GERAÇÃO DE COLUNAS E GRASP PARA O PROBLEMA DE CORTE DE ESTOQUE BIDIMENSIONAL GUILHOTINADO

André Soares Velasco
Instituto Federal Fluminense - IFF/ Universidade Federal Fluminense - UFF
Av. Souza Mota, 350. Parque Fundão - Campos dos Goytacazes - RJ. CEP: 28060-010
asvelasco@iff.edu.br

Eduardo Uchoa
Dep. de Engenharia de Produção - Universidade Federal Fluminense - UFF
Rua Passo da Pátria 156, Bloco D. São Domingos - Niterói - RJ. CEP: 22210-240
uchoa@producao.uff.br

RESUMO

Este trabalho considera o Problema de Corte de Estoque Bidimensional Guilhotinado com itens retangulares. As versões com e sem rotação de itens são consideradas. A primeira fase do algoritmo proposto é uma geração de colunas em que o subproblema de corte é relaxado de forma a permitir padrões irrestritos, possivelmente tendo mais cópias de certo item do que realmente é demandado. Esses subproblemas são resolvidos por programação dinâmica. As soluções da geração de colunas são arredondadas para baixo, atendendo a uma parte da demanda. Na segunda fase do algoritmo, a demanda residual é atendida pela aplicação sucessiva de uma heurística GRASP para o problema de corte restrito. Testes computacionais com instâncias da literatura obtiveram soluções provadamente ótimas na maior parte dos casos.

PALAVARAS CHAVE. Padrões de Corte Bidimensionais. Geração de Colunas. GRASP.

ABSTRACT

This work addresses the Two-dimensional Rectangular Guillotine Cutting Stock Problem. Version with or without rotation are considered. The first phase of the proposed algorithm is a column generation. The cutting subproblem in the pricing is relaxed in order to allow unrestricted patterns, possibly having more copies of an item than its demand. The subproblem is solved by dynamic programming. The fractional solutions of the column generation are rounded down, meeting part of the demand. In the second phase of the algorithm, the residual demand is fulfilled by the successive application of a GRASP heuristic for the restricted cutting problem. Computational experiments with instances from the literature obtained proved optimal solutions in most cases.

KEYWORDS. Two-dimensional Cutting Patterns. Column Generation. GRASP.

1. Introdução

O Problema de Corte de Estoque Bidimensional Guilhotinado (PCEBG) consiste em determinar como se deve cortar um conjunto de peças retangulares (Objetos), em quantidade suficiente, a partir de cortes ortogonais a um dos lados dessas peças, com o intuito de produzir certa quantidade de peças retangulares menores (Itens), utilizando a menor quantidade possível desses objetos. Com sua grande aplicabilidade em diversos setores produtivos, como por exemplo, metal mecânico, moveleiro, vidraceiro, entre outros, uma solução ótima para este problema apresentaria quais padrões bidimensionais, produzidos a partir de cortes com guilhotina, devem ser repetidos o mínimo de vezes para atender uma demanda.

A maneira de se determinar um padrão de corte com duas dimensões relevantes (comprimento e largura), que minimize a perda de matéria prima gerada (ou maximize o lucro total) com o corte dos itens, dá as características necessárias para identificar as variações do Problema de Corte Bidimensional (PCB), como conhecido na literatura. Aos itens, atribui-se um valor de utilidade que pode ser a medida de sua área (PCB sem valor) ou estar relacionado à sua importância em presença aos outros itens (PCB com valor). Uma rotação de 90º nos itens pode ser aceita (PCB com rotação) ou proibida (PCB sem rotação). Os cortes ortogonais podem ser guilhotinados e gerar dois novos retângulos (PCBG) ou não guilhotinados (PCBNG). A quantidade de cada item produzido a partir de cortes guilhotina pode ser restrita (PCBGR) ou ilimitada (PCBGI). O número de estágios no padrão de corte está relacionado à quantidade de mudanças permitidas na direção dos cortes guilhotina, ou seja, pode-se admitir apenas k rotações de 90º nas direções destes cortes (k-estagiado) ou não restringir esta quantidade (não estagiado). Todas essas variantes do PCB, assim como os Problemas de Corte de Estoque Bidimensional (PCEB) correspondentes, pertencem à classe NP-difícil.

Uma abordagem clássica para o tratamento de PCE em geral, é a Geração de Colunas, introduzida por Gilmore e Gomory (1961, 1963, 1965). A Geração de Colunas exige uma solução dos chamados problemas de corte com valor (subproblema), que buscam por um padrão de corte que maximize o lucro total dos itens produzidos pelo padrão. Na literatura, grandes avanços têm sido alcançados com o PCBGR k-estagiado ou PCBGI k-estagiado sendo tratado como um subproblema do PCEBG k-estagiado. Neste caso, abordagens utilizando algoritmos híbridos e heurísticas constitui-se uma tendência que pode ser conferida em Cintra *et al.* (2008) e Furini *et al.*(2012). Entretanto, sabe-se que o PCBGR não estagiado é mais difícil de resolver de forma exata do que o PCBGR k-estagiado ou o PCBGI não estagiado e, trabalhos que vêm contribuindo para sua resolução, como o algoritmo exato proposto por Christofides e Hadjiconstantinou (1995) e a abordagem heurística proposta por Morabito e Pureza (2010), apresentam tempos proibitivos.

Diante desta dificuldade, este trabalho propõe dois algoritmos híbridos, GCH-2D e GCH-$2D_R$, para o PCEBG sem e com rotação, respectivamente. Eles se baseiam na ideia apresentada por Cintra *et al.* (2008), onde o subproblema da geração de colunas é relaxado para um PCBGI e resolvido por Programação Dinâmica. Caso a solução ótima do PL Mestre não seja inteira, os PCBGR residuais são resolvidos com a utilização de dois algoritmos (GRASP-$2D_A$ e GRASP-$2D_{AR}$), fundamentados na metodologia *Greedy Randomized Adaptive Search Procedure* (FEO; RESENDE, 1989). Para comparar o desempenho dos algoritmos GCH-2D e GCH-$2D_R$ com os algoritmos CG, CG^P, CGR e CGR^P, apresentados em Cintra *et al.* (2008), foram realizados testes em 24 instâncias do PCEBG.

O presente artigo está dividido em sete seções. Uma breve introdução conceitual do método Geração de Colunas será apresentada na Seção 2. A Seção 3 apresentará o método Programação Dinâmica para o PCBGI, incluindo o algoritmo DDP que retorna o conjunto dos pontos de discretização (CINTRA *et al.*, 2008). A Seção 4 apresenta o GRASP-$2D_A$ e o GRASP-$2D_{AR}$, os principais algoritmos originais deste trabalho, usados para resolver os PCEBG residuais da geração de colunas. Na Seção 5, serão descritos os algoritmos completos de Geração de Colunas + GRASP propostos, GCH-2D e GCH-$2D_R$. Na Seção 6, serão mostrados os testes computacionais bem como a análise dos resultados. A Seção 7 será destinada às conclusões.

2. Formulação do PCEBG

Com os padrões de corte bidimensionais guilhotinados definidos e uma quantidade de objetos em estoque suficiente para atender a uma demanda de itens pré-estabelecida, o PCEBG pode ser modelado como um problema de Programação Linear Inteira. A formulação a seguir, consiste em determinar o número de vezes que cada padrão de corte é utilizado, de forma a atender essa demanda, consumindo-se o menor número possível de objetos.

$$\begin{aligned} \text{Min} \quad & \sum_{j=1}^{n} x_j \\ \text{s.a:} \quad & \sum_{j=1}^{n} a_{ij}\, x_j \geq d_i\,, \; i = 1,...,m \\ & x_j \geq 0 \;\text{ e Inteiro}, \; j = 1,...,n \end{aligned} \tag{2.1}$$

Onde:
- x_j é a variável que representa o número de vezes que o padrão de corte j é utilizado;
- a_{ij} é a constante que indica número de itens i gerados no padrão de corte j;
- d_i é a constante que indica a demanda de itens i;
- m é a constante que indica o número de itens distintos;
- n é a constante que indica o número total de padrões de corte.

O valor de *n* é tipicamente muito grande, impedindo que a Relaxação Linear dessa formulação seja resolvida diretamente por um método como o Simplex. Esse impedimento pode ser contornado com o método de geração de colunas (DANTIZG; WOLFE, 1960). A Geração de Colunas pode ser vista como uma extensão do Simplex Revisado, onde o *pricing* das variáveis é realizado através da resolução de um subproblema, e foi primeiramente introduzida por Gilmore e Gomory (1961, 1963) para o PCE Unidimensional. Na Geração de Colunas, o chamado Problema Linear Mestre (PL Mestre) é inicializado com apenas um pequeno subconjunto das variáveis, correspondendo a alguns padrões de corte. A cada iteração, é gerado e inserido um novo padrão, desde que ele possa melhorar o valor da função objetivo, até que se prove que a solução corrente é ótima. Sendo um problema de minimização, deve-se buscar por padrões que levem a variáveis com custo reduzido negativo. Isso resulta no seguinte subproblema:

$$\begin{aligned} \text{Min} \quad & (1 - \sum_{i=1}^{m} \pi_i y_i) \\ \text{s.a:} \quad & [y_1,..., y_m]^t \text{ é um padrão viável} \\ & y_i \geq 0 \;\text{ e Inteiro} \end{aligned} \tag{2.2}$$

Onde:
- y_i é a variável que representa o número de itens i no padrão de corte;
- π_i são as variáveis duais para cada restrição i do PL Mestre.

Neste trabalho, esse subproblema é abordado como o problema da geração de padrões não estagiados para o PCBGI com valor. Essa última particularidade se deve à dificuldade de resolver esse problema restrito de forma exata e, ainda, indica que podem ser reproduzidos padrões que contenham mais cópias do item i do que o estabelecido pela demanda d_i. Nas próximas seções serão apresentados três algoritmos, uma programação dinâmica para o PCBGI supracitado, utilizado na geração de novas colunas para o PL Mestre, e dois algoritmos GRASP designados a resolver o PCBGR não estagiado, com e sem rotação, da instância residual.

3. Programação Dinâmica para o PCBGI

Sabendo que o valor ótimo de um padrão de corte para o PCBGI pode ser igual à soma dos valores de utilidade da sua decomposição em faixas otimizadas, permite que este problema

seja resolvido por meio da técnica de programação dinâmica. Diferentemente da abordagem proposta por Gilmore e Gomory (1965), as fórmulas de recorrência apresentadas por Beasley (1985) não limitam a quantidade de estágios no padrão de corte e, juntamente com os conceitos de pontos de discretização de Herz (1972), são utilizadas por Cintra *et al*. (2008) no desenvolvimento de um algoritmo para geração de padrões do PCBGI com valor como segue.

3.1 Algoritmo DDP

Para determinar os pontos de discretização em um objeto R = (C, L), a partir das dimensões dos itens requisitados, utiliza-se uma técnica de programação dinâmica para o problema da mochila que providencia o preenchimento ótimo de uma mochila de capacidade inteira D, por itens de peso e valor igual a d_i, com i = 1,..., m. Define-se que os pontos de discretização do comprimento C, calculados como uma combinação cônica inteira dos comprimentos dos itens, pertencem ao conjunto P. De forma análoga, tem-se em Q os pontos associados a largura L. O pseudocódigo do algoritmo DDP (*Discretization using Dynamic Programming*), que retorna o conjunto P dos pontos de discretização para uma dimensão D, é apresentado a seguir.

Procedimento DDP (D, d_i)
1. $P = \{0\}$;
2. Para (j = 0,..., D) faça
3. $c_j = 0$;
4. Para (i = 1,..., m) faça
5. Para (j = d_i,..., D) faça
6. Se ($c_j < c_{j - di} + d_i$) faça
7. $c_j = c_{j - di} + d_i$;
8. Para (j = 1,..., D) faça
9. Se (c_j = j) faça
10. $P = P \cup \{j\}$;
11. Retorna (P);

Fim DPP

Sejam c_{min} e l_{min} o menor comprimento e a menor largura entre os itens, nessa ordem. Denomina-se por P_0 o conjunto dos pontos de discretização $p_i \leq C - c_{min}$, associados ao comprimento C do objeto. Analogamente, $q_j \leq L - l_{min}$ são os pontos de discretização de Q_0, associados à largura L. Com as devidas inserções no algoritmo DDP, a substituição dos respectivos conjuntos P e Q pelos conjuntos $P_1 = P_0 \cup \{C\}$ e $Q_1 = Q_0 \cup \{L\}$ vão produzir significativo ganho computacional na execução do próximo algoritmo.

3.2 Algoritmo DP

O algoritmo DP (*Dynamic Programming*) aproveita os pontos de discretização com o intuito de obter exclusivamente padrões canônicos. Isto significa que o padrão de corte ótimo que o DP retorna, apresenta suas faixas guilhotinas geradas a partir de cortes efetivados apenas nos pontos discretizados.

Considerando que $v(c, l) = \max(\{v_i | 1 \leq i \leq m, c_i \leq c, l_i \leq l\} \cup \{0\})$ é o maior valor obtido com a produção de um único item no subretângulo (c, l), ou 0 caso não caiba algum item, e admitindo a possibilidade de um padrão de corte não possuir cortes guilhotina, ou apresentar o primeiro corte guilhotina vertical na posição c', ou mostrar o primeiro corte guilhotina horizontal na posição l`, a seguinte fórmula de recorrência é utilizada construtivamente conforme descrito na sequência.

$$V(c,l) = \max(v(c,l), \{V(c',l) + V(c-c',l) \mid c' \in P_1 \text{ e } c' \leq c/2\}, \{V(c,l') + V(c,l-l') \mid l' \in Q_1 \text{ e } l' \leq l/2\})$$

Inicialmente, para cada subretângulo de dimensões $p_i \in P_1$ e $q_j \in Q_1$, o item de maior valor é designado. Na sequência, ainda para os subretângulos (p_i, q_j), verifica-se para cada possível corte, vertical em p_x ou horizontal em q_y, se promove alteração no valor ótimo de (p_i, q_j), com i = 1,..,r e j = 1,..,s. Sendo assim, na composição da solução ótima, o processo inicia-se com o menor subretângulo (p_i, q_j) e conforme as dimensões desses subretângulos vão aumentando, até

atingir as dimensões de R, as informações sobre as suas respectivas melhores soluções são utilizadas na determinação do padrão ótimo de corte.

Na reconstrução do padrão ótimo, DP recupera em determinada matriz as informações referentes à direção e posição do primeiro corte guilhotina, realizado nos subretângulos determinados a partir dos pontos de discretização. Se não houver ocorrência de corte guilhotina, é indicado qual item deve ser produzido nesse subretângulo. Isto é, os subretângulos de dimensões (p_i, q_j) apresentam valor de utilidade ótimo V(i, j) ≤ V(r, s), guilhotina(i, j) indica a direção horizontal ou vertical do primeiro corte e item(i, j) aponta onde deve ser feito este primeiro corte. Se guilhotina(i, j) = nil, então nenhum corte guilhotina é realizado e item(i, j) determina qual item deve ser gerado em (p_i, q_j).

Uma descrição do algoritmo DP, que retorna o padrão de corte ótimo (PCI_{Otimo}) para o PCBGI com valor, é apresentada a seguir.

Procedimento DP (C, L, c_i, l_i, v_i)
1. Determine $p_i < ... < p_r$, os pontos de discretização do conjunto P_I;
2. Determine $q_i < ... < q_s$, os pontos de discretização do conjunto Q_I;
3. Para (i = 1,..., r) faça
4. Para (j = 1,..., s) faça
5. $V(i, j) = \max(\{v_k | 1 \leq k \leq m, c_k \leq p_i \text{ e } l_k \leq q_j\} \cup \{0\})$;
6. $item(i, j) = \max(\{k | 1 \leq k \leq m, c_k \leq p_i, l_k \leq q_j \text{ e } v_k = V(i, j)\} \cup \{0\})$;
7. guilhotina(i, j) = nil;
8. Para (i = 2,..., r) faça
9. Para (j = 2,..., s) faça
10. $n = \max(k | 1 \leq k \leq i \text{ e } p_k \leq \lfloor p_i/2 \rfloor)$;
11. Para (x = 1,..., n) faça
12. $t = \max(k | 1 \leq k \leq r \text{ e } p_k \leq p_i - p_x)$;
13. Se ($V(i, j) < V(x, j) + V(t, j)$) faça
14. $V(i, j) = V(x, j) + V(t, j)$, $posição(i, j) = p_x$ e guilhotina(i, j) = 'V';
15. $n = \max(k | 1 \leq k \leq j \text{ e } q_k \leq \lfloor q_j/2 \rfloor)$;
16. Para (y = 1,..., n) faça
17. $t = \max(k | 1 \leq k \leq \text{s e } q_k \leq q_j - p_y)$;
18. Se ($V(i, j) < V(i, y) + V(i, t)$) faça
19. $V(i, j) = V(i, y) + V(i, t)$, $posição(i, j) = q_y$ e guilhotina(i, j) = 'H';
20. Retorna (PCI_{Otimo});

Fim DP

É importante destacar que o algoritmo descrito acima também pode gerar padrões ótimos para PCBGI com valor e rotação. Neste caso, para cada item de uma instância, com dimensões $c_i \neq l_i$, $c_i \leq C$, $l_i \leq L$ e valor de utilidade v_i, foi adicionado a essa instância um novo item (l_i, c_i), apresentando o mesmo valor de utilidade v_i, com i = 1,...,m.

4. Algoritmos GRASP-2D$_A$ e GRASP-2D$_{AR}$

O algoritmo GRASP-2D$_A$ foi idealizado, a partir de ampliações no algoritmo GRASP-2D (VELASCO *et al.*, 2008), com o propósito de resolver o PCBGR não estagiado, sem rotação e com o valor dos itens pertinente a sua respectiva medida de área (VELASCO; UCHOA, 2014). O algoritmo GRASP-2D$_{AR}$ é uma extensão do GRASP-2D$_A$, onde a rotação dos itens é admitida no processo de organização do padrão de corte. Assim como na implementação do algoritmo GRASP-2D$_A$, destacam-se alguns fundamentos para o GRASP-2D$_{AR}$ e, ainda que de forma concisa, é de extrema importância a apresentação destes.

Admitindo um estoque de objetos retangulares R = (C, L) de comprimento C e largura L, em quantidade suficiente para atender uma demanda finita de itens, também retangulares, p_i = (c_i, l_i), que possuem dimensões limitadas superiormente pelas de R e demandas d_i previamente estabelecidas, com i = 1,...,m. Em relação a um sistema de coordenadas cartesianas com origem no canto inferior esquerdo de R, se permitida uma rotação em 90 graus, os itens p_i são indicados por r_i = (l_i, c_i). Um item p_i = (c_i, l_i), ou com orientação r_i = (l_i, c_i), inserido no canto inferior esquerdo de R, dá origem a dois tipos de faixas com a execução de cortes do tipo guilhotina sobre

este objeto. Se uma faixa guilhotina horizontal $F_H = (C, l_i)$ é definida a começar de um corte guilhotinado horizontal de largura l_i, então esse corte ainda determina um subretângulo $R_{FH} = (C, L\text{-}l_i)$ que deve ser utilizado, provavelmente, na definição de outras faixas guilhotinas do padrão. Assim como, uma faixa guilhotina vertical $F_V = (c_i, L)$, obtida a partir de um corte guilhotinado vertical, de comprimento c_i em R, dá origem a outro subretângulo com esse corte vertical, $R_{FV} = (C\text{-}c_i, L)$, que também tem o propósito de ser aproveitado na configuração das demais faixas do padrão de corte. De forma análoga, produzir um item de orientação $r_i = (l_i, c_i)$, com um corte horizontal ou vertical, determina as faixas $F_H = (C, c_i)$ ou $F_V = (l_i, L)$ e os subretângulos $R_{FH} = (C, L\text{-}c_i)$ ou $R_{FV} = (C\text{-}l_i, L)$, respectivamente. Ao posicionar os itens que dão origem a essas faixas sempre no canto inferior esquerdo de R ou de R_F, fica estabelecido um padrão de corte na forma normalizada equivalente (WANG, 1983).

Considerando que uma faixa guilhotina pode ser identificada pela forma como um conjunto de itens estão dispostos, é utilizada uma sequência de caracteres alfanuméricos para descrever essas faixas. Os caracteres H e V são utilizados quando se executa um corte que dá origem a faixas guilhotina horizontal ou guilhotina vertical, respectivamente. Os itens com orientação p_i ou r_i, gerados nessas faixas, são diferenciados pelos caracteres P ou R, nessa ordem. Por exemplo, na figura 1, a faixa guilhotina vertical que se iniciou com dois itens r_2 é representada pela expressão V2R2. Enquanto, a faixa guilhotina horizontal iniciada com p_1 é caracterizada pela expressão H1P1. Se a faixa guilhotina apresenta outros itens, além do item relacionado na sua construção, novas expressões inicializadas com as letras M ou A são concatenadas as já iniciadas por H ou V. Esses caracteres M e A, identificam o modo pelo qual um novo item foi selecionado a configurar uma melhoria na faixa guilhotina. A inclusão destas últimas expressões está diretamente relacionada ao melhor aproveitamento das faixas. A figura 1 ilustra um padrão do corte com cinco estágios indicados por *Gk*, com k = 1,...5, e três faixas guilhotinas, incluindo suas respectivas melhorias, onde este pode ser descrito pela *string* de caracteres V2R2M1P5M1P8 H1P1A1R4A1P6 V1P7A4P2 H1R3.

Sendo os padrões de corte diferenciados pela maneira como um conjunto de itens estão agrupados em faixas guilhotinas e o arranjo dessas faixas, o valor das perdas associadas às medidas de área não aproveitadas nessas configurações certificam se há qualidade em uma solução para o PCBGR. Isto é, o somatório das medidas de superfície improdutivas no objeto R define a perda no padrão de corte e são categorizadas como perda interna e perda externa. Se determinada faixa guilhotina apresenta áreas improdutivas, estas são apontadas como perda interna. Quando um corte realizado em R não proporciona a opção de obter novas faixas guilhotinas, a medida de área desse subretângulo gerado é denominada perda externa. Para exemplificar, ainda na figura 1, destacam-se os dois tipos de perda encontrados nos padrões de corte.

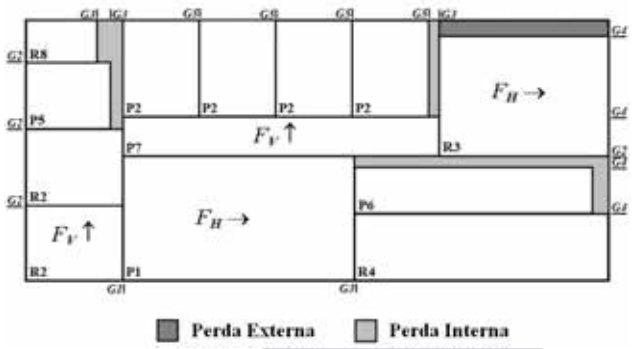

FIGURA 1 – Padrão de corte bidimensional guilhotinado.

Na sequência, são apresentados os procedimentos das fases de construção e melhoria da GRASP quando aplicados a geração de padrões de corte para o PCBGR com as características evidenciadas nesta seção.

4.1 Fase de Construção dos Algoritmos GRASP-2D$_{A/AR}$

Nesta fase, a construção de faixas guilhotinas viáveis é iniciada de forma iterativa, utilizando um item p_i de cada vez, retirada aleatoriamente da LRC. Sendo a LRC formada pelos itens com demanda ainda não atendida, onde são escolhidos em um processo com três elementos fundamentais: uma lista C' constituída dos itens aspirantes a originar uma faixa, uma função gulosa v: P→ℝ, que associa a cada item $p_i \in P$ um único valor igual à medida de sua área e um critério guloso $\beta = \max \{v(p_i); p_i \in C'\}$ para determinar o item pertencente a C' de maior área. Sabendo que $LRC = \{p_i \in C'/\alpha.\beta \leq v(p_i) \leq \beta\}$ apresenta os itens mais interessantes de C', segundo valor de utilidade, e sua cardinalidade é diretamente influenciada pelo parâmetro adotado $\alpha \in [0,1]$, uma seleção de forma aleatória determina o item responsável pela definição das dimensões das novas faixas guilhotinas.

O item p_k que vir a ser escolhido aleatoriamente na LRC dá início a construção de duas faixas guilhotinas viáveis, através de um corte guilhotina horizontal e outro vertical. Para estas faixas $F_H = (C_H, L_H)$ e $F_V = (C_V, L_V)$, não só são calculados os referentes η_k números de itens p_k inseridos e o valor de perda interna associada a cada uma destas, como também são determinados subretângulos $RF_H = (C_H - \eta_k.c_k, l_k)$ e $RF_V = (c_k, L_V - \eta_k.l_k)$, relativos às respectivas perdas internas. Caso o item sorteado não produza uma faixa guilhotina viável no padrão de corte, este item é imediatamente retirado de C' e as informações relacionadas ao problema são atualizadas para a próxima escolha na LRC. A possibilidade de p_k ser inserido com uma rotação de 90º, transformando-se em $r_k = (l_k, c_k)$ distingue a fase de construção dos algoritmos GRASP-2D$_A$ e GRASP-2D$_{AR}$. Os subretângulos gerados com a utilização de um item com orientação r_k, $RF_H = (C_H - \eta_k.l_k, c_k)$ e $RF_V = (l_k, L_V - \eta_k.c_k)$, evidenciam uma mudança de direção na busca por melhores soluções. Se o item k selecionado na LRC puder ser incluído tanto com a orientação p_k quanto com a rotação r_k, uma escolha aleatória define a orientação desse item na faixa.

Se as faixas F_H e F_V, construídas de forma sequencial após extração de um item p_k na LCR, não apresentam perda interna, estas podem ser repetidas no padrão de corte, em quantidades que não extrapolem as dimensões da placa, nem violem a demanda d_k, de acordo com ajuste feito pelo parâmetro φ. Este parâmetro determina a frequência de iterações que admitem a repetição destas faixas no padrão de corte presente.

4.2 Fase de Melhoria do Algoritmos GRASP-2D$_{A/AR}$

No procedimento de melhoria, faixas guilhotinas inicialmente construídas, são tratadas com o objetivo de maximizar o valor de utilidade total em ambas, obtendo um melhor aproveitamento dos respectivos subretângulos $RF_H = (C_H, L_H)$ e $RF_V = (C_V, L_V)$. Com a finalidade de evitar movimentos dispensáveis que resultariam em faixas guilhotinas impraticáveis nas dimensões do objeto ou provocariam proibições para execução do corte guilhotinado, o processo de melhoria das faixas é intensificado com a criação dos conjuntos B_H e B_V. Os itens pertencentes à B_H, assim como os incluídos em B_V, não apresentam dimensões que ultrapassem as faixas F_H e F_V, nessa ordem, mantendo assim a viabilidade das faixas quando selecionados para executar melhorias. No algoritmo GRASP-2D$_{AR}$ é permitido à inserção de itens rotacionados nos conjuntos em destaque.

Com o propósito de acelerar as ações na fase de melhoria, o conjunto B_H é iniciado com os itens de C' que podem ser incluídos no presente subretângulo RF_H. Considerando o movimento de melhoria simbolizado com o caractere M, o item considerado mais interessante é aquele que apresenta maior largura entre os itens pertencentes a B_H. Se outros itens pertencentes a este conjunto possuem medidas de largura iguais, uma escolha aleatória define o novo item a compor a faixa guilhotina. Para o movimento de melhoria indicado pelo caractere A, é escolhido o item que possui maior comprimento em B_H e constatando a ocorrência de mais itens com a mesma dimensão, a seleção também deve ser aleatória.

Em cada uma das estratégias, utilizadas separadamente na melhoria da faixa F_H, calcula-se o número η_j de itens p_j a serem incluídos nesta faixa. Decerto que este número de itens está diretamente relacionado às dimensões da perda interna e à sua demanda d_j, cada inclusão desta maior quantidade possível implica na atualização do conjunto B_H e do subretângulo RF_H. Sendo

$RF_H = (C_H - \eta_j.c_j, L_H)$ o subretângulo após utilização de p_j na melhoria representada por M, ou $RF_H = (C_H, L_H - \eta_j.l_j)$ depois da melhoria indicada por A. Se o item tem orientação r_j, $RF_H = (C_H - \eta_j.l_j, L_H)$ e $RF_H = (C_H, L_H - \eta_j.c_j)$ são os subretângulos das faixas que providenciam as melhorias indicadas por M ou A, nessa ordem. Com o intuito de gerar um melhor aproveitamento da faixa F_H, este processo é repetido até não existir mais itens em B_H.

Assim como B_H, o conjunto B_V é constituído pelos itens de C' que, ao serem incluídos no atual subretângulo RF_V, não impossibilitariam a efetivação da faixa F_V. As melhorias associadas ao uso dos caracteres M ou A, na expressão utilizada para a faixa F_V, aplicam-se quando o item escolhido em B_V apresentar o maior comprimento ou a maior largura, nessa ordem. O número η_m de itens p_m, a serem inseridos na faixa F_V, é determinado de acordo com as dimensões da perda interna e a sua demanda d_m. As melhorias indicadas por M ou A produzem os subretângulos $RF_V = (C_V, L_V - \eta_m.l_m)$ ou $RF_V = (C_V - \eta_m.c_m, L_V)$, respectivamente. De forma análoga, se o melhor item de B_V possui orientação r_j, $RF_V = (C_V, L_V - \eta_m.c_m)$ ou $RF_V = (C_V - \eta_m.l_m, L_V)$ são os subretângulos das faixas a partir das melhorias assinaladas por M ou A, nessa ordem. O conjunto B_V, assim como o B_H, também é atualizado, até não existir mais item que possa melhorar a faixa.

Ao final desta fase, espera-se ter produzido quatro configurações distintas de faixas guilhotinas e, então, aproveitar a faixa que apresenta o menor valor de perda interna na composição do padrão de corte corrente. Considerando que possa acontecer algum empate nos valores de perda interna entre as duas faixas F_H e as duas faixas F_V, a escolha da faixa a configurar no padrão de corte é aleatória.

4.3 Pseudocódigo dos Algoritmos GRASP-2D$_{A/AR}$

Inicialmente, os dados de entrada, de saída e as variáveis ainda não declarados são apresentados, para o melhor entendimento dos algoritmos GRASP-2D$_{A/AR}$, responsáveis por resolverem os PCBGR não estagiados neste trabalho.

- *maxiter* : número máximo de iterações;
- *pitens* : valor percentual do número de itens utilizados na geração dos padrões;
- *padcorteS** : melhor padrão de corte;
- *valorS** : valor de utilidade total do melhor padrão de corte;
- *padcorteS* : padrão de corte corrente;
- *valorS* : valor de utilidade total do padrão de corte corrente;
- $perdaintF_{H/V}$: valor da perda interna em uma faixa guilhotina horizontal ou vertical;
- F_{melhor} : faixa guilhotina incluída em um padrão de corte;
- $perdaintF_{melhor}$: valor da perda interna em uma faixa guilhotina.

Procedimento GRASP-2D$_{A/AR}$ (p_i, d_i, $v(p_i)$, R, α, φ , *maxiter*)

1. $padcorteS^* = \varnothing$ e $valorS^* = 0$;
2. Para (Iter = 1,..., *maxiter*) faça
3. $padcorteS = \varnothing$, $valorS = 0$ e defina $C = (p_1,..., p_m)$;
4. Enquanto ($|C| \leq (pitens.n)$) faça
5. $padcorte = \varnothing$, $R_F = R$ e crie $C' = (p_1,..., p_t)$, $t \leq m$;
6. Enquanto ($|C'| \neq 0$) faça
7. $\beta = max\ \{v(p_i); p_i \in C'\}$;
8. $LRC = \{p_i \in C' / \alpha.\beta \leq v(p_i) \leq \beta\}$;
9. Escolha, aleatoriamente, um item de *LRC* que mantenha a viabilidade;
10. Construa F_H e F_V;
11. Determine RF_H, $perdaintF_H$, RF_V e $perdaintF_V$;
12. Se (MOD(*maxiter*, φ) = 0) faça
13. Se ($perdaintF_{H/V} = 0$) faça
14. Repita $F_{H/V}$;
15. Crie B_H e B_V;
16. Enquanto ($|B_H| \neq 0$ e $|B_V| \neq 0$) faça
17. Selecione itens de B_H e itens de B_V, para usar *"M"* e *"A"*;
18. Melhore F_H e F_V;

19. Atualize B_H, RF_H, $perdaintF_H$, B_V, RF_V e $perdaintF_V$;
20. Determine $perdaintF_{melhor} = min\ \{perdaintF_H, perdaintF_V\}$;
21. Inclua F_{melhor}, associada à $perdaintF_{melhor}$,
22. $padcorte = padcorte \cup \{F_{melhor}\}$;
23. Atualize C' e $R_F = R_F - F_{melhor}$;
24. Calcular $valorS = \Sigma\ v_i.x_i$;
25. Se $valorS > valorS^*$ faça
26. Atualize $padcorteS^* = padcorteS$ e $valorS^* = valorS$;
27. Atualize C;
28. Retorna ($padcorteS^*$);

Fim GRASP-2D$_{A/AR}$

5. Algoritmos GCH-2D e GCH-2D$_R$

Os algoritmos GCH-2D (Geração de Colunas Híbrido 2D) e GCH-2D$_R$ (Geração de Colunas Híbrido 2D com rotação dos itens) foram desenvolvidos para o PCEBG não estagiado, baseado na estratégia apresentada por Gilmore e Gomory (1961), que além apresentar uma técnica de geração de colunas, propuseram uma abordagem residual que determina soluções inteiras aproximadas. Em geral, na Geração de Colunas, a solução ótima para o PL Mestre (2.1) relaxado não é inteira. Admitindo que nesse vetor solução exista pelo menos um padrão de corte com produção fracionada, consegue-se uma solução inteira aproximada arredondando para baixo o número de repetições desse padrão. Restando assim, uma instância residual com menor demanda, denominada problema residual do PCEBG. Em destaque neste trabalho, é proposto a utilização dos algoritmos GRASP-2D$_A$ e GRASP-2D$_{AR}$ na resolução desse problema residual, sem e com rotação dos itens, nessa ordem.

Outra questão importante a ser considerada é que, diante da dificuldade de se resolver de forma exata o subproblema (2.2) com padrões de corte para o PCEBG não estagiado, utiliza-se uma ideia apresentada por Cintra *et al.* (2008), que consiste em resolver este subproblema, gerando padrões irrestritos e não estagiados, até para instâncias que admitem rotação dos itens, por meio do algoritmo DP.

Basicamente, o algoritmos CGH-2D ou CGH-2D$_R$ iniciam com o PL Mestre contendo um conjunto de m+1 colunas, constituído de m padrões homogêneos, com os itens copiados em sua quantidade máxima em cada objeto, e adicionado de um padrão gerado, com o valor de utilidade dos itens igual as respectivas medidas de área, pelo algoritmo DP. A cada iteração seguinte, se for vantajoso, atribui-se aos itens os valores das suas respectivas variáveis duais, gera e insere uma nova coluna ao PL Mestre. Caso a solução ótima desse PL Mestre relaxado não seja inteira, a instância residual é extraída por arredondamento para o maior inteiro menor ou igual à solução relaxada corrente e, os algoritmos GRASP-2D$_A$ ou GRASP-2D$_{AR}$, solucionam o problema residual com padrões de corte restritos, compondo assim a solução inteira do PCEBG original.

Na seguinte descrição do pseudocódigo geral para os dois algoritmos enfatizados nessa seção, primeiramente, considera-se a definição das variáveis ainda não declaradas para os respectivos processos.

- S^*_{PCBGR} : melhor solução de corte;
- $valorS^*_{PCBGR}$: valor total do número de objetos na melhor solução;
- S_{Apr} : solução aproximada de corte corrente;
- $valorS_{Apr}$: valor total do número de objetos na solução inteira aproximada;
- $itens_{PR}$: número total de itens distintos da instância residual;

Procedimento CGH-2D$_{/R}$ (p_i, d_i, $v(p_i)$, R, α, φ , $maxiter$)

1. $S^*_{PCBGR} = \varnothing$ e $valorS^*_{PCBGR} = 0$;
2. Determine o conjunto $[y_1,..., y_m]^t$ de padrões homogêneos para o PL Mestre;
3. Gere nova coluna para o PL Mestre executando DP, com $v(p_i) = c_i.l_i$;
4. Resolva o PL Mestre;
5. Calcule as variáveis duais π_i;
6. Gere uma nova coluna para o PL Mestre executando DP, com $v(p_i) = \pi_i$;

7. Se $(1 - \sum_{i=1}^{m} \pi_i y_i < 0)$ faça
8. Adicione a nova coluna o PL Mestre e volte ao passo 4;
9. Senão $(1 - \sum_{i=1}^{m} \pi_i y_i = 0)$ faça
10. Determine uma solução inteira aproximada, S_{Apr} e $valorS_{Apr}$;
11. Determine os dados da instância residual, $p_i`$ e $d_i`$ e $itens_{PR}$;
12. Atualize $S^*_{PCBGR} = S_{Apr}$ e $valorS^*_{PCBGR} = valorS_{Apr}$;
13. Enquanto $(itens_{PR} > 0)$ faça
14. Resolva o problema residual executando GRASP-$2D_{A/AR}$, com $v(p_i`) = c_i`.l_i`$;
15. Atualize $S^*_{PCBGR} = S^*_{PCBGR} \cup padcorteS^*$ e $valorS^*_{PCBGR} = valorS^*_{PCBGR} + 1$;
16. Se $(d_i` = 0)$ faça
17. Atualize os dados da instância residual, $p_i`$ e $d_i`$ e $itens_{PR}$;
18. Retorna $(S^*_{PCBGR}, valorS^*_{PCBGR})$;

Fim CGH-2D

É importante ressaltar que Cintra *et al.* (2008) também propuseram quatro algoritmos baseado na Geração de Colunas para o PCEBG, sem restrições de estágios de corte, chamados de CG, CG^P, CGR e CGR^P, distinguindo os dois últimos pela necessidade de rotação ortogonal dos itens. Basicamente, o processo do algoritmo CG se dá iterativamente, com as colunas do PL Mestre relaxado sendo fornecidas pelo algoritmo DP, a cada atualização da instância residual, até que a solução de sua relaxação linear, arredondada para baixo, seja igual a zero. Neste último caso, os padrões do problema residual são obtidos pelo algoritmo *Modified Hybrid First Fit* (M-HFF), que consiste numa ampliação do HFF para o caso restrito. O algoritmo CG^P foi estruturado a partir de uma modificação feita no CG, que consiste em utilizar, nas mesmas condições, o M-HFF para obter um padrão com menor perda de área, atualiza-se as demandas e, se ainda houver demanda não atendida, reinicia-se o processo inserindo esta nova coluna ao PL Mestre relaxado. Os algoritmos CGR e CGR^P são versões adaptadas dos algoritmos CG e CGR, devido à rotação dos itens, onde o algoritmo *First Fit Decreasing Height using Rotations* (FFDHR) substitui o algoritmo M-HFF.

Na próxima seção, as diferentes estratégias apresentadas por Cintra *et al.* (2008), quando comparadas com resultados computacionais obtidos com os algoritmos GCH-2D e GCH-$2D_R$, demonstram que a composição com algoritmos DP e GRASP-2D pode ser aplicada com sucesso ao PCEBG não estagiado.

6. Resultados Computacionais

Os algoritmos propostos GCH-2D e GCH-$2D_R$ para o PCEBG não estagiado, sem e com rotação, respectivamente, foram implementados em linguagem C/C++, as relaxações do PL Mestre foram resolvidas pelo CPLEX *Optimization Studio* 12.5.1 e os testes foram realizados em um computador com processador Intel(R) Core(TM)i5-3320M 2.60 GHz, com 4GB de memória RAM e sistema operacional *Windows* 7 de 64 bits.

Os testes computacionais, para comparar os resultados obtidos com os algoritmos supracitados e os algoritmos CG, CG^P, CGR e CGR^P, foram divididos em dois grupos de 12 instâncias, contendo as mesmas informações, que se diferenciam por permitir ou não a rotação ortogonal dos itens. Originalmente, essas instâncias foram propostas por Beasley (1985), com o total de itens distintos iguais a 10, 20, 30 e 50, a serem cortados em objetos de dimensões (250, 250), (500, 500) e (1000, 1000), são referenciadas em diversos trabalhos que abordam a geração de padrões para o PCBGI, e estão disponibilizadas em OR-Library[1]. Como estas instâncias eram para o caso irrestrito, Cintra *et al.* (2008) providenciou demandas geradas aleatoriamente[2] e as renomeou de gcut1d,...,gcut12d e gcut1dr,...,gcut12dr, respectivamente. Estes algoritmos foram implementados em linguagem C e executados em um computador com processador Intel Pentium

[1] http://people.brunel.ac.uk/~mastjjb/jeb/info.html
[2] http://www.ime.usp.br/~glauber/gcut/instances/

IV 1.8 GHz, com 512 Mb de memória RAM, sistema operacional Linux e resolvedor CLP (COIN-OR LP Solver).

TABELA 1 – Resultados do GCH-2D com padrões de corte residuais GRASP-2D$_A$.

Instância	m	Colunas Geradas	Limite Inferior	Solução Aprox.	Solução Grasp-2D	$T_{g(s)}$	Solução GCH-2D	T(s)	Solução GCH-2D$_m$	$T_{m(s)}$	Solução CG/CG$_p$	$T_{c(s)}$
gcut1d	10	21	293.25	292	2	<0.01	294	<0.01	294.0	0.01	294	0.03
gcut2d	20	35	344.25	343	2	<0.01	345	<0.01	345.0	0.01	345	0.28
gcut3d	30	75	331.50	320	12	0.01	*332	0.03	332.8	0.04	333	0.77
gcut4d	50	93	835.83	823	13	0.02	*836	0.08	836.9	0.07	837	5.09
gcut5d	10	22	196.83	194	3	<0.01	*197	<0.01	197.6	0.01	198	0.03
gcut6d	20	43	342.67	338	5	<0.01	*343	0.01	343.5	0.01	344	0.21
gcut7d	30	53	591.00	585	6	<0.01	*591	0.01	591.9	0.01	592	0.33
gcut8d	50	133	690.00	675	16	0.02	*691	0.08	691.5	0.10	692	3.60
gcut9d	10	24	130.67	126	5	<0.01	*131	<0.01	131.2	0.01	132	0.06
gcut10d	20	33	293.00	290	3	<0.01	293	0.01	293.0	0.01	293	0.09
gcut11d	30	74	329.38	318	12	0.01	*330	0.04	330.7	0.04	331	1.07
gcut12d	50	114	671.50	662	10	0.01	672	0.06	672.2	0.07	672	3.35

TABELA 2 – Resultados do GCH-2D$_R$ com padrões de corte residuais GRASP-2D$_{AR}$.

Instância	m	Colunas Geradas	Limite Inferior	Solução Aprox.	Solução Grasp-2D	$T_{g(s)}$	Solução GCH-2D$_R$	T(s)	Solução GCH-2D$_{Rm}$	$T_{m(s)}$	Solução CGR	$T_{c(s)}$	Solução CGR$_p$	$T_{p(s)}$
gcut1dr	10	14	290.25	286	5	<0.01	291	0.01	291.0	0.01	291	0.04	291	0.06
gcut2dr	20	42	281.88	276	6	<0.01	*282	0.03	282.4	0.04	283	1.41	283	1.57
gcut3dr	30	125	312.57	300	13	0.01	*313	0.09	313.9	0.09	315	1.89	314	3.38
gcut4dr	50	91	835.50	823	13	0.01	836	0.05	836.1	0.07	836	4.40	836	6.82
gcut5dr	10	25	173.96	169	5	<0.01	*174	0.01	174.9	0.01	175	0.06	175	0.10
gcut6dr	20	51	300.50	294	7	0.01	*301	0.02	301.2	0.02	302	0.25	302	0.28
gcut7dr	30	61	542.00	539	3	<0.01	*542	0.02	542.8	0.02	544	0.99	543	1.18
gcut8dr	50	152	649.23	636	14	0.01	*650	0.23	650.8	0.25	651	6.47	651	7.10
gcut9dr	10	26	121.92	119	3	<0.01	*122	0.02	122.9	0.01	123	0.07	123	0.10
gcut10dr	20	44	269.50	268	2	<0.01	270	0.01	270.0	0.02	270	0.21	270	0.26
gcut11dr	30	97	297.39	287	11	0.01	*298	0.30	298.7	0.29	299	6.00	299	6.47
gcut12dr	50	143	601.00	591	10	0.01	601	0.28	602.1	0.34	602	19.24	601	19.57

As tabelas 1 e 2 apresentam os resultados obtidos em 10 execuções com os algoritmos GCH-2D e GCH-2D$_R$, assim como as soluções médias GCH-2D$_m$ e GCH-2D$_{Rm}$, para cada uma das instâncias exploradas, nessa ordem. Nos algoritmos GRASP-2D$_A$ e GRASP-2D$_{AR}$, o critério de parada ficou estabelecido em atender a demanda do problema residual. A variação do parâmetro α iniciou com 0.1 e foi incrementada de 0.1 até o valor 1. Dessa maneira, em 10 iterações, sendo 1 iteração para cada valor de α, utilizando até 10% dos itens, o melhor padrão gerado é escolhido e a demanda atualizada. O parâmetro $\varphi = 1$ ficou fixado em todas as rodadas. Os tempos computacionais para a melhor solução do PCE em ênfase (T), nos respectivos problemas residuais (T_g), com tempo médio de execução (T_m) e na execução dos algoritmos confrontados (T_c e T_p) são apresentados em segundos. É importante observar nessas tabelas os valores de soluções com *, pois indicam os melhores resultados conferidos nesta comparação.

Das instâncias utilizadas no PCEBG não estagiado sem rotação, verifica-se na tabela 1 que o algoritmo GCH-2D encontrou 11 soluções ótimas, ou seja, com o número de padrões de corte produzidos igual ao limite inferior arredondado para cima. Estes resultados quando comparados com o desempenho dos algoritmos CG e CG^P, que obtiveram a mesma solução nessas instâncias, evidenciam a eficiência do CGH-2D com 8 soluções melhores. Já na tabela 2, os resultados para o grupo de instâncias cabíveis de rotação, demonstram que o algoritmo CGH-2R também foi eficiente, superando em 9 e 8 instâncias os algoritmos CGR e CGR^P, respectivamente. Ainda nessa tabela 2, é possível conferir que o algoritmo proposto obteve 12 soluções ótimas com a configuração adotada. Deve-se observar que em nenhuma instância dos dois grupos, os algoritmos propostos apresentaram resultados inferiores CG, CG^P, CGR e CGR^P e que os tempos computacionais dos algoritmos são confrontáveis, comprovando não só a eficácia,

mas também a competitividade, destes algoritmos. Vale ressaltar, também, que depois de exaustivos testes, aumentando consideravelmente o número de iterações no algoritmo GRASP-$2D_A$, acredita-se que a instância gcut8d possa não apresentar solução inteira ótima igual a 690.

7. Conclusões

Neste trabalho foram apresentados os algoritmos GCH-2D e GCH-$2D_R$, que combinam a técnica de Geração de Colunas com o algoritmo DP e a metaheurística GRASP, utilizando os algoritmos GRASP-$2D_A$ e GRASP-$2D_{AR}$ para resolver os PCEBG residuais nos casos sem e com rotação. Os resultados foram satisfatórios e obtiveram soluções melhores do que as obtidas por Cintra *et al.* (2008) em 16 das 24 instâncias de teste. Além disso, os tempos computacionais são bastante razoáveis e apresentam menos de 0,4 segundos em todas as instâncias.

É interessante notar que, em 23 das instâncias de teste, foi possível obter uma solução de valor igual ao limite inferior obtido na fase de geração de colunas, provando sua otimalidade. Isso indica que mesmo com a relaxação do subproblema, permitindo padrões irrestritos, esse limite continua sendo bastante forte. Por outro lado, existe uma forte tendência de que a solução ótima fracionária da geração de colunas use padrões em que alguns itens ultrapassem sua demanda. Dessa forma, torna-se difícil usar essa solução para obter diretamente uma boa solução inteira para o PCEBG. Nesse contexto, o uso de heurísticas como o GRASP-$2D_A$ e GRASP-$2D_{AR}$, que tratam bem a variante restrita do problema de corte sem a limitação no número de estágios, para resolver o PCEBG residual, mostrou-se bastante efetivo.

Como perspectivas futuras, a combinação de outras técnicas exatas e outras metodologias heurísticas devem ser consideradas nas novas ampliações do GRASP-2D para resolução das variantes do PCEB.

Referências Bibliográficas

Beasley, J. E. (1985). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, p. 297-306.

Christofides, N. & Hadjiconstantinou, E. (1995). An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts. *European Journal of Operational Research*, v. 83, n. 1, p. 21-38.

Cintra, G. F., Miyazawa, F. K., Wakabayashi, Y., & Xavier, E. C. (2008). Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, 191(1), 61-85.

Dantzig, G.B. & Wolfe, P. (1960). Decomposition principle for linear programs. Operations Research, v.8, p.101-111.

Feo, T. A. & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally dificult set covering problem. *Operation Research Letters*, n.8, p.67-71.

Furini, F., Malaguti, E., Durán, R. M., Persiani, A., & Toth, P. (2012). A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, *218*(1), 251-260.

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

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

Gilmore, P.C. & Gomory, R.E. (1965). Multistage cutting problems of two and more dimensions, *Operations Research*, v.13, p. 94-119.

Herz, J.C. (1972). A recursive computational procedure for two-dimensional stock-cutting. *IBM Journal of Research and Development*, pp. 462–469.

Morabito, R. and Pureza, V. (2010) A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem. *Annals of Operation Research*, v.179, p. 297-315.

Velasco, A.S., Paula Junior, G.G. e Vieira Neto, E. (2008), Um Algoritmo Heurístico Baseado na GRASP para o Problema de Corte Bidimensional Guilhotinado e Restrito. *Revista Gepros - Gestão da Produção, Operações e Sistemas*, ed.1, p.129-141.

Velasco, A. S. e Uchoa, E. (2014). *Geração de Padrões de Corte Bidimensionais Guilhotinados via Grasp*. In: XLVI SBPO - Simpósio Brasileiro de Pesquisa Operacional, Salvador.

Wang, P.Y. (1983), Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, v.31, p.573-586.