Caixeiro Viajante é NP-Difícil

Caixeiro Viajante (Travelling Salesman Problem TSP)

- **Instância:** Grafo completo G com custos positivos nas arestas.
- **Objetivo:** Achar um ciclo de custo mínimo que passe por todos os vértices exatamente uma vez.
- **Problema relacionado (Ciclo Hamiltoniano):** Dado grafo simples (sem custos nas arestas), decidir se tem ciclo por todos os vértices exat. uma vez. Problema NP-Completo.

Teorema: TSP é NP-Difícil.

PROVA:

- Suponha que existe um algoritmo polinomial A que resolve o Caixeiro Viajante. Vamos usar A para decidir o problema do Ciclo Hamiltoniano em tempo polinomial.
- Seja G instância de Ciclo Hamilt: grafo simples (sem custos). Seja G^* grafo completo nos vértices de G tq custo aresta uv em G^* é 1 se uv é aresta de G, e $n+1$, c.c.
- Se o caixeiro tem percurso com custo n, então G possui Ciclo Hamiltoniano. Caso contrário, não tem.

Caixeiro Viajante não é aproximável poli.

Teorema (Sahni, Gonzalez, 1976):

TSP não tem algoritmo $\alpha(n)$-aproximativo polinomial para nenhuma função polinomial $\alpha(n)$, a menos que P=NP

PROVA:

- Suponha existe algoritmo poli $\alpha(n)$-aprox A para Caixeiro Viajante. Vamos usar A p/ decidir Ciclo Hamilt em tempo polinomial.
- Instância G de Ciclo Hamilt: grafo simples. Seja G^* grafo completo tq custo aresta uv em G^* é 1 se uv está em G, e $n \cdot \alpha(n)$, c.c.
- $A(G^*)$: custo solução de A p/ G^*. Logo $A(G^*) \leq opt \cdot \alpha(n)$.
- Se $A(G^*) > n \cdot \alpha(n)$, então $opt > n$ e então G não é hamiltoniano.
- Se $A(G^*) \leq n \cdot \alpha(n)$, então nenhuma aresta não-existente foi percorrida no ciclo do caixeiro e então G é hamiltoniano.
- Com isso, A pode ser usado para decidir Ciclo Hamilt. Logo P=NP.

Teorema 3.6 (Livro "*Approximation Algorithms*" de Vazirani).
Teorema 8.5 (Livro "*Introdução Sucinta a Algoritmos de Aproximação*").
Técnica do GAP para provar Inaproximabilidade.

Caixeiro Viajante Métrico (TSPM)

Instâncias satisfazem Desigualdade Triangular

- Grafo G com custos positivos nas arestas
- Custos satisfazem a desigualdade triangular: $c_{ij} \leq c_{ik} + c_{kj}$

Árvore Geradora Mínima (MST) T

- Seja *opt* o custo do percurso ótimo do Caixeiro.
- Removendo uma aresta do percurso ótimo, temos uma árvore geradora do grafo, com custo $\geq$ a da geradora mínima (MST) T.
- Logo: $opt \geq c(T)$

Ciclo Euleriano

- **Definição:** Ciclo que passa por todas as arestas do grafo.
- **Teorema de Euler:** Só existe se não há vértices de grau ímpar.
- Existem algoritmos $O(m)$ para obter ciclo euleriano, caso exista.
- Seja EULER um desses algoritmos.

Algoritmo TSPM-RSL (Rosenkrantz, Stearns, Lewis, 70's)

Algoritmo TSPM-RSL (G, c)

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

2 $T' \leftarrow T + E_T$

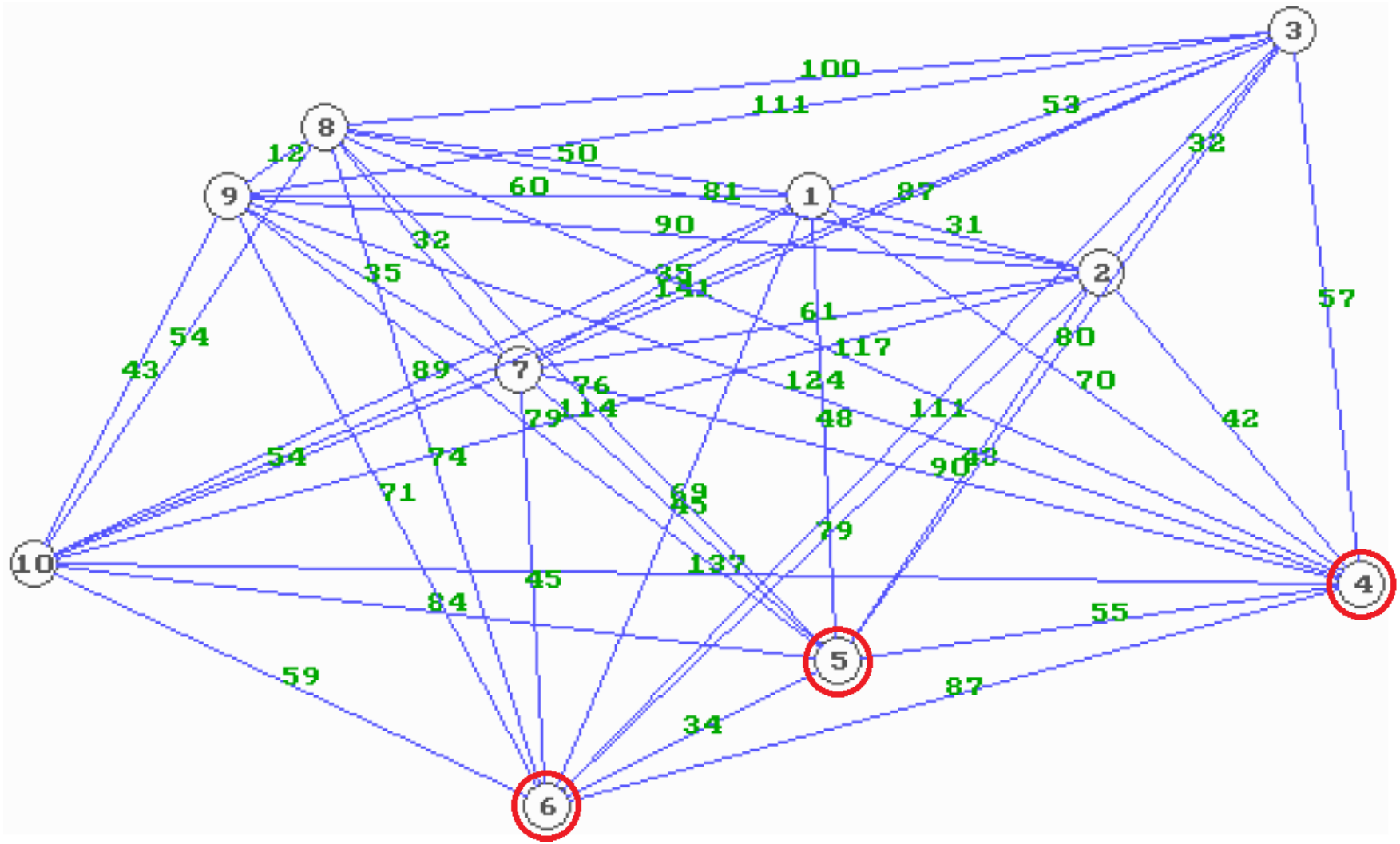
3 $P \leftarrow \text{EULER}(T')$

4 $C \leftarrow \text{ATALHO}(P)$

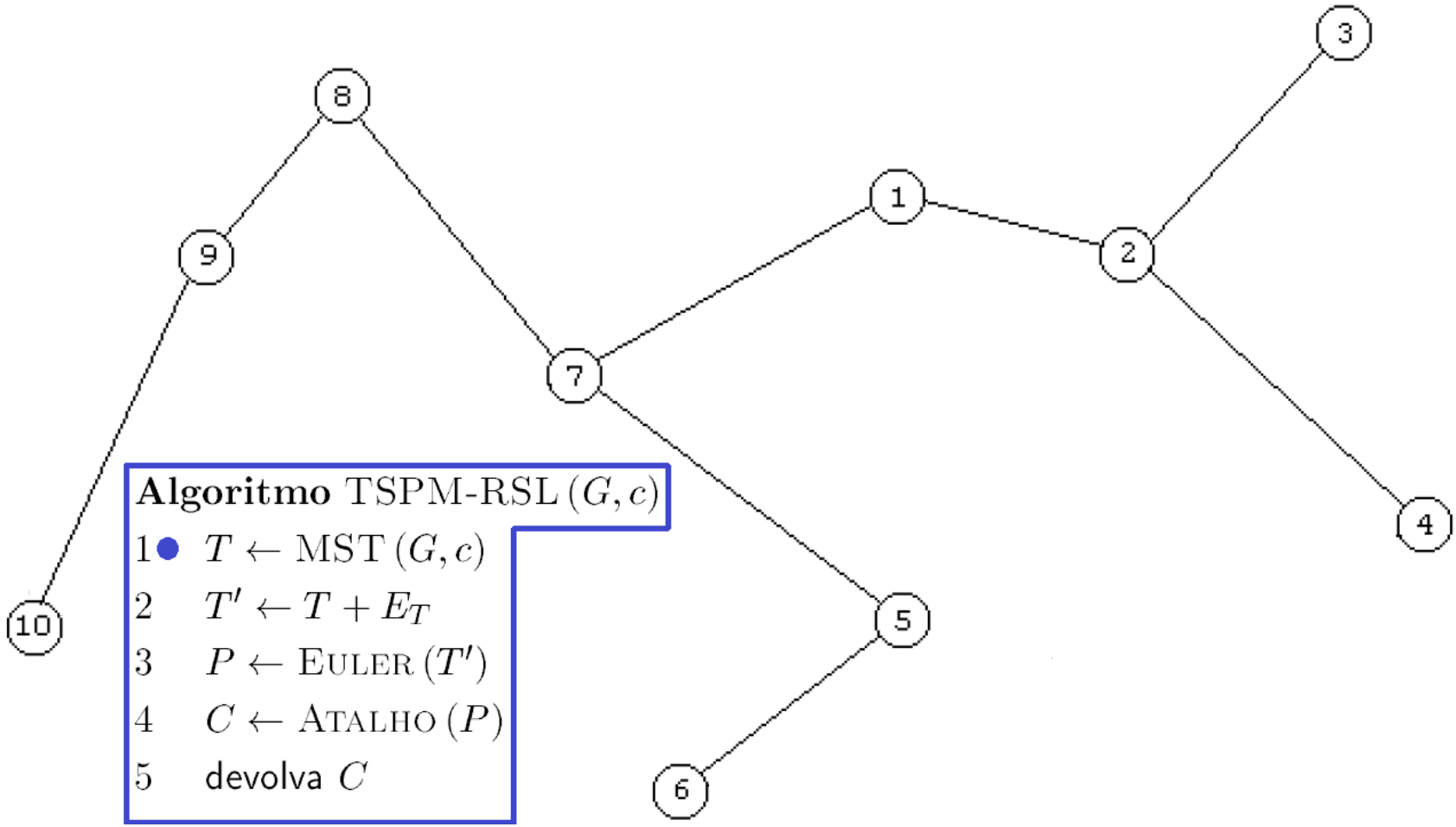
5 devolva C

Algoritmo ATALHO remove vértices repetidos de um ciclo.
Pela desigualdade triangular, vale a pena ir direto.
De um Ciclo Euleriano, obtemos um Ciclo Hamiltoniano.

Algoritmo TSPM-RSL (Rosenkrantz, Stearns, Lewis, 70's)



Algoritmo TSPM-RSL (Rosenkrantz, Stearns, Lewis, 70's)



Algoritmo TSPM-RSL (G, c)

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

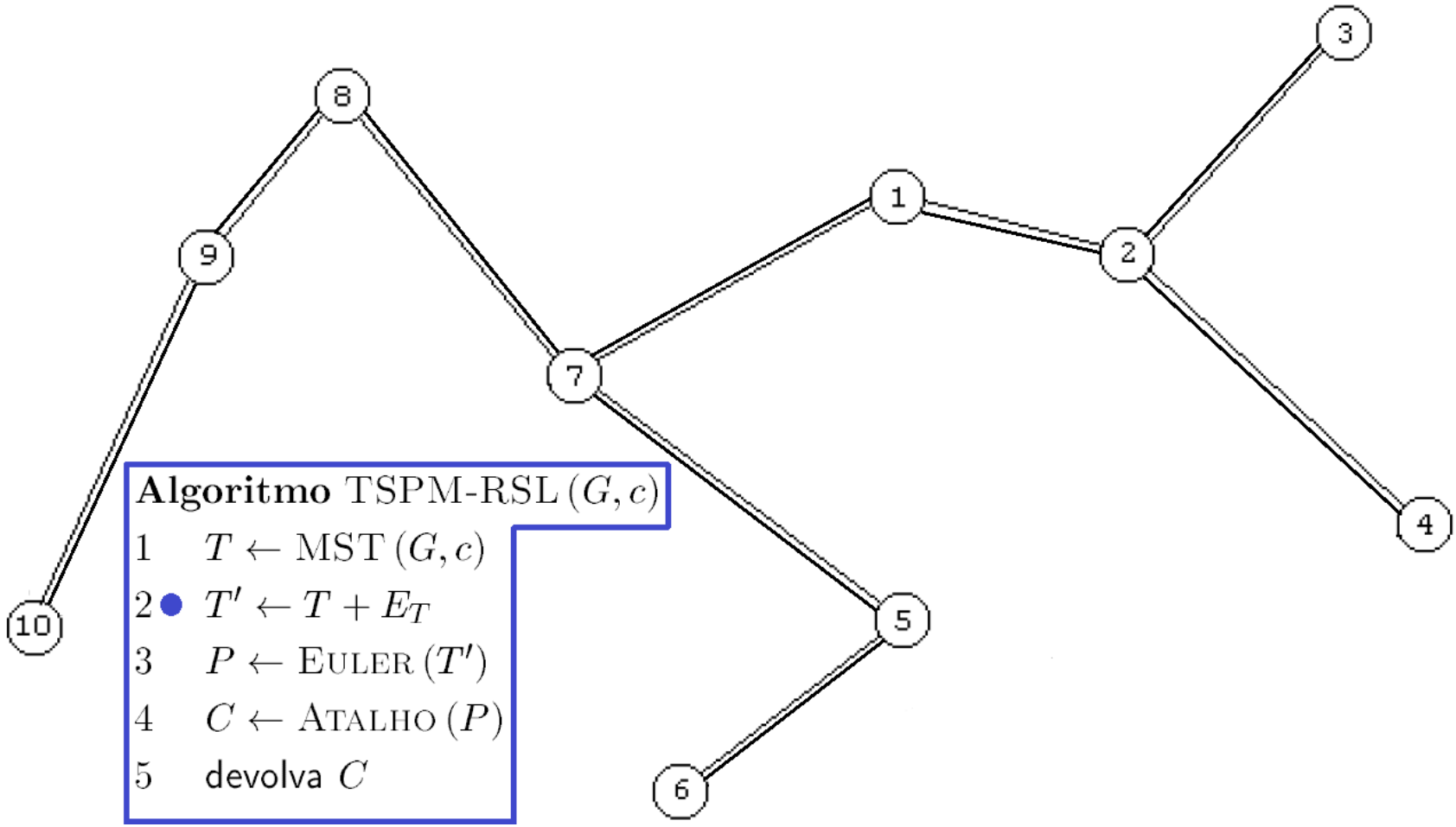
2 $T' \leftarrow T + E_T$

3 $P \leftarrow \text{EULER}\,(T')$

4 $C \leftarrow \text{ATALHO}\,(P)$

5 devolva C

Algoritmo TSPM-RSL (Rosenkrantz, Stearns, Lewis, 70's)



Algoritmo TSPM-RSL (G, c)

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

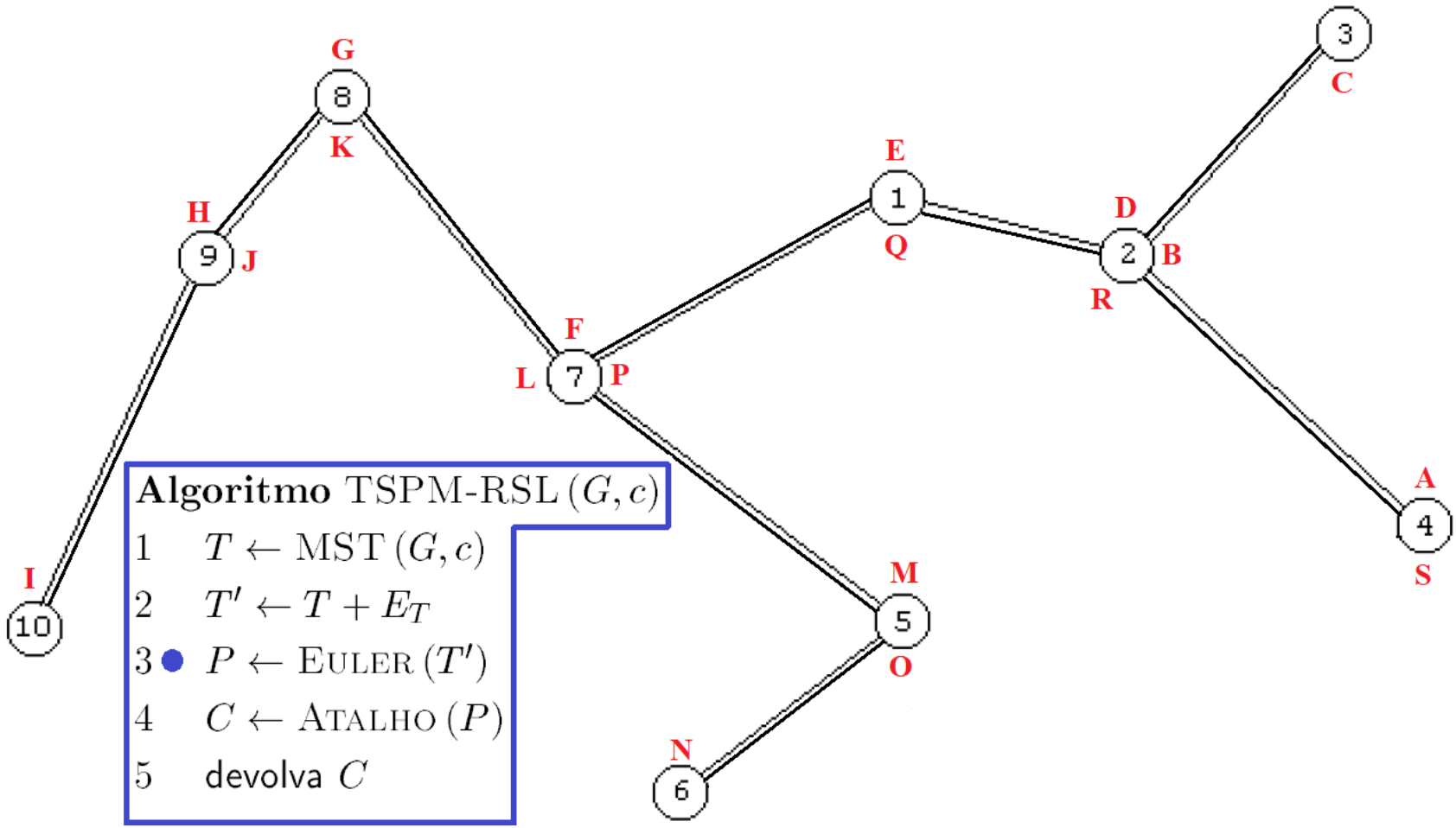
2 • $T' \leftarrow T + E_T$

3 $P \leftarrow \text{EULER}\,(T')$

4 $C \leftarrow \text{ATALHO}\,(P)$

5 devolva C

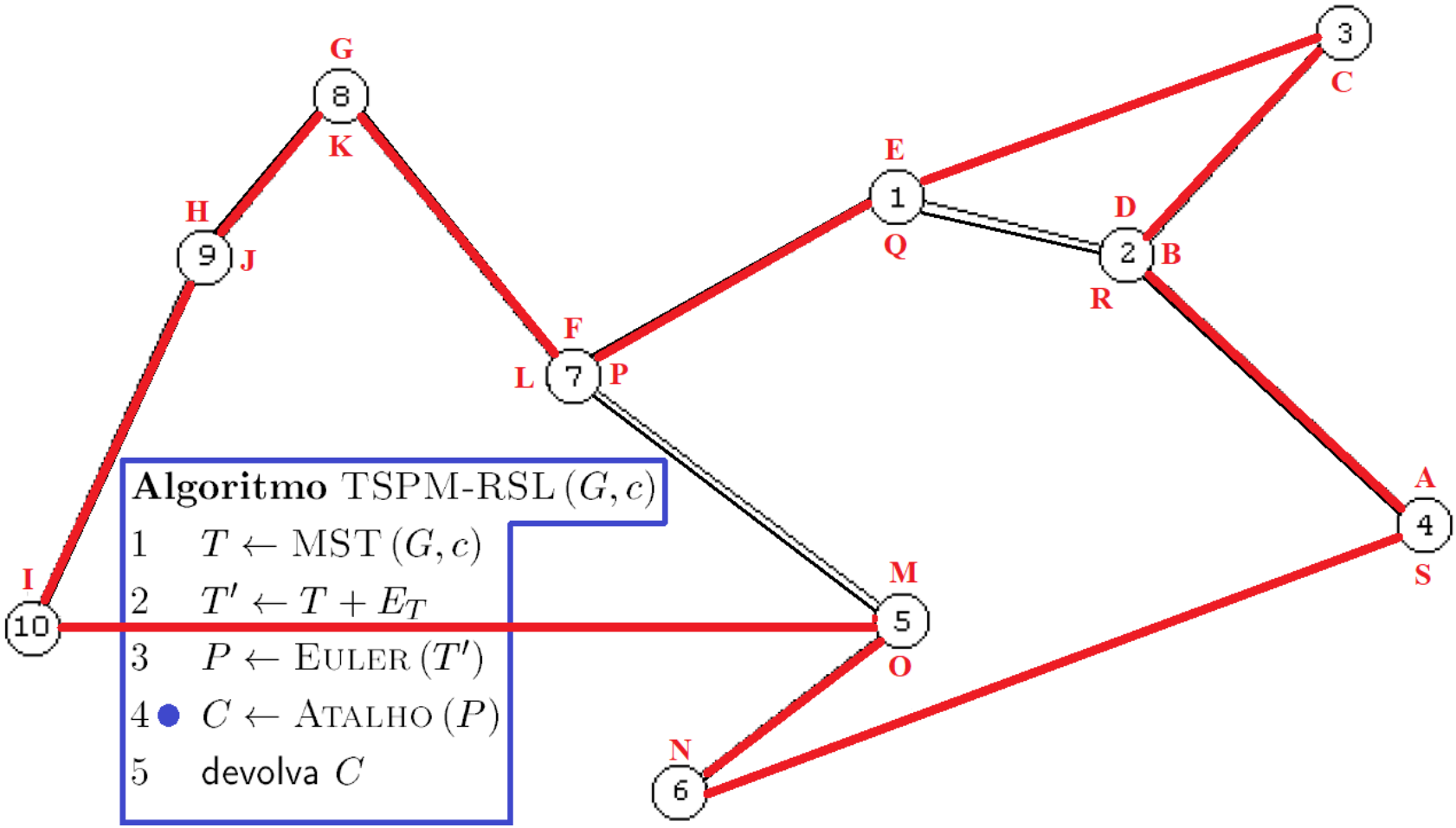
Algoritmo TSPM-RSL (Rosenkrantz, Stearns, Lewis, 70's)



Algoritmo TSPM-RSL (G, c)

1 $T \leftarrow \text{MST}(G, c)$
2 $T' \leftarrow T + E_T$
3 $P \leftarrow \text{EULER}(T')$
4 $C \leftarrow \text{ATALHO}(P)$
5 devolva C

Algoritmo TSPM-RSL (Rosenkrantz, Stearns, Lewis, 70's)



Algoritmo TSPM-RSL (G, c)

1 $T \leftarrow \text{MST}(G, c)$
2 $T' \leftarrow T + E_T$
3 $P \leftarrow \text{EULER}(T')$
4 $C \leftarrow \text{ATALHO}(P)$
5 devolva C

Algoritmo TSPM-RSL é 2-aproximativo

Algoritmo TSPM-RSL (G, c)

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

2 $T' \leftarrow T + E_T$

3 $P \leftarrow \text{EULER}(T')$

4 $C \leftarrow \text{ATALHO}(P)$

5 devolva C

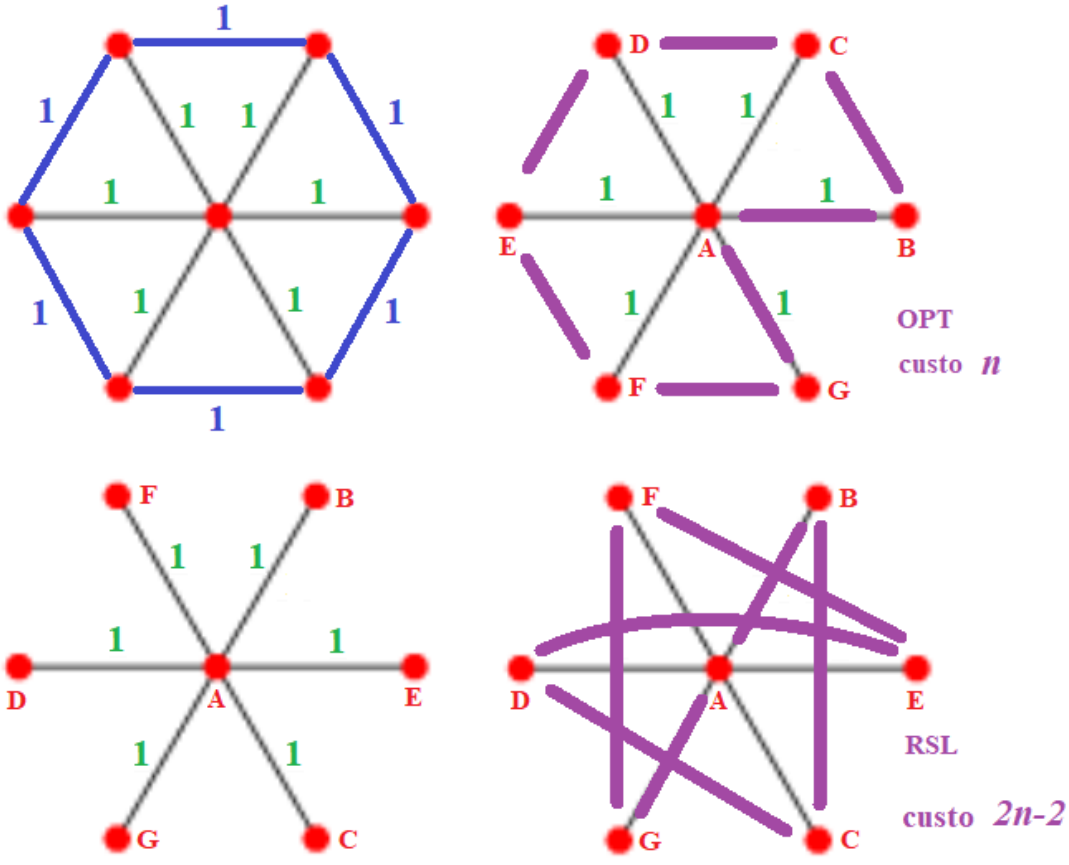
Teorema: TSPM-RSL é 2-aproximativo

PROVA:

- P é Ciclo Euleriano de $T' = T + E_T$. Logo:
- $c(C) \leq c(P) = 2 \cdot c(T) \leq 2 \cdot opt(G, c)$,
- pois **(a)** $c(C) \leq c(P)$ pela desigualdade triangular,
- **(b)** percurso *opt* menos 1 aresta é árvore geradora: $opt \geq c(T)$

Algoritmo TSPM-RSL é 2-aproximativo (e não menos !)

Tight Example



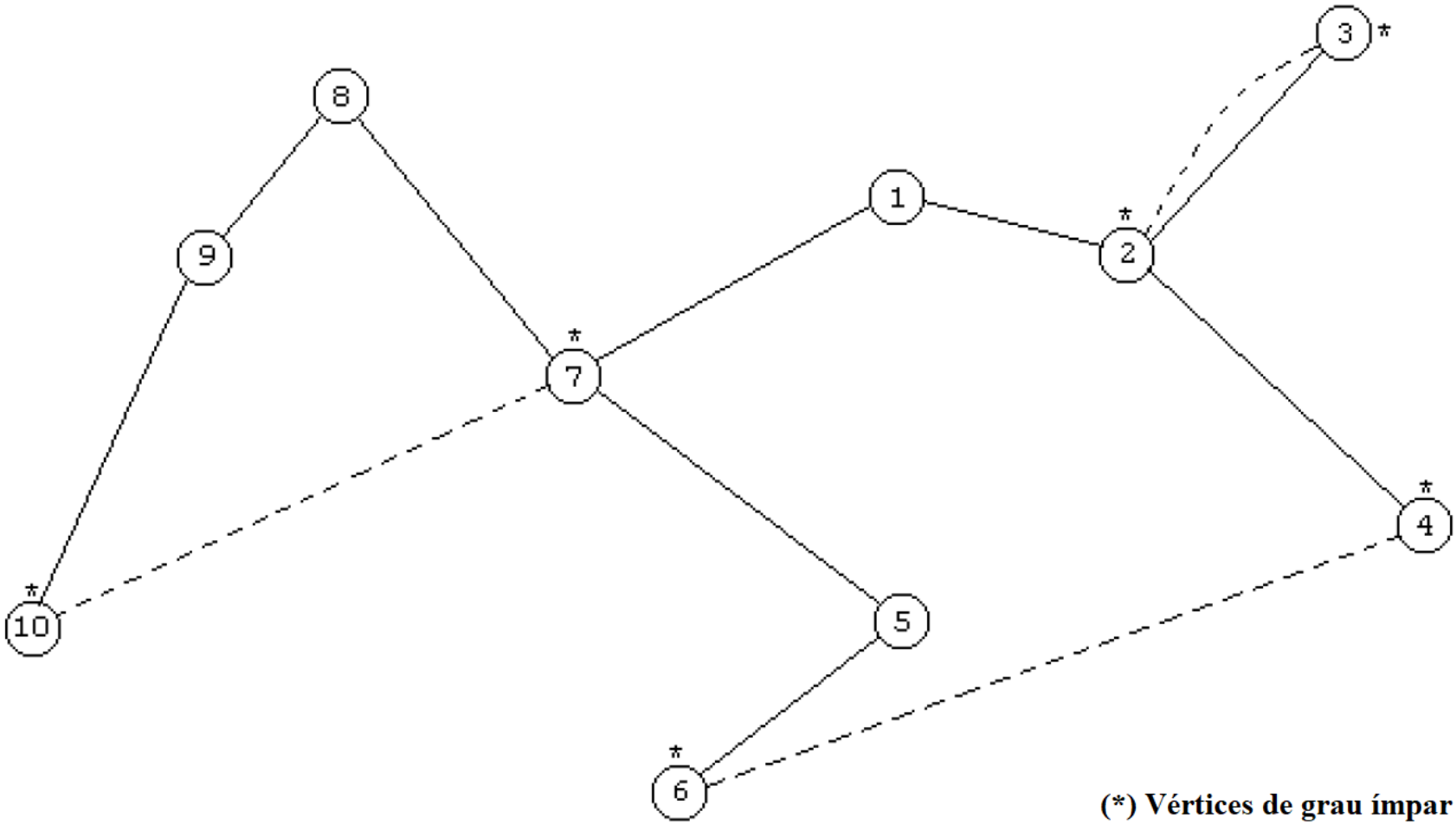
Algoritmo TSPM-Christofides (1976)

Algoritmo TSPM-CHRISTOFIDES (G, c)

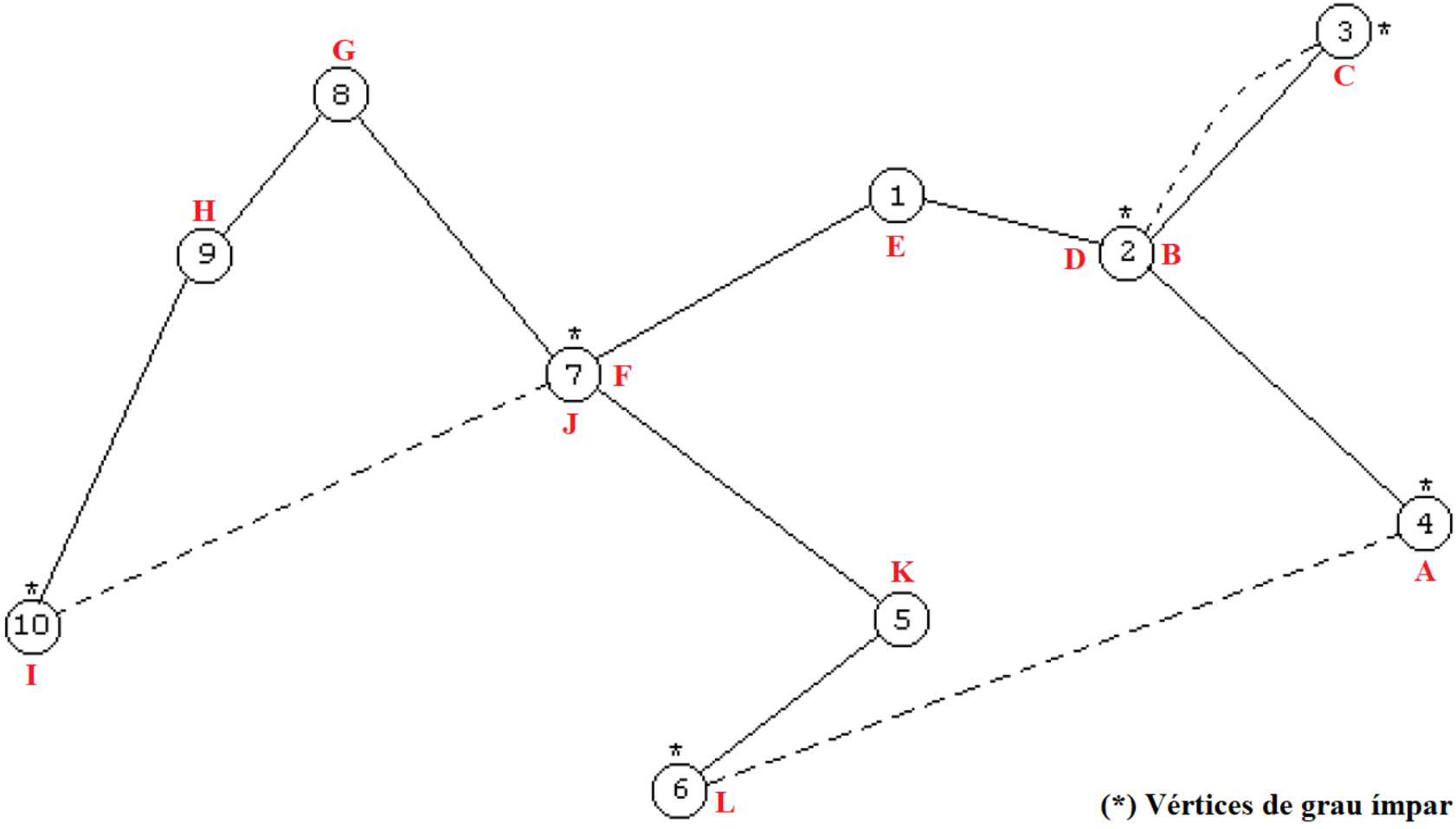
1 $T \leftarrow \text{MST}\,(G, c)$
2 seja I o conjunto dos vértices de grau ímpar de T
3 $M \leftarrow \text{EDMONDS}\,(G[I], c)$
4 $T' \leftarrow T + M$
5 $P \leftarrow \text{EULER}\,(T')$
6 $C \leftarrow \text{ATALHO}\,(P)$
7 devolva C

TSPM-Christofides é 1.5-aproximativo.
Melhor algoritmo p/ TSPM (Caixeiro Viajante Métrico) até 2020.
Foi melhorado em menos de um trilionésimo de porcento em 2020.
Conjectura: 1.333 é o melhor fator de aproximação.
ALMSS'92: Não existe PTAS para TSPM.

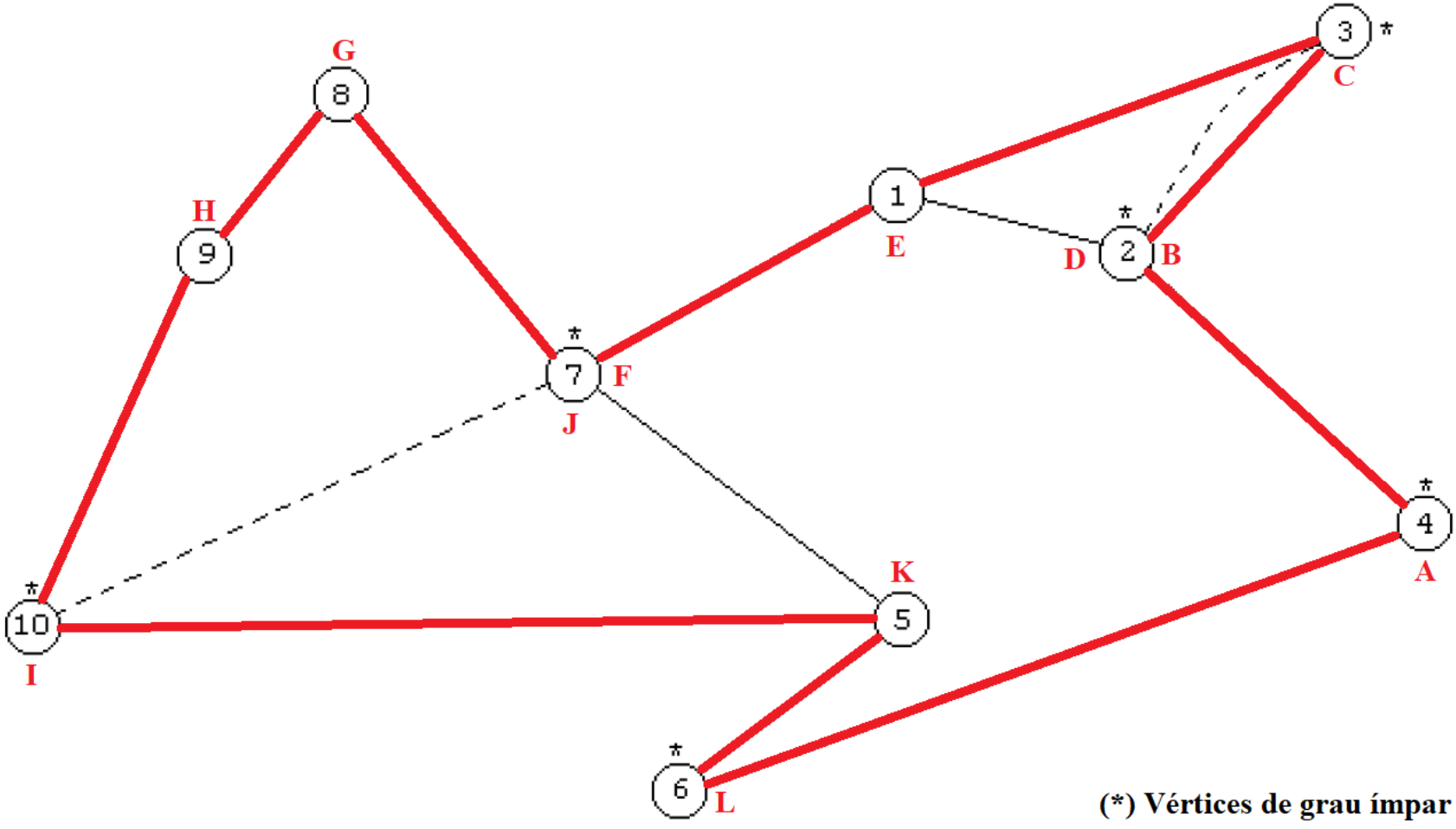
Algoritmo TSPM-Christofides (1976)



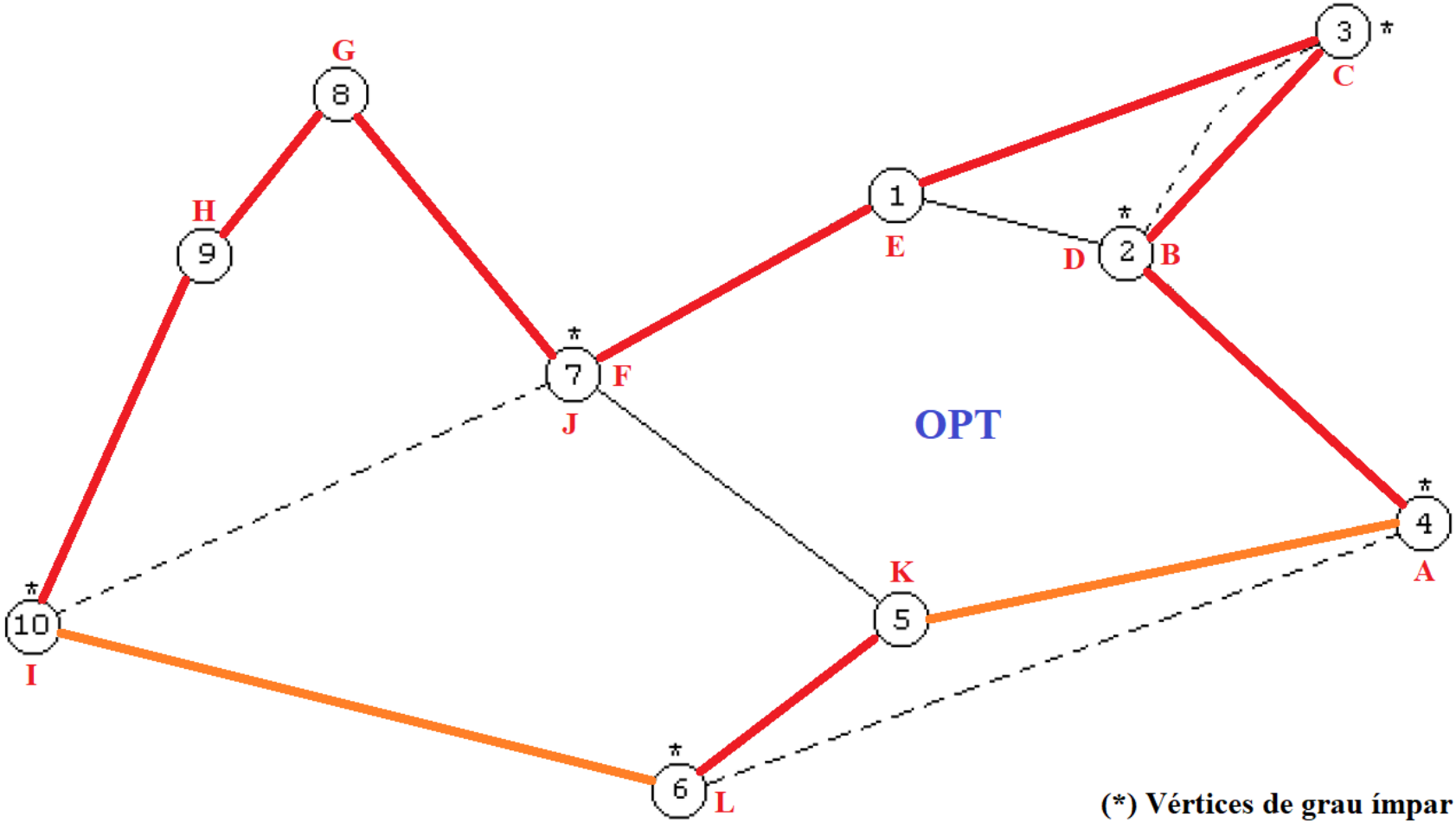
Algoritmo TSPM-Christofides (1976)



Algoritmo TSPM-Christofides (1976)



Algoritmo TSPM-Christofides (1976)



Algoritmo TSPM-Christofides é 1.5-aproximativo

Algoritmo TSPM-CHRISTOFIDES (G, c)

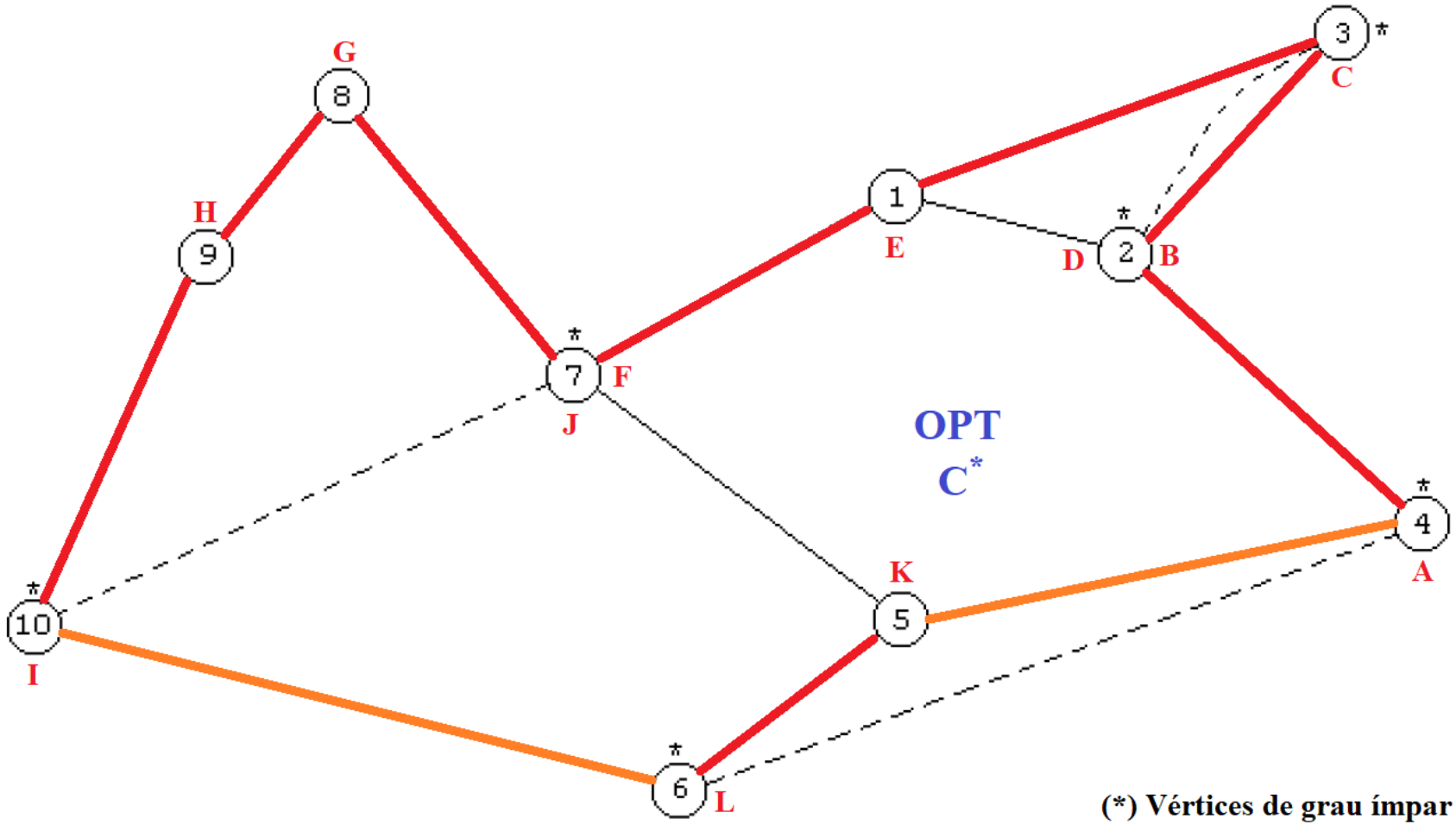
1 $T \leftarrow \text{MST}(G, c)$
2 seja I o conjunto dos vértices de grau ímpar de T
3 $M \leftarrow \text{EDMONDS}(G[I], c)$
4 $T' \leftarrow T + M$
5 $P \leftarrow \text{EULER}(T')$
6 $C \leftarrow \text{ATALHO}(P)$
7 devolva C

Teorema: TSPM-Christofides é 1.5-aproximativo

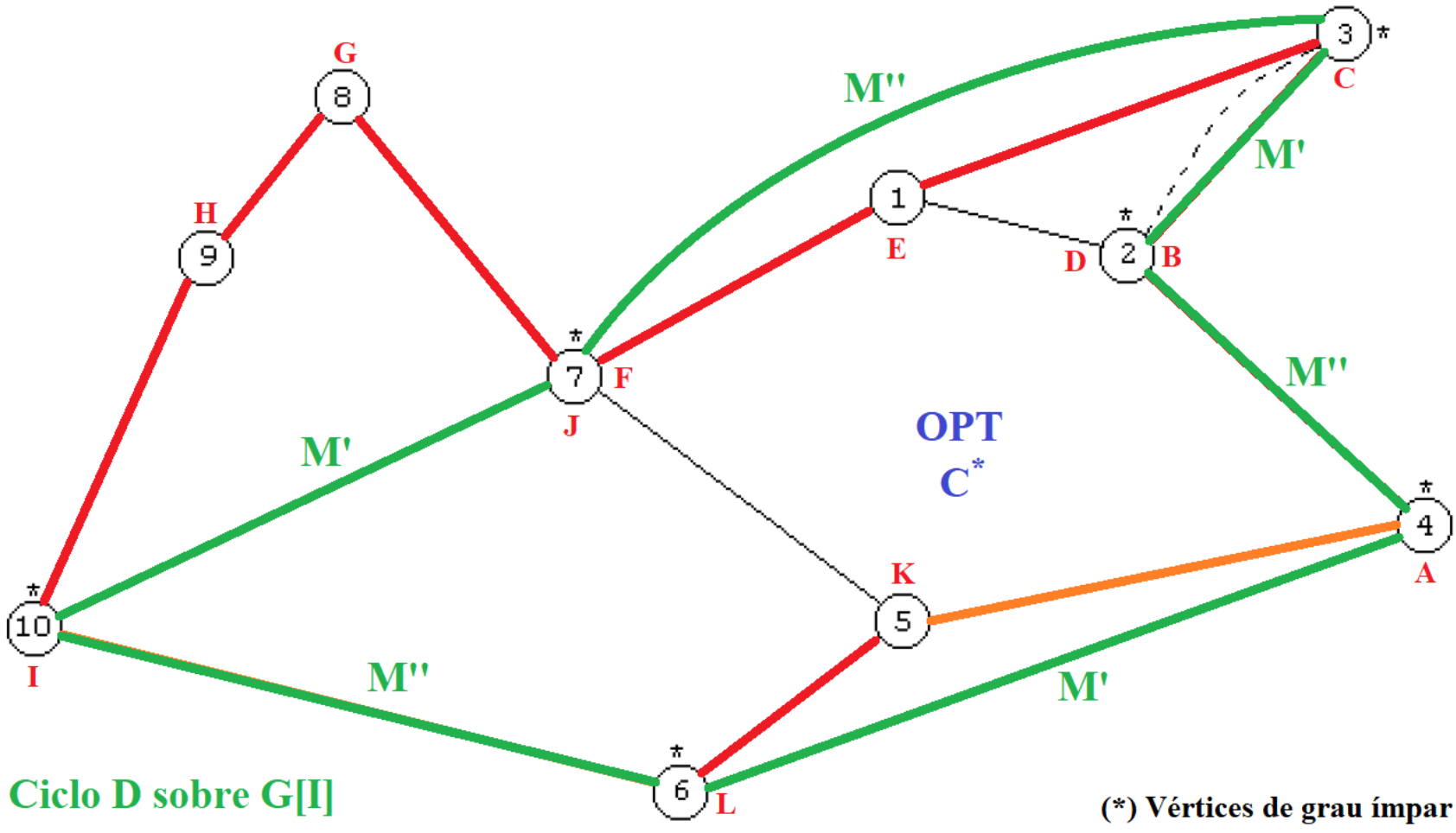
PROVA:

- P é Ciclo Euleriano de $T' = T + M$. Logo:
- $c(C) \leq c(P) = c(T) + c(M) \leq 1.5 \cdot opt(G, c)$,
- faltando provar que $c(M) \leq opt/2$. Para isso, seja C^* um ciclo opt.
- Seja D o ciclo induzido por C^* sobre I (vértices de grau ímpar de T).
- $|I|$ é par $\Rightarrow$ D é união de 2 *matchings* M' e M'' de $G[I]$.
- Logo $2 \cdot c(M) \leq c(M') + c(M'') = c(D) \leq c(C^*) = opt(G, c)$

Algoritmo TSPM-Christofides é 1.5-aproximativo

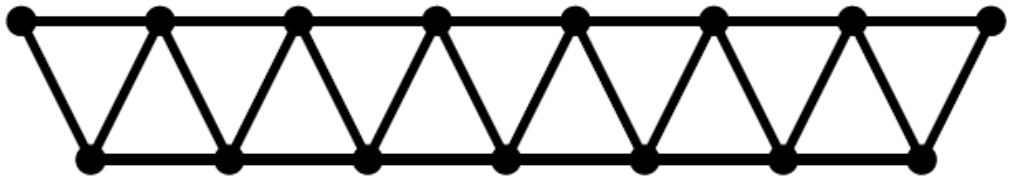


Algoritmo TSPM-Christofides é 1.5-aproximativo



TSPM-Christofides é 1.5-aproximativo (e não menos !)

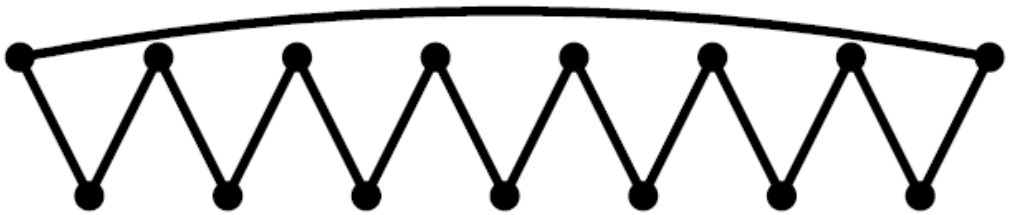
Tight Example



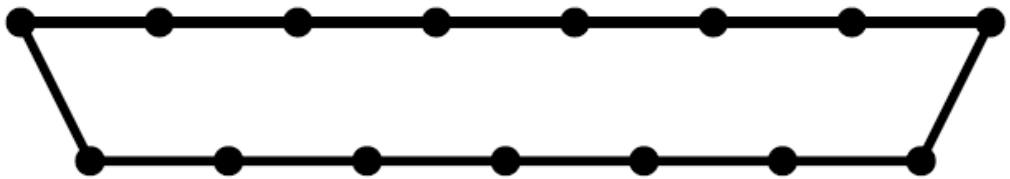
G
$2n+1$ vértices

arestas desenhadas: custo 1
demais arestas: custo = menor caminho

T



C
custo $2n+n$
$= 3n$



opt
custo $2n+1$

Conclusão (até agora)

* **Escalonamento:** Algoritmo 2-aproximativo.
* **TSPM:** Alg 1.5-aproximativo. Parece ser o melhor possível, na prática.
* **Cobertura por Conjuntos:** Algoritmo ($\ln n$)-aproximativo. Parece o melhor possível (parece não ter aproximativo para fator constante).
* **Mochila:** FPTAS. É o melhor possível.

- Problemas NP-Difíceis com FPTAS (Mochila)
- Problemas NP-Difíceis com PTAS, que provavelmente não tem FPTAS (falta provar)
- Problemas NP-Difíceis com α-aproximação (p/ α const), que provavelmente não tem PTAS (TSP Métrico - falta provar)
- Problemas NP-Difíceis com $\alpha(n)$-aproximação (para $\alpha(n)$ **não** const), que provavelmente não tem para α const (Set Cover)
- Problemas NP-Difíceis que provavelmente **não** tem algoritmo poli $\alpha(n)$-aproximativo para nenhum $\alpha(n)$ poli (Caixeiro viajante)