Universidade Federal Rural de Pernambuco - UFRPE
Departamento de Estatística e Informática - DEINFO
Bacharelado em Sistemas de Informação - BSI
Tópicos em Otimização

Documentação Código

Equipe:
Marianna Alves
Victor Olimpio

Professor:
Cláudio Cristino

Recife/PE
2024

[Maximização de Espaço 2D: Algoritmo para Armazenar Objetos]

Descrição

Dentro do contexto do armazenamento, a finalidade da função é otimizar a disposição de itens em um contêiner, considerando a área disponível e as dimensões de cada mercadoria. O objetivo é atingir a máxima eficiência, buscando a disposição que permita acomodar o maior número possível de objetos no espaço disponível, ao mesmo tempo em que define um mapa com a distribuição ótima dos itens dentro do contêiner.

No âmbito desse problema, a dimensão do volume não será levada em consideração. A abordagem se restringirá exclusivamente às dimensões de largura e altura, simplificando a resolução para um contexto bidimensional e linear. Será considerado apenas objetos de mesma dimensão.

Observando esse problema a partir da ótica da Programação Linear, em conjunto com uma pesquisa de referencial teórico de publicações que trataram do mesmo desafio, definimos nossa função objetivo da seguinte forma:

$$f(x, y) = \sum_{i=1}^{N} w_i \cdot h_i$$

onde
w_i é representa a largura do objeto
h_i é representa a altura do objeto
N é o número total de objetos **i**
O objetivo é encontrar os valores de **x e y** que maximizam essa função, sujeitos às restrições que definem as limitações do contêiner e as relações entre as coordenadas dos objetos. Se usarmos um sistema de coordenadas cartesianas, a coordenada x representaria a posição horizontal (largura) do canto inferior esquerdo do objeto, e a coordenada y representaria a posição vertical (altura) do mesmo ponto.

Instalação

O algoritmo foi desenvolvido na linguagem de programação Python. Para executar, basta usar uma IDE.

Pré-requisitos

- Python
- IDE (console) (Ex: vscode)

Link repositório com código: Modelo de Otimização para problema de empacotamento 2d

Estrutura

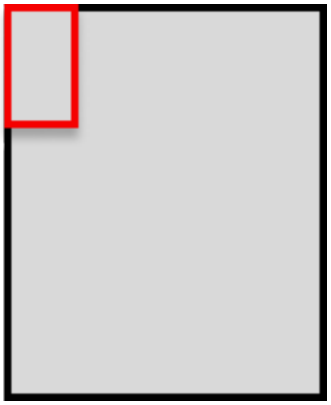
Entradas

- Através da função **get_dimensions**, o algoritmo pede a entrada de 4 variáveis: comprimento do container, largura do container, comprimento do objeto e largura do objeto.
  containerHeight, containerWidth, objectHeight, containerWidth
- É esperado pelo algoritmo que essas variáveis sejam numéricas e mais maiores que zero, atendendo a restrição de não negatividade do problema.
- Além da não negatividade, outra condição será verificada no início. Caso o valor do objeto (largura ou altura) exceda o valor do container (largura e altura), é retornado que não é possível alocar o objeto no container.

Testes de Layout

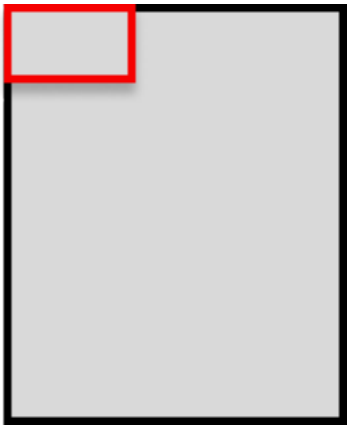
- A seguir, o objetivo é testar a entrada do objeto no que chamamos de layout 1 e 2, que consiste em, caso possível, testar a alocação dos objetos sob duas orientações:

    - Orientação onde a altura do objeto é paralela a altura do container.



Layout 1 (Imagem 1)

- Orientação com o comportamento contrário



Layout 2 (Imagem 2)

No entanto, há restrições quanto ao teste nos dois layouts. Dependendo do tamanho do objeto, é possível que só exista uma orientação possível para a alocação. Para esses casos, o código retorna zero indicando que para um tipo de orientação, o layout não funciona. Exemplo:

Imagem 3

Caso o formato objeto seja um quadrado, a orientação não faz diferença. Assim, o código só faz um tipo de investigação.

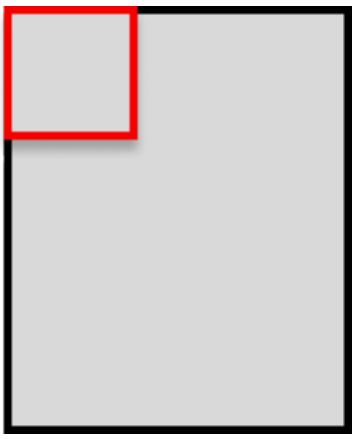


Imagem 4

Com o resultado dos testes, podemos dizer que, para o problema bidimensional, encontramos o que nós vamos chamar “organização ótima padrão”, ainda sem considerar possíveis sobras e que serão investigadas posteriormente. Em seguida, nas possíveis

sobras, o algoritmo avalia a possível alocação do objeto de forma diferente da orientação padrão do layout e gera o complemento da alocação.

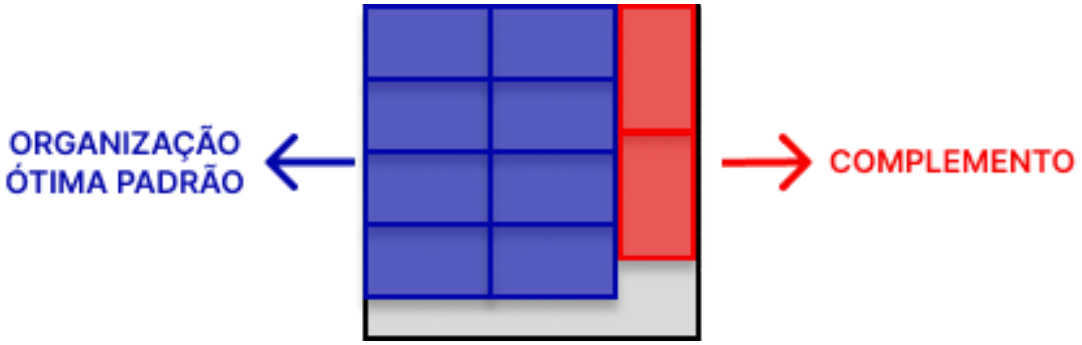


Imagem 5

No final, caso haja alocação possível no complemento, essa quantidade de partições encontrada é adicionada à quantidade de partições da orientação padrão. Na Imagem 5, foi considerado o seguinte exemplo.

containerHeight = 130
containerWidth = 120
objectHeight = 30
containerWidth = 50

Quantidade total de objetos no container: 10
Quantidade na organização padrão: 8
Quantidade no complemento: 2

Saída

Com base na Imagem 6, o resultado gerado pelo algoritmo inclui uma variável que indica o total de objetos, juntamente com uma matriz de layout que pode ser identificada como Layout 1 ou Layout 2. Nesse contexto, "Height" representa o número de linhas e "Width" o número de colunas na matriz, delineando assim a disposição otimizada dos objetos. Essa seria a organização ótima padrão. Caso ainda exista espaço não utilizado no contêiner, o algoritmo adiciona um layout complementar, mantendo o mesmo padrão do primeiro layout. O resultado visual dos layouts da Imagem 6 está representado na Imagem 7.

```
Digite a altura do container do item: 130
Digite a largura do container do item: 130
Digite a altura do objeto do item: 50
Digite a largura do objeto do item: 30
Total de objetos:  10
Layout 1-->  {'height': 2, 'width': 4} Altura do objeto paralela a altura da caixa
 Layout complementar-->  {'height': 1, 'width': 2}  Posição do item  contrária a orientação do layout 1
```

Imagem 6

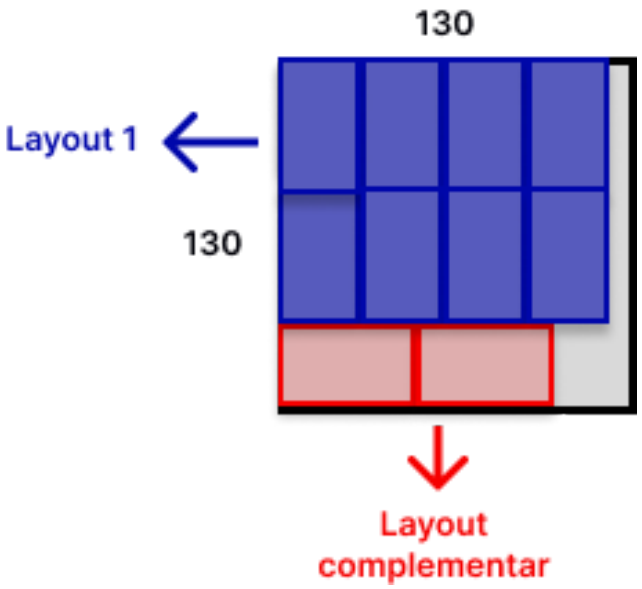


Imagem 7