# Problema de empacotamento de retângulos

## avaliação de métodos de solução baseados em *bottom-left*

Gabriel Medeiros Lopes Carneiro
Orientador: Pedro Belin Castellucci
Coorientador: Rafael de Santiago

Universidade Federal de Santa Catarina

25 de junho de 2023

UFSC

2023-06-25 Empacotamento de retângulos


Boa tarde, meu nome é Gabriel e hoje vou apresentar o meu tcc. O trabalho trata sobre a avaliação de métodos de solução baseados em *bottom-left* para problema de empacotamento de retângulos. Ele foi feito sob orientação do professor Pedro e teve coorientação do professor Rafael.

# Sumário







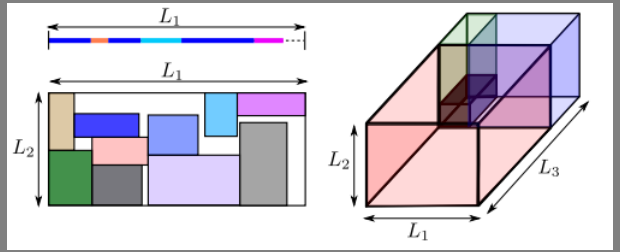
Sumário

1. Problema
2. Métodos de solução
3. Resultados
4. Conclusão

Vou começar explicando o problema em si, passando por suas características e classificações.
Vou mostrar o que é *bottom-left*, como ela funciona e os métodos de solução criados com base nela.
Também vou mostrar os resultados obtidos ao rodar instâncias de teste.
Por fim, vou apresentar algumas conclusões que podem ser feitas a partir do trabalho.

# Problema

Figura: Represeção 1D, 2D e 3D.




Fonte: CASTELLUCCI (2019).





Problema

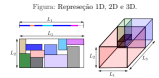


Figura: Represeção 1D, 2D e 3D.
Fonte: CASTELLUCCI (2019).

A premissa do problema é simples, alocar peças em um espaço. É algo simples, mas é de difícil resolução, podendo possuir $N$-dimensões e diversos tipos de peças, de modo que é preciso separar o problema em diferentes classes e ainda existem variantes dentro das classificações. Como exemplos:

O caso 1D pode ser usado para cortar uma barra de ferro em certo tamanhos, de forma que seu desperdício seja mínimo.

Já no 2D poderia ser aplicado em casos para alocação de paletes com altura fixa em um caminhão ou depósito.

O caso 3D pode ser usado para alocar caixas em um depósito ou container, acredito que esse seja o mais simples de pensar em aplicações.

# Problema

## Restrições

- Recipiente $\mathcal{B} = (W, H)$.
- Conjunto de itens $\mathcal{I}$.
- Conjunto solução $\mathcal{I}' \subseteq \mathcal{I}$.

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

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





Problema
Restrições
• Recipiente $\mathcal{B} = (W, H)$.
• Conjunto de itens $\mathcal{I}$.
• Conjunto solução $\mathcal{I}' \subseteq \mathcal{I}$.
$x_i \in \{0, \ldots, W - w_i\}, y_i \in \{0, \ldots, H - h_i\} \left(i \in \mathcal{I}'\right)$ (1)
$[x_i, x_i + w_i) \cap [x_j, x_j + w_j) = \emptyset$ ou $[y_i, y_i + h_i) \cap [y_j, y_j + h_j) = \emptyset \left(i, j \in \mathcal{I}', i \neq j\right)$ (2)

O trabalho lida somente com peças retangulares, então a dimensão de interesse é 2D. Para resolver o problema ele será representado como um modelo de otimização, portanto, são necessárias algumas restrições para o modelo.
Então, dado um recipiente $\mathcal{B}$ de largura $W$ e altura $H$, um conjunto de itens $\mathcal{I}$ e um conjunto solução $\mathcal{I}'$. Tem-se duas restrições.
A primeira restrição garante que um item só é alocado no recipiente se couber nele.
Já a segunda impede sobreposição entre as peças.

# Problema

## Empacotamento da mochila

Dado:

- Recipiente $\mathcal{B} = (W, H)$.
- Conjunto de itens $\mathcal{I}$.
- Cada item $i \in \mathcal{I}$ possui um valor $p_i$.

Encontrar:

- Conjunto solução $\mathcal{I}' \subseteq \mathcal{I}$.

De forma que:

$$\max \sum_{i \in \mathcal{I}'} p_i$$





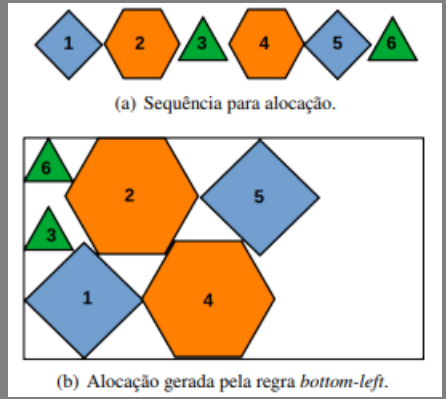

Problema
Empacotamento da mochila
Dado:
• Recipiente $\mathcal{B} = (W, H)$.
• Conjunto de itens $\mathcal{I}$.
• Cada item $i \in \mathcal{I}$ possui um valor $p_i$.
Encontrar:
• Conjunto solução $\mathcal{I}' \subseteq \mathcal{I}$.
De forma que:
$\max \sum_{i \in \mathcal{I}'} p_i$

Como já mencionado, existem muitas versões do problema e a de interesse é o empacotamento da mochila.
Nessa versão, dado um recipiente $\mathcal{B}$ e um conjunto de itens, onde cada item $i$ possui um valor $p_i$, deve-se encontrar um conjunto solução que maximize o somatório dos valores dos itens alocados.
No trabalho, o valor $p_i$ é a área do item $i$. Ou seja, o objetivo é maximizar a área ocupada do recipiente $\mathcal{B}$.

# Métodos de solução

*Bottom-left*

Figura: Representação de alocação.




Fonte: BARTMEYER et al. (2021).





Métodos de solução
*Bottom-left*
Figura: Representação de alocação.
(a) Sequência para alocação
(b) Alocação gerada pela regra bottom-left
Fonte: BARTMEYER et al. (2021).

Como o problema é NP-difícil, usar métodos exatos exigiria muitos recursos computacionais e/ou tempo de execução. Então, a heurística *bottom-left* foi escolhida para solucionar o problema.
Ela é bem simples, dado uma lista como entrada, os itens são retirados um a um e posicionados no ponto mais a baixo e mais a esquerda quanto for possível.
Caso a peça não caiba em nenhuma posição ela não entra na solução e passa-se para a próxima da fila.
Aqui fica claro que a sequência de alocação tem impacto direto na qualidade da solução. Mas como definir essa ordenação? Existe algum critério que seja melhor que os demais? Para responder essas perguntas, escolhi alguns critérios para ordenar a lista e comparar suas soluções.

# Métodos de solução

## Critérios de ordenação

- Área.
- Perímetro.
- Largura.
- Altura.
- Id.





Métodos de solução
Critérios de ordenação
• Área.
• Perímetro.
• Largura.
• Altura.
• Id.

A partir desse ponto começa de fato o desenvolvimento do trabalho.

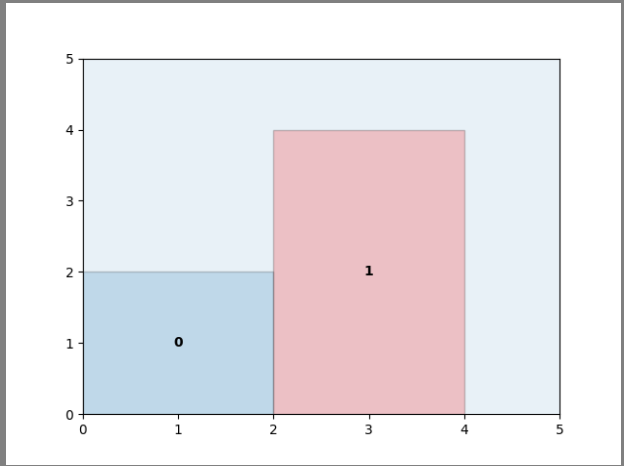
5 critérios de ordenação foram escolhidos: área, perímetro, largura, altura e id.

A ordenação por id considera a ordem em que os itens foram colocados na lista (ou criados), ou seja, seria a forma padrão de resolver, sem ordenações.

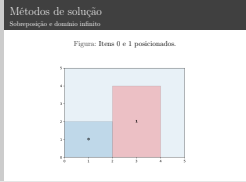
Cada critério pode ser usado de forma crescente ou decrescente.

Com os critérios definidos, podemos passar para os próximos desafios do problema, que são a sobreposição e o domínio contínuo.

Figura: Itens 0 e 1 posicionados.

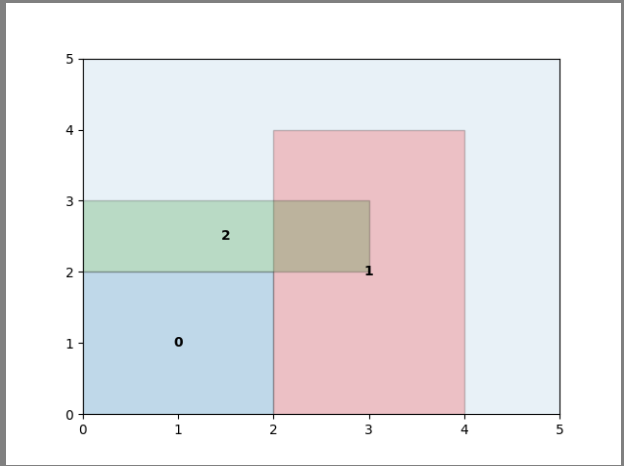




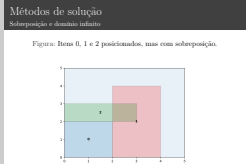



Bom, supondo que estejamos em um estado do algoritmo como mostra a figura, onde o item 0 foi o primeiro alocado e o item 1 foi alocado a sua direita na posição (2, 0), porque não cabia logo acima na posição (0, 2) devido a Restrição 1.

Figura: Itens 0, 1 e 2 posicionados, mas com sobreposição.

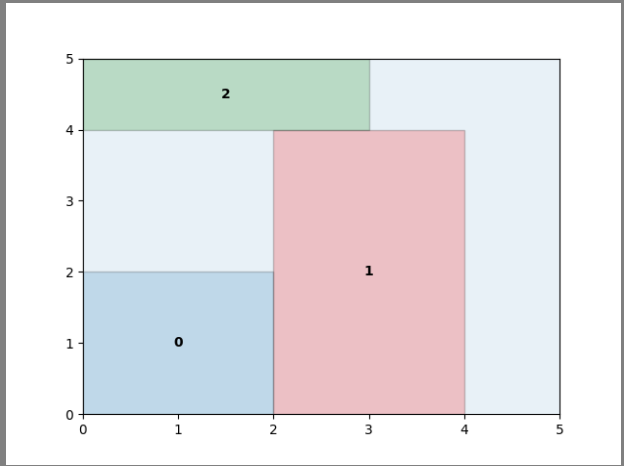




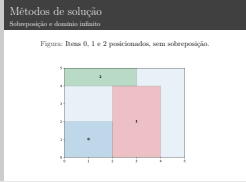
Agora queremos alocar um terceiro item, de largura 3 e altura 1. Ao posicionar a peça na posição (0, 2) percebe-se que a Restrição 1 é satisfeita, porém a Restrição 2 não.

Figura: Itens 0, 1 e 2 posicionados, sem sobreposição.



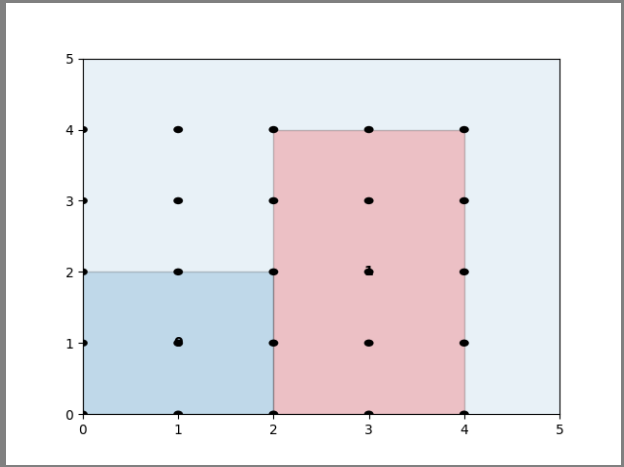





Nesse caso, com poucas peças alocadas e um auxílio visual, é fácil dizer que a posição (0, 4) é a primeira posição válida de acordo com a *bottom-left*. Mas como chegar até ela? Existem infinitos pontos entre as coordenadas (0, 2) e (0, 4).

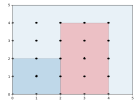
Figura: Itens 0 e 1 posicionados, com domínio discreto.





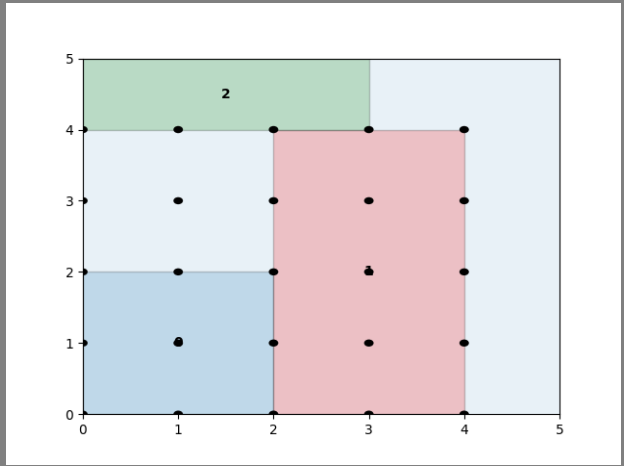



Métodos de solução
Sobreposição e domínio infinito
Figura: Itens 0 e 1 posicionados, com domínio discreto.



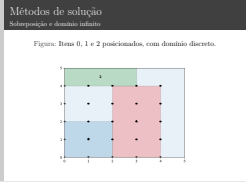
Como todas as instâncias de teste usadas tratam somente de peças e recipientes com valores inteiros uma abordagem possível seria discretizar o domínio.

Figura: Itens 0, 1 e 2 posicionados, com domínio discreto.








Dessa forma somente coordenadas de valores inteiros precisariam ser checadas, resolvendo parcialmente o problema com o domínio, já que ainda temos muitos pontos para checar, principalmente em recipientes grandes. Mas isso não resolve a parte de sobreposição. Para cada ponto ainda é necessário verificar se existe sobreposição com cada uma das peças já alocadas, algo extremamente custoso. Além disso, a discretização não funcionaria tão bem em casos com valores não inteiros, prejudicando a aplicação em vários problemas do mundo real.

Tabela: Resumo das regiões.

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





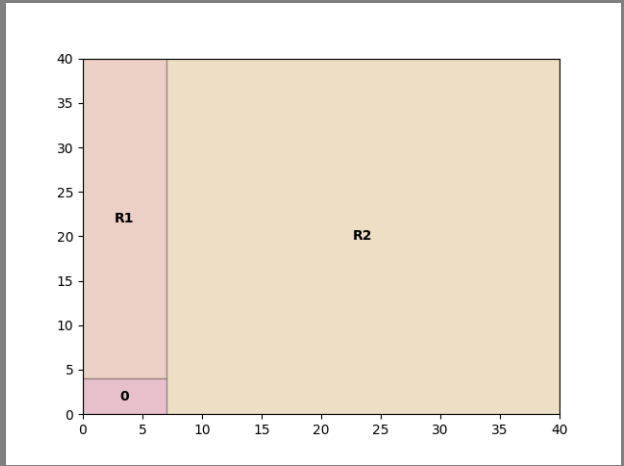

Métodos de solução
Regiões
Tabela: Resumo das regiões.
Modo Divisão Restrição 2 Tipo
1 Vertical Trivial Simples
2 Horizontal Trivial Simples
3 Maior área Trivial Simples
4 Regiões sobrepostas Não trivial Complexas

Ambos desafios, de sobreposição e de domínio infinito, podem ser resolvidos utilizando a estratégia de criação de regiões. Utilizando essa técnica a Restrição 2 é trivialmente satisfeita. Nela, ao posicionar uma peça, duas regiões são criadas e o item seguinte somente será posicionado se couber em uma dessas regiões.

O domínio passa a ser somente o canto inferior esquerdo de cada uma das regiões e sobreposições não são mais possíveis. Além disso, a regra para definir se uma peça cabe em dada região é igual a Restrição 1, tornando o algoritmo de solução bem simples.

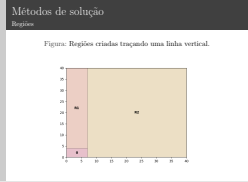
Escolhi criar as regiões de 4 formas diferentes, para identificar se isso teria algum impacto na solução. Teoricamente, resolver trivialmente a Restrição 2, pode causar prejuízos na qualidade de solução. Então, um dos modos não resolve trivialmente a Restrição 2 e foi criado verificar seu custo-benefício.

Figura: Regiões criadas traçando uma linha vertical.



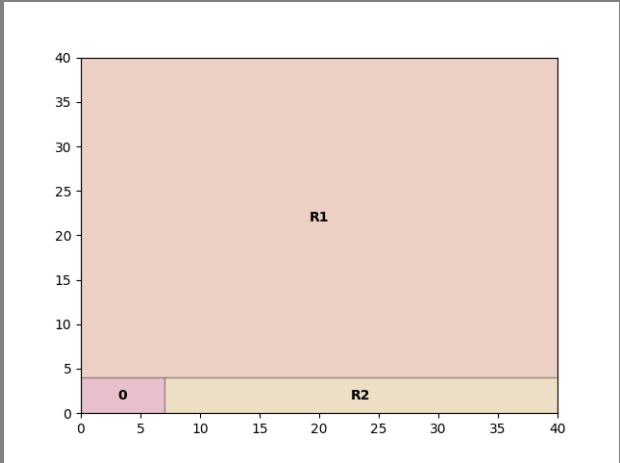





A primeira forma é traçando uma linha vertical a partir do canto superior direito de cada peça alocada. Nas figuras, retângulos indicados com um R são regiões.

Figura: Regiões criadas traçando uma linha horizontal.



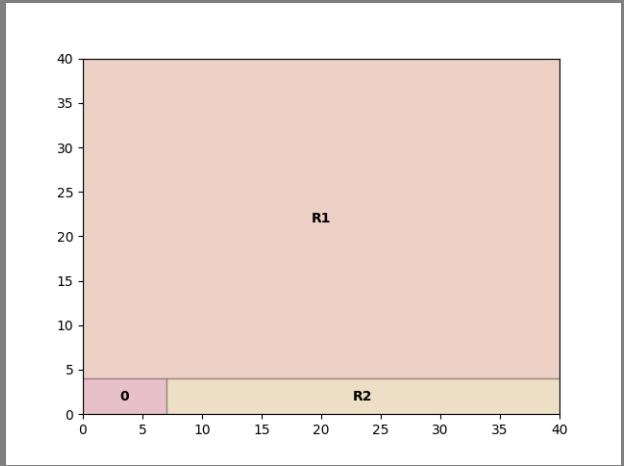




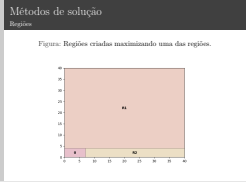

Métodos de solução
Regiões
Figura: Regiões criadas traçando uma linha horizontal.

A segunda é semelhante a primeira, porém usando uma linha horizontal.

Figura: Regiões criadas maximizando uma das regiões.

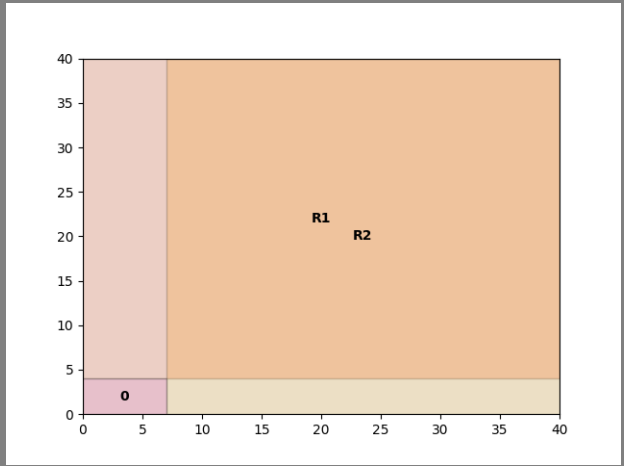




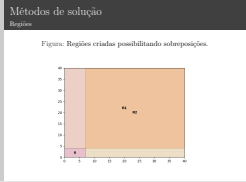




Já na terceira, é traçada uma linha (vertical ou horizontal) que maximize a área de uma das regiões geradas, basicamente identifica qual dos dois primeiros métodos gera a região de maior área. Isso é interessente pois dá uma garantia maior de que o item seguinte será alocado, em contrapartida pode gerar muitas regiões pequenas que podem não ser utilizadas, diminuindo a qualidade da solução.

Figura: Regiões criadas possibilitando sobreposições.






No último modo nenhuma linha é traçada, todas as regiões vão até o final do recipiente, ficando sobrepostas. Nesse caso sobreposições de peças podem ocorrer, então verificações são necessárias para cumprir a Restrição 2.

Tabela: Resumo das regiões.

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






Métodos de solução
Regiões
Tabela: Resumo das regiões.
Modo Divisão Restrição 2 Tipo
1 Vertical Trivial Simples
2 Horizontal Trivial Simples
3 Maior área Trivial Simples
4 Regiões sobrepostas Não trivial Complexas

Resumindo os modos das regiões, 3 geram regiões simples e resolvem trivialmente a Restrição 2 e o último deveria apresentar melhores resultados na qualidade de solução, já que a Restrição 2 é não-trivial.

# Resultados

## Testes

- 45 instâncias.
- $45 \cdot 5 \cdot 2 \cdot 4 = 1800$ casos de teste.
- 5 testes por configuração.
- $1800 \cdot 5 = 9000$ execuções.





Resultados
Testes
• 45 instâncias.
• $45 \cdot 5 \cdot 2 \cdot 4 = 1800$ casos de teste.
• 5 testes por configuração.
• $1800 \cdot 5 = 9000$ execuções.

Para testar os métodos de solução criados foram usadas 45 instâncias de teste.
Como temos 45 instâncias, 5 critérios de ordenação, cada critério pode ser crescente ou decrescente, e 4 formas de criar regiões, temos o total de 1800 casos a serem testados.
Cada método foi executado 5 vezes em cada uma das instâncias para se obter um média, totalizando 9000 execuções.

Tabela: Comparativo 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 |

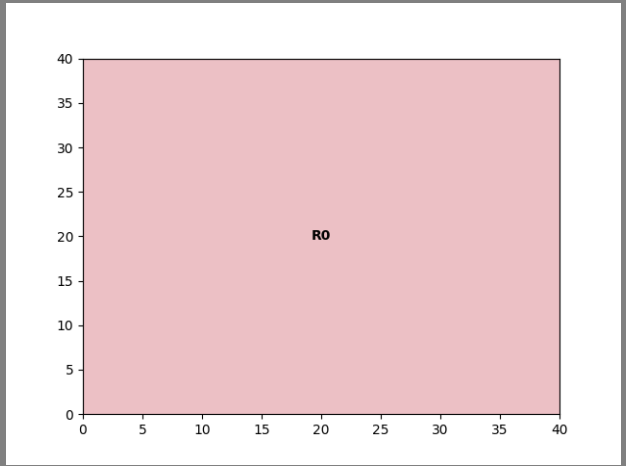




Resultados
Ordenação
Tabela: Comparativo entre ordenação crescente e decrescente.
Decrescente Vitórias Empates Qualidade % Tempo (s)
Sim 736 8 78.9136 1.7798e+00
Não 167 8 57.3060 2.3715e+00

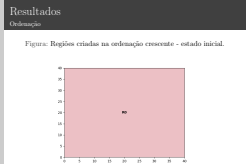
Nas tabelas, a coluna “Qualidade” indica qualidade da solução obtida pelo método, ou seja, a porcentagem, em média, da área ocupada dos recipientes. A coluna “Vitórias” indica quantas vezes o critério obteve o melhor resultado e a coluna “Empates” indica quantas vezes obteve o melhor resultado, mas não foi o único a conseguir. A primeira coisa que fica evidente com os resultados é discrepância na qualidade de solução entre a ordenação crescente e a decrescente, algo já esperado.

Figura: Regiões criadas na ordenação crescente - estado inicial.



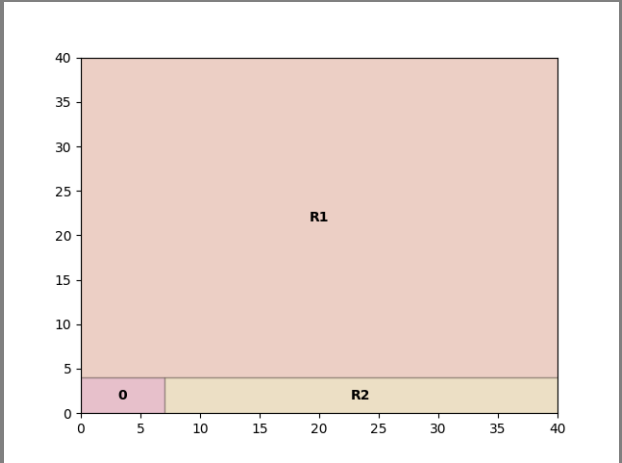




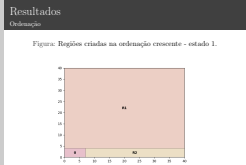



Isso se deve a como as regiões são criadas, as figuras mostram o caso para ordenação crescente pela altura e linha horizontal para criação de regiões.

Figura: Regiões criadas na ordenação crescente - estado 1.





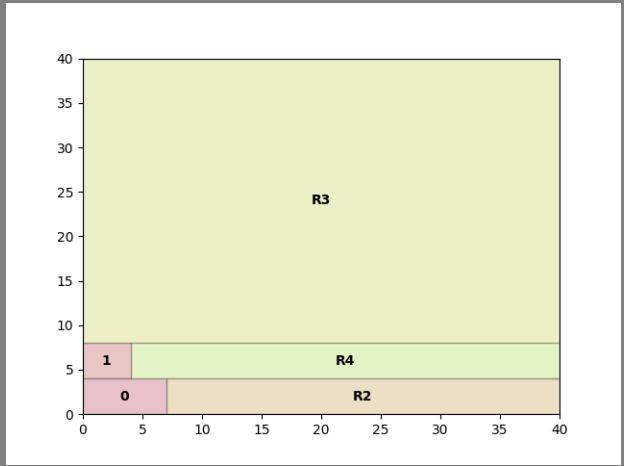



Ao posicionar uma peça uma das regiões ficará com a mesma altura do item recém-posicionado, como a ordenação é crescente a próxima peça terá no mínimo a mesma altura, mas o provável é que seja mais alta, impossibilitando que seja alocada nessa região.

# Resultados

## Ordenação

Figura: Regiões criadas na ordenação crescente - estado 2.

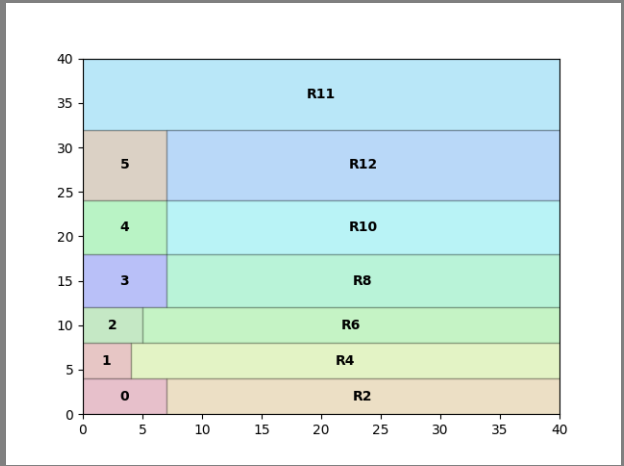



O mesmo ocorre para o segundo e demais itens, fazendo com que muitas regiões fiquem sem poder receber peças.

# Resultados

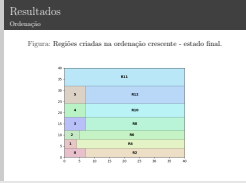
## Ordenação

Figura: Regiões criadas na ordenação crescente - estado final.







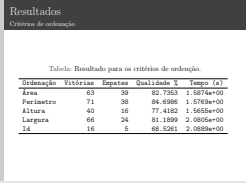
Essa figura mostra o estado final do algoritmo e grande parte do espaço ainda está livre, mas não será preenchido. Algo semelhante ocorre com outros critérios de ordenação e criação regiões.

Tabela: Resultado para os critérios de ordenção.

| 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 |








Em relação aos critérios de ordenação, fica claro que ter algum critério de ordenação melhora e muito a solução, já que ordernar por ID teve um péssimo desempenho. Mas o curioso é que todos os demais critérios são competitivos entre si. A literatura em geral usa somente ordenação pela área (CHEN et al., 2019), esses resultados podem indicar que algumas instâncias possuem características que torne mais interessante outro método de ordenação.

# Resultados

## Regiões

Tabela: Comparativo entre criação de regiões.

| 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 |






Resultados
Regiões
Tabela: Comparativo entre criação de regiões.
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

Indo para o comparativo entre regiões percebemos que a que permite sobreposições se saiu melhor, tanto em quantitativa quanto qualitativa, ainda que na maioria dos casos não foi a única que encontrou a melhor solução, porém com um custo altíssimo de tempo. Regiões criadas com linhas verticais e horizontais foram mais rápidas, mas com soluções de pior qualidade. Enquanto maximizando as regiões levou um pouco a mais de tempo, mas também com acréscimo na qualidade. Aqui a gente percebe que ter sobreposição demora em torno de 1000 vezes mais.

# Resultados

## Sobreposição

Tabela: Resultados para sobreposição.

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

Na última tabela eu trouxe os números da comparação entre regiões simples e complexas. Na primeira linha temos os resultados de todas as combinações possíveis com regiões simples e o tempo total que levou para executar todas as instâncias. Já na segunda linha somente a ordenação decrescente pela área e regiões complexas foi considerada, já que obteve os melhores resultados. Os demais critérios não foram considerados pois esse sozinho já ultrapassa o tempo de todos os métodos sem sobreposição, caso fossem considerados o tempo total seria cerca de 10 vezes maior enquanto a qualidade teria pouco acréscimo. Aqui fica nítido que compensa muito mais, tanto em qualidade quanto em tempo, rodar todas as combinações possíveis com regiões simples e escolher o melhor resultado. Mas por que tanta diferença no tempo de execução entre com e sem sobreposição?

# Resultados

## Complexidade

- Sem sobreposição: $R = O\left(\frac{n^2+n}{2}\right)$.
- Com sobreposição: $R = O\left(\frac{n^2+n}{2}\right), S = O\left(\frac{n^3-n}{3}\right)$.
  $n = 3152 \rightarrow R = 4\,969\,128, S = 10\,438\,481\,552$.





Resultados
Complexidade
• Sem sobreposição: $R = O\left(\frac{n^2+n}{2}\right)$.
• Com sobreposição: $R = O\left(\frac{n^2+n}{2}\right), S = O\left(\frac{n^3-n}{3}\right)$.
$n = 3152 \rightarrow R = 4\,969\,128, S = 10\,438\,481\,552$.

Isso se deve a complexidade dos algoritmos. Como dito antes, sem sobreposições temos que checar se um item cabe em uma região, no pior caso teremos que fazer isso para $(n^2+n)/2$ regiões, onde $n$ é número de itens. Enquanto com sobreposição, além de ter esse número de regiões, para cada uma delas também é necessário checar possíveis sobreposições com as peças já alocadas, sendo o número de verificações igual a $(n^3-n)/3$, isso no pior caso, algo custoso. Por exemplo, para uma instância com 3152 itens podem ser necessárias mais de 10 bilhões de verificações de sobreposição. Então, aquela diferença de 1000 vezes fica ainda maior de acordo com a quantidade de itens a serem alocados.

# Conclusão

- Múltiplos métodos de solução.
- Resultados inesperados.
- Com sobreposição × sem sobreposição.
  - Escalabilidade.





Conclusão
- Múltiplos métodos de solução.
- Resultados inesperados.
- Com sobreposição × sem sobreposição.
  - Escalabilidade.

Bom, indo para as conclusões. Foram testados vários métodos de solução, todos baseados em *bottom-left*, ficou evidente que ordenar a lista de entrada de forma decrescente é vantajoso.
Tivemos alguns resultados inesperados como a competitividade entre todos os critérios de ordenação, sendo necessária uma investigação sobre características das instâncias. E também a pouca, ou nenhuma, vantagem em termos de qualidade quando usamos regiões que permitem sobreposições.
De modo geral, pode-se resolver um problema com todas as combinações que usem regiões sem sobreposição e buscar a de melhor solução, já que seu tempo de execução é pequeno. Resolver usando regiões com sobreposição só é recomendado em casos onde o modelo será usado mais de uma vez e sem alterações.

Por fim, caso se queira aumentar a escala dos problemas, seja na dimensão ou na quantidade de itens, compensa somente trabalhar com regiões sem sobreposição. No caso de aumentar a dimensão seu custo é baixo, sendo necessário verificar somente um parâmetro extra.

# Referências I


BARTMEYER, Petra Maria et al. Aprendizado por reforço aplicado ao problema de empacotamento de peças irregulares em faixas. **Anais**, 2021. Disponível em: <https://repositorio.usp.br/directbitstream/455094df-864a-4fad-8a97-c5f59fd3d6ca/3051981.pdf>.

CASTELLUCCI, Pedro Belin. **Consolidation problems in freight transportation systems: mathematical models and algorithms**. 2019. Tese (Doutorado) – Universidade de São Paulo. Disponível em: <https://pdfs.semanticscholar.org/90e7/bd898951e1350c2694478b63fbcde508e189.pdf>.

CHEN, Mao et al. An efficient deterministic heuristic algorithm for the rectangular packing problem. **Computers & Industrial Engineering**, Elsevier, v. 137, p. 106097, 2019.







Referências I

BARTMEYER, Petra Maria et al. Aprendizado por reforço aplicado ao problema de empacotamento de peças irregulares em faixas. **Anais**, 2021. Disponível em: <https://repositorio.usp.br/directbitstream/455094df-864a-4fad-8a97-c5f59fd3d6ca/3051981.pdf>.

CASTELLUCCI, Pedro Belin. **Consolidation problems in freight transportation systems: mathematical models and algorithms**. 2019. Tese (Doutorado) – Universidade de São Paulo. Disponível em: <https://pdfs.semanticscholar.org/90e7/bd898951e1350c2694478b63fbcde508e189.pdf>.

CHEN, Mao et al. An efficient deterministic heuristic algorithm for the rectangular packing problem. **Computers & Industrial Engineering**, Elsevier, v. 137, p. 106097, 2019.

# Referências II

IORI, Manuel et al. 2DPackLib: a two-dimensional cutting and packing library. **Optimization Letters**, Springer, v. 16, n. 2, p. 471–480, 2022.





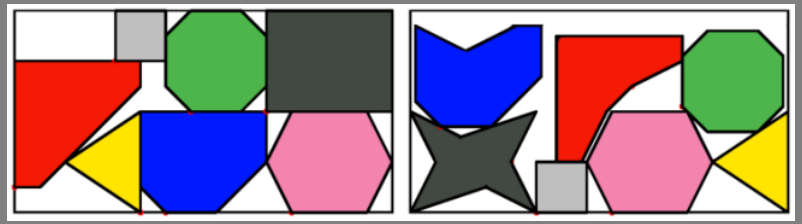
Referências II

IORI, Manuel et al. 2DPackLib: a two-dimensional cutting and packing library. **Optimization Letters**, Springer, v. 16, n. 2, p. 471–480, 2022.

# Problema

## Tipos de peças

Figura: Exemplos de peças convexas (esquerda) e côncavas (direita).



Fonte: BARTMEYER et al. (2021).





Problema
Tipos de peças
Figura: Exemplos de peças convexas (esquerda) e côncavas (direita).
Fonte: BARTMEYER et al. (2021).

- Convexas: todos ângulos internos são menores que 180°.
- Côncavas: pelo menos um ângulo interno maior que 180°.
- Regulares: peças convexas com todos lados e ângulos iguais.
- Irregulares: pelo menos um lado ou ângulo diferente.
- Outra forma de se definir entre convexa ou não é checar se existe alguma reta que atravesse o objeto em dois pontos diferentes, se sim, é côncava.
- O trabalho foca em peças retangulares.

# Problema

## Classificação

- Empacotamento em faixa.
- Empacotamento da mochila.
- Empacotamento em caixas.
- Empacotamento ortogonal.






Problema
Classificação
- Empacotamento em faixa.
- Empacotamento da mochila.
- Empacotamento em caixas.
- Empacotamento ortogonal.

- Empacotamento em faixa: Dado um conjunto de itens e uma caixa com comprimento fixo, queremos encontrar uma solução de altura mínima.
- Empacotamento da mochila: Nesse caso, queremos maximizar o valor da caixa (geralmente é a área da caixa).
- Empacotamento em caixas: Minimizar o número de caixas necessárias para empacotar todos os itens.
- Empacotamento ortogonal: Alocar todos os itens numa caixa.
- Todas classificações do problema são NP-difícil, com exceção da ortogonal (NP-completo)(IORI et al., 2022).

# Problema

## Variantes

- Corte guilhotinado.
- Rotações ortogonais.
- Restrições de carga e descarga.
- Caixas de tamanho variável.





Problema
Variantes
- Corte guilhotinado.
- Rotações ortogonais.
- Restrições de carga e descarga.
- Caixas de tamanho variável.

Aqui vou citar algumas variantes do problema, mas nenhuma foi usada no trabalho.

- Corte guilhotinado: Consiste em cortar a caixa de forma paralela a um dos lados de forma recursiva.
- Rotações ortogonais: É um modo de relaxar o problema, permitindo rotações de 90° nos itens.
- Restrições de carga e descarga: Algumas peças precisam ser posicionadas em certa posição ou próximas a outras.
- Caixas de tamanho variável: Define que caixas não precisam ter o mesmo tamanho (aplicável somente para Empacotamento em Caixas).

# Resultados

## Melhores combinações de solução

Tabela: Resultados da comparação entre todas combinações.

| Regiões | Ordenação | Desc. | V | E | Qualidade % | Tempo (s) |
|---|---|---|---|---|---|---|
| Vertical | Largura | Sim | 9 | 8 | 84.5497 | 2.4820e-03 |
| Maior área | Perímetro | Sim | 7 | 6 | 85.8682 | 1.2944e-02 |
| Sobrepostas | Área | Sim | 13 | 11 | 87.2957 | 6.4349e+00 |
| Sobrepostas | Largura | Sim | 16 | 10 | 85.9266 | 8.4384e+00 |





Resultados
Melhores combinações de solução
Tabela: Resultados da comparação entre todas combinações.
Regiões Ordenação Desc. V E Qualidade % Tempo (s)
Vertical Largura Sim 9 8 84.5497 2.4820e-03
Maior área Perímetro Sim 7 6 85.8682 1.2944e-02
Sobrepostas Área Sim 13 11 87.2957 6.4349e+00
Sobrepostas Largura Sim 16 10 85.9266 8.4384e+00

Aqui eu mostro uma tabela com os resultados das combinações que se saíram melhores. Em termos de quantidade, quem se saiu melhor foi a criação de regiões com sobreposição, com a largura como critério de ordenação. Já em qualidade o melhor resultado foi obtido com sopreposição e ordenação por área. A maximização de regiões e ordenação por perímetro ficou bem próxima, ainda mais consirando o custo-benefício. A primeira linha da tabela mostra a combinação entre a criação de regiões na vertical e ordenação pela largura, esse resultado é bem interessante pois têm um dos menores tempos e ainda consegue ser competitivo tanto em qualidade quanto em quantidade.

# Resultados

## Conjuntos de instâncias

Tabela: Resultados para os conjuntos de instância.

| Conjunto | Qualidade % | Itens % | Tempo Total (s) |
|---|---|---|---|
| BKW | 94.4783 | 85.7782 | 1.2688e+01 |
| GCUT | 84.6060 | 20.0994 | 2.0189e-01 |
| NGCUT | 88.2085 | 35.0307 | 8.1531e-01 |
| OF | 92.0714 | 34.0580 | 1.0821e-02 |
| OKP | 93.9360 | 22.8232 | 1.1026e-01 |





Resultados
Conjuntos de instâncias
Tabela: Resultados para os conjuntos de instância.
Conjunto Qualidade % Itens % Tempo Total (s)
BKW 94.4783 85.7782 1.2688e+01
GCUT 84.6060 20.0994 2.0189e-01
NGCUT 88.2085 35.0307 8.1531e-01
OF 92.0714 34.0580 1.0821e-02
OKP 93.9360 22.8232 1.1026e-01

Nessa tabela eu trouxe os resultados separados de acordo com o conjunto de instância, a BKW demorou mais por ter os maiores números de itens a serem alocados dentre todas as instâncias. Em geral, temos bons resultados para cada conjunto, mas vale investigar se algum deles possui alguma característica que torne melhor determinado método de solução.