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

16 de maio de 2023

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
16 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.

Sumário

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

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

Como nem todos podem estar familiarizados com alguns termos, vou fazer uma breve revisão de conceitos básicos.
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, apresentarei a conclusões que podem ser feitas a partir do trabalho.

Modelos de otimização

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

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

Modelos de otimização

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

- x: variável de decisão, $x = x_1, x_2, \ldots, x_n$.
- $\mathcal{X}$: 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, $x = x_1, x_2, \ldots, x_n$.
- $\mathcal{X}$: 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.

Tipos de soluções

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

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 × discreto

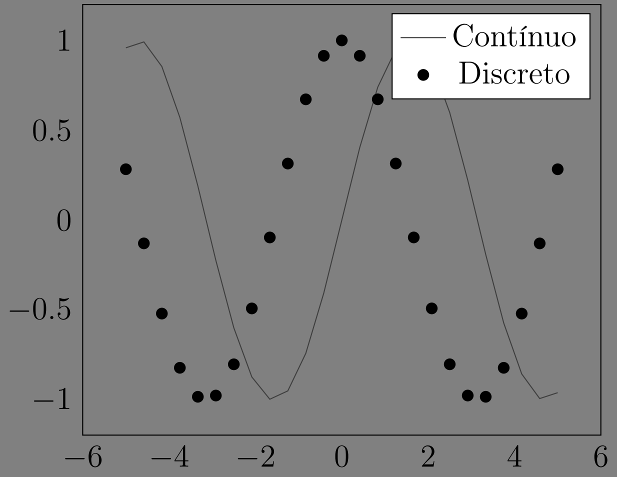
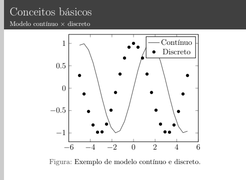


Figura: Exemplo de modelo contínuo e discreto.



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.

Métodos exatos × heurísticos

Exatos

- Solução ótima.
- Tempo.
- Recursos.

Heurísticos

- Solução factível.
- Simplicidade.
- Grande porte.

Métodos exatos × heurísticos
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.

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.

N-dimensões

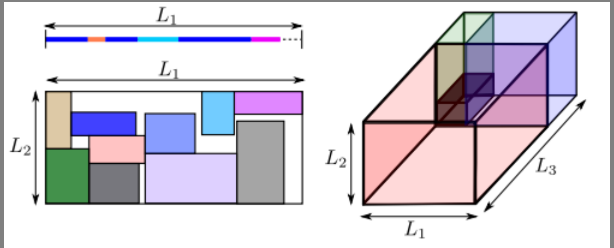
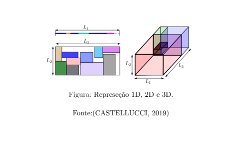


Figura: Represeção 1D, 2D e 3D.

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.

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

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

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

Tipos de peças

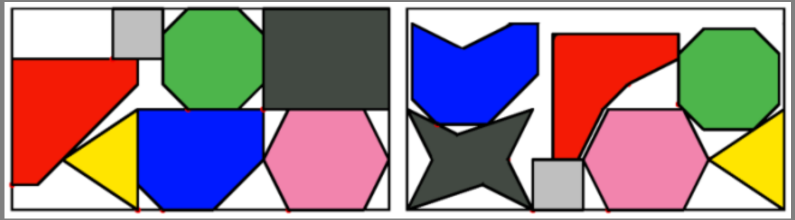
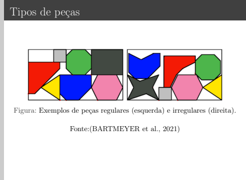


Figura: Exemplos de peças regulares (esquerda) e irregulares (direita).

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.
- O trabalho foca em peças regulares retangulares.

Classificação

- Empacotamento em faixa.

Classificação
• Empacotamento em faixa.

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

Classificação

- Empacotamento em faixa.
- Empacotamento da mochila.

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).

Classificação

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

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.

Classificação

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

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).

Variantes

- Corte guilhotinado.

Variantes
• Corte guilhotinado.

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

Variantes

- Corte guilhotinado.
- Rotações ortogonais.

Variantes
- Corte guilhotinado.
- Rotações ortogonais.

- 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° nos itens.

Variantes

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

Variantes

- 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° nos itens.
- Algumas peças precisam ser posicionadas em certa posição ou próximas a outras.

Variantes

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

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° 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).

Bottom-left

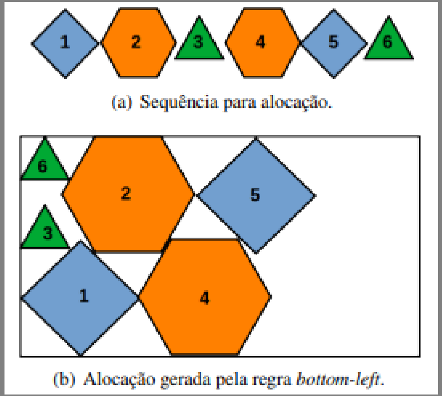
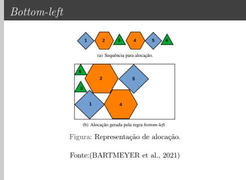


Figura: Representação de alocação.

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?

Critérios de ordenação

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

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

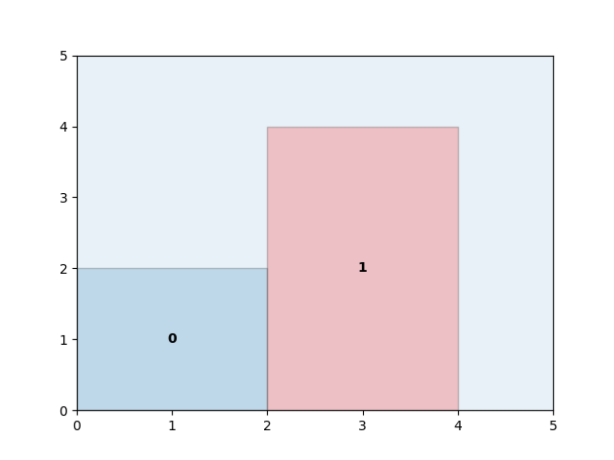
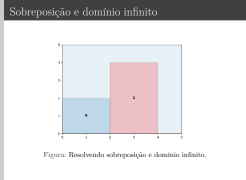


Figura: Resolvendo sobreposição e domínio infinito.



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

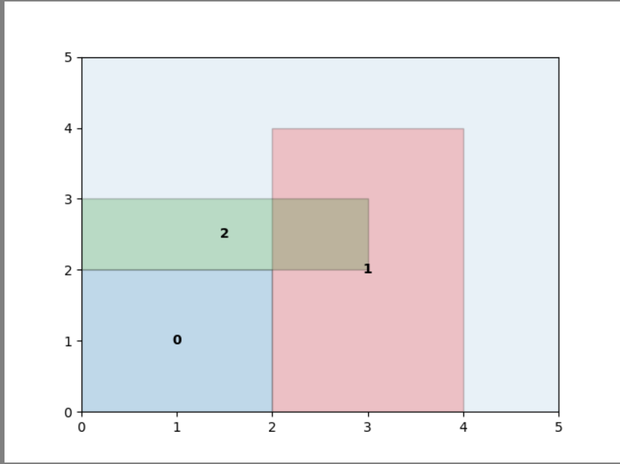
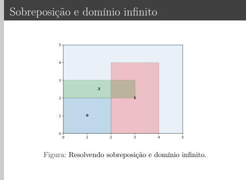


Figura: Resolvendo sobreposição e domínio infinito.



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

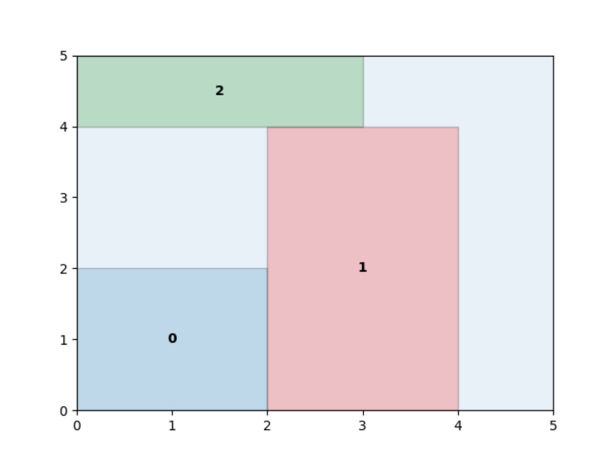
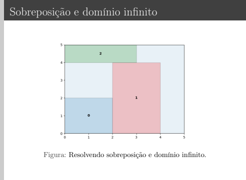


Figura: Resolvendo sobreposição e domínio infinito.



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

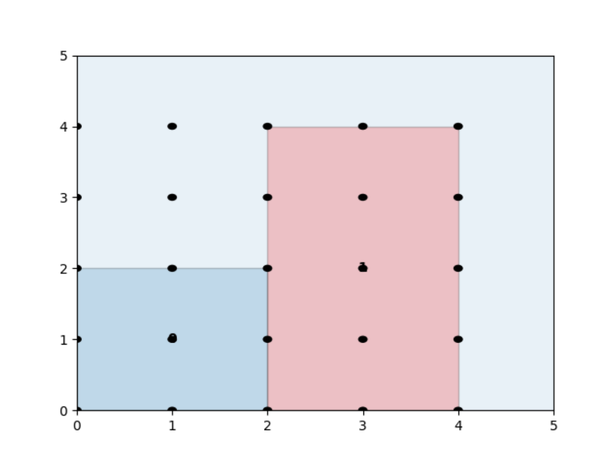
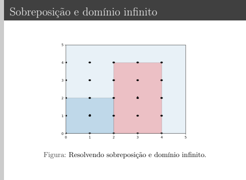


Figura: Resolvendo sobreposição e domínio infinito.



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

Sobreposição e domínio infinito

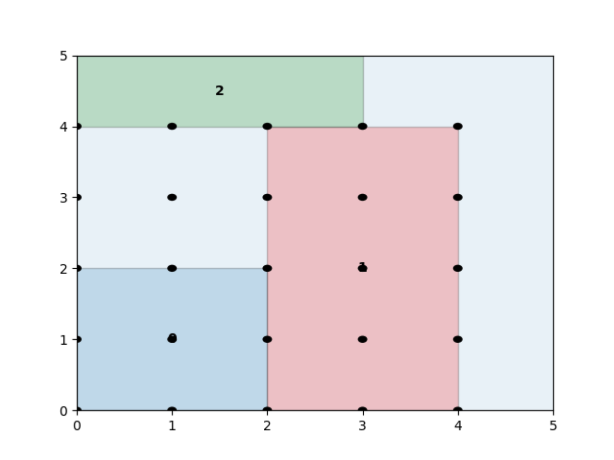
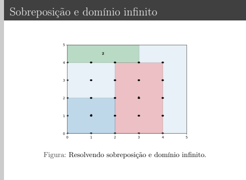


Figura: Resolvendo sobreposição e domínio infinito.



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 só funcionaria em casos como os das intâncias, com valores inteiros, não sendo aplicável em vários problemas do mundo real.

Regiões

- Vertical.
- Horizontal.
- max(área).
- Nenhuma.

Regiões

- Vertical.
- Horizontal.
- max(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.

Regiões

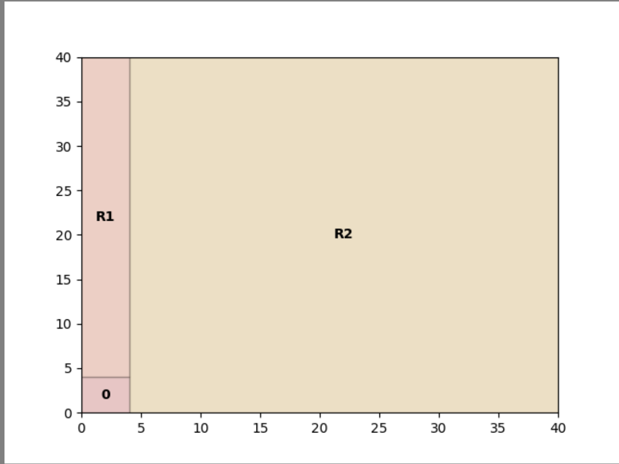
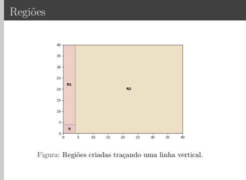


Figura: Regiões criadas traçando uma linha vertical.



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.

Regiões

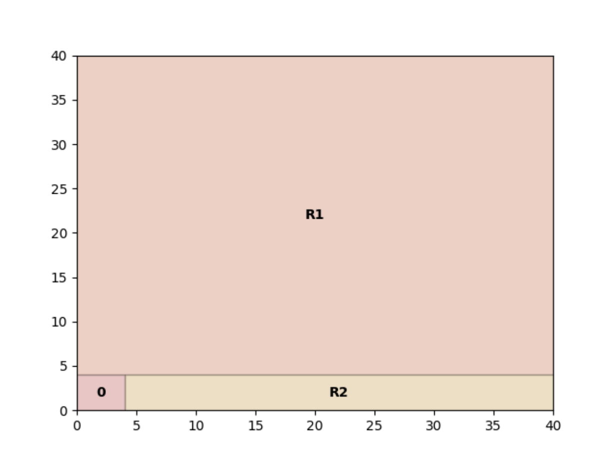
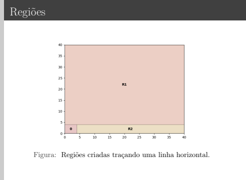


Figura: Regiões criadas traçando uma linha horizontal.



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

Regiões

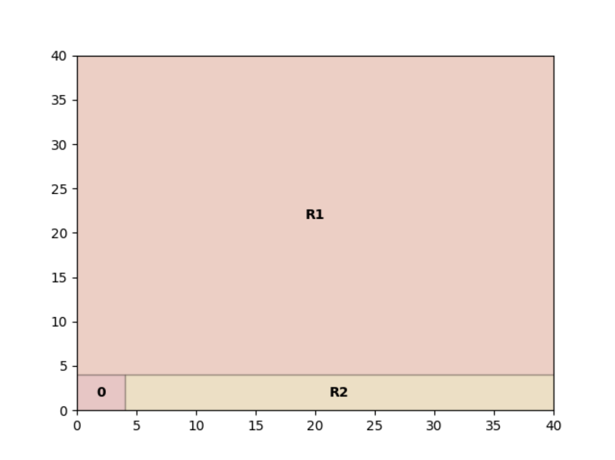
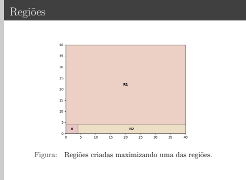


Figura: Regiões criadas maximizando uma das regiões.



Já na terceira é traçada uma linha (vertical ou horizontal) que maximize a área de uma das regiões geradas e com nenhuma linha, 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.

Regiões

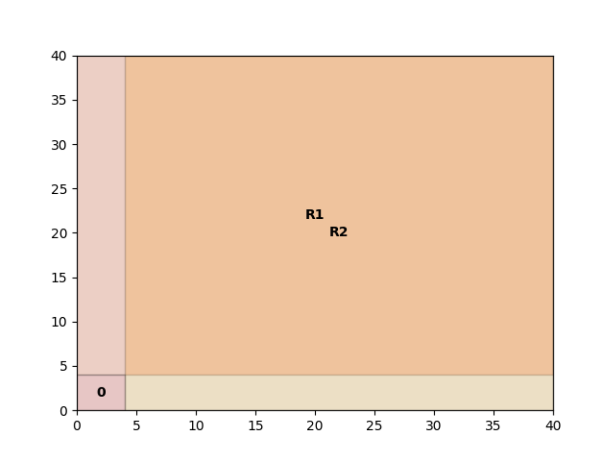
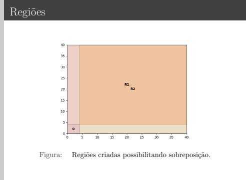


Figura: Regiões criadas possibilitando sobreposição.



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.

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.

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).

Comparativo
Ordenação

Tabela: Comparativo entre ordenação crescente e decrescente.

Descending	Wons	Draws	Quality %	Items %	Time (s)
F	167	8	57.306	47.6518	2.37153
T	736	8	78.9136	46.3642	1.77985

Comparativo
Ordenação

Tabela: Comparativo entre ordenação crescente e decrescente.

Descending	Wons	Draws	Quality %	Items %	Time (s)
F	167	8	57.306	47.6518	2.37153
T	736	8	78.9136	46.3642	1.77985

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

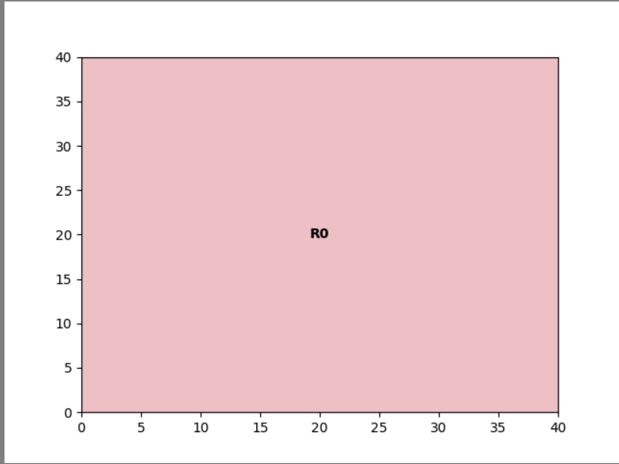
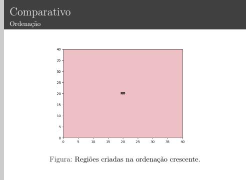


Figura: Regiões criadas na ordenação crescente.



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

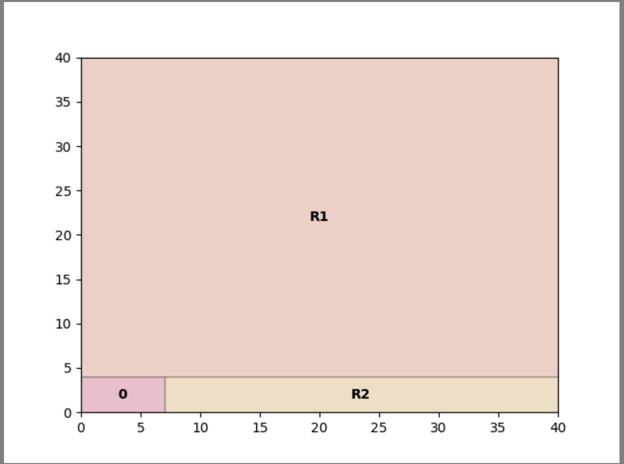
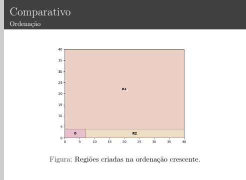


Figura: Regiões criadas na ordenação crescente.

UFSC



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

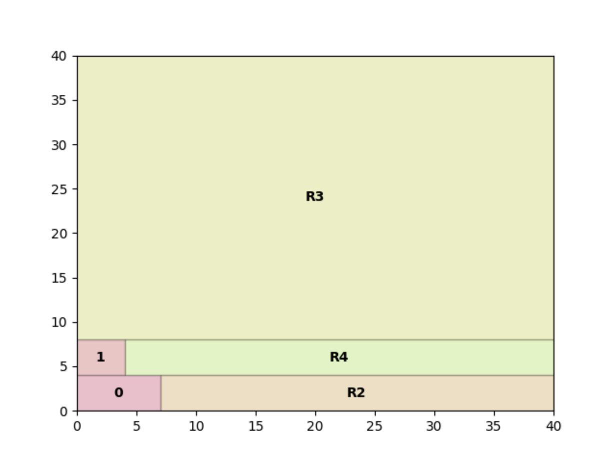


Figura: Regiões criadas na ordenação crescente.

2023-05-16

Comparativo
Ordenação

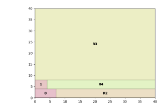


Figura: Regiões criadas na ordenação crescente.

Fazendo com que muitas regiões fiquem sem poder receber peças.

Comparativo

Ordenação

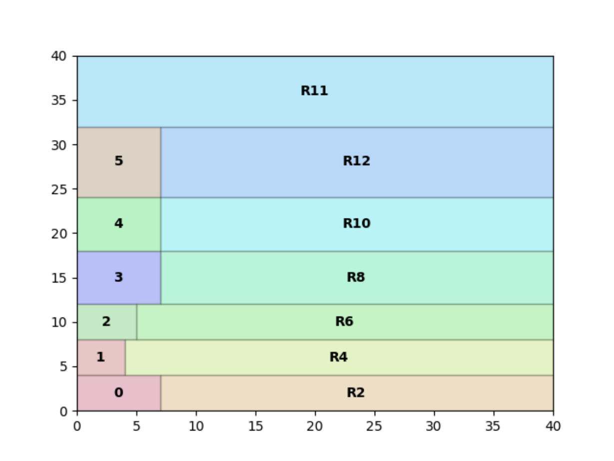
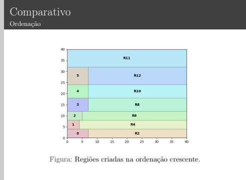


Figura: Regiões criadas na ordenação crescente.



Algo semelhante ocorre com outros critérios de ordenação e criação regiões.

Conclusão

- Resultados inesperados.
- Múltiplos métodos de solução.

2023-05-16
Problemas de Empacotamento
└─Conclusão

└─Conclusão

Conclusão

- Resultados inesperados.
- Múltiplos métodos de solução.

Referências

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>.

Referências

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>.