Introdução Linguagem de Programação Julia e ao pacote JuMP

Amanda Ferreira de Azevedo

Universidade Federal do Rio de Janeiro

13 de setembro de 2021

Repositório Github

https://github.com/afazevedo/learning-julia

Sumário

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/
http://leandro.iqm.unicamp.br/m3g/main/didatico/simulacoes/tutorial-Julia.pdf

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/
http://leandro.iqm.unicamp.br/m3g/main/didatico/simulacoes/tutorial-Julia.pdf

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/
http://leandro.iqm.unicamp.br/m3g/main/didatico/simulacoes/tutorial-Julia.pdf

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/
http://leandro.iqm.unicamp.br/m3g/main/didatico/simulacoes/tutorial-Julia.pdf

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/
http://leandro.iqm.unicamp.br/m3g/main/didatico/simulacoes/tutorial-Julia.pdf

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/
http://leandro.iqm.unicamp.br/m3g/main/didatico/simulacoes/tutorial-Julia.pdf

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/
http://leandro.iqm.unicamp.br/m3g/main/didatico/simulacoes/tutorial-Julia.pdf

Linguagem de Programação Julia

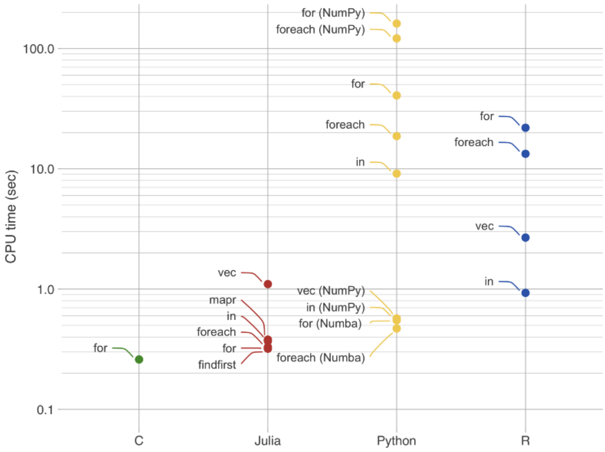

Figure 1: https://www.datasciencecentral.com/profiles/blogs/r-vs-python-vs-julia-how-easy-it-is-to-write-efficient-code

Julia for Mathematical Programming (JuMP)

- É um dos pacotes para Julia;
- Formular diversos problemas de Otimização;
- Tem um código "amigável";
- Permite o uso de diversos solvers

Documentação:
https://jump.dev/JuMP.jl/stable/

Julia for Mathematical Programming (JuMP)

- É um dos pacotes para Julia;
- Formular diversos problemas de Otimização;
- Tem um código "amigável";
- Permite o uso de diversos solvers

Documentação:
https://jump.dev/JuMP.jl/stable/

Julia for Mathematical Programming (JuMP)

- É um dos pacotes para Julia;
- Formular diversos problemas de Otimização;
- Tem um código “amigável”;
- Permite o uso de diversos solvers

Documentação:
https://jump.dev/JuMP.jl/stable/

Julia for Mathematical Programming (JuMP)

- É um dos pacotes para Julia;
- Formular diversos problemas de Otimização;
- Tem um código "amigável";
- Permite o uso de diversos solvers

Documentação:
https://jump.dev/JuMP.jl/stable/

Julia for Mathematical Programming (JuMP)

- É um dos pacotes para Julia;
- Formular diversos problemas de Otimização;
- Tem um código "amigável";
- Permite o uso de diversos solvers

Documentação:
`https://jump.dev/JuMP.jl/stable/`

Instalação do Julia

Download:
`https://julialang.org/downloads/`

Windows:

- Escolha por *64-bit ou 32-bit* e faça o download
- Aperte no executável baixado e siga os procedimentos;

Linux:

- Escolha por *generic Linux binaries on x86*
- Mova o arquivo para o diretório home;
- Abra o terminal nessa pasta e utilize o comando:
  tar -xzf julia-1.6.2-linux-x86_64.tar.gz para descompactar

Guardem o caminho de instalação utilizado!

Instalação do Editor de Texto: Visual Studio Code

Download:
`https://code.visualstudio.com/Download`

Windows:

- Escolha por *64-bit ou 32-bit* e faça o download;
- Aperte no executável baixado e siga os procedimentos

Linux:

- Escolha o *.deb* e faça o download
- Abra o terminal no diretório do arquivo, faça *sudo dpkg -i nome_do_arquivo.deb* e siga os procedimentos

Pacote Julia:

- Aperte *ctrl+shift+x* para acessar extensões
- Procure por "Julia" e instale o pacote
- Reinicie

Verificar nas configurações do pacote se o caminho está correto!

Instalação do Solver: Gurobi

Criar uma conta: `https://pages.gurobi.com/registration`

- Selecionar *Academic*;
- Preencha com seus dados pessoais: Em *Company Email Address* coloque o e-mail de sua universidade e em *University* o nome
- Em *Academic Position* selecione a opção *Student*
- Clique em *Access Now* e confirme em seu e-mail

Download:

- Clique em `https://www.gurobi.com/login/` e logue com sua conta
- Clique em `https://www.gurobi.com/downloads/gurobi-optimizer-eula/` e aperte em *I Accept the End User License Agreement*

Windows:

- Escolha Gurobi-9.1.2-win64.msi e faça o download

Linux:

- Escolha o *gurobi9.1.2_linux64.tar.gz*

Siga os procedimentos abaixo para instalação:

`https://www.gurobi.com/wp-content/uploads/2021/04/README_9.1.2.txt`

Instalação do Solver: Gurobi

Licença Acadêmica:

- Clique em `https://www.gurobi.com/downloads/end-user-license-agreement-academic/` e selecione a opção *I Acceot These Conditions*
- Vá mais abaixo na página até a parte de Installation . Copie o código que começa com **grbgetkey ...**

Windows:

- Abra o Command Prompter ao digitar em pesquisa por cmd e cole esse código

Linux:

- Cole o código diretamente no terminal

./grbgetkey ...

Configuração de Pacotes Esseciais

Criar um novo documento:

- Abrir o VScode;
- Clicar "ctrl + n" para abrir um novo arquivo
- Apertar "ctrl + s" para salvar
- Salve com ".jl" ao final, para identificar que é Julia

Instalação de pacotes:

- Digitar no documento *using Pkg* e apertar *alt+enter* para rodar
- Adicionar o JuMP: digite no *Julia REPL* abaixo *Pkg.add("JuMP")*
- Adicionar o Gurobi: digite no *Julia REPL* abaixo *Pkg.add("Gurobi")*, depois *Pkg.build("Gurobi")*

Após instalação, basta chamar as bibliotecas usando using

Julia for Mathematical Programming (JuMP)

Example 1.1 (Wolsey 1998):

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

- 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.

Variáveis:

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

Restrições:

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

Função objetivo:

- Cada pessoa deve efetuar uma única tarefa: $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

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

Função objetivo:

- Cada pessoa deve efetuar uma única tarefa: $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

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

Função objetivo:

- Cada pessoa deve efetuar uma única tarefa: $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

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:

- Cada pessoa deve efetuar uma única tarefa: $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

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

Função objetivo:

- Cada pessoa deve efetuar uma única tarefa: $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

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

Função objetivo:

- Cada pessoa deve efetuar uma única tarefa: $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

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

Função objetivo:

- Cada pessoa deve efetuar uma única tarefa: $\min \displaystyle\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

O Problema da Mochila 0-1

Objetivo

Selecionar um subconjunto de itens de máximo retorno sem exceder o orçamento disponível.

Variáveis:

- $x_j \in \{0,1\}$: 1 se considerar o item j, 0 caso contrário

Restrições:

- O peso dos itens não pode ultrapassar b
  $$\sum_{j=1}^{n} a_j x_j \leq b$$
- Todas as variáveis x_j são binárias 0-1: $x_j \in \{0,1\}, \forall j = 1, \ldots, n$

Função objetivo:

- Retorno total deve ser maximizado $\max \sum_{j=1}^{n} c_j x_j$

O Problema da Mochila 0-1

Objetivo

Selecionar um subconjunto de itens de máximo retorno sem exceder o orçamento disponível.

Variáveis:

- $x_j \in \{0,1\}$: 1 se considerar o item j, 0 caso contrário

Restrições:

- O peso dos itens não pode ultrapassar b
  $$\sum_{j=1}^{n} a_j x_j \leq b$$
- Todas as variáveis x_j são binárias 0-1: $x_j \in \{0,1\}, \forall j = 1, \ldots, n$

Função objetivo:

- Retorno total deve ser maximizado $\max \sum_{j=1}^{n} c_j x_j$

O Problema da Mochila 0-1

Objetivo

Selecionar um subconjunto de itens de máximo retorno sem exceder o orçamento disponível.

Variáveis:

- $x_j \in \{0, 1\}$: 1 se considerar o item j, 0 caso contrário

Restrições:

- O peso dos itens não pode ultrapassar b
$$\sum_{j=1}^{n} a_j x_j \leq b$$
- Todas as variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j = 1, \ldots, n$

Função objetivo:

- Retorno total deve ser maximizado $\max \sum_{j=1}^{n} c_j x_j$

O Problema da Mochila 0-1

Objetivo

Selecionar um subconjunto de itens de máximo retorno sem exceder o orçamento disponível.

Variáveis:

- $x_j \in \{0, 1\}$: 1 se considerar o item j, 0 caso contrário

Restrições:

- O peso dos itens não pode ultrapassar b
  $$\sum_{j=1}^{n} a_j x_j \leq b$$
- Todas as variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j = 1, \ldots, n$

Função objetivo:

- Retorno total deve ser maximizado $\max \sum_{j=1}^{n} c_j x_j$

O Problema da Mochila 0-1

Objetivo

Selecionar um subconjunto de itens de máximo retorno sem exceder o orçamento disponível.

Variáveis:

- $x_j \in \{0, 1\}$: 1 se considerar o item j, 0 caso contrário

Restrições:

- O peso dos itens não pode ultrapassar b
  $$\sum_{j=1}^{n} a_j x_j \leq b$$
- Todas as variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j = 1, \ldots, n$

Função objetivo:

- Retorno total deve ser maximizado $\max \sum_{j=1}^{n} c_j x_j$

O Problema de Recobrimento de Conjuntos

Objetivo

Definir, ao mínimo custo, quais os centros de serviço abrir.

Matriz de incidência

- $a_{ij} \in \{0, 1\}$: 1 se $i \in M$ pode ser atendida por $j \in N$

Variáveis:

- $x_j \in \{0, 1\}$: se um centro deve ou não ser instalado em $j \in N$

Restrições:

- $i \in M$ deve ser coberta por pelo menos um $j \in N$
  $$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M$$
- Variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j \in 1, \ldots, n$

Função objetivo:

- Custo de instalação de centros deve ser minimizado: $\min \sum_{j=1}^{n} c_j x_j$

O Problema de Recobrimento de Conjuntos

Objetivo

Definir, ao mínimo custo, quais os centros de serviço abrir.

Matriz de incidência

- $a_{ij} \in \{0,1\}$: 1 se $i \in M$ pode ser atendida por $j \in N$

Variáveis:

- $x_j \in \{0,1\}$: se um centro deve ou não ser instalado em $j \in N$

Restrições:

- $i \in M$ deve ser coberta por pelo menos um $j \in N$
  $$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M$$
- Variáveis x_j são binárias 0-1: $x_j \in \{0,1\}, \forall j \in 1, \ldots, n$

Função objetivo:

- Custo de instalação de centros deve ser minimizado: $\min \sum_{j=1}^{n} c_j x_j$

O Problema de Recobrimento de Conjuntos

Objetivo

Definir, ao mínimo custo, quais os centros de serviço abrir.

Matriz de incidência

- $a_{ij} \in \{0, 1\}$: 1 se $i \in M$ pode ser atendida por $j \in N$

Variáveis:

- $x_j \in \{0, 1\}$: se um centro deve ou não ser instalado em $j \in N$

Restrições:

- $i \in M$ deve ser coberta por pelo menos um $j \in N$
$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M$$
- Variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j \in 1, \ldots, n$

Função objetivo:

- Custo de instalação de centros deve ser minimizado: $\min \sum_{j=1}^{n} c_j x_j$

O Problema de Recobrimento de Conjuntos

Objetivo

Definir, ao mínimo custo, quais os centros de serviço abrir.

Matriz de incidência

- $a_{ij} \in \{0, 1\}$: 1 se $i \in M$ pode ser atendida por $j \in N$

Variáveis:

- $x_j \in \{0, 1\}$: se um centro deve ou não ser instalado em $j \in N$

Restrições:

- $i \in M$ deve ser coberta por pelo menos um $j \in N$

  $$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M$$

- Variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j \in 1, \ldots, n$

Função objetivo:

- Custo de instalação de centros deve ser minimizado: $\min \sum_{j=1}^{n} c_j x_j$

O Problema de Recobrimento de Conjuntos

Objetivo

Definir, ao mínimo custo, quais os centros de serviço abrir.

Matriz de incidência

- $a_{ij} \in \{0, 1\}$: 1 se $i \in M$ pode ser atendida por $j \in N$

Variáveis:

- $x_j \in \{0, 1\}$: se um centro deve ou não ser instalado em $j \in N$

Restrições:

- $i \in M$ deve ser coberta por pelo menos um $j \in N$
  $$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M$$
- Variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j \in 1, \ldots, n$

Função objetivo:

- Custo de instalação de centros deve ser minimizado: $\min \sum_{j=1}^{n} c_j x_j$

O Problema de Recobrimento de Conjuntos

Objetivo

Definir, ao mínimo custo, quais os centros de serviço abrir.

Matriz de incidência

- $a_{ij} \in \{0, 1\}$: 1 se $i \in M$ pode ser atendida por $j \in N$

Variáveis:

- $x_j \in \{0, 1\}$: se um centro deve ou não ser instalado em $j \in N$

Restrições:

- $i \in M$ deve ser coberta por pelo menos um $j \in N$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M$$

- Variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j \in 1, \ldots, n$

Função objetivo:

- Custo de instalação de centros deve ser minimizado: $\min \sum_{j=1}^{n} c_j x_j$

O Problema de Recobrimento de Conjuntos

Objetivo

Definir, ao mínimo custo, quais os centros de serviço abrir.

Matriz de incidência

- $a_{ij} \in \{0, 1\}$: 1 se $i \in M$ pode ser atendida por $j \in N$

Variáveis:

- $x_j \in \{0, 1\}$: se um centro deve ou não ser instalado em $j \in N$

Restrições:

- $i \in M$ deve ser coberta por pelo menos um $j \in N$
  $$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M$$
- Variáveis x_j são binárias 0-1: $x_j \in \{0, 1\}, \forall j \in 1, \ldots, n$

Função objetivo:

- Custo de instalação de centros deve ser minimizado: $\min \sum_{j=1}^{n} c_j x_j$

O Problema Do Lote Não-Capacitado

Objetivo

Minimizar os custos de instalações de depósitos e de transporte para suprir todos os clientes

Variáveis:

- $y_j \in \{0, 1\}, \forall j \in N$: abrir ou não um depósito em j.
- $x_{ij} \geq 0, \forall i \in M, j \in N$: fração da demanda do cliente i atendida pelo depósito j.

Restrições:

- 100% da demanda de $i \in M$ deve ser atendida:
  $$\sum_{j \in N} x_{ij} = 1, \forall i \in M$$
- m é o número máximo de clientes que $j \in N$ pode atender:
  $$\sum_{i \in M} x_{ij} \leq m y_j, \forall j \in N$$

Função objetivo:

- Minimizar custos de instalação e de transporte:
  $$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

O Problema Do Lote Não-Capacitado

Objetivo

Minimizar os custos de instalações de depósitos e de transporte para suprir todos os clientes

Variáveis:

- $y_j \in \{0,1\}, \forall j \in N$: abrir ou não um depósito em j.
- $x_{ij} \geq 0, \forall i \in M, j \in N$: fração da demanda do cliente i atendida pelo depósito j.

Restrições:

- 100% da demanda de $i \in M$ deve ser atendida: $\sum_{j \in N} x_{ij} = 1, \forall i \in M$
- m é o número máximo de clientes que $j \in N$ pode atender: $\sum_{i \in M} x_{ij} \leq m y_j, \forall j \in N$

Função objetivo:

- Minimizar custos de instalação e de transporte: $\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$

O Problema Do Lote Não-Capacitado

Objetivo

Minimizar os custos de instalações de depósitos e de transporte para suprir todos os clientes

Variáveis:

- $y_j \in \{0,1\}, \forall j \in N$: abrir ou não um depósito em j.
- $x_{ij} \geq 0, \forall i \in M, j \in N$: fração da demanda do cliente i atendida pelo depósito j.

Restrições:

- 100% da demanda de $i \in M$ deve ser atendida: $\sum_{j \in N} x_{ij} = 1, \forall i \in M$
- m é o número máximo de clientes que $j \in N$ pode atender: $\sum_{i \in M} x_{ij} \leq m y_j, \forall j \in N$

Função objetivo:

- Minimizar custos de instalação e de transporte:

$$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

O Problema Do Lote Não-Capacitado

Objetivo

Minimizar os custos de instalações de depósitos e de transporte para suprir todos os clientes

Variáveis:

- $y_j \in \{0, 1\}, \forall j \in N$: abrir ou não um depósito em j.
- $x_{ij} \geq 0, \forall i \in M, j \in N$: fração da demanda do cliente i atendida pelo depósito j.

Restrições:

- 100% da demanda de $i \in M$ deve ser atendida:
  $$\sum_{j \in N} x_{ij} = 1, \forall i \in M$$
- m é o número máximo de clientes que $j \in N$ pode atender:
  $$\sum_{i \in M} x_{ij} \leq m y_j, \forall j \in N$$

Função objetivo:

- Minimizar custos de instalação e de transporte:
  $$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

O Problema Do Lote Não-Capacitado

Objetivo

Minimizar os custos de instalações de depósitos e de transporte para suprir todos os clientes

Variáveis:

- $y_j \in \{0, 1\}, \forall j \in N$: abrir ou não um depósito em j.
- $x_{ij} \geq 0, \forall i \in M, j \in N$: fração da demanda do cliente i atendida pelo depósito j.

Restrições:

- 100% da demanda de $i \in M$ deve ser atendida:
  $$\sum_{j \in N} x_{ij} = 1, \forall i \in M$$
- m é o número máximo de clientes que $j \in N$ pode atender:
  $$\sum_{i \in M} x_{ij} \leq m y_j, \forall j \in N$$

Função objetivo:

- Minimizar custos de instalação e de transporte:
  $$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$