TEMA Tend. Mat. Apl. Comput., **4**, No. 2 (2003), 217-226.

Uma Experiência com Geração de Colunas em Problemas de Corte de Estoque[1]

C. PERIN, V. PODESTÁ GOMES, A.C. MORETTI[2], Departamento de Matemática Aplicada, Instituto de Matemática, Estatística e Computação Científica, UNICAMP, 13083-970 Campinas, SP, Brasil.

Resumo. Problemas de Corte são importantes modelos matemáticos para o planejamento da produção em indústrias de papel, aço, madeira e vidro. 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 corte (*Solver*) e outro para resolver programas matemáticos (CPLEX). Foram estudados modelos de carga fixa (*fixed charge*) cujos objetivos representam a soma de tempos de montagem (*setup*) com tempos de produção. Apresentamos testes computacionais comparando tempos de processamento e qualidade da solução obtida para exemplares de problemas de corte unidimensionais gerados aleatoriamente. Verificamos que o uso de heurísticas para gerar os padrões de corte é uma importante ferramenta para problemas grandes.

1. Introdução

Problemas de Corte e Empacotamento são modelos matemáticos de grande importância no planejamento da produção em indústrias de papel, aço, madeira e vidro, e no planejamento do empacotamento em transporte rodoviário, aéreo, marítimo, etc. Um problema de corte unidimensional consiste em cortar de maneira ótima um grupo de barras pequenas (itens) a partir de barras grandes (ou estoque). Um grande número de estratégias de modelagem (orientadas para itens, orientadas para cortes) e algoritmos já foram propostos. Neste trabalho vamos estudar problemas de corte unidimensional do tipo $1/V/I/M$, conforme tipologia proposta em [1, 2], que classifica os problemas por dimensionalidade, tipo de associação entre itens e estoque, tipo de estoque e tipo de itens.

Tais problemas costumam ser caracterizados implicitamente, pois é difícil construir um arquivo em formato MPS (Mathematical Programming System) com todos os coeficientes do programa matemático. Em geral, as colunas de tais programas matemáticos costumam ser construídas com um trabalho combinatorial extra para que o exemplar fique definido por completo. Desta forma, uma solução deve ser

[1]Apoio financeiro parcial da FAPESP, proc. no. 2001/102972-2
[2]clovis@ime.unicamp.br, valeria@ime.unicamp.br, moretti@ime.unicamp.br

apresentada como um vetor de pares do tipo: 1) especificação do padrão de corte (número de vezes de cada largura no padrão) e 2) quantidade a ser cortada deste padrão de corte.

Há situações para as quais estes problemas se tornam difíceis ou de resolução demorada. Isto engloba situações em que o tempo de *setup* – ajuste ou montagem de máquinas – é grande quando comparado com o tempo de produção propriamente dito. Também engloba situações em que o próprio número de *setups* constitui um gargalo da produção.

Duas estratégias básicas para resolver problemas de corte são apresentadas na literatura e são consideradas neste estudo.

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

A segunda estratégia, proposta por Gilmore e Gomory[5, 6] para problemas lineares e fracionários do tipo corte, trabalha com uma extensão do método simplex: uma coluna, que representa um padrão de corte, é gerada por um problema secundário (uma mochila unidimensional, bidimensional ou algum outro problema de otimização), 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).

Problemas de minimização de *setups* foram estudados por diversos autores. McDiarmid [11] chama a atenção de que se trata de um problema NP-hard e propõe uma heurística baseada em subconjuntos com pedidos balanceados.

Farley e Richardson [3] sugerem um procedimento que procura um compromisso entre o número de larguras e o número de padrões de corte distintos. O método inicia com a solução linear relaxada de perda mínima. Então, às custas de um material adicional, o número de variáveis básicas relacionadas aos padrões de corte é reduzido em uma série de iterações simplex, enquanto existir solução factível. Existem duas dificuldades nesta heurística: não há indicação de como decidir quais padrões de corte devem ser anulados e não há garantia de soluções inteiras.

Haessler [7] propõe uma solução heurística de compromisso: redução do número de padrões de corte às custas do aumento da quantidade de material. Ela é organizada em função de uma busca sequencial que relaciona os pedidos ainda não atendidos e as perdas para determinar o próximo padrão de corte.

Foerster e Wäscher [4] propõem uma heurística que pode ser considerada como uma generalização de outras. Ela faz uso do fato de que a soma dos pedidos de um conjunto de padrões de corte ativos tem que ser idêntica à soma dos pedidos de um conjunto substituto de padrões de corte para que a perda de material seja a mesma.

O objetivo deste trabalho é testar o comportamento de um pacote computacional para resolver problemas de corte unidimensional variando o tipo da função objetivo e o tipo das restrições.

Na seção 2 deste artigo apresentamos a descrição do modelo, na seção 3, os testes computacionais com limitantes, tempos e número de *setups* e crescimento do problema e, na seção 4, as considerações finais.

2. Descrição do modelo

Considere o problema de corte unidimensional definido por uma carteira com m pedidos de larguras ℓ_i nas quantidades b_i a serem cortadas de barras de comprimento L. Suponha que há n padrões de corte 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 largura i é considerada no padrão de corte j. Isto implica que $\sum_i a_{ij}\ell_i \leq L$ e o valor $L - \sum_i a_{ij}\ell_i$ é a perda unitária de material produzida pelo j-ésimo padrão de corte. Deseja-se determinar um n-vetor de quantidades $x = (x_j)$, solução ótima do programa matemático:

$$min\{ \sum_j f_j(x_j) : Ax \approx b,\ x \in \mathcal{X} \}$$

Neste modelo, $f_j(x_j)$ representa quantidade produzida, quantidade perdida de material, custo ou tempo associado ao padrão de corte j, que pode envolver tanto *setup* quanto produção propriamente dita. O símbolo $\approx$ deve ser substituído por um dos símbolos $=$ ou $\geq$, dependendo de como a carteira de pedidos deve ser atendida. O conjunto $\mathcal{X}$ pode tanto representar o n-produto cartesiano dos reais não-negativos $\Re_+^n$ como dos naturais $\mathcal{N}^n$.

Alguns artifícios costumam ser utilizados para reduzir o tempo computacional através da redução do número de padrões de corte. Por exemplo, para problemas com pedidos que podem ser satisfeitos com folgas (restrições do tipo $\geq$) é possível trabalhar apenas com padrões de corte maximais, isto é, padrões com perda de material inferior à menor das larguras. Isto tende a produzir uma quantidade maior dos ítens pequenos, de tal forma que o estoque destes ítens, em geral, pode aumentar, mas a perda de material é minimizada. Outro artifício é aplicável quando os ítens são contabilizados individualmente, de tal forma que nenhum padrão de corte pode ser construído com número de repetições de uma largura superior ao pedido desta largura.

Uma solução inicial muito utilizada é construída apenas com m padrões de corte homogêneos, isto é, padrões de corte definidos para cortar uma única largura.

Diversas heurísticas costumam ser consideradas para problemas de programação inteira (vide [12]). Há heurísticas baseadas no método de *best-fit-decreasing* para problemas de *bin packing*, assim como de redução sucessiva dos pedidos através do relaxamento linear do problema reduzido seguido do uso do piso (maior inteiro menor ou igual) da solução relaxada para obter uma solução inteira.

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) [8] trabalha associado com os *solvers* CPLEX (para programação linear e inteira) [9] e *Solver* (para *constraint programming*) [10]. 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.

Um experimento conduzido com 10 exemplares de $m = 15$ larguras inteiras ℓ_i geradas aleatoriamente em {1,2,...,100} para serem cortadas de barras de largura $L = 100$ produziu exemplares com $min(n) = 63$ a $max(n) = 63544$ colunas e média $med(n) = 7762,70$. Problemas do tipo minimizar produção com restrições que podem ser satisfeitas com folgas não-negativas podem ser trabalhados apenas com padrões de corte maximais (a largura do material perdido no padrão de corte é inferior à menor das larguras dos pedidos). Neste caso, os 10 problemas gerados aleatoriamente produziram $min(n) = 11$ a $max(n) = 2984$ colunas maximais e média $med(n) = 490,50$.

Ao considerar barras de largura $L = 100$ para serem cortadas em $m = 15$ larguras inteiras ℓ_i geradas aleatoriamente em {26,27,...,75}={$L/4$,...,$3L/4$} foram obtidos 10 exemplares com $min(n) = 49$ a $max(n) = 168$ colunas e média $med(n) = 82,50$. Considerando apenas padrões de corte maximais, o número de colunas ficou entre $min(n) = 31$ e $max(n) = 114$ e média $med(n) = 57,50$. A tentativa de gerar larguras em $(0, L/2)$ produziu exemplares com um número de colunas demasiadamente grande. Note que gerar larguras em $(L/2, L)$ produz exemplares com $n = m$ colunas sempre. Analisando estes resultados, optamos por trabalhar com colunas geradas em $(0, L)$.

3.1. Limitantes

Inicialmente foi feito um estudo para obter limitantes para o valor da solução ótima do modelo de programação inteira $\min\{c^T x : Ax \approx b,\ x \geq 0\}$ onde c tanto pode representar um vetor de quantidades produzidas com um vetor de quantidades perdidas. Consideramos quatro exemplares de problemas inteiros construídos com as características abaixo combinadas:

- minimizar *produção* versus *perda*
- satisfazer pedidos sem folgas (=) versus com folgas ($\geq$)

A Tabela 1 apresenta os resultados obtidos com a resolução de um problema inteiro gerado com $m = 15$ larguras inteiras ℓ_i em {1,2,...,100} e pedidos inteiros b_i em {1,2,...,100}, totalizando $\sum_i b_i = 37594$. As colunas pedido, min, prod, perda, folga, n_+, n, cpu, itr apresentam, respectivamente, a soma $\sum_i b_i$ dos pedidos, a função objetivo a ser minimizada, a produção total $\sum_j x_j$ da solução ótima encontrada, a perda total com a produção $\sum_j c_j x_j$, a folga total $\sum_j x_j - \sum_j c_j x_j - \sum_i b_i$, o número de variáveis positivas $\sum_j [x_j : x_j > 0]$ na solução, o número de padrões de corte considerados, o tempo de cpu necessário para obter a solução e o número de iterações do método simplex dual (opção de *default* do pacote utilizado). Deve-se ressaltar que o número de colunas n geradas para obter a solução exata deste exem-

plar é bem maior do que a de outros exemplares de mesmo porte e características similares. A primeira linha da tabela sumariza os dados da solução ótima que minimiza a quantidade total produzida $\sum_j x_j = 44100$, que é a soma das parcelas de perda com folga e pedido. A segunda linha apresenta uma solução ótima que minimiza a perda $\sum_j c_j x_j = 284$, que corresponde a uma perda relativa de $\sum_j c_j x_j / \sum_j x_j = 284/46900 = 0,606\%$.

É interessante notar que a primeira solução pode ser implementada com $n_+ = 16$ *setups* e a segunda com $n_+ = 13$ *setups*. Trata-se do número de variáveis com valor não nulo dentre os $n = 216$ padrões de corte maximais considerados. Tais padrões não permitem uma perda de material maior do que a menor das larguras da carteira de pedidos. Convém ressaltar que o tempo de resolução do exemplar sem folgas ($Ax = b$) foi bem superior, inclusive, este último teve um tempo de processamento superior a 24 horas (*).

≈ pedido	min	prod	perda	folga	n_+	n	cpu	itr
≥ 37594	prod	44100	304	6202	16	216	0,16	19
	perda	46900	284	9022	13	216	0,06	9
= 37594	prod	44100	6506	0	17	2687	1,51	6
	perda					2687	*	*

Tabela 1: Resultados para a solução exata (inteira)

Um limitante inferior para o valor da solução ótima do modelo de programação inteira pode ser obtido com a relaxação linear das variáveis inteiras em variáveis reais, mantendo a restrição de não-negatividade delas. Um limitante superior (primeira solução heurística) para o problema inteiro com restrições $Ax \geq b$ é obtido ao forçar cada variável de decisão relaxada para o teto (menor inteiro maior ou igual ao valor) do seu valor. O número de colunas geradas pelo programa linear foi $n_G = 32$ das quais $n_+ = 16$ têm valor positivo na solução heurística e $n_+ = 14$ têm valor positivo na solução relaxada. A Tabela 2 apresenta estes valores (vários estão arredondados para valores inteiros). Pode-se observar que a perda com a solução exata é superior à da solução linear e que o número de *setups* (n_+) necessários para implementar a solução também é maior. Infelizmente, esta heurística de arredondamento não obteve sucesso para a última aplicação, pois ela produziu uma solução inteira com folgas nas restrições dos pedidos.

Uma proposta de um limitante superior para o valor da solução ótima do modelo de programação inteira consiste em considerar apenas os padrões de corte gerados pelo método de Gilmore-Gomory para a solução linear (contínua). Para testar esta proposta contabilizamos n_G, o número de colunas geradas pelo programa linear onde n_+ têm valor positivo (vide Tabela 3). Os exemplares com restrições sem folgas são infactíveis (infac).

Em resumo, as heurísticas estudadas não apresentaram um comportamento satisfatório para problemas com pedidos sem folgas ($Ax = b$). Entretanto, os casos estudados não são suficientes para afirmar que elas não tem um comportamento

relaxamento linear

≈ pedido	min	prod	perda	folga	n_+	n	cpu
≥ 37594	prod	44050	305	6151	14	216	0,03
	perda	45978	284	8100	13	216	0,04
= 37594	prod	44050	6456	0	15	2687	1,38
	perda	44050	6456	0	15	2687	1,43
solução heurística							
≥ 37594	prod	44100	5107	1399	14	216	$0{,}03^+$
	perda	46100	284	8222	13	216	$0{,}04^+$
= 37594	prod	44300	6498	0	15	2687	$1{,}38^+$
	perda	44200	6469	137	15	2687	$1{,}43^+$

Tabela 2: Resultados para a primeira solução aproximada (inteira)

relaxamento linear

≈ pedido	min	prod	perda	folga	n_+	n_G	cpu
≥ 37594	prod	44050	5104	1352	13	16	0,03
	perda	45300	284	7422	13	27	0,02
= 37594	prod	44050	6456	0	15	18	0,03
	perda	44050	284	0	15	18	0,04
solução heurística							
≥ 37594	prod	44100	5107	1399	13	16	0,06
	perda	45400	284	7422	13	27	0,23
= 37594	prod					18	infac
	perda					18	infac

Tabela 3: Resultados para a segunda solução aproximada (inteira)

satisfatório para problemas com folgas.

3.2. Tempos de *setup*

Foi realizado um estudo do tempo (ou custo) total de produção e do tempo (ou custo) total de *setups* (montagens de máquinas), adotando objetivos do tipo

$$\min d^T y + c^T x$$

com restrições apropriadas, de tal forma que, para cada padrão de corte j, $x_j \in \Re_+$ é a sua produção, $y_j \in \{0,1\}$ é uma variável indicadora de seu uso ($x_j > 0$), ou não ($x_j = 0$), c_j é a velocidade da máquina para o padrão em quantidade/tempo, e d_j é o tempo de *setup* do padrão. Foram feitos experimentos com exemplares parametrizados com $c_j = \gamma$ e $d_j = \alpha + \beta f_j$ onde f_j é o número de larguras do padrão j de tal modo que d_j representa o número de facas para cortar o padrão j. A Tabela 4 apresenta os valores médios obtidos para 9 exemplares gerados e resolvidos. As colunas cpu.cp, cpu.pi, itr apresentam, respectivamente, o tempo

médio de processamento do *Solver* de *constraint programming*, o tempo médio da programação inteira e o número de iterações.

≈	min	t.total	t.setup	t.prod	n+	cpu.cp	cpu.pi	itr
≥	t.setup†	732	32	700	10	0,01	2,33	37556
≥	t.prod	500	-	500	16	0,01	0,09	27
=	t.total	536	36	500	13	0,19	2047,29	2088069
=	t.setup	727	30	697	14	0,21	438,40	556522
=	t.prod	500	-	500	17	0,18	0,18	12

Tabela 4: Resultados médios para tempos de *setup*. † solução heurística

Pode-se notar que o tempo de resolução associado à minimização do tempo de produção é bem menor do que o tempo total ou o tempo de *setup* devido à perda de importância das variáveis inteiras necessárias para modelar o tempo de *setup*, que não está presente na minimização do tempo de producão.

3.3. Número de *setups*

Foi realizado um estudo do menor número de *setups* a serem realizados, adotando, juntamente com restrições apropriadas, a função objetivo

$$\min \sum y_j$$

de tal forma que o padrão j tem um indicador $y_j \in \{0,1\}$ de uso ($y_j = 1$), ou não ($y_j = 0$) e uma quantidade a produzir $x_j \in \Re_+$ ($x_j = 0$ se $y_j = 0$). A Tabela 5 apresenta os valores médios obtidos para 9 exemplares gerados e resolvidos.

≈	n_+	n	prod	cpu.cp	cpu.pi	itr
≥	10	120	609	0,01	0,81	488
=	13	572	503	0,16	4019,65	3257636

Tabela 5: Resultados médios para minimização do número de *setups* n_+

Pode-se observar que o tempo médio de cpu e o número médio de iterações para os exemplares com restrições sem folgas são maiores do que os correspondentes para os exemplares com restrições com folgas. Isto é devido ao maior número médio de padrões de corte gerados para os primeiros.

3.4. Crescimento do problema

Finalmente, foi feito um estudo da dificuldade de resolver problemas de corte em função do número de larguras m. Para $m \in \{10, 11, ..., 18\}$ larguras foram gerados exemplares e anotados os tempos de cpu para gerar os padrões de corte (cpu.cp), para obter a solução exata (cpu.pi) e para obter um limitante através da relaxação linear (cpu.pl). Foi utilizado um par de sementes para gerar as larguras e os pedidos

de cada exemplar, de tal forma que as larguras e os pedidos de um problema menor fossem as larguras e os pedidos iniciais de cada um dos problemas maiores. Foram resolvidos 10 problemas com restrições de pedidos com folgas ($Ax \geq b$) para cada valor de m e com o objetivo de minimizar: a) o tempo total de produção e *setups*, b) o tempo total de *setup*, c) o número de *setups*, d) a produção e e) a perda. Os resultados médios dos tempos de cpu dos exemplares com larguras pares encontram-se na Tabela 6. As Figuras 1 e 2 foram construídas com os valores médios de todas as larguras, pares e ímpares.

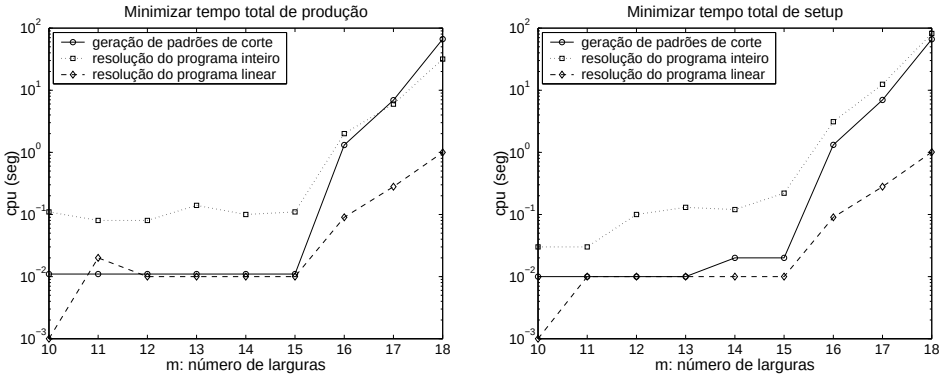

Figura 1: Segundos de cpu para gerar padrões de corte (linha cheia), resolver programa inteiro (linha pontilhada) e resolver programa linear (linha tracejada)

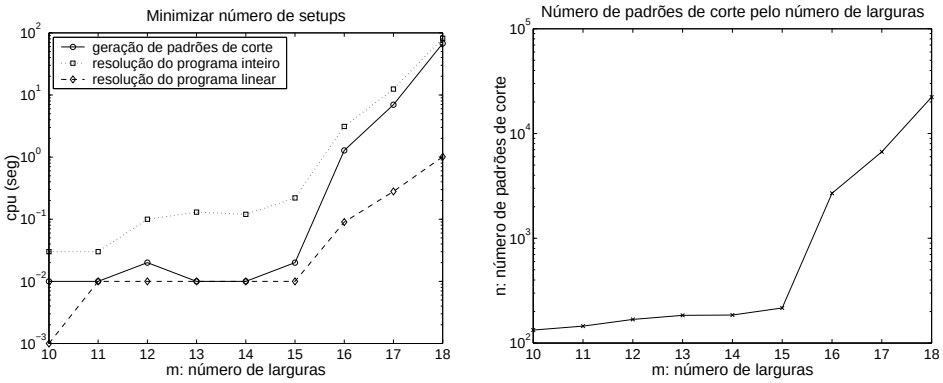

Figura 2: a) Segundos de cpu para gerar padrões de corte (linha cheia), resolver programa inteiro (linha pontilhada) e resolver programa linear (linha tracejada) b) Número de padrões de corte pelo número de larguras

Pode-se observar que o tempo de resolução do programa linear sempre é menor

minimizar	m larguras / n padrões	10 / 133	12 / 168	14 / 185	16 / 2688	18 / 22276
tempo total	cpu.cp	0,01	0,01	0,01	1,31	66,51
	cpu.pi	0,11	0,08	0,10	2,00	31,74
	cpu.pl	0,01	0,01	0,01	0,09	1,00
tempo setup	cpu.cp	0,01	0,01	0,02	1,32	66,22
	cpu.pi	0,03	0,10	0,12	3,11	81,88
	cpu.pl	0,00	0,01	0,01	0,09	1,01
número setups	cpu.cp	0,01	0,02	0,01	1,28	67,54
	cpu.pi	0,07	0,11	0,13	3,91	79,44
	cpu.pl	0,01	0,01	0,01	0,09	1,06
produção	cpu.cp	0,01	0,01	0,02	1,29	66,32
	cpu.pi	0,02	0,03	0,05	0,43	10,88
	cpu.pl	0,01	0,01	0,01	0,08	1,06
perda	cpu.cp	0,02	0,01	0,02	1,33	66,35
	cpu.pi	0,02	0,07	0,02	0,38	3,77
	cpu.pl	0,01	0,01	0,02	0,08	1,06

Tabela 6: Resultados médios para um número crescente de larguras m

do que o tempo de resolução do programa inteiro. Entretanto, o tempo de geração dos cortes, com os quais os programas linear e inteiro ficam especificados, tem uma curva de crescimento muito acentuada. Na realidade, a geração dos padrões de corte passa a consumir um tempo maior do que aquele necessário para resolver qualquer um dos dois problemas. A Figura 2 torna isto mais claro, ao apresentar o número de padrões de corte (número de variáveis para os programas inteiro e linear) que foram gerados para os diferentes números de larguras. A taxa de crescimento crescente salienta-se mesmo com o uso de escala logarítmica no eixo vertical.

4. Considerações finais

Os experimentos indicam que há uma dificuldade maior (tempos de cpu e número de iterações) para resolver problemas de corte unidimensionais sem folgas ($Ax = b$) do que com folgas ($Ax \geq b$). Isto foi observado para todos os tipos de funções objetivo testadas (quantidades produzidas ou perdidas, tempos de *setup* ou produção, número de *setups*). Em parte, isto se deve ao conjunto de soluções factíveis ter menor dimensão. Por outro lado, o conjunto de padrões de corte utilizado nos problemas de corte com folgas é menor do que o dos problemas de corte sem folgas. Os tempos de processamento sugerem que a estratégia de uso de soluções heurísticas deve ser considerada, uma vez que os problemas maiores estão exigindo um tempo de processamento muito grande, não somente na resolução do modelo de programação matemática como na geração dos padrões de corte. Finalmente, os resultados apresentados foram aqueles considerados mais interessantes até o mo-

mento.

Abstract. Cutting stock problems are mathematical models of great importance for production planning in paper, steel, and glass industries. This paper considers a computational study using a computational environment OPL-Studio (Optimization Programming Language) which calls a software for cutting patterns generation (called Solver) and a software for mathematical programs resolution (called CPLEX). We considered fixed charge problems whose objective functions represent the sum of *setup* times with production times. Computational tests were performed in order to compare cpu processing time and solution quality obtained from instances of unidimensional problems randomly generated. We verified that heuristics constitute important tools for pattern generation in large problems.

Referências

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

[2] H. Dyckhoff e U. Finke, "Cutting and Packing in Production and Distribuition", Springer-Verlag Co., Heidelberg, Germany, 1992.

[3] A.A. Farley e K.V. Richardson, Fixed charge problems with identical fixed charges, *European Journal of Operational Research*, **18** (1984), 245-249.

[4] H. Foerster e G. Wäscher, Pattern reduction in one-dimensional cutting stock problems, *International Journal of Production Research*, **38** (2000), 1657-1676.

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

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

[7] R.W. Haessler, Controlling cutting pattern changes in one-dimensional trim problems, *Operations Research*, **23** (1975), 483-493

[8] P.V. Hentenryck, "The OPL: Optimization Programming Language", The MIT Press, Cambridge, Massachusetts (1999).

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

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

[11] C. McDiarmid, Patterns minimisation in cutting stock problems, *Discrete Applied Mathematics*, **98** (1999), 121-130.

[12] H. Stadtler, A one dimensional cutting-stock problem in the aluminium industry and its solution, *European Journal of Operational Research,* **44** (1990), 209-223.