Algoritmos de aproximação - Problema do caixeiro viajante

Marina Andretta

ICMC-USP

30 de setembro de 2015

Baseado no livro Uma introdução sucinta a Algoritmos de Aproximação, de M. H. Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, K. S. Guimarães, F. K. Miyazawa, J. C. Piña Jr., J. A. R. Soares e Y. Wakabayashi.

Problema do caixeiro viajante

Lembre-se que um circuito hamiltoniano é um circuito que contém todos os vértices do grafo.

O problema do caixeiro viajante (*traveling salesman problem*), denotado por TSP, é definido da seguinte maneira:

Problema $\mathrm{TSP}(G, c)$: Dados um grafo G e um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e, determinar um circuito hamiltoniano C que minimize $c(C)$.

Problema do caixeiro viajante

Esse é talvez o mais famoso problema de otimização combinatória, em parte graças às conexões com vários outros problemas de otimização.

Ele é NP-difícil mesmo se $c_e \in \{1, 2\}$ para toda aresta e.

Além disso, não se conhece um algoritmo de aproximação com razão constante para o problema.

Nos restringimos a um caso particular do TSP que admite algoritmo de aproximação com razão constante.

Caixeiro viajante métrico

Suponha que o grafo G é completo e temos um custo c_{ij} associado a cada par ij de vértices.

Dizemos que os custos satisfazem a desigualdade triangular se

$$c_{ik} \leq c_{ij} + c_{jk}$$

para quaisquer três vértices i, j e k.

Caixeiro viajante métrico

O TSP restrito ao conjunto de instâncias (G, c) em que G é completo e c satisfaz a desigualdade triangular é conhecido como problema do caixeiro viajante métrico e será denotado aqui por TSPM.

Este problema também é NP-difícil.

Preliminares

Antes de apresentarmos dois algoritmos de aproximação para o TSPM, precisamos de algoritmos polinomiais para resolver três problemas importantes:

- árvore geradora de custo mínimo;
- ciclo euleriano;
- emparelhamento perfeito.

Problema da árvore geradora de custo mínimo

Problema $\mathrm{MST}(G, c)$: Dados um grafo G e um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e, encontrar uma árvore geradora de custo mínimo.

Existem algoritmos simples e eficientes para construir uma árvore geradora de custo mínimo em um grafo conexo.

Vamos designar por MST um algoritmo qualquer desse tipo.

Problema da árvore geradora de custo mínimo

Um algoritmo para resolver este problema, com $G = (V, E)$ um grafo conexo, é o algoritmo de Kruskal, proposto em 1956.

Algoritmo MST-KRUSKAL(G, c):

1 Faça $T \leftarrow \emptyset$;

2 faça $A \leftarrow E$;

3 enquanto $A \neq \emptyset$ e T não é uma árvore geradora, faça:

4 seja e uma aresta de A com menor custo c_e;

5 faça $A \leftarrow A \setminus \{e\}$;

6 se $T \cup \{e\}$ não contém um ciclo

7 então $T \leftarrow T \cup \{e\}$;

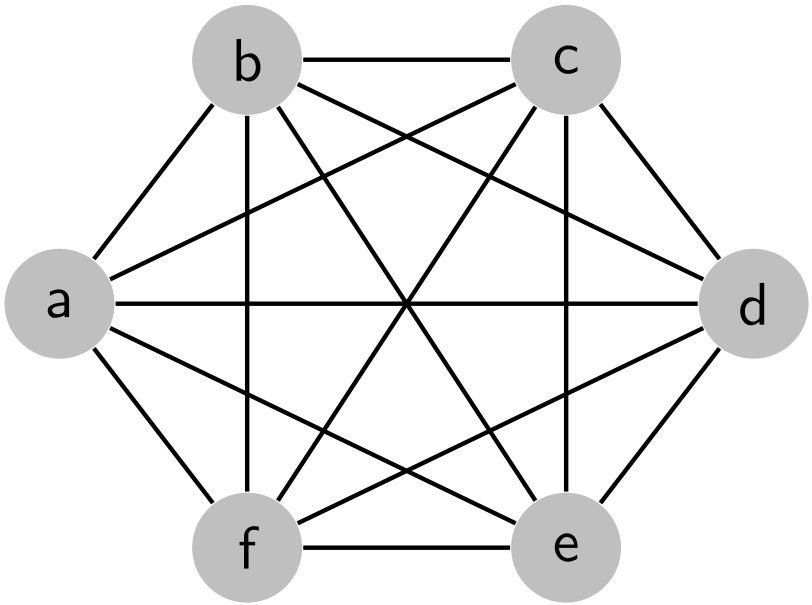
8 devolva T.

Problema da árvore geradora de custo mínimo

Claramente, este algoritmo é polinomial no número de vértices e arestas de G.

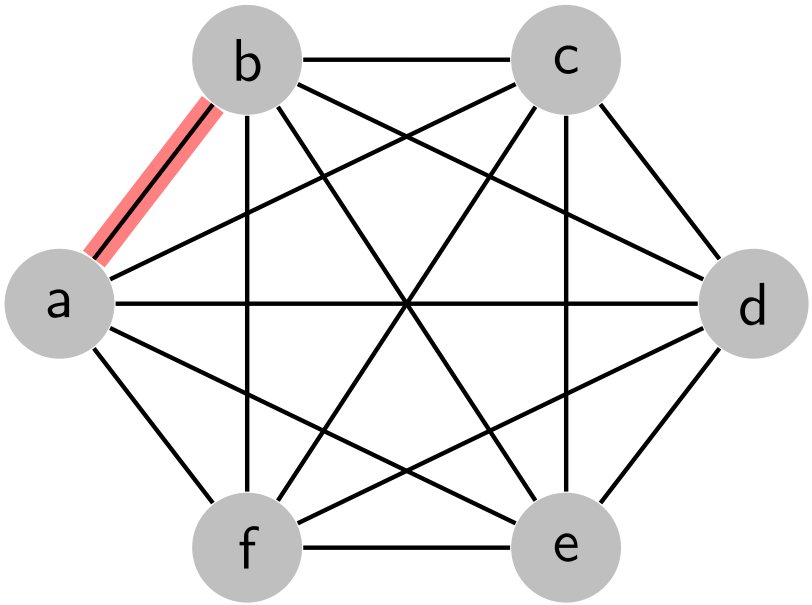
Vejamos como ele funciona através de um exemplo.

Problema da árvore geradora de custo mínimo



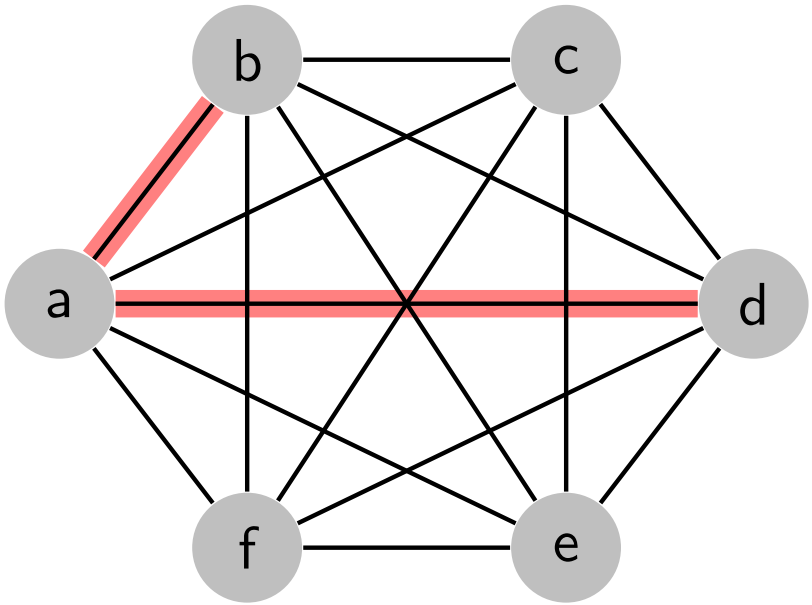
	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



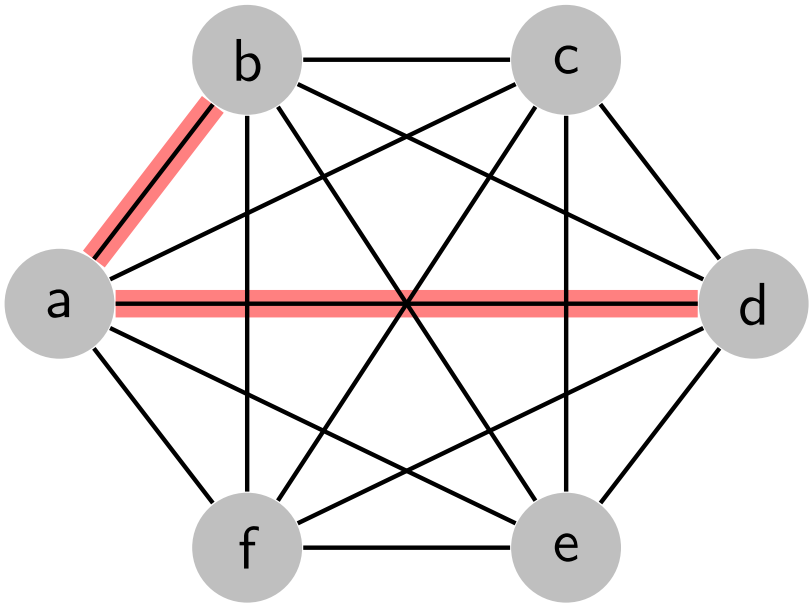
	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



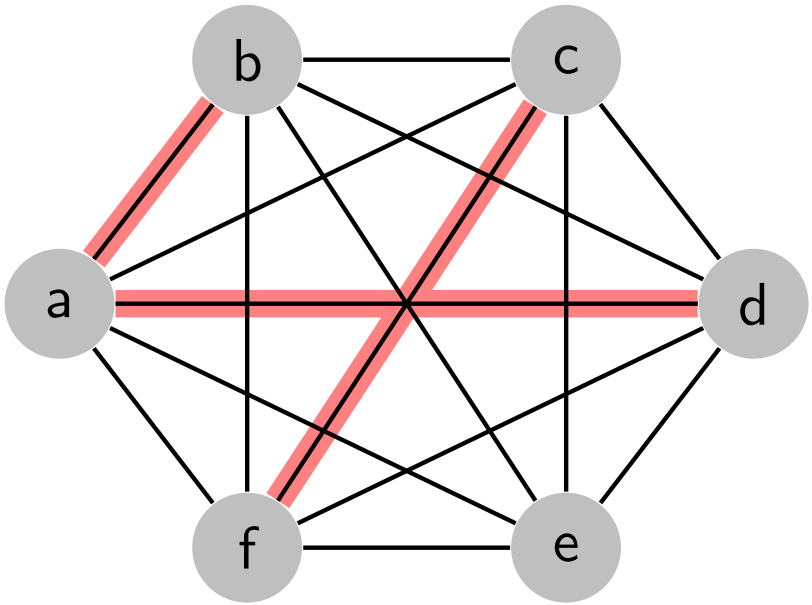
	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



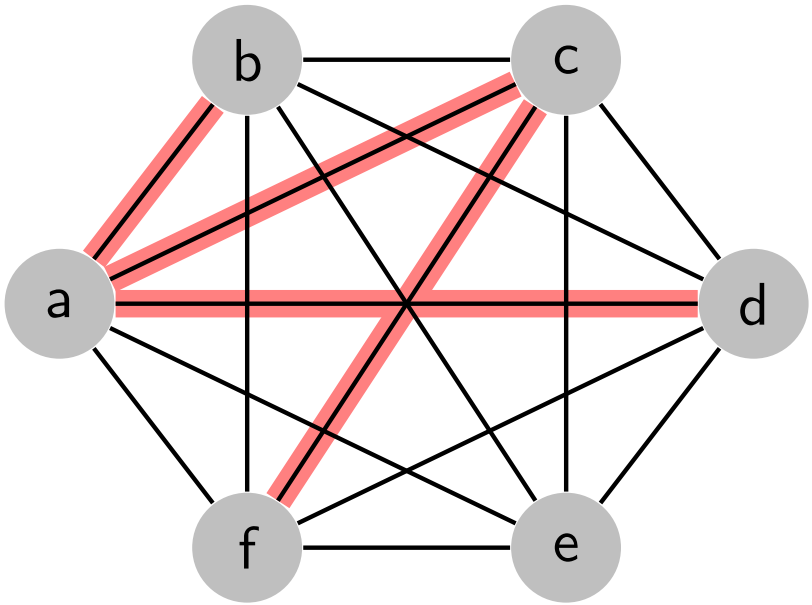
	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



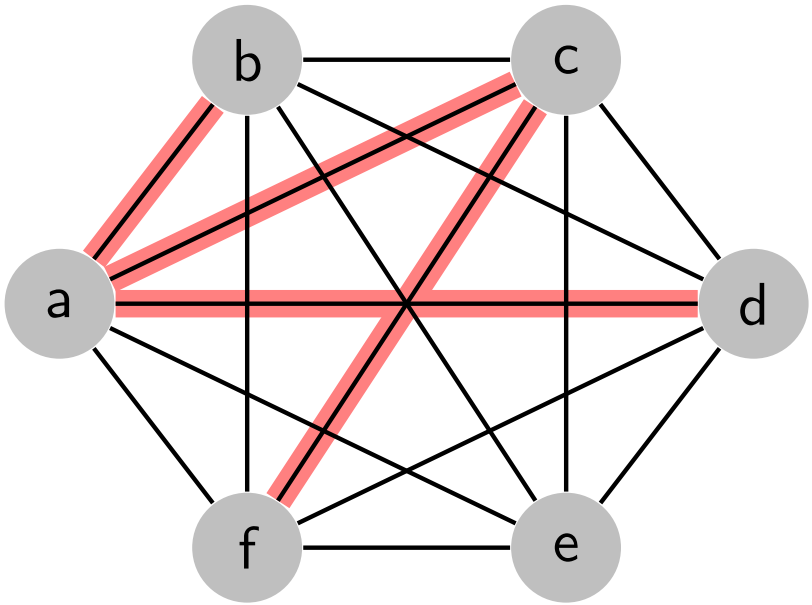
	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



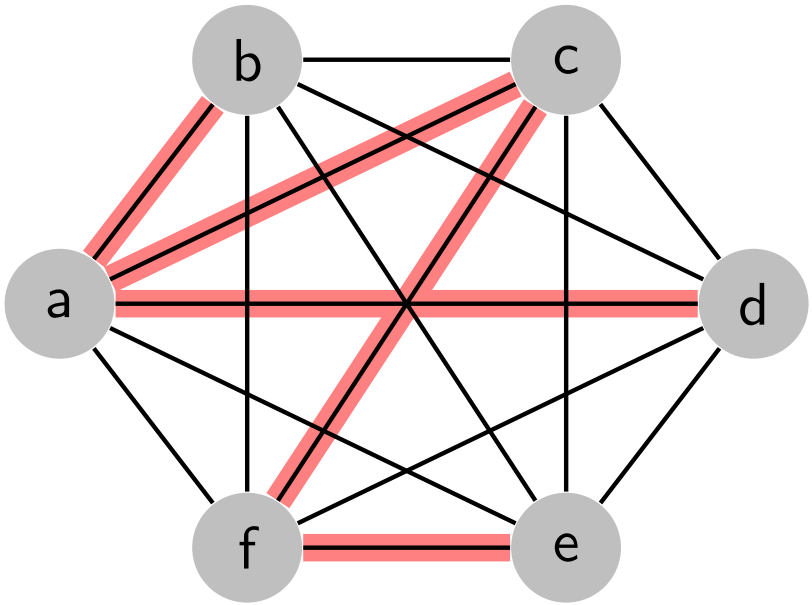
	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



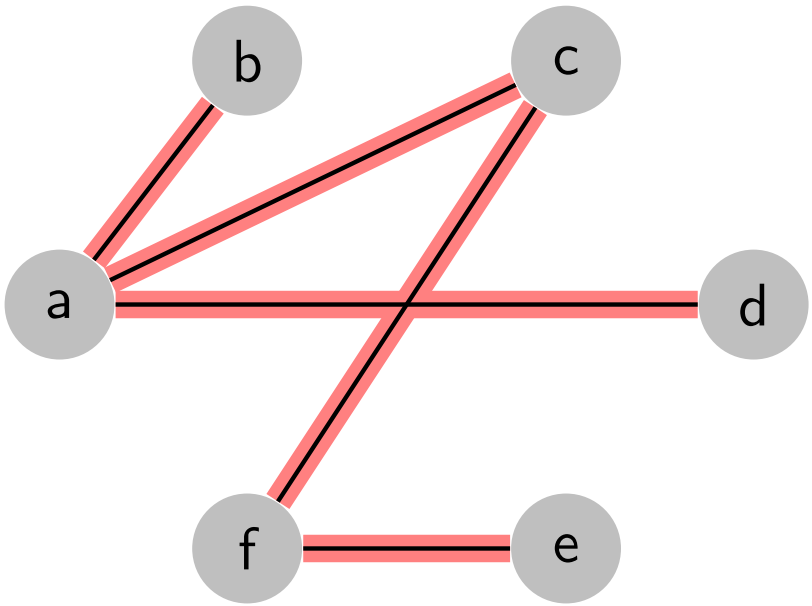
	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



	a	b	c	d	e	f
a	-	1	2	1	7	2
b	1	-	7	1	4	3
c	2	7	-	3	5	1
d	1	1	3	-	8	5
e	7	4	5	8	-	2
f	2	3	1	5	2	-

Problema da árvore geradora de custo mínimo



	a	b	c	d	e	f
a		1	2	1		
b	1					
c	2					1
d	1					
e						2
f			1		2	

Temos que $c(T) = 7$.

Problema do ciclo euleriano

Um ciclo euleriano em um grafo ou multigrafo G é qualquer ciclo que contém todas as arestas de G.

Um multigrafo conexo G tem um ciclo euleriano se e somente se cada um de seus vértices tem grau par.

São bem conhecidos os algoritmos que constroem um ciclo euleriano em um multigrafo conexo sem vértices de grau ímpar. Vamos designar por EULER um algoritmo qualquer desse tipo.

Problema do ciclo euleriano

Um algoritmo para resolver este problema para um grafo G conexo, com todos os vértices com grau par, é o proposto por Hierholzer, em 1873.

Problema do ciclo euleriano

Algoritmo EULER-HIERHOLZER(G):

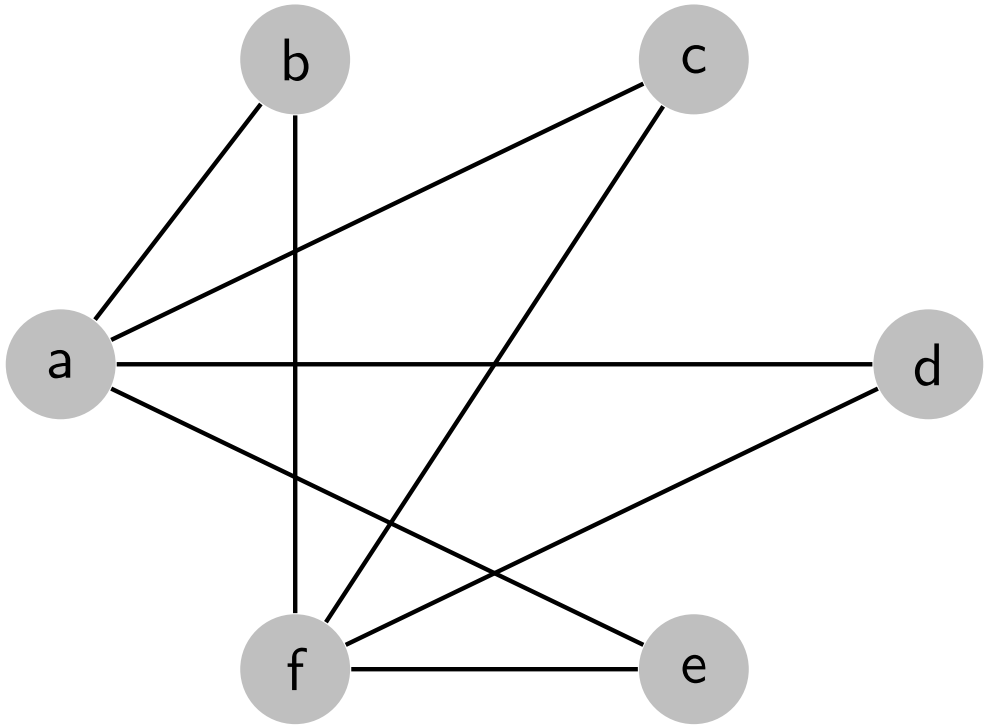
1 Faça $C \leftarrow \emptyset$ e $A \leftarrow E$;
2 seja v um vértice qualquer de G;
3 acrescente v à sequência de vértices C;
4 enquanto $A \neq \emptyset$, faça:
5 se não existe nenhuma aresta vw em A,
6 então escolha um vértice v de C tal que exista $vw \in A$;
7 escolha uma aresta vw de A;
7 acrescente w depois de v na sequência de vértices C;
8 faça $v \leftarrow w$;
9 faça $A \leftarrow A \setminus \{vw\}$;
10 devolva C.

Problema do ciclo euleriano

Claramente, o consumo de tempo do algoritmo EULER-HIERHOLZER é proporcional ao número de arestas do grafo G.

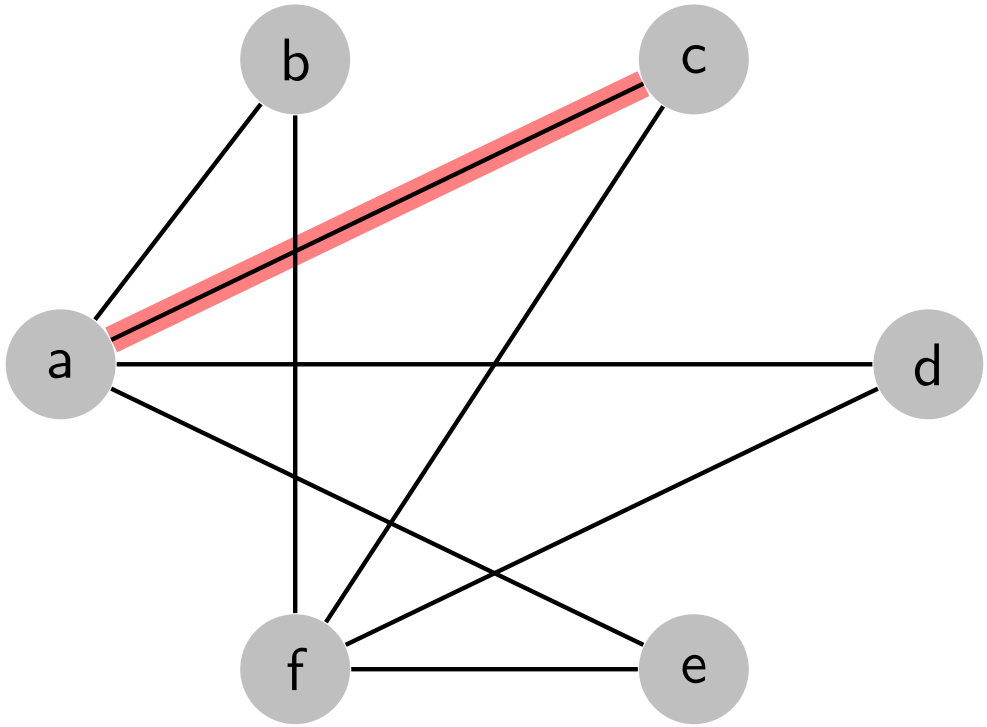
Vejamos um exemplo da execução do algoritmo.

Problema do ciclo euleriano



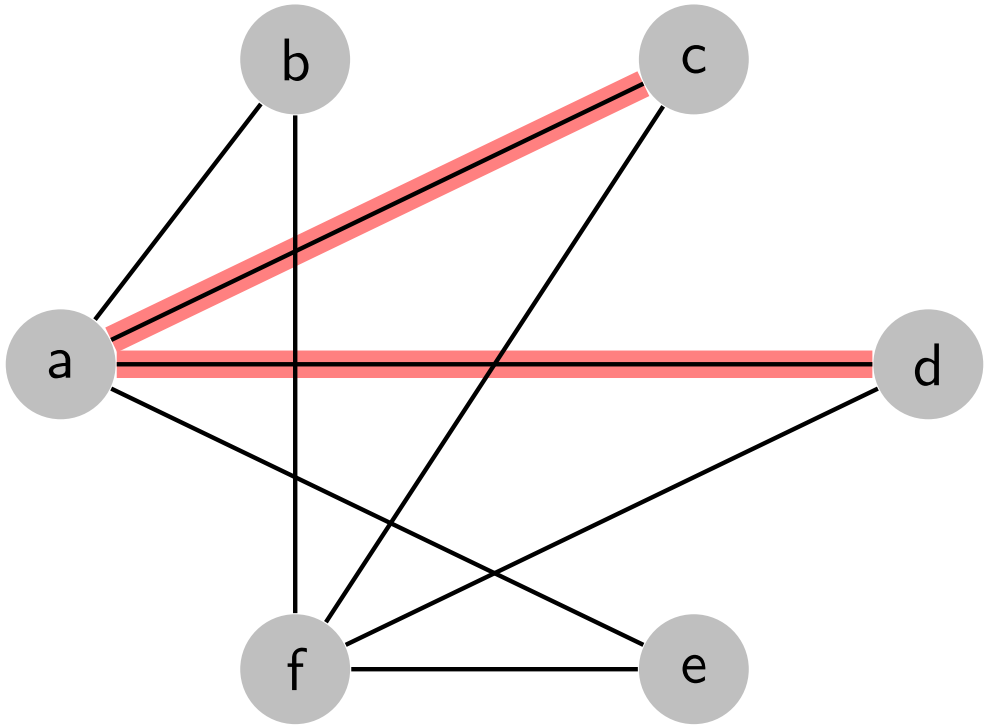
$$C = (c$$

Problema do ciclo euleriano



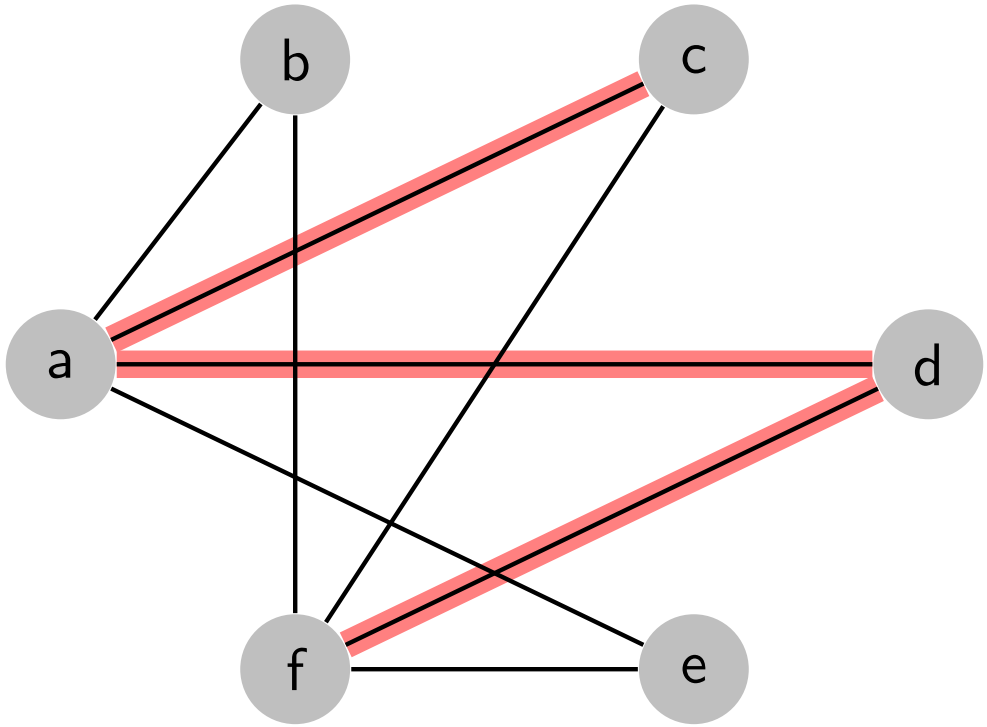
$$C = (c, a$$

Problema do ciclo euleriano



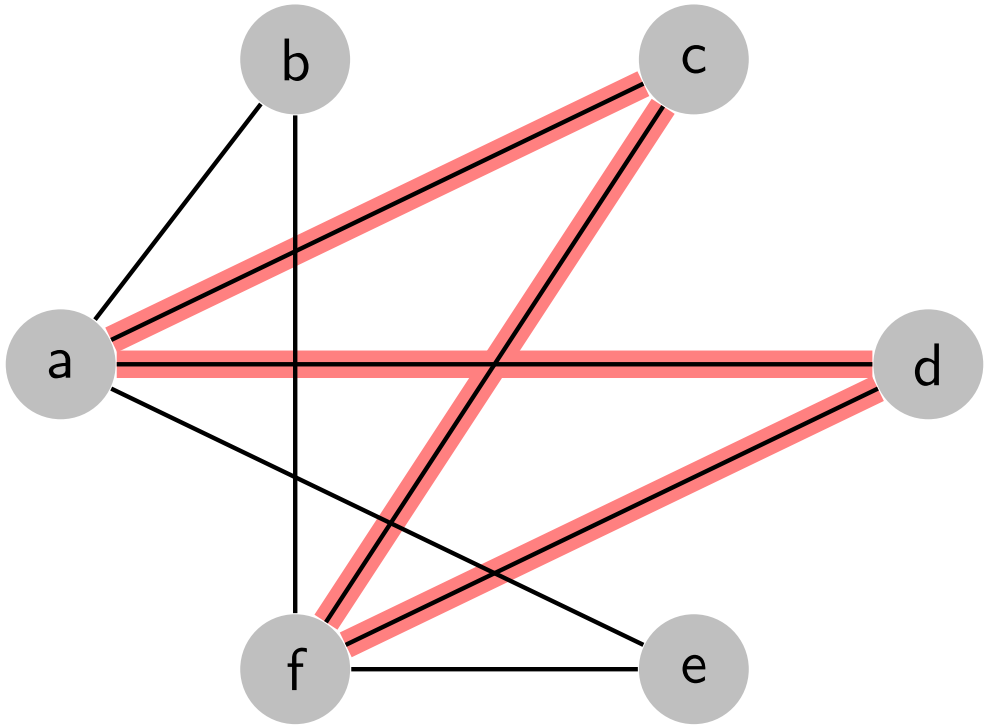
$$C = (c, a, d$$

Problema do ciclo euleriano



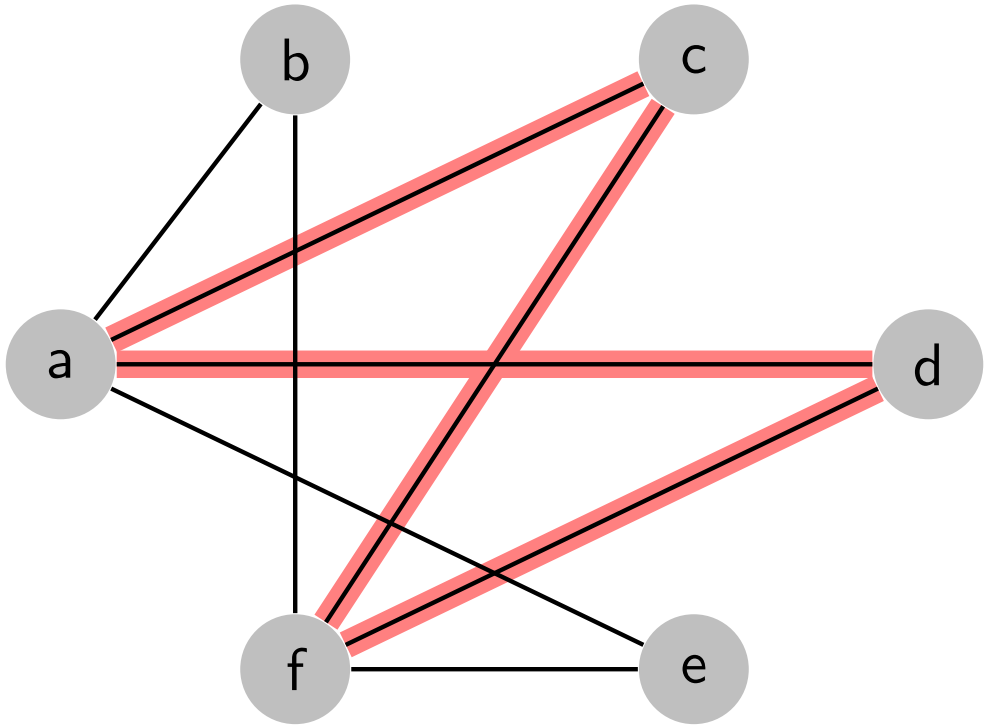
$$C = (c, a, d, f$$

Problema do ciclo euleriano



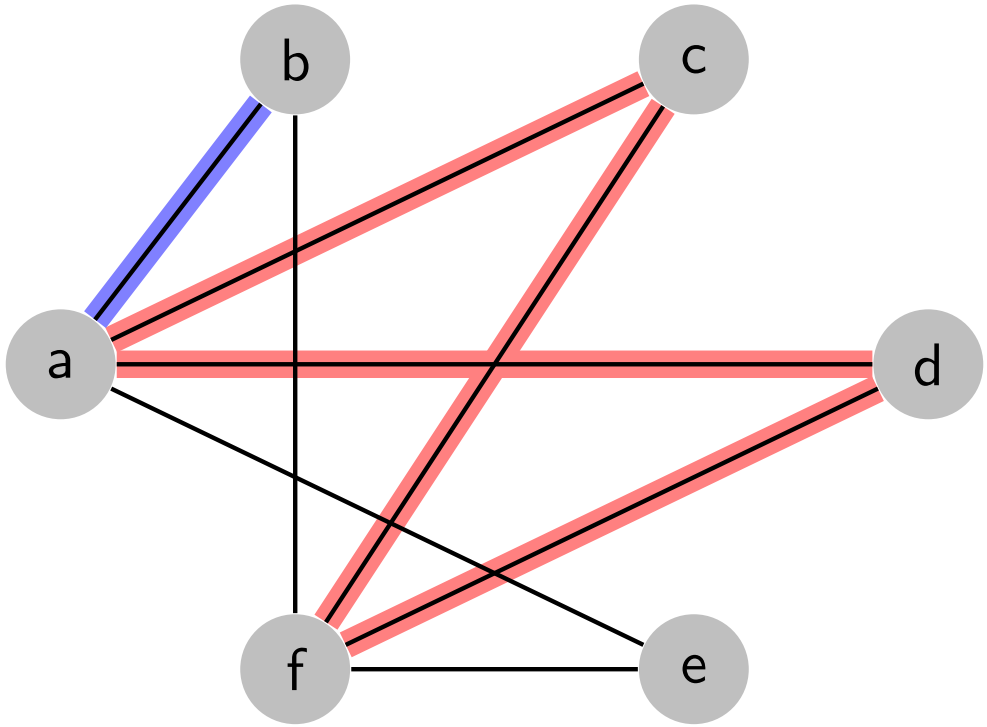
$$C = (c, a, d, f, c)$$

Problema do ciclo euleriano



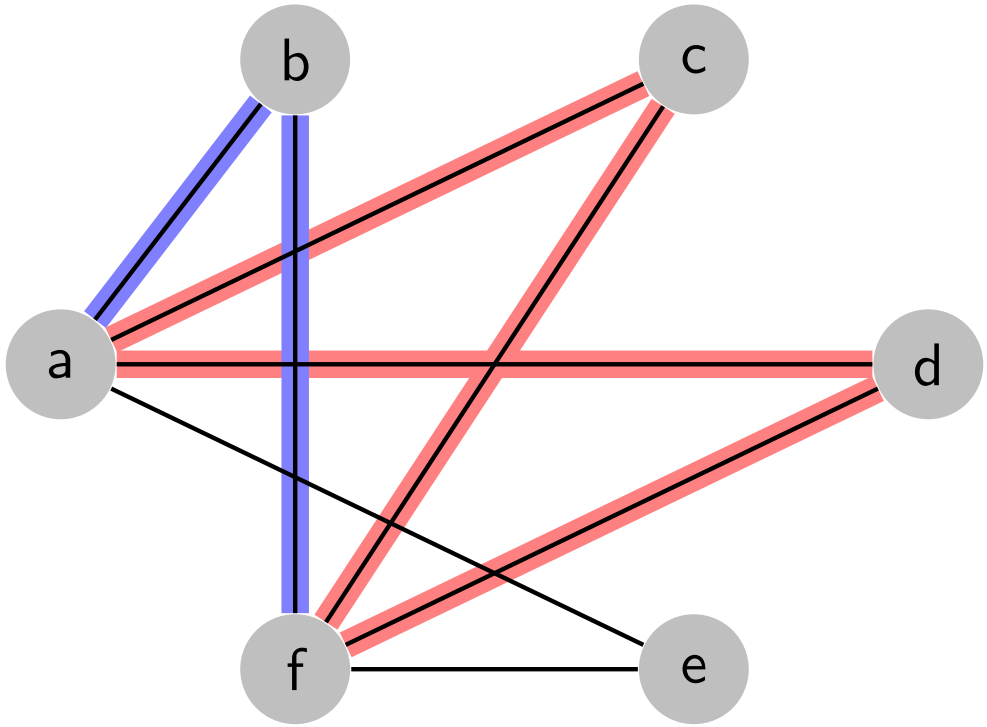
$$C = (c, a, d, f, c)$$

Problema do ciclo euleriano



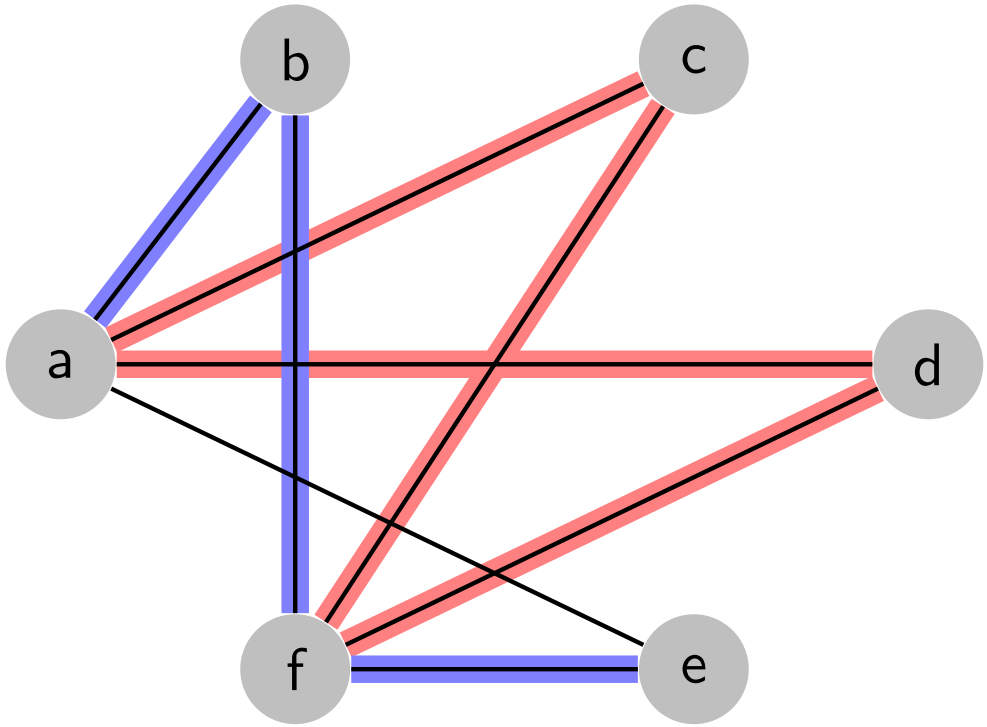
$$C = (c, a, b, d, f, c)$$

Problema do ciclo euleriano



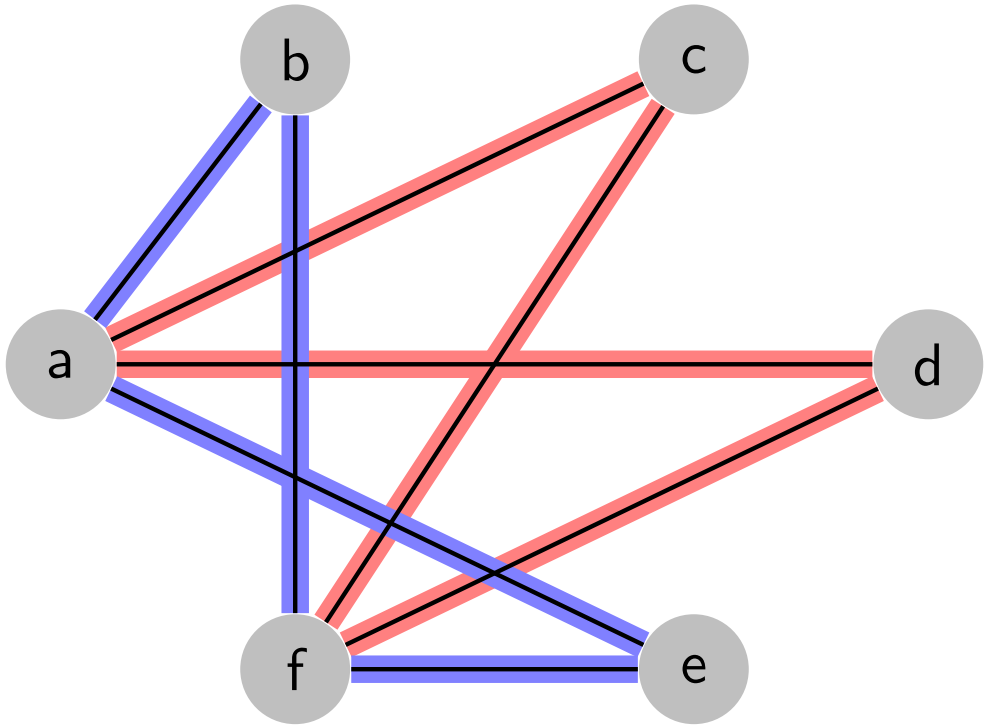
$$C = (c, a, b, f, d, f, c)$$

Problema do ciclo euleriano



$$C = (c, a, b, f, e, d, f, c)$$

Problema do ciclo euleriano



$$C = (c, a, b, f, e, a, d, f, c)$$

Problema do emparelhamento perfeito

Um emparelhamento em um grafo G é um conjunto de arestas sem extremos em comum, ou seja, cada vértice pertence a no máximo uma das arestas do emparelhamento.

Um emparelhamento M é perfeito se todo vértice de G pertence a alguma aresta de M.

O algoritmo de Edmonds, que denotaremos por EDMONDS, encontra um emparelhamento perfeito de custo mínimo em tempo $O(n^3)$, onde n é o número de vértices do grafo.

Algoritmos de aproximação para o TSPM

Agora que já vimos como resolver alguns problemas, vamos definir dois algoritmos de aproximação para o TSPM.

A estratégia utilizada pelos dois algoritmos tem quatro passos:

1. construir uma árvore geradora T de G;
2. acrescentar novas arestas a T para obter um novo grafo T' cujos vértices têm grau par;
3. obter um ciclo euleriano P em T';
4. obter um circuito hamiltoniano em G a partir de P.

A diferença entre os dois algoritmos está apenas na política adotada para acrescentar novas arestas à árvore T.

Algoritmos de aproximação para o TSPM

Uma árvore geradora de custo mínimo, calculada no passo 1 da estratégia, dá uma boa delimitação inferior para o valor ótimo do problema $\mathrm{TSPM}(G, c)$: se removemos uma aresta de um circuito hamiltoniano temos uma árvore geradora de custo não superior ao do circuito.

Portanto,

$$opt(G, c) \geq c(T).$$

Algoritmos de aproximação para o TSPM

O passo 2 da estratégia pode ser formalizado da seguinte maneira: para qualquer conjunto F de pares não-ordenados de vértices de T, seja $T + F$ o multigrafo $(V_T, E_T \dot{\cup} F)$, onde $E_T \dot{\cup} F$ denota o multiconjunto que tem duas cópias de cada elemento de $E_T \cap F$.

Como o grafo G, do qual T é uma árvore geradora, é completo, cada aresta do multigrafo $T + F$ tem um custo bem definido.

Após a execução do passo 2, temos a garantia que T' tem algum ciclo euleriano. O passo 3 encontra um destes ciclos.

Algoritmos de aproximação para o TSPM

O passo 4 da estratégia transforma um ciclo gerador, ou seja, um ciclo que contém todos os vértices do multigrafo, em um circuito hamiltoniano.

Para isso, basta extrair uma subsequência maximal sem vértices repetidos da sequência $(v_0, v_1, ..., v_m)$ de vértices do ciclo gerador.

Algoritmos de aproximação para o TSPM

Algoritmo ATALHO(P):

1 Seja $P = (v_0, v_1, ..., v_m, v_0)$;

2 $w_0 \leftarrow v_0$

3 $n \leftarrow 0$

4 para i de 1 a m, faça:

5 se $v_i \notin \{w_0, ..., w_n\}$

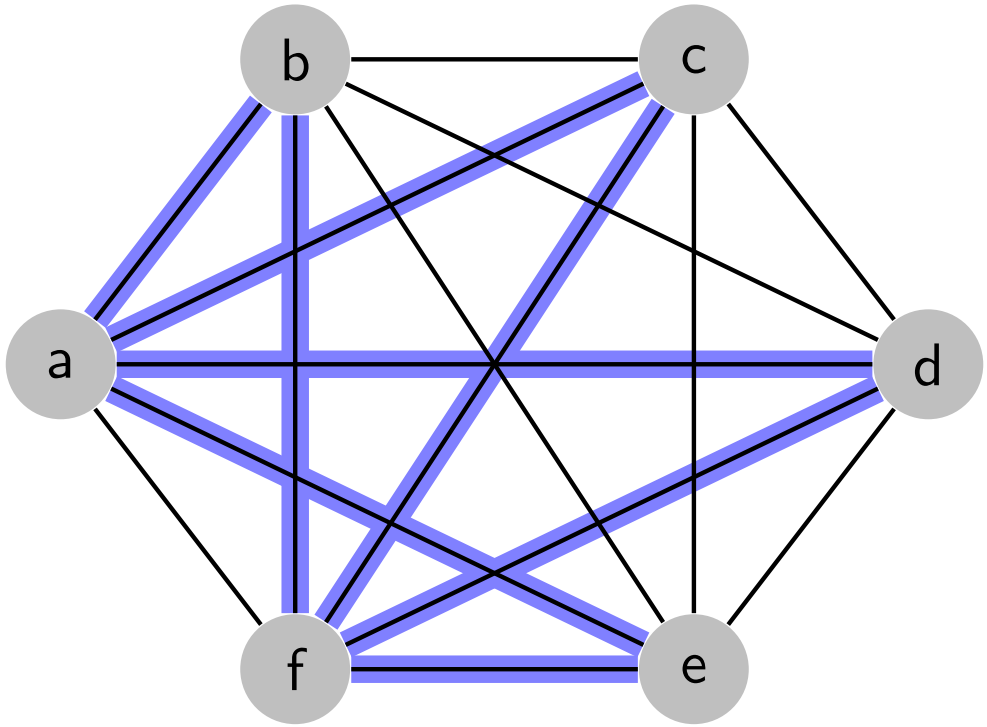
6 então $n \leftarrow n + 1$;

7 $w_n \leftarrow v_i$;

8 faça $C \leftarrow (w_0, w_1, ..., w_n, w_0)$;

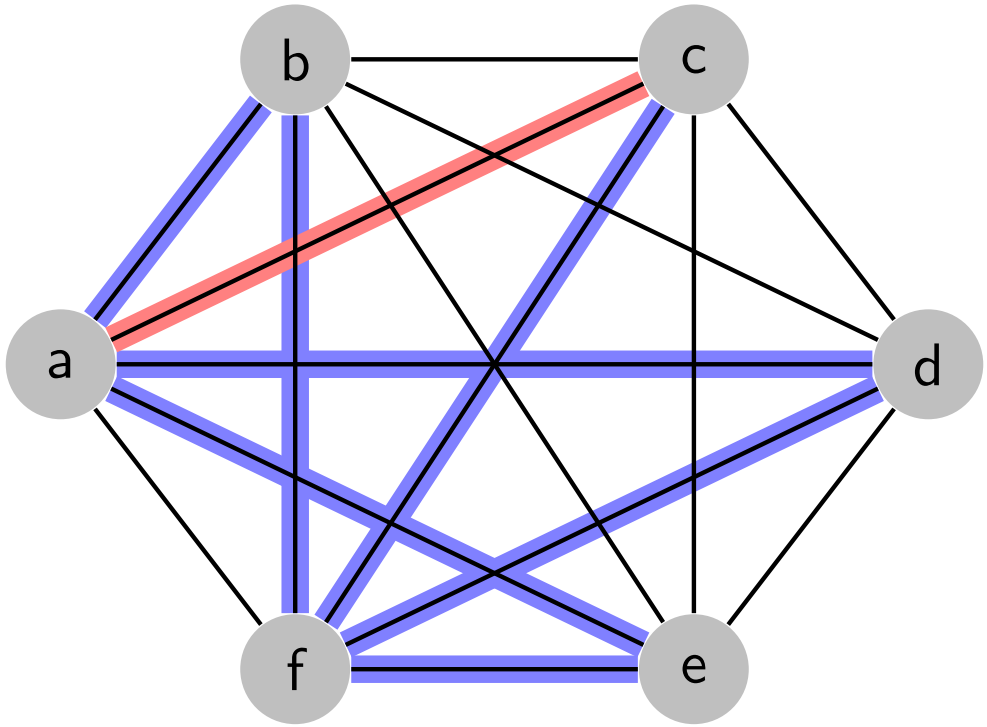
9 devolva C.

Algoritmos de aproximação para o TSPM



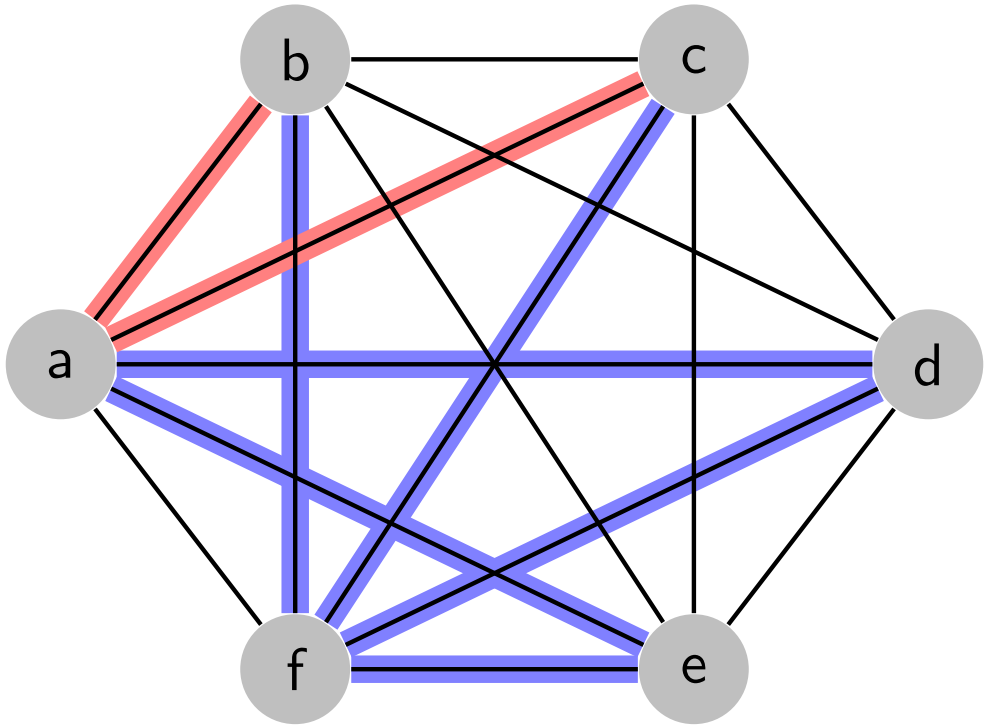
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c$$

Algoritmos de aproximação para o TSPM



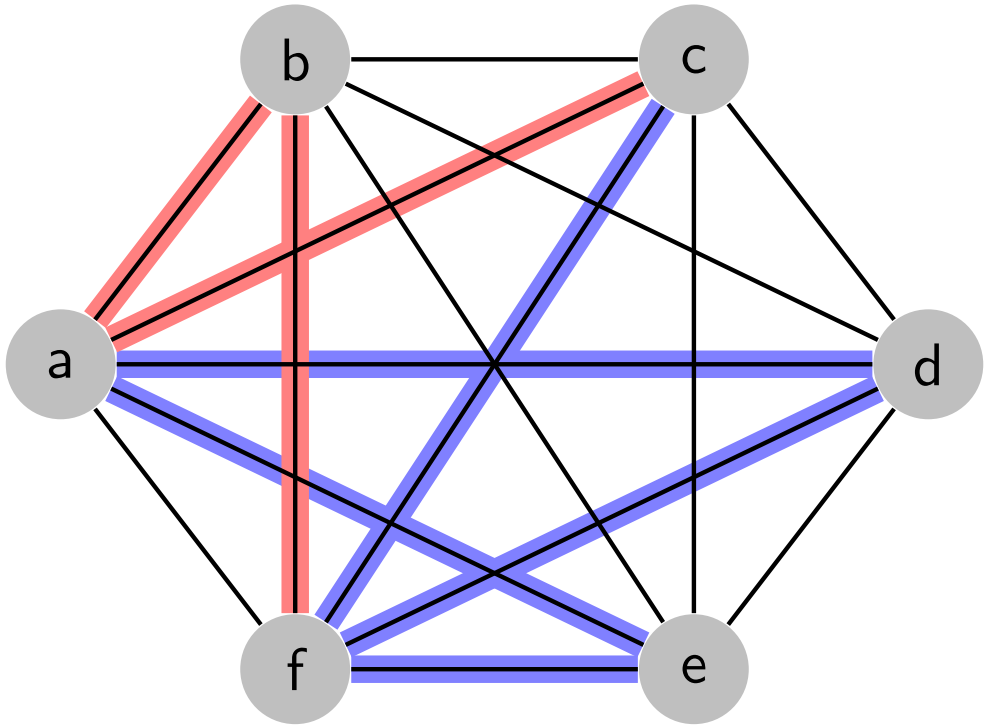
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a$$

Algoritmos de aproximação para o TSPM



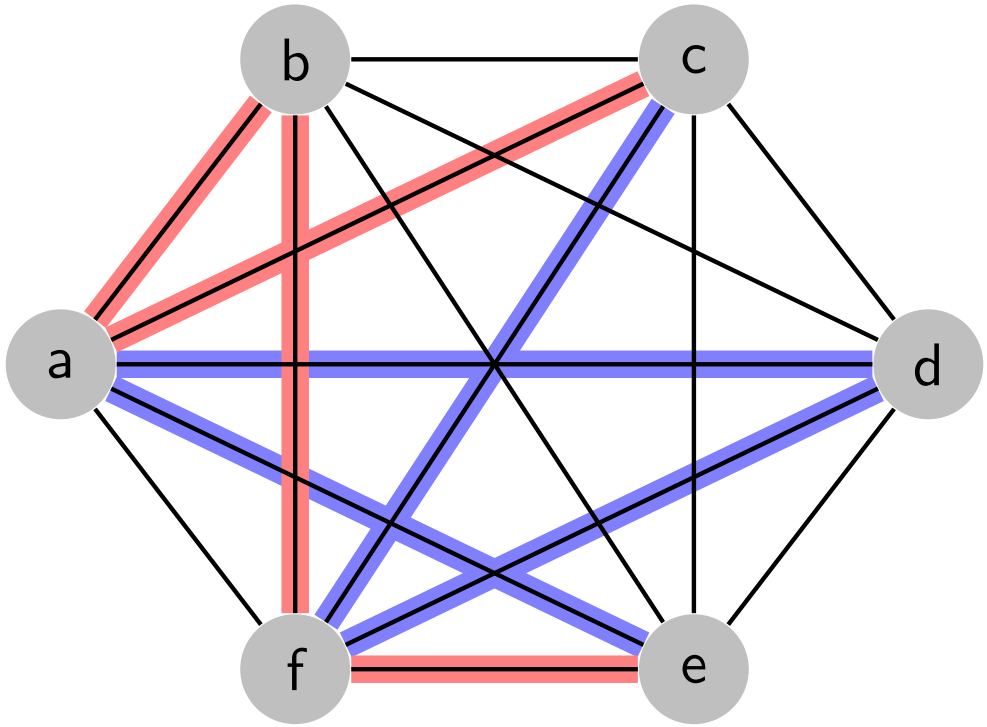
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b$$

Algoritmos de aproximação para o TSPM



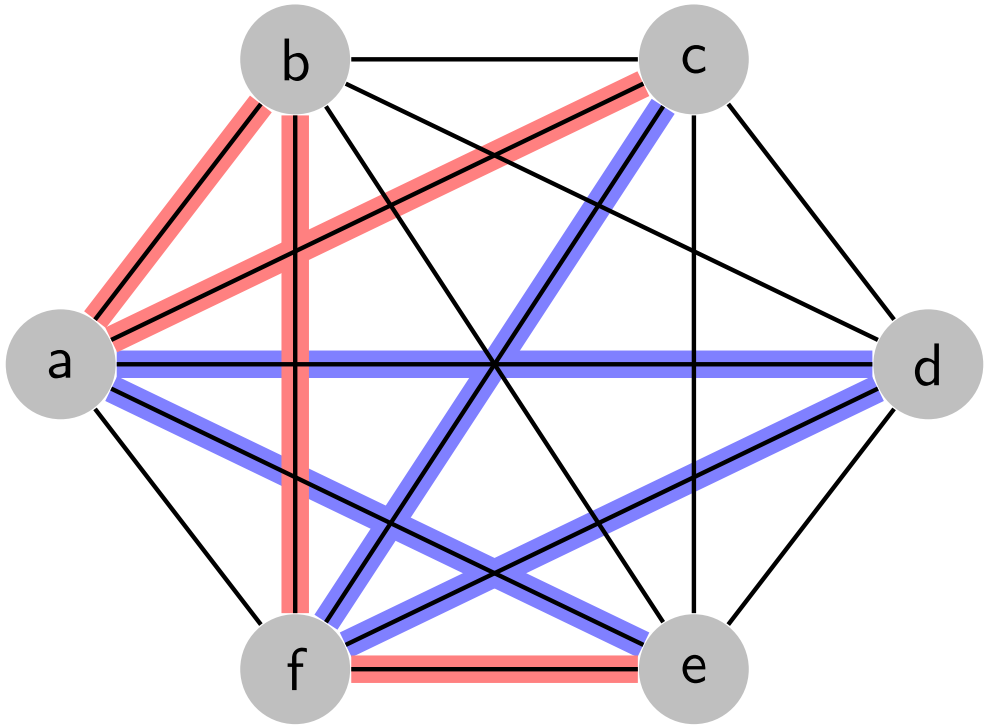
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f$$

Algoritmos de aproximação para o TSPM



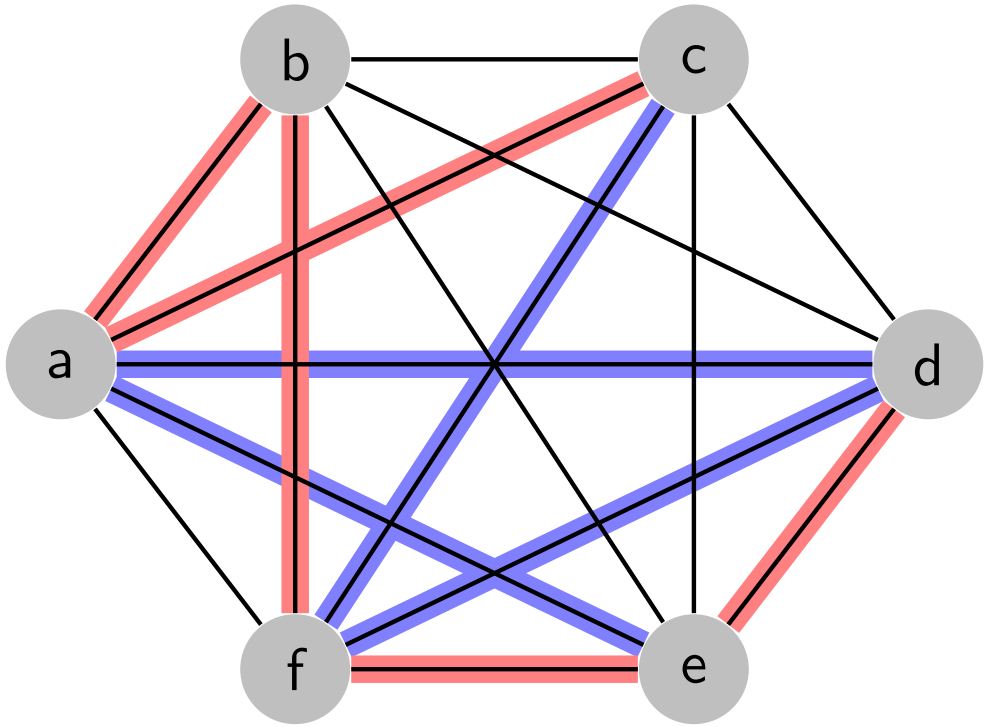
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f, e$$

Algoritmos de aproximação para o TSPM



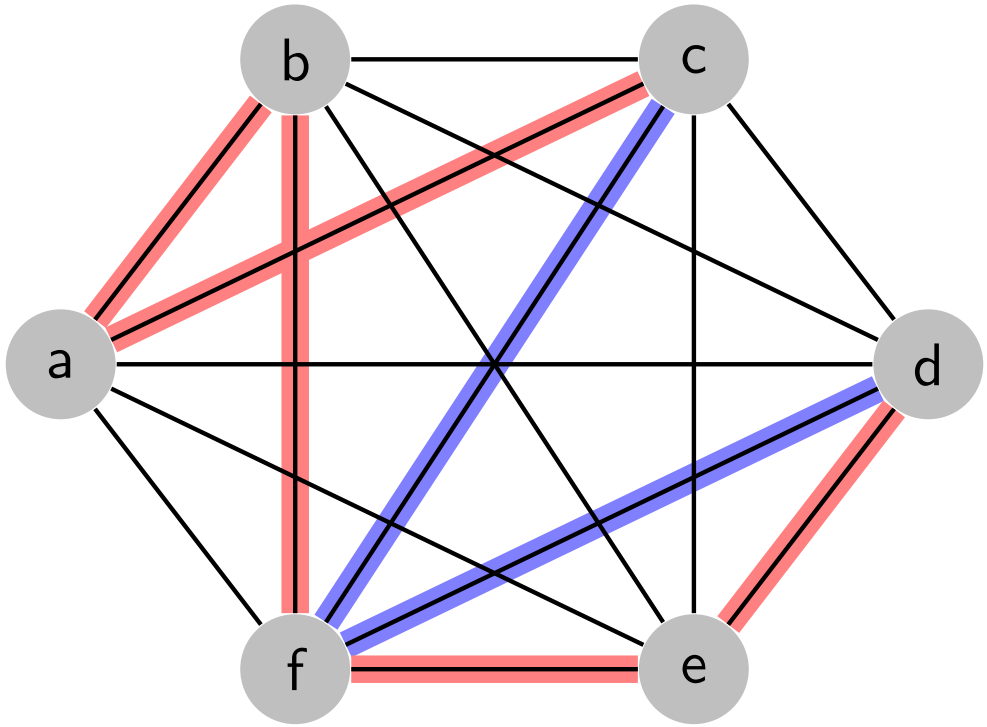
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f, e$$

Algoritmos de aproximação para o TSPM



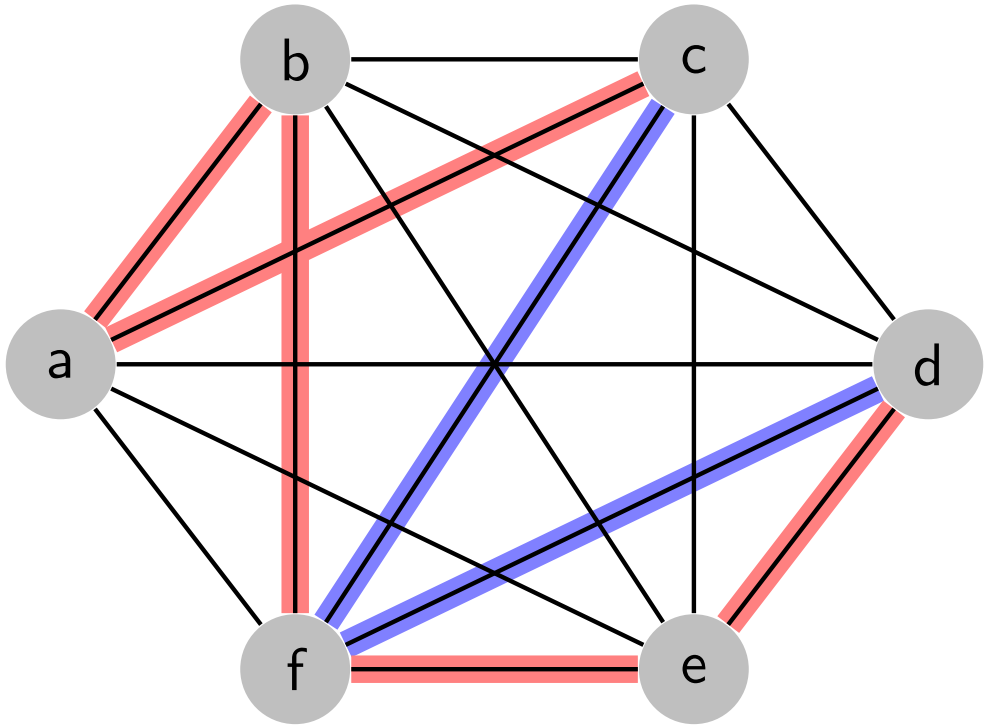
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f, e, d$$

Algoritmos de aproximação para o TSPM



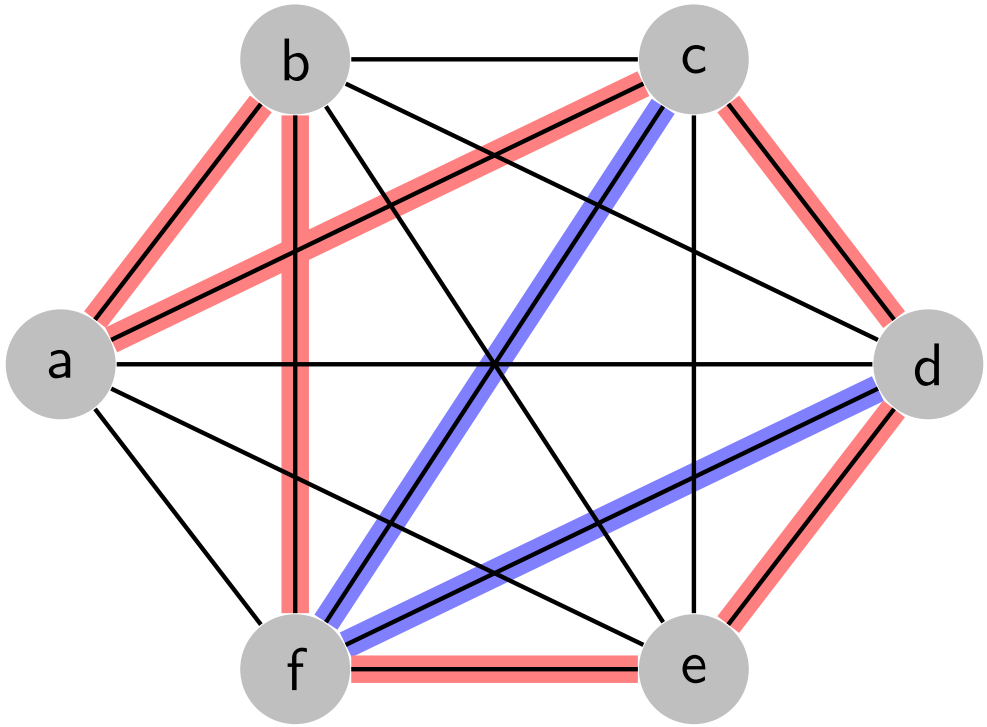
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f, e, d$$

Algoritmos de aproximação para o TSPM



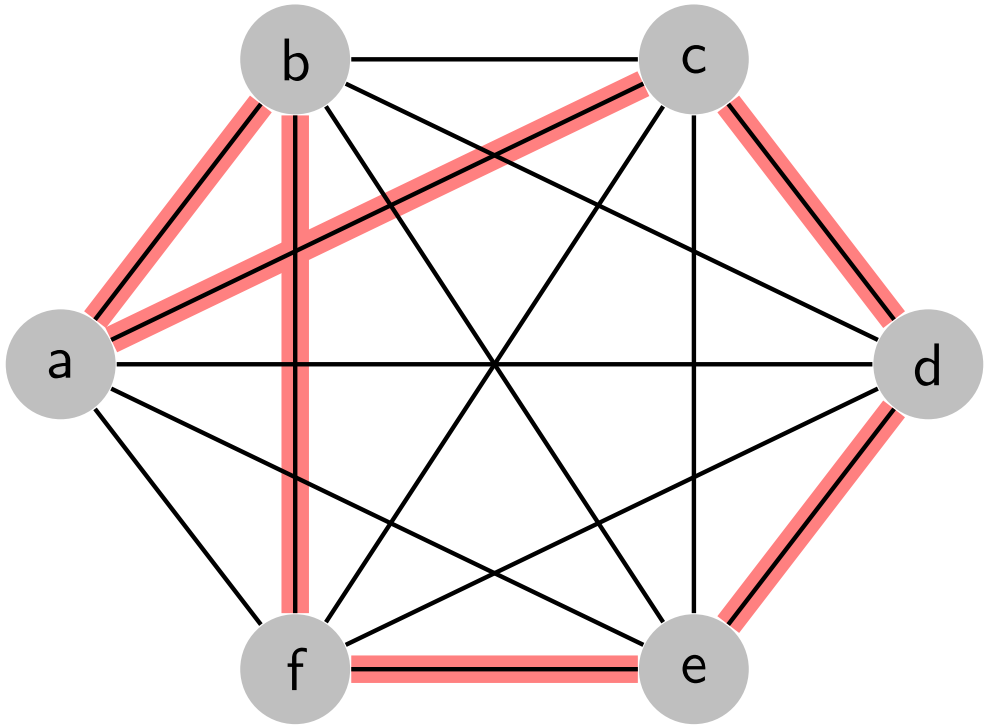
$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f, e, d$$

Algoritmos de aproximação para o TSPM



$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f, e, d, c)$$

Algoritmos de aproximação para o TSPM



$$P = (c, a, b, f, e, a, d, f, c) \Rightarrow C = (c, a, b, f, e, d, c)$$

Algoritmos de aproximação para o TSPM

Como o grafo G é completo, a sequência $C = (w_0, w_1, ..., w_n, w_0)$ define um circuito.

O circuito C contém todos os vértices do grafo, pois o ciclo P contém todos os vértices.

Cada par (w_j, w_{j+1}) de vértices consecutivos em C é ligado por um segmento $(v_i, v_{i+1}, ..., v_{i+p})$ em P. Graças à desigualdade triangular, o custo da aresta $w_j w_{j+1}$ não é maior que o custo do segmento.

Portanto, o custo do circuito resultante C não é maior que o do ciclo dado P. Ou seja,

$$c(C) \leq c(P).$$

O tempo gasto por ATALHO é proporcional ao número de arestas do ciclo P, ou seja, ao número de arestas do grafo.

Algoritmo de Rosenkrantz, Stearns e Lewis

No algoritmo descrito a seguir, apresentado em um artigo de Rosenkrantz, Stearns e Lewis, o multigrafo T' (passo 2 da estratégia) é obtido por meio da duplicação de cada uma das arestas da árvore geradora T.

Algoritmo $\mathrm{TSPM\text{-}RSL}(G, c)$:

1 $T \leftarrow \mathrm{MST}(G, c)$;

2 $T' \leftarrow T + E_T$;

3 $P \leftarrow \textsc{Euler}(T')$;

4 $C \leftarrow \textsc{Atalho}(P)$;

5 devolva C.

Algoritmo de Rosenkrantz, Stearns e Lewis

Evidentemente, todo vértice de T' tem grau par e, portanto, T' tem um ciclo euleriano.

O algoritmo EULER determina um tal ciclo.

Como o conjunto de vértices de T' é V_G, o ciclo euleriano P é gerador.

O circuito C devolvido por ATALHO na linha 4 é, então, um circuito hamiltoniano de G.

Algoritmo de Rosenkrantz, Stearns e Lewis

Teorema 1: O algoritmo TSPM-RSL é uma 2-aproximação polinomial para o TSPM.

Demonstração: Como P é um ciclo euleriano em $T + E_T$, temos que $c(P) = 2c(T)$.

Como $opt(G, c) \geq c(T)$ e $c(C) \leq c(P)$,

$$c(C) \leq c(P) = 2c(T) \leq 2opt(G, c).$$

A linha 1 do algoritmo consome tempo polinomial. As demais linhas consomem tempo $O(|V_G|)$, pois o número de arestas de T' é menor que $2|V_G|$. Ou seja, o algoritmo TSPM-RSL é polinomial.

Algoritmo de Christofides

O algoritmo de Christofides acrescenta à árvore geradora um emparelhamento perfeito no subgrafo de G induzido pelos vértices que têm grau ímpar em T.

Algoritmo TSPM-CHRISTOFIDES(G, c):

1 $T \leftarrow \text{MST}(G, c)$;

2 Seja I o conjunto dos vértices de grau ímpar de T;

3 $M \leftarrow$ EDMONDS$(G[I], c)$;

4 $T' \leftarrow T + M$;

5 $P \leftarrow$ EULER(T');

6 $C \leftarrow$ ATALHO(P);

7 devolva C.

Algoritmo de Christofides

Como M é um emparelhamento perfeito em $G[I]$, todo vértice de $T + M$ tem grau par e, portanto, o multigrafo T' na linha 4 tem um ciclo euleriano.

O ciclo é gerador pois T é geradora.

Na linha 6 do algoritmo, C é um circuito hamiltoniano de G.

Algoritmo de Christofides

Teorema 2: O algoritmo TSPM-CHRISTOFIDES é uma $\frac{3}{2}$-aproximação polinomial para o TSPM.

Demonstração: Precisamos mostrar que C tem custo no máximo $\frac{3}{2}opt(G, c)$.

Temos que $c(C) \leq c(P)$. Além disso,

$$c(P) = c(T') = c(T) + c(M).$$

Como $opt(G, c) \geq c(T)$, temos que

$$c(C) \leq c(T) + c(M) \leq opt(G, c) + c(M).$$

Algoritmo de Christofides

Precisamos mostrar agora que

$$c(M) \leq \frac{1}{2} opt(G, c) \Rightarrow opt(G, c) \geq 2c(M).$$

Seja C^* uma solução ótima para o TSPM.

Note que, como I é o conjunto de vértices de grau ímpar de T, $|I|$ é par.

Sejam $u_1, u_2, ..., u_{2k}$ os vértices de I na ordem em que aparecem em C^*.

Algoritmo de Christofides

Como G é completo, a sequência $D := (u_1, u_2, ..., u_{2k}, u_1)$ é um circuito em $G[I]$.

Em outras palavras, D pode ser obtido de C^* pela substituição de cada segmento de C^* que liga u_i a u_{i+1} pela aresta $u_i u_{i+1}$ de G.

A desigualdade triangular garante que $c(D) \leq c(C^*)$.

Algoritmo de Christofides

Além disso, como D tem comprimento par, E_D é a união de dois emparelhamentos perfeitos em $G[I]$ mutuamente disjuntos, digamos M' e M''.

Como M é um emparelhamento perfeito de custo mínimo,

$$2c(M) \leq c(M') + c(M'').$$

Logo,

$$2c(M) \leq c(M') + c(M'') = c(D) \leq c(C^*) = opt(G, c),$$

como gostaríamos.

Algoritmo de Christofides

A linha 3 consome tempo $O(|V_G|^3)$, enquanto que as demais linhas consomem tempo polinomial no número de vértices e arestas de G.

Portanto, o algoritmo TSPM-CHRISTOFIDES é polinomial.

Proposto em 1976, TSPM-CHRISTOFIDES é ainda o melhor algoritmo de aproximação conhecido para o TSPM.

O algoritmo TSPM-RSL pode ser uma boa alternativa, já que ele consome menos tempo que o TSPM-CHRISTOFIDES e é bem mais simples, pois não envolve a determinação de um emparelhamento perfeito de custo mínimo.