Problema de empacotamento de retângulos: avaliação de métodos de solução baseados em *bottom-left*

Gabriel Medeiros Lopes Carneiro[1], **Pedro Belin Castellucci**[1], **Rafael de Santiago**[1]

[1] Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianólis, SC – Brasil

gabriel.mlc@grad.ufsc.br, pedro.castellucci@ufsc.br, r.santiago@ufsc.br

Resumo. *Problemas de empacotamento consistem em alocar um conjunto de itens $\mathcal{I}$ em uma caixa $\mathcal{B}$. No problema de empacotamento da mochila, foco deste trabalho, cada item é associado a um valor e busca-se uma solução que maximize a soma dos valores dos itens alocados. Este trabalho compara 40 métodos de solução criados com base na heurística bottom-left para o problema de empacotamento de retângulos. Os métodos criados são uma combinação de diferentes formas de ordenação dos itens e criação de regiões, as quais evitam as sobreposições e o domínio contínuo presentes no problema. O principal resultado foi a alta competitividade de diferentes modos de ordenação, não sendo a área a única relevante, com o perímetro obtendo os melhores resultados.*

Abstract. *Packing problems consist of allocating a set of items $\mathcal{I}$ into a box $\mathcal{B}$. In the knapsack packing problem, the focus of this work, each item is associated with a value and a solution is sought that maximizes the sum of the values of the allocated items. This work compare 40 created solution methods based on bottom-left heuristic for the rectangle packing problem. The methods created are a combination of different ways of ordering items and creating regions, which avoid superposition and continuous domain present in the problem. The main result was the high competitiveness of different ordering modes, the area not being the only relevant one, with the perimeter obtaining the best results.*

1. Introdução

Serviços de loja *online* com entrega como Amazon e Mercado Livre estão tornando-se cada vez mais presentes no dia a dia. Para tornar as entregas mais rápidas é necessário fazer uma série de estudos de logística e planejamento sobre como organizar os produtos nos estoques e nos veículos de entrega [Silva et al. 2022, Morabito Neto and Widmer 1992], muitas vezes sendo necessário considerar a ordem em que eles precisarão ser retirados. Além de tornar o processo mais rápido, a organização também pode permitir o melhor uso de espaços, aumentando a quantidade máxima de itens ou evitando o desperdício dos espaços.

Ainda sobre evitar desperdícios, esse quesito é muito importante para as indústrias de papel, móveis, têxtil e metal-mecânica [Queiroz 2022, Cavali 2004, Belluzzo and Morabito 2005]. Todas essas áreas querem gerar o máximo de produtos com o mínimo de recursos materiais utilizados, para evitar o descarte desnecessário do material e prejuízos financeiros.

Os problemas citados são considerados problemas de corte e empacotamento. Problemas de corte envolvem cortar um objeto, como blocos de gesso, chapas de aço e barras de ferro, em itens menores. Enquanto problemas de empacotamento tratam sobre alocar um conjunto de itens $\mathcal{I}$ em um recipiente $\mathcal{B}$. Ambos são equivalentes entre si e é possível separá-los de acordo com sua dimensão.

O caso 2D, dimensão de estudo deste trabalho, possui uma vasta literatura de métodos de solução. As abordagens de solução se dividem entre exatas, que buscam a solução ótima do problema, e heurísticas, as quais podem não encontrar uma solução ótima, mas conseguem uma solução aceitável em tempo hábil. Dentre os métodos de solução exatos, um que se destaca é o procedimento de busca em árvore [Beasley 1985], mas existem muitos outros na literatura [Iori et al. 2021, Fekete and Schepers 1997, Delorme et al. 2016, Kenmochi et al. 2009]. Na parte de heurísticas, tem-se a *bottom-left* [Baker et al. 1980, Chehrazad et al. 2022] e *skyline* [Wei et al. 2011], as heurísticas também possuem grande presença na literatura [Burke et al. 2004, Rakotonirainy and van Vuuren 2020, Hopper and Turton 2001b, Chen et al. 2019, Huang and Chen 2007, Hopper and Turton 2001a].

Este trabalho visa criar métodos de solução para o problema de empacotamento no espaço de duas dimensões, onde as peças são retangulares e com um recipiente também retangular, mais especificamente na versão do empacotamento 2D da mochila, considerado NP-difícil [Iori et al. 2022]. Nessa versão do problema, dado um conjunto de itens $\mathcal{I}$, com cada item i possuindo um valor p_i, e uma caixa $\mathcal{B}$, o objetivo é maximizar a soma dos valores dos itens alocados dentro do recipiente. A abordagem escolhida para resolver o problema foi utilizar a heurística *bottom-left*, devido a sua simplicidade e aos limites computacionais e de tempo ao escolher algum método exato. Mesmo com a heurística sendo proposta em 1980, ela ainda está presente na literatura recente [Chehrazad et al. 2022, Hopper and Turton 2001b, Wei et al. 2011].

O principal objetivo deste trabalho é criar métodos de solução para o problema de empacotamento da mochila de peças retangulares, todos baseados na heurística *bottom-left*. Outros objetivos mais específicos são: implementar a *bottom-left* e os métodos derivados em Python, executá-los com instâncias de teste da literatura, comparar seus resultados e identificar vantagens e desvantagens de cada um.

2. Definição

Com o escopo do estudo definido como problema de empacotamento de retângulos, é possível ver sua definição formal. De acordo com [Iori et al. 2022], dado uma caixa retangular $\mathcal{B} = (W, H)$ de comprimento $W \in \mathbb{Z}_+$ e altura $H \in \mathbb{Z}_+$ e um conjunto $\mathcal{I}$ de itens também retangulares, onde cada item $i \in \mathcal{I}$ com comprimento $w_i \in \mathbb{Z}_+, w_i \leq W$ e altura $h_i \in \mathbb{Z}_+, h_i \leq H$. Um empacotamento $\mathcal{I}' \subseteq \mathcal{I}$ em $\mathcal{B}$ pode ser descrito como uma função $\mathcal{F} : \mathcal{I}' \to \mathbb{Z}_+^2$ que mapeie cada item $i \in \mathcal{I}'$ para um par de coordenadas $\mathcal{F}(i) = (x_i, y_i)$, de forma:

$$x_i \in \{0, \ldots, W - w_i\}, y_i \in \{0, \ldots, H - h_i\} \left(i \in \mathcal{I}'\right) \quad (1)$$

$$[x_i, x_i + w_i) \cap [x_j, x_j + w_j) = \emptyset \text{ ou } [y_i, y_i + h_i) \cap [y_j, y_j + h_j) = \emptyset \left(i, j \in \mathcal{I}', i \neq j\right) \quad (2)$$

Nesse modo de representação a caixa está posicionada no plano cartesiano, com seu canto inferior esquerdo na origem. Já as coordenadas $\mathcal{F}(i) = (x_i, y_i)$ representam a posição em que o canto inferior esquerdo da peça será alocado. A Restrição 1 garante que cada item deve estar inteiramente na caixa, enquanto a Restrição 2 impede sobreposição entre itens. Ambas restrições indicam uma orientação fixa, ou seja, peças não podem ser rotacionadas.

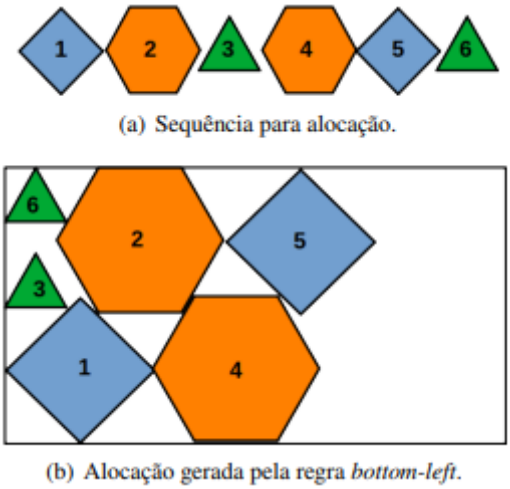
3. Métodos de solução

Como descrito na Seção 1, a maioria das classes do problema são NP-difíceis. Isso torna métodos de soluções exatos, os quais buscam pela solução ótima, extremamente custosos em tempo e recursos computacionais em instâncias de porte moderado, muitas vezes sendo inviáveis por falta de algum desses dois motivos. Consequentemente a literatura é dominada por abordagens que usam heurísticas e meta-heurísticas, sendo a *bottom-left* uma das principais estratégias de solução e será usada no estudo deste trabalho.

A *bottom-left* é uma heurística construtiva proposta por [Baker et al. 1980]. Embora tenha sido proposta a décadas, ainda é bastante usada na literatura atual, além de poder ser usada como componente de algoritmos mais sofisticados e para diferentes classes e variantes do problema. Ela foi utilizada nos trabalhos de [Hopper and Turton 2001b] na comparação de vários métodos de solução, [Wei et al. 2011] trazendo uma revisão do método e seus derivados e, mais recentemente, [Chehrazad et al. 2022] através de uma adaptação para o empacotamento de itens irregulares de forma gulosa.

Sua premissa é simples, dado uma fila de itens como entrada, enquanto ela não estiver vazia, basta retirar o primeiro item dela e alocar no canto mais a baixo e à esquerda quanto for possível [Bartmeyer et al. 2021], sem sobreposições entre peças. Caso não exista uma posição válida, a peça é desconsiderada e passa-se para próxima da fila. A Figura 1 mostra um exemplo de alocação para um dado conjunto de peças regulares.

Figura 1. Representação de alocação usando *bottom-left*.



(a) Sequência para alocação.

(b) Alocação gerada pela regra *bottom-left*.

Vale destacar que a própria ordem da fila pode gerar resultados diferentes, alterando a qualidade da solução. Um dos resultados esperados deste trabalho é identificar se há alguma forma de ordenação que se destaque na qualidade de solução, através da

comparação entre os diferentes modos. Para isso, serão usados conjuntos de instâncias frequentemente utilizados na literatura.

3.1. Critérios de ordenação

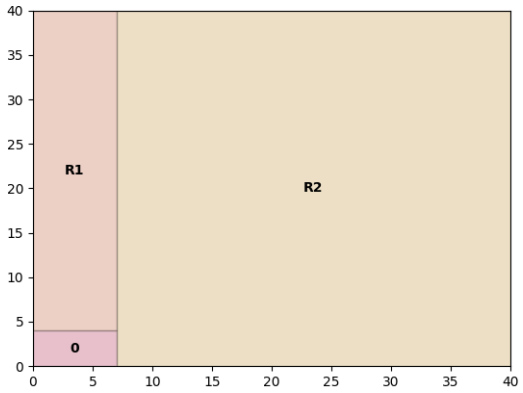
Para determinar o impacto da ordenação da fila, cinco critérios de ordenação foram escolhidos, sendo eles: área, perímetro, largura, altura e *id*. A ordenação por *id* considera a ordem em que os itens foram colocados na lista, ou seja, seria a forma padrão de resolver e ele será a base para definir se os demais critérios possuem algum benefício. Além disso, cada critério pode ser usado para ordenar a fila em ordem crescente ou decrescente, algo que também será analisado. Na literatura o mais comum é utilizar a ordenação decrescente pela área [Chen et al. 2019].

3.2. Criação de regiões

A sobreposição de peças e o domínio contínuo podem ser resolvidos utilizando a estratégia de criação de regiões. Com essa técnica, a Restrição 2 é trivialmente satisfeita. Nela, ao posicionar uma peça, duas regiões são criadas e o item seguinte será somente posicionado se couber em uma das regiões disponíveis.

Supondo um recipiente com altura e largura 40 e um item 0 com altura 4 e largura 7. Quando o item for posicionado na coordenada (0, 0), duas regiões, R1 e R2, serão criadas (Figura 2). A região R1 começará na coordenada (0, 4) e a R2 na (7, 0).

Figura 2. Regiões criadas traçando uma linha vertical.

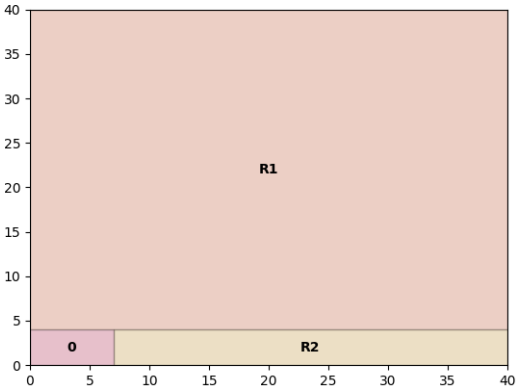


Agora o domínio passa a ser somente o canto inferior esquerdo de cada uma das regiões e sobreposições deixam de ser possíveis. Além disso, a regra para definir se uma peça cabe em dada região é igual a Restrição 1, simplificando o algoritmo. A fim de identificar o impacto das regiões na solução do modelo, quatro formas de criação delas foram usadas.

A primeira delas é **traçando uma linha vertical** a partir do canto superior direito de cada peça alocada (Figura 2). Nela a região R1 terá altura 36 e largura 7, indo até à coordenada (7, 40). Enquanto a R2 possuirá altura 40 e largura 33, chegando até à coordenada (40, 40).

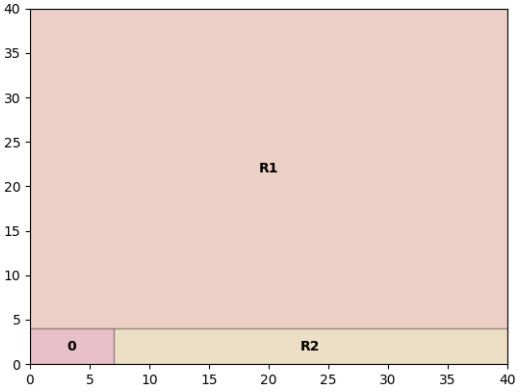
A segunda é semelhante à primeira, porém **traçando uma linha horizontal** (Figura 3). Nesse caso, R1 terá altura 36 e largura 40, indo até à coordenada (40, 40). Já R2 possuíra altura 4 e largura 33, chegando até à coordenada (40, 4).

Figura 3. Regiões criadas traçando uma linha horizontal.



Na terceira, a linha traçada (vertical ou horizontal) depende da área das regiões criadas com cada linha. Nesse modo o objetivo é maximizar a área de uma das regiões geradas, ele **identifica qual linha irá gerar a região de maior área e a traça**. Por exemplo, a Figura 2 gerou uma região com 252 de área e outra com 1320, enquanto a Figura 3 obteve regiões com 1440 e 132, então, nesse caso, a linha traçada será a horizontal (Figura 4).

Figura 4. Regiões criadas maximizando uma das regiões.

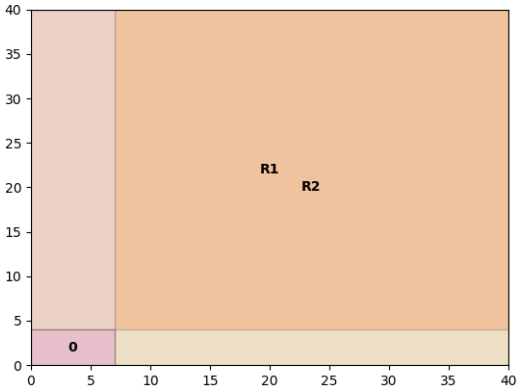


Maximizar uma região pode ser interessante, pois aumenta as chances do próximo item conseguir ser alocado, visto que uma das regiões será mais espaçosa. Em contrapartida, esse método também pode acabar gerando muitas regiões pequenas que não sejam utilizadas, diminuindo a qualidade da solução.

No quarto e último modo de criar regiões nenhuma linha é traçada, todas as regiões vão até o final do recipiente (Figura 5), criando **regiões sobrepostas**. R1 terá altura

40 e largura 36, enquanto R2 possuirá altura 40 e largura 33. Então, R1 e R2 terminarão na coordenada (40, 40). Nesse caso, sobreposições de peças podem ocorrer, então verificações são necessária para cumprir a Restrição 2. Ao fazer isso, é possível que mais peças sejam alocadas, visto que todas as regiões possuem área máxima. Esse modo foi criado para identificar se é de fato melhor que os demais e qual seu custo.

Figura 5. Regiões criadas possibilitando sobreposições.



Com os critérios para criação de regiões explicados, é possível diferenciá-los em dois tipos. O primeiro é dos que permitem sobreposição entre peças e, por isso, precisam de verificações para respeitar a Restrição 2 (regiões complexas), nesse tipo se encaixa somente o quarto modo. O segundo tipo contém os três primeiros critérios, onde somente a Restrição 1 precisa ser checada (regiões simples).

A Tabela 1 mostra, de forma resumida, os quatro modos de criar regiões, seu tipo (regiões simples ou complexas) e sua relação com a Restrição 2. A coluna "Divisão" indica o critério usado ao criar as regiões, enquanto a coluna "Restrição 2" mostra se a Restrição 2 é trivialmente satisfeita ou não. A última coluna ("Tipo"), indica se a região é simples ou complexa e está diretamente relacionada a Restrição 2.

Tabela 1. Modos de criar regiões, seu tipo e sua relação com a Restrição 2.

Modo	Divisão	Restrição 2	Tipo
1	Vertical	Trivial	Simples
2	Horizontal	Trivial	Simples
3	Maior área	Trivial	Simples
4	Regiões sobrepostas	Não trivial	Complexas

Todas as variações propostas foram implementadas em Python e avaliadas computacionalmente, os resultados são apresentados no Seção 4.

4. Resultados

Para testar os métodos de solução criados foram usadas 45 instâncias de teste da literatura, separadas em cinco conjuntos de instância de características diferentes: BKW,

Tabela 2. Configuração do computador de testes.

CPU	AMD Ryzen™ 5 3600X
RAM	16 GiB
Python	3.11.0
SO	Linux Mint 21.1 Cinnamon
Kernel	5.15

GCUT, NGCUT, OF e OKP. Todas as elas foram obtidas através da biblioteca pública 2DPackLib[1] [Iori et al. 2022].

O foco do trabalho é no empacotamento 2D da mochila, com o critério de maximização sendo a área ocupada do espaço. Mas nem todos conjuntos foram feitos para ser resolvidos dessa forma, nesses casos foram feitas leves adaptações para usá-los. O motivo de usar instâncias feitas com outro objetivo é para não viciar o modelo em instâncias específicas.

Como são cinco critérios de ordenação (Seção 3.1), com cada critério podendo ser crescente ou decrescente, quatro formas de criar regiões (Seção 3.2) e 45 instâncias, tem-se o total de 1800 casos de teste. Além disso, para conseguir resultados mais fiéis, a média, mediana e desvio padrão do tempo de execução foram calculados. Por isso, cada caso foi executado cinco vezes, totalizando 9000 execuções. Outros dados como a qualidade de solução (objetivo do trabalho), porcentagem de itens alocados e tempo, também foram computados. Todas as execuções foram feitas em um mesmo computador, com configurações conforme a Tabela 2.

Ao analisar a média, a mediana e o desvio padrão do tempo de execução, observou-se que a média e mediana possuem valores quase idênticos, enquanto o desvio padrão é pequeno ao ponto de poder ser ignorado, indicando que cinco execuções por caso de teste são suficientes. Portanto, a mediana e desvio padrão serão omitidos no restante do trabalho, podendo ser encontrados na versão completa dos dados gerados no Github[2].

Nas tabelas apresentadas nas seções seguintes, as colunas "Vitórias" e "Empates" trazem os resultados de forma quantitativa, enquanto a coluna "Qualidade %" de modo qualitativo. Nos testes, o valor p_i dos itens sempre é sua área, desconsiderando os valores dados pelas instâncias, caso hajam.

A qualidade de solução é determinada pela área ocupada do recipiente, no caso de todos os itens serem alocados, a qualidade é 100% independente da área do recipiente. A coluna "Vitórias" indica quantas vezes tal método de solução obteve o melhor resultado em comparação com os demais métodos em outras linhas. Enquanto a coluna "Empates" mostra a quantidade de vezes que o método conseguiu a melhor qualidade, mas outros também conseguiram. Essas colunas foram feitas da seguinte forma: entre cada combinação de critério de ordenação, modo de criar regiões e instâncias, é feita a comparação se a qualidade de solução foi melhor para ordenação crescente ou decrescente. No caso de am-

[1]Disponível em: https://site.unibo.it/operations-research/en/research/2dpacklib. Acessado em: 7 de julho de 2023.

[2]Disponível em: https://github.com/G-Carneiro/packing-problem/. Acessado em: 7 de julho de 2023.

bas conseguirem o melhor resultado, é acrescido 1 tanto na coluna "Vitórias", quanto na "Empates" de ambas. Por fim, a coluna "Tempo (s)" mostra o tempo médio de execução do método em segundos.

4.1. Ordenação crescente × decrescente

Uma primeira observação é a discrepância na qualidade de solução entre a ordenação crescente e a decrescente, algo já esperado. Na Tabela 3 é possível notar que ordenando de forma decrescente é possível ocupar cerca de 20% a mais do espaço (coluna "Qualidade $\%$"), quando comparado a ordenação crescente, em média.

Tabela 3. Resultado da comparação entre ordenação crescente e decrescente.

Decrescente	Vitórias	Empates	Qualidade %	Tempo (s)
Sim	736	8	78.9136	1.7798e+00
Não	167	8	57.3060	2.3715e+00

Com isso, fica claro que ordenar a fila de entrada da *bottom-left* de modo decrescente é vantajoso em termos de qualidade, quantidade e tempo de execução.

4.2. Comparativo entre critérios de ordenação

A Tabela 4 mostra o comparativo entre os critérios de criação de regiões, somente considerando a ordenação decrescente, já que não existem motivos para usar a crescente. Representando os dados dessa forma fica fácil identificar que utilizar algum critério de ordenação para a fila de entrada é vantajoso, pois ao usar o *id* os resultados foram os piores. Além disso, percebe-se que as ordenações por área e perímetro obtiveram os melhores resultados, ainda que os demais também sejam competitivos.

Tabela 4. Resultado da comparação entre critérios de ordenação decrescente.

Ordenação	Vitórias	Empates	Qualidade %	Tempo (s)
Área	63	39	82.7353	1.5874e+00
Perímetro	71	38	84.6986	1.5769e+00
Altura	40	16	77.4182	1.5655e+00
Largura	66	24	81.1899	2.0805e+00
Id	16	5	68.5261	2.0889e+00

A alta competitividade entre os critérios de ordenação é interessante, pois a maioria dos trabalhos na literatura, como o de [Chen et al. 2019], usam somente ordenação pela área, e isso pode ser um forte indicativo que os demais critérios devem ser mais explorados em certas circunstâncias.

4.3. Comparativo entre criação de regiões

Conforme mostra a Tabela 5, os modos criados traçando uma linha vertical ou horizontal apresentaram qualidades semelhantes e os menores tempos de execução, mas o método o qual traça uma linha vertical obteve mais vitórias. Regiões criadas para maximizar uma das mesmas conseguiram o segundo melhor resultado qualitativo e quantitativo, ao custo de um pequeno acréscimo no tempo de execução em relação aos dois primeiros. O último

Tabela 5. Resultado da comparação entre criação de regiões - ordenação decrescente.

Divisão	Vitórias	Empates	Qualidade %	Tempo (s)
Vertical	98	79	76.4030	2.7157e-03
Horizontal	70	60	75.9970	6.2101e-03
Maior área	104	89	79.7175	1.3743e-02
Sobrepostas	176	119	83.6420	7.2176e+00

modo de fato conseguiu os melhores resultados, porém a um custo altíssimo, levando cerca de 1000 vezes mais tempo que métodos mais rápidos.

A Tabela 6 traz um comparativo entre os dois tipos de criação de regiões, os que é preciso checar sobreposição e os que não (Seção 3.2). Na primeira linha da tabela a coluna "Qualidade %" representa a média do melhor resultado obtido em cada instância e a coluna "Tempo Total (s)" mostra a soma dos tempos que cada método de solução levou para cada instância. As duas colunas consideram somente métodos de solução que usam regiões onde não são necessárias verificações de sobreposição, ou seja, são considerados 30 modos de solução. A segunda linha considera somente método de solução o qual utiliza ordenação decrescente pela área e criação de regiões onde é necessário verificar sobreposições, esse método foi escolhido para o comparativo por apresentar os melhores resultados quantitativos e ser o segundo melhor qualitativamente.

Tabela 6. Resultado da comparação entre tipos de regiões.

Sobreposição	Qualidade %	Tempo Total (s)
Não	90.8278	1.6299e+01
Sim	87.2957	2.8313e+02

Com a Tabela 6 fica claro que, por mais que usar regiões onde é preciso checar sobreposições apresente o melhor resultado, é melhor executar todos demais métodos os quais não precisem e escolher somente o de melhor solução, pois assim é possível obter, na média, soluções de maior qualidade e ainda levando 10 vezes menos tempo.

5. Conclusão

No trabalho foram avaliados experimentalmente 40 métodos de solução baseados na heurística *bottom-left* para o problema de empacotamento de retângulos, considerando a versão da mochila e com o valor dos itens sua própria área. O resultado mais óbvio obtido foi a supremacia da ordenação decrescente sobre a crescente (Seção 4.1), não compensando utilizar a segunda para solucionar o problema.

O alto uso da ordenação decrescente pela área na literatura [Chen et al. 2019] foi justificado pelos resultados, já que essa composição conseguiu alguns dos melhores números (Seção 4.2). Os resultados também indicam que qualquer critério de ordenação obtêm melhores resultados do que deixar a fila desordenada (ordenação por *id*). Mas a alta competitividade de outros critérios de ordenação, esses não tão comuns na literatura, conseguindo melhores resultados em algumas instâncias, mostra que ainda há espaço para outras formas de ordenação.

Em relação às diferentes regiões criadas, por mais que regiões complexas tenham obtido melhores resultados (Seção 4.3), não compensa utilizá-las. Ao executar todos os métodos de regiões simples é possível conseguir melhores resultados e em menor tempo, quando comparado ao método de ordenação decrescente pela área e usando regiões complexas (ver a Tabela 6 na Seção 4.3).

Referências

Baker, B. S., Coffman Jr, E. G., and Rivest, R. L. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846.

Bartmeyer, P. M., Oliveira, L. T. d., Toledo, F. M. B. d., and Leao, A. A. S. (2021). Aprendizado por reforço aplicado ao problema de empacotamento de peças irregulares em faixas. *Anais*.

Beasley, J. E. (1985). An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 33(1).

Belluzzo, L. and Morabito, R. (2005). Otimização nos padrões de corte de chapas de fibra de madeira reconstituída: um estudo de caso. *Pesquisa Operacional*, 25:391–415.

Burke, E. K., Kendall, G., and Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671.

Cavali, R. (2004). Problemas de corte e empacotamento na indústria de móveis: um estudo de caso.

Chehrazad, S., Roose, D., and Wauters, T. (2022). A fast and scalable bottom-left-fill algorithm to solve nesting problems using a semi-discrete representation. *European Journal of Operational Research*, 300(3):809–826.

Chen, M., Wu, C., Tang, X., Peng, X., Zeng, Z., and Liu, S. (2019). An efficient deterministic heuristic algorithm for the rectangular packing problem. *Computers & Industrial Engineering*, 137:106097.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20.

Fekete, S. P. and Schepers, J. (1997). A new exact algorithm for general orthogonal d-dimensional knapsack problems. In *Algorithms—ESA'97: 5th Annual European Symposium Graz, Austria, September 15–17, 1997 Proceedings 5*, pages 144–156. Springer.

Hopper, E. and Turton, B. C. (2001a). A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16:257–300.

Hopper, E. B. C. H. and Turton, B. C. H. (2001b). An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57.

Huang, W. and Chen, D. (2007). An efficient heuristic algorithm for rectangle-packing problem. *Simulation Modelling Practice and Theory*, 15(10):1356–1365.

Iori, M., de Lima, V. L., Martello, S., Miyazawa, F. K., and Monaci, M. (2021). Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, 289(2):399–415.

Iori, M., de Lima, V. L., Martello, S., and Monaci, M. (2022). 2dpacklib: a two-dimensional cutting and packing library. *Optimization Letters*, 16(2):471–480.

Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., and Nagamochi, H. (2009). Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research*, 198(1):73–83.

Morabito Neto, R. and Widmer, J. A. (1992). *Abordagem em grafo-e-ou para o problema do empacotamento: aplicacao ao carregamento de paletes e conteineres*. PhD thesis.

Queiroz, L. R. d. S. (2022). *Estudo de problemas de corte de itens irregulares com incertezas*. PhD thesis, Universidade de São Paulo.

Rakotonirainy, R. G. and van Vuuren, J. H. (2020). Improved metaheuristics for the two-dimensional strip packing problem. *Applied Soft Computing*, 92:106268.

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

Wei, L., Oon, W.-C., Zhu, W., and Lim, A. (2011). A skyline heuristic for the 2d rectangular packing and strip packing problems. *European Journal of Operational Research*, 215(2):337–346.