Pesquisa Operacional II

Otimização Inteira - Árvore Geradora Mínima

Prof. M.Sc. Diego Ascânio Santos (ascanio@cefetmg.br)

Aula baseada sobre o material do professor Dr. João Fernando Machry Sarubbi (joao@cefetmg.br - DECOM), vídeoaulas do curso de algoritmos de pesquisa operacional da Universidade Nacional de Taiwan e referências da documentação do Google OR-Tools

Belo Horizonte, 2023.

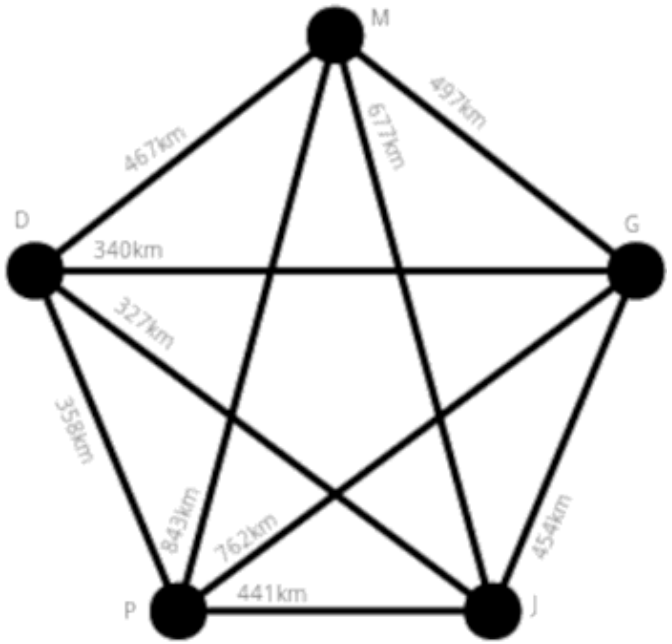
Roteiro

1. Introdução
2. Modelo em Programação Inteira
3. Resolução via Solver (tempo exponencial)
4. Resolução algoritmica (tempo logaritimico)

Introdução

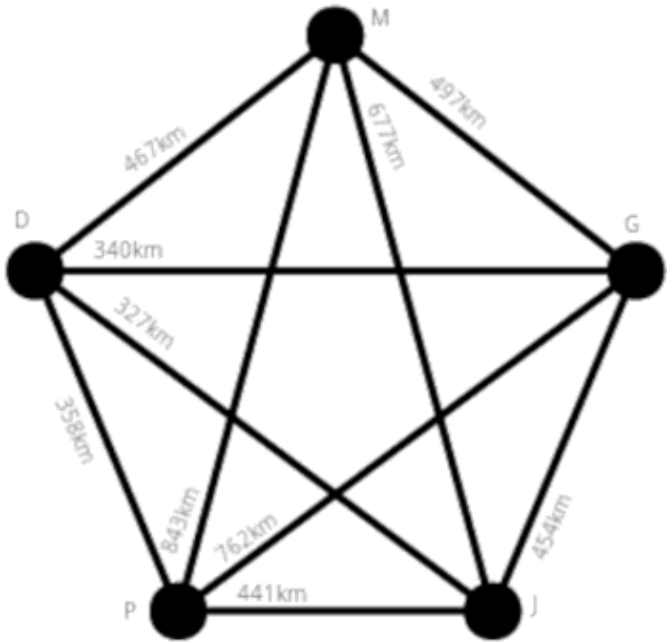
Introdução

A empresa Ascânio Telecom deseja instalar links de fibra óptica para conectar as cidades de Divinópolis, Montes Claros, Governador Valadares, Juiz de Fora e Poços de Caldas.



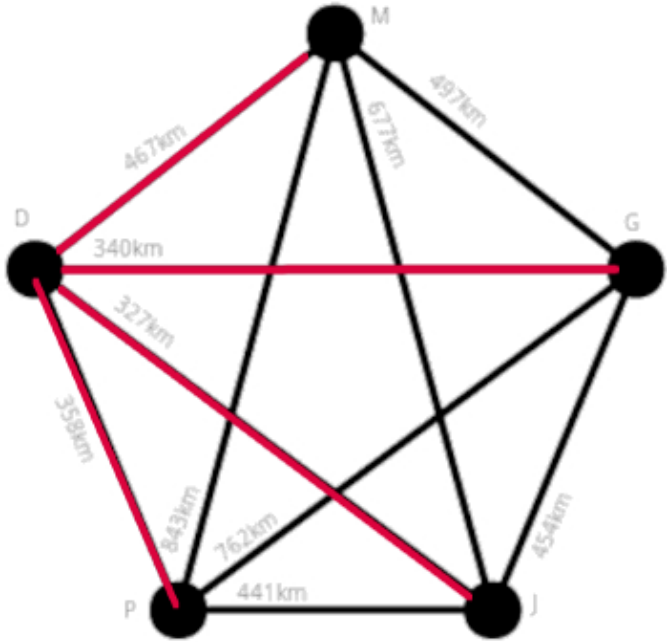
Introdução

Seu objetivo como especialista em pesquisa operacional é o de utilizar a menor quantidade possível de cabos (para minimizar os custos) para formar uma rede de cabos entre as cinco cidades de forma que todas elas estejam interligadas sem cabos extras ou caminhos desnecessários.



Introdução

A árvore geradora mínima - por causa de seu formato de raízes / entroncamentos - é a estrutura que representa este conjunto de ligações entre cidades que conecta todas elas com o menor custo possível.



Então, podemos dizer que: Árvore geradora mínima é toda estrutura que liga cidades (nós) em uma região (grafo) através dos caminhos (arestas) de menor custo possível.

Sua aplicação na área de infraestrutura é bastante difundida, servindo para projetar redes de luz, água, telefone, esgoto, dentre outras.

Modelo em Programação Linear Inteira

Modelo

O problema da árvore geradora mínima pode ser modelado por programação inteira.

De forma similar ao PCV*, é necessário escolher caminhos que conectam cidades para formar uma ligação entre elas. Entretanto, enquanto no PCV a escolha dos caminhos deve formar um ciclo que começa e termina na mesma cidade, na árvora geradora mínima isto não pode ocorrer.

Assim, as variáveis de decisão do problema são variáveis binárias da forma $X_{ij} \in \{0, 1\}$ que representam a escolha (1) ou não escolha (0) de um caminho que liga a cidade i à cidade j para fazer parte da árvore geradora mínima.

Modelo

- A variável de decisão do problema é $X_{ij} \in \{0, 1\}$
- Todo caminho que liga uma cidade i à uma cidade j tem uma distancia D_{ij}.
- Para conectar n cidades em uma árvore geradora mínima não podem existir nem mais, nem menos do que $n - 1$ caminhos.
- Na árvore geradora mínima não são admitidos nenhum tipo de ciclo, ou seja, nenhuma cidade possui mais do que um caminho de entrada e um de saída.

Modelo

Logo, nosso objetivo é minimizar a distância total de cabos a conectarem as cidades, representado pela função:

$$Z = \min(\sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij} \cdot X_{ij})$$

Temos a restrição de que para conectar n cidades em uma árvore geradora mínima não podem existir nem mais, nem menos do que $n-1$ caminhos:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij} = n - 1$$

Modelo

Temos a nossa segunda restrição, que especifica que para todo e qualquer subconjunto S possível de cidades - exceto o conjunto vazio e o próprio conjunto V de cidades - este subconjunto deve possuir no máximo $|S|^*$ - 1 caminhos possíveis:

$$\sum_{i,j \in S} X_{ij} \leq |S| - 1 \; \forall S \subseteq V, \; S \neq \emptyset \text{ e } S \neq V$$

Logo esta restrição desdobra-se em 2^n restrições possíveis para todo $S \subseteq V$.

Modelo

Considerando nosso conjunto de cidades $\{D, M, G, J, P\}$

Seus subconjuntos possíveis: $\{\}, \{D\}, \{M\}, \cdots, \{D, M, G\}, \cdots \{D, M, G, J, P\}$

Para cada subconjunto $S \in P$ (conjunto de partes, ou powerset), devemos adicionar uma restrição no nosso modelo.

Exemplo: $S = \{D, M, G\}$. $|S| = 3$

Conjunto de caminhos de S: $\{X_{00}, X_{01}, X_{02}, X_{10}, X_{11}, X_{12}, X_{20}, X_{21}, X_{22}\}$

Logo, adiciona-se a restrição: $\sum_{i,j \in S} X_{ij} \leq 3 - 1$

Faça isso para cada um dos outros subconjuntos possíveis de V

Modelo Final

$$\text{Min Z} = \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} \cdot X_{ij}$$

subject to

$$\sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} = n - 1$$

$$\sum_{i,j \in S} X_{ij} \leq |S| - 1 \; \forall S \subseteq V, \; S \neq \emptyset \text{ e } S \neq V$$

$$x_{ij} \in \{0, 1\} \; \forall i = 1, \cdots, n; \; \forall j = 1, \cdots, n$$

Resolução

Thanks to OVH (https://www.ovh.com/) and GESIS Notebooks (https://notebooks.gesis.org) for supporting us! 🎉
mybinder.org will be updating the base image from Ubuntu 18.04 to 22.04 soon. See this discussion (https://discourse.jupyter.org/t/mybinder-org-upgrade-to-ubuntu-22-04-and-docker-to-podman-base-image-to-ubuntu-22-04/19865) for details.

Starting repository: DiegoAscanio/aula-10/master

New to Binder? Check out the Binder Documentation (https://mybinder.readthedocs.io/en/latest/) for more information

Resolução via Solver

O aspecto negativo da Resolução via Solver é seu tempo exponencial, pois, é necessário adicionar 2^n restrições para impedir formações de ciclos.
Por isso, será abordado na sequência um algoritmo capaz de resolver o problema da árvore geradora mínima em tempo logaritimico, o algoritmo de Prim.

Algoritmo de Prim

Algoritmo de Prim

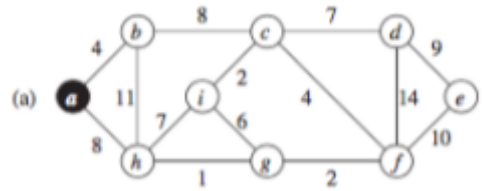
O algoritmo de prim é um algoritmo com custo logaritimico usado para encontrar uma árvore geradora mínima em um mapa (grafo) de cidades e seus caminhos.

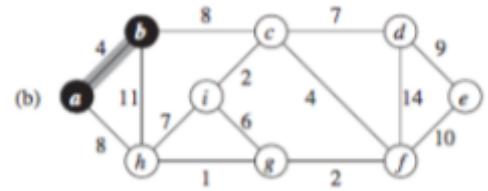
É possível esquematizar o algoritmo em quatro passos, como veremos a seguir:

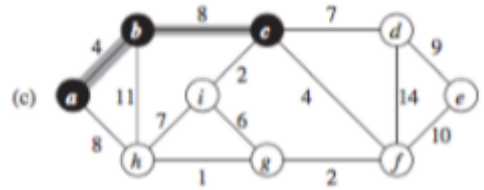
Algoritmo de Prim

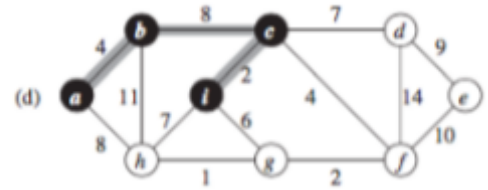
1. Crie um conjunto de cidades visitadas, visite uma cidade qualquer do mapa e adicione esta cidade ao conjunto.
2. Verifique todos os caminhos que saem do conjunto de cidades visitadas e visite a cidade (não visitada) mais próxima.
3. Adicione esta cidade (e seu caminho) ao conjunto de cidades visitadas.
4. Repita os passos 2 e 3 até que todas as cidades do mapa tenham sido visitadas.

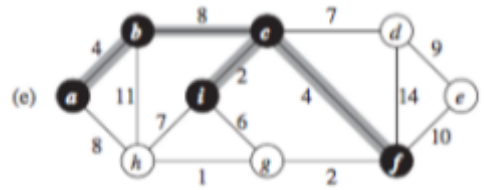
O conjunto de caminhos representa a árvore geradora mínima!

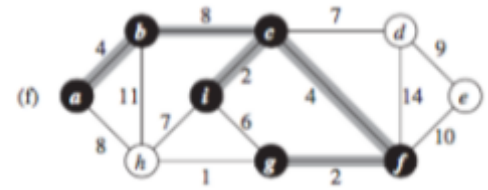

(a)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2

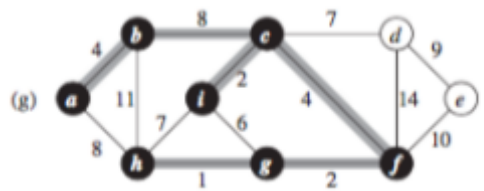

(b)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2

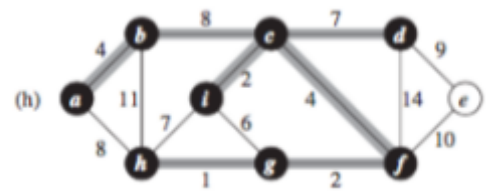

(c)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2

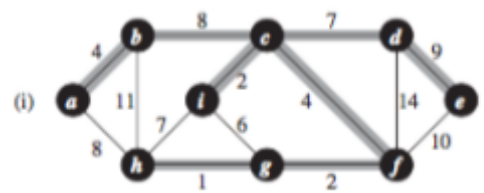

(d)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2


(e)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2


(f)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2


(g)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2


(h)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2


(i)
a b c d e f g h i
4 8 8 7 9 11 7 14 6 10 2 4 1 2

Pseudo Código - Algoritmo de Prim

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Resolução Árvore Geradora Mínima SciPy

Thanks to OVH (https://www.ovh.com/) and GESIS Notebooks (https://notebooks.gesis.org) for supporting us! 🎉
mybinder.org will be updating the base image from Ubuntu 18.04 to 22.04. Some binders may break (https://discourse.jupyter.org/t/mybinder-org-upgrading-base-image-to-ubuntu-22-04/19865) for details.

launch binder