

# Introdução Otimização Combinatória usando Linguagem de Programação Julia



Amanda Ferreira de Azevedo

## Definição Geral

$$\min \{cx : Ax \leq b, x \geq 0\}$$

## Definição Geral

$$\min \{cx : Ax \leq b, x \geq 0\}$$

- ▶ Função objetivo

## Definição Geral

$$\min \{cx : Ax \leq b, x \geq 0\}$$

- ▶ Função objetivo
- ▶ Restrições

## Definição Geral

$$\min \{cx : Ax \leq b, x \geq 0\}$$

- ▶ Função objetivo
- ▶ Restrições
- ▶ Variáveis de decisão

## Definição Geral

$$\min \{cx : Ax \leq b, x \geq 0\}$$

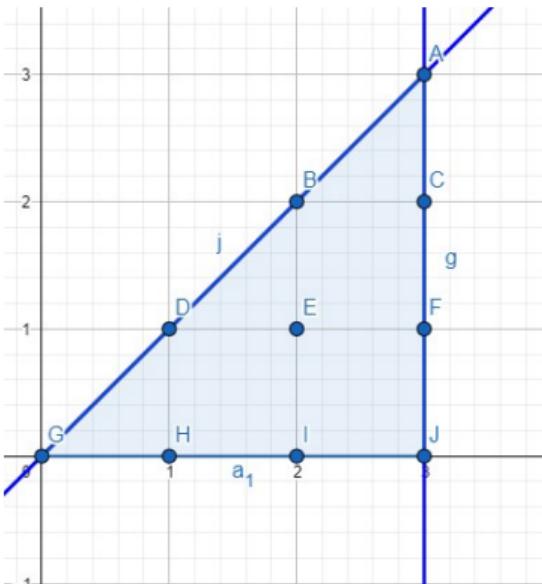
- ▶ Função objetivo
- ▶ Restrições
- ▶ Variáveis de decisão

### Definição

A Otimização Combinatória (COP) é um ramo da ciência da computação e da matemática aplicada que estuda problemas de otimização em conjuntos finitos.

## Definição

COP pode ser formulada como um problema de programação inteira (linear) ou programação inteira binária.



$$\max x + y$$

$$x - y \leq 0$$

$$-x + 3 \geq 0$$

$$x \geq 0, y \geq 0, x, y \in \mathbb{Z}$$

Figura: Wolsey, L. A. (1998)

# Linguagem de Programação Julia



## Linguagem de Programação Julia

- ▶ Lançada em 2012;



## Linguagem de Programação Julia

- ▶ Lançada em 2012;
- ▶ Atender requisitos da computação numérica de alto desempenho;



## Linguagem de Programação Julia

- ▶ Lançada em 2012;
- ▶ Atender requisitos da computação numérica de alto desempenho;
- ▶ Open source;



## Linguagem de Programação Julia

- ▶ Lançada em 2012;
- ▶ Atender requisitos da computação numérica de alto desempenho;
- ▶ Open source;
- ▶ Possibilita chamada de funções em Python e C, por meio de APIs;



## Linguagem de Programação Julia

- ▶ Lançada em 2012;
- ▶ Atender requisitos da computação numérica de alto desempenho;
- ▶ Open source;
- ▶ Possibilita chamada de funções em Python e C, por meio de APIs;
- ▶ Tipagem dinâmica



## Linguagem de Programação Julia

- ▶ Lançada em 2012;
- ▶ Atender requisitos da computação numérica de alto desempenho;
- ▶ Open source;
- ▶ Possibilita chamada de funções em Python e C, por meio de APIs;
- ▶ Tipagem dinâmica

Documentação:

<https://docs.julialang.org/en/v1/>

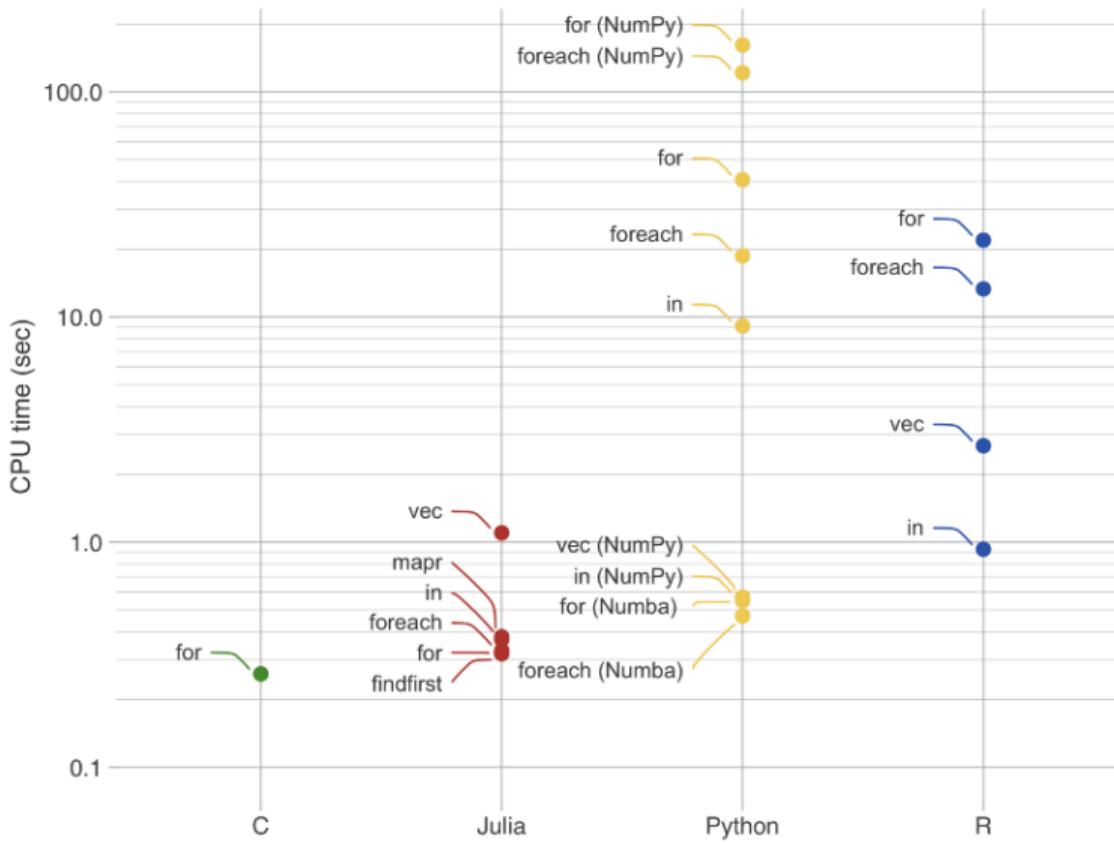


Figura: <https://www.datasciencecentral.com/profiles/blogs/r-vs-python-vs-julia-how-easy-it-is-to-write-efficient-code>

## Julia for Mathematical Programming (JuMP)

- ▶ É um dos muitos pacotes para Julia;



## Julia for Mathematical Programming (JuMP)

- ▶ É um dos muitos pacotes para Julia;
- ▶ Formular diversos problemas de Otimização;

## Julia for Mathematical Programming (JuMP)

- ▶ É um dos muitos pacotes para Julia;
- ▶ Formular diversos problemas de Otimização;
- ▶ Tem um interface didática;

## Julia for Mathematical Programming (JuMP)

- ▶ É um dos muitos pacotes para Julia;
- ▶ Formular diversos problemas de Otimização;
- ▶ Tem um interface didática;
- ▶ Permite o uso de diversos solvers

## Julia for Mathematical Programming (JuMP)

- ▶ É um dos muitos pacotes para Julia;
- ▶ Formular diversos problemas de Otimização;
- ▶ Tem um interface didática;
- ▶ Permite o uso de diversos solvers

Documentação:

<https://jump.dev/JuMP.jl/stable/>

Exemplo 1.1 (Wolsey 1998):

$$\begin{aligned} & \max \quad x_1 + 0.64x_2 \\ \text{s.t.} \quad & 50x_1 + 31x_2 \leq 250 \\ & 3x_1 - 2x_2 \geq -4 \\ & x_1, x_2 \geq 0, \text{ and integer} \end{aligned}$$

- ▶ Solução usando programação linear:  $(x_1, x_2) = (\frac{376}{193}, \frac{950}{193}) \approx (1.9, 4.9)$
- ▶ Solução ótima inteira:  $(x_1, x_2) = (5, 0)$

# O Problema da Atribuição

## Objetivo

Atribuir tarefas a pessoas ao menor custo total possível.

# O Problema da Atribuição

## Objetivo

Atribuir tarefas a pessoas ao menor custo total possível.

Variáveis:

- ▶  $x_{ij} \in \{0, 1\}$ : atribuição da tarefa  $j$  para a pessoa  $i$

# O Problema da Atribuição

## Objetivo

Atribuir tarefas a pessoas ao menor custo total possível.

Variáveis:

- ▶  $x_{ij} \in \{0, 1\}$ : atribuição da tarefa  $j$  para a pessoa  $i$

Restrições:

- ▶ Cada pessoa deve efetuar uma única tarefa:

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, \dots, n$$

# O Problema da Atribuição

## Objetivo

Atribuir tarefas a pessoas ao menor custo total possível.

Variáveis:

- ▶  $x_{ij} \in \{0, 1\}$ : atribuição da tarefa  $j$  para a pessoa  $i$

Restrições:

- ▶ Cada pessoa deve efetuar uma única tarefa:

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, \dots, n$$

- ▶ Cada tarefa deve ser executada por uma única pessoa:

$$\sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, \dots, n$$

# O Problema da Atribuição

## Objetivo

Atribuir tarefas a pessoas ao menor custo total possível.

Variáveis:

- ▶  $x_{ij} \in \{0, 1\}$ : atribuição da tarefa  $j$  para a pessoa  $i$

Restrições:

- ▶ Cada pessoa deve efetuar uma única tarefa:  
 $\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, \dots, n$
- ▶ Cada tarefa deve ser executada por uma única pessoa:  
 $\sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, \dots, n$
- ▶ As variáveis  $x_{ij}$  são binárias 0-1:  
 $x_{ij} \in \{0, 1\}, \forall i = 1, \dots, n, \forall j = 1, \dots, n$

# O Problema da Atribuição

## Objetivo

Atribuir tarefas a pessoas ao menor custo total possível.

Variáveis:

- ▶  $x_{ij} \in \{0, 1\}$ : atribuição da tarefa  $j$  para a pessoa  $i$

Restrições:

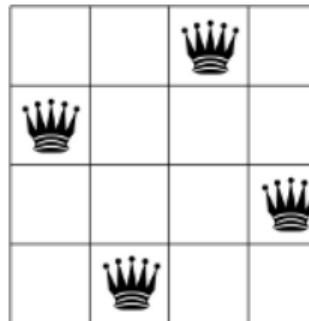
- ▶ Cada pessoa deve efetuar uma única tarefa:  
 $\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, \dots, n$
- ▶ Cada tarefa deve ser executada por uma única pessoa:  
 $\sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, \dots, n$
- ▶ As variáveis  $x_{ij}$  são binárias 0-1:  
 $x_{ij} \in \{0, 1\}, \forall i = 1, \dots, n, \forall j = 1, \dots, n$

Função objetivo:

- ▶ O somatório dos custos deve ser minimizado:  $\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$

## O Problema das N-Rainhas

- ▶ Variável  $X_{ij} \in \{0, 1\}$  : 1 se uma rainha estiver na posição  $(i, j)$ , 0 caso contrário.



$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

► **Função Objetivo:** Maximizar o número de rainhas em um tabuleiro  $N \times N$ .

► **Função Objetivo:** Maximizar o número de rainhas em um tabuleiro  $N \times N$ .

$$\max \sum_{i=1}^N \sum_{j=1}^N X_{ij}$$

## Restrições

- ▶ Só podemos ter exatamente uma rainha na horizontal

## Restrições

- ▶ Só podemos ter exatamente uma rainha na horizontal

$$\sum_{j=1}^N X_{ij} \leq 1 \quad i = 1, \dots, N$$

## Restrições

- ▶ Só podemos ter exatamente uma rainha na horizontal

$$\sum_{j=1}^N X_{ij} \leq 1 \quad i = 1, \dots, N$$

- ▶ Só podemos ter exatamente uma rainha na vertical

## Restrições

- ▶ Só podemos ter exatamente uma rainha na horizontal

$$\sum_{j=1}^N X_{ij} \leq 1 \quad i = 1, \dots, N$$

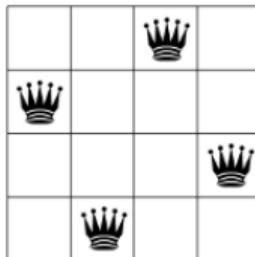
- ▶ Só podemos ter exatamente uma rainha na vertical

$$\sum_{i=1}^N X_{ij} \leq 1 \quad j = 1, \dots, N$$

## Restrições

- Só podemos ter exatamente uma rainha na diagonal positiva

$$\underbrace{\sum_{i=1}^N \sum_{j=1}^N}_{i-j=k} X_{ij} \leq 1 \quad k = -(N-2), \dots, N-2$$

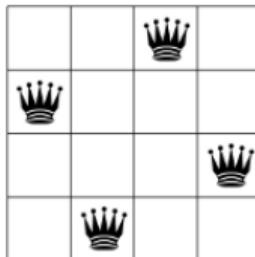


$$\begin{bmatrix} 0 & 0 & \underbrace{1 & 0}_{k = i-j = -2} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \underbrace{0 & 1 & 0}_{k = i-j = -1} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

## Restrições

- Só podemos ter exatamente uma rainha na diagonal negativa

$$\underbrace{\sum_{i=1}^N \sum_{j=1}^N}_{i+j=k} X_{ij} \leq 1 \quad k = 3, \dots, 2N - 1$$



$$\begin{bmatrix} & & \overbrace{0 & 0 & 1 & 0}^{k = i+j = 3} \\ \overbrace{1 & 0 & 0 & 0}^{k = i+j = 4} & & & \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} & & \overbrace{0 & 0 & 1 & 0}^{k = i+j = 3} \\ \overbrace{1 & 0 & 0 & 0}^{k = i+j = 4} & & & \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} & & \overbrace{0 & 0 & 1 & 0}^{k = i+j = 3} \\ \overbrace{1 & 0 & 0 & 0}^{k = i+j = 4} & & & \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



**UFRRJ**  
MATEMÁTICA  
45 anos



# Obrigada!

Contato:  
[afazevedo29@gmail.com](mailto:afazevedo29@gmail.com)

XIV  
SEMAT  
UFRRJ

Matemática  
Invisível

Descobrindo o lado oculto da Matemática