

# Problemas de Empacotamento

métodos de solução baseados em *bottom-left*

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

Universidade Federal de Santa Catarina

19 de maio de 2023



2023-05-19

## Problemas de Empacotamento

Problemas de Empacotamento  
métodos de solução baseados em *bottom-left*

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

Universidade Federal de Santa Catarina

19 de maio de 2023



Meu nome é Gabriel e hoje vou apresentar uma prévia do meu tcc.

O trabalho trata sobre métodos de solução baseados em *bottom-left* para problema de empacotamento, ele foi feito sob orientação do professor Pedro e teve coorientação do professor Rafael.

1. Conceitos básicos
2. Problema
3. *Bottom-left*
4. Resultados
5. Conclusão



### └ Sumário

Vou começar explicando alguns termos que devo usar ao longo da apresentação.

Depois vou explicar o problema em si, passando por suas características e classificações.

Vou mostrar o que é *bottom-left*, como ela funciona e as adaptações feitas 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.

$$\min/\max f(x), x \in \mathcal{X}.$$

- $x$ : variável de decisão,  $x = x_1, x_2, \dots, x_n$ .
- $\mathcal{X}$ : conjunto factível ou domínio;
- $f(x)$ : função objetivo.

$$\min/\max f(x), x \in \mathcal{X}.$$

- $x$ : variável de decisão,  $x = x_1, x_2, \dots, x_n$ .
- $\mathcal{X}$ : conjunto factível ou domínio;
- $f(x)$ : função objetivo.

Modelos de otimização são aproximações da realidade, representam o problema de maneira simples e objetiva, usando restrições. Geralmente quer minimizar ou maximizar uma função  $f(x)$  com  $x$  obedecendo algumas restrições.

- $x$ : variável de decisão,  $x = x_1, x_2, \dots, x_n$ .
- $\mathcal{X}$ : conjunto factível ou domínio, possui todas as soluções possíveis para o problema.
- $f(x)$ : função objetivo, a qual determinará o critério de escolha da solução.



- Factível.
  - Ótima.
  - Problema ilimitado.
- Problema infactível.



- └ Conceitos básicos
  - └ Tipos de soluções
    - └ Tipos de soluções

- Factível.
  - Ótima.
  - Problema ilimitado.
- Problema infactível.

- Factível: satisfaz todas as restrições do problema.
- Ótima: melhor solução factível.
- Problema ilimitado: não é possível encontrar uma solução ótima, ou seja, sempre é possível achar uma melhor.
- Problema infactível: quando o problema não possui solução, geralmente devido a muitas restrições.

# Conceitos básicos

Modelo contínuo  $\times$  discreto

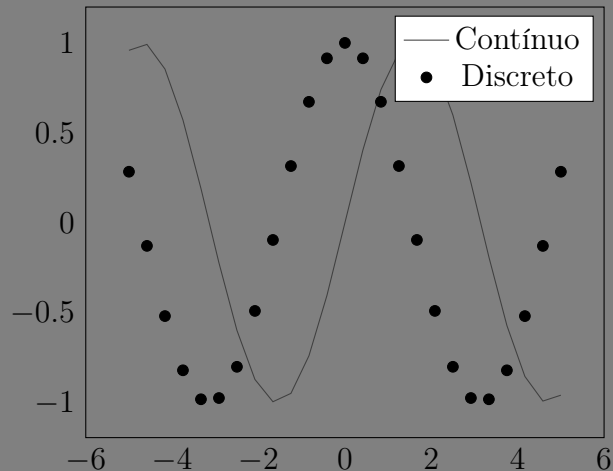


Figura: Exemplo de modelo contínuo e discreto.



2023-05-19

## Problemas de Empacotamento

└ Conceitos básicos

└ Tipos de soluções

└ Conceitos básicos

Conceitos básicos

Modelo contínuo  $\times$  discreto

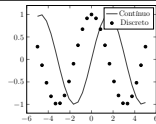


Figura: Exemplo de modelo contínuo e discreto.

Um modelo é contínuo quando sua região factível é contínua, ou seja, dado um ponto dessa região todos os seus vizinhos também serão uma solução.

Modelos discretos não possuem seu domínio contínuo.

Exatos	Heurísticos
<ul style="list-style-type: none"><li>• Solução ótima.</li><li>• Tempo.</li><li>• Recursos.</li></ul>	<ul style="list-style-type: none"><li>• Solução factível.</li><li>• Simplicidade.</li><li>• Grande porte.</li></ul>

### Exatos

- Solução ótima.
- Tempo.
- Recursos.

### Heurísticos

- Solução factível.
- Simplicidade.
- Grande porte.

Métodos exatos sempre vão garantir a solução ótima para o problema, porém encontrar tal solução pode requerer grande tempo e/ou muitos recursos computacionais.

Já heurísticas buscam por soluções factíveis e são geralmente usadas em problemas de grande porte.

O problema de interesse é NP-difícil, então buscar uma solução ótima fica praticamente inviável devido a limitações de tempo e recursos computacionais. Uma heurística será utilizada para obter uma solução boa em tempo hábil.

# Problema

Alocar peças em um espaço.

- Difícil resolução.
- $N$ -dimensional.
- Tipos de peças.
- Classificação.
- Variantes.



2023-05-19

## Problemas de Empacotamento

└ Problema

└ Problema

Problema

Alocar peças em um espaço.

- Difícil resolução.
- $N$ -dimensional.
- Tipos de peças.
- Classificação.
- Variantes.

A premissa do problema é simples, alocar peças em um espaço. Pode parecer algo bobo de resolver, mas é de difícil resolução já que pode possuir  $N$ -dimensões e diversos tipos de peças, de modo é preciso separar o problema em diferentes classes e ainda existem variantes dentro das classificações.

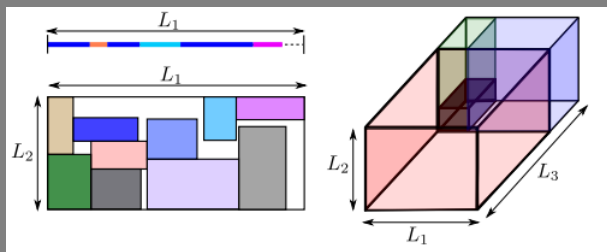


Figura: Representação 1D, 2D e 3D.

Fonte:(CASTELLUCCI, 2019)

└ Problema

└└  $N$ -dimensões

└└└  $N$ -dimensões

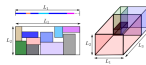


Figura: Representação 1D, 2D e 3D.

Fonte:(CASTELLUCCI, 2019)

- O caso 1D pode ser usado para empilhar caixas de mesma profundidade e largura.
- Já no 2D poderia ser aplicado em casos onde somente a profundidade é fixa.
- E o 3D seria alocar caixas em um depósito ou container.
- O trabalho se concentra somente no caso 2D.



# Problema

## Restrições

$$x_i \in \{0, \dots, W - w_i\}, y_i \in \{0, \dots, H - h_i\} \quad (i \in \mathcal{I}') \tag{1}$$
$$[x_i, x_i + w_i) \cap [x_j, x_j + w_j) = \emptyset \text{ ou } [y_i, y_i + h_i) \cap [y_j, y_j + h_j) = \emptyset \quad (i, j \in \mathcal{I}', i \neq j) \tag{2}$$



2023-05-19

Problemas de Empacotamento

- └ Problema
  - └ Restrições
    - └ Problema

Como já definimos a dimensão do problema, podemos ver as restrições do modelo.

A primeira restrição garante que um item só é alocado no recipiente se couber nele.

Já a segunda impede sobreposição entre as peças.

Problema

Restrições

$$x_i \in \{0, \dots, W - w_i\}, y_i \in \{0, \dots, H - h_i\} \quad (i \in \mathcal{I}') \tag{1}$$
$$[x_i, x_i + w_i) \cap [x_j, x_j + w_j) = \emptyset \text{ ou } [y_i, y_i + h_i) \cap [y_j, y_j + h_j) = \emptyset \quad (i, j \in \mathcal{I}', i \neq j) \tag{2}$$

# Tipos de peças

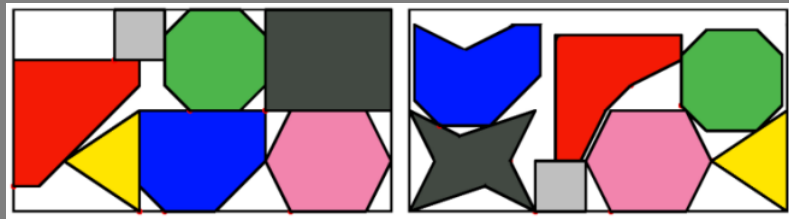


Figura: Exemplos de peças regulares (esquerda) e irregulares (direita).

Fonte:(BARTMEYER et al., 2021)

2023-05-19

## Problemas de Empacotamento

└ Problema

└ Tipos de peças

└ Tipos de peças

Tipos de peças



Figura: Exemplos de peças regulares (esquerda) e irregulares (direita).

Fonte:(BARTMEYER et al., 2021)

- Regulares: Possuem formato convexo.
- Irregulares: Possuem formato côncavo.
- Outra forma de se definir é checar se existe alguma reta que atravessasse o objeto em dois pontos diferentes.
- O trabalho foca em peças regulares retangulares.

- Empacotamento em faixa.

- Dado um conjunto de itens e uma caixa com comprimento fixo, queremos encontrar uma solução de altura mínima.

- Empacotamento em faixa.
- Empacotamento da mochila.

2023-05-19

## Problemas de Empacotamento

└ Problema

└ Classificação

└ Classificação

Classificação

- Empacotamento em faixa.
- Empacotamento da mochila.

- Dado um conjunto de itens e uma caixa com comprimento fixo, queremos encontrar uma solução de altura mínima.
- Nesse caso, queremos maximizar o valor da caixa (geralmente é a área da caixa).



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



└ Problema

└ Classificação

└ Classificação

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

- Dado um conjunto de itens e uma caixa com comprimento fixo, queremos encontrar uma solução de altura mínima.
- Nesse caso, queremos maximizar o valor da caixa (geralmente é a área da caixa).
- Minimizar o número de caixas necessárias para empacotar todos os itens.

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



2023-05-19

## Problemas de Empacotamento

└ Problema

└ Classificação

└ Classificação

Classificação

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

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

- Corte guilhotinado.

- Corte guilhotinado.

- Consiste em cortar a caixa de forma paralela a um dos lados de forma recursiva.

- Corte guilhotinado.
- Rotações ortogonais.



└ Problema

└ Variantes

└ Variantes

- Consiste em cortar a caixa de forma paralela a um dos lados de forma recursiva.
- É um modo de relaxar o problema, permitindo rotações de  $90^\circ$  nos itens.

- Corte guilhotinado.
- Rotações ortogonais.



- Corte guilhotinado.
- Rotações ortogonais.
- Restrições de carga e descarga.



- Corte guilhotinado.
- Rotações ortogonais.
- Restrições de carga e descarga.

- Consiste em cortar a caixa de forma paralela a um dos lados de forma recursiva.
- É um modo de relaxar o problema, permitindo rotações de  $90^\circ$  nos itens.
- Algumas peças precisam ser posicionadas em certa posição ou próximas a outras.

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



└ Problema

└ Variantes

└ Variantes

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

- Consiste em cortar a caixa de forma paralela a um dos lados de forma recursiva.
- É um modo de relaxar o problema, permitindo rotações de  $90^\circ$  nos itens.
- Algumas peças precisam ser posicionadas em certa posição ou próximas a outras.
- Define que caixas não precisam ter o mesmo tamanho (aplicável somente para Empacotamento em Caixas).

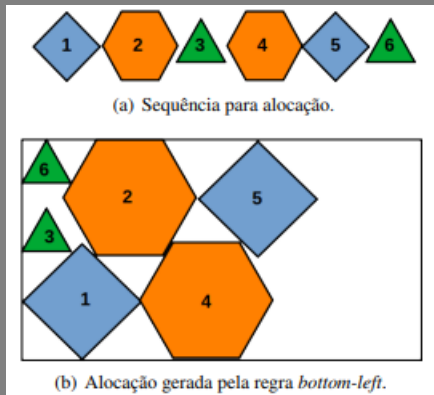


Figura: Representação de alocação.

Fonte:(BARTMEYER et al., 2021)



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ *Bottom-left*

Bottom-left

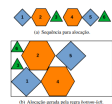


Figura: Representação de alocação.  
Fonte:(BARTMEYER et al., 2021)

Como o problema é NP-difícil uma heurística será usada e a *bottom-left* foi a escolhida.

Ela é bem simples, dado uma lista como entrada, os itens são retirados um a um e posicionados no ponto mais a baixo a 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 e é um ponto a ser resolvido. Como definir essa ordenação? Existe algum critério que se sobressai dos demais?

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



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Critérios de ordenação

└ Critérios de ordenação

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

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.

Cada critério pode ser usado de forma crescente ou decrescente. Com os critérios definidos, podemos passar para os próximos pontos do problema, que são a sobreposição e o domínio infinito.

# Sobreposição e domínio infinito

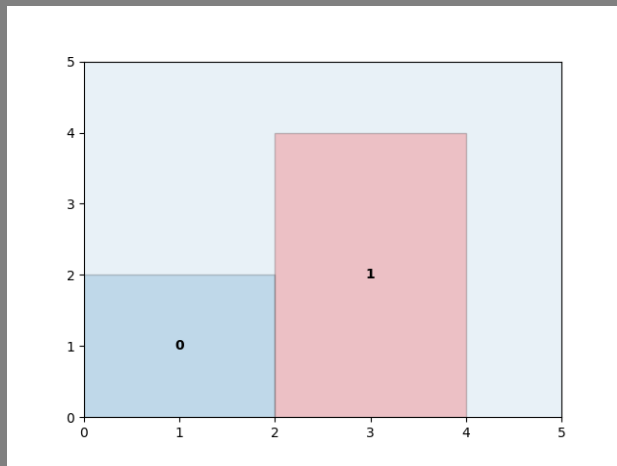


Figura: Resolvendo sobreposição e domínio infinito.



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Sobreposição e domínio infinito

└ Sobreposição e domínio infinito

Sobreposição e domínio infinito

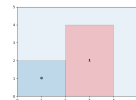


Figura: Resolvendo sobreposição e domínio infinito.

Supondo que estejamos em um estado do modelo 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.

# Sobreposição e domínio infinito

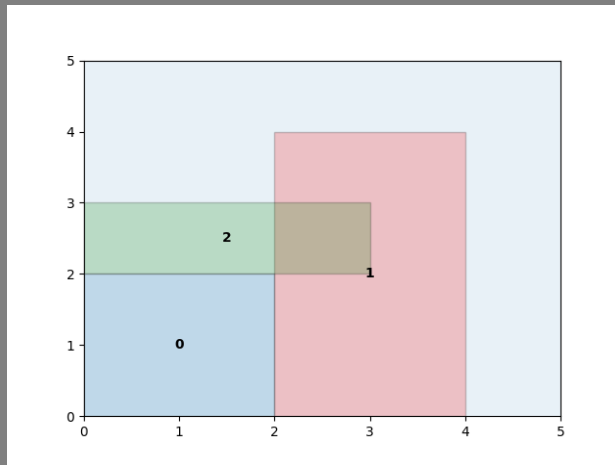


Figura: Resolvendo sobreposição e domínio infinito.



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Sobreposição e domínio infinito

└ Sobreposição e domínio infinito

Sobreposição e domínio infinito

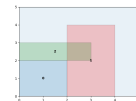


Figura: Resolvendo sobreposição e domínio infinito.

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.

# Sobreposição e domínio infinito

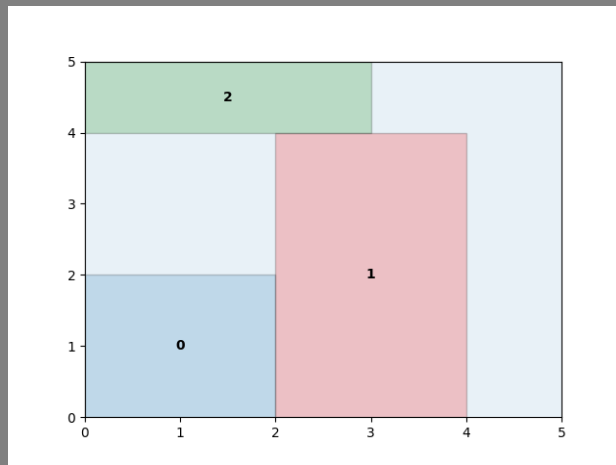


Figura: Resolvendo sobreposição e domínio infinito.



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Sobreposição e domínio infinito

└ Sobreposição e domínio infinito

Sobreposição e domínio infinito

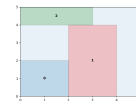


Figura: Resolvendo sobreposição e domínio infinito.

Nesse caso, com poucas peças, com caixa pequena e um auxílio visual é fácil dizer que a posição  $(0, 4)$  é válida, mas como chegar até ela? Existem infinitos pontos entre as coordenadas  $(0, 2)$  e  $(0, 4)$ .

# Sobreposição e domínio infinito

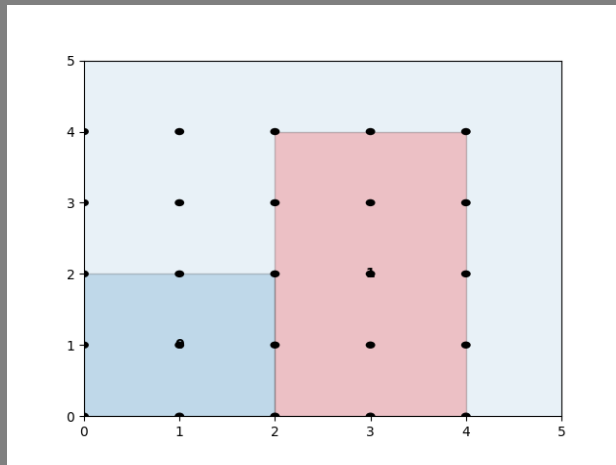


Figura: Resolvendo sobreposição e domínio infinito.



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Sobreposição e domínio infinito

└ Sobreposição e domínio infinito

Sobreposição e domínio infinito

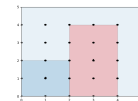


Figura: Resolvendo sobreposição e domínio infinito.

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



# Sobreposição e domínio infinito

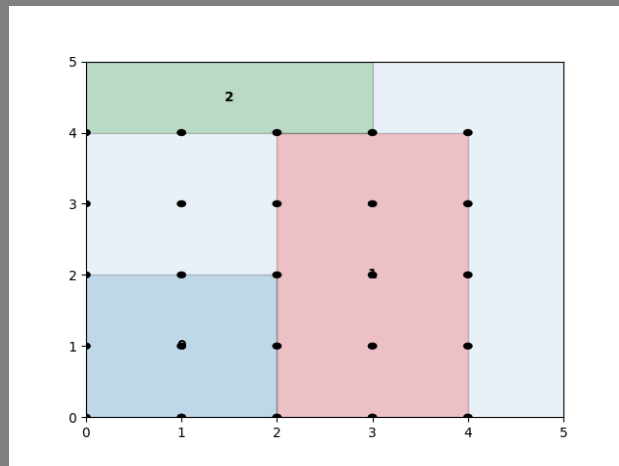


Figura: Resolvendo sobreposição e domínio infinito.



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Sobreposição e domínio infinito

└ Sobreposição e domínio infinito

Sobreposição e domínio infinito



Figura: Resolvendo sobreposição e domínio infinito.

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 instâncias 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 diferentes, com valores não inteiros, prejudicando a aplicação em vários problemas do mundo real.

- Vertical.
- Horizontal.
- $\max(\text{área})$ .
- Nenhuma.



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Regiões

└ Regiões

Regiões

- Vertical.
- Horizontal.
- $\max(\text{área})$ .
- Nenhuma.

Ambos os problemas, de sobreposição e de domínio infinito, podem ser resolvidos utilizando a estratégia de criação de regiões. Utilizando essa técnica é possível ignorar a restrição 2. 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.

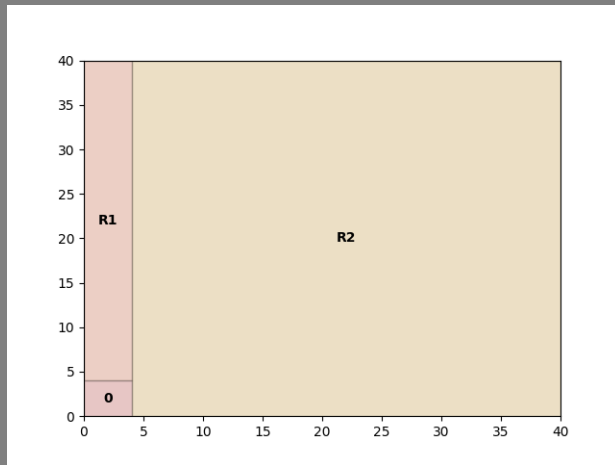
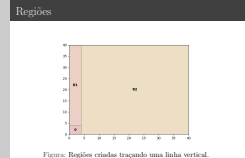


Figura: Regiões criadas traçando uma linha vertical.

A primeira é traçando uma linha vertical a partir do canto superior direito de cada peça alocada. Nas figuras, retângulos indicados com um R no começo são regiões.

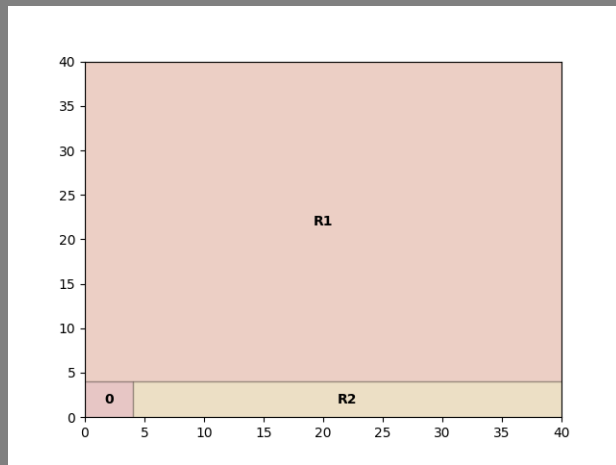


Figura: Regiões criadas traçando uma linha horizontal.



2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Regiões

└ Regiões

Regiões

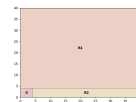


Figura: Regiões criadas traçando uma linha horizontal.

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

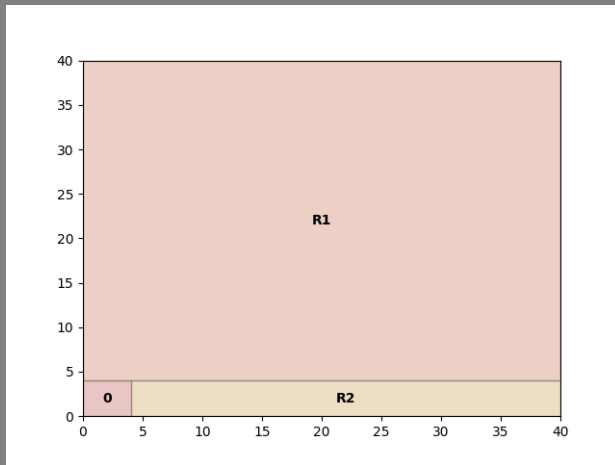


Figura: Regiões criadas maximizando uma das regiões.



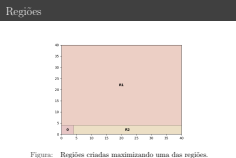
2023-05-19

## Problemas de Empacotamento

└ *Bottom-left*

└ Regiões

└ 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 maior área. Isso é interessante 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.

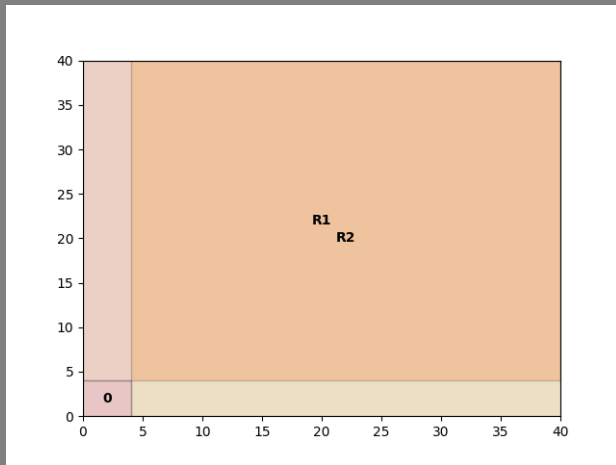


Figura: Regiões criadas possibilitando sobreposição.



2023-05-19

## Problemas de Empacotamento

└ Bottom-left

└ Regiões

└ Regiões

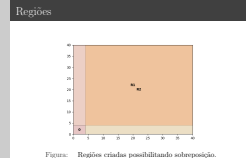


Figura: Regiões criadas possibilitando sobreposição.

No último modo nenhuma linha é traçada, todas as regiões vão até o final do recipiente. Nesse caso sobreposições de peças podem ocorrer, então verificações são necessárias para cumprir a restrição 2. Teoricamente ao permitir sobreposições possibilita que mais peças sejam alocadas. Esse modo foi criado justamente para verificar isso e qual seu custo.

- 45 instâncias.
  - BKW.
  - GCUT.
  - NGCUT.
  - OF.
  - OKP.
- 5 testes por configuração.
- $45 \cdot 5 \cdot 2 \cdot 4 \cdot 5 = 9000$  execuções.
- $\pm 5$  horas.



└ *Bottom-left*

└ Testes

└ Testes

- 45 instâncias.
  - BKW.
  - GCUT.
  - NGCUT.
  - OF.
  - OKP.
- 5 testes por configuração.
- $45 \cdot 5 \cdot 2 \cdot 4 \cdot 5 = 9000$  execuções.
- $\pm 5$  horas.

Para testar os métodos de solução criados foram usados 5 conjuntos de instâncias: BKW, GCUT, NGCUT, OF e OKP, totalizando 45 instâncias de teste.

Cada método foi executado 5 vezes em cada uma das instâncias para se obter uma média, também foi calculado a mediana e desvio padrão.

Como temos 45 instâncias, 5 critérios de ordenação, cada critério pode ser crescente ou decrescente, 4 formas de criar regiões e cada uma dessas combinações foi executada 5 vezes, temos o total de 9000 execuções.

O tempo somado de todas as execuções foi de aproximadamente 5 horas (valor que ainda será alterado, pois falta rodar a maior instância com o método de solução mais demorado).

Comparativo

Ordenação

Tabela: Comparativo entre ordenação crescente e decrescente.

Desc.	Wons	Draws	Quality %	Items %	Time (s)
F	167	8	57.3060	47.6518	2.3715e+00
T	736	8	78.9136	46.3642	1.7798e+00

Tabela: Comparativo entre ordenação crescente e decrescente.

Desc.	Wons	Draws	Quality %	Items %	Time (s)
F	167	8	57.3060	47.6518	2.3715e+00
T	736	8	78.9136	46.3642	1.7798e+00

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.





# Comparativo

## Ordenação

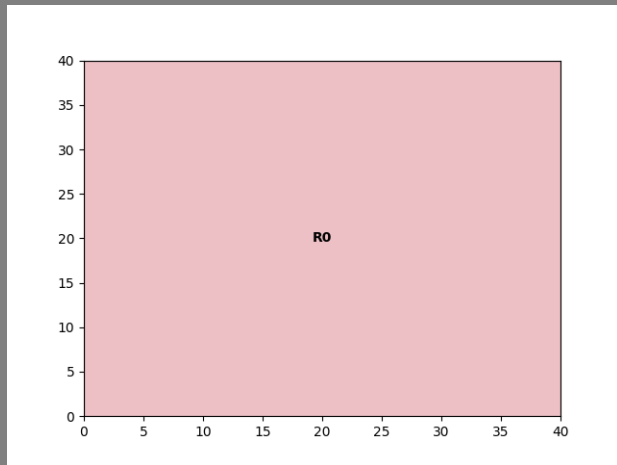


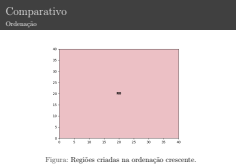
Figura: Regiões criadas na ordenação crescente.



2023-05-19

## Problemas de Empacotamento

- Resultados
  - Comparativo - Ordenação
    - Comparativo



Isso se deve a como as regiões são criadas, as figuras mostram o caso para ordenação crescente com a altura como critério e linha horizontal para criar a região.

# Comparativo

## Ordenação

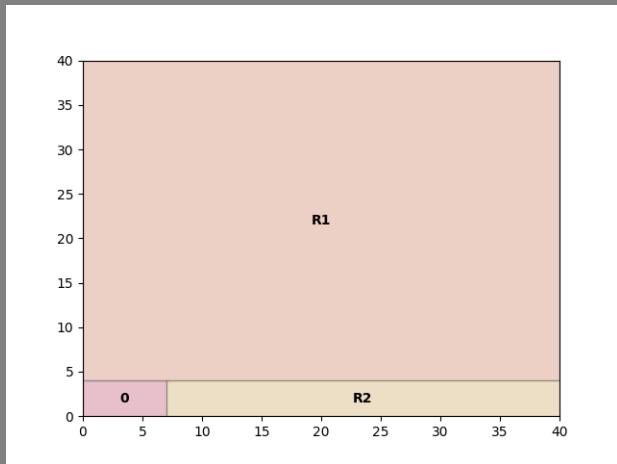


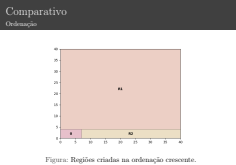
Figura: Regiões criadas na ordenação crescente.



2023-05-19

## Problemas de Empacotamento

- Resultados
  - Comparativo - Ordenação
    - Comparativo



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.

# Comparativo

## Ordenação

2023-05-19

### Problemas de Empacotamento

- Resultados
  - Comparativo - Ordenação
    - Comparativo

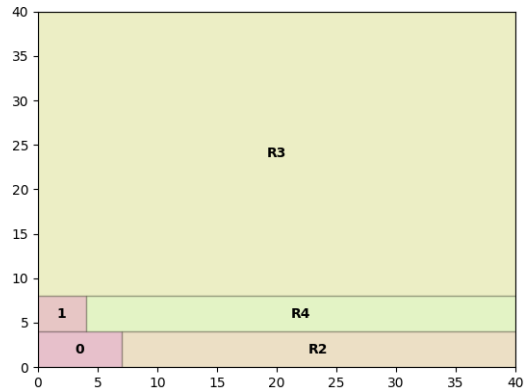
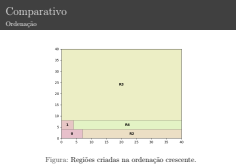


Figura: Regiões criadas na ordenação crescente.



Fazendo com que muitas regiões fiquem sem poder receber peças.

# Comparativo

## Ordenação

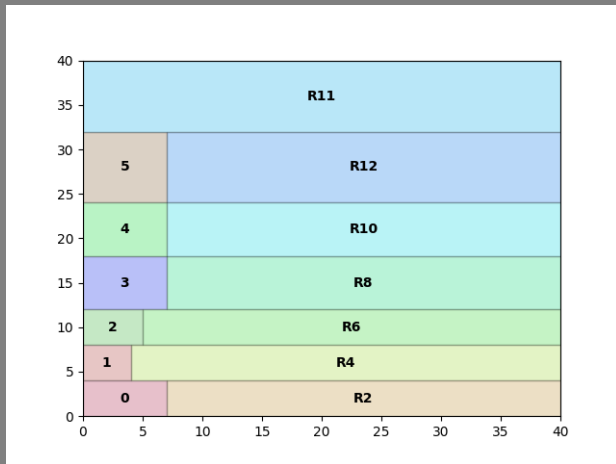
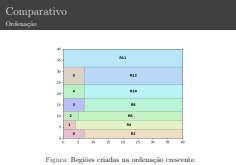


Figura: Regiões criadas na ordenação crescente.



2023-05-19

Problemas de Empacotamento  
└ Resultados  
└ Comparativo - Ordenação  
└ Comparativo



Essa figura mostra o estado final do modelo e grande parte do espaço ainda está livre. Algo semelhante ocorre com outros critérios de ordenação e criação regiões.

Comparativo

Critérios de ordenação

Tabela: Resultado para os critérios de ordenação.

OrderKey	Wons	Draws	Quality %	Items %	Time (s)
A	63	39	82.7353	44.0979	1.5874e+00
P	71	38	84.6986	44.8012	1.5769e+00
H	40	16	77.4182	46.3004	1.5655e+00
W	66	24	81.1899	47.6751	2.0805e+00
I	16	5	68.5261	48.9461	2.0889e+00

Tabela: Resultado para os critérios de ordenção.

OrderKey	Wons	Draws	Quality %	Items %	Time (s)
A	63	39	82.7353	44.0979	1.5874e+00
P	71	38	84.6986	44.8012	1.5769e+00
H	40	16	77.4182	46.3004	1.5655e+00
W	66	24	81.1899	47.6751	2.0805e+00
I	16	5	68.5261	48.9461	2.0889e+00

Os próximos resultados consideram somente os casos com ordenação decrescente, já que se fosse considerado a ordenação crescente faria com que a média ficasse abaixo do resultado real, além de poder causar interpretações erradas na coluna de quantidade de vitórias. Aqui fica claro que ter algum critério de ordenação melhora e muito na solução, já que ordenar 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, esses resultados podem indicar que algumas instâncias possuem características que torne mais interessante outro método de ordenação.



Comparativo

Regiões

Tabela: Comparativo entre criação de regiões.

SplitMode	Wons	Draws	Quality %	Items %	Time (s)
V	98	79	76.4030	45.0191	2.7157e-03
H	70	60	75.9970	45.5439	6.2101e-03
M	104	89	79.7175	47.6795	1.3743e-02
N	176	119	83.6420	47.2335	7.2176e+00

Tabela: Comparativo entre criação de regiões.

SplitMode	Wons	Draws	Quality %	Items %	Time (s)
V	98	79	76.4030	45.0191	2.7157e-03
H	70	60	75.9970	45.5439	6.2101e-03
M	104	89	79.7175	47.6795	1.3743e-02
N	176	119	83.6420	47.2335	7.2176e+00

Indo para o comparativo entre regiões percebemos que a que permite sobreposições se saiu melhor, tanto em quantidade como em qualidade, ainda que na maioria dos casos não foi a única que encontrou a melhor solução, porém com um custo autíssimo de tempo. Regiões criadas com linhas verticais e horizontais foram mais rápidas, mas com soluções de pior qualidade. Enquanto maximizando as regiões criadas 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. Mas por que tanta diferença entre com e sem sobreposição?



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



2023-05-19

## Problemas de Empacotamento

- └ Resultados
  - └ Comparativo - Ordenação
    - └ Comparativo

Comparativo  
Regiões

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

Como dito antes, sem sobreposições temos somente que checar se um item cabe em uma região, no pior caso teremos que fazer isso para  $(n^2 + n)/2$  regiões. 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 o somatório de  $(n^3 - n)/3$ , isso no pior caso, algo extremamente 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.

# Comparativo

Melhores combinações de solução

- 2023-05-19
- Problemas de Empacotamento
    - Resultados
      - Comparativo - Ordenação
        - Comparativo

Tabela: Resultados da comparação entre todas combinações.

R	O	Desc.	W	D	Quality %	Items %	Time (s)
V	W	T	9	8	84.5497	47.0580	0.0024
M	P	T	7	6	85.8682	46.3078	0.0129
N	A	T	13	11	85.3558	42.9123	6.2918
N	W	T	16	10	84.0171	48.1854	8.2509



Comparativo

Melhores combinações de solução

Tabela: Resultados da comparação entre todas combinações.

R	O	Desc.	W	D	Quality %	Items %	Time (s)
V	W	T	9	8	84.5497	47.0580	0.0024
M	P	T	7	6	85.8682	46.3078	0.0129
N	A	T	13	11	85.3558	42.9123	6.2918
N	W	T	16	10	84.0171	48.1854	8.2509

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 ao maximizar uma região e com ordenação pelo perímetro. A combinação de regiões com sobreposição e ordenação por área ficou bem próxima, mas consirando o custo-benefício não vale a pena. 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.



Tabela: Resultados para os conjuntos de instância.

InstanceSet	Quality %	Items %	Time (s)
BKW	79.4939	86.6104	6.2457e+00
GCUT	73.3823	17.3529	1.2040e-04
NGCUT	79.3747	46.2721	2.7272e-04
OF	77.8973	29.3071	5.3137e-04
OKP	86.8768	23.8969	1.0189e-03

Tabela: Resultados para os conjuntos de instância.

InstanceSet	Quality %	Items %	Time (s)
BKW	79.4939	86.6104	6.2457e+00
GCUT	73.3823	17.3529	1.2040e-04
NGCUT	79.3747	46.2721	2.7272e-04
OF	77.8973	29.3071	5.3137e-04
OKP	86.8768	23.8969	1.0189e-03

Nessa última 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.



- Múltiplos métodos de solução.
- Resultados inesperados.
- Com sobreposição  $\times$  sem sobreposição.



- Múltiplos métodos de solução.
- Resultados inesperados.
- Com sobreposição  $\times$  sem sobreposição.

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 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.

ARENALES, Marcos et al. **Pesquisa Operacional**. [S.l.]: Elsevier, 2007.

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>>.

BELLUZZO, Luciano; MORABITO, Reinaldo. Otimização nos padrões de corte de chapas de fibra de madeira reconstituída: um estudo de caso. **Pesquisa Operacional**, SciELO Brasil, v. 25, p. 391–415, 2005. Disponível em: <<https://www.scielo.br/j/pope/a/tTXXckvGTHbDfZQkmzCqdkp>>.



2023-05-19

## Problemas de Empacotamento

### └─ Conclusão

### └─ Referências

ARENALES, Marcos et al. **Pesquisa Operacional** [S.l.]: Elsevier, 2007.

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>>.

BELLUZZO, Luciano; MORABITO, Reinaldo. Otimização nos padrões de corte de chapas de fibra de madeira reconstituída: um estudo de caso. **Pesquisa Operacional**, SciELO Brasil, v. 25, p. 391–415, 2005. Disponível em: <<https://www.scielo.br/j/pope/a/tTXXckvGTHbDfZQkmzCqdkp>>.

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>>.

CAVALI, Roberto. Problemas de corte e empacotamento na indústria de Móveis: um estudo de caso. Universidade Estadual Paulista (Unesp), 2004. Disponível em: <[https://repositorio.unesp.br/bitstream/handle/11449/94286/cavali\\_r\\_me\\_sjrp.pdf](https://repositorio.unesp.br/bitstream/handle/11449/94286/cavali_r_me_sjrp.pdf)>.



IORI, Manuel; DE LIMA, Vinícius L. et al. Exact solution techniques for two-dimensional cutting and packing. **European Journal of Operational Research**, v. 289, n. 2, p. 399–415, 2021. ISSN 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2020.06.050>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0377221720306111>>.

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



MORABITO NETO, Reinaldo; WIDMER, Joao Alexandre. **Abordagem em grafo-e-ou para o problema do empacotamento: aplicacao ao carregamento de paletes e containeres**. 1992. Tese (Doutorado). Disponível em: <<https://repositorio.usp.br/item/000734666>>.

QUEIROZ, Layane Rodrigues de Souza. **Estudo de problemas de corte de itens irregulares com incertezas**. 2022. Tese (Doutorado) – Universidade de São Paulo. Disponível em: <<https://www.teses.usp.br/teses/disponiveis/55/55134/tde-10032022-110656/en.php>>.

