Trends in Computational and Applied Mathematics, **23**, N. 4 (2022), 683-703
Sociedade Brasileira de Matemática Aplicada e Computacional
Online version ISSN 2676-0029
www.scielo.br/tcam
doi: 10.5540/tcam.2022.023.04.00683

Uma Heurística Baseada em Programação Dinâmica para o Problema de Corte Bidimensional Guilhotinado 2-Estágios

N. S. ASSIS[1*] e S. RANGEL[2]

Recebido em 4 de agosto de 2021 / Aceito em 27 de maio de 2022

RESUMO. Problemas de corte e empacotamento fazem parte do processo de planejamento da produção em muitas indústrias (*e.g.* papel, vidro, móveis). Em algumas dessas indústrias, um objeto retangular grande deve ser cortado em itens retangulares menores e existe uma capacidade limitada para o estoque dos itens. Nesse contexto, surge o problema de corte bidimensional guilhotinado 2-estágios restrito (PCBG-2est). Alguns autores têm proposto algoritmos de programação dinâmica para resolver o problema no caso irrestrito. Para o caso restrito essa técnica ainda apresenta alguns desafios devido à dimensão do espaço de estados. Nesse artigo propõem-se duas heurísticas baseadas em programação dinâmica e no método de Gilmore e Gomory para resolver o PCBG-2est restrito. São apresentados resultados do estudo computacional realizado com três conjuntos de instâncias que mostram a eficiência da proposta. Em particular, para instâncias similares às encontradas na indústria moveleira foram obtidas soluções com gap máximo médio de 4.4%.

Palavras-chave: Problema de corte bidimensional guilhotinado 2-estágios, problema restrito, programação dinâmica, heurística.

1 INTRODUÇÃO

O Problema de Corte consiste em cortar um objeto grande em itens menores de forma a aproveitar ao máximo da área do objeto e faz parte do processo de planejamento da produção de muitas indústrias, por exemplo: papel, vidro e móveis [7]. Nesse artigo abordamos o problema no contexto de uma indústria moveleira em que o objeto e os itens são retangulares e existe uma capacidade limitada para o estoque dos itens (caso restrito). O equipamento de corte na indústria considerada permite apenas cortes do tipo guilhotina ortogonal, isto é, cortes realizados de um lado ao outro do objeto (guilhotinado) e paralelos ao outro lado do objeto (ortogonal). Considera-se também que o objeto é rotacionado apenas uma vez para obter os itens (corte em 2-estágios).

*Autor correspondente: Nícolas Samuel Assis – E-mail: nicolas.s.assis@unesp.br
[1]Universidade Estadual Paulista (Unesp), IBILCE, Departamento de Matemática, Câmpus de São José do Rio Preto, R. Cristóvão Colombo, 2265, 15054-000, SP, Brasil – E-mail: nicolas.s.assis@unesp.br https://orcid.org/0000-0002-4937-2716
[2]Universidade Estadual Paulista (Unesp), IBILCE, Departamento de Matemática, Câmpus de São José do Rio Preto, R. Cristóvão Colombo, 2265, 15054-000, SP, Brasil – E-mail: socorro.rangel@unesp.br https://orcid.org/0000-0003-3910-8804

Apesar do objeto a ser cortado ser tridimensional, somente o comprimento e largura são relevantes para o corte já que a espessura do objeto e dos itens é a mesma. Uma solução factível para o problema é chamada de padrão de corte e representa a forma que os itens estão posicionados (alocados) no objeto antes do corte. Assim como em outros Problemas de Corte e Empacotamento (PCE), duas condições básicas devem ser satisfeitas para obter uma solução factível: os itens incluídos não podem ultrapassar as dimensões do objeto (restrição de contenção) e não podem estar sobrepostos (restrição de não sobreposição). Um modelo geral para o PCE considerando que, a cada item é atribuído um valor e o critério de otimização é maximizar o valor total dos itens incluídos no objeto, é proposto em [30]. De acordo com a tipologia proposta em [37] o problema tratado é uma variação do problema de alocação de objeto único bidimensional e é chamado nesse texto de Problema de Corte Bidimensional Guilhotinado 2-estágios restrito (PCBG-2est).

O problema de corte bidimensional guilhotinado é amplamente estudado por diversos autores. Em [9] é proposto um algoritmo de busca em árvore para resolver o problema. Como forma de obter limitantes superiores e inferiores para a solução ótima, o caso irrestrito do problema é resolvido por um algoritmo de programação dinâmica [13,14,15]. Outros autores propõem heurísticas de combinação para encontrar soluções factíveis [23, 34, 36]. O algoritmo proposto em [36] constrói interativamente diferentes sub-retângulos que são obtidos a partir da combinação vertical e horizontal dos itens. Quando não há mais possibilidade de combinar os itens em sub-retângulos, o padrão de corte com menor sobra é escolhido como solução. Etapas de melhoria para esse algoritmo são propostas em [34] e [23]. As duas propostas incluem estratégias para diminuir o número de nós da árvore de enumeração implícita formados pelas combinações dos itens em sub-retângulos. Na proposta apresentada em [34] o nó é fechado verticalmente (horizontalmente) caso não haja como combinar mais itens verticalmente (horizontalmente). Em [23] é usada a ideia de perda máxima, se um sub-retângulo tiver uma perda maior do que a máxima, o nó associado é podado. A perda máxima é determinada pela solução do problema irrestrito via programação dinâmica.

A programação dinâmica (PD) é uma ferramenta muito útil para resolver o problema no caso irrestrito, no entanto, para o caso restrito sua utilização direta apresenta algumas dificuldades devido ao aumento da dimensão do espaço de estado (*e.g.* [29, 33, 35]). Esse aumento da dimensão se dá porque além de computar o melhor valor referente à variação de comprimento e largura do objeto, também é necessário explicitar a frequência de cada item para impedir que sejam incluídos mais itens do que o disponível. Para superar essa dificuldade, alguns autores começaram a utilizá-la juntamente com outras ferramentas. Em [8] são propostas melhorias no algoritmo de busca em árvore presente em [9] obtendo novos limitantes para o problema a partir da relaxação do espaço de estado da programação dinâmica. Em [12] são utilizadas ideias presentes em [13,36] propondo uma heurística baseada na programação dinâmica. Os autores decompõem o problema em uma sequência de problemas da mochila, gerando faixas verticais e horizontais que são combinadas para formar o padrão de corte. Em [33] é proposto uma relaxação do espaço de estado junto com uma heurística de factibilização. Pesquisas mais recentes mostram que o problema ainda se mostra interessante e desafiador. Em [35] é proposta uma relaxação do espaço de es-

tado encontrando os pesos associados aos itens via programação inteira e em [20] são propostos diferentes modelos de otimização matemática para o problema sem limite de estágios de corte.

Modelos matemáticos são propostos para resolver o problema do corte considerando apenas 2-estágios. Em [18] são apresentados dois modelos de otimização linear inteira. O Modelo 1, que mais se destaca, associa uma variável binária diferente para cada item (mesmo que haja várias cópias do mesmo tipo). A estratégia é construir faixas horizontais respeitando as dimensões do objeto. Em [38] são propostos três modelos de otimização linear inteira, sendo dois deles adaptados de modelos da literatura. Resultados computacionais indicam a eficiência dos modelos.

Embora a evolução computacional aliada às modelagens matemáticas mais eficientes contribuam para que os *softwares* comerciais resolvam diversas classes de problemas de otimização eficientemente, ainda há problemas de otimização que se mostram desafiadores, mesmo que seja permitido tempo de resolução elevado (*e. g.* [21]). É nessa perspectiva que são discutidas nesse artigo heurísticas para o PCBG-2est. A heurística proposta é similar à apresentada em [12], no sentido da decomposição do problema PCBG-2est em uma sequência de problemas da mochila e à apresentada em [33] pela inclusão de uma etapa de factibilização. A diferença principal é que em [33] a factibilização é feita iterativamente na solução do problema relaxado durante a execução da heurística (método do tipo otimização do subgradiente) e em [12] a decomposição do problema é feita verticalmente e horizontalmente, gerando mais faixas, e são construídos dois padrões de corte.

Esse artigo tem a seguinte organização, na Seção 2 é abordado o problema da mochila restrito e uma adaptação do algoritmo de programação dinâmica proposto em [25, 28]. Na Seção 3 é apresentado em detalhes a heurística proposta para resolver o problema PCBG-2est. Resultados do estudo computacional são apresentados na Seção 4 e considerações finais são feitas na Seção 5.

2 O PROBLEMA DA MOCHILA

O Problema da Mochila (PM) aparece frequentemente como um subproblema na solução de diversos problemas de otimização e em particular de outros PCE. Para formular matematicamente o PM, considera-se que existe um conjunto $\mathscr{I}$ com m itens e um único objeto (uma mochila) com apenas uma dimensão relevante que é o seu peso máximo C (capacidade). Para cada item $i \in \mathscr{I}$ também é considerado relevante apenas uma dimensão que é o peso $p_i \leq C$, e é conhecido seu valor v_i. Vamos supor que os itens serão alocados na mochila um em seguida do outro, sem sobreposição, e tal que a soma dos pesos dos itens selecionados não exceda a capacidade máxima da mochila. Nesse caso, as restrições de contenção e de não sobreposição podem ser substituídas por uma única expressão matemática, a inequação (2.2). Considerando que o objetivo seja determinar um subconjunto de itens $\mathscr{I}^*$ que forneça o maior valor para a mochila, o PM é representado pelo modelo (2.1)-(2.3).

Essa formulação é equivalente ao problema da mochila binária ou $0-1$, pois considera-se que existe apenas uma unidade de cada item e a decisão associada a cada item é se ele será incluído ou

não na mochila. Segundo a tipologia proposta em [37], o problema é classificado como problema da mochila única unidimensional.

A versão que será explorada nesse texto considera que existem b_i cópias do item $i \in \mathscr{I}$ e a decisão a ser tomada é quantas cópias de cada item serão selecionadas, Problema da Mochila Inteiro (PMI). Para formular matematicamente, é necessário definir a variável de decisão y_i para representar o número de itens do tipo i que serão alocados na mochila, substituir a restrição (2.2) pela restrição (2.5) e incluir as restrições (2.6)-(2.7) que determinam o intervalo de variação da variável y_i e seu domínio. Se $b_i = 1, i \in \mathscr{I}$, o PMI torna-se equivalente ao problema da mochila binária. De acordo com a restrição (2.6), se para algum i, $b_i < \lfloor \frac{C}{p_i} \rfloor$, o problema é dito Problema da Mochila Restrito (PMR), caso contrário é considerado irrestrito. É necessário também reformular a função objetivo (2.1) para considerar a alocação de várias cópias de um mesmo item, conforme descrito na expressão (2.4). Segundo a tipologia apresentada em [37], esse problema é uma variação do problema de alocação de objeto único unidimensional e é formulado matematicamente de acordo com (2.4)-(2.7).

$$\sum_{i \in \mathscr{I}^*} v_i = Max \sum_{i \in \mathscr{I}} v_i \tag{2.1}$$

$$S.a: \sum_{i \in \mathscr{I}} p_i \leq C \tag{2.2}$$

$$\mathscr{I}^* \subseteq \mathscr{I} \tag{2.3}$$

$$Max \sum_{i=1}^{m} v_i \cdot y_i \tag{2.4}$$

$$S.a: \sum_{i=1}^{m} p_i \cdot y_i \leq C \tag{2.5}$$

$$0 \leq y_i \leq b_i,\ i = 1 \ldots m \tag{2.6}$$

$$y_i \in \mathbb{Z}_+, i = 1 \ldots m \tag{2.7}$$

2.1 Algoritmo de programação dinâmica para o PMI

A programação dinâmica é uma técnica que tem sido muito empregada em problemas de otimização cuja solução ótima pode ser obtida a partir da resolução de problemas de menor dimensão. O problema de otimização (original) é decomposto em estágios e em cada estágio são definidos estados. Cada combinação estágio/estado está associado a um problema de dimensão menor que o problema original (subproblema). Através de uma sequência de decisões ótimas associadas aos estágios/estados se obtém a solução ótima do problema original [2].

A formulação via programação dinâmica do PMI considera que um estágio k, $1 \leq k \leq m$, é definido pelo número de itens distintos considerado no subproblema, e o estado c, $0 \leq c \leq C$, é definido pela capacidade considerada. Para cada par, estágio k e estado c, temos um problema da mochila PMI_k^c conforme definido em (2.8) em que v e p são vetores k-dimensionais que representam, respectivamente, o valor e o peso dos itens, e $y \in \mathbb{Z}^k$ é um vetor cujas coordenadas

indicam o número de itens de cada tipo incluídos na mochila. A política ótima é obtida decidindo, para cada k $(k = 1 \ldots m)$ e c $(c = 0 \ldots C)$ se é melhor inserir o item k na mochila de capacidade c ou carregar seu estado presente para o próximo. Esse processo induz a fórmula recursiva (2.9) e sua respectiva solução (2.10).

$$PMI_k^c = Max\left\{v \cdot y \mid p \cdot y \leq c,\ y \in \mathbb{Z}_+^k\right\} \tag{2.8}$$

$$z_k(c) = Max\{j \cdot v_k + z_{k-1}(c - j \cdot p_k),\ j = 0 \ldots \lfloor c/p_k \rfloor,\ j \in \mathbb{N}\} \tag{2.9}$$

$$y_k(c) = \{j \mid z_k(c)\} \tag{2.10}$$

Para resolver o PMI utilizando (2.9) é necessário calcular todos os $y_k(c)$ e $z_k(c)$ até o último estado do último estágio, *i.e.* são resolvidos $m \cdot (C+1)$ subproblemas. Para poupar tempo e memória computacional, existem técnicas que eliminam estágios e estados que não levariam a uma solução ótima. Em [1] é proposta uma dominância limiar (*threshold dominance*), que pode ser vista como uma generalização de outras relações de dominância (simples, múltipla e coletiva). Um conjunto de itens do tipo $\mathscr{I}$ domina limiarmente o item j se $\sum_{i \in \mathscr{I}} p_i \cdot y_i \leq p_j \cdot y_j$ e $\sum_{i \in \mathscr{I}} v_i \cdot y_i \geq v_j \cdot y_j$ com $y_i, y_j \in \mathbb{Z}_+$. A dominância simples é dada se $\mathscr{I}$ é um conjunto unitário e $y_i = y_j = 1$.

Em [25, 28] essas relações de dominância são usadas para reduzir o espaço de estados e eliminar estágios em um algoritmo lexicográfico para resolver o problema da mochila inteira irrestrito. A estratégia é uma extensão do método proposto em [16] para o problema da mochila binária. Como é mostrado a seguir, o algoritmo é facilmente adaptável para tratar o caso restrito (PMR).

Considere o terno $(c, z_k(c), y_k(c))$ cuja primeira coordenada é a capacidade da mochila no estado c, a segunda coordenada é o valor do PMI no estágio k e estado c $(z_k(c))$, e a terceira coordenada é a solução do estágio k no estado c $(y_k(c))$. Cada estágio $k = 1 \ldots m$ é identificado como uma sequência S^k desses ternos, ordenados de forma crescente em relação à primeira coordenada, e não dominados de acordo com o critério de dominância simples. Dizemos que $(c, z_k(c), y_k(c))$ domina simplesmente o terno $(\bar{c}, z_k(\bar{c}), y_k(\bar{c}))$, se $c \leq \bar{c}$ e $z_k(c) \geq z_k(\bar{c})$. A sequência S^1 (2.11) é facilmente obtida a partir de múltiplos j do peso e valor do item 1. Os ternos com $j \cdot p_1 < c < (j+1)p_1$ para $j = 0 \ldots \lfloor C/p_i \rfloor - 1$ não são incluídos, pois são simplesmente dominados pelo de menor capacidade c [28].

$$S^1 = \{(0,0,0), (p_1, v_1, 1), \ldots, (\lfloor C/p_1 \rfloor \cdot p_1, \lfloor C/p_1 \rfloor \cdot v_1, \lfloor C/p_1 \rfloor)\} \tag{2.11}$$

A sequência S^k, com $k = 2 \ldots m$ é obtida a partir da sequência S^{k-1} como segue. Para cada $j = 0 \ldots \lfloor C/p_k \rfloor$ inteiro, são inicialmente construídas sequências S_j^k intermediárias obtidas pela soma de $(j \cdot p_k, j \cdot v_k, j)$ a cada terno de S^{k-1}. Essa soma é a usual do $\mathbb{R}^3$ e $S_0^k = S^{k-1}$. Em seguida, é aplicado um processo de *merge*, que é a união não dominada das S_j^k respeitando a ordem da primeira coordenada, ou seja, $S^k = \dot{\bigcup}_j S_j^k$. O resultado é uma sequência crescente, não dominada e viável de ternos [28].

O valor ótimo para o problema é encontrado no último terno de S^m $(z^* = z_m(c'))$ com c' sendo a capacidade da mochila utilizada. Para obter a solução referente à esse valor, necessita-se de um

processo de recuperação de solução iterativo. Inicie tomando o último terno de S^m, e recuperando a solução associada $y_m^* = y_m(c')$, $c' = C$. Depois, iterativamente, faça $c' = c' - p_k \cdot y_k^*$ e obtenha $y_{k-1}^* = y_{k-1}(c')$ para $k = m \ldots 2$. A solução ótima será dada pelo vetor m-dimensional y^* [28].

Observe que nesse método, o único fator que limita a quantidade do item k é a capacidade da mochila, $y_k \leq \lfloor C/p_k \rfloor$. Para tratar o caso restrito basta redefinir o intervalo de variação de j para $j = 0 \ldots \min\{b_k, \lfloor C/p_k \rfloor\}$ na definição de S_j^k. Essa adaptação será chamada de PMR^{PD}, e o pseudocódigo é apresentado no Algoritmo 1. Os dados de entrada são os pesos (p), valores (v) e limites (b) dos itens, e a capacidade C da mochila. A saída é o vetor S m-dimensional em que cada coordenada é uma matriz de três colunas com a sequência S^k. Cada linha dessa matriz é um terno da sequência S^k conforme (2.12).

$$S = \left\{ \begin{bmatrix} 0 & 0 & 0 \\ p_1 & v_1 & 1 \\ 2p_1 & 2v_1 & 2 \\ \vdots & \vdots & \vdots \\ \lfloor C/p_1 \rfloor p_1 & \lfloor C/p_1 \rfloor v_1 & \lfloor C/p_1 \rfloor \end{bmatrix}, \cdots, \begin{bmatrix} 0 & 0 & 0 \\ s_{21}^m & s_{22}^m & s_{23}^m \\ s_{31}^m & s_{32}^m & s_{33}^m \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ s_{n1}^m & s_{n2}^m & s_{n3}^m \end{bmatrix} \right\} \tag{2.12}$$

Na linha 1 do Algoritmo 1, o vetor S é inicializado com os ternos nulos que iniciam cada uma das sequências S^k. Nas linhas 2 a 15 as sequências S^k $(k = 1 \ldots m)$ são construídas iterativamente. Inicialmente, linhas 3 a 6, recupera-se S^{k-1}. Nas linhas 7 e 8 as matrizes M e M_{op} são inicializadas com a S^k atual. Nas linhas 9 a 12 são construídas as sequências S_j^k por meio da matriz M_{op} e armazenadas em M de forma iterativa. A restrição (2.6) referente ao limite máximo do item k é considerada na linha 9. Na linha 13 a função *Merge* (ordenação e eliminação de ternos dominados ou inviáveis) é chamada para a obtenção de S^k. Na linha 14 armazena-se S^k em S.

Algoritmo 1 PMR via Programação Dinâmica (Adaptado de [25])

Entrada: p, v, b, C
Saída: S vetor de matrizes de resultados

1 $S = \{[0\ 0\ 0], [0\ 0\ 0], \ldots, [0\ 0\ 0]\}$ {vetor que recebe iterativamente a solução do PMI com os k primeiros itens}
2 **para** $k = 1 \ldots m$ **faça**
3 **se** $k > 1$ **então**
4 Faça a coordenada k em S igual à coordenada $k-1$ de S
5 Faça a terceira coluna da coordenada k de S igual a 0
6 **fim**
7 $M = S^k$
8 $M_{op} = S^k$
9 **para** $j = 1 \ldots min\{b_k, \lfloor C/p_k \rfloor\}$ **faça**
10 $M_{op} = M_{op} .+ [p_k\ \ v_k\ \ 1]$ {o sinal $(.+)$ indica a soma de $[p_k\ v_k\ 1]$ aos elementos de cada linha de M_{op}}
11 Insira as linhas de M_{op} no final de M
12 **fim**
13 Aplique a função *Merge* a M {ordenação e eliminação das linhas dominadas ou inviáveis}
14 Faça S^k igual à M
15 **fim**

Através da estrutura da programação dinâmica é possível recuperar facilmente soluções ótimas para o PMR considerando instâncias do problema com $k < m$ itens. Essa possibilidade será explorada na Seção 3. Salientamos que o Algoritmo 1 mostrou ser eficiente para os testes realizados no contexto deste artigo, outros métodos de solução para o PMR podem ser encontrados, por exemplo, em [26, 27].

3 O PROBLEMA DE CORTE BIDIMENSIONAL

No problema de corte bidimensional considerado nesse artigo, temos um único objeto retangular e duas dimensões relevantes para o corte, o comprimento L e a largura W. Os itens também são retangulares e duas dimensões são relevantes, o comprimento l_i e a largura w_i $(i = 1 \ldots m)$. O critério de otimização é a maximização da área total cortada. Sob essas condições, o problema pode ser classificado como uma variação do problema de alocação de objeto único bidimensional.

Uma solução factível para esse problema é chamada de padrão de corte conforme a Definição 3.1. A sua modelagem matemática não é trivial (*e.g.* [18]), diversos fatores como geometria, posicionamento e rotação devem ser considerados. É importante também considerar outras características que são referentes às restrições do equipamento de corte. Em algumas indústrias, a máquina de corte só realiza cortes guilhotinados ortogonais, Definição 3.2, que é o caso abordado nesse artigo. Para obter todos os itens em um padrão de corte guilhotinado é necessário fazer rotações de 90 graus no objeto, o que define o número de estágios do padrão de corte conforme a Definição 3.3. Em algumas situações pode ser ainda necessário limitar a quantidade máxima de um ou mais itens na solução do problema. Nesse caso, assim como no problema da mochila inteira, dizemos que o problema é restrito, e uma solução factível é chamada de padrão de corte restrito (Definição 3.4).

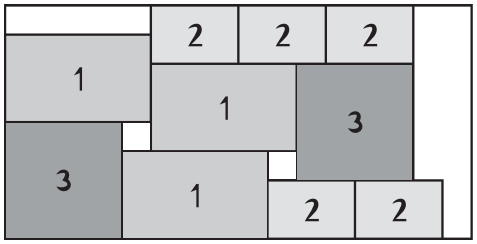
Definição 3.1. *Um padrão de corte é uma solução factível para um problema de corte e indica como os itens estão posicionados (alocados) no objeto antes de serem cortados respeitando as condições de não sobreposição e contenção dos itens no objeto. Sua representação algébrica é dada pelo vetor coluna $A_j \in Z^m$, com a_{ij} representando a frequência do item i no padrão de corte j.*

Definição 3.2. *(i) Um corte guilhotinado ortogonal em um objeto retangular é um corte paralelo a um dos lados do objeto e realizado de uma extremidade a outra, obtendo-se dois retângulos menores. (ii) Um padrão de corte é guilhotinado se para obter todos os itens são realizados apenas cortes guilhotinados, (iii) caso contrário, o padrão de corte é não guilhotinado.*

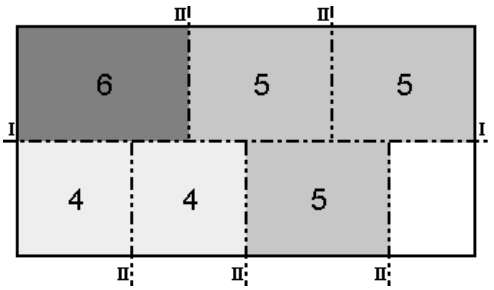
Definição 3.3. *(i) Um padrão de corte guilhotinado é k-estágios se são feitos k-1 rotações de 90 graus no objeto para obter todos os itens. (ii) O padrão de corte é exato se ao final do último estágio todos os itens forem obtidos sem precisar de aparas. (iii) Caso contrário o padrão de corte é não exato.*

Definição 3.4. *Um padrão de corte é restrito se a frequência de pelo menos um item i no padrão de corte tem um limite máximo* $b_i < \left\lfloor \frac{L}{l_i} \right\rfloor \cdot \left\lfloor \frac{W}{w_i} \right\rfloor$. *Caso contrário, ele é irrestrito.*

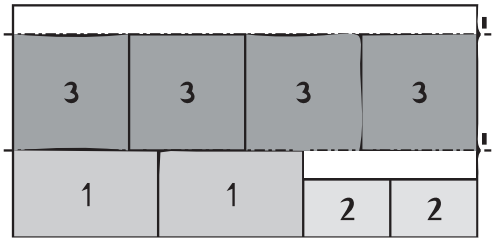
Considerando as definições 3.1, 3.2, 3.4, e apenas uma rotação do objeto (k=2 na Definição 3.3), temos o problema de Corte Bidimensional Guilhotinado 2-estágios restrito (PCBG-2est). Na Figura 1 são apresentados três padrões de corte distintos, sendo cada item representado por um número e os espaços em branco representando o desperdício do corte. Na Figura 1a o padrão de corte é não guilhotinado. Note que qualquer tentativa de fazer cortes guilhotinados ocasiona dano em pelo menos um item. O padrão de corte ilustrado na Figura 1b é 2-estágio exato, pois ao executar 6 cortes (1 no estágio 1 (I) e 5 no estágio 2 (II)) todos os itens serão obtidos (não há necessidade de aparas). Na Figura 1c é ilustrado um padrão 2-estágios não exato, pois após os cortes guilhotinados do estágio 1, indicados por I, é necessário rotacionar o objeto em 90 graus para fazer os cortes do segundo estágio, indicados por II. Ao final do segundo estágio, para obter cada um dos itens 2 é necessário um processo de apara. Cabe dizer que o processo de apara pode ser considerado como um novo estágio ou não, isso irá depender das condições da indústria, pois algumas fazem as aparas na máquina principal (nesse caso temos um novo estágio), enquanto outras fazem as aparas em uma máquina secundária.



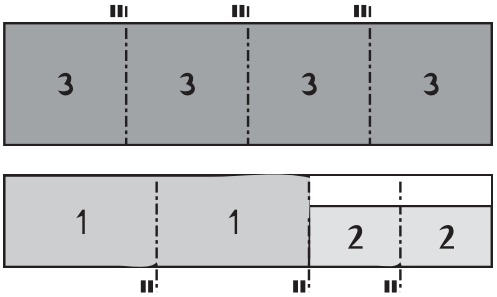
(a) Padrão de corte não guilhotinado.



(b) Padrão de corte 2-estágio exato.



(c) Padrão de corte 2-estágios não exato (estágio 1).



(d) Padrão de corte 2-estágios não exato (estágio 2).

Figura 1: Diferentes tipos de padrões de corte.

3.1 Adaptação do Método em duas etapas de Gilmore e Gomory para o caso restrito

Para resolver o PCBG-2est usamos uma adaptação do método de duas etapas proposto em [13] que gera padrões bidimensionais 2-estágios (*e.g.* [10]). Inicialmente os itens são ordenados de forma que $w_i \leq w_{i+1}, i = 1...m$. Na primeira etapa do método de Gilmore e Gomory adaptado (MGGA) são geradas m faixas de largura w_k $(k = 1...m)$ e comprimento L, considerando apenas os itens de largura $w_i \leq w_k$, resolvendo m problemas da mochila restritos (3.1)-(3.4). A variável γ_{ki} $(k, i = 1 \ldots m)$ indica a quantidade do item i na faixa k e é limitada superiormente pelo parâmetro b_i (Restrição (3.3)) e π_i é a área do item i. Na segunda etapa do MGGA, as faixas são combinadas para gerar um padrão de corte resolvendo o problema de otimização (3.5)-(3.8). A variável β_k é o número de vezes que a faixa k é incluída no padrão de corte, e π^* é o valor da faixa k (valor ótimo do k-ésimo PMR resolvido na primeira etapa).

$$\pi^*_{k=1\ldots m} = Max \sum_{i=1}^{k} \pi_i \cdot \gamma_{ki} \tag{3.1}$$

$$S.a : \sum_{i=1}^{k} l_i \cdot \gamma_{ki} \leq L \tag{3.2}$$

$$\gamma_{ki} \leq b_i,\ i = 1 \ldots k \tag{3.3}$$

$$\gamma_{ki} \in \mathbb{Z}_+, i = 1 \ldots k \tag{3.4}$$

$$Max \sum_{k=1}^{m} \pi^*_k \cdot \beta_k \tag{3.5}$$

$$S.a : \sum_{k=1}^{m} w_k \cdot \beta_k \leq W \tag{3.6}$$

$$\gamma^{*T} \cdot \beta \leq b \tag{3.7}$$

$$\beta_k \in \mathbb{Z}_+, k = 1 \ldots m \tag{3.8}$$

No método de Gilmory e Gomory original, o problema a ser resolvido na segunda etapa é um PMI. No MGGA proposto para resolver PCBG-2est é necessário acrescentar um novo conjunto de restrições que formam um sistema triangular superior (3.7), e a estrutura de um problema da mochila é perdida. Ao multiplicar a transposta de γ^* pelo vetor coluna β obtêm-se exatamente a quantidade de cada item no padrão de corte. Com isso, as restrições (3.7) limitam o máximo de vezes que cada item é incluído no padrão de corte. A matriz triangular inferior γ^* associada a essas restrições é obtida a partir das soluções ótimas dos m PMR resolvidos na primeira etapa.

É possível utilizar o Algoritmo 1 proposto na Seção 2.1 para resolver simultaneamente todos os PMR da primeira etapa, considerando que $p = l$, $v = \pi$ e $C = L$. Para realizar a segunda etapa é necessário recuperar m soluções ótimas, ao invés de apenas a solução para o estágio $k = m$ e estado $c = C$ conforme discutido na Seção 2. Um algoritmo para recuperar essas soluções ótimas pode ser visto em [3].

Como a estrutura de um problema da mochila é perdida na segunda etapa, propomos na Seção 3.2 uma heurística de relaxação para resolver o problema (3.5)-(3.8) usando o Algoritmo 1.

3.2 Heurística para resolver o Problema de Corte Bidimensional 2-estágios Restrito

A heurística de relaxação (H^{PD}) proposta para resolver o problema de otimização da segunda etapa do MGGA possui quatro fases: (i) relaxar o problema (3.5)-(3.8) retirando o conjunto de restrições (3.7); (ii) definir novos limites superiores para a frequência j da faixa k (β_k); (iii) utilizar o Algoritmo 1 para resolver o problema relaxado formado pelas expressões (3.5, 3.6, 3.8) e considerando os novos limites superiores para β_k; (iv) aplicar um processo de factibilização. A seguir são apresentados os detalhes das fases (ii) e (iv).

Para definir novos limitantes para β_k, fase (ii) da heurística H^{PD}, considere inicialmente que o número máximo de vezes que uma faixa k pode ser incluída no padrão depende da sua largura w_k, assim $\beta_k \leq \lfloor W/w_k \rfloor$. Supondo que para cada k $(k = 1,\ldots,m)$, somente as inequações i $(i = 1\ldots k)$, tenham os escalares γ^*_{ki} diferente de zero, pode-se verificar que $\beta_k \leq min\left\{\lfloor b_i/\gamma^*_{ki} \rfloor, i = 1\ldots k\right\}$. Se considerarmos ainda o limite superior para o número de itens no padrão de corte, podemos definir um limite superior para β_k de acordo com (3.9).

$$\beta_k \leq \bar{b}_k = min\{b_k, \lfloor W/w_k \rfloor, min\{\lfloor b_i/\gamma^*_{ki} \rfloor, i = 1\ldots k\}\}, k = 1\ldots m \tag{3.9}$$

O uso da programação dinâmica (Algoritmo 1) para resolver o problema de otimização na fase (iii) permite a recuperação de m padrões de corte. No entanto, alguns destes padrões podem ser infactíveis para o PCBG-2est por conter mais itens que o permitido. Nesse caso é necessário aplicar a fase (iv) da heurística H^{PD}. A factibilização de um padrão de corte consiste na retirada dos itens em excesso. Isso pode fazer com que o padrão de corte associado ao estágio $k = m$ e estado $c = C$ do Algoritmo 1 tenha um valor menor do que o valor de algum outro padrão de corte. Por esse motivo a heurística retorna, além de todos os m padrões de corte, o de maior valor total.

O pseudocódigo da heurística H^{PD} está descrito no Algoritmo 2. As entradas do algoritmo são a matriz γ^* e os parâmetros para utilizar o Algoritmo 1 ($p = w, v = \pi^*, C = W$). A saída é a matriz A cujas colunas representam os m padrões de corte e A_{j^*} o de maior valor. As linhas 1, 2 e 3 são respectivamente as fases (i), (ii) e (iii). Nas linhas 6-12 é feito a factibilização dos m padrões obtidos e na linha 13 é avaliado o valor dos m padrões e se armazena o de maior valor em A_{j^*}.

Em resumo, a heurística PC^{PD}_H proposta para resolver o PCBG-2est consiste em:

- Ordenar os itens de forma que $w_i \leq w_{i+1}$, $i = 1\cdots m$;
- Etapa 1: Aplicar o Algoritmo 1 ($p = l$, $v = \pi$ e $C = L$) para obter m faixas de largura w_k $(k = 1...m)$ e comprimento L;
- Etapa 2: Aplicar o Algoritmo 2 para obter um padrão de corte restrito.

Para ilustrar a heurística PC^{PD}_H considere um objeto de dimensões $(L,W) = (165,70)$ e três itens $(l_i, w_i, b_i) \in \{(30,23,5),(45,45,6),(70,56,2)\}$, já ordenados de acordo com $(w_i \leq w_{i+1}, \forall i)$. Na Etapa 1, após a aplicação do Algoritmo 1, são obtidas três faixas de comprimento 165 ilustradas

Algoritmo 2 Heurística H^{PD}

Entrada: w, π^*, b, W, γ^*
Saída: A matriz de padrões de corte e A_{j^*} o de maior valor

1 Relaxe o problema (3.5)-(3.8) retirando (3.7)
2 Calcule os novos limites superiores ($\bar{b}_k$) para β_k de acordo com (3.9)
3 Resolva o problema de otimização (3.5,3.6, 3.8) pelo Algoritmo 1 considerando que $p = w$, $v = \pi^*$, $b = \bar{b}$, $C = W$
4 Recupere as m soluções do problema resolvido na linha 3 e guarde em β^*
5 Recupere os m padrões de corte A_j: $a_{ij} = \sum_{k=1}^{j} \gamma_{ki}^* \beta_{jk}^*$ para todo $j, i = 1 \ldots m,\ i \leq j$
6 **para** $j = 1 \ldots m$ **faça**
7 **para** $i = 1 \ldots j$ **faça**
8 **enquanto** $a_{ij} > b_i$ **faça**
9 $a_{ij} = a_{ij} - 1$
10 **fim**
11 **fim**
12 **fim**
13 Calcule o valor de cada padrão de corte e armazene em A_{j^*} o de maior valor.

na Figura 2a e armazenadas na matriz γ^* exibida em (3.10). A faixa 1, de largura 23, possui 5 itens 1 (linha 1 da matriz), a faixa 2, de largura 45, possui 1 item 1 e 3 itens 2 (linha 2 da matriz), e a faixa 3, de largura 56, possui 2 itens 2 e 1 item 3 (linha 3 da matriz). Considerando π_i^* a área utilizada pela faixa i obtemos o problema $m+1$ do MGGA expresso em (3.12)-(3.17).

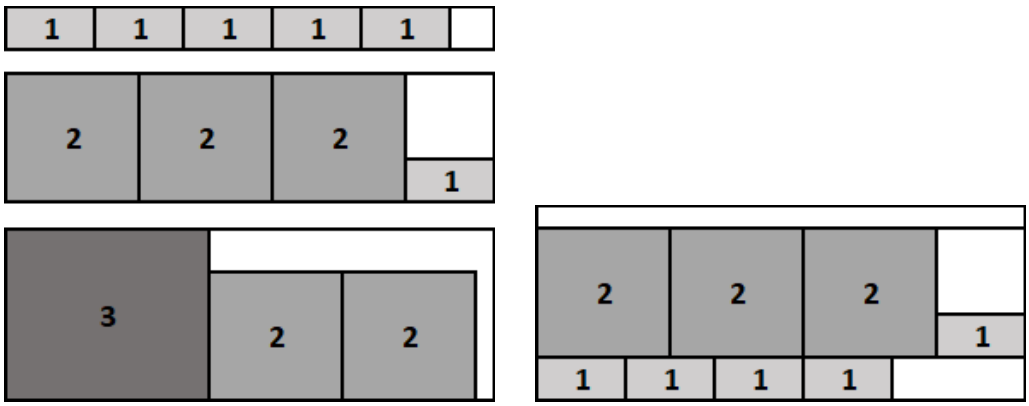
$$\gamma^* = \begin{bmatrix} 5 & 0 & 0 \\ 1 & 3 & 0 \\ 0 & 2 & 1 \end{bmatrix} \tag{3.10}$$

$$(\beta^*)^T = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

$$Max\ 3450\beta_1 + 6765\beta_2 + 7970\beta_3 \tag{3.12}$$
$$S.a: 23\beta_1 + 45\beta_2 + 56\beta_3 \leq 70 \tag{3.13}$$
$$5\beta_1 + 1\beta_2 + 0\beta_3 \leq 5 \tag{3.14}$$
$$0\beta_1 + 3\beta_2 + 2\beta_3 \leq 6 \tag{3.15}$$
$$0\beta_1 + 0\beta_2 + 1\beta_3 \leq 2 \tag{3.16}$$
$$\beta \in \mathbb{Z}_+^3 \tag{3.17}$$

Na Etapa 2 utilizamos o Algoritmo 2 para obter padrões de corte restritos. Um (PMR) é obtido removendo-se as restrições (3.14)-(3.16) do modelo de otimização obtido na Etapa 1 (linha 1). Novos limites superiores para a variável β são calculados: $\bar{b}_k = 1$ para $k = 1 \ldots 3$ (linha 2). O Algoritmo 1 é então aplicado para combinar as faixas e obter os três padrões de corte que são armazenados na matriz β^{*T} exibida em (3.11) (linha 3). O primeiro padrão de corte é composto por uma faixa 1 (coluna 1), o segundo e terceiro padrões de corte são iguais e compostos por uma

faixa 1 e uma faixa 2 (colunas 2 e 3). Esses padrões de corte são armazenados como: $A_1 = [5\ 0\ 0]^T$ e $A_2 = A_3 = [6\ 3\ 0]^T$ (linha 5), note que os padrões de corte 2 e 3 são infactíveis (excedem em uma unidade o item 1). Para factibilizar os padrões de corte, uma unidade do item 1 é retirada dos padrões de corte 2 e 3 (linhas de 6 até 12) obtendo $A_2 = A_3 = [5\ 3\ 0]^T$. Na linha 13 do Algoritmo 2 o padrão de corte de maior valor ($A_2 = A_3$) é armazenado em A_j^*. Na Figura 2b é exibido o padrão de corte obtido após a factibilização.



(a) Faixas geradas na Etapa 1 da heurística PC_H^{PD}. (b) Padrão de corte após factibilização.

Figura 2: Faixas e Padrões de corte gerados pela Heurística PC_H^{PD}.

4 ESTUDO COMPUTACIONAL

Nesta seção são apresentados os resultados do estudo computacional realizado para avaliar o desempenho da heurística PC_H^{PD} descrita na Seção 3. Como os métodos propostos em [12, 23, 33, 34, 36] não limitam o número de estágios no problema de corte, duas outras estratégias de solução são empregadas para *benchmark*: a heurística PC_G^{PD} que difere da heurística PC_H^{PD} por usar um resolvedor genérico na Etapa 2; e a estratégia M1LM que consiste na resolução direta de instâncias do modelo de otimização inteira mista (Modelo 1) proposto em [18]. Para a estratégia M1LM utilizamos b_i como sendo o mínimo entre o número de cópias do item i e a quantidade de vezes que o item i cabe no objeto. Esse modelo foi escolhido por ser o mais citado na literatura para resolver o PCBG-2est (*e.g.* [21]). As heurísticas PC_H^{PD} e PC_G^{PD}, e o modelo M1LM foram codificados na linguagem Júlia usando o pacote JuMP [6]. O resolvedor genérico utilizado foi o Cplex versão 12.1 [17] com parâmetros *default*. O estudo computacional foi realizado em uma máquina com processador Core i7 de quarta geração e 3.50 GHz de CPU, memória RAM de 32 GB e sistema operacional Windows 10 Pro de 64 bits.

Foram usados três conjuntos de instâncias, para todas consideramos somente padrões de corte 2-estágios restrito com faixas horizontais e sem rotação dos itens. O Conjunto 1 consiste em 15 instâncias da literatura com soluções ótimas conhecidas. Essas instâncias são consideradas pequenas e com poucas cópias de cada item, $m \in [10,50]$ e $b_i \in [1,5]$ $(i = 1,\ldots,m)$. As instâncias

OF1, OF2 [23] e wang20 [36] consideram a dimensão do objeto igual a 70×40. As instâncias gcut1-gcut4, gcut5-gcut8 e gcut9-gcut12 [24] consideram objetos quadrados de lado 250, 500 e 1000, respectivamente. É importante dizer que as instâncias gcut1-gcut12 foram propostas inicialmente para o problema irrestrito e adaptadas limitando em uma cópia para cada item (*e.g.* [21]). O objetivo de usar essas instâncias é investigar o comportamento das heurísticas PC_H^{PD} e PC_G^{PD} considerando as seguintes perguntas: As heurísticas obtiveram solução ótima? No caso positivo, qual parte do método contribuiu para isso? No caso negativo, o que provocou este resultado?

O Conjunto 2 é composto por instâncias geradas aleatoriamente usando o 2DCPackGen proposto em [32]. Foi usada a semente 2021 (seed), tipo de objeto 2 (longo e estreito (ID_o)), tipo de itens de 1 à 16 (ID_i), $m = \{10, 20, 40\}$, com distribuição 5 (TD). Para cada combinação (ID_i, m) foram geradas 5 instâncias (NI), totalizando $16 \times 3 \times 5 = 240$ instâncias. A dimensão dos objetos varia de 100 à 200 (D_o), dos itens de 25 à 100 (D_i) e $b_j \in [1, 5]$ $(j = 1, \ldots, m)$. Para essas instâncias a área dos itens em relação à área do objeto varia em média de 8,81% e 35,27%. O objetivo de usar essas instancias é analisar a robustez das heurísticas PC_H^{PD} e PC_G^{PD}.

O Conjunto 3 contém instâncias geradas aleatoriamente usando o 2DCPackGen e que simulam cenários encontrados na indústria de móveis (*e.g.* [4]). Foram utilizados os seguintes parâmetros: $ID_i = \{1, 10\}$, $m = \{10, 20, 40, 100\}$, $NI = 10$, $D_o \in [1850, 2750]$, $D_i \in [50, 400]$ e $b_j \in [40, 200]$ $(j = 1, \ldots, m)$. Os parâmetros omitidos são os mesmos do Conjunto 2, totalizando $2 \times 4 \times 10 = 80$ instâncias. Nesse conjunto de instâncias a área dos itens em relação à área do objeto varia em média de 0,29% e 1,76%. O objetivo de usar esse conjunto é analisar o comportamento das heurísticas com instâncias mais realistas.

4.1 Resultados para as instâncias do Conjunto 1

Na Tabela 1 são apresentados os resultados obtidos pelos algoritmos PC_H^{PD} e PC_G^{PD} para as instâncias do Conjunto 1. Nas colunas 1, 2 e 3 é exibido o nome das instâncias, total de itens e sua solução ótima. Nas colunas 4-6 e 7-9 são apresentados a solução (sol), gap e tempo computacional em segundos (t(s)) para cada instância e método respectivamente. O gap foi calculado em relação à solução ótima (sol^*) da literatura do seguinte modo $gap = 100 * (sol^* - sol)/(sol + 10^{-10})$.

Pela Tabela 1, o algoritmo PC_G^{PD} obteve a solução ótima em 14 das 15 instâncias (93,33%) e seu tempo de execução foi em torno de 4 segundos. Uma explicação para esse resultado é que as faixas geradas na Etapa 1 podem não ser as faixas ótimas para o problema de corte restrito. Isso acontece porque ao gerar as faixas (Etapa 1), não é levado em conta que ao combiná-las para formar o padrão de corte o número total dos itens de um mesmo tipo pode exceder o seu limite superior. Isto é, usando apenas as faixas obtidas na Etapa 1, nem sempre é possível obter o padrão de corte ótimo para o problema restrito.

A heurística PC_H^{PD} obteve a solução ótima para 11 das 15 instâncias (73,33%) com tempo de execução abaixo de 2 segundos. O gap médio das soluções factíveis obtidas para as outras quatro

Tabela 1: Resultados para as instâncias do Conjunto 1.

		Literat.	PC_H^{PD}			PC_G^{PD}		
Instância	m	*sol**	sol	gap(%)	t(s)	sol	gap(%)	t(s)
OF1	23	2.713	2.713	0,00	1,81	2.713	0,00	3,94
OF2	22	2.515	2.515	0,00	1,82	2.515	0,00	3,93
wang20	42	2.623	2.623	0,00	1,84	2.623	0,00	3,93
gcut1	10	43.024	43.024	0,00	1,80	43.024	0,00	3,92
gcut2	20	57.996	57.996	0,00	1,81	57.996	0,00	3,98
gcut3	30	59.895	59.895	0,00	1,97	59.895	0,00	3,96
gcut4	50	60.504	60.504	0,00	1,80	60.504	0,00	3,94
gcut5	10	193.379	170.411	11,88	1,81	193.379	0,00	4,07
gcut6	20	224.399	224.399	0,00	1,85	224.399	0,00	3,96
gcut7	30	238.974	238.974	0,00	1,81	238.974	0,00	3,93
gcut8	50	245.758	245.758	0,00	1,81	245.758	0,00	4,17
gcut9	10	919.476	696.847	24,21	1,78	878.821	4,42	3,97
gcut10	20	856.445	664.464	22,42	1,80	856.445	0,00	4,28
gcut11	30	942.219	687.428	27,04	1,80	942.219	0,00	3,99
gcut12	50	970.744	970.744	0,00	1,80	970.744	0,00	3,99

instâncias foi de 21,39%. Além da mesma razão já elencada para o desempenho do algoritmo PC_G^{PD}, existe outra justificativa para o desempenho do algoritmo PC_H^{PD} que ocorre na presença do cenário CN1.

CN1: Suponha que as faixas ótimas para o problema de corte foram geradas em algum dos m problemas da mochila. Além disso, existe pelo menos uma faixa (ótima ou não) muito atrativa.

Ao fazer a combinação das faixas, o método PC_H^{PD} usa uma estratégia gulosa que privilegia faixas de maior valor e atribui a elas uma frequência maior do que a frequência ótima. Na presença do cenário CN1, essa estratégia faz com que o total de um ou mais itens no padrão ultrapasse seu limite superior, sendo então necessário realizar a etapa de factibilização, o que leva a perda de otimalidade. Esse cenário ocorreu nas instâncias gcut5, gcut10 e gcut11, para as quais o algoritmo PC_G^{PD} obteve o padrão de corte ótimo.

Em resumo, para as instâncias do Conjunto 1 o algoritmo PC_G^{PD} obteve 20% a mais de soluções ótimas do que o algoritmo PC_H^{PD}. Em relação a tempo de execução, o algoritmo PC_H^{PD} obteve um tempo aproximadamente 180% melhor do que o algoritmo PC_G^{PD}.

4.2 Resultados para as instâncias do Conjunto 2

Os resultados para as instâncias do Conjunto 2 são exibidos através do perfil de desempenho proposto em [11]. Através de um teste de hipótese, esse gráfico no plano cartesiano fornece uma fer-

ramenta para fazer comparação entre dois ou mais conjuntos de resultados de diferentes métodos a partir de um valor de referência. Para cada conjunto de resultados, o teste de hipótese é: "o resultado obtido pelo método está a uma distância menor ou igual a τ do valor de referência?". Para cada instância resolvida, o melhor resultado obtido entre os métodos é considerado o valor de referência. O eixo x representa a variação do fator τ e o eixo y informa quantas instâncias satisfizeram o teste de hipótese (em porcentagem) para esse fator τ. Assim, para $\tau = 0$ temos, em porcentagem, a quantidade de instâncias em que o método obteve os melhores resultados. Conforme o fator τ cresce, mais distante o resultado obtido pelo método está do valor de referência. As Figuras 3 e 4 obtidas com a planilha disponibilizada em [22] exibem o perfil de desempenho associados ao gap e ao tempo computacional. Para esse conjunto de instâncias o gap foi calculado em relação ao limitante superior (*LimSup*) obtido pela estratégia M1LM da seguinte forma $gap = 100 * (LimSup - sol)/(sol + 10^{-10})$.

A solução ótima foi obtida para 100% das instâncias usando a estratégia M1LM, enquanto que os algoritmos PC_H^{PD} e PC_G^{PD} encontraram a solução ótima para aproximadamente 86% e 92% das instâncias, respectivamente, conforme pode ser visto na Figura 3. Em comparação ao Conjunto 1, a heurística PC_G^{PD} obteve decréscimo de 1,33% no desempenho, enquanto a heurística PC_H^{PD} obteve uma melhora de 13,33% no desempenho. Verificou-se que o cenário CN1 ocorreu em aproximadamente 7% das instâncias.

Analisando a Figura 4 relativa ao tempo computacional, nota-se que o algoritmo PC_H^{PD} obteve o melhor tempo em 100% das instâncias com um tempo computacional máximo de 1,95 segundos. O algoritmo PC_G^{PD} obteve um desempenho inferior que a estratégia M1LM para 96% das instâncias, mas a diferença máxima de tempo computacional em relação ao algoritmo PC_H^{PD} foi de 2,55s a mais, enquanto que a estratégia M1LM teve um tempo computacional médio de 2,62 segundos mas com diferença máxima de 12,49 segundos a mais.

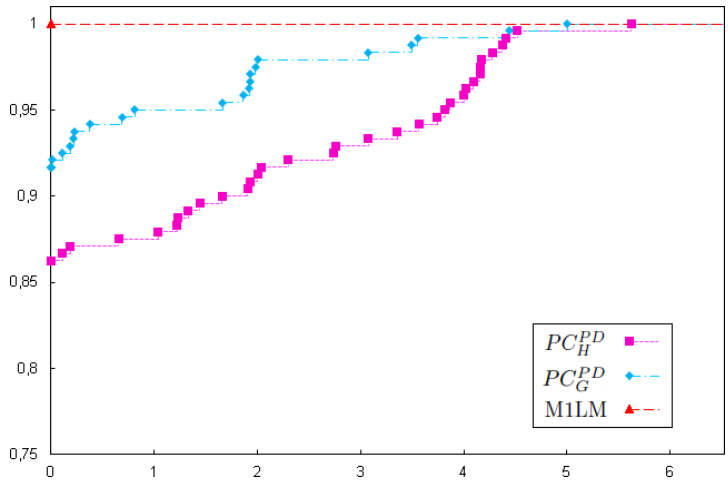


Figura 3: Perfil de desempenho do gap (Conjunto 2).

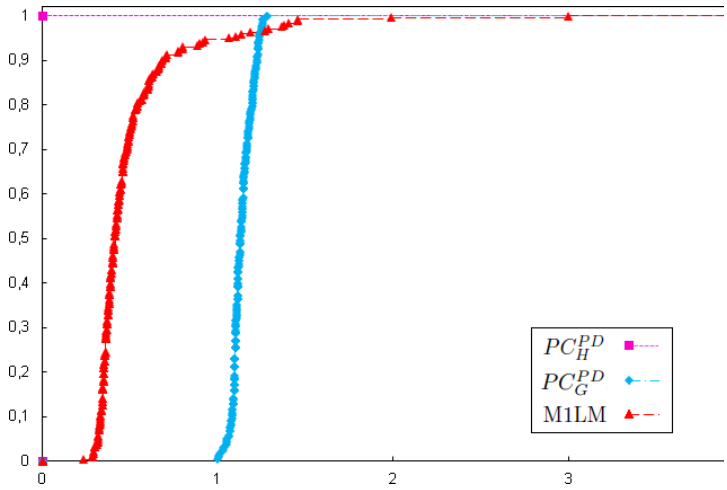


Figura 4: Perfil de desempenho do tempo (Conjunto 2).

Os resultados também foram analisados considerando o tipo de item. Por essa análise, constatou-se que nas instâncias com os itens de tipo 1 (itens pequenos e quadrados) os algoritmos PC_H^{PD} e PC_G^{PD} obtiveram solução ótima em 46,66% e 60% das instâncias, respectivamente (pior desempenho), e as com tipo 10 (itens retangulares com comprimento longo e altura média) obtiveram solução ótima em todas as instâncias (melhor desempenho).

4.3 Resultados para as instâncias do Conjunto 3

O perfil de desempenho relativo ao gap e tempo computacional para as instâncias do Conjunto 3 são exibidas nas Figuras 5 e 6, respectivamente. Além dos resultados para PC_H^{PD} e PC_G^{PD}, são exibidos também o perfil de desempenho para três variações do método M1LM. As variações são relativas ao tempo máximo de resolução definidos em 5 ($M1LM_5$), 10 ($M1LM_{10}$) e 20 ($M1LM_{20}$) minutos. É relevante dizer que na estratégia $M1LM_{20}$ a execução foi interrompida antes do tempo total por falta de memória. Aumentar o tempo de execução do Cplex não proporcionaria melhoras significativas. Para esse conjunto de instâncias consideramos a área do objeto como limitante superior (*LimSup*), sendo o mesmo para todos os métodos de solução. O gap foi calculado pela mesma expressão utilizada para o Conjunto 2.

De acordo com o perfil de desempenho referente ao gap exibido na Figura 5, as heurísticas PC_H^{PD} e PC_G^{PD} obtiveram as melhores soluções em aproximadamente 88% e 92% das instâncias, respectivamente, enquanto a estratégia M1LM obteve melhores soluções em menos de 8% das instâncias, para as três variações. Além disso, os algoritmos conseguiram obter uma solução factível para todas as instâncias com gap máximo médio de 4.4%. A estratégia M1LM obteve solução factível para 77,5% das instâncias, em sua melhor variação ($M1LM_{20}$). Para as instâncias em que a estratégia $M1LM_{20}$ não obteve solução factível (22,5% do total), em 55% delas o Cplex retornou a mensagem *"Warning: MIP starts not constructed because of out-of-memory*

status". Isto é, o Cplex não conseguiu iniciar a resolução da instância do modelo devido a falta de memória para armazenamento (nas instâncias com m = 100, o número médio de variáveis e restrições é da ordem de 10^7 e 10^4 respectivamente).

Em relação ao tempo computacional (Figura 6), os algoritmos PC_H^{PD} e PC_G^{PD} tiveram o melhor tempo em aproximadamente 38% e 61% das instâncias, respectivamente, enquanto a estratégia $M1LM$ obteve tempo computacional maior para todas as instâncias. Podemos perceber que nas instâncias do Conjunto 3 a heurística PC_G^{PD} foi melhor do que a PC_H^{PD} em relação ao tempo computacional. O problema de otimização na Etapa 2 é resolvido rapidamente pelo Cplex. Em contrapartida, a heurística PC_H^{PD} resolve m problemas da mochila na Etapa 2 simultaneamente por programação dinâmica, e para as instâncias do Conjunto 3 o espaço de estado é de grande porte provocando maiores tempos de execução.

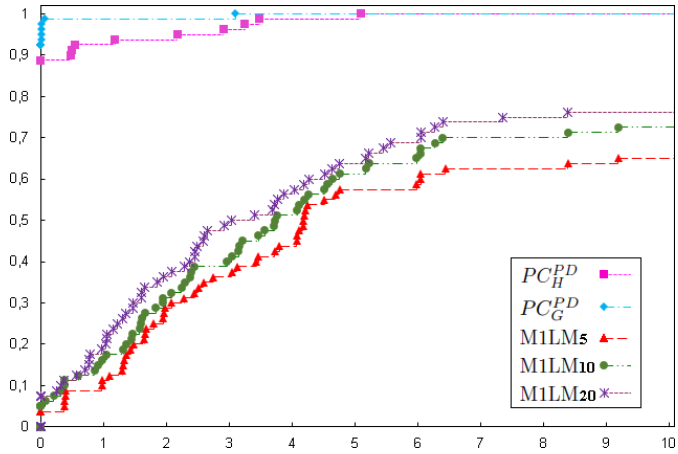


Figura 5: Perfil de desempenho do gap (Conjunto 3).

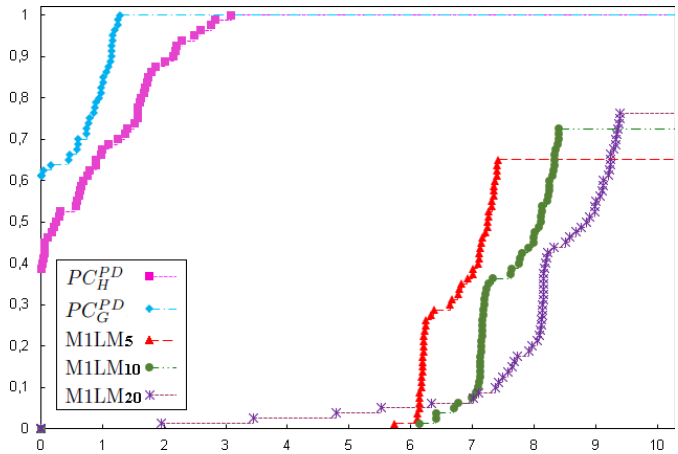


Figura 6: Perfil de desempenho do tempo (Conjunto 3).

5 CONSIDERAÇÕES FINAIS

Nesse artigo propomos um algoritmo de programação dinâmica para resolver o problema da mochila restrita e heurísticas para resolver o problema de corte bidimensional guilhotinado 2-estágios restrito. Os resultados computacionais para as instâncias de pequeno porte do $PCBG-2est$ (Conjunto 1 e 2) mostraram a robustez das heurísticas (PC_H^{PD} e PC_G^{PD} que resolveram, respectivamente, em torno de 80% e 92% das instâncias na otimalidade). A estratégia M1L1 resolveu todas as instâncias do conjunto 2 na otimalidade com tempo computacional similar ao das heurísticas (diferença máxima de 2,55 segundos) mostrando a superioridade nesse conjunto de instâncias. Para as instâncias de grande porte (Conjunto 3) a estratégia M1L1 obteve as melhores soluções em apenas 7,5% das instâncias considerando 20 minutos de execução e não encontrou solução factível em 22,5% das instâncias. As heurísticas encontraram solução factível para todas as instâncias e obtiveram as melhores soluções em torno de 90% delas. O tempo computacional das heurísticas foram em média de 11 segundos mostrando a superioridade em relação à estratégia M1L1 para instâncias de maior porte. Como pesquisa futura é interessante implementar no Algoritmo 1 a geração de faixas verticais (na versão atual somente faixas horizontais são geradas) e considerar também a geração de padrões de corte combinando faixas verticais e/ou faixas horizontais. Outra proposta é adicionar a rotação dos itens no Algoritmo 1. A qualidade das soluções obtidas com a Heurística PC_H^{PD}, em particular para as instâncias de grande porte, estimulam a investigação futura da sua eficiência para resolver o subproblema *pricing* do método de geração de colunas para o Problema de Corte de Estoque Mono-objetivo (*e.g.* [19, 31]) e Multiobjetivo (*e.g.* [4, 5]).

Agradecimentos

Esta pesquisa foi parcialmente financiada pela CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior e pela FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo (Proc.2016/01860-1, 2013/07375-0). Agradecemos à Dra. Jennifer C. Borges pela gentileza de ceder o código em Julia do modelo de otimização matemática usado na estratégia M1LM do estudo computacional. Agradecemos às pessoas que fizeram a revisão *ad hoc* por suas sugestões e comentários.

ABSTRACT. Cutting and packing problems are part of the production planning process in many industries (e.g. paper, glass, furniture). In some furniture industries, large rectangular objects have to be cut into smaller rectangles and there is a limited storage space for work in process . In this case there is interest in solving the constrained two-dimensional two-stages guillotine cutting problem (PCBG-2est). Several authors applied dynamic programming algorithms for solving the unconstrained two-dimensional cutting problem. However, for the constrained case this technique still presents some challenges due to the size of the state space. We propose a heuristic based on the two-step method of Gilmore and Gomory for the constrained PCBG-2est considering special constraints associated with the cutting equipment. The results of a computational study with three sets of instances show the effi-

ciency of the proposal. In particular, for instances that are similar to the furniture industry, solutions were obtained with an average maximum gap of 4.4%.

Keywords: Two-dimensional two-stages guillotine cutting problem, constrained problem, dynamic programming, heuristic.

REFERÊNCIAS

[1] R. Andonov, V. Poirriez & S. Rajopadhye. Unbounded knapsack problem: Dynamic programming revisited. *European Journal of Operational Research*, **123**(2) (2000), 394–407.

[2] M. Arenales, R. Morabito, V. Armentano & H. Yanasse. "Pesquisa operacional: para cursos de engenharia". Elsevier Brasil (2015).

[3] N.S. Assis. "O problema de corte de estoque bidimensional: geração de padrões de corte 2-estágios restritos". Master's thesis, Universidade Estadual Paulista (2019).

[4] J. Borges. "Um Estudo sobre Métodos de Solução para o Problema de Corte de Estoque Biobjetivo". Ph.D. thesis, UNESP, São Jose do Rio Preto - SP (2021).

[5] J.C. Borges, S. Rangel & H. de O. Florentino. A study of column generation embedded in scalarization methods for the biobjective cutting stock problem. In "preparation" (2022).

[6] P.B. Castellucci. JULIA E JuMP: NOVAS FERRAMENTAS PARA PROGRAMAÇÃO MATEMÁTICA. *Pesquisa Operacional para o Desenvolvimento*, **9**(2) (2017), 48–61.

[7] A.C. Cherri & M.N. Arenales. O Problema de Corte de Estoque com Reaproveitamento das Sobras de Material–Heurística FFD Modificada. In "Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional, Gramado" (2005), p. 1700–1711.

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

[9] N. Christofides & C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, **25**(1) (1977), 30–44.

[10] L.L.S. Costa. "Um estudo sobre o problema de corte de estoque bidimensional 2-estágios". Master's thesis, IMECC - UNICAMP, Campinas - SP (2016).

[11] E.D. Dolan & J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, **91**(2) (2002), 201–213.

[12] D. Fayard, M. Hifi & V. Zissimopoulos. An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *Journal of the Operational Research Society*, **49**(12) (1998), 1270–1277.

[13] P.C. Gilmore & R.E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations research*, **13**(1) (1965), 94–120.

[14] P.C. Gilmore & R.E. Gomory. The theory and computation of knapsack functions. *Operations Research*, **14**(6) (1966), 1045–1074.

[15] J.C. Herz. Recursive computational procedure for two-dimensional stock cutting. *IBM Journal of Research and Development*, **16**(5) (1972), 462–469.

[16] E. Horowitz & S. Sahni. Computing Partitions with Applications to the Knapsack Problem. *J. ACM*, **21**(2) (1974), 277–292.

[17] ILOG. "CPLEX Reference Manual. url: https://www.ibm.com/br-pt/products/ilog-cplex-optimization-studio. Acesso em: 22 de julho de 2021". IBM, 12.1 ed. (2009).

[18] A. Lodi & M. Monaci. Integer linear programming models for 2-staged two-dimensional Knapsack problems. *Mathematical Programming*, **94**(2) (2003), 257–278.

[19] R. Macedo, C. Alves & J. Valério de Carvalho. Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research*, **37**(6) (2010), 991–1001.

[20] M. Martin, E. Birgin, R. Lobato, R. Morabito & P. Munari. Models for the two-dimensional rectangular single large placement problem with guillotine cuts and constrained pattern. *International Transactions in Operational Research*, **27** (2019), 767–793.

[21] M. Martin, R. Morabito & P. Munari. A bottom-up packing approach for modeling the constrained two-dimensional guillotine placement problem. *Computers & Operations Research*, **115** (2020), 104851.

[22] P.A. Munari-Junior. Comparação de softwares científicos utilizando perfis de desempenho: automatização dos cálculos pela planilha perfis. *Repositório Institucional ICMC-USP*, (2009).

[23] J.F. Oliveira & J.S. Ferreira. An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research*, **44**(2) (1990), 256–266.

[24] OR-Library. Copyright (c) (2010) (J E Beasley). url: http://people.brunel.ac.uk/ mastjjb/jeb/info.html. Acesso em: 4 de novembro de 2019 (1990).

[25] C. Perin & S. Rangel. O problema do corte bidimensional. In "Anais do Congresso Nacional de Matemática Aplicada e Computacional". São José do Rio Preto - SP (1989), p. 131–134.

[26] D. Pisinger. A Minimal Algorithm for the Bounded Knapsack Problem. *INFORMS Journal on Computing*, **12**(1) (2000), 75–82.

[27] D. Pisinger & P. Toth. Knapsack Problems. *in Handbook of Combinatorial Optimization*, **1** (1998), 299–428.

[28] S. Rangel. "O problema do corte bidimensional". Master's thesis, Universidade Estadual de Campinas (1990).

[29] M. Russo, M. Boccia, A. Sforza & C. Sterle. Constrained two-dimensional guillotine cutting problem: upper-bound review and categorization. *International Transactions in Operational Research*, **27**(2) (2020), 794–834.

[30] G. Scheithauer. "Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods", volume 263. Springer (2018).

[31] E. Silva, F. Alvelos & J. Valério de Carvalho. An integer programming model for two- and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research*, **205**(3) (2010), 699–708.

[32] E. Silva, J.F. Oliveira & G. Wäscher. 2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research*, **237**(3) (2014), 846–856.

[33] R.J. Silveira & R. Morabito. Um método heurístico baseado em programação dinâmica para o problema de corte bidimensional guilhotinado restrito. *Gestão & Produção*, **9** (2002), 78–92.

[34] F.J. Vasko. A computational improvement to Wang's two-dimensional cutting stock algorithm. *Computers & Industrial Engineering*, **16**(1) (1989), 109–115.

[35] A.S. Velasco & E. Uchoa. Improved state space relaxation for constrained two-dimensional guillotine cutting problems. *European Journal of Operational Research*, **272**(1) (2019), 106–120.

[36] P.Y. Wang. Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, **31**(3) (1983), 573–586.

[37] G. Wäscher, H. Haußner & H. Schumann. An improved typology of cutting and packing problems. *European journal of operational research*, **183**(3) (2007), 1109–1130.

[38] H.H. Yanasse & R. Morabito. Modelos lineares e não lineares inteiros para problemas da mochila bidimensional restrita a 2 estágios. *Production [online]*, **23**(4) (2013), 887–896.