Pesquisa Operacional para o Desenvolvimento
Vol. 18, e18002, p. 1-18, 2024

ISSN 1984-3534

COMPARAÇÃO ENTRE PACOTES COMPUTACIONAIS BASEADOS EM *BRANCH-AND-CUT* PARA O EMPACOTAMENTO DE RETÂNGULOS[1]

Pedro Belin Castellucci[a] *, Aline A. S. Leao[b], Eduardo Delcides Bernardes[c]

[a] Departamento de Informática e Estatística,
Universidade Federal de Santa Catarina, Florianópolis (UFSC), Florianópolis-SC, Brasil

[b] Departamento de Matemática,
Universidade Estadual de Londrina (UEL), Londrina-PR, Brasil

[c] Departamento de Ciências Exatas,
Universidade Estadual de Santa Cruz (UESC), Ilhéus-BA, Brasil

Recebido 27/02/2023, aceito 06/03/2024

RESUMO

A importância prática do problema de empacotamento de retângulos é destacada na literatura para diversos tipos de indústrias. Em particular, pesquisadores da área de Otimização têm proposto diversos modelos de Programação Linear Inteira Mista e métodos de solução. No entanto, há uma carência de investigação empírica e comparativa entre modelos e pacotes computacionais. Para explorar tal lacuna, foram avaliados pacotes gratuitos e comerciais baseados em *branch-and-cut* para a solução de instâncias típicas da literatura do problema de empacotamento de retângulos. Os experimentos compararam os pacotes e modelos segundo a qualidade dos limitantes e o tempo computacional. Os resultados permitem não apenas uma comparação entre os pacotes e modelos, mas também uma avaliação gerencial sobre o investimento na obtenção de licença de pacotes comerciais.

Palavras-chave: Empacotamento, Solvers, Otimização, Programação linear inteira mista.

ABSTRACT

The rectangle packing problem is relevant in many practical contexts as shown in the Optimization literature. Particularly, researchers have been proposing different Mixed Integer Linear Programming Models and solution methods for decades. However, there is a lack of empirical and comparative evaluation between models and solvers. To bridge this gap, we evaluate free and commercial branch-and-cut-based solvers when applied to typical literature instances of the rectangle packing problem. The experiments analyze solvers and models according to bound quality and computational time. The results allow not only a comparison between solvers and models but also management insights related to of commercial software licenses.

Keywords: Packing, Solvers, Optimization, Mixed integer linear programming.

* Autor para correspondência. E-mail: pedro.castellucci@ufsc.br
DOI: https://doi.org/10.4322/PODes.2024.004

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

1. Introdução

Problemas de corte e empacotamento possuem bastante importância científica e industrial. Esses problemas vêm sendo estudados há décadas na área de Pesquisa Operacional (Wäscher et al., 2007; Dyckhoff, 1990). Além de estarem relacionados ao desenvolvimento da área de Otimização em si (Kantorovich, 1960; Gilmore e Gomory, 1961), o número de publicações relacionadas tem aumentado recentemente (Iori et al., 2021).

Os problemas de corte e empacotamento atendem a diversas demandas industriais. Um dos desafios dos profissionais da área de Pesquisa Operacional está nas escolhas das ferramentas a serem utilizadas para atacar o problema de interesse. Isso também ocorre ao se definir uma abordagem para a solução de problemas de empacotamento. Em particular, a escolha de pacotes computacionais resolvedores de Problemas de Programação Linear Inteira Mista, baseados no *brach-and-cut*, (p.e. CBC, SCIP, Gurobi e CPLEX) pode considerar além da qualidade das soluções obtidas, o custo de se adquirir a ferramenta. Em geral, há um compromisso entre o custo da ferramenta e a qualidade da solução que será obtida.

Gleixner et al. (2021) ressaltam que os resultados obtidos com experimentos comparando o desempenho médio de resolvedores não devem ser diretamente estendidos para qualquer classe de problemas (ou aplicações). Dessa forma, com este trabalho, pretende-se auxiliar na escolha da ferramenta de melhor custo-benefício para indústria interessada tanto para aplicação direta de modelos de Programação Linear Inteira Mista como para implementação de soluções baseadas em modelos (métodos exatos e *matheurísticas*).

Há diversas classes de problemas de empacotamento (vide, por exemplo, o trabalho de Wäscher et al. (2007)). Para o escopo deste estudo foi selecionado o problema de empacotamento bidimensional de itens retangulares dentro de um único recipiente retangular com o objetivo de maximizar o valor dos itens escolhidos para o empacotamento. Na tipologia de Wäscher et al. (2007), o problema de empacotamento de retângulos mencionado pertence às classes *Single Large Object Placement Problem* (SLOPP) e/ou *Single Knapsack Problem* (SKP).

Existem algumas restrições em relação ao padrão de empacotamento que podem ser necessárias como, por exemplo, o corte guilhotinado com limitação ou não do número de estágios. Esta restrição surge principalmente em problemas de corte devido às características do tipo de máquina utilizada para realizar o corte. Neste trabalho, consideramos o padrão de empacotamento não guilhotinado, que pode surgir isoladamente, como no problema carregamento de paletes (Alvarez-Valdés et al., 2005), quando a máquina a ser utilizada no processo de corte não é do tipo guilhotina, ou em combinação com outros problemas (Silva et al., 2022). Outras aplicações para problemas de corte em empacotamento bidimensionais podem ser encontradas no trabalho de Iori et al. (2021).

Embora existam trabalhos comparando a qualidade de pacotes computacionais (sendo o de Gleixner et al. (2021) um dos mais relevantes), esses estudos são realizados em instâncias desafiadoras para Programação Linear Inteira Mista em geral, sem haver necessariamente características similares entre as instâncias. Não é de conhecimento dos autores um trabalho avaliando diferentes resolvedores e modelos para conjuntos de instâncias de problemas de empacotamento bidimensional não guilhotinado. Ao explorar essa lacuna, o objetivo desse trabalho é apresentar resultados comparativos em ambiente computacional equivalente para instâncias do problema de empacotamento de retângulos resolvidas utilizando os pacotes CBC (Forrest e Lougee-Heimer, 2005), SCIP (Bestuzheva et al., 2021), CPLEX (Nickel et al., 2022) e Gurobi (Gurobi Optimization, 2021).

O restante do texto é organizado da seguinte forma. Na Seção 2, é apresentada uma revisão da literatura para problemas de empacotamento bidimensional de itens retangulares e avaliação de resolvedores para problemas de Programação Linear Inteira Mista. Na Seção 3, é definido o problema tratado e são descritos os modelos utilizados nos experimentos que, por sua vez, são apresentados na Seção 4. Por fim, são tecidas conclusões na Seção 5.

2. Revisão da Literatura

Como descrito em Iori et al. (2021), na literatura, é possível encontrar aplicações para problemas de empacotamento bidimensional, por exemplo, nas áreas de telecomunicações (Martello, 2014), diagramação de páginas (Strecker e Hennig, 2009), escalonamento (Kwon e Lee, 2015), logística (Xu e Lee, 2018), entre outros (Oliveira et al., 2016; Júnior et al., 2022). Essa diversidade motivou diferentes classificações de problemas de corte e empacotamento como a de Wäscher et al. (2007), frequentemente utilizada.

Na classificação de Wäscher et al. (2007) os problemas podem ser classificados de acordo com a sua função objetivo, com a heterogeneidade do conjunto de recipientes, e do conjunto de itens e com a dimensão do problema (unidimensional, bidimensional, tridimensional etc). Os problemas de interesse, aqui, são os das classes SLOPP e SKP, conforme mencionado na Seção 1. Ambas NP-difíceis (Iori et al., 2022).

Esta revisão destaca trabalhos relacionados à modelagem e utilização de modelos de Programação Linear Inteira Mista para o problema de empacotamento de itens retangulares. Revisões mais abrangentes foram feitas por De Carvalho (2002), Lodi et al. (2002), Leao et al. (2020) e Iori et al. (2021).

Seguindo o paradigma de Programação Linear Inteira Mista, pode-se encontrar modelos na literatura de tamanho polinomial, pseudo-polinomial ou exponencial. Um modelo polinomial pode ser obtido adaptando a proposta de Chen et al. (1995) desenvolvida para a versão tridimensional. Um modelo similar foi proposto por Fasano (1999) cuja análise poliedral é realizada por Padberg (2000). Este tipo de modelo está baseado na decisão de posições relativas entre os pares de itens. Há também modelos de tamanho polinomial baseado em grafos como o de Fekete et al. (2007). Modelos como o originalmente proposto por Beasley (1985b), que indexam variáveis de decisão pelas possíveis coordenadas de posicionamento das peças, são tipicamente pseudo-polinomiais. Por fim, modelos baseados no problema de cobertura de conjuntos (p.e. Gilmore e Gomory (1965)), que enumeram padrões de empacotamento, possuem um número exponencial de variáveis.

Na literatura, esses modelos são estendidos para resolver outros tipos de problemas. Em Andrade et al. (2014), o modelo de Chen et al. (1995) foi adaptado para o problema de reaproveitamento de sobras. O modelo de Beasley (1985b) também foi adaptado para problemas que consideram estabilidade da carga, ordem de carregamento dos itens, carregamento de paletes e problemas com dimensões abertas (Alvarez-Valdés et al., 2005; de Queiroz e Miyazawa, 2014; Martello e Monaci, 2015; Junqueira et al., 2012a,b). Já os modelos exponenciais, tipicamente dão origem a métodos exatos de *branch-and-cut-and-price* (Belov e Scheithauer, 2006; Pisinger e Sigurd, 2007) e decomposição de Benders (Côté et al., 2014; Delorme et al., 2017; Côté et al., 2021).

Em termos industriais, é importante não apenas identificar as abordagens promissoras frequentemente (não exclusivamente) elaboradas em um contexto acadêmico, mas também desenvolver tais abordagens para o ambiente industrial. Para isso, é necessária a escolha da(s) ferramenta(s) computacionais que serão utilizadas no desenvolvimento. No caso de Programação Linear Inteira Mista, dentre as ferramentas a serem definidas está o pacote computacional (resolvedor/*solver*) que será utilizado na implementação.

Exemplos de estudos comparativos de pacotes computacionais nesse contexto podem ser encontrados, pelo menos, desde a década de 50. Hoffman et al. (1953) propuseram diferentes algoritmos para solução de problemas de otimização linear e observaram que muitas das conjecturas sobre os desempenhos relativos dos algoritmos só puderam ser avaliados empiricamente. A avaliação empírica de algoritmos para problemas de Programação Linear Inteira Mista continua sendo o principal método de comparação entre os algoritmos (Bixby et al., 1998; Achterberg et al., 2006; Gleixner et al., 2021).

Com o estabelecimento de pacotes computacionais para a solução de problemas de Programação Linear Inteira Mista, foram também se estabelecendo conjunto de instâncias de *benchmar-*

king para a comparação experimental entre esses resolvedores. Essa comparação utiliza conjuntos de instâncias desafiadoras (notadamente da biblioteca MIPLIB (Gleixner et al., 2021)). Mittelmann (2022) mantém resultados comparando o desempenho de diversos pacotes computacionais para tais instâncias. No entanto, a generalização das conclusões para outras instâncias (potencialmente mais frequentes na prática) não é recomendada (Bartz-Beielstein et al., 2020). Além disso, a melhora de algoritmos para uma classe de instâncias potencialmente degrada o desempenho desses algoritmos para outras classes, o que é uma consequência de teoremas de *no-free lunch* (Haftka, 2016). Por exemplo, de Oliveira et al. (2016) compararam os desempenhos dos resolvedores CPLEX e Gurobi para um problema de transporte com custo-fixo e concluíram que o Gurobi forneceu melhores resultados. Os próprios autores notaram que os resultados divergem do trabalho de Meindl e Templ (2012), que compararam ambos os resolvedores (entre outros) para um problema relacionado à segurança de dados.

Também encontramos outros trabalhos que apresentaram uma comparação de performance ou uma revisão dos resolvedores para a resolução de problemas de Programação Linear Inteira Mista usando sempre instâncias da MIPLIB, ou propuseram alguma estratégia ou configuração dos parâmetros relacionados aos recursos disponíveis. Em Linderoth e Ralphs (2005), os autores apresentaram uma revisão de oito resolvedores não comerciais ou de código aberto, que podem ser utilizados de forma gratuita, destacando os seus recursos a fim de facilitar uma escolha por qual usar para a resolução de um determinado problema. Já Atamtürk e Savelsbergh (2005) fizeram uma revisão dos resolvedores comerciais, especificamente dos CPLEX, LINDO e XPRESS-MP, apontando seus recursos e possíveis ajustes dos parâmetros para um melhor desempenho na resolução de problemas.

Estudos comparando resolvedores para problemas de empacotamento são escassos, duas exceções são os trabalhos de Silva et al. (2019) e Becker et al. (2023). Silva et al. (2019) realizaram experimentos comparando modelos para a versão 3D das classes SLOPP e SKP, considerando os pacotes CPLEX e Gurobi. Os resultados mostraram que o Gurobi foi superior ao CPLEX com mais frequência. Conclusões semelhantes foram apresentadas em experimentos preliminares do trabalho de Becker et al. (2023), que avaliou diferentes formulações para o problema de empacotamento bidimensional guilhotinado.

Apesar da literatura explorar modelos e métodos baseados em modelo para a solução de problemas de empacotamento de retângulos (e suas variantes), não é de conhecimento dos autores um trabalho comparando experimentalmente diferentes pacotes computacionais para a implementação dos modelos e métodos propostos. Em termos práticos, essa avaliação empírica é necessária para uma decisão informada sobre a escolha dos pacotes a serem utilizados. Este trabalho explora essa lacuna, para isso, na Seção 3 é definido o problema de interesse e os dois modelos de Programação Linear Inteira Mista implementados.

3. Definição do Problema e Modelos Avaliados

Seja $\mathcal{B}$ um conjunto de itens retangulares, cada um com seu comprimento e largura (ℓ_i, w_i) e valor $v_i, i \in \mathcal{B}$, respectivamente. Considere também um recipiente retangular de comprimento L e largura W. O problema de empacotamento de retângulos consiste em escolher um subconjunto de itens em $\mathcal{B}$ a serem posicionados no recipiente de forma que a soma dos valores dos itens escolhidos seja a maior possível. Para isso, é necessário que os itens escolhidos estejam completamente contidos no recipiente e não haja sobreposição entre qualquer par de itens $i, j \in \mathcal{B} \times \mathcal{B} : i < j$. Ainda, os itens devem ser posicionados com suas faces paralelas às faces do recipiente e não é permitido rotacionar os itens. Em resumo, os parâmetros do problema são:

- $\mathcal{B}$: conjunto de itens disponíveis.
- ℓ_i, w_i: comprimento e largura, respectivamente, do item i.
- v_i: valor do item i.
- L, W: Comprimento e largura, respectivamente, do recipiente.

Nas subseções 3.1 e 3.2 são apresentados dois modelos de Programação Linear Inteira Mista para o problema. No primeiro (Subseção 3.1), considera-se um sistema de coordenadas contínuo para o posicionamento das peças, por isso o modelo está sendo denominado *Modelo contínuo*. No segundo (Subseção 3.2), a área para o posicionamento é discretizada (*Modelo discreto*). Destaca-se que a discretização utilizada garante a obtenção de solução ótima com o mesmo valor de solução do modelo com área contínua. O primeiro modelo está baseado no modelo de Chen et al. (1995), proposto para o caso tridimensional. O segundo modelo avaliado foi proposto por Beasley (1985b).

3.1. Modelo Contínuo

Sejam p_i, $i \in \mathcal{B}$, variáveis binárias definidas de forma que

$$p_i = \begin{cases} 1, & \text{se o item } i \text{ é posicionado no recipiente} \\ 0, & \text{caso contrário.} \end{cases}$$

Para o posicionamento dos itens, pode-se definir um sistema de coordenadas ortogonais (com eixo horizontal e vertical) cuja origem é o canto inferior esquerdo do recipiente. Assim, pode-se decidir, para cada par de itens, a posição relativa entre eles ao longo de cada eixo. Para isso, em relação ao eixo horizontal, se definem as variáveis binárias a_{ij} e b_{ij}, $i, j \in \mathcal{B} \times \mathcal{B} : i < j$ de forma que

$$a_{ij}(b_{ij}) = \begin{cases} 1, & \text{se o item } i \text{ é posicionado antes (depois) do item } j \text{ no eixo horizontal} \\ 0, & \text{caso contrário.} \end{cases}$$

De forma análoga, pode-se definir as variáveis binárias c_{ij} e d_{ij}, $i, j \in \mathcal{B} \times \mathcal{B} : i < j$, relacionadas ao eixo vertical, de forma que

$$c_{ij}(d_{ij}) = \begin{cases} 1, & \text{se o item } i \text{ é posicionado antes (depois) do item } j \text{ no eixo vertical} \\ 0, & \text{caso contrário.} \end{cases}$$

Por fim, é necessário posicionar os itens dentro do recipiente, ou seja, definir uma posição (x_i, y_i), $i \in \mathcal{B}$, para a extremidade inferior direita de cada item no sistema de coordenadas definido. Ou seja, $x_i \geq 0$ e $y_i \geq 0$ definem a posição no eixo horizontal e vertical do item i, respectivamente.

Com isso, pode-se definir o modelo (1)–(10).

$$\text{Maximizar} \sum_{i \in \mathcal{B}} v_i p_i \tag{1}$$

sujeito a:

$$x_i + \ell_i \leq x_j + (1 - a_{ij})L \qquad i, j \in \mathcal{B} \times \mathcal{B} : i < j, \tag{2}$$
$$x_j + \ell_j \leq x_i + (1 - b_{ij})L \qquad i, j \in \mathcal{B} \times \mathcal{B} : i < j, \tag{3}$$
$$y_i + w_i \leq x_j + (1 - c_{ij})W \qquad i, j \in \mathcal{B} \times \mathcal{B} : i < j, \tag{4}$$
$$y_j + w_i \leq x_i + (1 - d_{ij})W \qquad i, j \in \mathcal{B} \times \mathcal{B} : i < j, \tag{5}$$
$$a_{ij} + b_{ij} + c_{ij} + d_{ij} \geq p_i + p_j - 1 \qquad i, j \in \mathcal{B} \times \mathcal{B} : i < j, \tag{6}$$
$$0 \leq x_i \leq L - \ell_i \qquad i \in \mathcal{B}, \tag{7}$$
$$0 \leq y_i \leq W - w_i \qquad i \in \mathcal{B}, \tag{8}$$
$$p_i \in \{0, 1\} \qquad i \in \mathcal{B}, \tag{9}$$
$$a_{ij}, b_{ij}, c_{ij}, d_{ij} \in \{0, 1\} \qquad i, j \in \mathcal{B} \times \mathcal{B} : i < j. \tag{10}$$

A função objetivo (1) maximiza o valor dos itens empacotados. Para garantir a factibilidade do empacotamento, os itens posicionados não devem se sobrepor no eixo horizontal (restrições (2) e (3)) nem no eixo vertical (restrições (4) e (5)). As restrições (6) controlam a ativação das restrições (2)–(5) para cada par de itens que é escolhido para ser posicionado. Além disso, o domínio das variáveis é definido pelas restrições (7)–(10). Note que (7) e (8) também garantem que os itens são posicionados de forma a ficarem inteiramente contidos no recipiente.

3.2. Modelo Discreto

Para o segundo modelo, considera-se que o recipiente é discretizado em um conjunto de pontos $\mathcal{K}$. Para o caso de itens e recipientes retangulares há discretizações que garantem a otimalidade da solução. Diferentes discretizações são discutidas e avaliadas em Côté e Iori (2018) e em de Almeida Cunha et al. (2020) aqui foram utilizados os *padrões normais* definidos por Herz (1972) e Christofides e Whitlock (1977), com a melhoria proposta por Beasley (1985a), que pode ser definida a partir de (11) e (12).

$$\mathcal{K}_\ell = \left\{ x = \sum_{j \in \mathcal{B}} \ell_j \alpha_j : 0 \leq x \leq L, \alpha_j \in \{0,1\}, j \in \mathcal{K} \right\}, \tag{11}$$

$$\mathcal{K}_w = \left\{ y = \sum_{j \in \mathcal{B}} w_j \alpha_j : 0 \leq y \leq W, \alpha_j \in \{0,1\}, j \in \mathcal{K} \right\}. \tag{12}$$

Com isso, o recipiente é discretizado nos seguintes pontos $\mathcal{K}_0 = \mathcal{K}_\ell \times \mathcal{K}_w$ e cada item pode ser posicionado em um ponto do conjunto $\mathcal{K} \subset \mathcal{K}_0$ definido por (13) em que ℓ_{min} e w_{min} são respectivamente o menor comprimento e a menor largura entre todos os itens.

$$\mathcal{K} = \{(x,y) \in \mathcal{K}_0 : 0 \leq x \leq L - \ell_{min} \text{ e } 0 \leq y \leq W - w_{min}.\} \tag{13}$$

Assim, podem-se definir variáveis binárias z_{ik}, $i \in \mathcal{B}$, $k \in \mathcal{K}$, de forma que

$$z_{ik} = \begin{cases} 1, & \text{se a extremidade inferior esquerda do item } i \text{ é posicionado no ponto } k \\ 0, & \text{caso contrário.} \end{cases}$$

Note que $z_{ik} = 0$, caso posicionar o item i no ponto $k \in \mathcal{K}$ implique em o item não esteja completamento dentro do recipiente.

Considere também um parâmetro auxiliar binário g_{iuk}, $i \in \mathcal{B}$, $u \in \mathcal{K}_0$, $k \in \mathcal{K}$, tal que

$$g_{iuk} = \begin{cases} 1, & \text{se o item } i, \text{ ao ser posicionado no ponto } k, \text{ ocupa o ponto } u \\ 0, & \text{caso contrário.} \end{cases}$$

Pode-se, assim, definir o modelo (14)–(17).

$$\text{Maximizar} \sum_{i \in \mathcal{B}} \sum_{k \in \mathcal{K}} v_i z_{ik} \tag{14}$$

sujeito a:

$$\sum_{i \in \mathcal{B}} \sum_{k \in \mathcal{K}} g_{iuk} z_{ik} \leq 1 \qquad u \in \mathcal{K}_0, \tag{15}$$

$$\sum_{k \in \mathcal{K}} z_{ik} \leq 1 \qquad i \in \mathcal{B}, \tag{16}$$

$$z_{ik} \in \{0,1\} \qquad i \in \mathcal{B}, k \in \mathcal{K}. \tag{17}$$

A função objetivo (14) é análoga à função (1), maximiza o valor dos itens empacotados. As restrições (15) evitam as sobreposições entre os pares de itens. As restrições (16) garantem que cada item é posicionado no máximo uma vez. Por fim, o domínio das variáveis é dado por (17).

3.2.1. Potenciais Melhorias

O modelo discreto originalmente proposto por Beasley (1985b) agrupa os itens por tipo e cada tipo de item passa a ter um número de cópias disponíveis, reduzindo o número de variáveis para casos em que há poucos tipos de itens. Além disso, há outras estratégias de discretização do recipiente (Côté e Iori, 2018). Como o objetivo principal deste trabalho é comparar pacotes computacionais, optou-se por utilizar as versões descritas na tentativa de fornecer um limitante inferior para o desempenho. Ainda assim, os experimentos corroboram a superioridade do modelo discreto (Seção 4). Em geral, espera-se que a diminuição no número de pontos de discretização melhore o desempenho do pacote computacional, algo já explorado na literatura (de Almeida Cunha et al., 2020).

3.3. Comparação entre os Modelos

Os modelos apresentados nas subseções 3.1 e 3.2 foram escolhidos para os experimentos por possuírem características distintas em termos conceituais de Programação Inteira. Além disso, são modelos utilizados na literatura que podem ser implementados diretamente nos pacotes computacionais de Programação Linear Inteira Mista sem a necessidade do uso de técnicas mais sofisticadas como estratégias de decomposição.

O modelo contínuo (1)–(10) possui tamanho (número de variáveis e restrições) polinomial no número de itens. O número de variáveis binárias é $O(|\mathcal{B}^2|)$ e o número de variáveis contínuas é $O(|\mathcal{B}|)$. O número de restrições é $O(|\mathcal{B}|^2)$. Por outro lado, o modelo discreto ((14)–(17)) é pseudo-polinomial, tanto o seu número de variáveis (todas binárias) quanto de restrições é proporcional às dimensões do recipiente. A expectativa de desempenho é que o modelo contínuo seja capaz de fornecer boas soluções factíveis, mas com dificuldade de prova de otimalidade devido à baixa qualidade de sua relaxação linear, que fornece o limitante dual. O que foi confirmado pelos experimentos (Seção 4).

4. Experimentos Computacionais

O objetivo dos experimentos computacionais é comparar empiricamente os resolvedores de problemas de Programação Linear Inteira Mista dos pacotes CBC 2.10, SCIP 8.0, CPLEX 22.1 e Gurobi 10.0, considerados os dois modelos apresentados nas subseções 3.1 e 3.2, respectivamente. Em relação ao *hardware*, foi utilizado um computador *desktop*, com 16 GB de memória RAM e processador Intel®Core™ i7-7700CPU @ 3.6GHz. O desenvolvimento e execução foram realizados no sistema operacional Ubuntu 22.04.

Os pacotes CPLEX e Gurobi foram escolhidos pois são frequentemente utilizados na literatura para diversos problemas (não apenas de empacotamento), são pacotes computacionais comerciais de alto desempenho e possuem uma versão acadêmica gratuita. Contudo, não possuem seu código-fonte aberto, colocando um desafio adicional para a comunidade científica. CBC e SCIP, por outro lado, possuem código-fonte livre e são potenciais alternativas de baixo custo para uso na indústria. CBC e SCIP foram escolhidos pois suas licenças permitem o uso inclusive para aplicações comerciais e ambos possuem compatibilidade com bibliotecas de modelagem como PuLP (Mitchell et al., 2011) e JuMP (Lubin et al., 2023). Assim sendo, foram escolhidos dois pacotes que implicariam em um investimento para sua utilização e dois pacotes de uso gratuito.

Para a comparação foram utilizados os conjuntos de instâncias apresentadas por Beasley (1985a) (*gcut*) e Beasley (1985b) (*ngcut*). Embora sejam instâncias propostas há décadas, ainda são utilizadas como *benchmark* e com instâncias em aberto, vide, por exemplo, Baldacci e Boschetti (2007) e Iori et al. (2021). Foram coletadas as métricas de limitante superior, limitante inferior, limitante do nó raiz da árvore de *branch-and-cut* e tempo computacional. Para cada instância, foi utilizado um tempo máximo de 3600s com uma *thread*, todos os demais parâmetros foram mantidos na configuração padrão do respectivo pacote. Em todos os casos apresentados, o

gap foi calculado como a diferença entre os limitantes superior e inferior, em relação ao limitante superior em porcentagem.

As instâncias variam nas dimensões do recipiente e dos itens. O número de itens varia entre 10 e 50 no conjunto *gcut* e os recipientes são quadrados com comprimentos variando entre 250 e 3000. As instâncias *ngcut* variam de 7 a 22 itens com recipientes quadrados e retangulares com dimensões entre $(10, 10)$ e $(30, 30)$. O detalhamento de cada instância pode ser visto nas tabelas 2, 3, 4 ou 5.

Em média, o Gurobi foi mais eficaz em reduzir o *gap* de otimalidade para os conjuntos *gcut* e *ngcut* (Tabela 1), utilizando o modelo (1)–(10). O *gap* médio do Gurobi para as instâncias do *gcut* foi de aproximadamente 20%, do CPLEX foi de 28%, seguido respectivamente pelo CBC e pelo SCIP com 35% e 39%. Note que, para o conjunto *ngcut* há uma inversão na ordem das medidas entre o CBC e o SCIP, com o Gurobi mantendo o desempenho superior em relação ao *gap* médio. Em relação ao *gap* mediano, o Gurobi foi superior ao CPLEX e o SCIP superior ao CBC.

Tabela 1: Visão geral da comparação entre os pacotes CBC, SCIP, CPLEX e Gurobi. São reportadas a média e a mediana do *gap* e do tempo computacional (em segundos) além no número de soluções ótimas encontradas (#Opt) e do número de instâncias para a qual pelo menos uma solução factível foi encontrada (#Fact). Os resultados são separados por modelo e por conjunto de instâncias (*gcut* e *ngcut*).

Modelo contínuo (1)–(10)						
			CBC	**SCIP**	**CPLEX**	**Gurobi**
gcut	**Gap**	**Média**	35,37	39,96	28,23	20,32
		Mediana	44,77	42,66	26,21	0,00
	Tempo	**Média**	2129,78	2231,57	1959,64	1691,10
		Mediana	3600,00	3600,00	3600,00	1203,60
	#Opt		6	5	6	8
	#Fact.		13	13	13	13
ngcut	**Gap**	**Média**	17,54	9,03	2,18	0,00
		Mediana	21,23	0,00	0,00	0,00
	Tempo	**Média**	2256,31	1424,62	438,67	11,48
		Mediana	3600,00	1024,74	52,01	2,82
	#Opt		5	9	11	12
	#Fact.		12	12	12	12
Modelo discreto (14)–(17)						
			CBC	**SCIP**	**CPLEX**	**Gurobi**
gcut	**Gap**	**Média**	0,00	0,00	0,00	0,00
		Mediana	0,00	0,00	0,00	0,00
	Tempo	**Média**	1715,93	1396,79	1163,82	1117,58
		Mediana	351,62	23,03	32,03	5,13
	#Opt		7	8	9	9
	#Fact.		7	8	9	9
ngcut	**Gap**	**Média**	0,00	0,00	0,00	0,00
		Mediana	0,00	0,00	0,00	0,00
	Tempo	**Média**	222,92	144,55	66,37	12,33
		Mediana	2,22	3,21	0,27	0,61
	#Opt		12	12	12	12
	#Fact.		12	12	12	12

Fonte: Elaboração própria.

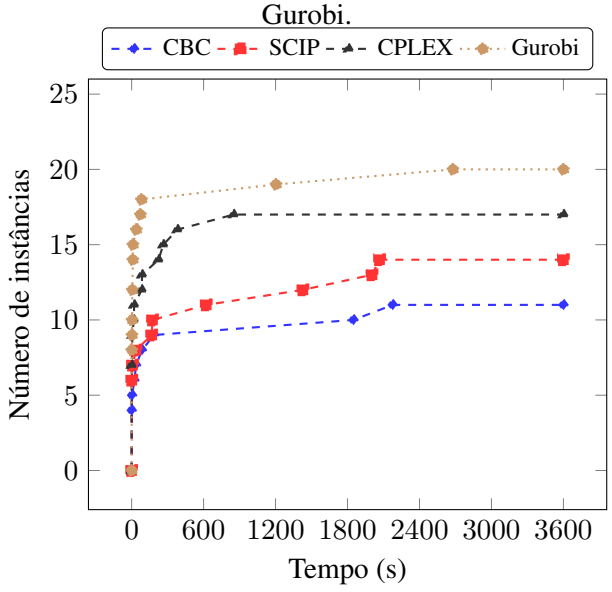
Em relação ao tempo computacional, o Gurobi foi o mais rápido para ambos os conjuntos de instância, na média (Tabela 1), utilizando o modelo (1)–(10). Para o conjunto *gcut*, além de ter utilizado aproximadamente 86% do tempo computacional do CPLEX, 80% do tempo do CBC e 75% do tempo do SCIP, o Gurobi em média, também foi capaz de provar a otimalidade de oito soluções, contra seis do CBC e CPLEX e cinco do SCIP. Para o conjunto *ngcut* a economia de

tempo foi maior, com o Gurobi utilizando menos de um décimo do tempo do CPLEX, em média. O CPLEX, por sua vez, utilizou cerca de um terço do tempo do SCIP e um quinto do tempo do CBC. Para o *ngcut*, o Gurobi foi o único capaz de provar a otimalidade da solução para todas as instâncias.

Para instâncias com menos de 30 itens, todos os pacotes avaliados conseguiram provar a otimalidade para todas as instâncias, utilizando o modelo (14)–(17) (Tabela 1). A aplicabilidade do modelo (14)–(17) é limitada pela alta demanda de memória (quando comparado com o modelo (1)–(10)), mas ainda assim é possível diferenciar o desempenho de alguns dos pacotes computacionais. Para as instâncias em que foi possível encontrar alguma solução, também foi possível provar a otimalidade (resultando em gap médio nulo na tabela). No entanto, CPLEX e Gurobi conseguiram obter soluções para nove instâncias do *gcut* contra oito e sete do SCIP e CBC, respectivamente. Além disso, os tempos computacionais utilizados pelo CPLEX e pelo Gurobi foram bastante próximos ambos inferiores ao do SCIP que, por sua vez, foi inferior ao do CBC. O desempenho relativo dos pacotes se mantém ao se observar as instâncias do *ngcut*. Como um dos principais desafios relacionados à utilização do modelo (14)–(17) é o seu tamanho pseudo-polinomial, especula-se que os pacotes com uso mais eficiente de memória e melhores técnicas de preprocessamento, apresentem um melhor desempenho.

Apesar da diferença de desempenho entre os pacotes computacionais, a maioria das instâncias resolvidas tiveram o seu ótimo comprovado nos primeiros 10 minutos (Figura 1). As curvas para cada pacote apresentadas na Figura 1 indicam que passados 600 segundos, inicia-se estabilização no número de instâncias cujo ótimo será comprovado dentro do tempo limite de 3600 segundos. Embora cada pacote atinja um nível diferente, o tempo computacional em que esse nível é atingindo foi similar. Além disso, pode-se observar a performance superior do Gurobi nos conjuntos de instâncias utilizados, seguido pelo CPLEX, SCIP e CBC, respectivamente.

Figura 1: As curvas apresentam o número de instâncias resolvidos até o tempo indicado no eixo horizontal, em segundos, para os pacotes CBC, SCIP, CPLEX e Gurobi.



Fonte: Elaboração própria.

Em geral, o conjunto de instâncias *gcut* foi mais desafiador para os pacotes computacionais do que o *ngcut*. Embora não seja o objetivo principal deste trabalho comparar os conjuntos de instâncias utilizados (mas o desempenho dos pacotes computacionais), pode-se observar na Tabela 1 que o desempenho médio e mediano dos pacotes computacionais foi melhor nas instâncias *ngcut*. Potencialmente, isso possa ser explicado pois nenhuma das instâncias do *ngcut* possui mais de 25

itens, enquanto que sete (das 13) instâncias do *gcut* possuem pelo menos 30 itens. Outra diferença é em relação ao tamanho do recipiente, o maior recipiente do conjunto *ngcut* possui dimensões $(30, 30)$ enquanto que o menor contêiner do *gcut* possui dimensão $(250, 250)$. Os detalhes de cada instância podem ser observados nas Tabelas 2, 3, 4 ou 5.

Para todas as instâncias utilizadas, o Gurobi foi capaz de obter o melhor limitante no nó raiz da árvore de *branch-and-cut* quando comparado com CBC, SCIP e CPLEX. Entre CBC, SCIP e CPLEX, o CPLEX, apresenta os melhores limitantes no nó raiz com maior frequência, mas há casos em que CBC ou SCIP fornecem os melhores resultados. Embora o limitante da raiz não seja determinante para um bom resultado final, a qualidade de tal limitante impacta toda a exploração da árvore de *branch-and-cut*, por isso, escolheu-se também observá-lo.

4.1. Aspectos Gerenciais

Com os resultados apresentados, é possível traçar alguns indicativos do ponto de vista gerencial para a situação em que se deseja avaliar a aplicabilidade de um pacote computacional e da escolha de qual pacote utilizar. Para instâncias com até 30 itens qualquer um dos pacotes computacionais avaliados (CBC, SCIP, CPLEX e Gurobi) foram capazes de produzir soluções ótimas com tempo mediano inferior a 10 segundos (vide Tabela 1). Portanto, para instâncias dessa magnitude, não se justificaria o investimento nas alternativas comerciais.

Mesmo para instâncias de tamanhos maiores pode ser difícil justificar o investimento. A diferença em relação ao *gap* médio é de aproximadamente 8 pontos percentuais a favor do Gurobi, contra o CPLEX, vide Tabela 1, para o modelo contínuo. Contudo, ao se comparar as melhores soluções factíveis das Tabelas 2 e 3 a diferença entre os pacotes não ultrapassa 5%, em média (e é equivalente em mediana), como resumido na Tabela 6. Para contextos práticos em que a qualidade da solução é prioritária em relação à prova de otimalidade (e ao tempo computacional), o investimento em pacotes comerciais implicaria em uma melhora de 2% em média, se comparado à melhor alternativa gratuita.

Ainda do ponto de vista de gerência, o ganho marginal de se investir em mais tempo computacional (ou em poder de processamento) dificilmente é justificado. Como apresentado nos perfis da Figura 1, o *plateau* de soluções obtidas aparece em tempos relativamente curtos. Isso é esperado, devido à NP-dificuldade do problema, implicando em um custo exponencial em função do tamanho das instâncias. Para casos de instâncias maiores, técnicas mais sofisticadas podem ser necessárias como o fornecimento de soluções iniciais para o pacote computacional, estratégias de decomposição e/ou algoritmos especializados (exatos ou heurísticos).

5. Conclusões

A utilização de pacotes computacionais (*solvers*) para a solução de problemas de Programação Linear Inteira Mista é frequente tanto na academia quanto na indústria. Embora, academicamente, as propostas de modelos e métodos de solução sejam feitas frequentemente em um contexto de prova de conceito, industrialmente a escolha das ferramentas de implementação, para o desenvolvimento dessas soluções é mais relevante. Um dos desafios particularmente relevantes para a indústria, portanto, é a avaliação dos pacotes computacionais. Isso é feito aqui para problemas de empacotamento de retângulos considerando os pacotes CBC, SCIP, CPLEX e Gurobi.

Os resultados indicam que, para instâncias da literatura, o Gurobi é particularmente mais eficiente que os demais pacotes avaliados quando se trata de obter soluções comprovadamente ótimas. Contudo, a qualidade das soluções factíveis obtidas dentro do tempo limite experimental de uma hora atinge 98% da solução do Gurobi, em média, se utilizada a melhor alternativa gratuita (SCIP). Em mediana, não se identifica diferença entre os desempenhos dos pacotes computacionais.

Destaca-se que, assim como outros trabalhos na literatura comparando pacotes computacionais, a generalização dos resultados para outros conjuntos de instâncias ou para outras versões dos

Tabela 2: Resultados para cada instância para os pacotes CBC e SCIP considerando o modelo (1)–(10). São reportados o limitante da raiz da árvore de *branch-and-cut* z^r, o limitante inferior (LB) e superior (UB) e *gap* ($100(UB - LB)/UB$) ao final do tempo de execução t (em segundos). TL indica que o tempo limite (3600s) foi atingido. Também são mostradas características das instâncias: o número de itens n e o comprimento (L) e largura do recipiente (W).

			CBC					**SCIP**				
Id	n	(L, W)	z^r	LB	UB	**Gap**	t **(s)**	z^r	LB	UB	**Gap**	t **(s)**
gcut01	**10**	**(250, 250)**	53015,60	48368,00	48368,00	0,00	0,51	65030,00	48368,00	48368,00	0,00	0,30
gcut02	**20**	**(250, 250)**	129422,00	59798,00	59798,00	0,00	2176,82	134782,80	59798,00	104278,00	42,66	TL
gcut03	**30**	**(250, 250)**	207270,00	59839,00	160921,60	62,81	TL	233885,50	60241,00	186570,58	67,71	TL
gcut04	**50**	**(250, 250)**	293806,00	61380,00	285480,88	78,50	TL	356864,90	61305,00	307426,00	80,06	TL
gcut05	**10**	**(500, 500)**	275426,00	195582,00	195582,00	0,00	0,96	305882,00	195582,00	195582,00	0,00	0,55
gcut06	**20**	**(500, 500)**	525675,00	236305,00	236305,00	0,00	186,98	592596,70	236305,00	236305,00	0,00	166,29
gcut07	**30**	**(500, 500)**	731043,00	240143,00	512228,00	53,12	TL	861897,40	238974,00	608889,00	60,75	TL
gcut08	**50**	**(500, 500)**	1269980,00	241121,00	1250930,53	80,72	TL	1538853,00	243606,00	1250471,49	80,52	TL
gcut09	**10**	**(1000, 1000)**	1129320,00	939600,00	939600,00	0,00	36,21	1270354,00	939600,00	939600,00	0,00	6,96
gcut10	**20**	**(1000, 1000)**	2037380,00	937349,00	937349,00	0,00	85,66	2263104,00	937349,00	937349,00	0,00	36,37
gcut11	**30**	**(1000, 1000)**	3086130,00	964139,00	2467608,39	60,93	TL	3912621,00	957457,00	2962792,92	67,68	TL
gcut12	**50**	**(1000, 1000)**	4679220,00	917009,00	4360081,21	78,97	TL	5692618,00	957704,00	4489791,60	78,67	TL
gcut13	**32**	**(3000, 3000)**	14315500,00	7906040,00	14315508,00	44,77	TL	14315510,00	8386899,00	14315508,00	41,41	TL
ngcut01	**10**	**(10, 10)**	192,00	164,00	164,00	0,00	3,16	195,50	164,00	164,00	0,00	0,95
ngcut02	**17**	**(10, 10)**	434,00	230,00	325,86	29,42	TL	448,00	230,00	230,00	0,00	2003,12
ngcut03	**21**	**(10, 10)**	411,00	247,00	389,11	36,52	TL	446,50	247,00	392,00	36,99	TL
ngcut04	**7**	**(15, 10)**	302,00	268,00	268,00	0,00	0,86	302,00	268,00	268,00	0,00	0,29
ngcut05	**14**	**(15, 10)**	615,00	358,00	487,00	26,49	TL	581,00	358,00	358,00	0,00	173,70
ngcut06	**15**	**(15, 10)**	479,00	289,00	397,00	27,20	TL	495,00	289,00	289,00	0,00	1427,96
ngcut07	**8**	**(20, 20)**	430,00	430,00	430,00	0,00	0,02	430,00	430,00	430,00	0,00	0,05
ngcut08	**13**	**(20, 20)**	1411,00	834,00	1226,00	31,97	TL	1411,00	834,00	1116,00	25,27	TL
ngcut09	**18**	**(20, 20)**	1454,00	924,00	924,00	0,00	1849,02	1454,00	924,00	924,00	0,00	621,52
ngcut10	**13**	**(30, 30)**	2436,00	1452,00	1452,00	0,00	22,62	2201,00	1452,00	1452,00	0,00	0,91
ngcut11	**15**	**(30, 30)**	2540,00	1688,00	2008,66	15,96	TL	2540,00	1688,00	1688,00	0,00	2066,90
ngcut12	**22**	**(30, 30)**	3850,00	1865,00	3268,93	42,95	TL	3850,00	1865,00	3457,47	46,06	TL
Média			**1149909,66**	**512616,88**	**1031244,05**	**26,81**	**2190,51**	**1262334,13**	**533277,48**	**1063748,00**	**25,11**	**1844,23**
Mediana			**53015,60**	**48368,00**	**48368,00**	**26,49**	**TL**	**65030,00**	**48368,00**	**48368,00**	**0,00**	**2003,12**

Fonte: Elaboração própria.

Tabela 3: Resultados para cada instância para os pacotes CPLEX e Gurobi considerando o modelo (1)–(10). São reportados o limitante da raiz da árvore de *branch-and-cut* z^r, o limitante inferior (LB) e superior (UB) e o *gap* $(100(UB - LB)/UB)$ ao final do tempo de execução t (em segundos). TL indica que o tempo limite (3600s) foi atingido. Também são mostradas características das instâncias: o número de itens n e o comprimento (L) e largura do recipiente (W).

			CPLEX					**Gurobi**				
Id	n	(L, W)	z^r	LB	UB	**Gap**	t **(s)**	z^r	LB	UB	**Gap**	t **(s)**
gcut01	**10**	**(250, 250)**	72707,20	48368,00	48368,00	0,00	0,03	83712,00	48368,00	48368,00	0,00	0,05
gcut02	**20**	**(250, 250)**	134725,00	59798,00	59798,00	0,00	263,64	168315,00	59798,00	59798,00	0,00	81,71
gcut03	**30**	**(250, 250)**	226074,50	61275,00	94889,00	35,42	TL	254945,00	61275,00	61277,00	0,00	2678,48
gcut04	**50**	**(250, 250)**	334839,69	60942,00	232986,00	73,84	TL	369060,00	61380,00	219648,00	72,06	TL
gcut05	**10**	**(500, 500)**	309472,33	195582,00	195582,00	0,00	0,22	444908,00	195582,00	195582,00	0,00	0,11
gcut06	**20**	**(500, 500)**	602060,25	236305,00	236305,00	0,00	5,83	665489,00	236305,00	236305,00	0,00	10,00
gcut07	**30**	**(500, 500)**	780347,00	240143,00	325445,41	26,21	TL	980698,00	240143,00	240143,00	0,00	1203,60
gcut08	**50**	**(500, 500)**	1414956,33	240389,00	1125089,00	78,63	TL	1584546,00	243838,00	1095054,65	77,73	TL
gcut09	**10**	**(1000, 1000)**	1258186,00	939600,00	939600,00	0,00	0,48	1443517,00	939600,00	939600,00	0,00	0,54
gcut10	**20**	**(1000, 1000)**	2491807,50	937349,00	937349,00	0,00	5,14	2657938,00	937349,00	937349,00	0,00	9,83
gcut11	**30**	**(1000, 1000)**	3824738,72	964139,00	1537906,92	37,31	TL	4315727,00	969709,00	1143969,00	15,23	TL
gcut12	**50**	**(1000, 1000)**	5333453,60	964035,00	3889913,55	75,22	TL	6080949,00	976877,00	2382598,00	59,00	TL
gcut13	**32**	**(3000, 3000)**	14315500,00	8538659,00	14315508,00	40,35	TL	14315510,00	8577752,00	14315508,00	40,08	TL
ngcut01	**10**	**(10, 10)**	214,67	164,00	164,00	0,00	0,20	305,00	164,00	164,00	0,00	0,12
ngcut02	**17**	**(10, 10)**	448,00	230,00	230,00	0,00	86,38	490,00	230,00	230,00	0,00	5,50
ngcut03	**21**	**(10, 10)**	425,67	247,00	247,00	0,00	850,78	485,00	247,00	247,00	0,00	72,73
ngcut04	**7**	**(15, 10)**	302,00	268,00	268,00	0,00	0,02	302,00	268,00	268,00	0,00	0,01
ngcut05	**14**	**(15, 10)**	615,00	358,00	358,00	0,00	20,35	615,00	358,00	358,00	0,00	0,62
ngcut06	**15**	**(15, 10)**	511,00	289,00	289,00	0,00	83,67	511,00	289,00	289,00	0,00	5,15
ngcut07	**8**	**(20, 20)**	0,00	430,00	430,00	0,00	0,00	430,00	430,00	430,00	0,00	0,00
ngcut08	**13**	**(20, 20)**	1411,00	834,00	834,00	0,00	381,92	1411,00	834,00	834,00	0,00	9,21
ngcut09	**18**	**(20, 20)**	1454,00	924,00	924,00	0,00	17,60	1921,00	924,00	924,00	0,00	2,63
ngcut10	**13**	**(30, 30)**	2436,00	1452,00	1452,00	0,00	0,14	2436,00	1452,00	1452,00	0,00	0,06
ngcut11	**15**	**(30, 30)**	2540,00	1688,00	1688,00	0,00	223,01	2911,00	1688,00	1688,00	0,00	3,01
ngcut12	**22**	**(30, 30)**	3850,00	1865,00	2526,19	26,17	TL	4357,00	1865,00	1865,00	0,00	38,76
Média			**1244523,02**	**539813,32**	**957926,00**	**15,73**	**1229,58**	**1335259,52**	**542269,00**	**875357,95**	**10,56**	**884,88**
Mediana			**72707,20**	**48368,00**	**48368,00**	**0,00**	**86,38**	**83712,00**	**48368,00**	**48368,00**	**0,00**	**9,21**

Fonte: Elaboração própria.

Tabela 4: Resultados para cada instância para os pacotes CBC e SCIP considerando o modelo (14)–(17). São reportados o limitante da raiz da árvore de *branch-and-cut* z^r, o limitante inferior (LB) e superior (UB) ao final do tempo de execução t (em segundos). TL indica que o tempo limite (3600s) foi atingido e '–' indica que nenhuma solução viável foi encontrada. Também são mostradas características das instâncias: o número de itens n e o comprimento (L) e largura do recipiente (W).

			CBC				**SCIP**			
Id	n	(L, W)	z^r	LB	UB	t **(s)**	z^r	LB	UB	t **(s)**
gcut01	**10**	**(250, 250)**	48368,00	48368,00	48368,00	0,46	48368,00	48368,00	48368,00	0,06
gcut02	**20**	**(250, 250)**	61148,00	59798,00	59798,00	351,62	60582,95	59798,00	59798,00	12,62
gcut03	**30**	**(250, 250)**	–	–	–	TL	61537,50	61275,00	61275,00	112,69
gcut04	**50**	**(250, 250)**	–	–	–	TL	–	–	–	TL
gcut05	**10**	**(500, 500)**	221068,00	195582,00	195582,00	16,96	206584,70	195582,00	195582,00	0,62
gcut06	**20**	**(500, 500)**	242543,00	236305,00	236305,00	53,49	240134,40	236305,00	236305,00	5,87
gcut07	**30**	**(500, 500)**	243670,00	240143,00	240143,00	246,12	241611,10	240143,00	240143,00	23,03
gcut08	**50**	**(500, 500)**	–	–	–	TL	–	–	–	TL
gcut09	**10**	**(1000, 1000)**	946845,00	939600,00	939600,00	0,57	939600,00	939600,00	939600,00	0,14
gcut10	**20**	**(1000, 1000)**	967987,00	937349,00	937349,00	37,86	948190,80	937349,00	937349,00	3,28
gcut11	**30**	**(1000, 1000)**	–	–	–	TL	–	–	–	TL
gcut12	**50**	**(1000, 1000)**	–	–	–	TL	–	–	–	TL
gcut13	**32**	**(3000, 3000)**	–	–	–	TL	–	–	–	TL
ngcut01	**10**	**(10, 10)**	184,01	164,00	164,00	0,74	182,31	164,00	164,00	0,96
ngcut02	**17**	**(10, 10)**	253,27	230,00	230,00	3,70	251,84	230,00	230,00	5,68
ngcut03	**21**	**(10, 10)**	255,50	247,00	247,00	0,69	253,33	247,00	247,00	4,04
ngcut04	**7**	**(15, 10)**	261,00	268,00	268,00	0,03	268,00	268,00	268,00	0,06
ngcut05	**14**	**(15, 10)**	343,00	358,00	358,00	0,06	358,00	358,00	358,00	0,14
ngcut06	**15**	**(15, 10)**	313,14	289,00	289,00	8,27	302,69	289,00	289,00	7,30
ngcut07	**8**	**(20, 20)**	430,00	430,00	430,00	0,04	430,00	430,00	430,00	0,19
ngcut08	**13**	**(20, 20)**	906,00	834,00	834,00	102,92	897,81	834,00	834,00	21,05
ngcut09	**18**	**(20, 20)**	934,99	924,00	924,00	11,22	928,67	924,00	924,00	2,38
ngcut10	**13**	**(30, 30)**	1455,35	1452,00	1452,00	0,37	1452,00	1452,00	1452,00	0,20
ngcut11	**15**	**(30, 30)**	1776,96	1688,00	1688,00	695,28	1776,96	1688,00	1688,00	33,69
ngcut12	**22**	**(30, 30)**	1945,68	1865,00	1865,00	1851,69	1942,77	1865,00	1865,00	1658,91
Média			**144246,73**	**140310,21**	**140310,21**	**178,00**	**137782,69**	**136358,45**	**136358,45**	**94,65**
Mediana			**1455,35**	**1452,00**	**1452,00**	**8,27**	**1614,48**	**1570,00**	**1570,00**	**3,66**

Fonte: Elaboração própria.

Tabela 5: Resultados para cada instância para os pacotes CPLEX e Gurobi considerando o modelo (14)–(17). São reportados o limitante da raiz da árvore de *branch-and-cut* z^r, o limitante inferior (LB) e superior (UB) ao final do tempo de execução t (em segundos). TL indica que o tempo limite (3600s) foi atingido e '–' indica que nenhuma solução viável foi encontrada. Também são mostradas características das instâncias: o número de itens n e o comprimento (L) e largura do recipiente (W).

			CPLEX				**Gurobi**			
Id	n	(L, W)	z^r	LB	UB	t **(s)**	z^r	LB	UB	t **(s)**
gcut01	**10**	**(250, 250)**	48368,00	48368,00	48368,00	0,03	50960,00	48368,00	48368,00	0,02
gcut02	**20**	**(250, 250)**	60890,12	59798,00	59798,00	32,03	61240,39	59798,00	59798,00	4,50
gcut03	**30**	**(250, 250)**	61761,74	61275,00	61275,00	111,67	61785,33	61275,00	61275,00	29,46
gcut04	**50**	**(250, 250)**	–	–	–	TL	–	–	–	TL
gcut05	**10**	**(500, 500)**	210200,15	195582,00	195582,00	0,28	228381,90	195582,00	195582,00	0,59
gcut06	**20**	**(500, 500)**	240881,69	236305,00	236305,00	7,68	242556,10	236305,00	236305,00	1,69
gcut07	**30**	**(500, 500)**	242308,03	240143,00	240143,00	25,15	243671,00	240143,00	240143,00	5,13
gcut08	**50**	**(500, 500)**	–	–	–	TL	–	–	–	TL
gcut09	**10**	**(1000, 1000)**	939600,00	939600,00	939600,00	0,09	947014,00	939600,00	939600,00	0,04
gcut10	**20**	**(1000, 1000)**	947275,57	937349,00	937349,00	2,75	967987,40	937349,00	937349,00	1,40
gcut11	**30**	**(1000, 1000)**	982287,72	969709,00	969709,00	549,96	983752,50	969709,00	969709,67	85,65
gcut12	**50**	**(1000, 1000)**	–	–	–	TL	–	–	–	TL
gcut13	**32**	**(3000, 3000)**	–	–	–	TL	–	–	–	TL
ngcut01	**10**	**(10, 10)**	182,00	164,00	164,00	0,08	205,84	164,00	164,00	0,13
ngcut02	**17**	**(10, 10)**	257,50	230,00	230,00	0,41	257,50	230,00	230,00	1,16
ngcut03	**21**	**(10, 10)**	252,66	247,00	247,00	0,12	255,50	247,00	247,00	0,15
ngcut04	**7**	**(15, 10)**	0,00	268,00	268,00	0,00	271,00	268,00	268,00	0,01
ngcut05	**14**	**(15, 10)**	358,00	358,00	358,00	0,01	359,00	358,00	358,00	0,01
ngcut06	**15**	**(15, 10)**	302,56	289,00	289,00	1,08	315,93	289,00	289,00	1,07
ngcut07	**8**	**(20, 20)**	430,00	430,00	430,00	0,05	430,00	430,00	430,00	0,03
ngcut08	**13**	**(20, 20)**	906,00	834,00	834,00	0,88	906,00	834,00	834,00	4,57
ngcut09	**18**	**(20, 20)**	930,16	924,00	924,00	1,55	934,99	924,00	924,00	1,50
ngcut10	**13**	**(30, 30)**	1452,00	1452,00	1452,00	0,07	1478,29	1452,00	1452,00	0,07
ngcut11	**15**	**(30, 30)**	1775,47	1688,00	1688,00	16,82	1776,96	1688,00	1688,00	39,39
ngcut12	**22**	**(30, 30)**	1942,25	1865,00	1865,00	775,38	1945,68	1865,00	1865,00	99,86
Média			**178207,70**	**176041,81**	**176041,81**	**72,67**	**180785,01**	**176041,81**	**176041,84**	**13,16**
Mediana			**1775,47**	**1688,00**	**1688,00**	**0,88**	**1776,96**	**1688,00**	**1688,00**	**1,16**

Fonte: Elaboração própria.

Tabela 6: Comparação entre as melhores soluções factíveis encontradas pelos pacotes CBC, SCIP, CPLEX e Gurobi, dentro do tempo limite de 3600s com o modelo contínuo. Os valores mostrados são normalizados pelo valor da solução obtida pelo Gurobi.

		Melhor solução normalizada			
		CBC	**SCIP**	**CPLEX**	**Gurobi**
gcut	**Média**	0,95	0,98	1,00	1,00
	Mediana	1,00	1,00	1,00	1,00
ngcut	**Média**	1,00	1,00	1,00	1,00
	Mediana	1,00	1,00	1,00	1,00

Fonte: Elaboração própria.

pacotes deve ser feita com cautela e, frequentemente, novos experimentos devem ser realizados. Ainda assim, os resultados aqui presentes podem informar praticantes e tomadores de decisão, além de permitir uma estimativa do desempenho comparativo entre os pacotes avaliados.

Por fim, como trabalhos futuros, considera-se expandir os resultados para englobar instâncias práticas atualizadas, além da realização de análise do espaço de instâncias. Também é possível incluir outros pacotes computacionais gratuitos e comerciais no estudo.

Agradecimentos. Os autores agradecem aos revisores anônimos que contribuíram para a versão final do artigo com seus comentários e sugestões durante o processo de revisão.

Referências

Achterberg, T., Koch, T., e Martin, A. MIPLIB 2003. *Operations Research Letters*, v. 34, n. 4, p. 361–372, 2006.

Alvarez-Valdés, R., Parreño, F., e Tamarit, J. M. A branch-and-cut algorithm for the pallet loading problem. *Computers & Operations Research*, v. 32, n. 11, p. 3007–3029, 2005.

Andrade, R., Birgin, E. G., Morabito, R., e Ronconi, D. P. MIP models for two-dimensional non-guillotine cutting problems with usable leftovers. *Journal of the Operational Research Society*, v. 65, n. 11, p. 1649–1663, 2014.

Atamtürk, A. e Savelsbergh, M. W. Integer-programming software systems. *Annals of operations research*, v. 140, n. 1, p. 67–124, 2005.

Baldacci, R. e Boschetti, M. A. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. *European Journal of Operational Research*, v. 183, n. 3, p. 1136–1149, 2007.

Bartz-Beielstein, T., Doerr, C., Berg, D. v. d., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., et al. Benchmarking in optimization: Best practice and open issues. *arXiv preprint arXiv:2007.03488*, v. , 2020.

Beasley, J. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, v. 36, n. 4, p. 297–306, 1985a.

Beasley, J. E. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, v. 33, n. 1, p. 49–64, 1985b.

Becker, H., Martin, M., Araujo, O., Buriol, L. S., e Morabito, R. Comparative analysis of mathematical formulations for the two-dimensional guillotine cutting problem. *International Transactions in Operational Research*, v. , 2023.

Belov, G. e Scheithauer, G. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European journal of operational research*, v. 171, n. 1, p. 85–106, 2006.

Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., et al. The scip optimization suite 8.0. *arXiv preprint arXiv:2112.08872*, v. , 2021.

Bixby, R. E., Ceria, S., McZeal, C. M., e Savelsbergh, M. W. *An updated mixed integer programming library: MIPLIB 3.0*. , Rice University, 1998.

Chen, C., Lee, S.-M., e Shen, Q. An analytical model for the container loading problem. *European Journal of operational research*, v. 80, n. 1, p. 68–76, 1995.

Christofides, N. e Whitlock, C. An algorithm for two-dimensional cutting problems. *Operations Research*, v. 25, n. 1, p. 30–44, 1977.

Côté, J.-F., Dell'Amico, M., e Iori, M. Combinatorial benders' cuts for the strip packing problem. *Operations Research*, v. 62, n. 3, p. 643–661, 2014.

Côté, J.-F., Haouari, M., e Iori, M. Combinatorial benders decomposition for the two-dimensional bin packing problem. *INFORMS Journal on Computing*, v. 33, n. 3, p. 963–978, 2021.

Côté, J.-F. e Iori, M. The meet-in-the-middle principle for cutting and packing problems. *INFORMS Journal on Computing*, v. 30, n. 4, p. 646–661, 2018.

de Almeida Cunha, J. G., De Lima, V. L., e De Queiroz, T. A. Grids for cutting and packing problems: a study in the 2d knapsack problem. *4OR*, v. 18, p. 293–339, 2020.

De Carvalho, J. V. Lp models for bin packing and cutting stock problems. *European Journal of Operational Research*, v. 141, n. 2, p. 253–273, 2002.

de Oliveira, C. R. V., Schmidt, C. E., e da Silva, A. C. L. *Análise da resolução do problema de transporte com custo fixo utilizando o Cplex e o Gurobi*, 2016. Disponível em: .

de Queiroz, T. A. e Miyazawa, F. K. Order and static stability into the strip packing problem. *Annals of Operations Research*, v. 223, n. 1, p. 137–154, 2014.

Delorme, M., Iori, M., e Martello, S. Logic based benders' decomposition for orthogonal stock cutting problems. *Computers & Operations Research*, v. 78, p. 290–298, 2017.

Dyckhoff, H. A typology of cutting and packing problems. *European journal of operational research*, v. 44, n. 2, p. 145–159, 1990.

Fasano, G. Cargo analytical integration in space engineering: A three-dimensional packing model. In: *Operational Research in Industry*p. 232–246. : Springer, 1999.

Fekete, S. P., Schepers, J., e Van der Veen, J. C. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, v. 55, n. 3, p. 569–587, 2007.

Forrest, J. e Lougee-Heimer, R. Cbc user guide. In: *Emerging theory, methods, and applications*p. 257–277. : INFORMS, 2005.

Gilmore, P. C. e Gomory, R. E. A linear programming approach to the cutting-stock problem. *Operations research*, v. 9, n. 6, p. 849–859, 1961.

Gilmore, P. C. e Gomory, R. E. Multistage cutting stock problems of two and more dimensions. *Operations research*, v. 13, n. 1, p. 94–120, 1965.

Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, v. 13, n. 3, p. 443–490, 2021.

Gurobi Optimization, L. *Gurobi optimizer reference manual*, 2021. Disponível em: `www.gurobi.com`.

Haftka, R. T. Requirements for papers focusing on new or improved global optimization algorithms. *Structural and Multidisciplinary Optimization*, v. 54, n. 1, p. 1–1, 2016.

Herz, J. Recursive computational procedure for two-dimensional stock cutting. *IBM Journal of Research and Development*, v. 16, n. 5, p. 462–469, 1972.

Hoffman, A., Mannos, M., Sokolowsky, D., e Wiegmann, N. Computational experience in solving linear programs. *Journal of the Society for Industrial and Applied Mathematics*, v. 1, n. 1, p. 17–33, 1953.

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, n. 2, p. 399–415, 2021.

Iori, M., de Lima, V. L., Martello, S., e Monaci, M. 2dpacklib: a two-dimensional cutting and packing library. *Optimization Letters*, v. p. 1–10, 2022.

Júnior, A. N., Silva, E., Francescatto, M., Rosa, C. B., e Siluk, J. The rectangular two-dimensional strip packing problem real-life practical constraints: A bibliometric overview. *Computers & Operations Research*, v. 137, p. 105521, 2022.

Junqueira, L., Morabito, R., e Sato Yamashita, D. MIP-based approaches for the container loading problem with multi-drop constraints. *Annals of Operations Research*, v. 199, p. 51–75, 2012a.

Junqueira, L., Morabito, R., e Yamashita, D. S. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, v. 39, n. 1, p. 74–85, 2012b.

Kantorovich, L. V. Mathematical methods of organizing and planning production. *Management science*, v. 6, n. 4, p. 366–422, 1960.

Kwon, B. e Lee, G. M. Spatial scheduling for large assembly blocks in shipbuilding. *Computers & Industrial Engineering*, v. 89, p. 203–212, 2015.

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.

Linderoth, J. T. e Ralphs, T. K. Noncommercial software for mixed-integer linear programming. *Integer programming: theory and practice*, v. 3, p. 253–303, 2005.

Lodi, A., Martello, S., e Monaci, M. Two-dimensional packing problems: A survey. *European journal of operational research*, v. 141, n. 2, p. 241–252, 2002.

Lubin, M., Dowson, O., Garcia, J. D., Huchette, J., Legat, B., e Vielma, J. P. Jump 1.0: recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, v. p. 1–9, 2023.

Martello, S. Two-dimensional packing problems in telecommunications. *Pesquisa Operacional*, v. 34, p. 31–38, 2014.

Martello, S. e Monaci, M. Models and algorithms for packing rectangles into the smallest square. *Computers & Operations Research*, v. 63, p. 161–171, 2015.

Meindl, B. e Templ, M. Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, v. 20, 2012.

Mitchell, S., OSullivan, M., e Dunning, I. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, v. 65, 2011.

Mittelmann, H. *Benchmarks for optimization software*, 2022. Disponível em: `http://plato.asu.edu/bench.html`.

Nickel, S., Steinhardt, C., Schlenker, H., e Burkart, W. Ibm ilog cplex optimization studio—a primer. In: *Decision Optimization with IBM ILOG CPLEX Optimization Studio: A Hands-On Introduction to Modeling with the Optimization Programming Language (OPL)*p. 9–21. : Springer, 2022.

Oliveira, J. F., Neuenfeldt Júnior, A., Silva, E., e Carravilla, M. A. A survey on heuristics for the two-dimensional rectangular strip packing problem. *Pesquisa Operacional*, v. 36, p. 197–226, 2016.

Padberg, M. Packing small boxes into a big box. *Mathematical Methods of Operations Research*, v. 52, p. 1–21, 2000.

Pisinger, D. e Sigurd, M. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, v. 19, n. 1, p. 36–51, 2007.

Silva, E. F., Toffolo, T. A. M., e Wauters, T. Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems. *Computers & Operations Research*, v. 109, p. 12–27, 2019.

Silva, L. C. d., Queiroz, T. A. d., e Toledo, F. M. B. d. Integer formulations for the integrated vehicle routing problem with two-dimensional packing constraints. *Pesquisa Operacional*, v. 42, p. e248686, 2022.

Strecker, T. e Hennig, L. Automatic layouting of personalized newspaper pages. In: *Operations Research Proceedings 2008: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), September 3-5, 2008*. University of Augsburg. Springer, 2009. p. 469–474.

Wäscher, G., Haußner, H., e Schumann, H. An improved typology of cutting and packing problems. *European journal of operational research*, v. 183, n. 3, p. 1109–1130, 2007.

Xu, Z. e Lee, C.-Y. New lower bound and exact method for the continuous berth allocation problem. *Operations Research*, v. 66, n. 3, p. 778–798, 2018.