento.

A primeira é devida ao modo implícito como o problema costuma ser posto. Pode-se dizer que é uma forma natural de trabalhar com problemas de programação matemática definidos implicitamente. Inicialmente, as colunas ou padrões de empacotamento são construídos explicitamente para montar o problema por completo. Posteriormente, aplica-se um método de programação linear/inteira para obter a solução ótima.

A segunda estratégia, proposta por Gilmore e Gomory[7, 8] para problemas lineares e fracionários do tipo corte, trabalha com uma extensão do método simplex: uma coluna, que representa um padrão de empacotamento, é gerada por um problema secundário, do tipo problema da mochila, cuja função objetivo depende em sua essência do custo da coluna a ser gerada e dos multiplicadores simplex. O objetivo do problema secundário representa, portanto, o custo relativo da coluna gerada, que se for negativo indica que ela efetivamente deve entrar na base (minimização). Infelizmente, esta estratégia não pode ser aplicada diretamente para problemas com variáveis de decisão inteiras.

O objetivo deste trabalho é testar o comportamento de um pacote computacional para resolver problemas de empacotamento bidimensional para diferentes tamanhos de exemplares gerados aleatoriamente. É proposta uma estratégia de *constraint programming* a partir do estudo modelos alternativos para a geração dos padrões de empacotamento necessários para a definição explícita do problema.

Na seção 2 deste artigo apresentamos a descrição do modelo, na seção 3, os testes computacionais realizados com exemplares gerados aleatoriamente com peças retangulares e quadradas, e na seção 4, as considerações finais.

2 Modelo de Empacotamento

Considere o problema de corte bidimensional não-guilhotinado, definido por uma carteira com m pedidos de peças retangulares com larguras ℓ_i e alturas h_i, em quantidades b_i, a serem empacotadas em objetos retangulares com 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 peça i é considerada no padrão de empacotamento j. Isto implica que $\sum_i a_{ij}\ell_i h_i \leq LH$. Deseja-se determinar um n-vetor de quantidades $w = (w_j)$, solução ótima do programa matemático:

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

Neste modelo, é possível representar diversas funções objetivo que não a quantidade perdida de material, como por exemplo, quantidade produzida ($f_j = 1$), perda ($f_j = LH - \sum_i a_{ij}\ell_i h_i$), custo, tempo de produção, 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 (linear) inteiro.

Alguns artifícios costumam ser utilizados para reduzir o tempo computacional através da redução do número de padrões de empacotamento. Por exemplo, para problemas com pedidos que podem ser satisfeitos com folgas (restrições do tipo $\geq$) é possível trabalhar apenas com padrões de empacotamento maximais, isto é, padrões cujos pedaços de material perdido são menores do que qualquer das peças demandadas. Isto tende a produzir uma quantidade maior das peças pequenas, de tal forma que o estoque destas peças, em geral, pode aumentar, mas a perda de material ainda é minimizada. Outro artifício é aplicável em situações denominadas de demanda pequena. Neste caso, as peças são contabilizadas individualmente, e nenhum padrão de empacotamento com um número de repetições de uma peça superior ao pedido desta peça é gerado.

Uma solução inicial muito utilizada é construída apenas com m padrões de empacotamento homogêneos, isto é, cada padrão de empacotamento produz um único tipo de peça.

Deve-se ressaltar que é comum o uso de estratégias heurísticas para estes problemas.

2.1 Geração de padrões

A resolução de um problema de empacotamento, em geral, envolve a geração de padrões de empacotamento. Vamos utilizar pares de variáveis de decisão (x_i, y_i) para denotar as coordenadas do canto inferior esquerdo da peça i no padrão de empacotamento. Assim, a matriz de restrições A deve ser construída de tal modo que cada peça $\ell_i \times h_i$ seja alocada inteiramente no objeto $L \times H$ e deve satisfazer as condições

$$x_i + l_i \leq L \qquad \text{e} \qquad y_i + h_i \leq H$$

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

$$\begin{aligned} x_i + l_i &\leq x_j \quad \text{ou} \\ x_j + l_j &\leq x_i \quad \text{ou} \\ y_i + h_i &\leq y_j \quad \text{ou} \\ y_j + h_j &\leq y_i \end{aligned} \tag{2}$$

cuja implementação, em modelos de programação matemática, é construída com a inclusão de variáveis binárias. Entretanto, isto não é necessário para modelos de *constraint programming*, de modo que vamos manter esta formulação. Neste

caso, é interessante considerar restrições redundantes para auxiliar a busca das soluções, reduzindo o espaço de busca. Um conjunto efetivo de tais restrições é obtido considerando uma reta horizontal traçada em uma altura q do objeto $L \times H$, como podemos observar na Figura 1. Esta reta horizontal corta algumas das peças alocadas e a soma destes cortes horizontais é menor ou igual à largura da placa L. Assim,

$$\sum_{i \,:\, y_i \leq q,\ q < y_i + h_i} l_i \leq L \quad \forall\ q \in [0, H] \tag{3}$$
$$\sum_{i \,:\, x_i \leq p,\ p < x_i + l_i} h_i \leq H \quad \forall\ p \in [0, L]$$

O segundo grupo de restrições está associado a retas verticais traçadas na largura p do objeto.

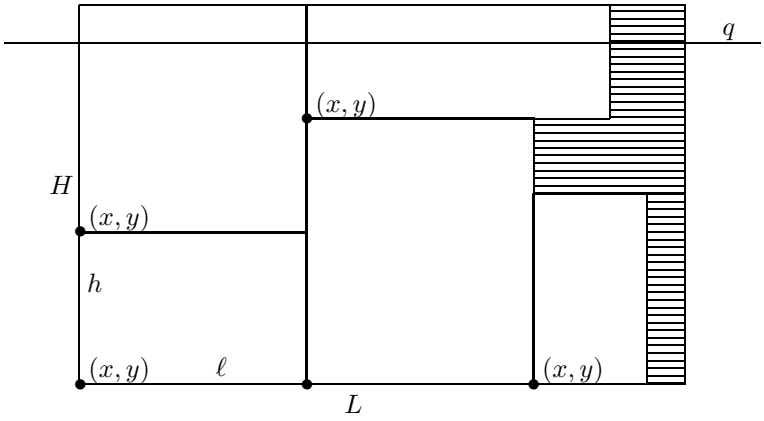

Figura 1: Empacotamento com 5 peças em um objeto LH

A estratégia de *constraint programming* proposta consiste de uma regra de enumeração implícita das peças para compor padrões de empacotamento. Para cada padrão com uma solução factível do modelo 2 é montada uma coluna da matriz de restrições de 1. A regra de enumeração implícita evita testar um padrão de empacotamento com as peças $p_1, p_2, ..., p_k, p_{k+1}$ sempre que um empacotamento com as peças $p_1, p_2, ..., p_k$ já tiver sido verificado ser inviável.

2.2 Outros Modelos

A literatura científica registra diversos modelos de programação inteira mista 0-1 para empacotamento bidimensional/tridimensional que eventualmente consideram a possibilidade de rotação das peças que compõem a carteira de pedidos.

Em [3] é proposto um modelo adequado para ser utilizado na estratégia interativa de Gilmore e Gomory [7, 8] pois são alocadas todas as peças 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. Esta seleção é feita para otimizar o valor das peças selecionadas. O modelo utiliza variáveis para fixar o canto inferior esquerdo frontal de cada peça (variáveis (x_i, y_i) no caso bidimensional).

Em [14] é proposto um outro modelo com restrições sobre o número de peças de um mesmo tipo na solução. O modelo tem por objetivo utilizar o menor número total de objetos e utiliza as mesmas variáveis (x_i, y_i) para fixar o canto inferior esquerdo de cada peça i selecionada.

Modelos de busca em grafo do tipo E/OU são propostos 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.

Em [9] é proposta uma abordagem híbrida, considerando uma combinação de programação dinâmica e heurísticas, numa situação que envolve algumas peças 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 Testes computacionais

Os testes computacionais foram conduzidos em ambiente OPL-Studio/Windows instalado em um microcomputador pentium IV com processador INTEL 1.7GHz e 392MB de memória RAM. O pacote OPL Studio (Optimization Programming Language) [12] trabalha associado com os *solvers* CPLEX (para programação linear e inteira) [10] e *Solver* (para *constraint programming*) [11]. Estes *solvers* foram utilizados com as suas opções de *default* que englobam o método simplex dual para a programação linear no CPLEX e a regra de melhor limitante para o método *branch-and-bound* também no CPLEX. Neste trabalho estamos relatando os testes computacionais efetuados com exemplares gerados aleatoriamente e utilizando a função objetivo que minimiza a produção total $min \sum_j w_j$, onde w_j é o número de paletes do j-ésimo padrão de empacotamento.

Todos os testes relatados neste trabalho foram efetuados com exemplares com paletes de dimensões $(L, H) = (20, 20)$, com m peças ($m = 5, 10, 15, 20, 25$) e com peças de dimensões inteiras (ℓ_i, h_i), geradas aleatoriamente em $\ell_i \in [L/4, 3L/4]$, $h_i \in [L/4, 3L/4]$. Na subseção 3.1 relatamos os resultados observados com apenas um exemplar, enquanto nas demais subseções, para cada valor de m, relatamos as médias de tempo computacional e de número de padrões de empacotamento para 10 exemplares.

3.1 Restrições Redundantes

Um certo exemplar com $m = 15$ peças apresentou $n = pad = 51$ padrões de empacotamento, objetivo $obj = 51$ e tempos de cpu, em segundos, de acordo com a Tabela 1. Foram considerados dois modelos de *constraint programming*, o primeiro, definido pelas restrições [2] e o segundo, definido por [2, 3], que inclui restrições redundantes. Pode-se observar que o tempo computacional obtido com o modelo com mais restrições é menor. Resultados similares foram obtidos com outros exemplares, de modo que nos demais experimentos utilizamos sempre as restrições redundantes.

restrições	obj	pad	cpu
2	51	87	1,1
2, 3	51	87	0,6

Tabela 1: Resultados para diferentes modelos

3.2 Peças Retangulares

A Tabela 2 apresenta os valores médios de tempo de cpu em segundos (*cpu*) e de número de padrões de empacotamentos (*pad*), obtidos para grupos de 10 exemplares gerados aleatoriamente com $m = 5, 10, 15, 20, 25$ peças. Pode-se verificar um comportamento de crescimento exponencial para *pad* e *cpu* em função do número m de peças.

m	pad	cpu
5	16	0,1
10	148	2,5
15	790	35
20	3437	338
25	11102	3813

Tabela 2: Resultados para peças retangulares

3.3 Peças Quadradas

A Tabela 3 apresenta os valores médios de tempo de cpu em segundos (*cpu*) e de número de padrões de empacotamentos (*pad*), obtidos para grupos de 10 exemplares gerados aleatoriamente com $m = 5, 10, 15, 20, 25$ peças. Para gerar exemplares com peças quadradas foi atribuído $\ell_i = h_i$. Pode-se notar um comportamento de crescimento exponencial para *pad* e *cpu* em função do número m de peças, corroborando com a discussão da seção anterior.

m	pad	cpu
5	12	0,1
10	90	1,8
15	588	151
20	2527	2603
25	7982	14789

Tabela 3: Resultados para peças quadradas

3.4 Peças Quadradas Ordenadas

A Figura 2 foi construída com os resultados apresentados nas Tabelas 2 e 3, para peças retangulares e quadradas. Pode-se verificar que, para o mesmo número de peças, a média do número de padrões gerados com peças quadradas tende a ser menor do que para as peças retangulares, contrastando com o tempo computacional, cuja média tende a ser maior para as peças quadradas.

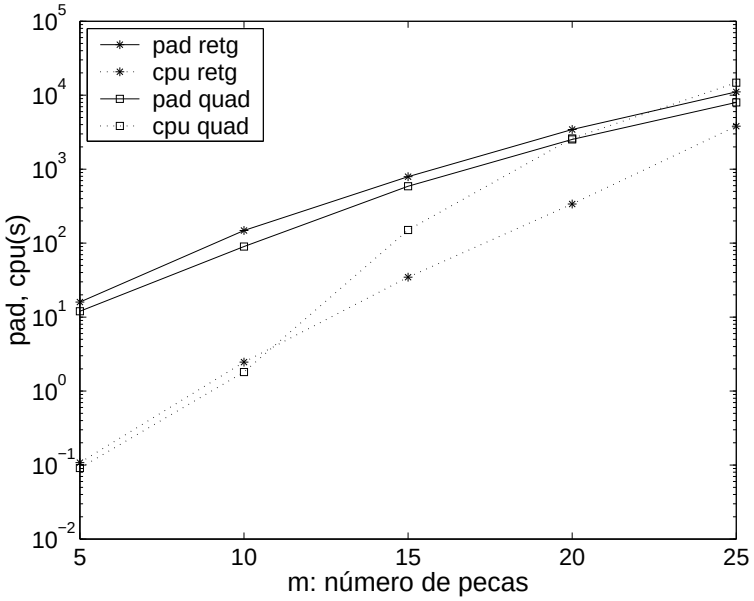

Figura 2: Número de padrões gerados e tempo de cpu em segundos para peças retangulares e quadradas

A Tabela 4 apresenta resultados obtidos para gerar todos os padrões de empacotamento em paletes com $(L, H) = (20, 20)$ e $m = 15$ peças de dimensões (ℓ_i, h_i) geradas aleatoriamente em $\ell_i, h_i \in [L/4, 3L/4]$ em quantidades $b_i \in [0, 20]$ relativos ao uso da mesma estratégia de solução aplicada a um exemplar

com peças do tipo (1) retângulos, (2) quadrados sem ordenação, (3) quadrados em ordem crescente de ℓ_i e (4) quadrados em ordem decrescente de ℓ_i. Os dados destes três últimos exemplares são os mesmos e iguais às larguras dos retângulos. A altura dos retângulos foi gerada de modo independente. Foi anotado o valor da solução (*obj*), o número de padrões gerados (*pad*), o número de padrões recusados (*padr*) e os tempos de cpu em segundos gastos no total (*cpu*), na verificação dos padrões aceitos (*cpug*) e verificação dos padrões recusados (*cpur*). Deve-se salientar que $cpug+cpur \geq cpu$ pois o tempo de *cpu* total inclui também o tempo de resolução do programa linear para obter s solução ótima.

Pode-se observar que a verificação dos padrões aceitos tende a ser inferior em quantidade e em tempo de cpu ao dos padrões recusados, exceto para os quadrados postos em ordem crescente. Isto é influenciado pela ordem adotada pela regra de enumeração das peças: nenhuma seqüência de peças é testada se ela apresenta uma subseqüência inicial associada a um padrão recusado.

Por outro lado, as duas últimas colunas da tabela indicam que o número médio de padrões verificados por segundo, para padrões gerados ($[p/s]_g$) é superior ao de padrões recusados ($[p/s]_n$). Ou seja, é mais rápido testar se uma seqüência de peças forma um padrão de empacotamento quando ela corresponde realmente a um padrão.

tipo	padg	padn	cpu	cpug	cpun	(p/s)g	(p/s)n
retângulos	1627	2386	126,48	7,03	83,41	231	29
quadrados sem ordem	1063	2057	304,07	4,03	299,92	264	7
quadrados crescentes	1063	253	90,85	4,45	86,33	239	3
quadrados decrescentes	1063	4368	343,80	3,99	399,73	267	13

Tabela 4: Resultados com peças quadradas ordenadas

4 Considerações Finais

As tentativas de trabalhar com um modelo de geração de padrões de empacotamento utilizando uma função objetivo em conjunto com a abordagem de Gilmore e Gomory não deram resultados satisfatórios. Embora o pacote de *constraint programming* permita incluir uma função objetivo, neste particular experimento, o tempo computacional crescia muito. Em um outro experimento, não discutido neste trabalho, para o corte bidimensional guilhotinado, a abordagem de Gilmore e Gomory apresentou resultados melhores.

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] 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.

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

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

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

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

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

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

[10] ILOG CPLEX 7.1, User's Manual, ILOG, France, 2001.

[11] ILOG SOLVER 5.1, User's Manual, ILOG, France, 2001.

[12] ILOG OPL Studio 3.5: *The Optimization Language.* ILOG, France, 2001.

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