Pesquisa Operacional para o Desenvolvimento
Vol. 20, e20004, p. 1-21, 2026

ISSN 1984-3534

PROBLEMAS DE CORTE BI- E TRIDIMENSIONAL GUILHOTINADOS K-ESTÁGIOS E RESTRITOS: UMA EXTENSÃO DO ALGORITMO DE WANG[1]

Vitor Ferrari Henriques[a], Mateus Martin[a] *

[a] Departamento de Engenharia de Produção, Universidade Federal de São Carlos (UFSCar), Rodovia João Leme dos Santos km 110, Sorocaba-SP, Brasil

Recebido 06/02/2026, aceito 21/07/2026

RESUMO

Este trabalho aborda o problema de corte guilhotinado k-estágios e restrito, em suas variantes bidimensional e tridimensional, relevante em indústrias como vidro, madeira e aço. O objetivo consiste em maximizar o valor total (área ou volume) dos itens obtidos a partir do corte de um único objeto, sujeito a cortes guilhotinados ortogonais limitados a, no máximo, k-estágios e a limites superiores de produção por tipo de item. Propõe-se um método exato baseado na extensão do algoritmo *bottom-up* de Wang (1983), inicialmente proposto para o problema bidimensional não-estagiado. A abordagem proposta controla explicitamente o número de estágios guilhotinados via *strings* e incorpora uma regra de poda específica. Experimentos computacionais com instâncias da literatura avaliam o desempenho do algoritmo em termos de qualidade das soluções e tempo computacional, além de permitir a análise do compromisso entre o número de estágios de corte e a utilização do objeto, fornecendo subsídios à tomada de decisão em contextos industriais.

Palavras-chave: Corte e empacotamento, Corte bidimensional e tridimensional, Corte guilhotinado, Algoritmo exato.

ABSTRACT

This work addresses the constrained k-stage guillotine cutting problem in its two- and three-dimensional variants, which are relevant to industries such as glass, wood, and steel. The objective is to maximize the total value (area or volume) of the items obtained from a single object, subject to orthogonal guillotine cuts limited to at most k stages and upper bounds on the production of each item type. An exact method is proposed based on an extension of Wang (1983)'s bottom-up algorithm, originally developed for the non-staged bidimensional problem. The proposed approach explicitly controls the number of guillotine stages via strings and incorporates a specific pruning rule. Computational experiments using benchmark instances from the literature evaluate the algorithm's performance regarding solution quality and computational processing time. Furthermore, the study analyzes the trade-off between the number of cutting stages and material utilization, providing useful insights for decision-making in industrial contexts.

Keywords: Cutting and packing, Two-dimensional and three-dimensional cutting, Guillotine cut, Exact algorithm.

* Autor para correspondência. E-mail: mpmartin@ufscar.br
DOI: https://doi.org/10.59254/PODes.2026.004

[1] *Todos os autores assumem a responsabilidade pelo conteúdo do artigo.*

1. Introdução

Diversos sistemas de manufatura incorporam operações de corte em alguma etapa do processo produtivo, nas quais os itens demandados são obtidos a partir do corte de objetos maiores disponíveis em estoque. Tais operações estão presentes em uma ampla variedade de aplicações industriais, incluindo o corte de barras de aço (Eilon, 1960; Armbruster, 2002), bobinas de papel (Gilmore e Gomory, 1963; Matsumoto et al., 2011), placas de madeira (Morabito e Arenales, 2000; Alem e Morabito, 2012), vidros planos (Dyson e Gregory, 1974; Durak e Aksu, 2017), chapas metálicas (Vasko e Bartkowski, 2009), materiais têxteis (Hifi, 1997; Leao et al., 2020), espumas para colchões e blocos de aço (Martin et al., 2021; Baazaoui et al., 2023), entre outros. A determinação de como os objetos estocados devem ser processados é uma atividade de planejamento formalmente representada pelos problemas de corte e empacotamento (Oliveira et al., 2023), cujo objetivo consiste usualmente em encontrar padrões de corte (isto é, arranjar geometricamente os itens em um ou mais objetos) enquanto se minimiza o desperdício ou se maximiza o valor total obtido (Wäscher et al., 2007).

Nesse contexto, o problema de corte bidimensional guilhotinado e restrito (PC2GR-k) e sua extensão tridimensional (PC3GR) destacam-se como variantes em que os itens devem ser manufaturados respeitando a restrição tecnológica de cortes guilhotinados ortogonais (Scheithauer, 2018). Por restrito denota-se a imposição de limites superiores para a quantidade produzida de cada item, visando evitar superprodução. A sequência de cortes pode ser organizada em estágios guilhotinados, em que cada estágio compreende um conjunto de cortes paralelos entre si, iniciando-se um novo estágio sempre que a direção de corte é alterada em 90 graus. A eficiência de um padrão é frequentemente medida pela taxa de utilização do objeto, definida como a razão entre a soma dos valores (ou áreas/volumes) dos itens cortados e o produto das dimensões do objeto original. Observa-se que essa taxa tende a aumentar à medida que o número de estágios guilhotinados k cresce, em função da maior flexibilidade proporcionada pelo uso de padrões de corte mais complexos. No entanto, esse ganho ocorre ao custo de maior complexidade na operação de corte (i.e., com impacto negativo na produtividade da máquina de corte), evidenciando um *trade-off* entre o número de estágios guilhotinados e a eficiência de utilização do material. Enquanto padrões não-estagiados ($k \to \infty$) podem oferecer o menor desperdício teórico, eles frequentemente tornam-se inviáveis ou excessivamente lentos para o maquinário industrial. Assim, em geral, busca-se maximizar a utilização do material e minimizar o número de estágios guilhotinados k para elevar a produtividade da máquina, medida em metros quadrados ou cúbicos cortados por unidade de tempo.

A importância de se estudar o problema com k estágios, em contraposição à versão não-estagiada, reside justamente na necessidade de fornecer aos tomadores de decisão a flexibilidade de ajustar esse parâmetro conforme as limitações operacionais de cada planta produtiva. Apesar da relevância desse equilíbrio, a literatura é extensa em métodos para casos não-estagiados ou para as variantes fixas de dois e três estágios (Lodi et al., 2002; Silva et al., 2010; Puchinger e Raidl, 2007; Becker et al., 2022), persistindo uma lacuna no desenvolvimento de abordagens que tratem padrões com até k estágios, sendo k um parâmetro de entrada passível de ser variado (Martin et al., 2020). Formulações baseadas em níveis (tiras ou prateleiras) ou abordagens de programação dinâmica para o caso não-estagiado não são diretamente adaptáveis quando se deseja impor um limite superior genérico e fixo para os estágios.

Neste trabalho, abordam-se o problema de corte bidimensional guilhotinado k-estágios e restrito (PC2GR-k) e sua extensão tridimensional (PC3GR-k). Busca-se selecionar e posicionar itens (retângulos ou cuboides) em um único objeto, respeitando simultaneamente as restrições de corte guilhotinado com no máximo k estágios (restrição tecnológica) e os limites de produção (restrição de produção). As principais contribuições deste estudo são:

- Proposição de um algoritmo combinatório *bottom-up* para os problemas PC2GR-k e PC3GR-k, estendendo o método de Wang (1983), que foi originalmente concebido para o caso não-estagiado (PC2GR-∞);

- Introdução de uma representação de soluções parciais que incorpora a sequência de estágios guilhotinados via *strings*, permitindo o controle direto do limite k de estágios guilhotinados;
- Desenvolvimento de um critério de dominância e estratégia de poda que explora explicitamente o limite k de estágios para mitigar a explosão do espaço de busca.

A relevância deste estudo é evidenciada tanto pelo viés científico, onde o PC2GR-k e o PC3GR-k surgem como subproblemas em modelos de corte de estoque (Barreto et al., 2024), quanto pelo viés aplicado em setores de madeira e vidro (Kozyreff Filho e Araújo, 2024; Muzulon et al., 2024). Experimentos computacionais com instâncias da literatura avaliam a qualidade das soluções e o tempo de processamento, analisando o impacto de k na taxa de utilização. Ao melhor conhecimento dos autores, não há na literatura abordagens que tratem diretamente problemas de corte restritos com variação explícita do número k de estágios guilhotinados.

O restante deste texto está organizado da seguinte forma. A Seção 2 apresenta uma revisão da literatura sobre o PC2GR-k, o PC3GR-k e problemas correlatos. A Seção 3 descreve formalmente os problemas estudados. A Seção 4 revisa o algoritmo de Wang (1983) e apresenta a adaptação proposta. A Seção 5 reporta os experimentos computacionais e discute o compromisso entre a utilização do material e o número de estágios. Por fim, a Seção 6 apresenta as conclusões e direções para trabalhos futuros.

2. Revisão da Literatura

As estratégias de resolução do problema de corte bidimensional guilhotinado fundamentam-se em duas vertentes principais: a abordagem *top-down*, proposta inicialmente por Christofides e Whitlock (1977), que subdivide recursivamente a chapa original em retângulos menores; e, a abordagem *bottom-up*, introduzida por Wang (1983), que constrói padrões a partir da combinação sucessiva de itens. Os principais trabalhos têm sido revisados e categorizados em *surveys*, com destaque para Russo et al. (2020), além das revisões abrangentes de Lodi et al. (2002), Wäscher et al. (2007), Iori et al. (2021) e Oliveira et al. (2023). A literatura indica que esses problemas podem ser resolvidos por uma ampla variedade de métodos, incluindo algoritmos baseados em programação dinâmica (Hadjiconstantinou e Christofides, 1995; Velasco e Uchoa, 2019), algoritmos de busca em árvore (Viswanathan e Bagchi, 1993; Cung et al., 2000; Yoon et al., 2013), heurísticas e meta-heurísticas (Daza et al., 1995; Alvarez-Valdés et al., 2002), além de modelos de programação linear inteira mista (PLIM) (Lodi e Monaci, 2003; Puchinger e Raidl, 2007; Silva et al., 2010; Macedo et al., 2010). A seguir, por limitação de escopo, esta revisão concentra-se em trabalhos que utilizaram o princípio *bottom-up*.

O trabalho pioneiro de Wang (1983) estabeleceu o princípio *bottom-up* para a geração de padrões de corte por meio de combinações (construções ou *builds*) horizontais e verticais de retângulos (ou soluções parciais), utilizando um parâmetro de desperdício para controlar a explosão do número de combinações exploradas. Evoluções subsequentes buscaram melhorar o desempenho computacional sem comprometer a qualidade das soluções. Por exemplo, Vasko (1989) introduziu os conceitos de completude horizontal e vertical, com o objetivo de eliminar precocemente soluções parciais que excederiam as dimensões da chapa. Paralelamente, Oliveira e Ferreira (1990) propuseram uma adaptação no nível de aspiração, empregando uma formulação de programação dinâmica para resolver a versão irrestrita do problema e obter limites inferiores para o desperdício, o que permite rejeitar heurística e antecipadamente soluções parciais inviáveis.

Viswanathan e Bagchi (1993) desenvolveram um algoritmo de busca em árvore do tipo *best-first*, que utiliza uma função de avaliação $f'(R) = g(R) + h'(R)$, em que $g(R)$ representa o lucro interno do padrão e $h'(R)$ é um limite superior para o valor que pode ser obtido na área não ocupada P. Esse limite é calculado por meio da função de mochila bidimensional de Gilmore e Gomory (1965), permitindo que a busca priorize, sem perda de generalidade, os nós com maior potencial de lucro. Essa linha de pesquisa foi refinada por Cung et al. (2000), que introduziram o

uso de códigos de padrão para identificar e descartar padrões simétricos ou duplicados em tempo constante. Além disso, os autores implementaram estratégias de paralelização do algoritmo. Posteriormente, De Armas et al. (2012) ampliaram essa eficiência ao propor regras de dominância pós-geração. Por exemplo, uma das regras filtra a lista de soluções parciais em busca de padrões com o mesmo conjunto de peças, porém com dimensões menores, enquanto outra verifica se já existem padrões de mesmo tamanho com conjuntos de peças dominantes, economizando esforço computacional em combinações redundantes. De forma complementar, Yoon et al. (2013) introduziram estratégias de ramificação que estabelecem limites de largura e altura para as combinações, evitando a formação de padrões não promissores. Ao aprimorarem os limites superiores ao considerar a restrição do número de peças restantes, os autores conseguiram reduzir significativamente o tempo de processamento.

Mais recentemente, os trabalhos buscaram superar as limitações de memória típicas da abordagem *bottom-up* pura por meio de hibridismos estruturais. Wei e Lim (2015) introduziram uma abordagem bidirecional, na qual as peças são agrupadas em blocos via *bottom-up*, enquanto o processo de disposição ocorre de forma *top-down*, dividindo sucessivamente a chapa em espaços menores. Essa técnica comprime a altura da árvore de busca, o que reduz drasticamente a necessidade de armazenar uma quantidade exponencial de padrões parciais na memória. Considera-se que o estado da arte é representado pelo algoritmo EATKG, proposto por Wang et al. (2025), o qual emprega um pré-processamento avançado para reduzir o número de tipos e cópias de itens em cerca de 20% por meio de relações de dominância e redimensionamento da chapa. O algoritmo incorpora uma técnica de enumeração de combinação iterativa, acionada quando há risco de estouro de memória: nessa fase, o problema é decomposto em dois níveis, resolvendo-se inicialmente a seleção do subconjunto e, em seguida, verificando-se a viabilidade do padrão. Essa robustez permitiu ao algoritmo resolver 93% de um conjunto de 1277 instâncias, encontrando 46 novas soluções ótimas e limites superiores mais apertados para problemas de grande escala que eram anteriormente considerados intratáveis.

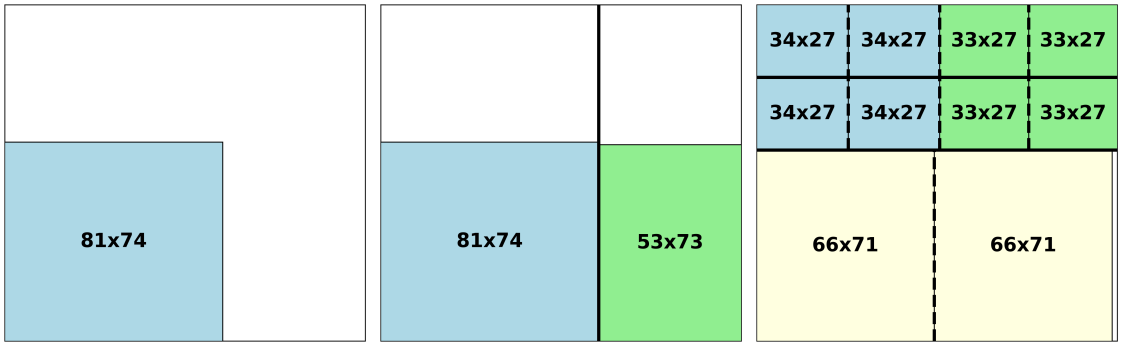
A complexidade do problema aumenta ao se transitar para a versão tridimensional; no entanto, apesar da potencial aplicabilidade do PC3GR-k, a literatura do problema é mais escassa. Queiroz et al. (2012) avançaram nesse campo ao propor algoritmos de programação dinâmica para a versão irrestrita do PC3GR-k (isto é, quando não se limita o número de cópias por tipo de item), introduzindo o uso de *reduced raster points* para otimizar a busca por padrões de corte exatos. Martin et al. (2021) desenvolveram modelos de PLIM e um algoritmo *bottom-up* específico para o caso tridimensional, abordando tanto padrões não-estagiados quanto padrões de 3-estágios. Complementando essas abordagens, Do Nascimento et al. (2022) introduziram uma técnica iterativa de dois níveis que combina PLIM com programação por restrições, garantindo que as restrições de não-sobreposição e a sequência de estágios guilhotinados sejam respeitadas com eficiência em instâncias de complexidade moderada.

3. Descrição do Problema

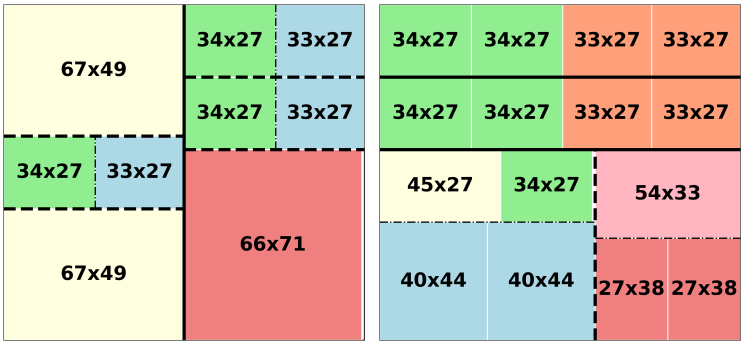
Formalmente, o PC2GR-k consiste em selecionar e posicionar um subconjunto de itens em um único objeto retangular de dimensões $L \times W$. Para cada tipo de item i pertencente ao conjunto $I = \{1, \ldots, m\}$, definem-se as dimensões $l_i \times w_i$, um limite superior de cópias u_i para evitar a superprodução e um valor $v_i = l_i \cdot w_i$ (área). Assume-se, portanto, a versão não-ponderada do problema, visto que a versão ponderada permitiria que o valor v_i fosse arbitrário. A operação de corte é regida por duas condições: (i) a restrição tecnológica de cortes guilhotinados ortogonais realizados em, no máximo, k estágios; e (ii) a restrição de produção, que impõe limites às quantidades produzidas de cada tipo de item a fim de evitar superprodução. Relembre que: um corte guilhotinado ortogonal é definido como aquele realizado de borda a borda, dividindo um retângulo em dois menores; e, a sequência de cortes é organizada em estágios, nos quais cada estágio compreende um conjunto de cortes paralelos entre si, sendo que um novo estágio é iniciado

sempre que a direção do corte é rotacionada em 90 graus em relação à anterior. A função objetivo do PC2GR-k consiste em maximizar a soma do valor dos itens selecionados para corte. Assume-se que o corte de aparas, isto é, que separa um item e desperdício, não é um estágio guilhotinado adicional. A variante do PC2GR-k no qual não se limita o número de estágios guilhotinados é denotada na literatura como problema não-estagiado (PC2GR-∞). Rotações de itens não são permitidas; assim, as dimensões L e l_i referem-se ao eixo x (abscissas), enquanto W e w_i referem-se ao eixo y (ordenadas).

A Figura 1 ilustra cinco soluções ótimas para o PC2GR-k da instância da literatura CU03, obtidas pelo algoritmo proposto (Algoritmo 2), ao variar o número de estágios guilhotinados de zero a quatro. Os retângulos coloridos representam os itens produzidos, enquanto as áreas em branco indicam o desperdício de material. No contexto da perspectiva *top-down*, os estágios de corte são diferenciados pela simbologia das linhas: linhas contínuas pretas indicam o primeiro estágio; linhas tracejadas pretas, o segundo; linhas traço-ponto pretas, o terceiro; e linhas contínuas brancas, o quarto estágio. Note que uma solução para o PC2GR-0, como a ilustrada na Figura 1a, corresponde a uma única cópia de um tipo de item, enquanto, para o PC2GR-1, como na Figura 1b, uma solução corresponde a uma única tira horizontal ou vertical. As soluções para o PC2GR-2, como a ilustrada na Figura 1c, consistem em um conjunto de tiras horizontais ou verticais obtidas por cortes de primeiro estágio, nas quais os itens são separados entre si por cortes de segundo estágio. As soluções para o PC2GR-k, com $k = 3$ e $k = 4$, correspondem a padrões de corte mais complexos, conforme ilustrado nas Figuras 1d e 1e. Por outro lado, na perspectiva *bottom-up*, tem-se que: a solução da Figura 1b corresponde a uma única construção (ou *build*) no eixo x; e, cada tira horizontal da solução da Figura 1c foi gerada via construções no eixo x, e depois essas três tiras horizontais são agregadas via construções no eixo y. Observe que, para cada instância, existe um número máximo de estágios k a partir do qual incrementos adicionais não resultam em aumento da taxa de utilização. Em particular, para a instância CU03, a solução apresentada na

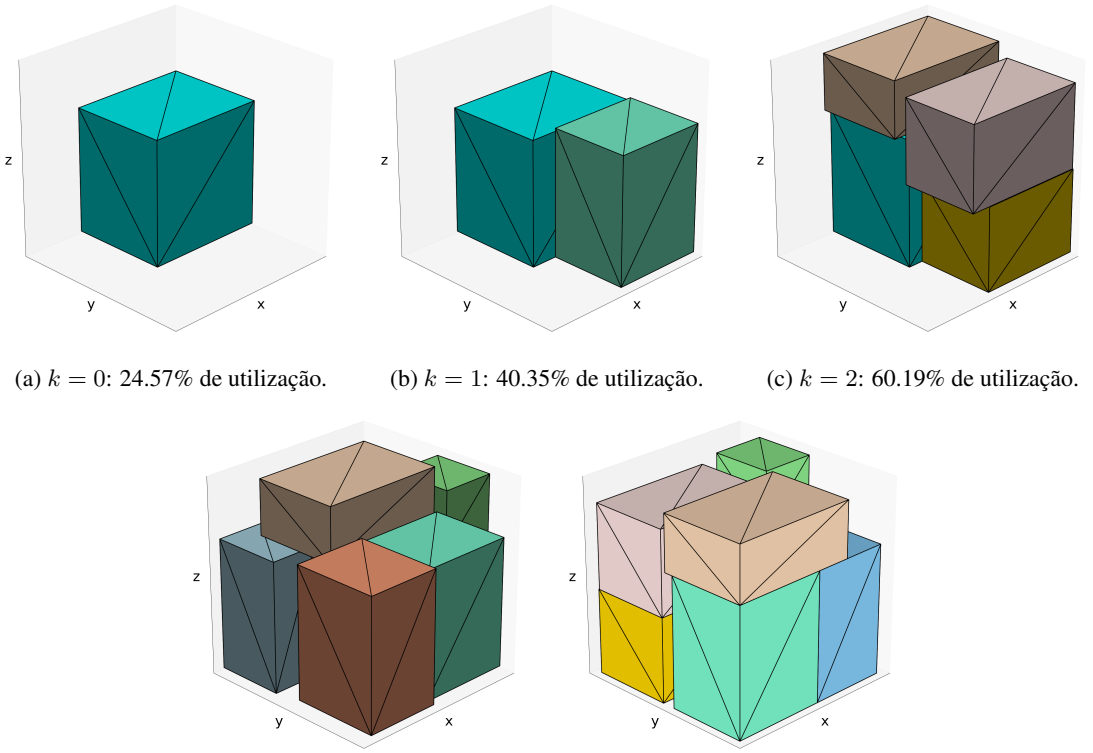


(a) $k = 0$: 35.79% de utilização. (b) $k = 1$: 58.88% de utilização. (c) $k = 2$: 99.15% de utilização.



(d) $k = 3$: 99.58% de utilização. (e) $k = 4$: 99.84% de utilização.

Figura 1: Ilustração de soluções ótimas para o PC2GR-k da instância CU03 ($L \times W = 134 \times 125$ e $m = 45$), considerando de zero a quatro estágios guilhotinados.
Fonte: Elaboração própria.



(a) $k = 0$: 24.57% de utilização. (b) $k = 1$: 40.35% de utilização. (c) $k = 2$: 60.19% de utilização.

(d) $k = 3$: 71.25% de utilização. (e) $k = 4$: 76.28% de utilização.

Figura 2: Ilustração de soluções ótimas para o PC3GR-k da instância gcut06_3d_cons ($L \times W \times H = 500 \times 500 \times 500$ e $m = 20$), considerando de zero a quatro estágios guilhotinados.
Fonte: Elaboração própria.

Figura 1e, com $k = 4$, corresponde à solução ótima do PC2GR-∞, isto é, permitir valores de k superiores a quatro não conduz a melhorias adicionais na taxa de utilização.

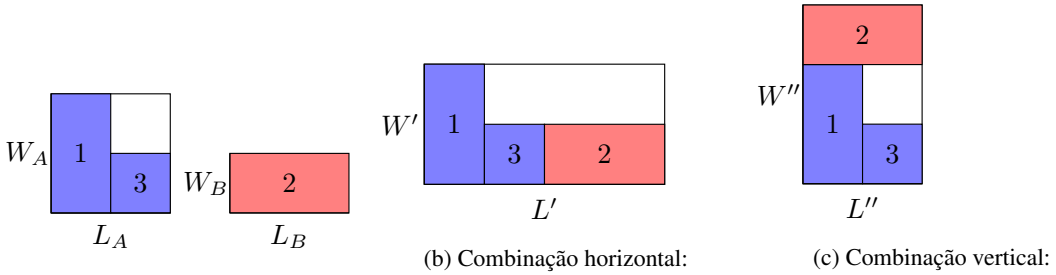
A extensão do PC2GR-k para o corte de um objeto tridimensional de dimensões $L \times W \times H$, visando à obtenção de itens com dimensões $l_i \times w_i \times h_i$ e valor $v_i = l_i \cdot w_i \cdot h_i$ (volume), caracteriza o PC3GR-k. Assume-se que as dimensões H e h_i referem-se ao eixo z (cota). De acordo com a tipologia de Wäscher et al. (2007) para problemas de corte e empacotamento, o PC2GR-k e o PC3GR-k são classificados como problemas de alocação em um único objeto grande (*single large object placement problem*) ou de mochila única (*single knapsack problem*), nos quais as limitações de corte guilhotinado ortogonal em k estágios e de produção restrita configuram restrições adicionais. De forma análoga à Figura 1, a Figura 2 ilustra cinco soluções ótimas para o PC3GR-k da instância da literatura gcut06_3d_cons, obtidas pelo algoritmo proposto (Algoritmo 2), ao variar o número de estágios guilhotinados de zero a quatro. Observe-se que cada corte guilhotinado ortogonal no PC3GR-k divide um cuboide (isto é, um paralelepípedo retangular) em dois cuboides menores e que a solução ótima da Figura 2e, com $k = 4$, corresponde à solução ótima do PC3GR-∞ para essa instância. Para o PC3GR-k, na perspectiva *bottom-up*, assume-se que as construções nos eixos x, y e z correspondem, respectivamente, a: um corte vertical paralelo ao plano Oyz; um corte em profundidade paralelo ao plano Oxz; e um corte horizontal paralelo ao plano Oxy.

4. Algoritmos *Bottom-up*

A seguir, na Seção 4.1, revisa-se o algoritmo *bottom-up* de Wang (1983) proposto para o PC2GR-∞. Em seguida, na Seção 4.2, descreve-se o algoritmo *bottom-up* proposto para o PC2GR-k e o PC3GR-k.

4.1. Algoritmo de Wang (1983) para o PC2GR-∞

Wang (1983) propôs um algoritmo construtivo *bottom-up* que avalia soluções parciais e as combina sucessivamente para gerar padrões associados a retângulos de dimensões crescentes. Uma solução parcial é representada por uma 4-tupla $(l, w, v, \mathbf{a})$, em que $l \times w$ é a dimensão do padrão, $\mathbf{a} = (a_1, \ldots, a_m)^\top \in \mathbb{Z}_+^m$ indica o número de cópias de cada tipo de item $i \in I$, e $v = \sum_{i \in I} v_i a_i$ é o valor da solução (i.e., a área total dos itens cortados). Dadas duas soluções parciais factíveis (l, w, v, a) e (l', w', v', a'): a combinação horizontal gera $(l+l', \max\{w, w'\}, v+v', \mathbf{a}+\mathbf{a}')$; e, a combinação vertical gera: $(\max\{l, l'\}, w+w', v+v', \mathbf{a}+\mathbf{a}')$. Novos padrões são gerados por combinações horizontais e verticais de soluções parciais, conforme ilustrado na Figura 3. Note que a perda de corte P associada a uma solução parcial, quando posicionada no (canto inferior-esquerdo do) objeto $L \times W$, é calculada por $P = LW - \sum_{i \in I} v_i a_i$.



(a) Soluções parciais A e B.

(b) Combinação horizontal: $L' = L_A + L_B$ e $W' = \max\{W_A, W_B\}$.

(c) Combinação vertical: $L'' = \max\{L_A, L_B\}$ e $W'' = W_A + W_B$.

Figura 3: Ilustração da combinação horizontal (eixo x) e vertical (eixo y) de duas soluções parciais A e B.
Fonte: Elaboração própria.

No PC2GR-∞, a solução parcial resultante de uma combinação é factível se suas dimensões não excederem $L \times W$ e se $\mathbf{a} + \mathbf{a}' \leq \mathbf{u}$. A notação utilizada é apresentada na Tabela 1, e o pseudocódigo correspondente para a resolução do PC2GR-∞ é apresentado no Algoritmo 1. Seja $F^{(k)}$ o conjunto de 4-tuplas factíveis geradas na iteração k. A inicialização do algoritmo considera padrões com uma única cópia de item, isto é, $e_i \in F^{(0)}$ para todo $i \in I$, onde e_i é o i-ésimo vetor unitário de $\mathbb{Z}_+^m$ (linha 2 do algoritmo). Para cada iteração $k \geq 1$, o algoritmo combina as soluções parciais geradas na iteração anterior ($F^{(k-1)}$) com todas as soluções parciais geradas até então ($L^{(k-1)}$) (laço 5–11 do algoritmo).

Tabela 1: Notação matemática do algoritmo *bottom-up*.

$0 \leq \beta \leq 1$	porcentagem máxima de perda aceitável em uma solução parcial
T	solução parcial representada por uma 4-tupla $(l, w, v, \mathbf{a})$
$F^{(k)}$	conjunto de todas as 4-tuplas $(l, w, v, \mathbf{a})$ factíveis geradas na iteração k
$L^{(k)}$	conjunto de todas as 4-tuplas $(l, w, v, \mathbf{a})$ factíveis geradas até a iteração k

Para controlar o crescimento explosivo do número de soluções parciais, Wang propôs duas variantes que diferem no critério de aspiração: uma limita a perda interna como fração da área do objeto, e a outra como fração da área da própria solução parcial (usada na linha 10 do algoritmo). Em ambos os casos, são estabelecidas condições que permitem certificar otimalidade ou avaliar a proximidade da solução obtida em relação à ótima. Assim, o parâmetro β controla o compromisso entre qualidade da solução e esforço computacional: valores maiores ampliam o espaço de busca, favorecendo padrões com maior ocupação ao custo de tempos de processamento mais elevados; e, valores menores produzem o efeito oposto. O Teorema 1 apresenta a condição de otimalidade de

Algoritmo 1: Algoritmo *bottom-up* de Wang (1983) para o PC2GR-∞.

Entrada: Dimensões do objeto $L \times W$, dimensões dos itens $l_i \times w_i$ e limites superiores $u_i,\ \forall i \in I$.

1 Escolher $\beta \in [0, 1]$
2 $F^{(0)} \leftarrow \{T_i : i \in I\}$
3 $L^{(0)} \leftarrow F^{(0)}$
4 $k \leftarrow 0$
5 **enquanto** $F^{(k)} \neq \varnothing$ **faça**
6 $\quad k \leftarrow k + 1$
7 $\quad$ Gerar $F^{(k)}$ combinando elementos de $F^{(k-1)}$ com elementos de $L^{(k-1)}$ tais que:
8 $\quad\quad$ (i) T é obtido por combinações nos eixos x (horizontal) e y (vertical);
9 $\quad\quad$ (ii) número de cópias a_i de cada tipo de item $i \in I$ não excede u_i;
10 $\quad\quad$ (iii) a perda $P = LW - \sum_{i \in I} v_i a_i$ não excede o liminar βLW.
11 $\quad L^{(k)} \leftarrow L^{(k-1)} \cup F^{(k)}$
12 Selecionar em $L^{(k-1)}$ a solução parcial T com menor perda P

Saída: Solução ótima para o PC2GR-∞.

Wang, a qual é suficiente para garantir a otimalidade da melhor solução T encontrada na busca, gerada com β fixo.

Teorema 1. *Se a perda P da melhor solução T obtida ao final do Algoritmo 1, gerada com β fixo, satisfizer $P \leq \beta LW$, então T é uma solução ótima.*

Demonstração. Seja $\mathcal{O}$ o conjunto de soluções ótimas, com perda mínima P^*. Seja β^* o menor valor de β que permite gerar ao menos uma solução de $\mathcal{O}$. Assuma $P \leq \beta LW$ e suponha, por contradição, que $T \notin \mathcal{O}$. Então $\beta < \beta^*$, o que implica $P^* \geq \beta^* LW > \beta LW \geq P$, contradizendo a definição de P^*. Logo, $T \in \mathcal{O}$. □

Diferentemente dos métodos de *branch-and-bound* (B&B) considerados em Programação Linear Inteira, que utilizam relaxações lineares para derivar limitantes duais e realizar a enumeração implícita, o Algoritmo 1 fundamenta-se na estratégia combinatória *bottom-up*. No B&B, a poda de um subproblema ocorre quando seu limitante dual é inferior à melhor solução conhecida. Por outro lado, um algoritmo *bottom-up* realiza uma enumeração exaustiva das construções por meio da agregação sucessiva de itens e subpadrões. A eliminação de soluções parciais decorre exclusivamente da aplicação de critérios de dominância, como o Teorema 1, e do cumprimento de restrições de viabilidade, sem recorrer a estimativas de potencial de otimalidade. Portanto, por não operar sobre relaxações do problema original, o algoritmo não gera limitantes válidos durante sua execução, o que inviabiliza o cálculo de um *gap* de otimalidade.

4.2. Algoritmo Proposto para o PC2GR-k e PC3GR-k

O algoritmo proposto para o PC2GR-k e o PC3GR-k representa e controla explicitamente a sequência de estágios guilhotinados ao longo do processo de combinações por meio do uso de *strings*. Enquanto o algoritmo original para o PC2GR-∞ constrói soluções parciais sem rastrear tal sequência, a abordagem para k-estágios requer que cada solução parcial carregue informações suficientes para garantir que as combinações futuras respeitem o limite k estabelecido. O pseudocódigo correspondente é apresentado no Algoritmo 2. A principal distinção em relação à literatura encontra-se no passo da linha 11, que exige a atualização da sequência de estágios. Para o PC2GR-k, redefine-se solução parcial para uma 5-tupla $T = (l, w, v, \mathbf{a}, s)$, em que $s \in \{x, y\}^*$ é uma *string* que representa a sequência de orientação dos estágios guilhotinados. De forma análoga,

no PC3GR-k, define-se uma 6-tupla $T = (l, w, h, v, \mathbf{a}, s)$, em que h é a dimensão no eixo z e $s \in \{x, y, z\}^*$. Diferente da literatura associada de problemas de corte guilhotinado, que requer supor a orientação dos cortes de primeiro estágio e reexecutar a instância, o presente algoritmo obtém a solução ótima sob o limite k sem restrições quanto à orientação inicial.

Algoritmo 2: Algoritmo *bottom-up* proposto para o PC2GR-k (resp. PC3GR-k)

Entrada: Número k de estágios guilhotinados, Dimensões do objeto $L \times W$ (resp. $L \times W \times H$), dimensões dos tipos de itens $l_i \times w_i$ (resp. $l_i \times w_i \times h_i$) e limites superiores u_i, $\forall i \in I$.

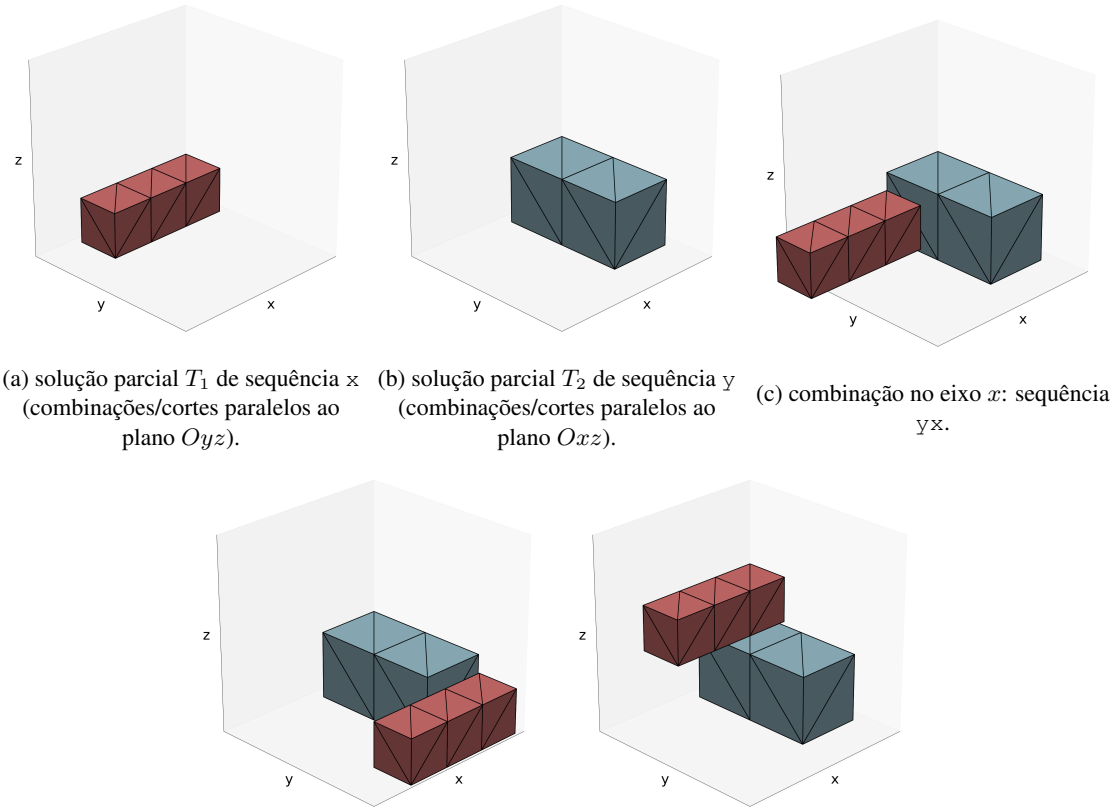
Escolher $\beta \in [0, 1]$
$F^{(0)} \leftarrow \{T_i : i \in I\}$
$L^{(0)} \leftarrow F^{(0)}$
$k \leftarrow 0$
enquanto $F^{(k)} \neq \emptyset$ **faça**
    $k \leftarrow k + 1$
    Gerar $F^{(k)}$ combinando elementos de $F^{(k-1)}$ com elementos de $L^{(k-1)}$ tais que:
        (i) T é obtido por combinações nos eixos x e y (resp. x, y e z);
        (ii) o número de cópias de cada item $i \in I$ não excede u_i;
        (iii) a perda P não excede o limiar βLW (resp. βLWH);
        (iv) o número de estágios guilhotinados não excede o limite k.
    $L^{(k)} \leftarrow L^{(k-1)} \cup F^{(k)}$
Selecionar em $L^{(k-1)}$ a solução com menor perda P

Saída: Solução ótima para o PC2GR-k (resp. PC3GR-k).

Uma *string* s codifica a sequência dos estágios guilhotinados necessários para a obtenção dos itens da solução parcial. Cada caractere representa a orientação de um conjunto de combinações (ou cortes), e o comprimento $|s|$ define o número total de estágios. Sob a perspectiva *bottom-up*, as combinações seguem a ordem direta de s (do primeiro ao último símbolo); sob a perspectiva *top-down*, a leitura é invertida. Assume-se que o símbolo $\varnothing$ representa a *string* vazia (uma única cópia do item) e que a separação entre item e desperdício não constitui um estágio guilhotinado.

Para o PC2GR-k, as sequências correspondentes às Figuras 1a a 1e são, respectivamente, $s = \varnothing$, `x`, `xy`, `xyx` e `xyxy`. Para $s =$ `xyxy`, que ilustra um padrão de 4 estágios (Figura 1e), os cortes associados ao primeiro e terceiro estágios são perpendiculares ao eixo y (corte horizontal ou construção no eixo y), enquanto os do segundo e quarto estágios são perpendiculares ao eixo x (corte vertical ou construção no eixo x).

A Figura 4 ilustra o conceito de sequência de estágios para o PC3GR-k. As soluções parciais T_1 (Figura 4a) e T_2 (Figura 4b) são geradas por combinações nos eixos x e y, respectivamente. A Figura 4c exemplifica a combinação de T_1 e T_2 no eixo x, resultando em uma solução definida pela sequência `yx` (combinação no eixo y seguida pelo eixo x). Nesse contexto, a representação por *strings* vincula as perspectivas *top-down* e *bottom-up*. Enquanto a visão *top-down* modela a operação física da máquina ao particionar o objeto original por planos de corte ortogonais sucessivos (paralelos aos planos Oyz, Oxz ou Oxy), a abordagem *bottom-up* constrói o padrão de forma inversa, agregando subpadrões. A *string* `yx` registra a ordem exata dessa agregação (primeiro no eixo y, depois no x), garantindo que cada união corresponda a um corte viável na perspectiva *top-down*. Essa inversão ótica permite ao algoritmo um controle combinatório direto e eficiente sobre a estrutura, assegurando o cumprimento estrito do limite de estágios k a cada nova agregação.



(a) solução parcial T_1 de sequência `x` (combinações/cortes paralelos ao plano Oyz).

(b) solução parcial T_2 de sequência `y` (combinações/cortes paralelos ao plano Oxz).

(c) combinação no eixo x: sequência `yx`.

(d) combinação no eixo y: sequência `xy`.

(e) combinação no eixo z: sequências possíveis `xyz` ou `yxz`.

Figura 4: Ilustração de sequências de estágios guilhotinados para o PC3GR-k.
Fonte: Elaboração própria.

Determinando a sequência de estágios guilhotinados (passo 11 do Algoritmo 2)

Ao combinar duas soluções T_1 (com sequência s_1) e T_2 (com sequência s_2) ao longo de um eixo α, a nova sequência s não pode ser obtida por simples concatenação. Uma concatenação $s_1 s_2$ ignoraria a possibilidade de ambas as soluções compartilharem estágios iniciais de mesma orientação. Para ilustrar, considere duas soluções parciais com sequências $s_1 = \texttt{xy}$ e $s_2 = \texttt{yx}$, ambas com 2-estágios. Uma concatenação ingênua $s_1 s_2 = \texttt{xyyx}$ ou $s_2 s_1 = \texttt{yxxy}$ resultaria em 4-estágios, desconsiderando a possibilidade de intercalar os cortes de ambas as soluções de forma mais eficiente. Assim, o requisito é que a sequência resultante s preserve a ordem relativa dos cortes tanto em s_1 quanto em s_2. Em termos formais, exige-se que s_1 e s_2 sejam subsequências de s. Esse problema é conhecido na literatura de ciência da computação como o problema da *shortest common supersequence* (SCS) (Vazirani, 2001). Dadas duas *strings* s_1 e s_2 sobre um alfabeto Σ (no caso, $\Sigma = \{x, y\}$ para o PC2GR-k e $\Sigma = \{x, y, z\}$ para o PC3GR-k), uma *string* s é uma super-sequência comum de s_1 e s_2 se ambas são subsequências de s. Uma *string* s^* é uma SCS mínima se $|s^*|$ é mínima entre todas as super-sequências comuns. A complexidade para determinar s varia significativamente entre os dois problemas tratados.

Para o PC2GR-k, a determinação é direta. Ao combinar T_1 e T_2 no eixo $\alpha \in \{x, y\}$, primeiro normalizam-se s_1 e s_2 adicionando α ao final, caso já não terminem com esse caractere. Essa normalização assegura que o último estágio de corte da solução combinada seja consistente com a orientação da combinação. Como o alfabeto possui apenas dois símbolos e os estágios devem alternar orientações para serem minimizados, a SCS mínima entre as sequências s_1 e s_2 normalizadas é única e corresponde simplesmente àquela de maior comprimento. Por exemplo,

combinando no eixo x a sequência $s_1 = \varnothing$ (que, após normalização, torna-se $\mathtt{x}$) com a sequência $s_2 = \mathtt{x}$ (que se mantém inalterada após a normalização), obtém-se uma nova solução parcial de sequência $\mathtt{x}$. Já a combinação no eixo y de s_1 (normalizada para $\mathtt{y}$) com $s_2 = \mathtt{x}$ (normalizada para $\mathtt{xy}$) produz uma nova solução parcial de sequência $\mathtt{xy}$.

No PC3GR-k, a existência de três eixos ($\alpha \in \{x, y, z\}$) torna o problema combinatório, pois poderá haver múltiplas formas de intercalar os estágios. Considere $s_1 = \mathtt{x}$ e $s_2 = \mathtt{y}$, conforme ilustrado nas Figuras 4a e Figuras 4b, respectivamente. Ao combiná-las no eixo z, as sequências normalizadas tornam-se $s_1' = \mathtt{xz}$ e $s_2' = \mathtt{yz}$. As SCS mínimas resultantes são $\mathcal{S} = \{\mathtt{xyz}, \mathtt{yxz}\}$, conforme ilustrado na Figura 4e, isto é, possuem 3-estágios, mas representam processos de corte distintos. O cálculo do conjunto de todas as SCS mínimas, isto é, de $\mathcal{S} = \mathrm{SCS}(s_1, s_2)$ é realizado em duas fases via programação dinâmica (Cormen et al., 2009; Gusfield, 1997; Maier, 1978). Na primeira fase, calcula-se o comprimento mínimo ℓ^* através da relação de recorrência dada pela Equação (1).

$$\mathrm{dp}[i][j] = \begin{cases} |s_2| - j, & \text{se } i = |s_1|, \\ |s_1| - i, & \text{se } j = |s_2|, \\ 1 + \mathrm{dp}[i+1][j+1], & \text{se } s_1[i] = s_2[j], \\ 1 + \min\{\mathrm{dp}[i+1][j], \mathrm{dp}[i][j+1]\}, & \text{caso contrário.} \end{cases} \tag{1}$$

Na segunda fase, um procedimento de *backtracking* com memoização recupera todas as *strings* de comprimento ℓ^*. Para cada $s \in \mathcal{S}$ tal que $|s| \leq k$, gera-se uma nova solução parcial distinta. Essa exploração de todas as SCS mínimas é essencial: diferentes ordenações de eixos, embora geometricamente idênticas no estágio atual, possuem potenciais diferentes para combinações futuras sob o limite k.

Diferente de abordagens que utilizam tabelas de consulta estáticas (Martin et al., 2021), o uso de programação dinâmica para SCS permite que o algoritmo proposto seja generalizável para qualquer número de dimensões e trate de forma exata a restrição para qualquer número k de estágios guilhotinados. Por fim, nota-se que o limite de tempo também foi adotado como critério de parada nos Algoritmos 1 e 2 durante os experimentos computacionais da Seção 5. Quando esse limite é atingido, o certificado de otimalidade garantido pelo Teorema 1 deixa de ser assegurado.

4.3. Estratégias de Melhoria

Com o objetivo de reduzir o número de soluções parciais, Vasko (1989) introduziu o conceito de completude, que identifica padrões que já saturam a capacidade geométrica do objeto em determinada direção. Seja $T = (l, w, v, \mathbf{a})$ uma solução parcial factível. Diz-se que T é horizontalmente completa se não existe solução parcial factível $T' = (l', w', v', \mathbf{a}')$ cuja combinação horizontal com T produza comprimento não superior a L (isto é, $l + l' > L$ para qualquer T' admissível). De modo análogo, T é verticalmente completa se nenhuma combinação vertical factível resulta em largura não superior a W. Nessas situações, T não pode ser estendida naquela orientação. A completude implica que soluções horizontalmente (resp., verticalmente) completas deixam de participar de combinações nessa direção. Além disso, a dimensão correspondente é ajustada ao limite do objeto ($l \leftarrow L$ ou $w \leftarrow W$) e a perda é atualizada. Quando uma solução é completa em ambas as orientações, sua perda fornece um limitante superior válido, permitindo atualizar β e eliminar de $L^{(k)}$ soluções com perda maior que o novo limiar, sem perda de otimalidade. ionais propostas por Vasko (1989) ao algoritmo de Wang (1983). Nota-se que, na implementação computacional da Seção 4, os Algoritmos 1 e 2 incorporam a melhoria de completude proposta por Vasko (1989).

5. Experimentos Computacionais

Nesta seção, reportam-se os resultados dos experimentos computacionais realizados com o objetivo de avaliar o desempenho do Algoritmo 2, proposto para a resolução do PC2GR-k e do PC3GR-k. Para fins de validação e análise de desempenho, adota-se como *benchmark* o algoritmo *bottom-up* original de Wang (1983) (Algoritmo 1), aplicado ao contexto das versões não-estagiadas PC2GR-∞ e PC3GR-∞. Esta escolha justifica-se pelo fato de que, ao melhor conhecimento dos autores, não há na literatura métodos que enderecem diretamente o problema de corte restrito com variação explícita e arbitrária do número de estágios guilhotinados. Assim, a comparação com o caso não-estagiado provê um limite superior para a taxa de utilização, permitindo quantificar o custo de oportunidade operacional associado à redução do parâmetro k em prol da produtividade. Todos os experimentos foram executados em um computador equipado com processador Intel Xeon E5-2680v2 (2,8 GHz), 32 GB de memória RAM, sob o sistema operacional CentOS Linux 7.2.1511. As implementações foram desenvolvidas em C++, e cada execução foi limitada a um tempo máximo de 600 segundos. O foco da análise não se concentra primordialmente no tempo de processamento ou no consumo de memória – aspectos reconhecidamente críticos em algoritmos *bottom-up*, em razão da explosão combinatória no número de soluções parciais –, mas sim na avaliação do compromisso entre a taxa de utilização do material e o número de estágios guilhotinados.

5.1. Conjuntos de Instâncias de *Benchmark*

Foram considerados três conjuntos de instâncias da literatura: dois para o PC2GR-k e um para o PC3GR-k. O conjunto #A contém 15 instâncias de *benchmark* para o PC2GR-k. As instâncias gcut01-gcut12 foram obtidas do repositório online OR-Library[1], enquanto as demais foram extraídas de Wang (1983) e Oliveira e Ferreira (1990). O conjunto #B é composto por 11 instâncias de *benchmark* para o PC2GR-k, propostas por Fayard et al. (1998) e disponibilizadas em `https://oscar-oliveira.github.io/2D-Cutting-and-Packing/`. O conjunto #C reúne 24 instâncias de *benchmark* para o PC3GR-k. Destas, 12 foram propostas por Queiroz et al. (2012) (gcut01_3d-gcut12_3d) e estão disponíveis em `https://www.loco.ic.unicamp.br/files/instances/3duk/`. As 12 instâncias restantes consideram $u_i = 1$, $\forall i \in I$, possuem o sufixo "_cons" e seguem a mesma configuração adotada em Martin et al. (2021), ao passo que as instâncias de Queiroz et al. (2012) admitem $u_i \geq 1$.

O parâmetro β, que limita a perda admissível em soluções parciais, é definido de forma adaptativa ao longo das execuções do Algoritmo 2. Inicialmente, para $k = 0$, adota-se $\beta = 1.0$. Para as execuções subsequentes com $k \geq 1$, o valor de β é atualizado com base na melhor solução obtida na configuração $k-1$. Especificamente, define-se $\beta_k = \min\{\beta_{k-1}, \frac{P_{k-1}}{LW} + \epsilon\}$, em que P_{k-1} representa a perda da melhor solução encontrada anteriormente e LW (ou LWH) a dimensão total do objeto. Essa estratégia permite que cada execução herde um limiar de perda consistente com o melhor padrão de corte obtido com menor complexidade estrutural, intensificando a poda do espaço de busca à medida que k cresce. Adicionalmente, para mitigar a explosão combinatória de soluções parciais, impõem-se limites superiores empíricos para β, denotados por $\bar{\beta}_{\text{set}}$, que variam conforme o conjunto de instâncias e o número de estágios. No Algoritmo 2, para $k \geq 2$ (em 2D) ou $k \geq 3$ (em 3D), e no Algoritmo 1, aplica-se o ajuste $\beta \leftarrow \min\{\beta, \bar{\beta}_{\text{set}}\}$. Os valores adotados foram $\bar{\beta}_{\text{set}} = 0.250$ para o conjunto #A, $\bar{\beta}_{\text{set}} = 0.075$ para o conjunto #B e $\bar{\beta}_{\text{set}} = 0.300$ para o conjunto #C. Tais limiares foram estabelecidos experimentalmente para equilibrar a qualidade da solução final e a viabilidade de memória das listas de soluções parciais.

[1] `https://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/`

Tabela 2: Resultados para o conjunto #A de instâncias.

m	n	Instância	PC2GR-0			PC2GR-1			PC2GR-2			PC2GR-3			PC2GR-4			PC2GR-∞		
			Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt
10	10	gcut01	49.16	< 0.01	✓	67.76	0.01	✓	77.39	< 0.01	✓	77.39	< 0.01	✓	77.39	< 0.01	✓	77.39	< 0.01	✓
		gcut05	40.16	< 0.01	✓	67.98	< 0.01	✓	77.35	< 0.01	✓	77.35	< 0.01	✓	78.23	< 0.01	✓	78.23	< 0.01	✓
		gcut09	31.50	< 0.01	✓	54.94	0.01	✓	91.95	0.02	✓	91.95	0.02	✓	91.95	0.02	✓	91.95	0.01	✓
	23	OF1	17.68	< 0.01	✓	84.14	0.01	✓	96.89	0.05	✓	96.89	0.01	✓	96.89	0.01	✓	97.75	0.15	✓
	24	OF2	20.57	< 0.01	✓	53.36	0.01	✓	90.07	0.09	✓	96.07	0.07	✓	96.07	0.02	✓	96.07	0.16	✓
19	42	wang20	47.61	< 0.01	✓	81.32	0.02	✓	93.68	0.04	✓	97.18	0.02	✓	97.18	0.01	✓	97.18	0.05	✓
20	20	gcut02	40.18	< 0.01	✓	66.08	< 0.01	✓	94.89	0.01	✓	94.89	0.01	✓	94.89	0.01	✓	94.89	0.02	✓
		gcut06	38.19	< 0.01	✓	67.95	< 0.01	✓	94.52	0.01	✓	94.52	< 0.01	✓	94.52	< 0.01	✓	94.52	0.01	✓
		gcut10	53.80	< 0.01	✓	73.09	0.01	✓	90.34	0.01	✓	90.34	< 0.01	✓	90.34	< 0.01	✓	90.34	0.02	✓
30	30	gcut03	39.64	< 0.01	✓	69.23	0.01	✓	95.83	0.07	✓	96.39	0.02	✓	96.39	0.02	✓	96.39	0.06	✓
		gcut07	54.60	< 0.01	✓	69.78	< 0.01	✓	95.59	0.03	✓	95.59	0.01	✓	95.59	0.01	✓	95.59	0.03	✓
		gcut11	40.30	< 0.01	✓	66.03	0.02	✓	95.44	0.07	✓	95.54	0.03	✓	95.54	0.03	✓	95.54	0.08	✓
50	50	gcut04	50.12	< 0.01	✓	71.29	0.13	✓	97.51	0.69	✓	97.51	0.07	✓	97.51	0.07	✓	97.51	0.60	✓
		gcut08	54.16	0.01	✓	72.13	0.11	✓	98.30	0.59	✓	98.30	0.04	✓	98.30	0.04	✓	98.30	0.53	✓
		gcut12	52.71	< 0.01	✓	71.96	0.03	✓	97.07	0.21	✓	97.07	0.04	✓	97.07	0.03	✓	97.07	0.19	✓

Tabela 3: Resultados para o conjunto #B de instâncias.

m	n	Instância	PC2GR-0			PC2GR-1			PC2GR-2			PC2GR-3			PC2GR-4			PC2GR-∞		
			Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt
25	56	CU07	32.25	< 0.01	✓	59.84	0.06	✓	95.24	0.27	✓	95.85	0.17	✓	95.85	0.17	✓	95.85	0.37	✓
	76	CU09	31.41	< 0.01	✓	59.20	0.08	✓	97.78	0.22	✓	97.78	0.05	✓	97.78	0.05	✓	97.78	0.09	✓
	82	CU01	32.23	< 0.01	✓	62.95	0.70	✓	98.50	0.80	✓	98.64	0.17	✓	98.64	0.14	✓	98.64	1.07	✓
34	90	CU02	33.30	< 0.01	✓	59.54	0.24	✓	99.43	0.75	✓	99.43	0.05	✓	99.43	0.05	✓	99.43	0.44	✓
35	78	CU08	35.57	< 0.01	✓	63.95	0.16	✓	97.39	0.44	✓	97.51	0.14	✓	97.65	0.19	✓	97.65	0.17	✓
40	129	CU10	24.15	< 0.01	✓	57.46	112.75	✓	97.17	tl	–	98.90	345.87	✓	98.94	30.83	✓	97.17	tl	–
45	113	CU04	32.05	< 0.01	✓	60.54	2.44	✓	97.83	5.10	✓	98.46	1.95	✓	98.62	1.85	✓	98.62	3.07	✓
	124	CU06	38.23	< 0.01	✓	61.59	6.63	✓	99.65	4.69	✓	99.65	0.04	✓	99.65	0.04	✓	99.65	3.96	✓
	158	CU03	35.79	< 0.01	✓	58.88	9.30	✓	99.15	5.21	✓	99.58	1.14	✓	99.84	0.38	✓	99.84	2.67	✓
48	134	CU11	20.28	< 0.01	✓	59.09	tl	–	*	*	*	*	*	*	*	*	*	96.33	tl	–
50	120	CU05	39.25	< 0.01	✓	64.08	2.18	✓	97.77	4.34	✓	98.55	1.43	✓	98.75	1.03	✓	98.75	3.64	✓

Notas: (i) m é o número de tipos de itens e n é o número total de itens (i.e., $n = \sum_{i \in I} u_i$); (ii) as soluções para PC2GR-k com $k = 0, 1, 2, 3, 4$ e $k = \infty$ foram obtidas respectivamente pelos Algoritmos 2 e 1; (iii) a coluna Ut.[%] indica a utilização de material em porcentagem, a coluna t[s] o tempo computacional em segundos, e a coluna opt se a solução é ótima (✓) ou não (–); (iv) o símbolo "*" indica estouro de memória; e, (v) tl indica que o tempo limite de 600 segundos foi atingido.

5.2. Resultados Computacionais

As Tabelas 2 e 3 apresentam os resultados dos experimentos para os conjuntos #A e #B do PC2GR-k, respectivamente. Para cada instância, são indicados o número de tipos de itens (m), o número total de cópias (n) e o nome da instância. No caso do Algoritmo 2, considerou-se k variando de zero a quatro estágios guilhotinados. Para cada método, são reportados a utilização de material em porcentagem, o tempo de processamento computacional em segundos e a presença de certificado de otimalidade da solução. A análise da Tabela 2 para o conjunto #A evidencia um ganho substancial na utilização ao se passar de $k = 0$ (soluções correspondentes a um único item) para $k = 1$ (uma única tira) e, principalmente, para $k = 2$ (padrões de 2-estágios). Em várias instâncias, $k = 2$ já atinge o mesmo nível de utilização do caso não-estagiado PC2GR-∞ (por exemplo, gcut01, gcut02, gcut06, gcut09, gcut10, gcut04, gcut08 e gcut12), indicando que, para esse conjunto, a "fronteira" de qualidade é frequentemente alcançada com baixa complexidade estrutural. Ainda assim, existem casos em que o acréscimo de estágios produz incrementos marginais, porém mensuráveis: em wang20 e gcut03, a utilização melhora ao passar de $k = 2$ para $k = 3$ (até coincidir com PC2GR-∞), enquanto em gcut05 a melhora ocorre apenas com $k = 4$. Um ponto relevante é que quase todas as instâncias do conjunto #A atingem a utilização de PC2GR-∞ com $k \leq 4$ (com exceção de OF1, em que mesmo $k = 4$ permanece abaixo do valor reportado para PC2GR-∞), reforçando empiricamente que o limite explícito de estágios consegue capturar a maior parte do potencial de ocupação sem exigir padrões de corte mais complexos do que o necessário.

A Tabela 3, referente ao conjunto #B, confirma o mesmo comportamento de retornos decrescentes, embora em instâncias mais exigentes: $k = 2$ já eleva a utilização para a faixa de $\approx 95\%$–99% na maioria dos casos, enquanto estágios adicionais tendem a refinar a solução com ganhos menores (por exemplo, CU03 evolui de $99{,}15\%$ em $k = 2$ para $99{,}58\%$ em $k = 3$ e $99{,}84\%$ em $k = 4$, coincidindo com PC2GR-∞; comportamento similar ocorre em CU04, CU05 e CU08). Do ponto de vista do compromisso utilização *versus* número de estágios, isso sugere que políticas limitando k a 2 ou 3 podem capturar quase toda a eficiência de material, reservando $k = 4$ apenas para casos em que frações adicionais de utilização justificam maior complexidade operacional.

Os resultados para o conjunto #B evidenciam também a pressão computacional típica dos algoritmos *bottom-up*, pois há instâncias em que o tempo limite e/ou estouro de memória passam a dominar (CU11 atinge o tempo limite já em $k = 1$ e não viabiliza $k \geq 2$ por memória; além disso, PC2GR-∞ também não certifica otimalidade). Um aspecto interessante é que a restrição em k pode atuar como mecanismo prático de "focalização" da busca: em CU10, por exemplo, a melhor utilização reportada com $k = 3$ e $k = 4$ supera aquela obtida pelo PC2GR-∞ sob o mesmo limite de tempo, ilustrando que controlar a complexidade estrutural pode não apenas reduzir padrões inviáveis do ponto de vista produtivo, mas também conduzir a um espaço de busca mais tratável e, na prática, a soluções de melhor qualidade dentro do orçamento computacional. Quanto ao estouro de memória na instância CU11 para $k \geq 2$, o fenômeno resulta da alta densidade de subpadrões produzida pela sensibilidade das dimensões dos itens ao parâmetro β. Tal fato demonstra que a complexidade do Algoritmo 2 é governada pela efetividade da poda e pela estrutura dos dados, independentemente da escala do problema. A configuração da CU11 induz uma explosão combinatória superior à das instâncias do conjunto #C, comportamento que poderia ser evitado mediante uma calibração distinta de β.

A Tabela 4 apresenta os resultados dos experimentos para o conjunto #C do PC3GR-k. Nela, observa-se um *trade-off* ainda mais acentuado: os valores de utilização são tipicamente baixos para $k \leq 2$ e aumentam significativamente ao permitir $k = 3$, ou seja, quando padrões com maior capacidade de cortes em três eixos passam a ser admissíveis (por exemplo, gcut09_3d sobe de $55{,}15\%$ em $k = 2$ para $93{,}16\%$ em $k = 3$; gcut03_3d_cons aumenta de $59{,}04\%$ para $83{,}58\%$; e gcut06_3d_cons cresce de forma mais gradual até $76{,}28\%$ em $k = 4$, já igual ao PC3GR-∞).

Tabela 4: Resultados para o conjunto #C de instâncias.

m	n	Instância	PC3GR-0			PC3GR-1			PC3GR-2			PC3GR-3			PC3GR-4			PC3GR-∞		
			Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt	Ut.[%]	t[s]	opt
10	10	gcut01_3d_cons	28.12	< 0.01	✓	40.73	< 0.01	✓	58.34	< 0.01	✓	62.73	0.01	✓	50.07	0.01	–	50.07	0.01	–
		gcut05_3d_cons	23.06	< 0.01	✓	35.36	< 0.01	✓	51.20	0.01	✓	56.86	0.02	✓	51.20	0.01	–	51.20	0.01	–
		gcut09_3d_cons	20.17	< 0.01	✓	29.22	< 0.01	✓	46.00	0.03	✓	61.45	0.11	✓	61.45	0.11	–	61.45	0.08	–
	57	gcut01_3d	28.12	< 0.01	✓	40.73	0.01	✓	58.34	0.07	✓	80.57	0.23	✓	80.57	0.26	✓	80.57	0.27	✓
	60	gcut05_3d	23.06	< 0.01	✓	46.12	< 0.01	✓	59.15	0.24	✓	83.83	0.94	✓	84.34	1.17	✓	84.34	1.55	✓
	82	gcut09_3d	20.17	< 0.01	✓	34.96	0.01	✓	55.15	4.63	✓	93.16	20.32	✓	93.16	1.97	✓	93.16	38.15	✓
20	20	gcut02_3d_cons	21.70	< 0.01	✓	36.56	0.01	✓	52.14	0.50	✓	72.40	3.61	–	73.30	3.85	✓	73.83	2.36	✓
		gcut06_3d_cons	24.57	< 0.01	✓	40.35	< 0.01	✓	60.19	0.03	✓	71.25	0.05	✓	76.28	0.04	✓	76.28	0.04	✓
		gcut10_3d_cons	28.02	< 0.01	✓	50.20	< 0.01	✓	59.25	0.01	✓	71.63	0.02	✓	71.63	0.01	✓	71.63	0.01	✓
	75	gcut10_3d	28.02	0.01	✓	50.20	0.01	✓	62.47	0.05	✓	85.16	0.13	✓	85.16	0.13	✓	85.16	0.14	✓
	82	gcut06_3d	24.57	< 0.01	✓	40.90	0.01	✓	65.79	0.35	✓	81.84	0.73	✓	84.84	1.20	✓	84.84	0.94	✓
	113	gcut02_3d	21.70	< 0.01	✓	36.56	0.03	✓	55.28	27.07	✓	84.35	78.42	✓	84.35	274.79	✓	84.88	234.26	✓
30	30	gcut03_3d_cons	17.90	< 0.01	✓	37.71	0.14	✓	59.04	93.19	✓	83.58	239.79	✓	85.40	177.20	✓	87.09	306.04	✓
		gcut07_3d_cons	31.70	< 0.01	✓	45.48	0.02	✓	61.55	1.12	✓	80.59	4.00	✓	80.88	3.01	✓	80.88	5.23	✓
		gcut11_3d_cons	22.63	< 0.01	✓	37.08	0.04	✓	57.72	3.19	✓	74.92	21.79	✓	78.21	20.24	✓	78.21	11.86	✓
	132	gcut07_3d	31.70	< 0.01	✓	49.28	0.04	✓	65.35	14.75	✓	87.24	32.27	✓	88.11	22.52	✓	88.11	45.36	✓
	133	gcut11_3d	22.63	< 0.01	✓	39.37	0.06	✓	64.64	36.01	✓	90.78	23.62	✓	91.44	7.26	✓	91.44	32.25	✓
	207	gcut03_3d	17.90	< 0.01	✓	40.96	0.28	✓	62.44	tl	–	87.66	tl	–	89.11	tl	–	87.66	tl	–
50	50	gcut04_3d_cons	29.57	< 0.01	✓	47.24	0.54	✓	63.21	tl	–	87.30	tl	–	90.86	258.48	✓	88.07	tl	–
		gcut08_3d_cons	22.75	< 0.01	✓	42.61	0.39	✓	60.88	tl	–	86.58	tl	–	87.20	197.05	✓	86.58	tl	–
		gcut12_3d_cons	27.73	< 0.01	✓	45.27	0.24	✓	61.31	319.44	✓	87.68	400.70	✓	87.68	49.89	✓	87.68	410.04	✓
	251	gcut12_3d	27.73	< 0.01	✓	49.02	0.43	✓	64.85	tl	–	91.23	tl	–	92.65	144.58	✓	91.23	tl	–
	270	gcut08_3d	22.75	< 0.01	✓	45.49	0.67	✓	68.40	tl	–	88.90	tl	–	92.89	tl	–	88.90	tl	–
	274	gcut04_3d	29.57	< 0.01	✓	47.24	0.97	✓	66.10	tl	–	91.59	tl	–	94.93	tl	–	91.59	tl	–

Notas: (i) m é o número de tipos de itens e n é o número total de itens (i.e., $n = \sum_{i \in I} u_i$); (ii) as soluções para PC3GR-k com $k = 0, 1, 2, 3, 4$ e $k = \infty$ foram obtidas respectivamente pelos Algoritmos 2 e 1; (iii) a coluna Ut.[%] indica a utilização de material em porcentagem, a coluna t[s] o tempo computacional em segundos, e a coluna opt se a solução é ótima (✓) ou não (–); e, (iv) tl indica que o tempo limite de 600 segundos foi atingido.

Em diversas instâncias de porte moderado, $k = 4$ é suficiente para atingir o mesmo nível de utilização do caso não-estagiado, com certificado de otimalidade, sugerindo que, assim como no caso bidimensional, a utilidade marginal de liberar estágios adicionais tende a se esgotar rapidamente após certo limiar. Entretanto, à medida que n aumenta, tornam-se frequentes os tempos limites no PC3GR-∞ e também para alguns valores intermediários de k, momento em que a limitação explícita de estágios pode se revelar vantajosa: há instâncias em que $k = 4$ ainda permite certificar soluções de alta utilização enquanto o PC3GR-∞ permanece sem certificado dentro do tempo (por exemplo, instâncias maiores "_cons" como gcut04_3d_cons e gcut08_3d_cons).

Por outro lado, em algumas instâncias pequenas "_cons" (gcut01_3d_cons, gcut05_3d_cons e gcut09_3d_cons), nota-se que os resultados para $k = 4$ e para ∞ não apresentam melhora (e, em alguns casos, ficam abaixo de $k = 3$, sem certificado). Isso é consistente com o fato de que, na prática, a combinação entre limite de tempo e regras de poda baseadas em β pode tornar o procedimento mais restritivo do que o necessário. Nesse sentido, os dados reforçam a recomendação operacional de interpretar k como um controle de complexidade e selecionar, para cada instância, o menor k que estabiliza a utilização (ou o melhor valor obtido ao longo de $k = 0, \ldots, 4$), em vez de presumir ganhos monotônicos ao intensificar simultaneamente as podas do espaço de busca.

5.3. Discussões de Caráter Gerencial

A análise dos resultados computacionais permite formular recomendações operacionais para gestores que utilizam máquinas de corte guilhotinado. Nesses ambientes, o parâmetro k representa diretamente o número máximo de reorientações de corte admitidas em um padrão, com impacto negativo na produtividade da máquina em função da maior complexidade de operação. O *trade-off* consiste em determinar o menor valor de k que proporciona utilização satisfatória do material, sem impor padrões de corte cuja complexidade estrutural não se justifica do ponto de vista operacional.

A Figura 5 ilustra a evolução da utilização média por conjunto de instâncias em função de k. Para os conjuntos do PC2GR-k, observa-se ganho expressivo na transição de $k = 1$ para $k = 2$, a partir do qual os incrementos tornam-se rapidamente marginais: a utilização média do conjunto #A passa de $69{,}1\%$ em $k = 1$ para $92{,}4\%$ em $k = 2$ e estabiliza em torno de $93{,}2\%$ para $k \geq 3$; de forma análoga, o conjunto #B atinge $98{,}0\%$ em $k = 2$, com incrementos inferiores a $0{,}6$ p.p. nos estágios seguintes. Para o conjunto do PC3GR-k, o comportamento difere: o pico de acréscimo ocorre na transição de $k = 2$ para $k = 3$ (de $60{,}0\%$ para $81{,}0\%$), o que reflete a maior capacidade expressiva conferida pelos cortes em três eixos, tornando $k = 3$ no PC3GR-k o análogo de $k = 2$ no PC2GR-k.

Esse comportamento é quantificado na Figura 6, que apresenta os ganhos marginais médios de utilização ($\Delta\overline{\mathrm{Ut.}} = \overline{\mathrm{Ut.}}[k] - \overline{\mathrm{Ut.}}[k-1]$) por transição de estágio e por conjunto. No PC2GR-k, os maiores incrementos concentram-se nas duas primeiras transições, com ganhos adicionais inferiores a 1 p.p. a partir de $k = 3$. No PC3GR-k, o maior ganho marginal ocorre precisamente em $k = 2 \rightarrow 3$. Os incrementos associados a $k = 4$ e ao caso não-estagiado são virtualmente nulos ou ligeiramente negativos em alguns conjuntos, reflexo das restrições de tempo computacional que, sob orçamento fixo, tornam configurações menos estruturadas nem sempre superiores às mais restritas.

Com base nessas evidências, propõe-se o seguinte protocolo de seleção de k: (i) no PC2GR-k, adotar $k = 2$ como configuração de referência, avaliando $k = 3$ apenas quando a diferença de utilização em relação ao PC2GR-k-∞ for economicamente relevante para a instância em questão; (ii) no PC3GR-k, adotar $k = 3$ como referência, com $k = 4$ reservado a instâncias de porte moderado nas quais o certificado de otimalidade seja necessário; (iii) em ambos os casos, executar o Algoritmo 2 de forma incremental, de $k = 0$ a $k = 4$, e selecionar o menor k a partir do qual a utilização estabiliza, aproveitando o mecanismo adaptativo de β para reduzir o custo computacional das execuções subsequentes.

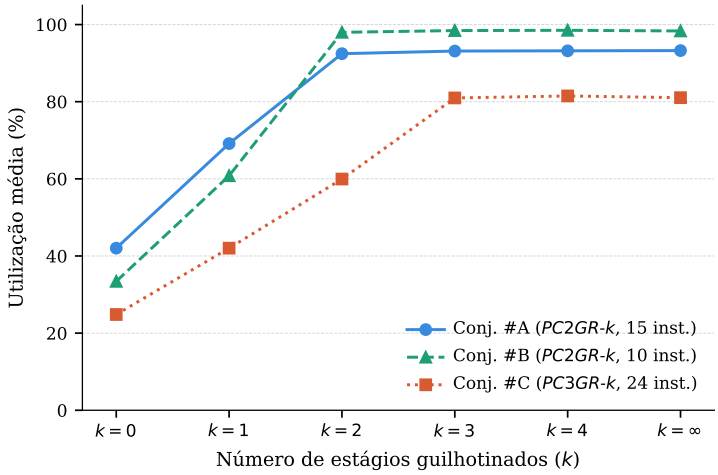


Figura 5: Utilização média por conjunto de instâncias em função de k.
Fonte: Elaboração própria.

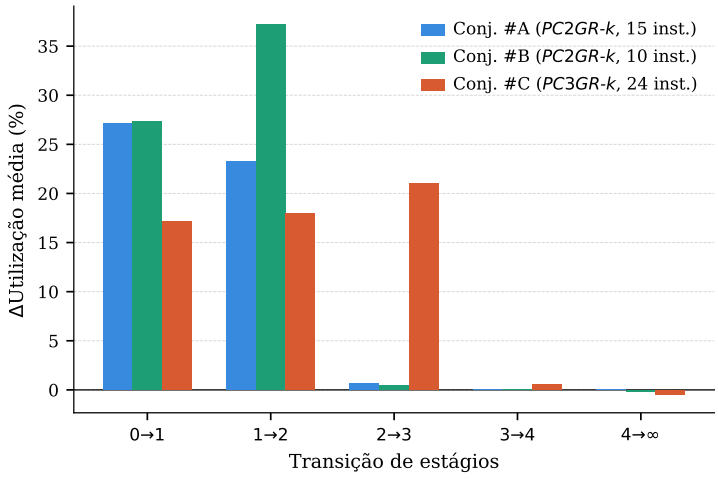


Figura 6: Ganho marginal médio de utilização por transição de estágio.
Fonte: Elaboração própria.

6. Conclusões

Este trabalho abordou os problemas de corte bidimensional e tridimensional guilhotinados k-estágios e restritos (PC2GR-k e PC3GR-k), variantes de relevância prática em setores como vidro, madeira e aço, nos quais a produtividade está fortemente associada à simplicidade estrutural dos padrões de corte. Foi proposto um algoritmo combinatório *bottom-up* que estende o método de Wang (1983), incorporando o controle explícito do número de estágios guilhotinados por meio de *strings* e a resolução do problema da super-sequência comum mínima para verificar a factibilidade das combinações. Os experimentos computacionais com instâncias de *benchmark* mostram que o algoritmo captura o *trade-off* entre utilização do material e complexidade estrutural dos padrões. Para a maioria das instâncias bidimensionais, padrões com 2- ou 3-estágios já atingem níveis de utilização equivalentes aos de padrões não-estagiados. No caso tridimensional, os ganhos tornam-se mais significativos a partir do terceiro estágio. Além disso, a imposição do limite k mostrou-se eficaz, no contexto das instâncias consideradas, para conter a explosão combinatória típica de métodos *bottom-up*, permitindo a obtenção de soluções de alta qualidade em tempos computacionais competitivos.

Como perspectivas futuras, destaca-se a extensão do algoritmo para a versão ponderada do problema, na qual cada tipo de item possui valor arbitrário não necessariamente proporcional à área ou volume. Outra direção relevante é o desenvolvimento de regras de dominância e estratégias de poda mais fortes, possivelmente baseadas em limites superiores provenientes de relaxações lineares ou lagrangianas, visando aumentar a escalabilidade para instâncias maiores ou com maior diversidade de itens. Por fim, a integração do algoritmo *bottom-up* como gerador de colunas em esquemas de *column generation* para problemas de corte de estoque constitui também uma linha promissora, especialmente quando restrições de estágios precisam ser preservadas em larga escala.

Agradecimentos. Os autores agradecem o apoio financeiro da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP–Brasil) [números de processo 2024/20161-3, 2022/05803-3] e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq–Brasil) [408722/2023-1, 402492/2025-0]. Pesquisa desenvolvida com utilização dos recursos computacionais do Centro de Ciências Matemáticas Aplicadas à Indústria (CeMEAI), financiados pela FAPESP [número de processo 2013/07375-0].

Referências

Alem, D. J. e Morabito, R. Production planning in furniture settings via robust optimization. *Computers & Operations Research*, v. 39, p. 139–150, 2012.

Alvarez-Valdés, R., Parajón, A., e Tamarit, J. M. A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Computers & Operations Research*, v. 29, p. 925–947, 2002. ISSN 03050548.

Armbruster, M. A solution procedure for a pattern sequencing problem as part of a one-dimensional cutting stock problem in the steel industry. *European Journal of Operational Research*, v. 141, p. 328–340, 2002. ISSN 03772217.

Baazaoui, M., Elleuch, S., e Kammoun, H. An efficient constructive heuristic for the cutting stock problem applied in a foam mattress industry. *International Journal of Applied Decision Sciences*, v. 16, n. 3, p. 313–333, 2023.

Barreto, A., Cherri, A. C., Cherri, L. H., e Nogueira Nascimento, D. O uso de k soluções para o

problema de corte de estoque com sobras aproveitáveis. *Pesquisa Operacional para o Desenvolvimento*, v. 18, p. 1–24, 2024.

Becker, H., Araújo, O., e Buriol, L. S. Enhanced formulation for the guillotine 2d cutting knapsack problem. *Mathematical Programming Computation*, v. 14, p. 673–697, 2022. ISSN 18672957.

Christofides, N. e Whitlock, C. An algorithm for two-dimensional cutting problems. *Operations Research*, v. 25, p. 30–44, 1977. ISSN 0030-364X.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. *Introduction to Algorithms*. Cambridge, MA: MIT Press3 edition, 2009.

Cung, V.-D., Hifi, M., e Cun, B. L. Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. *International Transactions in Operational Research*, v. 7, p. 185–210, 2000. ISSN 09696016.

Daza, V. P., de Alvarenga, A. G., e de Diego, J. Exact solutions for constrained two-dimensional cutting problems. *European Journal of Operational Research*, v. 84, p. 633–644, 1995. ISSN 03772217.

De Armas, J., Miranda, G., e León, C. Improving the efficiency of a best-first bottom-up approach for the constrained 2d cutting problem. *European Journal of Operational Research*, v. 219, p. 201–213, 2012. ISSN 03772217.

Do Nascimento, O. X., de Queiroz, T. A., e Junqueira, L. A MIP-CP based approach for two- and three-dimensional cutting problems with staged guillotine cuts. *Annals of Operations Research*, v. 316, p. 805–835, 2022. ISSN 15729338.

Durak, B. e Aksu, D. T. Dynamic programming and mixed integer programming based algorithms for the online glass cutting problem with defects and production targets. *International Journal of Production Research*, v. 55, n. 24, p. 7398–7411, 2017.

Dyson, R. G. e Gregory, A. S. The cutting stock problem in the flat glass industry. *Journal of the Operational Research Society*, v. 25, n. 1, p. 41–53, 1974.

Eilon, S. Optimizing the shearing of steel bars. *Journal of Mechanical Engineering Science*, v. 2, n. 2, p. 129–142, 1960. http://dx.doi.org/10.1243/JMES_JOUR_1960_002_022_02.

Fayard, D., Hifi, M., e Zissimopoulos, V. An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *Journal of the Operational Research Society*, v. 49, n. 12, p. 1270–1277, 1998. ISSN 1476-9360.

Gilmore, P. C. e Gomory, R. E. A linear programming approach to the cutting stock problem—part ii. *Operations Research*, v. 11, p. 863–888, 1963. ISSN 0030-364Xhttps://pubsonline.informs.org/doi/10.1287/opre.11.6.863.

Gilmore, P. C. e Gomory, R. E. Multistage cutting stock problems of two and more dimensions. *Operations Research*, v. 13, p. 94–120, 1965. ISSN 0030-364X.

Gusfield, D. *Algorithms on Strings, Trees, and Sequences*. Cambridge: Cambridge University Press, 1997.

Hadjiconstantinou, E. e Christofides, N. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, v. 83, p. 39–56, 1995. ISSN 03772217.

Hifi, M. An improvement of viswanathan and bagchi's exact algorithm for constrained two-dimensional cutting stock. *Computers & Operations Research*, v. 24, p. 727–736, 1997. ISSN 03050548.

Iori, M., De Lima, V. L., Martello, S., Miyazawa, F. K., e Monaci, M. Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, v. 289, p. 399–415, 2021. ISSN 03772217.

Kozyreff Filho, E. e Araújo, S. A. Minimização do comprimento de ciclo em máquinas de corte de vidro. *Pesquisa Operacional para o Desenvolvimento*, v. 18, p. 1–18, 2024.

Leao, A. A., Toledo, F. M., Oliveira, J. F., Carravilla, M. A., e Alvarez-Valdés, R. Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, v. 282, n. 3, p. 803–822, 2020. ISSN 0377-2217.

Lodi, A., Martello, S., e Monaci, M. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, v. 141, p. 241–252, 2002. ISSN 03772217.

Lodi, A. e Monaci, M. Integer linear programming models for 2-staged two-dimensional knapsack problems. *Mathematical Programming, Series B*, v. 94, p. 257–278, 2003. ISSN 00255610.

Macedo, R., Alves, C., e de Carvalho, J. M. V. Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research*, v. 37, p. 991–1001, 2010. ISSN 03050548http://dx.doi.org/10.1016/j.cor.2009.08.005.

Maier, D. The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, v. 25, n. 2, p. 322–336, 1978.

Martin, M., Morabito, R., e Munari, P. A bottom-up packing approach for modeling the constrained two-dimensional guillotine placement problem. *Computers & Operations Research*, v. 115, 2020. ISSN 03050548.

Martin, M., Oliveira, J. F., Silva, E., Morabito, R., e Munari, P. Three-dimensional guillotine cutting problems with constrained patterns: MILP formulations and a bottom-up algorithm. *Expert Systems with Applications*, v. 168, 2021. ISSN 09574174.

Matsumoto, K., Umetani, S., e Nagamochi, H. On the one-dimensional stock cutting problem in the paper tube industry. *Journal of Scheduling*, v. 14, p. 281–290, 2011. ISSN 10946136.

Morabito, R. e Arenales, M. Optimizing the cutting of stock plates in a furniture company. *International Journal of Production Research*, v. 38, p. 2725–2742, 2000. ISSN 00207543.

Muzulon, N. Z., Leal, G. C. L., e Lima, R. H. P. Proposta e aplicação de um método para seleção de softwares de geração de padrões e planos de corte em micro e pequenas empresas moveleiras. *Pesquisa Operacional para o Desenvolvimento*, v. 18, p. 1–26, 2024.

Oliveira, J. F. e Ferreira, J. S. An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research*, v. 44, p. 256–266, 1990. ISSN 03772217.

Oliveira, O., Gamboa, D., e Silva, E. An introduction to the two-dimensional rectangular cutting and packing problem. *International Transactions in Operational Research*, v. 30, p. 3238–3266, 2023. ISSN 14753995.

Puchinger, J. e Raidl, G. R. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, v. 183, p. 1304–1327, 2007. ISSN 03772217.

Queiroz, T. A. D., Miyazawa, F. K., Wakabayashi, Y., e Xavier, E. C. Algorithms for 3d guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing. *Computers & Operations Research*, v. 39, p. 200–212, 2012. ISSN 03050548.

Russo, M., Boccia, M., Sforza, A., e Sterle, C. Constrained two-dimensional guillotine cutting problem: upper-bound review and categorization. *International Transactions in Operational Research*, v. 27, p. 794–834, 2020. ISSN 14753995.

Scheithauer, G. *Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods*. Cham: Springer, 2018.

Silva, E., Alvelos, F., e de Carvalho, J. M. V. An integer programming model for two- and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research*, v. 205, p. 699–708, 2010. ISSN 03772217.

Vasko, F. J. A computational improvement to Wang's two-dimensional cutting stock algorithm. *Computers & Industrial Engineering*, v. 16, p. 109–115, 1989. ISSN 03608352.

Vasko, F. J. e Bartkowski, C. L. Using Wang's two-dimensional cutting stock algorithm to optimally solve difficult problems. *International Transactions in Operational Research*, v. 16, p. 829–838, 2009. ISSN 14753995.

Vazirani, V. V. *Approximation Algorithms*. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-540-65367-7, 2001.

Velasco, A. S. e Uchoa, E. Improved state space relaxation for constrained two-dimensional guillotine cutting problems. *European Journal of Operational Research*, v. 272, p. 106–120, 2019. ISSN 0377-2217.

Viswanathan, K. V. e Bagchi, A. Best-first search methods for constrained two-dimensional cutting stock problems. *Operations Research*, v. 41, p. 768–776, 1993. ISSN 0030-364X.

Wang, P. Y. Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, v. 31, p. 573–586, 1983. ISSN 0030-364X.

Wang, S., Baldacci, R., Liu, Q., e Wei, L. EATKG: An open-source efficient exact algorithm for the two-dimensional knapsack problem with guillotine constraints. *European Journal of Operational Research*, v. , 2025. ISSN 03772217.

Wei, L. e Lim, A. A bidirectional building approach for the 2d constrained guillotine knapsack packing problem. *European Journal of Operational Research*, v. 242, p. 63–71, 2015. ISSN 03772217.

Wäscher, G., Haußner, H., e Schumann, H. An improved typology of cutting and packing problems. *European Journal of Operational Research*, v. 183, p. 1109–1130, 2007. ISSN 03772217.

Yoon, K., Ahn, S., e Kang, M. An improved best-first branch-and-bound algorithm for constrained two-dimensional guillotine cutting problems. *International Journal of Production Research*, v. 51, p. 1680–1693, 2013. ISSN 00207543.