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

24 de junho de 2023

2023-06-24

Empacotamento de retângulos

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
24 de junho de 2023

Meu nome é Gabriel e hoje vou apresentar uma prévia do meu tcc.
O trabalho trata sobre a avaliação demétodos de solução baseados em *bottom-left* para problema de empacotamento. Ele foi feito sob orientação do professor Pedro e teve coorientação do professor Rafael.

Sumário

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

Sumário

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

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.

Problema

Alocar peças em um espaço.

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

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 que é preciso separar o problema em diferentes classes e ainda existem variantes dentro das classificações.

Problema

N-dimensões

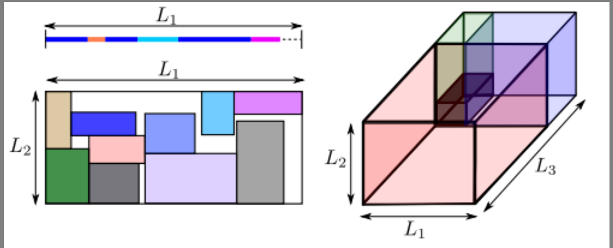
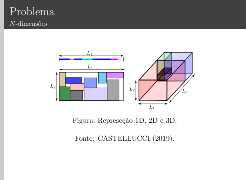


Figura: Represeção 1D, 2D e 3D.

Fonte: CASTELLUCCI (2019).



Como eu disse, o problema pode ter N-dimensões, aqui vou citar alguns exemplos.

- 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

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

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

Problema
Restrições
$x_i \in \{0, \dots, W - w_i\}, y_i \in \{0, \dots, H - h_i\} \left(i \in \mathcal{I}'\right)$ (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)

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.

Bottom-left

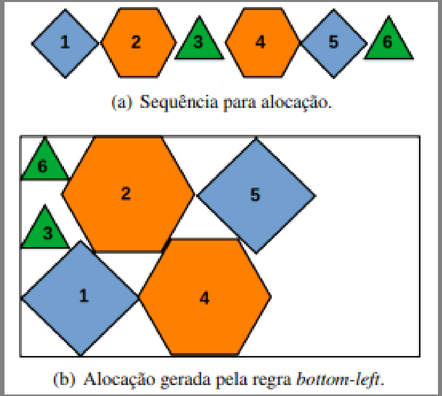
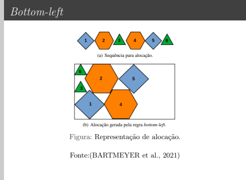


Figura: Representação de alocação.

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. Mas como definir essa ordenação? Existe algum critério que se sobressai dos demais?

Bottom-left

Critérios de ordenação

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

Bottom-left
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.

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.

Bottom-left

Sobreposição e domínio infinito

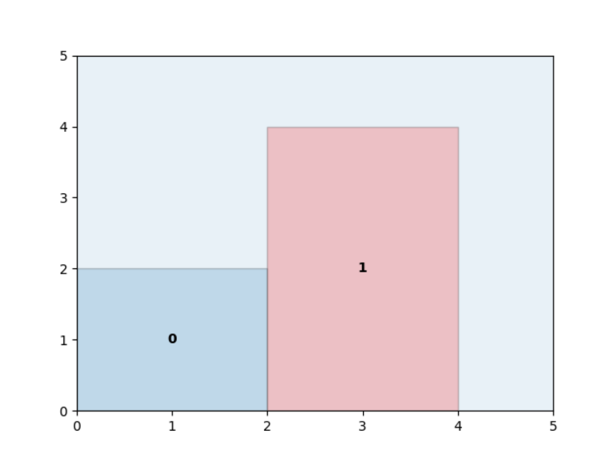
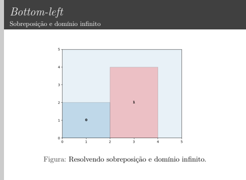


Figura: Resolvendo sobreposição e domínio infinito.



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.

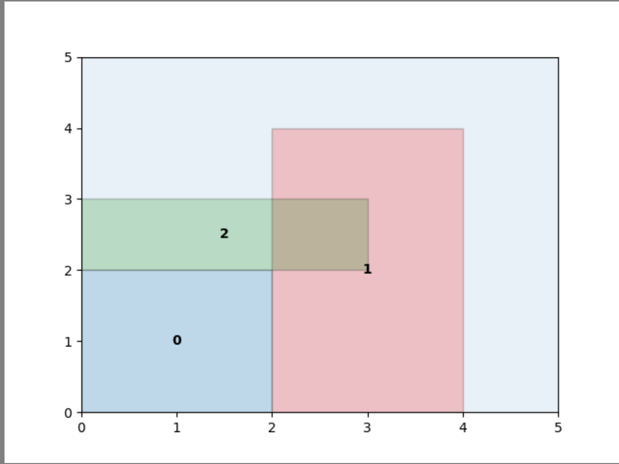
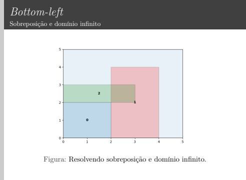


Figura: Resolvendo sobreposição e domínio infinito.



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.

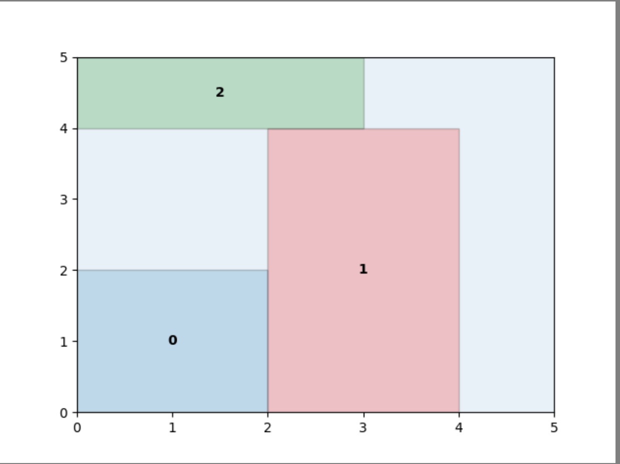
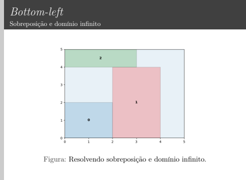


Figura: Resolvendo sobreposição e domínio infinito.



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).

Bottom-left

Sobreposição e domínio infinito

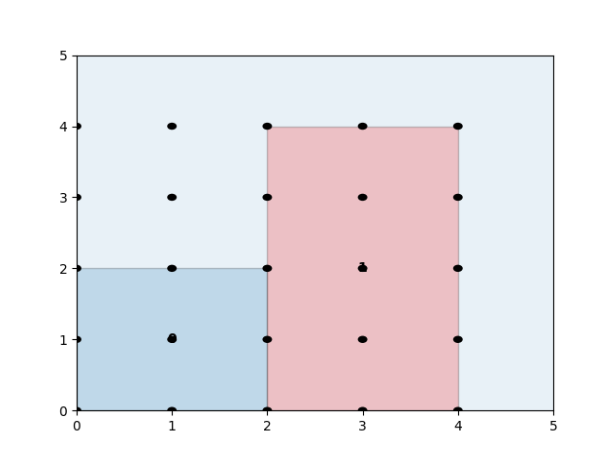
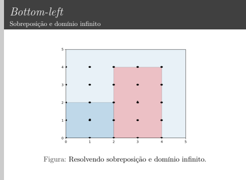


Figura: Resolvendo sobreposição e domínio infinito.



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

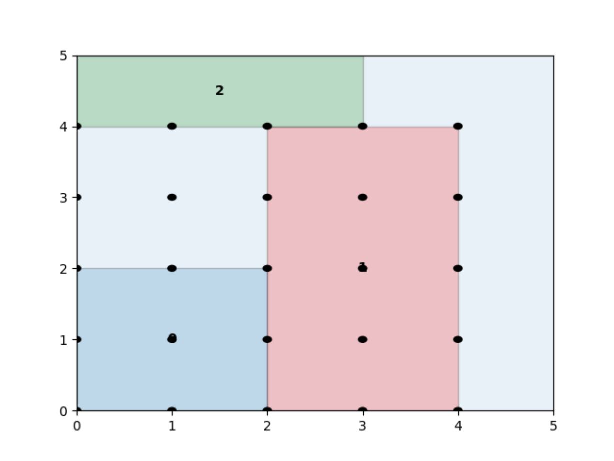
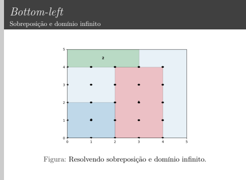


Figura: Resolvendo sobreposição e domínio infinito.



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.

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

Bottom-left
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 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.

Bottom-left

Regiões

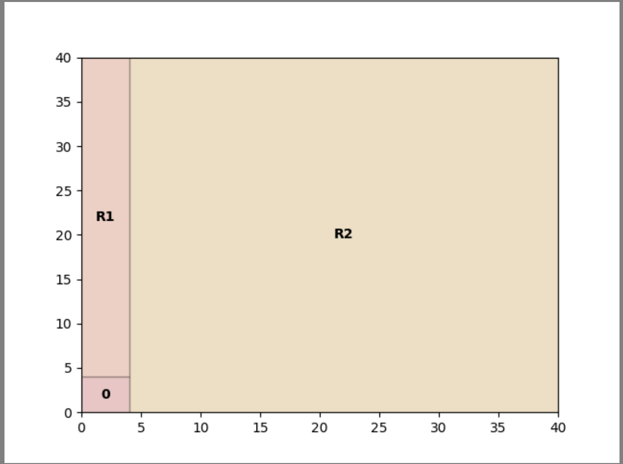
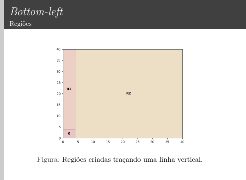


Figura: Regiões criadas traçando uma linha vertical.



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.

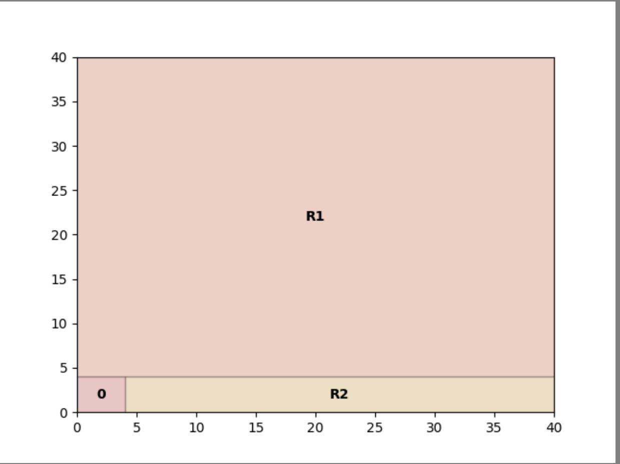
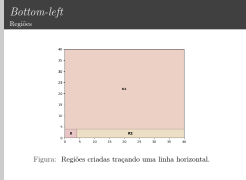


Figura: Regiões criadas traçando uma linha horizontal.



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

Bottom-left

Regiões

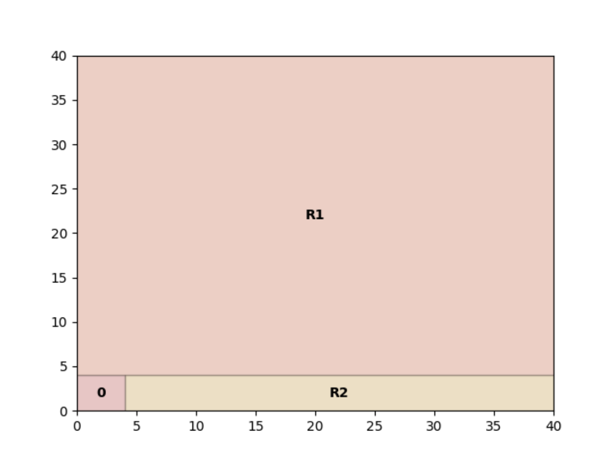
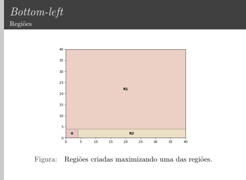


Figura: Regiões criadas maximizando uma das regiões.



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 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.

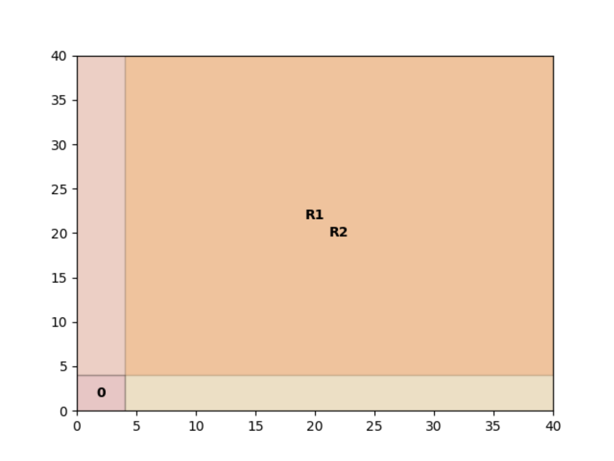
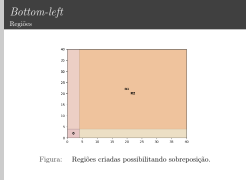


Figura: Regiões criadas possibilitando sobreposição.



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.

Bottom-left

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

2023-06-24

Bottom-left
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

Bottom-left

Testes

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

Bottom-left
Testes
- 45 instâncias.
  - BKW.
  - GCUT.
  - NGCUT.
  - OF.
  - OKP.
- 5 testes por configuração.
- $45 \cdot 5 \cdot 2 \cdot 4 \cdot 5 = 9000$ execuções.
- ± 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 um 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).

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

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

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.

Resultados

Ordenação

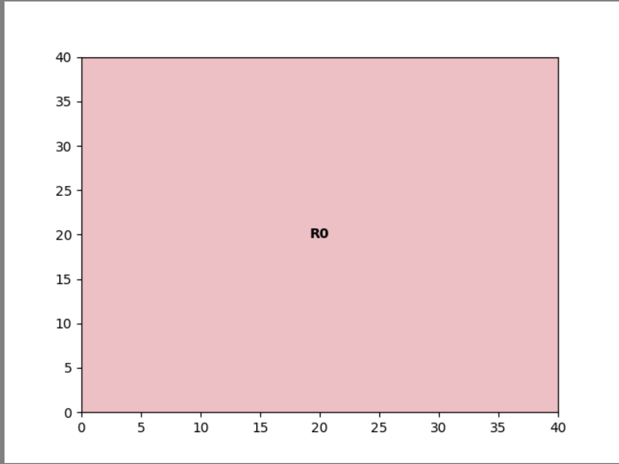
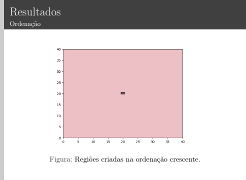


Figura: Regiões criadas na ordenação crescente.



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.

Resultados

Ordenação

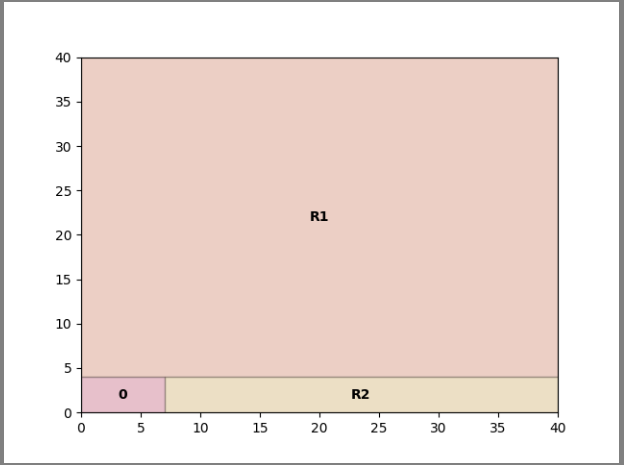
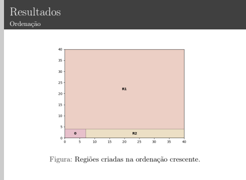


Figura: Regiões criadas na ordenação crescente.



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

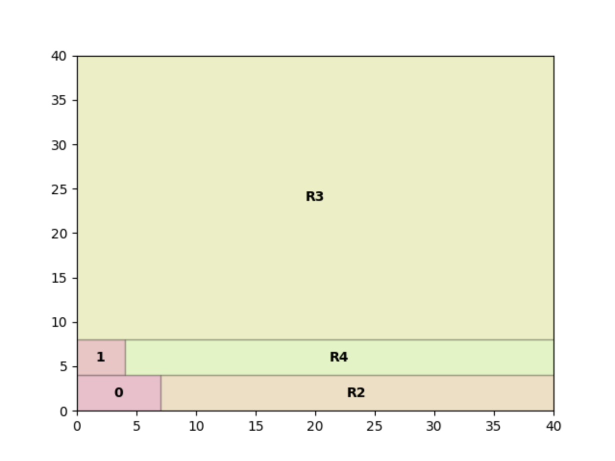
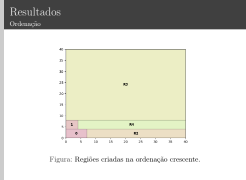


Figura: Regiões criadas na ordenação crescente.



Fazendo com que muitas regiões fiquem sem poder receber peças.

Resultados

Ordenação

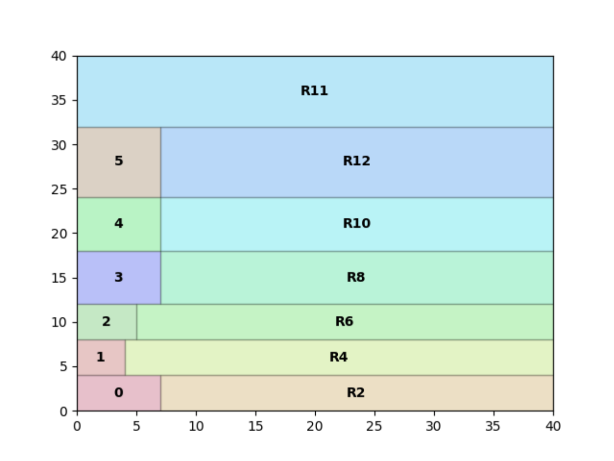
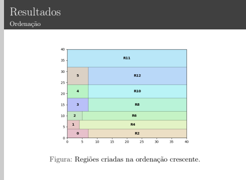


Figura: Regiões criadas na ordenação crescente.



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.

Resultados

Critérios de ordenação

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

Resultados
Critérios de ordenação
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

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 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, 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

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?

Resultados

Regiões

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

Resultados
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.

Tabela: Resultados para sobreposição.

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

Resultados
Sobreposição

Tabela: Resultados para sobreposição.

Superposição	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 com e sem sobreposição. Na primeira linha temos os resultados de todas as combinações possíveis com regiões sem sobreposição e o tempo total que levou para executar todas as instâncias. Já na segunda linha somente o critério de ordenação pela área foi considerado, 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 sem sobreposição e escolher o melhor resultado.

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>.

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

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>.

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

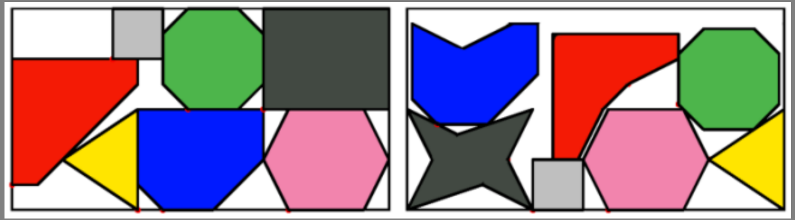
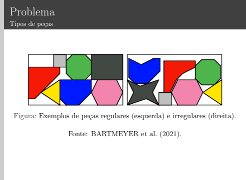


Figura: Exemplos de peças regulares (esquerda) e irregulares (direita).

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 atravesse o objeto em dois pontos diferentes, se sim, é irregular.
- O trabalho foca em peças regulares 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).

- 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.