IA881 – Otimização Linear

Aula: Árvore Geradora Mínima (*Minimum spanning tree*)

Ricardo C. L. F. Oliveira

Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

1º Semestre 2019

Tópicos

Conceitos, Definições, Notações I

Definição 1

Sejam N um conjunto de vértices e A um conjunto de arestas ligando os vértices $v \in N$. Define-se grafos como sendo $G(N, A)$. $n = |N|$ representa o número (cardinalidade) de vértices e $m = |A|$ o número (cardinalidade) de arestas.

- Observação: vértice = nó; aresta = ramo (não-orientado) ou arco (orientado)

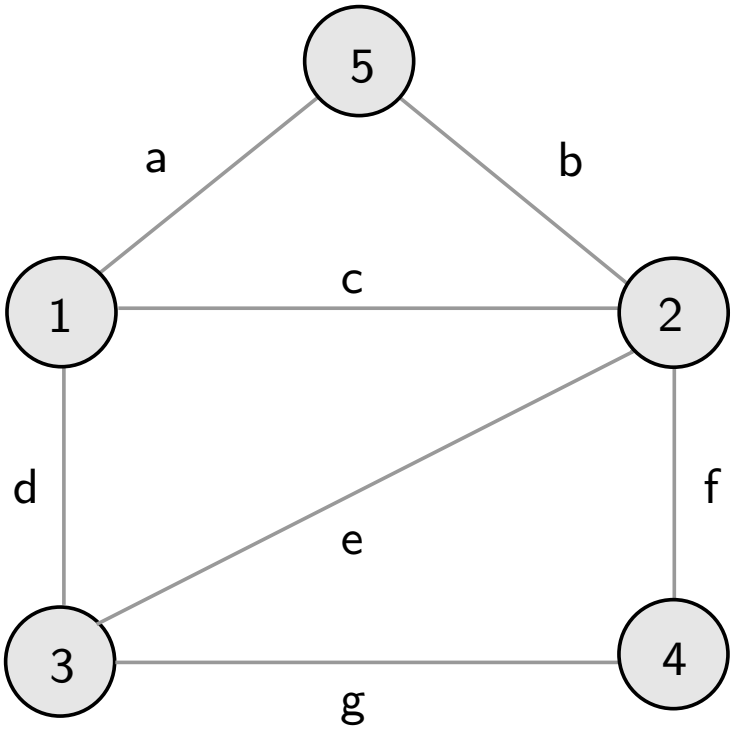

Figura 1: Grafo não orientado.

- Nós=$\{1, 2, 3, 4, 5\}$
- Arestas=$\{a, b, c, d, e, f, g\}$

Conceitos, Definições, Notações II

- Uma alternativa para representar uma aresta é por meio da notação (x, y) com $x, y \in N$. Por exemplo, no grafo da Figura 1, a aresta b poderia ser representada por $(2,5)$ ou $(5,2)$. Caso a aresta seja direcionada (arco), convenciona-se que a primeira componente seja o vértice de origem e a segunda o vértice de destino.

Definição 2

Grafo orientado ou direcionado (não orientado ou não direcionado) – quando as arestas têm (não têm) orientação.

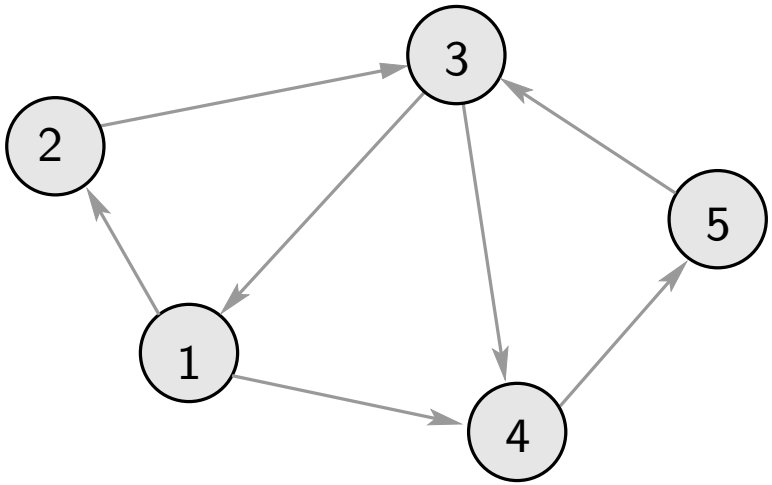

Figura 2: Exemplo de um grafo orientado.

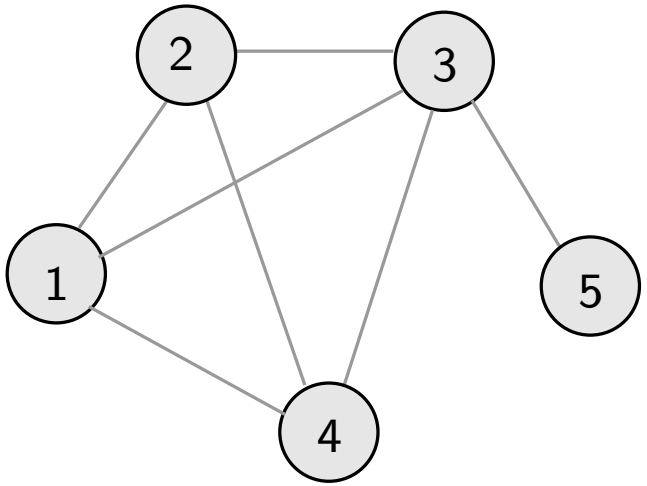

Figura 3: Exemplo de um grafo não orientado.

Conceitos, Definições, Notações III

Definição 3

Um grafo é dito ponderado (ou valorado) se suas arestas possuem custos (ou pesos) associados. Usa-se a notação c_{ij} (ou $c(i,j)$) para denotar o custo da aresta entre os vértices i e j.

Definição 4

$G_s(N_s, A_s)$ é um sub-grafo de $G(N, A)$ se $N_s \subseteq N$ e $A_s \subseteq A$ tal que se $(i,j) \in A_s \Rightarrow i,j \in N_s$. Um grafo $G_s(N_s, A_s)$ é um sub-grafo gerador de $G(N, A)$ se $N_s = N$ e $A_s \subseteq A$.

Conceitos, Definições, Notações IV

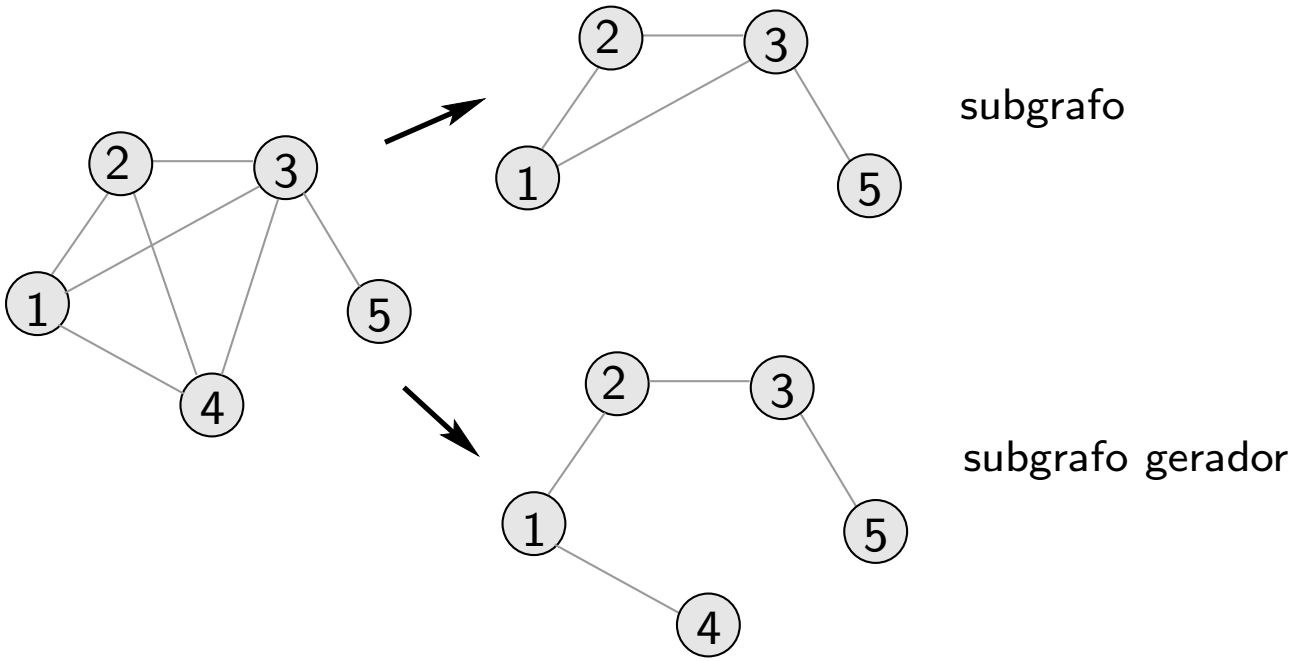

Figura 4: Exemplo de subgrafos (gerador e não gerador).

Conceitos, Definições, Notações V

Definição 5

Grau de um vértice é o número de arestas que incidem nele (no caso orientado, arcos que entram mais os que saem).

Definição 6

Cadeia é uma sequência consecutiva de arestas em que todos os nós visitados são **distintos**. *Exemplo: na Figura 2 –* $\{(2,3)(5,3)(4,5)(1,4)\}$.

Definição 7

Caminho é um caso particular de cadeia na qual os arcos têm os **mesmos sentidos**. *Exemplo Figura 2 –* $\{(2,3)(3,1)(1,4)(4,5)\}$

Definição 8

Comprimento de um caminho é a soma dos pesos (ou custos) das arestas do caminho.

Conceitos, Definições, Notações VI

Definição 9

Um grafo é dito ser conexo se sempre existe uma cadeia entre qualquer par de vértices.

Definição 10

Ciclo ou laço é uma cadeia fechada (termina no nó que iniciou). Exemplo na Figura 2 – $\{(3,1)(1,4)(3,4)\}$

Definição 11

Circuito (ciclo direcionado) é um caminho fechado. Exemplo na Figura 2 – $\{(2,3)(3,1)(1,2)\}$

Definição 12

Uma árvore é um grafo conexo que não contém ciclos.

Conceitos, Definições, Notações VII

- Exemplos de árvore obtidas a partir do grafo da Figura 2: (1) removendo-se as arestas $(2,3)$, $(1,4)$ e $(5,3)$; (2) removendo-se as arestas $(1,4)$, $(3,1)$ e $(3,4)$; Existem outras possibilidades.

Conceitos, Definições, Notações VIII

- A seguir são apresentadas propriedades elementares associadas a árvores que são fundamentais para o desenvolvimento dos algoritmos apresentados nesta aula.

Propriedades de uma árvore

Um grafo G com n vértices é uma árvore se e somente se qualquer uma das seguintes condições é satisfeita:

- G possui $n-1$ arestas e nenhum ciclo.
- G possui $n-1$ arestas e é conexo.
- G é conexo, mas a remoção de uma aresta torna-o desconexo.
- G não tem ciclos (acíclico), mas introduzir uma nova aresta produz um ciclo.
- Quaisquer dois vértices de G estão conectados por um caminho único.

Definição 13

Uma árvore T é uma árvore geradora de um grafo $G(N,A)$ se T é um sub-grafo gerador de G.

Problema da árvore geradora mínima

- Seja $G(N, A)$ um grafo ponderado (arestas possuem custos) e considere T como o conjunto de todas as árvores de um grafo G.

Problema

Dentre todas as árvores $t_k \in T$ do grafo, determinar a árvore geradora mínima $t_k^\star$, isto é, a árvore cujas as arestas somam o menor comprimento (ou custo).

Aplicações

- Considere o problema de cabeamento telefônico entre uma central telefônica e um conjunto de residências numa determinada vizinhança de um bairro ou cidade.

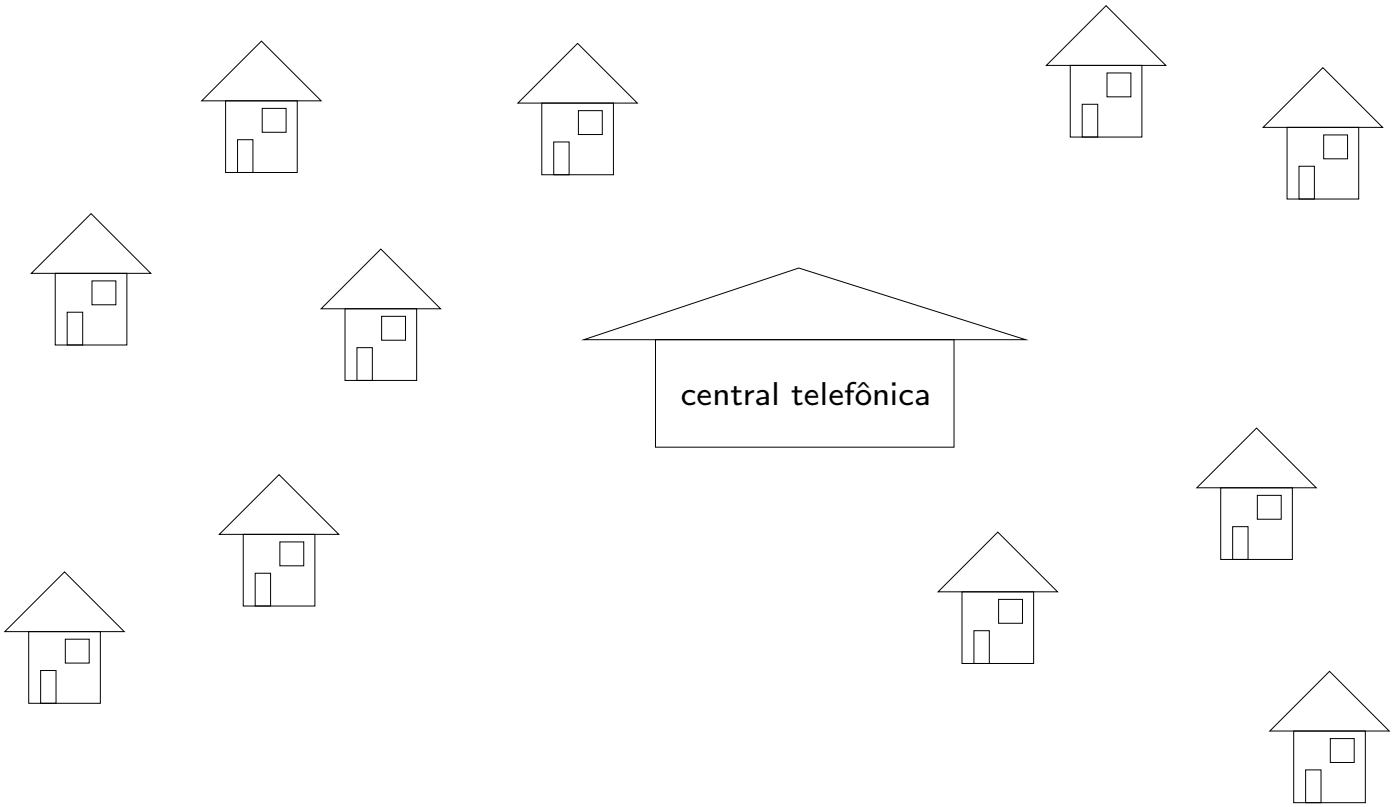

Figura 5: Problema de cabeamento telefônico.

Aplicações

- Solução óbvia (centralizada) entretanto custosa (em termos da quantidade de fio necessária).

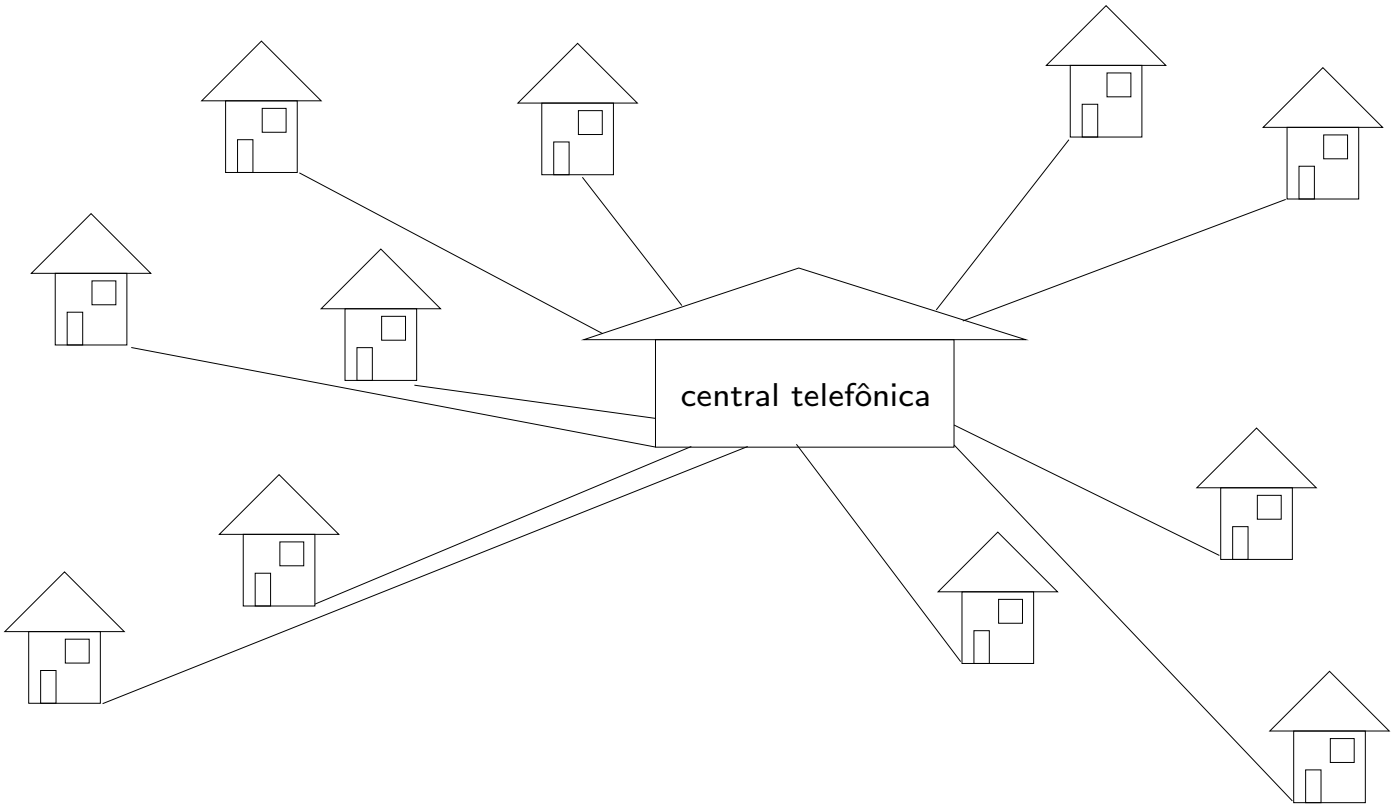

Figura 6: Problema de cabeamento telefônico – solução custosa.

Aplicações

- Solução mais econômica (embora menos segura). A árvore geradora mínima pode ser útil nessa modelagem.

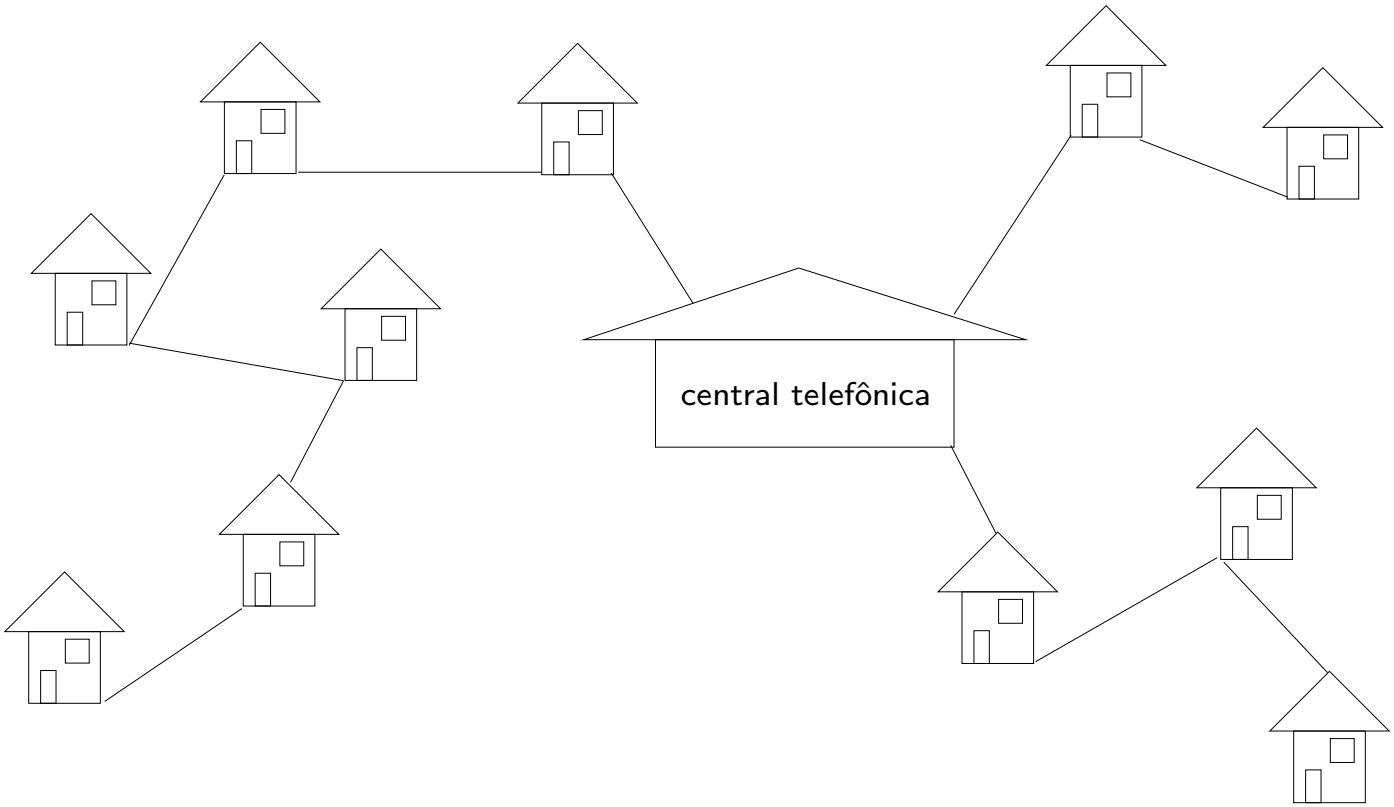

Figura 7: Problema de cabeamento telefônico – solução mais econômica.

Aplicações

Outras aplicações

- Extensão do problema de cabeamento telefônico para tratar outros recursos. Por exemplo, distribuição de água, linhas transmissão de energia elétrica, rede de esgoto, etc.
- Construção de rodovias ou ferrovias para conectar várias cidades ou regiões.
- Projeto de uma rede de computadores local (por exemplo, numa faculdade).
- Reconhecimento de escrita manual.
- Processamento de imagens com aplicação em medicina.
- Aproximação do problema do caixeiro viajante (veremos mais adiante).

Formulação matemática

- Matematicamente, o problema pode ser expresso em termos do seguinte problema de otimização

$$\begin{cases} \min_{t_k \in T} & z(t_k) \\ s.a & z(t_k) = \sum_{(i,j) \in t_k} c_{ij} \end{cases}$$

- Força bruta: testar todas as árvores geradoras. Problemas: implementação difícil e podem existir muitas árvores geradoras (para um grafo completo, da ordem de n^{n-2}). Sugestão: explorar propriedades.

Condições de otimalidade

Definição 14

Um corte é um particionamento do conjunto de vértices N do grafo em dois subconjuntos S e $\bar{S} = N - S$. Cada corte define um conjunto de arestas consistindo nas arestas que possuem um extremo em S e o outro extremo em $\bar{S}$.

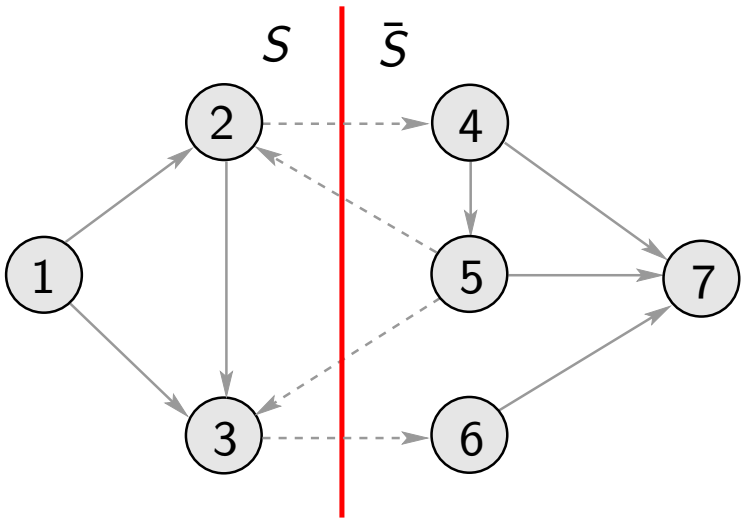

Figura 8: Exemplo de um corte com $S = \{1,2,3\}$, $\bar{S} = \{4,5,6,7\}$. O conjunto de arestas neste corte é dado por $\{(2,4)(5,2)(5,3)(3,6)\}$.

Condições de otimalidade

- Observações elementares sobre árvores:
  - Para toda aresta (k,ℓ) que não pertence à árvore geradora, existe um caminho único entre os nós k e ℓ na árvore geradora. A aresta (k,ℓ) juntamente com esse caminho único definem um ciclo.
  - A remoção de qualquer aresta (i,j) presente na árvore geradora define um corte.

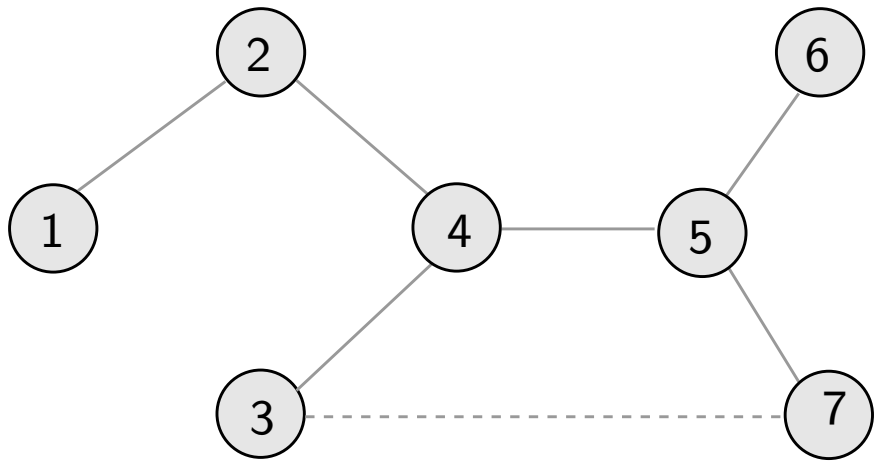

Figura 9: A aresta $(3,7)$ possui um caminho único na árvore geradora e, caso incorporada à árvore, produz um ciclo.

Condição de otimalidade – corte e caminho

Teorema – Otimalidade baseada em corte

Uma árvore geradora $T^\star$ é uma árvore geradora mínima se e somente se ela atende a seguinte condição de otimalidade de corte: Para qualquer aresta $(i,j) \in T^\star$, $c_{ij} \leq c_{k\ell}$ para toda aresta (k,ℓ) contida no corte gerado pela remoção da aresta (i,j) de $T^\star$.

- A prova deste teorema é feita por contradição: Se $c_{ij} > c_{k\ell}$ então substituindo a aresta (i,j) pela aresta (k,ℓ) em $T^\star$ obtém-se uma nova árvore geradora de custo menor, contradizendo a otimalidade de $T^\star$.

Teorema – Otimalidade baseada em caminho

Uma árvore geradora $T^\star$ é uma árvore geradora mínima se e somente se ela atende a seguinte condição de otimalidade de caminho: Para qualquer aresta $(k,\ell) \in G$ e que não está em $T^\star$, $c_{ij} \leq c_{k\ell}$ para toda aresta (i,j) contida no caminho em $T^\star$ que conecta os vértices k e ℓ.

- Prova (por contradição): Se $c_{ij} > c_{k\ell}$ então substituindo a aresta (i,j) pela aresta (k,ℓ) em $T^\star$ obtém-se uma nova árvore geradora de custo menor, contradizendo a otimalidade de $T^\star$.

Algoritmos "simplistas"

- Usando a condição de otimalidade baseada em caminho, pode-se conceber algoritmos "simplistas" (ou genéricos) para resolver o problema da árvore geradora mínima.

Algorithm 1 Algoritmo simplista 1

1: Comece com uma árvore geradora arbitrária.
2: **enquanto** a condição de otimalidade de caminho falhar para alguma aresta fora da árvore **faça**
3: substitua a aresta na árvore.
4: **fim enquanto**

- Complexidade: $\mathcal{O}(nm^3)$

- E se usarmos a condição de otimalidade baseada em corte?

- É possível melhorar? Sim, por exemplo Prim, Kruskal e Borůvka (algoritmos "gulosos").

Algoritmo de Prim I

Robert Clay Prim.

- Pessoas envolvidas: Vojtěch Jarník (1930), R. C. Prim (1957) e E. W. Dijkstra (1959).
- Estratégia: Começa de um vértice arbitrário, e acrescenta um vértice por vez na árvore, usando como critério a aresta com menor custo entre a árvore atual e os vértices restantes.
- Complexidade: $\mathcal{O}(mn)$ (existem implementações mais eficientes)

Definição 15

A franja (fringe) *de uma subárvore (subgráfo que seja uma árvore) T de um grafo G é o conjunto de todas as arestas de G que têm uma ponta em T e outra ponta fora.*

Algoritmo de Prim II

- Considere o grafo G apresentado na Figura 10 sendo que as arestas em preto denotam uma subárvore T (subgrafo conexo). Na Figura 11 são mostradas em vermelho as arestas da franja entre T e G.

Algoritmo de Prim III

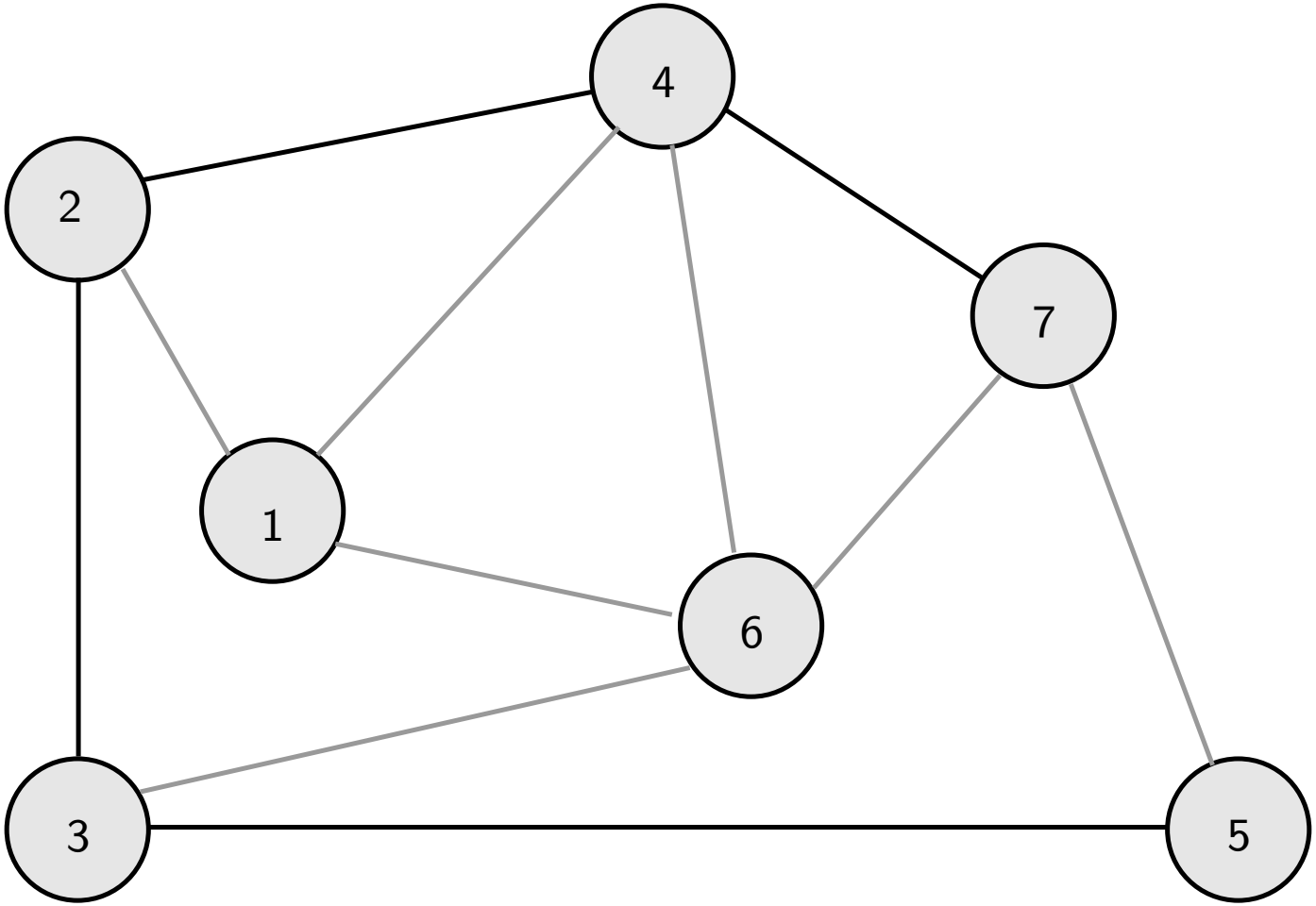

Figura 10: Grafo com as arestas em preto denotando uma subárvore.

Algoritmo de Prim IV

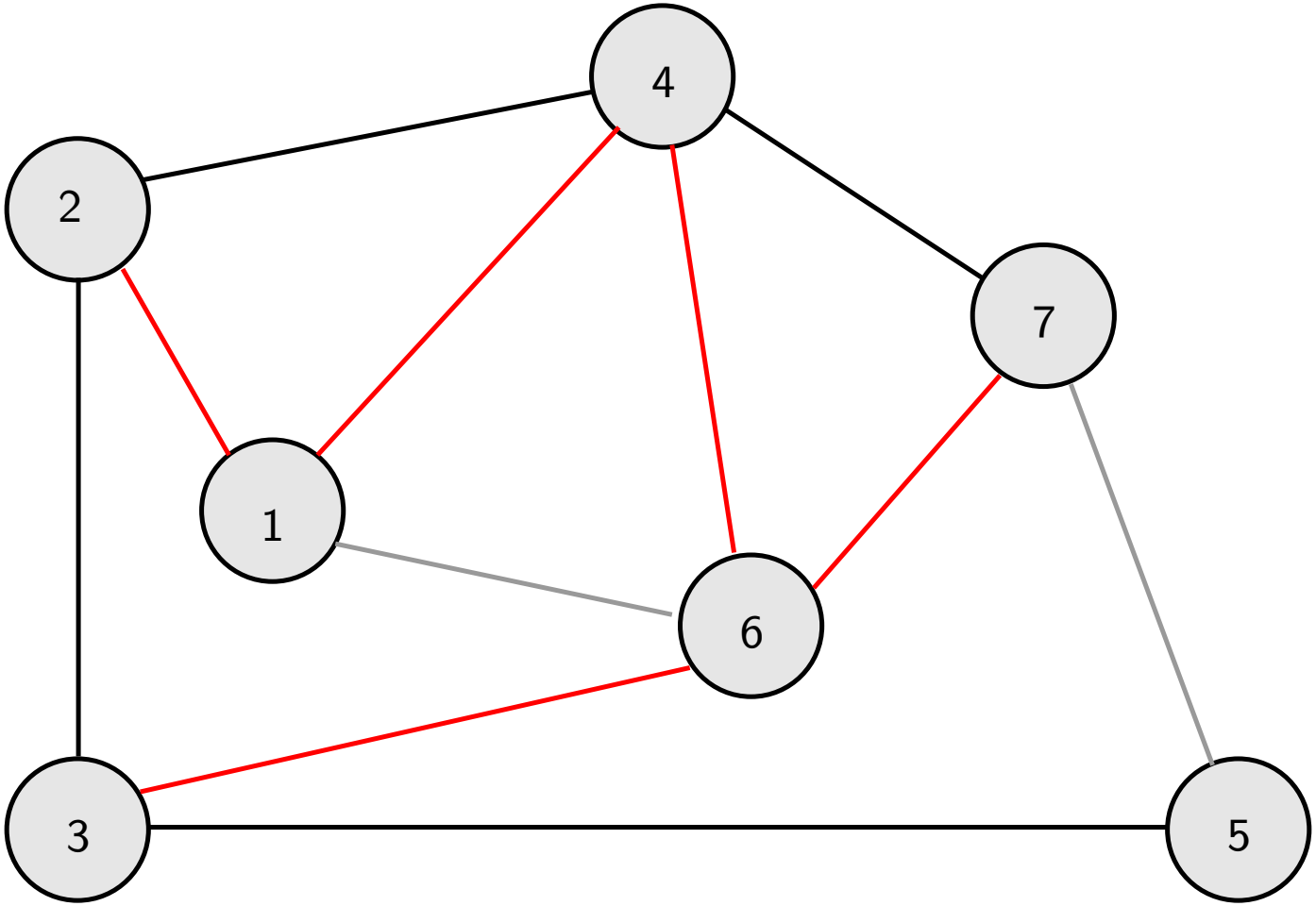

Figura 11: Grafo com as arestas em vermelho denotando a franja.

Algoritmo de Prim V

Algorithm 2 Algoritmo de Prim

1: Inicializar a árvore com um vértice do grafo escolhido arbitrariamente.
2: **enquanto** árvore não estiver completa (n vértices) **faça**
3: introduza na árvore a aresta com menor custo formada pelos vértices da árvore atual e os vértices restantes (conjunto "franja").
4: **fim enquanto**

Proposição

O algoritmo de Prim calcula a árvore geradora mínima de qualquer grafo conexo.

■ Prova: Imediata a partir do teorema otimalidade baseada em corte. A árvore em crescimento sempre define um corte que exclui as arestas que já estão na árvore e o algoritmo seleciona a aresta da franja com menor peso, garantindo que a árvore cresce de forma ótima até estar completa.

Algoritmo de Prim VI

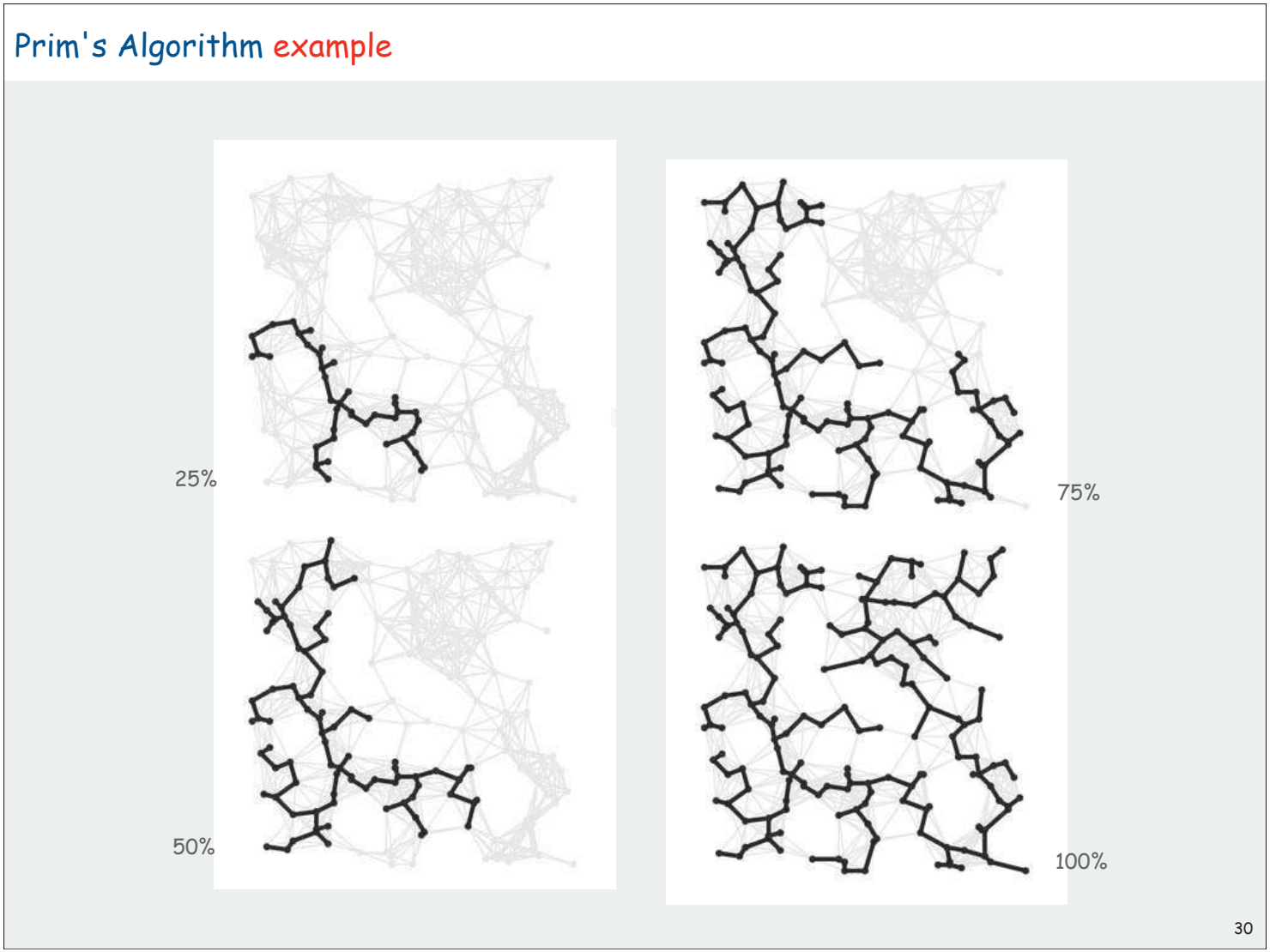

Figura 12: Fonte: Algorithms in Java, Chapter 20

Exemplo I

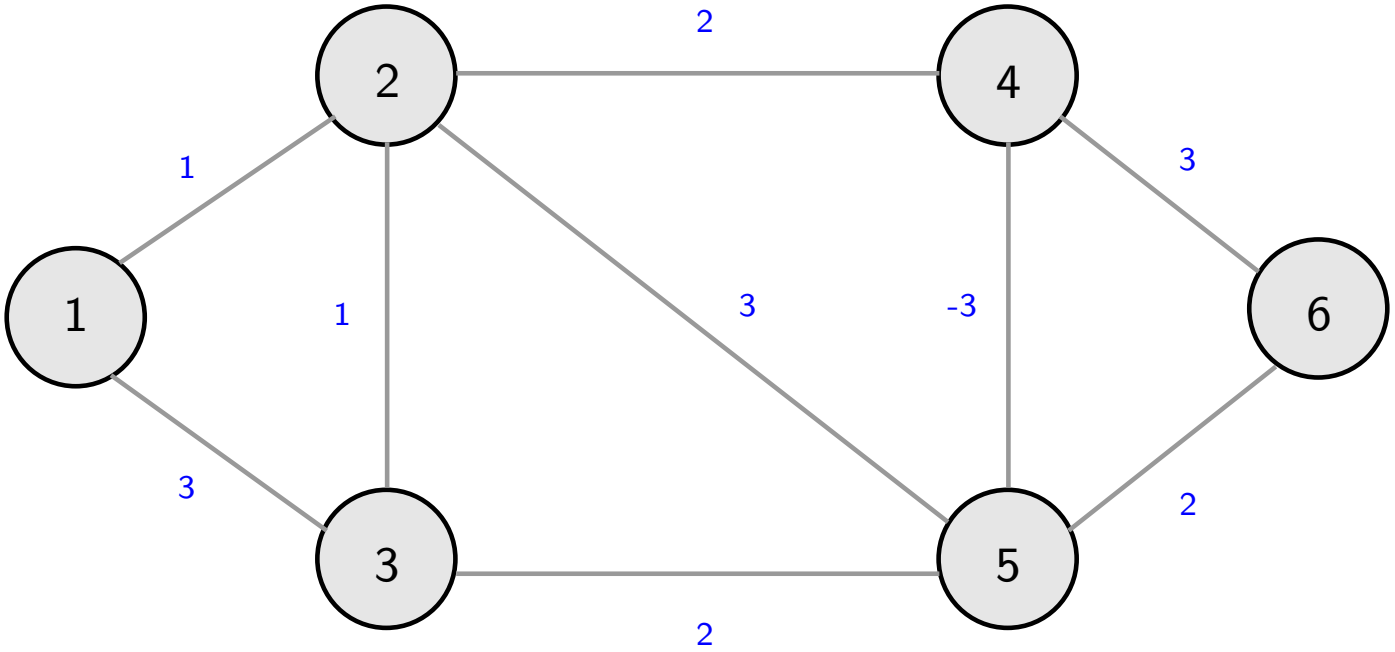

Figura 13: Exemplo para a aplicação do algoritmo de Prim

Exemplo II

Nós na AGM	subárvore	franja	custo
1	–	**(1,2)** (1,3)	0
1,2	(1,2)	(2,4)(2,5)**(2,3)**(1,3)	1
1,2,3	(1,2)(2,3)	**(2,4)**(2,5)(3,5)	2
1,2,3,4	(1,2)(2,3)(2,4)	(2,5)(3,5)**(4,5)**(4,6)	4
1,2,3,4,5	(1,2)(2,3)(2,4)(4,5)	(4,6)**(5,6)**	1
1,2,3,4,5,6	(1,2)(2,3)(2,4)(4,5)(5,6)		3

Algoritmo de Kruskal I

Joseph Bernard Kruskal Jr.

- Em 1956 Kruskal (1928–2010) publicou ("*On the shortest spanning subtree of a graph and the traveling salesman problem*", Proc. Amer. Math. Soc. 7 (1956), 48-50) um algoritmo para o cálculo da árvore geradora mínima.
- Para grafos não conexos $\rightarrow$ floresta geradora mínima (uma árvore geradora mínima para cada componente conexa).
- Estratégia: "Transforma" uma floresta geradora (família de árvores) numa árvore geradora (garantidamente mínima) "juntando" as árvores por meio da inserção ordenada de arestas (menos custosas para mais custosas).
- Complexidade: $\mathcal{O}(m \log n)$

Algoritmo de Kruskal II

Definição 16

Floresta (grafo acíclico) é um grafo sem ciclos não triviais (possui mais de duas arestas). Estendendo essa definição, árvore é uma floresta conexa.

- O grafo mostrado na Figura 14 é um exemplo de floresta (duas árvores desconexas). Unindo-se os vértices 4 e 7, por exemplo, teríamos uma árvore.

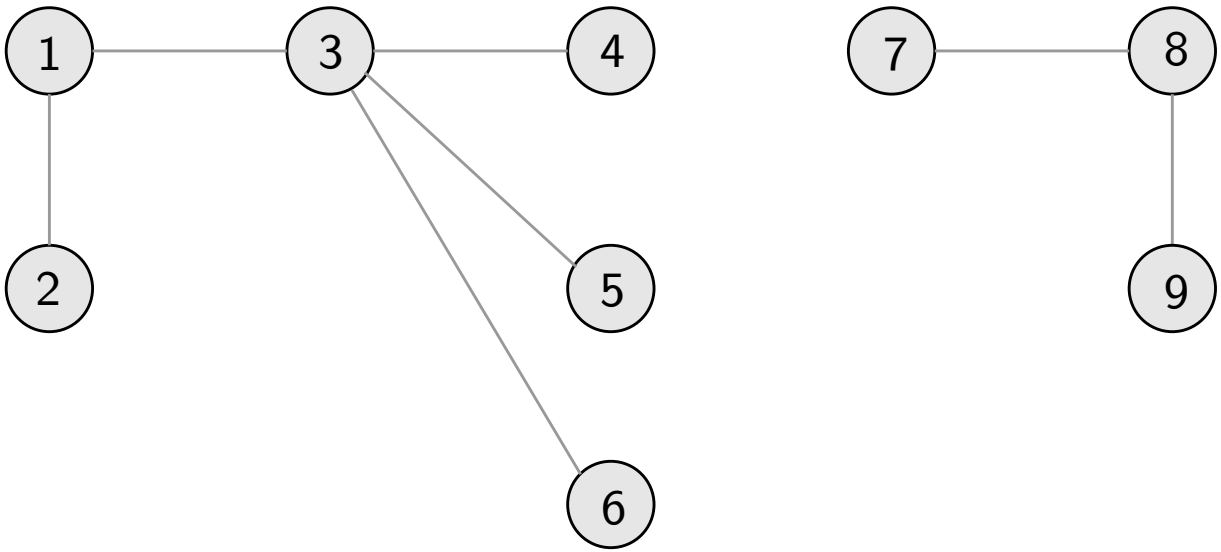

Figura 14: Exemplo de floresta.

Algoritmo de Kruskal III

Algorithm 3 Algoritmo de Kruskal

1: Criar uma floresta F sendo que cada vértice do grafo é uma árvore distinta.
2: Criar um conjunto S contendo todas as arestas do grafo.
3: **enquanto** S não for vazio **faça**
4: remova uma aresta com custo mínimo de S
5: **se** esta aresta conecta duas árvores diferentes **então**
6: adicione à floresta, combinando duas árvores numa única árvore
7: **else**
8: descarte a aresta
9: **fim se**
10: **fim enquanto**

- Ao fim do algoritmo, a floresta tem apenas uma componente e forma uma árvore geradora mínima para o grafo.

Proposição

O algoritmo de Kruskal calcula a árvore geradora mínima de qualquer grafo conexo.

Algoritmo de Kruskal IV

- Prova: Se uma aresta candidata a entrar na árvore criar um ciclo, então seu peso certamente será maior do que todas as arestas do ciclo (inseridas em ordem crescente), portanto ela não está na árvore ótima. Caso ela não crie um ciclo, ela define um corte e certamente é a aresta de menor custo, e portanto pertence à arvore ótima.

Evolução do algoritmo num grafo denso

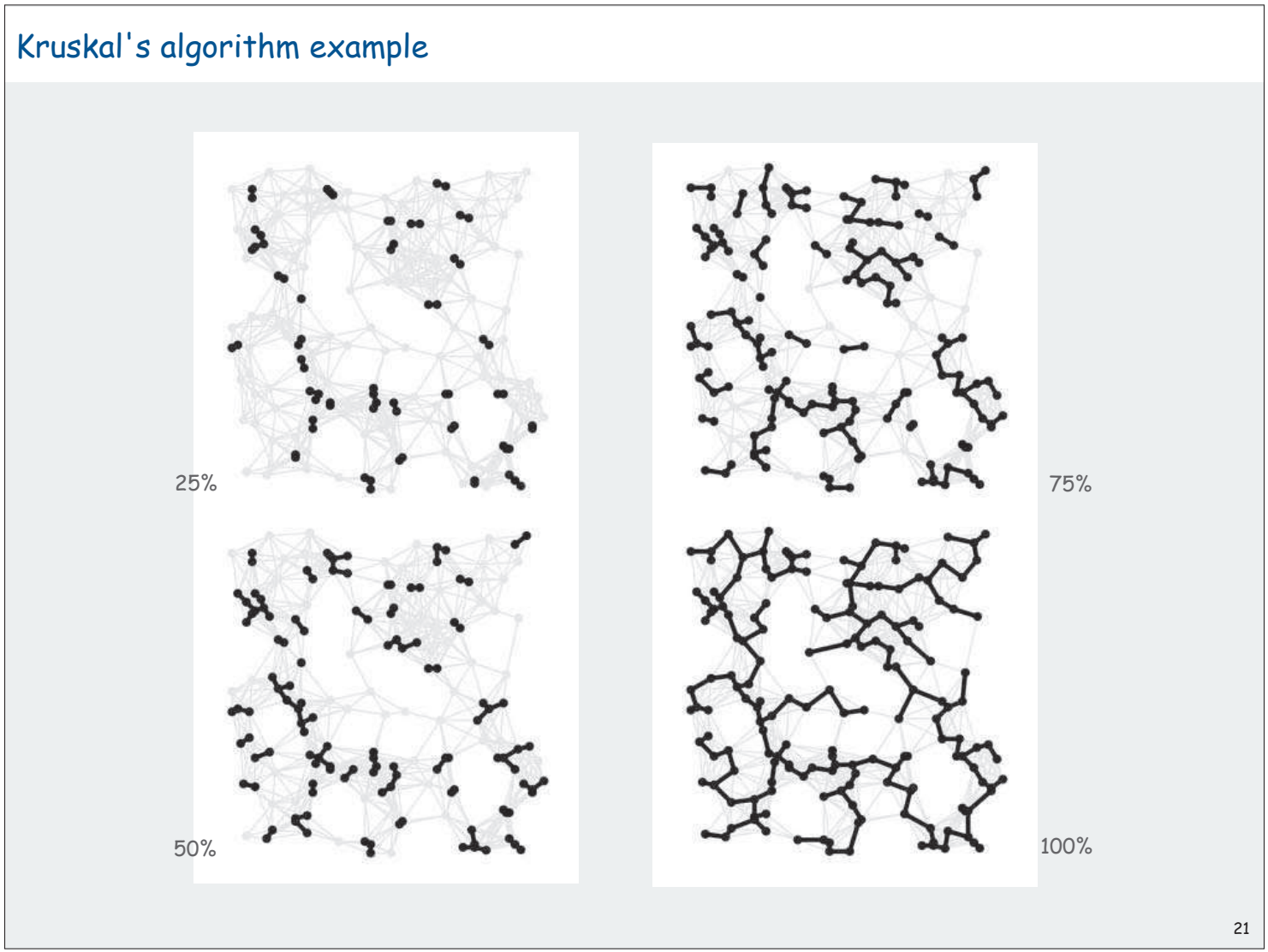

Figura 15: Fonte: Algorithms in Java, Chapter 20

Exemplo I

- Exemplo: Aplique o algoritmo de Kruskal no grafo da Figura 16.

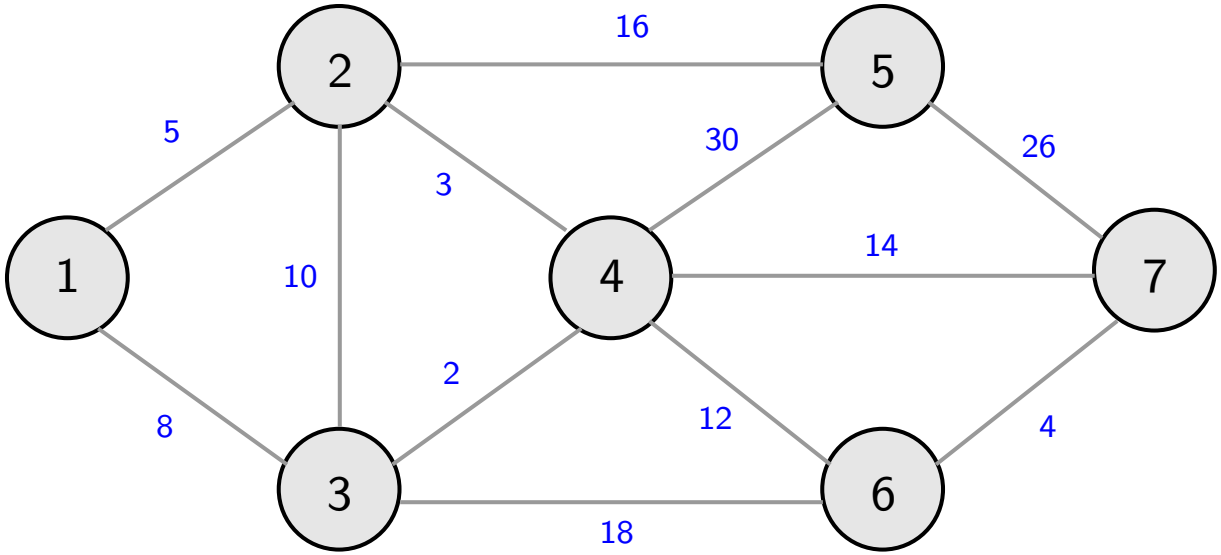

Figura 16: Exemplo para aplicação do algoritmo de Kruskal.

- Ordenamento das arestas em função dos custos associados (do menor para o maior).

Exemplo II

Aresta	(3,4)	(2,4)	(6,7)	(1,2)	(1,3)	(2,3)	(4,6)	(4,7)	(2,5)	(3,6)	(5,7)	(4,5)
Custo	2	3	4	5	8	10	12	14	16	18	26	30

Árvore	custo
–	0
(3,4)	2
(3,4) (2,4)	5
(3,4) (2,4) (6,7)	9
(3,4) (2,4) (6,7) (1,2)	14
(3,4) (2,4) (6,7) (1,2) (4,6)	26
(3,4) (2,4) (6,7) (1,2) (4,6) (2,5)	42

Algoritmo de Borůvka I

Otakar Borůvka

- Em 1926 Borůvka (1899–1995) publicou um algoritmo para o cálculo da árvore geradora mínima (aplicado a uma rede elétrica).
- Para grafos não conexos $\rightarrow$ floresta geradora mínima (uma árvore geradora mínima para cada componente conexa).
- Estratégia "híbrida" entre os métodos de Prim e Kruskal.
- Complexidade: $\mathcal{O}(m \log n)$

■ Assim como o Kruskal, evolui uma floresta de árvores (inicialmente contendo apenas um vértice cada) por meio da inserção de arestas. De modo similar ao Prim, adiciona a aresta de menor custo conectada a cada árvore.

Algoritmo de Borůvka II

Algorithm 4 Algoritmo de Borůvka

1: Inicialize todos os vértices como árvores individuais T_i de uma floresta F.
2: Inicialize a árvore geradora como vazia;
3: **enquanto** a árvore geradora não estiver completa **faça**
4: **para** cada árvore da floresta **faça**
5: Encontre a aresta de menor custo que conecta esta árvore a qualquer outra.
6: Adicione essa aresta à árvore geradora (se já não estiver presente).
7: **fim para**
8: Verifique a possibilidade de unir árvores
9: **fim enquanto**

Exemplo I

- Aplique o algoritmo de Borůvka no grafo da Figura 17

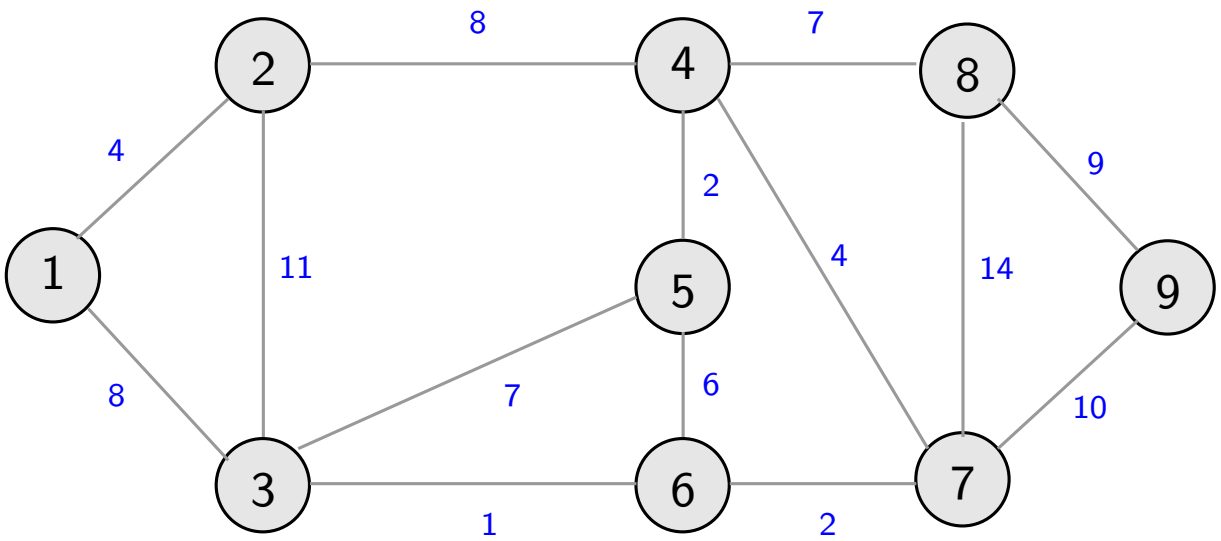

Figura 17: Exemplo para aplicação do algoritmo de Borůvka.

- Inicialização: $F = \{T_1, T_2, \ldots, T_9\},\ T_i = \{i\},\ i = 1, \ldots, 9.$
- Primeira iteração:

Exemplo II

Componente	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$	$\{6\}$	$\{7\}$	$\{8\}$	$\{9\}$
Aresta menor custo	$(1,2)$	$(2,1)$	$(3,6)$	$(4,5)$	$(5,4)$	$(6,3)$	$(7,6)$	$(8,4)$	$(9,8)$

Atualização das componentes: $F = \{T_1, T_2, T_3\}$, $T_1 = \{1,2\}$, $T_2 = \{3,6,7\}$, $T_3 = \{4,5,8,9\}$.

- Segunda iteração:

Componente	$\{1,2\}$	$\{3,6,7\}$	$\{4,5,8,9\}$
Aresta menor custo	$(2,4)$ (ou $(1,3)$)	$(7,4)$	$(4,7)$

Atualização das componentes: $F = \{T_1\}$, $T_1 = \{1,2,3,4,5,6,7,8,9\}$.

- Observação: algoritmo passível de implementação paralela.

Problema do caixeiro viajante I

- Seja um grafo em que os vértices são as cidades, as arestas são as estradas entre as cidades, e o custo das arestas são as distâncias das estradas. O problema do caixeiro viajante consiste em encontrar a viagem de menor custo que passe por todas as cidades (uma única vez) e retorne à cidade de origem (ciclo Hamiltoniano). A Figura 19 ilustra um exemplo de 4 cidades.

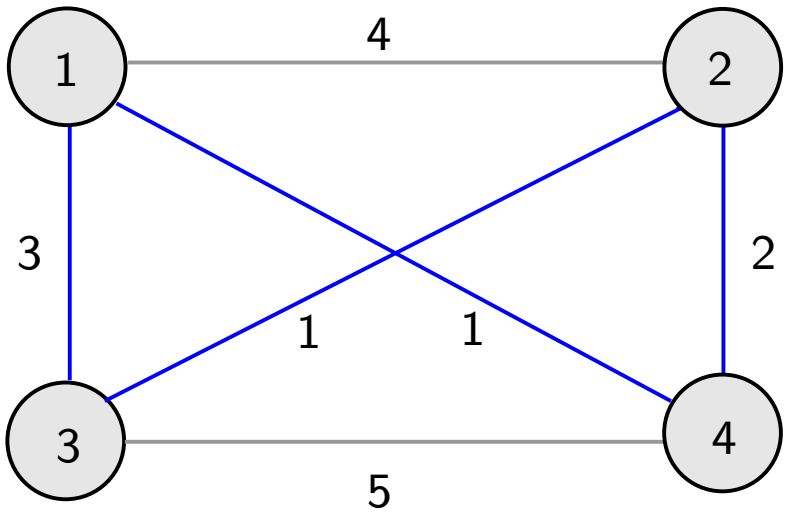

Figura 18: Viagem de custo mínimo (arestas em azul): $1 \to 4 \to 2 \to 3 \to 1$. Custo: 7

Problema do caixeiro viajante II

Formalmente

Dado um grafo completo não direcionado $G(N, A)$ com custos inteiros não negativos c_{ij} associados a cada aresta (i, j) que pertence a G, o problema é encontrar um ciclo Hamiltoniano (viagem (em inglês *tour*)) de G com custo mínimo.

- Problema NP-completo! Contudo, podemos encontrar algoritmos aproximados (não garantem o ótimo) polinomiais.

- Uma opção de algoritmo aproximado: via árvore geradora mínima.

- Também assumiremos a seguinte hipótese (o problema continua sendo NP-completo): Os custos das arestas atendem a restrição $c_{ij} \leq c_{ik} + c_{kj}$ (desigualdade triangular).

Limitante inferior

Lema 1

Seja $H^\star$ a viagem de custo ótimo em um grafo G. Então

$$c(T^\star) \leq c(H^\star) \tag{1}$$

sendo $c(T^\star)$ o custo da árvore geradora mínima.

- A prova é simples (intuitiva): Ao retirar uma aresta da viagem ótima, temos uma árvore geradora (concordam?). Não sabemos o custo dela, mas sabemos que certamente ela tem custo menor que $c(H^\star)$. Como conhecemos o custo da árvore com menor custo de todas, concluímos que $c(T^\star) \leq c(H^\star)$.

Algoritmos de aproximação

- Considere um problema de minimização e um algoritmo $\mathscr{A}$ que, para uma dada instância I, retorna uma solução $\mathscr{A}(I)$

Definição 17

A taxa de aproximação do algoritmo $\alpha_{\mathscr{A}}$ é tal que

$$1 \leq \frac{\mathscr{A}(I)}{OPT(I)} \leq \alpha_{\mathscr{A}}$$

- Em outras palavras, $\alpha_{\mathscr{A}}$ mede o fator pelo qual a saída do algoritmo $\mathscr{A}$ excede, para a instância de pior caso, a solução ótima do problema.

- Se o algoritmo produz a solução ótima, por exemplo, o algoritmo de Prim para a árvore geradora mínima, então $\alpha_{\mathscr{A}} = 1$.

Algoritmo aproximado TSP-MSP I

- Considere o algoritmo aproximado para o problema do caixeiro viajante que utiliza o resultado da árvore geradora mínima (TSP-MSP) como ponto de partida.

Algorithm 5 Algoritmo TSP-MSP

1: Escolha um vértice $v \in G$ para ser o vértice raiz.
2: Compute a árvore geradora mínima $T^\star$ via Prim utilizando v como vértice inicial.
3: Crie um percurso completo P sobre $T^\star$ começando e terminado em v (cada aresta é percorrida duas vezes).
4: Usando a desigualdade triangular, extraia um ciclo Hamiltoniano H a partir de P (equivale a percorrer $T^\star$ em pré-ordem).
5: Retorne H.

Lema 2

O Algoritmo TSP-MSP é um algoritmo de aproximação $\alpha_{TSP\text{-}MSP} = 2$ para o problema do caixeiro viajante.

Algoritmo aproximado TSP-MSP II

- Prova: Do Lema 1 sabemos que $c(T^\star) \leq c(H^\star)$. Seja P um percurso completo sobre $T^\star$, passando exatamente 2 vezes por cada aresta. Assim tem-se

$$c(P) = 2c(T^\star) \tag{2}$$

Dessas relações concluímos que

$$c(P) \leq 2c(H^\star) \Rightarrow \frac{c(P)}{c(H^\star)} \leq 2$$

ou seja, o custo $c(P)$ está dentro de um fator de 2 da viagem ótima.

- Problema: P não é um viagem (repete vértices).

- Solução: eliminar as visitas repetidas (aos vértices) usando a desigualdade triangular sem aumentar o custo (equivale a percorrer a árvore em pré-ordem). Seja P^* o percurso da árvore $T^\star$ em pré-ordem. Assim temos

$$c(P^*) \leq c(P)$$

Algoritmo aproximado TSP-MSP III

- Combinando os resultados, temos finalmente que

$$c(P^*) \leq 2c(H^\star) \Rightarrow \frac{c(P^*)}{c(H^\star)} \leq 2$$

- Se eliminarmos a hipótese da desigualdade triangular: a aproximação não pode ser mais realizada em tempo polinomial (a não ser que $P = NP$).

- Existem outras estratégias de aproximação que funcionam melhor na prática. Por exemplo, o algoritmo de Christofides (fator de aproximação $\alpha_{\text{Chr}} = 1.5$)

Exemplo

- Considere o grafo mostrado na Figura 19 no qual as arestas na cor azul representam a árvore geradora mínima (obtida via Prim).

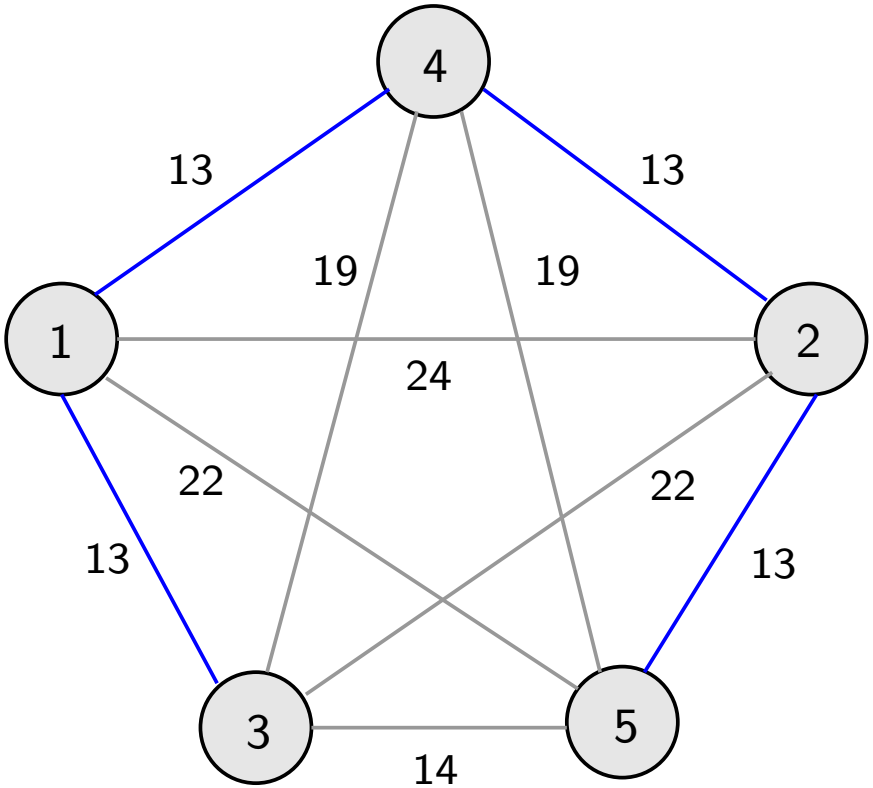

Figura 19: Grafo completo com 5 vértices ($N = 5$) com árvore geradora mínima em azul ($c(T^\star) = 52$).

Exemplo

- A Figura 20 mostra um percurso completo sobre $T^\star$ partindo do nó 1, fornecendo um custo de $2c(T^\star) = 104$. Note diversos vértices são visitados 2 vezes (portanto o percurso não é uma viagem).

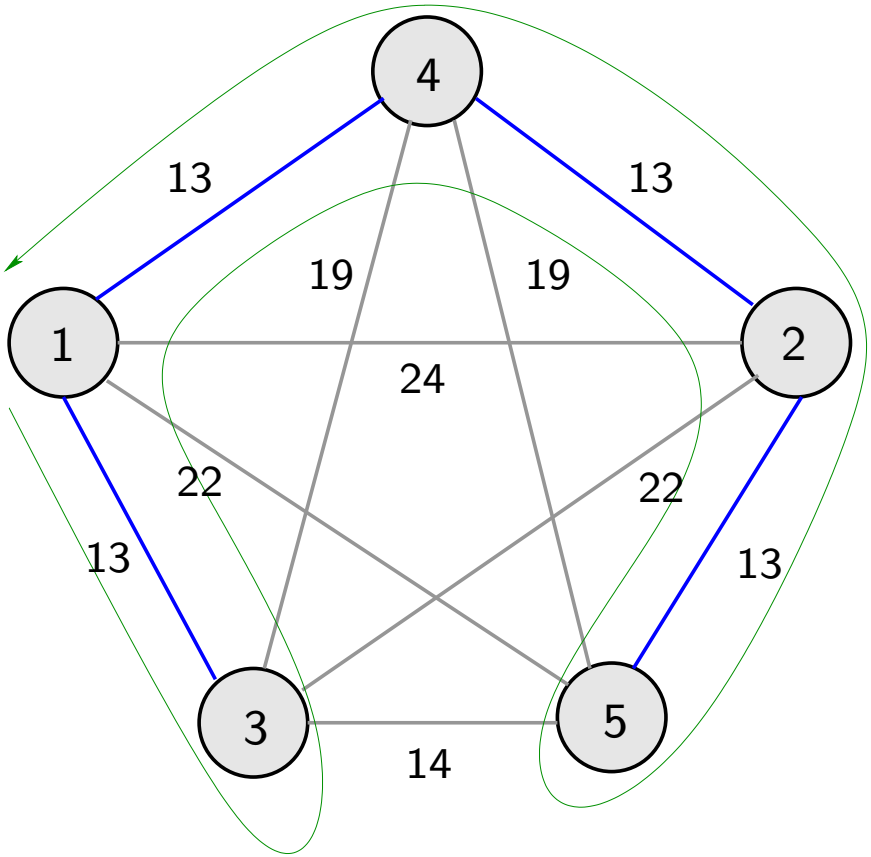

Figura 20: Percurso completo (ainda não é uma viagem) sobre a árvore geradora com custo $2c(T^\star) = 104$.

Exemplo

- A Figura 21 mostra uma viagem enraizada no vértice 1 (obtida por meio do emprego da desigualdade triangular), fornecendo um custo de $c(P^*) = 80$.

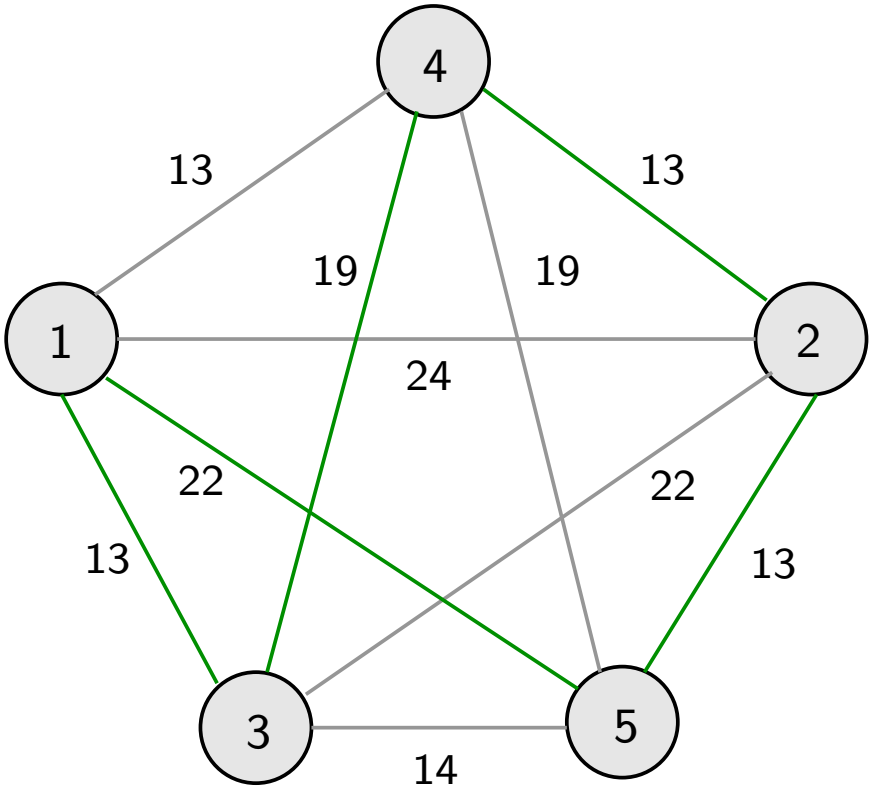

Figura 21: Ciclo hamiltoniano obtido a partir da Figura 20 usando a desigualdade triangular.

Exemplo

- A Figura 22 mostra a viagem ótima $H^\star$. Note que $c(P^*) \leq 2c(H^\star) \Rightarrow 80 \leq 2 \times 66$, e também que $c(T^\star) \leq c(H^\star) \Rightarrow 52 \leq 66$.

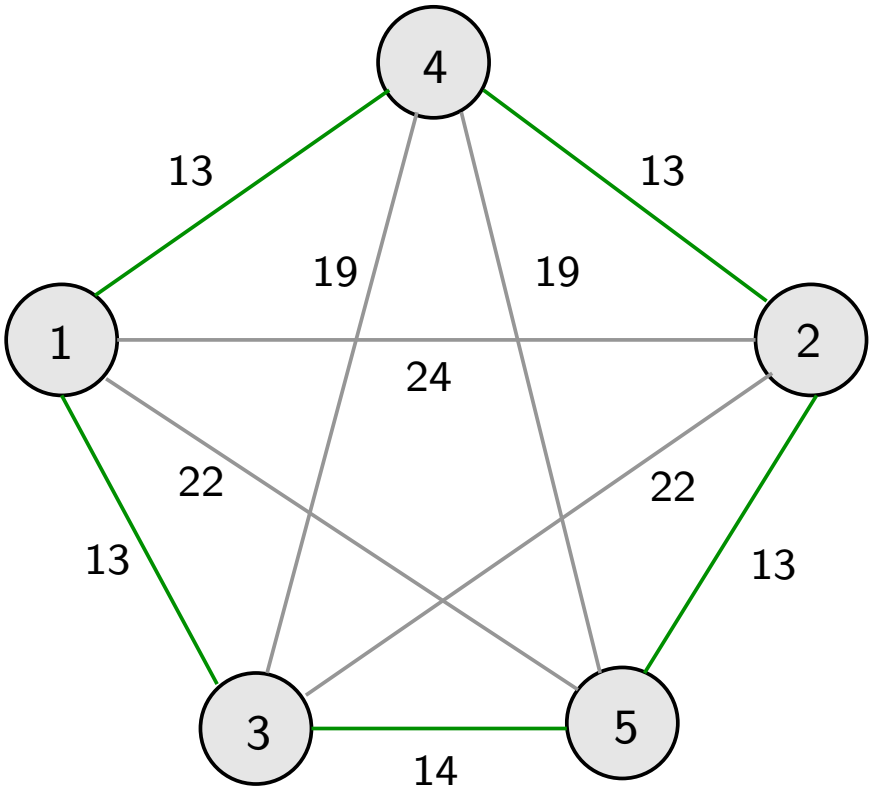

Figura 22: Viagem ótima $c(H^\star) = 66$.

Referências I


R. K. Ahuja, T. L. Magnanti, and J. B. Orlin.
Network flows: theory, algorithms, and applications.
Prentice-Hall, Upper Saddle River, NJ, 1993.


T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein.
Introduction to Operations Research.
MIT Press, Cambridge, Massachusetts, 3 edition, 2009.


P. Feofiloff.
Algoritmos para grafos em C via sedgewick.
`http://www.ime.usp.br/~pf/algoritmos_para_grafos/index.html`.
Acessado: Setembro de 2016.


M. C. Goldbarg and H. P. L. Luna.
Otimização combinatória e programação linear – Modelos e Algoritmos.
Elsevier, Rio de Janeiro, RJ, 2 edition, 2005.

Referências II

R. Sedgewick and K. Wayne.

Algorithms.

Pearson Education, Boston, MA, 4 edition, 2011.