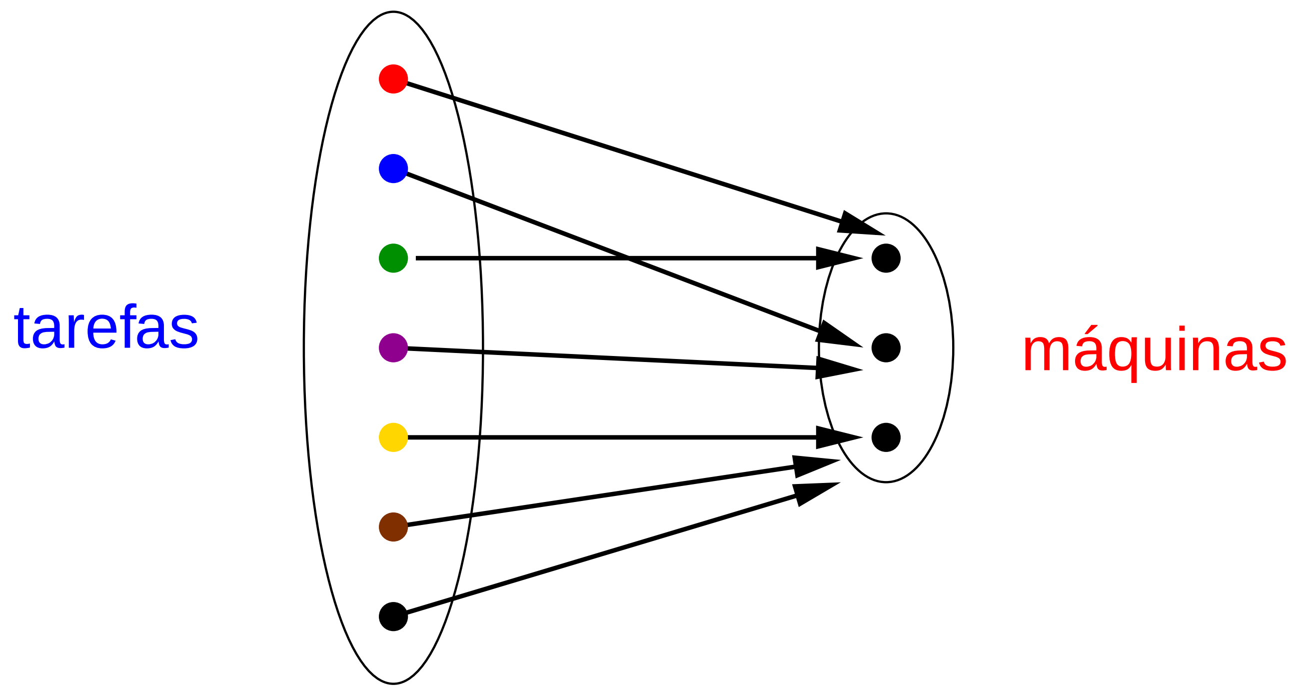
Algoritmos de Aproximação

Segundo Semestre de 2012

Escalonamento de máquinas idênticas

Dados: m máquinas
n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$

um escalonamento é uma partição $\{M[1], \ldots, M[m]\}$ de $[n]$



Encontrar escalonamento com tempo de conclusão mínimo.

Algoritmo de busca local

Começa com um escalonamento qualquer.

Algoritmo de busca local

Começa com um escalonamento qualquer.

Repita o seguinte:

Seja j uma tarefa que termina por último.

Algoritmo de busca local

Começa com um escalonamento qualquer.

Repita o seguinte:

Seja j uma tarefa que termina por último.

Seja C_j o instante em que a tarefa j termina.

Algoritmo de busca local

Começa com um escalonamento qualquer.

Repita o seguinte:

Seja j uma tarefa que termina por último.

Seja C_j o instante em que a tarefa j termina.

Se existe máquina i ociosa antes do instante $C_j - d_j$,
então mova a tarefa j para a máquina i.

Algoritmo de busca local

Começa com um escalonamento qualquer.

Repita o seguinte:

Seja j uma tarefa que termina por último.

Seja C_j o instante em que a tarefa j termina.

Se existe máquina i ociosa antes do instante $C_j - d_j$,
então mova a tarefa j para a máquina i.

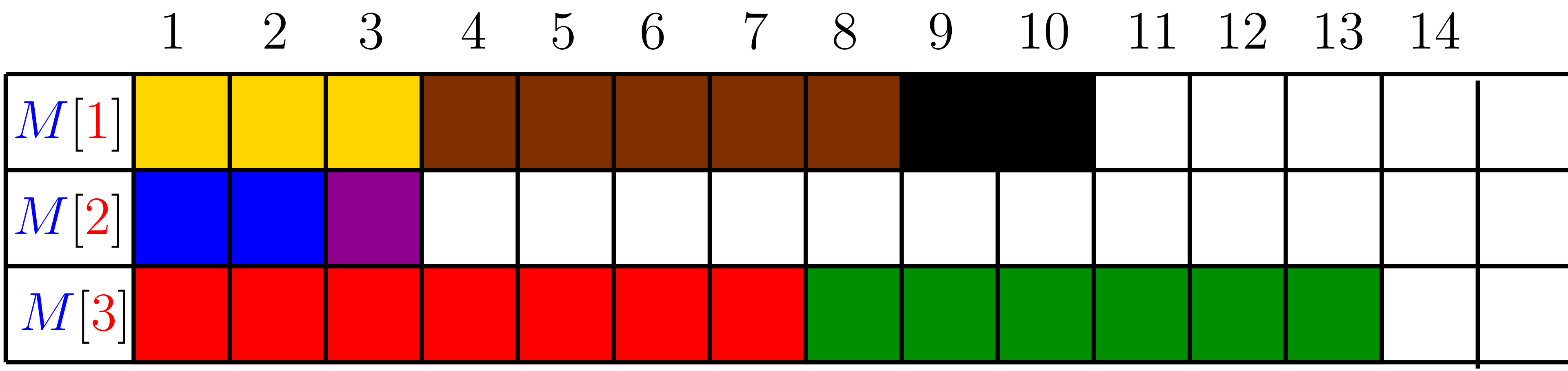
Quanto tempo consome este algoritmo?

Qual é a razão de aproximação?

Exemplo

$m = 3 \qquad n = 7$

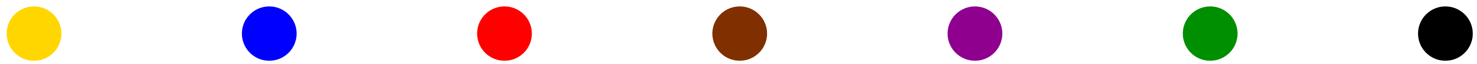
$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$
3	2	7	5	1	6	2



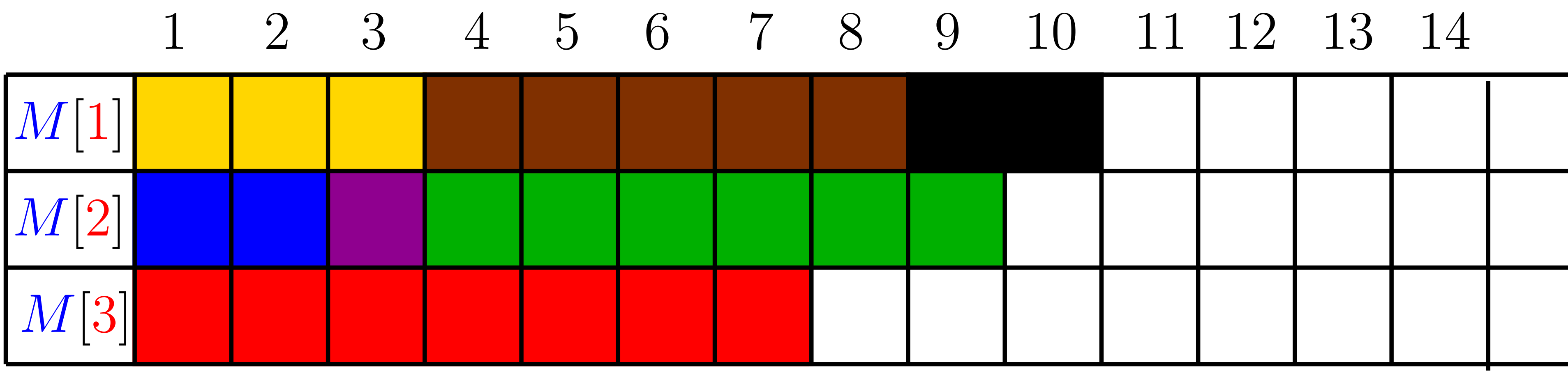
$\{\{1,4,7\},\{2,5\},\{3,6\}\} \Rightarrow$ Tempo de conclusão $= 13$

Exemplo

$m = 3 \quad n = 7$



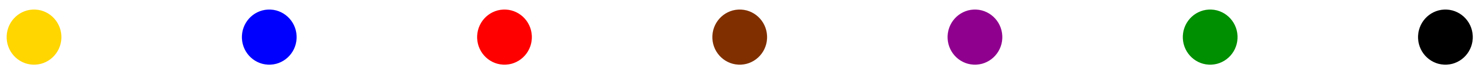
$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$
3	2	7	5	1	6	2



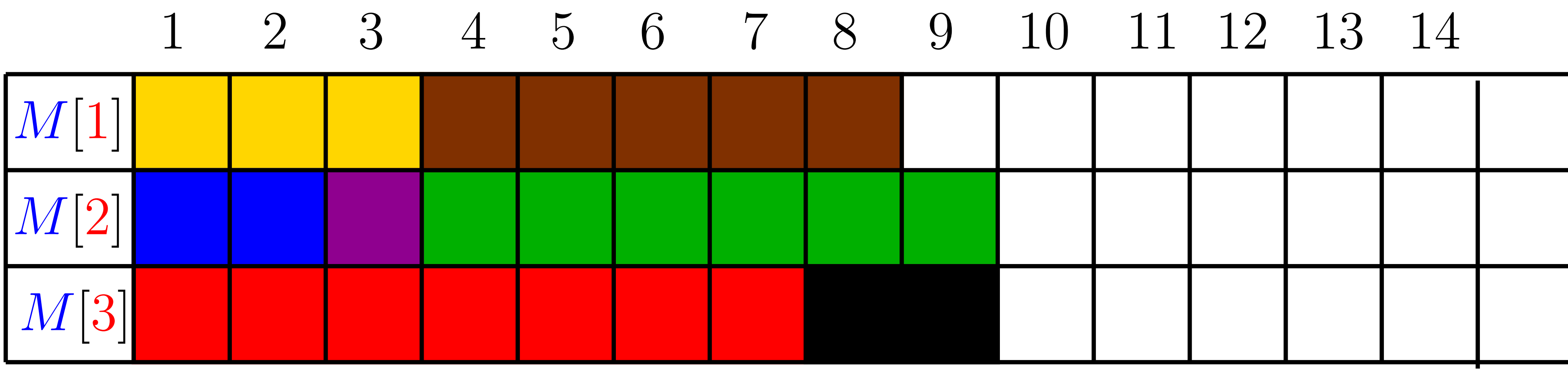
$\{\{1, 4, 7\}, \{2, 5, 6\}, \{3\}\} \Rightarrow$ Tempo de conclusão $= 10$

Exemplo

$m = 3 \qquad n = 7$



$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$
3	2	7	5	1	6	2



$\{\{1,4\},\{2,5,6\},\{3,7\}\} \Rightarrow$ Tempo de conclusão $= 9$

Consumo de tempo

Consideremos a variante do algoritmo que sempre escolhe a máquina i que fica ociosa mais cedo.

Consumo de tempo

Consideremos a variante do algoritmo que sempre escolhe a máquina i que fica ociosa mais cedo.

Cada tarefa muda de máquina no máximo uma vez!

Consumo de tempo

Consideremos a variante do algoritmo que sempre escolhe a máquina i que fica ociosa mais cedo.

Cada tarefa muda de máquina no máximo uma vez!

Seja C_i o instante em que a máquina i termina de executar suas tarefas.

Seja $C_{\min}$ o valor do menor C_i.

Consumo de tempo

Consideremos a variante do algoritmo que sempre escolhe a máquina i que fica ociosa mais cedo.

Cada tarefa muda de máquina no máximo uma vez!

Seja C_i o instante em que a máquina i termina de executar suas tarefas.

Seja $C_{\min}$ o valor do menor C_i.

Veja que $C_{\min}$ não diminui durante a execução do algoritmo.

Consumo de tempo

Consideremos a variante do algoritmo que sempre escolhe a máquina i que fica ociosa mais cedo.

Cada tarefa muda de máquina no máximo uma vez!

Seja C_i o instante em que a máquina i termina de executar suas tarefas.

Seja $C_{\min}$ o valor do menor C_i.

Veja que $C_{\min}$ não diminui durante a execução do algoritmo.

Se a tarefa j foi de uma máquina i para i' e mais tarde de i' para i'', isso leva a uma contradição, pois i'' teria que estar livre mais cedo que i'.

Qualidade do escalonamento produzido

A mesma análise do algoritmo de Graham se aplica!

Qualidade do escalonamento produzido

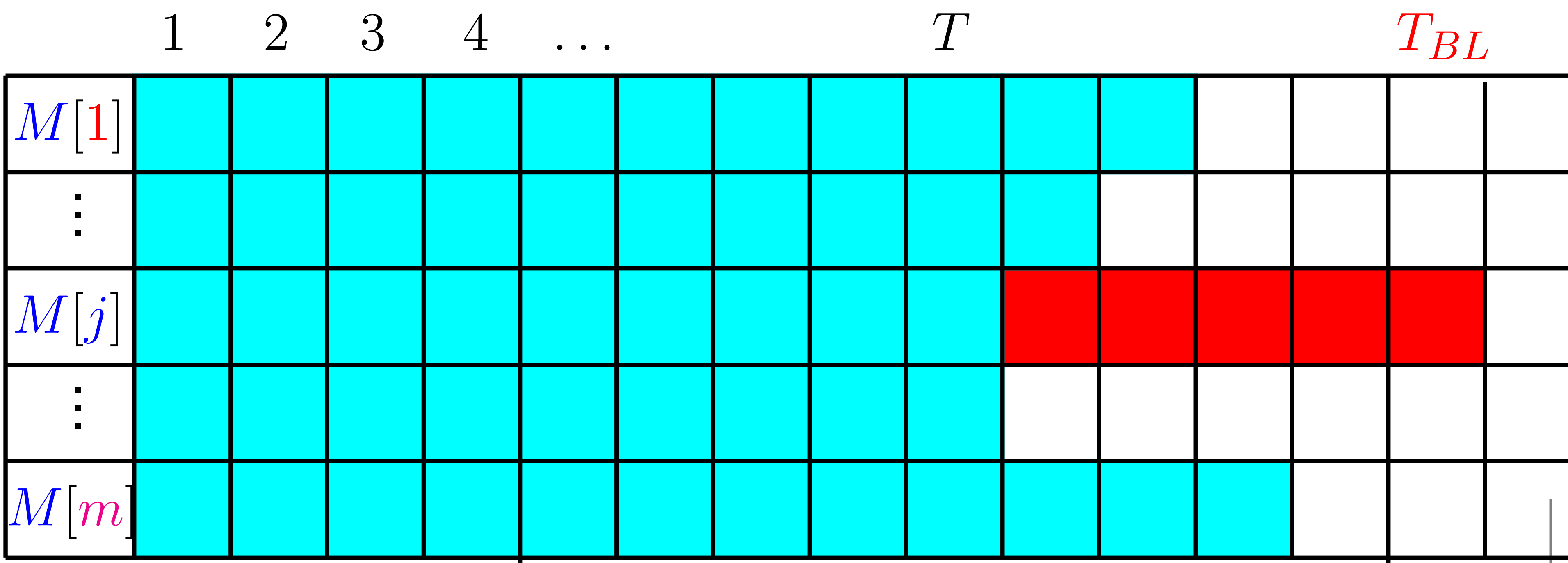
A mesma análise do algoritmo de Graham se aplica!

O escalonamento produzido tem a mesma característica do Graham usada na análise: todas as máquinas estão ocupadas até o instante T anterior à última tarefa a terminar começar a sua execução.

Qualidade do escalonamento produzido

A mesma análise do algoritmo de Graham se aplica!

O escalonamento produzido tem a mesma característica do Graham usada na análise: todas as máquinas estão ocupadas até o instante T anterior à última tarefa a terminar começar a sua execução.



Delimitações para OPT

$\mathrm{OPT} =$ menor tempo de conclusão de um escalonamento

- Duração da tarefa mais longa:

$$\mathrm{OPT} \geq \max\{d[1], d[2], \ldots, d[n]\}$$

- Distribuição balanceada:

$$\mathrm{OPT} \geq \frac{d[1] + d[2] + \cdots + d[n]}{m}$$

Delimitações para OPT

OPT $=$ menor tempo de conclusão de um escalonamento

- Duração da tarefa mais longa:

$$\mathrm{OPT} \geq \max\{d[1], d[2], \ldots, d[n]\}$$

- Distribuição balanceada:

$$\mathrm{OPT} \geq \frac{d[1] + d[2] + \cdots + d[n]}{m}$$

$$T_{BL} \leq \frac{d[1] + \cdots + d[n]}{m} + \max\{d[1], \ldots, d[n]\} \leq 2\,\mathrm{OPT}$$

Delimitações para OPT

$\mathrm{OPT} =$ menor tempo de conclusão de um escalonamento

- Duração da tarefa mais longa:

$$\mathrm{OPT} \geq \max\{d[1], d[2], \ldots, d[n]\}$$

- Distribuição balanceada:

$$\mathrm{OPT} \geq \frac{d[1] + d[2] + \cdots + d[n]}{m}$$

$$T_{BL} \ \leq \ \frac{d[1] + \cdots + d[n]}{m} + \max\{d[1], \ldots, d[n]\} \ \leq \ 2\,\mathrm{OPT}$$

Teorema: O algoritmo de busca local é uma 2-aproximação.

Problema do Corte Máximo

Grafo $G = (V, E)$ sem laços.

Para $X \subseteq V$, denotamos por $\bar{X}$ o complemento de X.

Problema do Corte Máximo

Grafo $G = (V, E)$ sem laços.

Para $X \subseteq V$, denotamos por $\bar{X}$ o complemento de X.

$\delta(X)$: conjunto das arestas com exatamente uma ponta em X.

$\delta(X)$ é um corte em G.

Problema do Corte Máximo

Grafo $G = (V, E)$ sem laços.

Para $X \subseteq V$, denotamos por $\bar{X}$ o complemento de X.

$\delta(X)$: conjunto das arestas com exatamente uma ponta em X.

$\delta(X)$ é um corte em G.

Problema: Dado G, encontrar um corte máximo em G.

Problema do Corte Máximo

Grafo $G = (V, E)$ sem laços.

Para $X \subseteq V$, denotamos por $\bar{X}$ o complemento de X.

$\delta(X)$: conjunto das arestas com exatamente uma ponta em X.

$\delta(X)$ é um corte em G.

Problema: Dado G, encontrar um corte máximo em G.

Sabemos encontrar cortes mínimos em tempo polinomial.
Mas o problema acima é NP-difícil...

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

- Se existe v em X com mais vizinhos em X que em $\bar{X}$, então remova v de X.
- Se existe v em $\bar{X}$ com mais vizinhos em $\bar{X}$ que em X, então inclua v em X.

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

- Se existe v em X com mais vizinhos em X que em $\bar{X}$, então remova v de X.
- Se existe v em $\bar{X}$ com mais vizinhos em $\bar{X}$ que em X, então inclua v em X.

Pare quando não houver mais alterações e devolva $\delta(X)$.

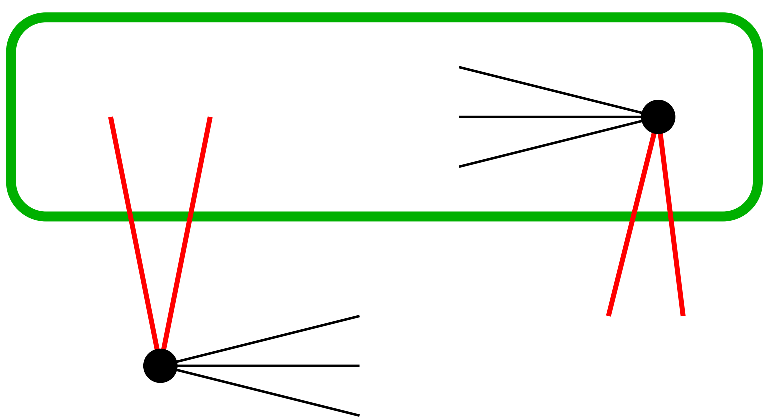
Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

- Se existe v em X com mais vizinhos em X que em $\bar{X}$, então remova v de X.
- Se existe v em $\bar{X}$ com mais vizinhos em $\bar{X}$ que em X, então inclua v em X.

Pare quando não houver mais alterações e devolva $\delta(X)$.



Algoritmo de busca local

Para cada vértice v em G,
$g(v)$ é o grau de v e $\delta(v) = \delta(\{v\})$.

Algoritmo de busca local

Para cada vértice v em G,
$g(v)$ é o grau de v e $\delta(v) = \delta(\{v\})$.

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

Se existe v tal que $|\delta(v) \cap \delta(X)| < |\delta(v) \setminus \delta(X)|$,
então $X \leftarrow X \oplus \{v\}$.

Pare quando não houver mais alterações e devolva $\delta(X)$.

Algoritmo de busca local

Para cada vértice v em G,
$g(v)$ é o grau de v e $\delta(v) = \delta(\{v\})$.

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

Se existe v tal que $|\delta(v) \cap \delta(X)| < |\delta(v) \setminus \delta(X)|$,
então $X \leftarrow X \oplus \{v\}$.

Pare quando não houver mais alterações e devolva $\delta(X)$.

O algoritmo é polinomial:
a cada iteração, o corte $\delta(X)$ cresce.

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

Se existe v tal que $|\delta(v) \cap \delta(X)| < |\delta(v) \setminus \delta(X)|$,
então $X \leftarrow X \oplus \{v\}$.

Pare quando não houver mais alterações e devolva $\delta(X)$.

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

> Se existe v tal que $|\delta(v) \cap \delta(X)| < |\delta(v) \setminus \delta(X)|$,
> então $X \leftarrow X \oplus \{v\}$.

Pare quando não houver mais alterações e devolva $\delta(X)$.

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

> Se existe v tal que $|\delta(v) \cap \delta(X)| < |\delta(v) \setminus \delta(X)|$,
> então $X \leftarrow X \oplus \{v\}$.

Pare quando não houver mais alterações e devolva $\delta(X)$.

Para cada vértice v em G,
pelo menos $g(v)/2$ arestas estão em $\delta(X)$.

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

Se existe v tal que $|\delta(v) \cap \delta(X)| < |\delta(v) \setminus \delta(X)|$,
então $X \leftarrow X \oplus \{v\}$.

Pare quando não houver mais alterações e devolva $\delta(X)$.

Para cada vértice v em G,
pelo menos $g(v)/2$ arestas estão em $\delta(X)$.

Então $|\delta(X)| \;\geq\; \frac{1}{2} \sum_{v \in V} \frac{g(v)}{2} \;\geq\; \frac{2|E|}{4} \;=\; \frac{|E|}{2} \;\geq\; \frac{\mathrm{OPT}}{2}$.

Algoritmo de busca local

Comece com um corte $\delta(X)$ arbitrário de G.

Repita o seguinte:

Se existe v tal que $|\delta(v) \cap \delta(X)| < |\delta(v) \setminus \delta(X)|$,
então $X \leftarrow X \oplus \{v\}$.

Pare quando não houver mais alterações e devolva $\delta(X)$.

Para cada vértice v em G,
pelo menos $g(v)/2$ arestas estão em $\delta(X)$.

Então $|\delta(X)| \ \geq \ \frac{1}{2} \sum_{v \in V} \frac{g(v)}{2} \ \geq \ \frac{2|E|}{4} \ = \ \frac{|E|}{2} \ \geq \ \frac{\mathrm{OPT}}{2}$.

Teorema: O algoritmo acima é uma 2-aproximação.

Algoritmo guloso

$G = (V, E)$ $X \subseteq V$ $v \in V \setminus X$

Seja $\beta(v, X)$ o número de arestas em E de v para X.

Algoritmo guloso

$G = (V, E)$ $\qquad X \subseteq V$ $\qquad v \in V \setminus X$

Seja $\beta(v, X)$ o número de arestas em E de v para X.

GULOSO (n, E) $\qquad \triangleright\ V = \{1, \ldots, n\}$
$X \leftarrow \{1\}$ $\qquad Y \leftarrow \emptyset$
para $i \leftarrow 2$ **até** n **faça**
  se $\beta(i, X) \leq \beta(i, Y)$
    então $X \leftarrow X \cup \{i\}$
    senão $Y \leftarrow Y \cup \{i\}$
devolva $\delta(X)$

Algoritmo guloso

$G = (V, E)$ $\qquad$ $X \subseteq V$ $\qquad$ $v \in V \setminus X$

Seja $\beta(v, X)$ o número de arestas em E de v para X.

GULOSO (n, E) $\qquad \triangleright\ V = \{1, \ldots, n\}$
1 $\quad X \leftarrow \{1\} \qquad Y \leftarrow \emptyset$
2 $\quad$ **para** $i \leftarrow 2$ **até** n **faça**
3 $\qquad$ **se** $\beta(i, X) \leq \beta(i, Y)$
4 $\qquad\quad$ **então** $X \leftarrow X \cup \{i\}$
5 $\qquad\quad$ **senão** $Y \leftarrow Y \cup \{i\}$
6 $\quad$ **devolva** $\delta(X)$

Teorema: GULOSO é uma 2-aproximação.

Algoritmo guloso

Teorema: GULOSO é uma 2-aproximação.

Algoritmo guloso

Teorema: GULOSO é uma 2-aproximação.

r_i: número de arestas de G pelas quais i é responsável

Algoritmo guloso

Teorema: GULOSO é uma 2-aproximação.

r_i: número de arestas de G pelas quais i é responsável

Para X e Y da iteração i, temos que

$$r_i := \beta(i, X) + \beta(i, Y)$$

Algoritmo guloso

Teorema: GULOSO é uma 2-aproximação.

r_i: número de arestas de G pelas quais i é responsável

Para X e Y da iteração i, temos que

$$r_i := \beta(i, X) + \beta(i, Y)$$

Vale que $|E| = \sum_i r_i$.

Algoritmo guloso

Teorema: GULOSO é uma 2-aproximação.

r_i: número de arestas de G pelas quais i é responsável

Para X e Y da iteração i, temos que

$$r_i := \beta(i, X) + \beta(i, Y)$$

Vale que $|E| = \sum_i r_i$.

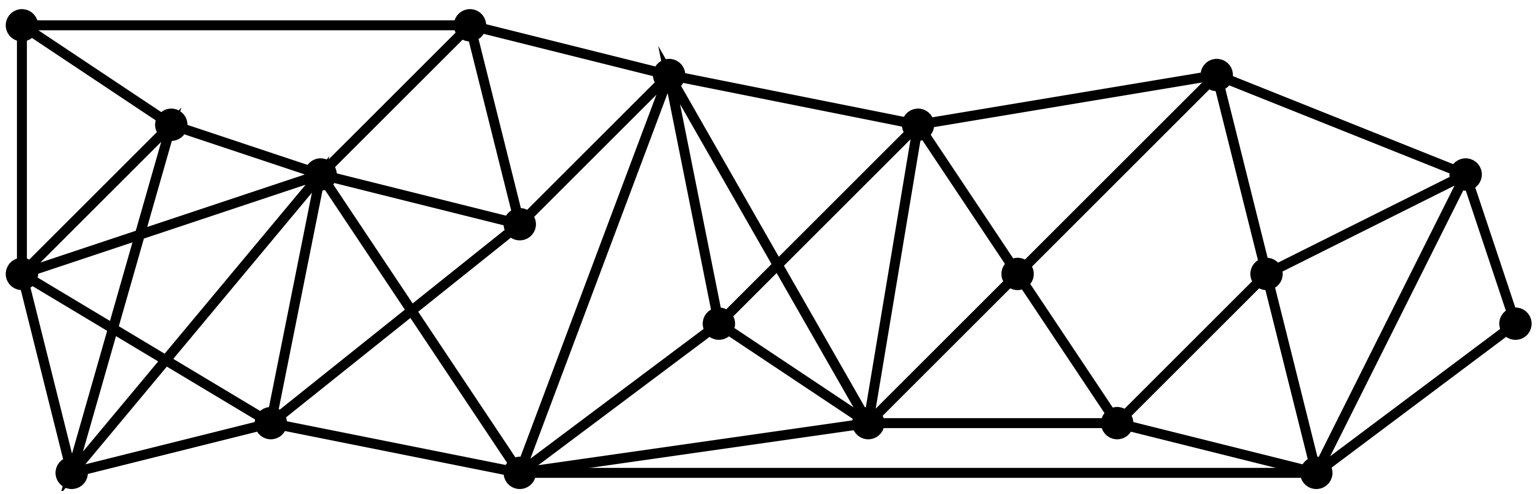
$$|\delta(X)| \geq \sum_i \frac{r_i}{2} \geq \frac{|E|}{2} \geq \frac{\text{OPT}}{2}.$$

Problema do Caixeiro Viajante

Dados

grafo G

comprimento l_{ij} da aresta ij $\quad (ij \in E_G)$

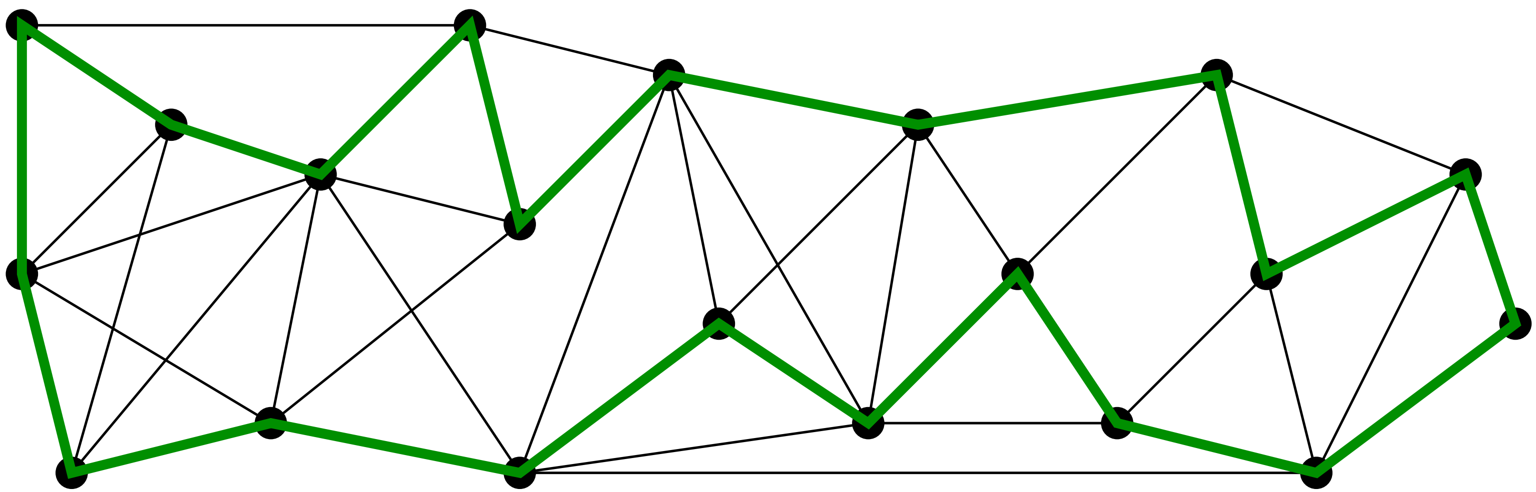


Problema do Caixeiro Viajante

Dados

grafo G

comprimento l_{ij} da aresta ij $\quad(ij \in E_G)$



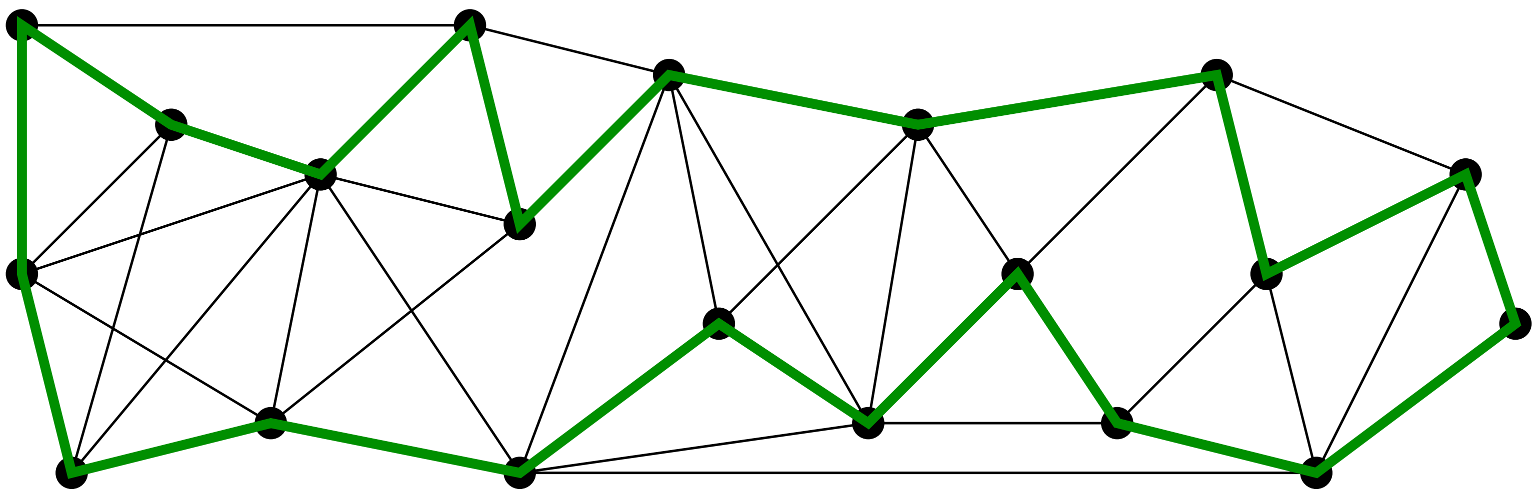
Circuito hamiltoniano: circuito que passa por todos os vértices

Problema do Caixeiro Viajante

Dados

grafo G

comprimento l_{ij} da aresta ij $(ij \in E_G)$

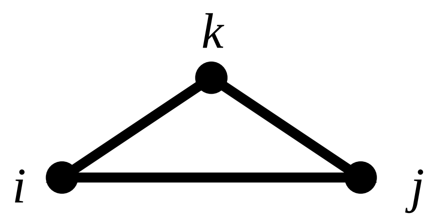


Circuito hamiltoniano: circuito que passa por todos os vértices

Problema (TSP): Dados G e l, encontrar circuito hamiltoniano C em G de comprimento $l(C)$ mínimo.

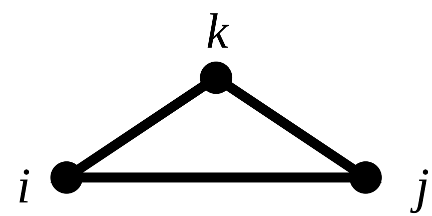
Variantes do Caixeiro Viajante

- TSP métrico
  - grafo completo
  - função comprimento l satisfaz desigualdade triangular: $l_{ij} \leq l_{ik} + l_{kj} \quad \forall i, j, k \in V_G$



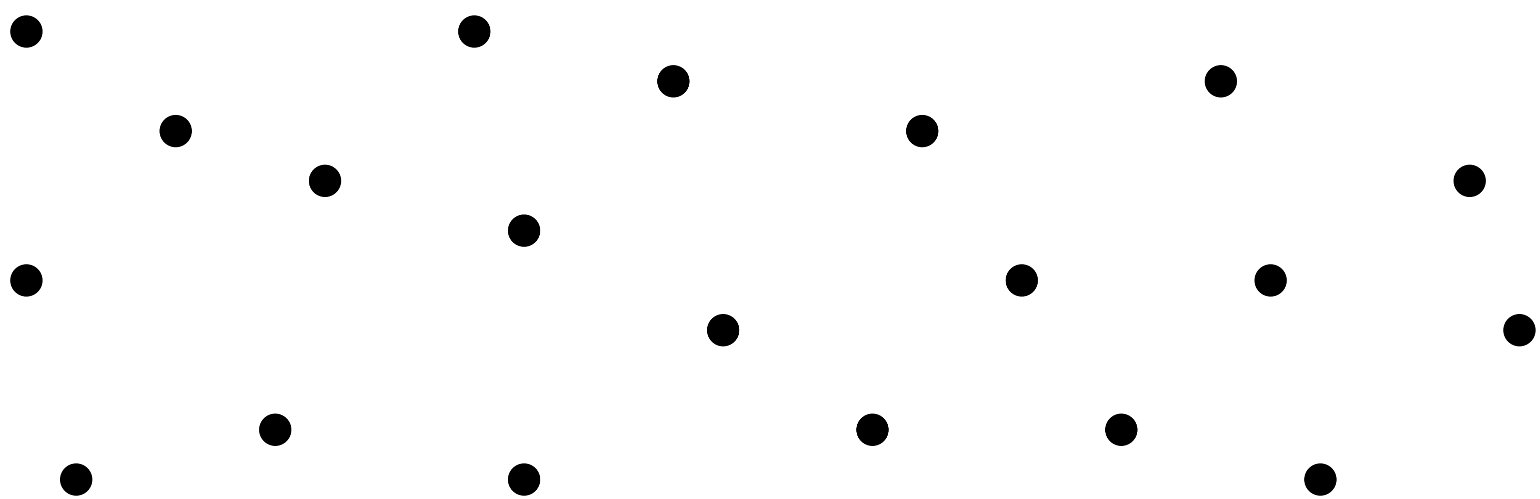
Variantes do Caixeiro Viajante

- TSP métrico
  - grafo completo
  - função comprimento l satisfaz desigualdade triangular: $l_{ij} \leq l_{ik} + l_{kj} \quad \forall i, j, k \in V_G$



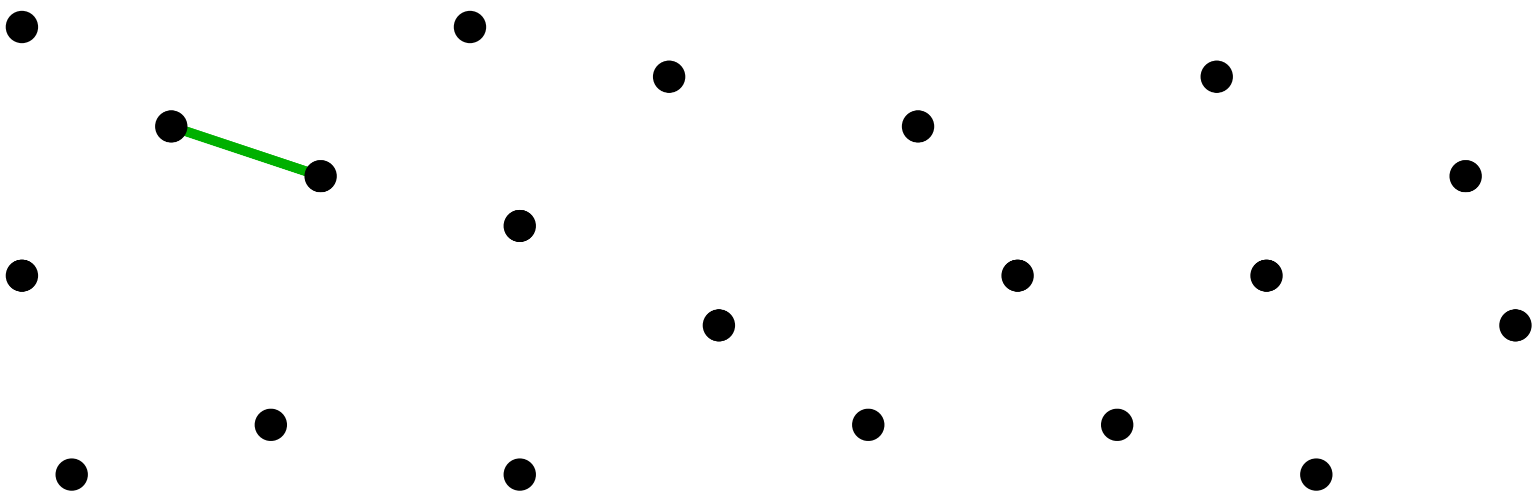
- TSP euclideano
  - Caso particular do métrico
  - Vértices são pontos no plano
    (ou num espaço euclideano qq de dimensão fixa)
  - l_{ij} é a distância euclideana entre i e j

Algoritmo guloso para o TSP métrico



Algoritmo guloso para o TSP métrico

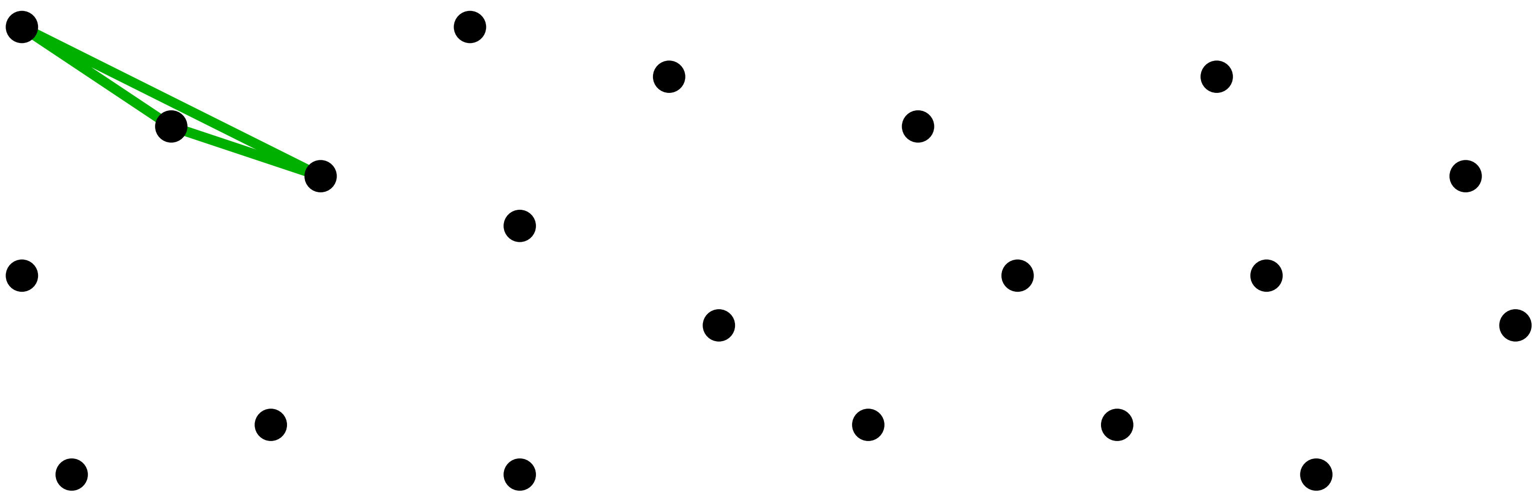
Começa com um par de pontos mais próximos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

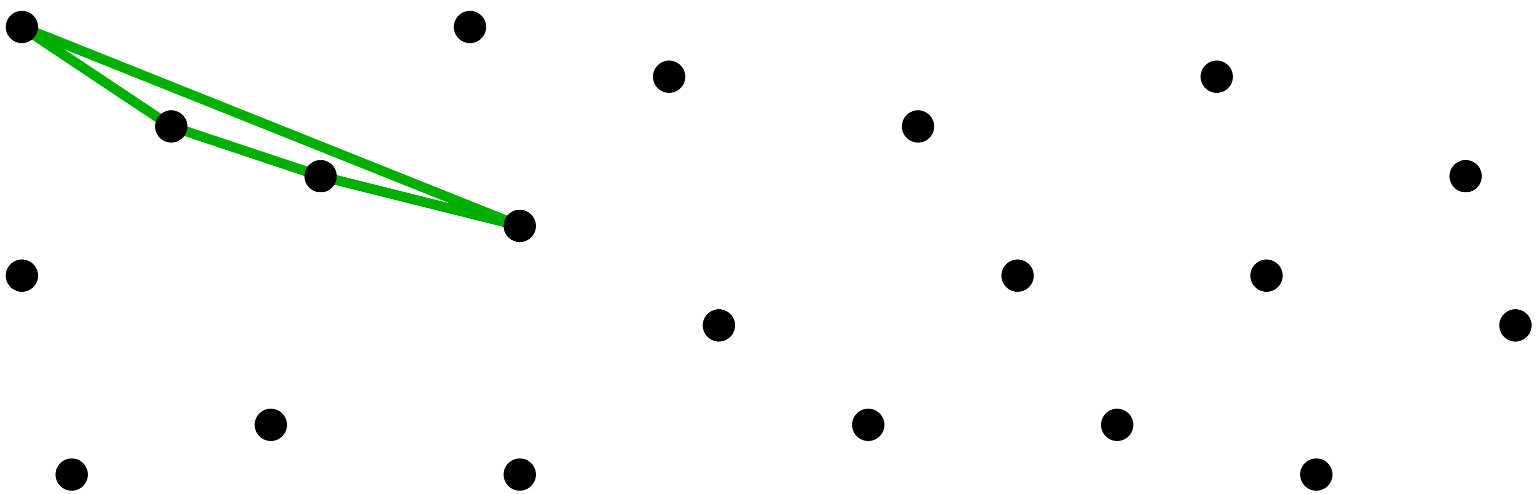
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

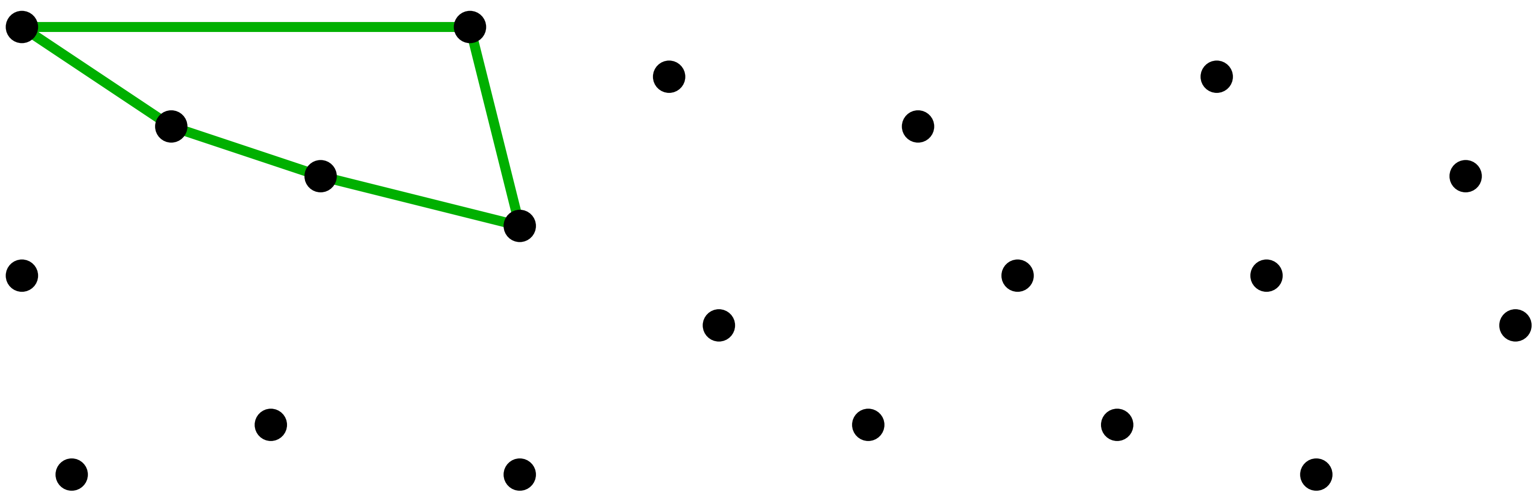
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

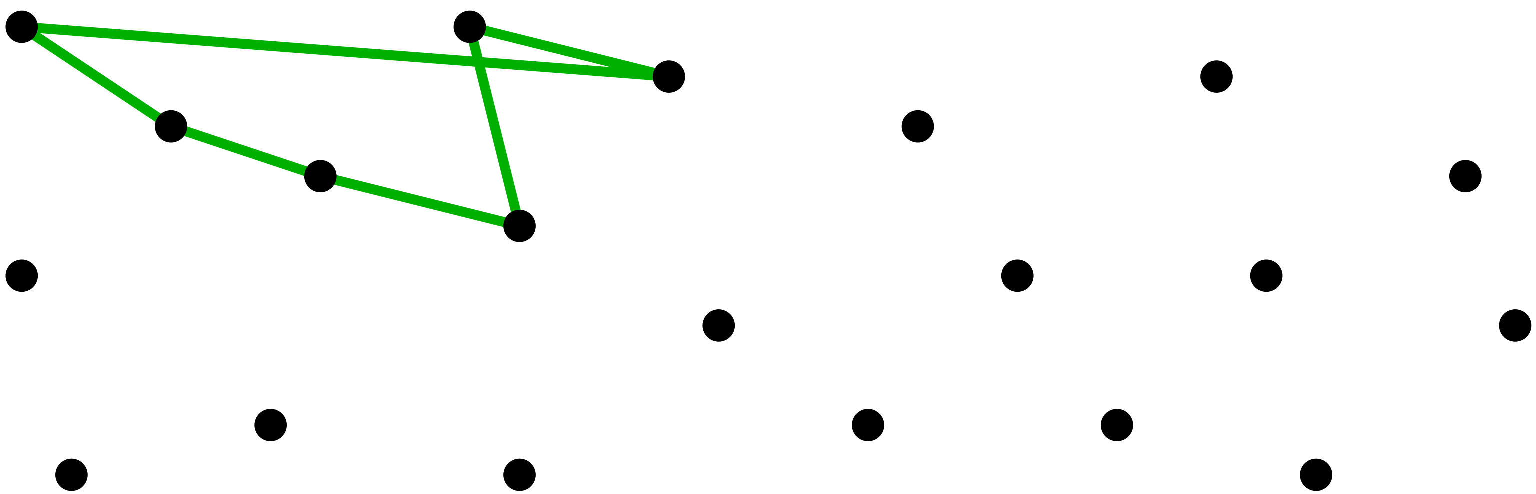
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

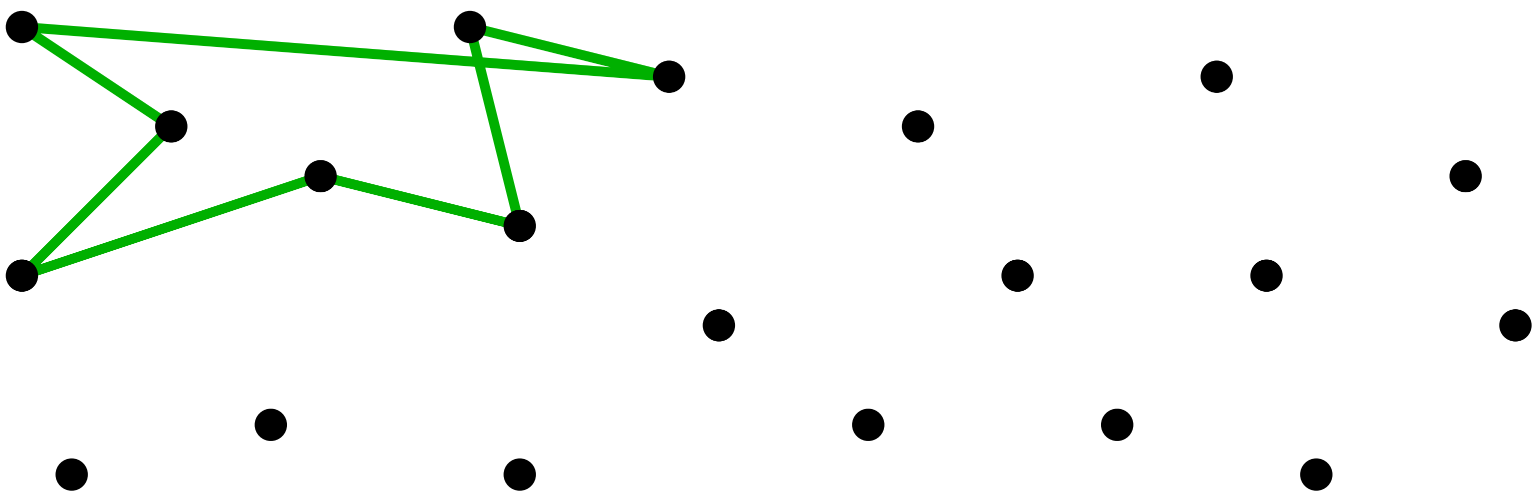
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

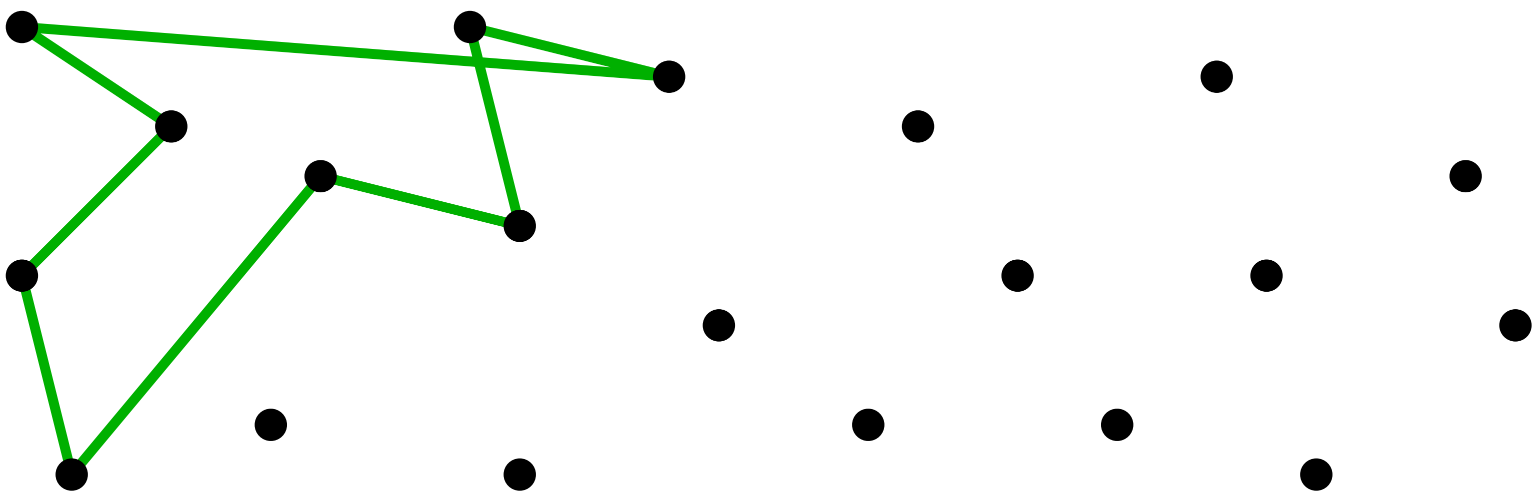
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

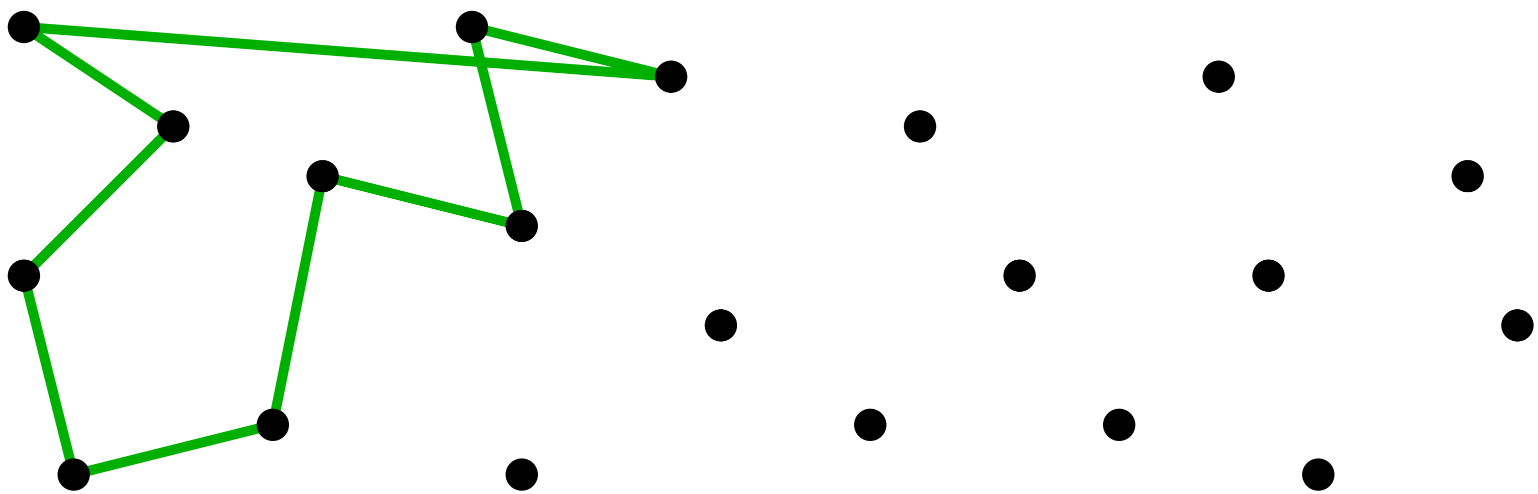
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

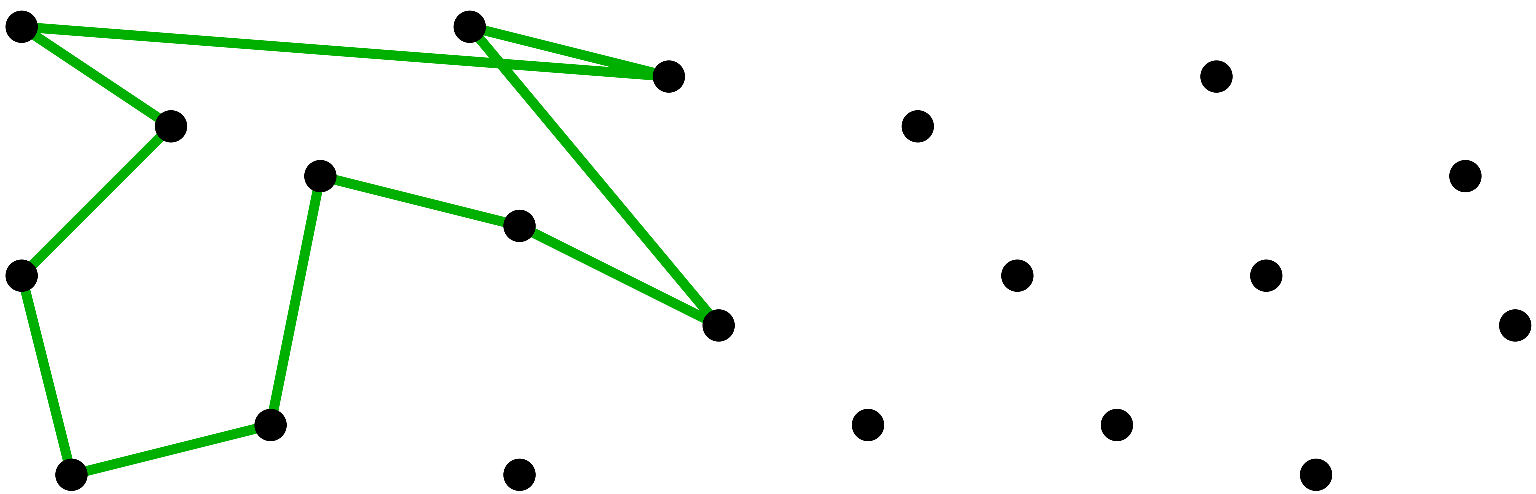
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

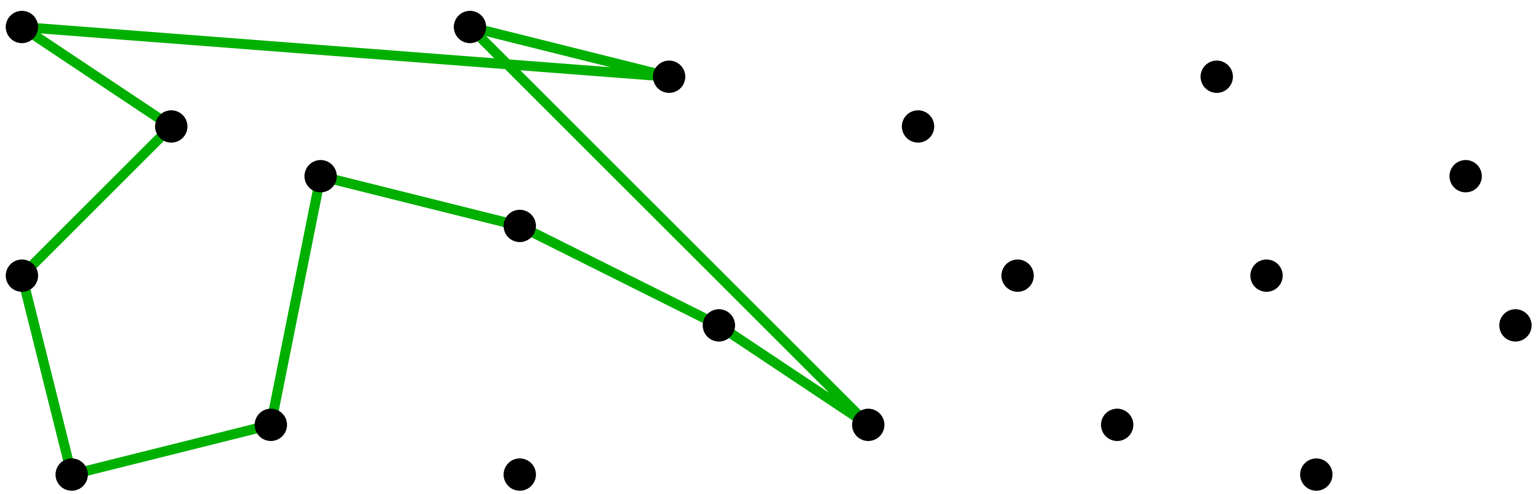
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

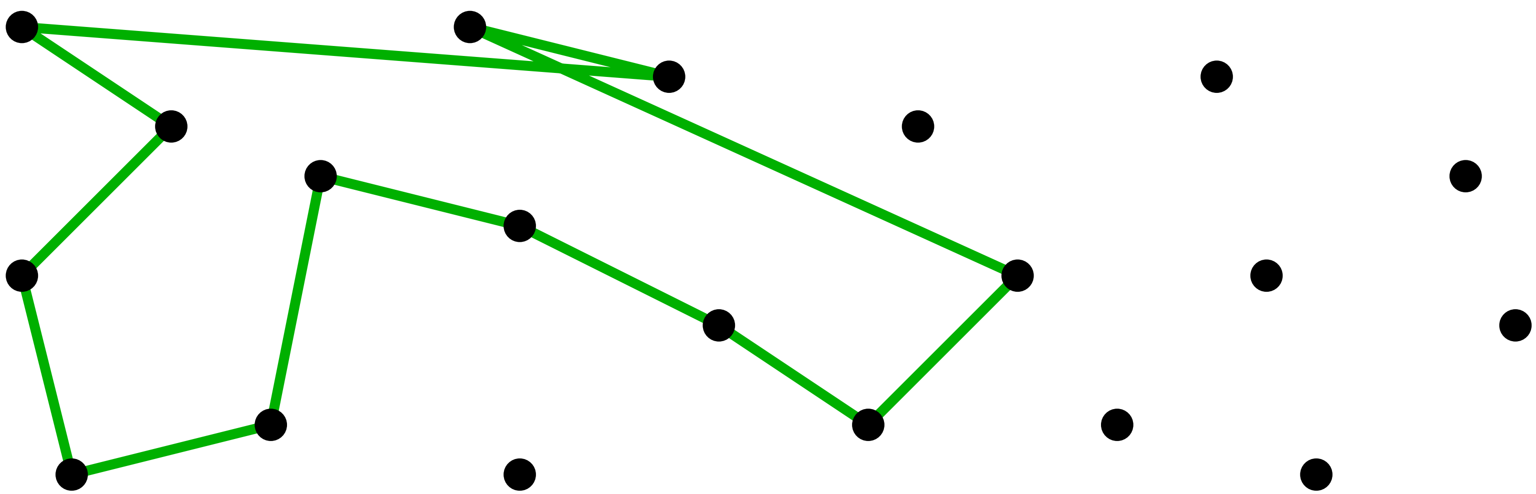
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

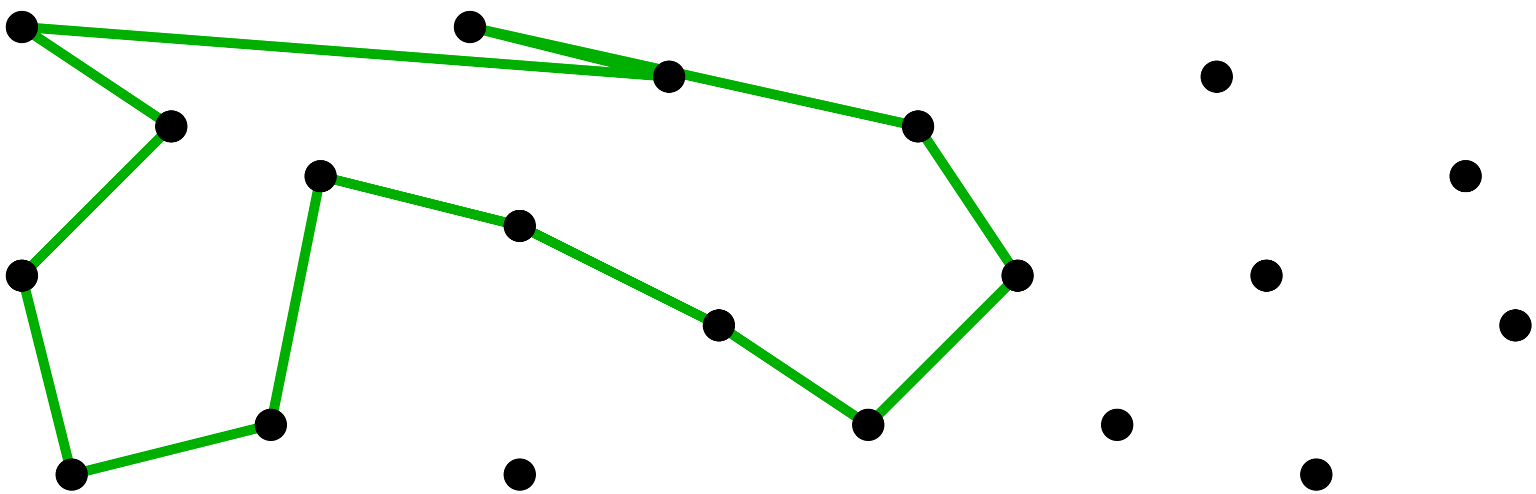
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

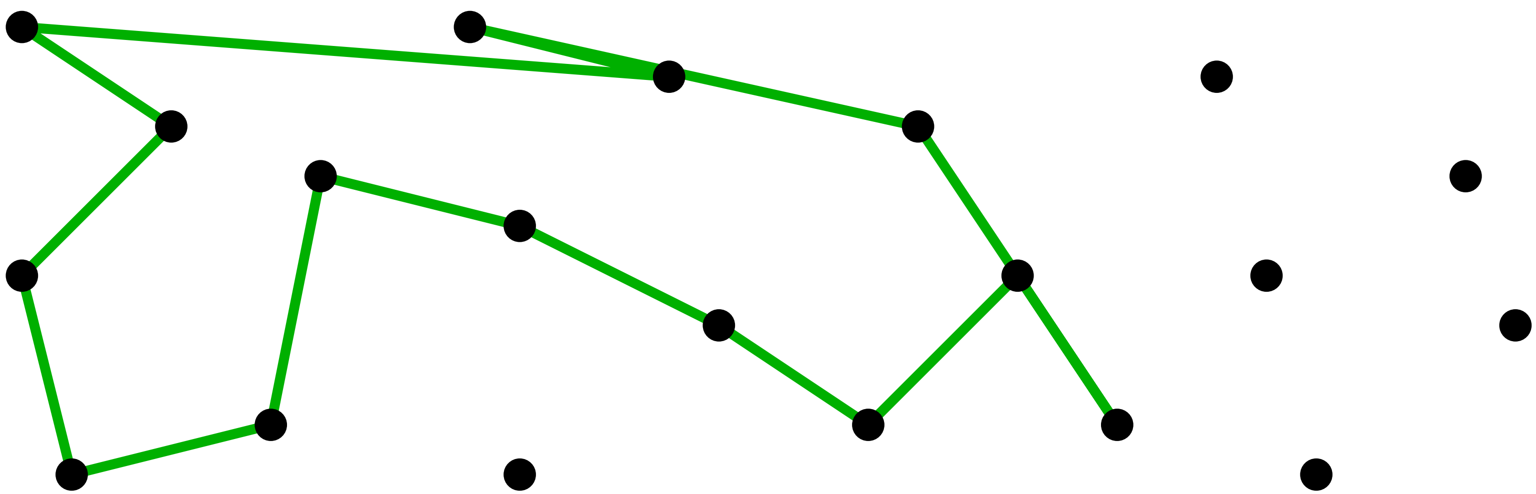
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

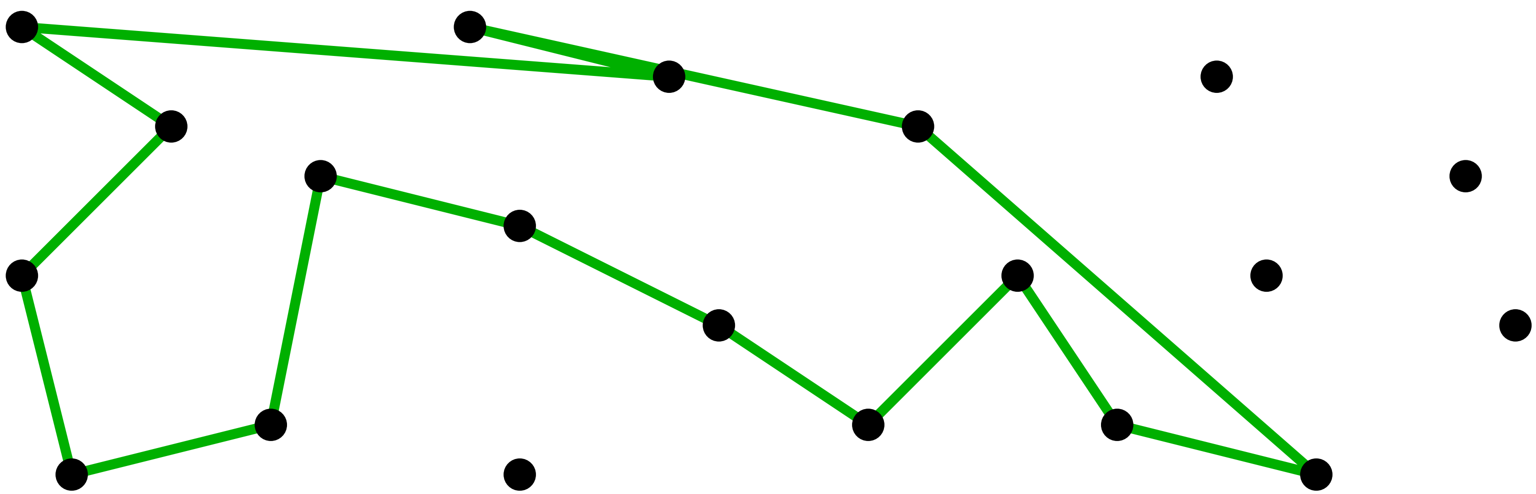
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

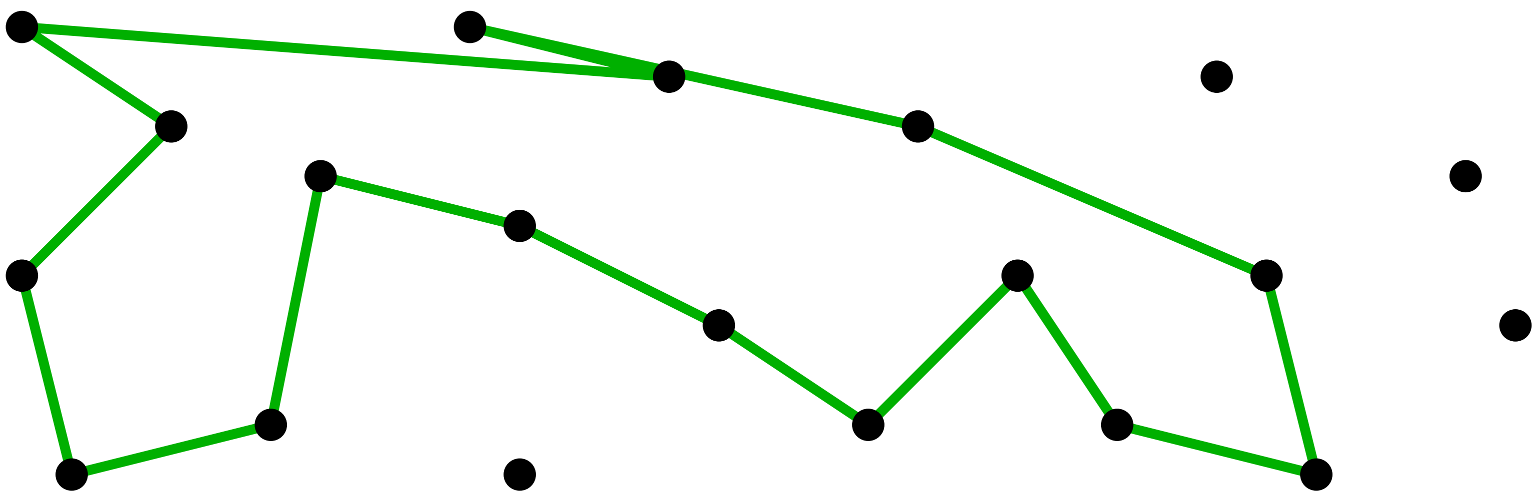
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

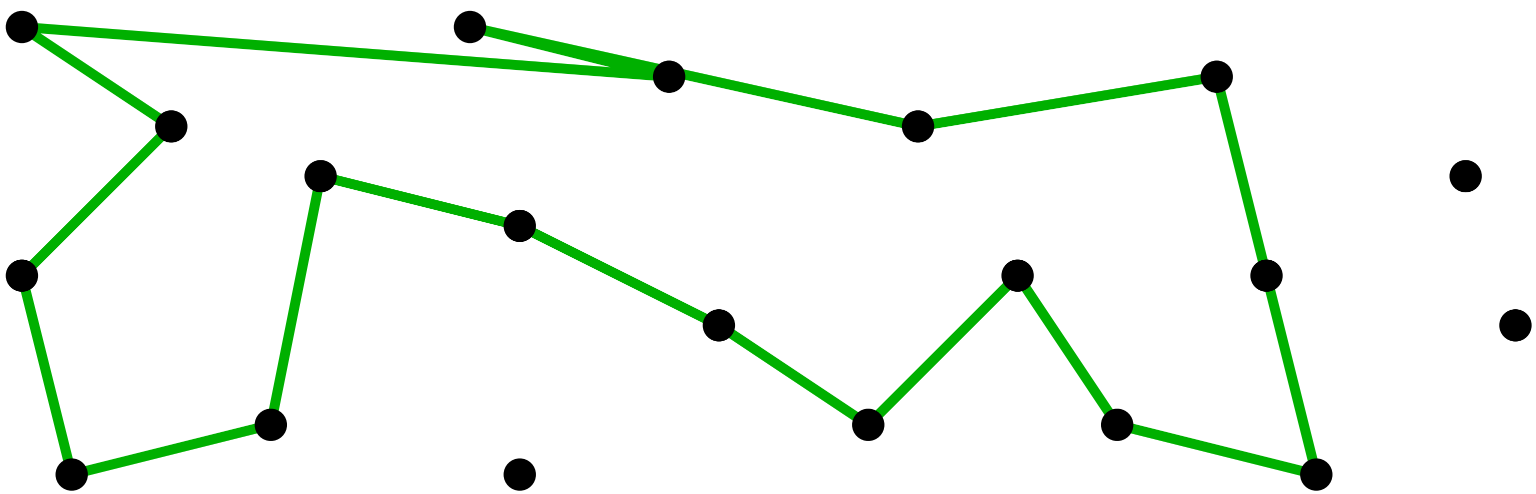
Inclui um ponto mais próximo aos já incluídos.



Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

Inclui um ponto mais próximo aos já incluídos.

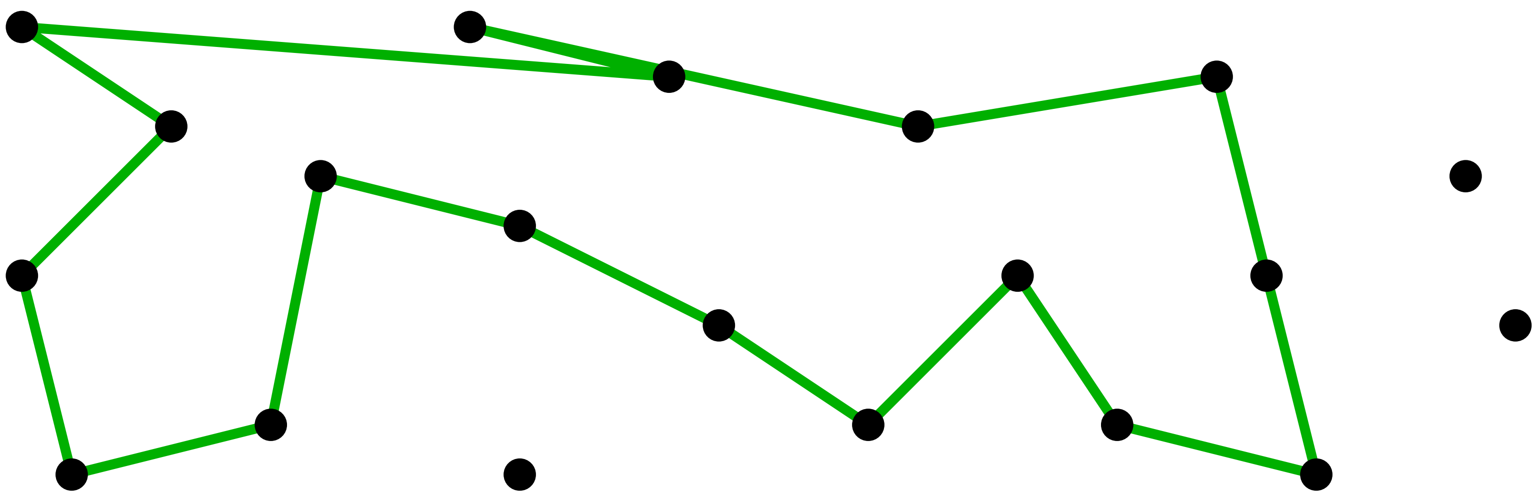


A cada passo, aumenta no máximo $2\,c_{ij}$,
onde j é o ponto de fora mais próximo a um já incluído i.

Algoritmo guloso para o TSP métrico

Começa com um par de pontos mais próximos.

Inclui um ponto mais próximo aos já incluídos.

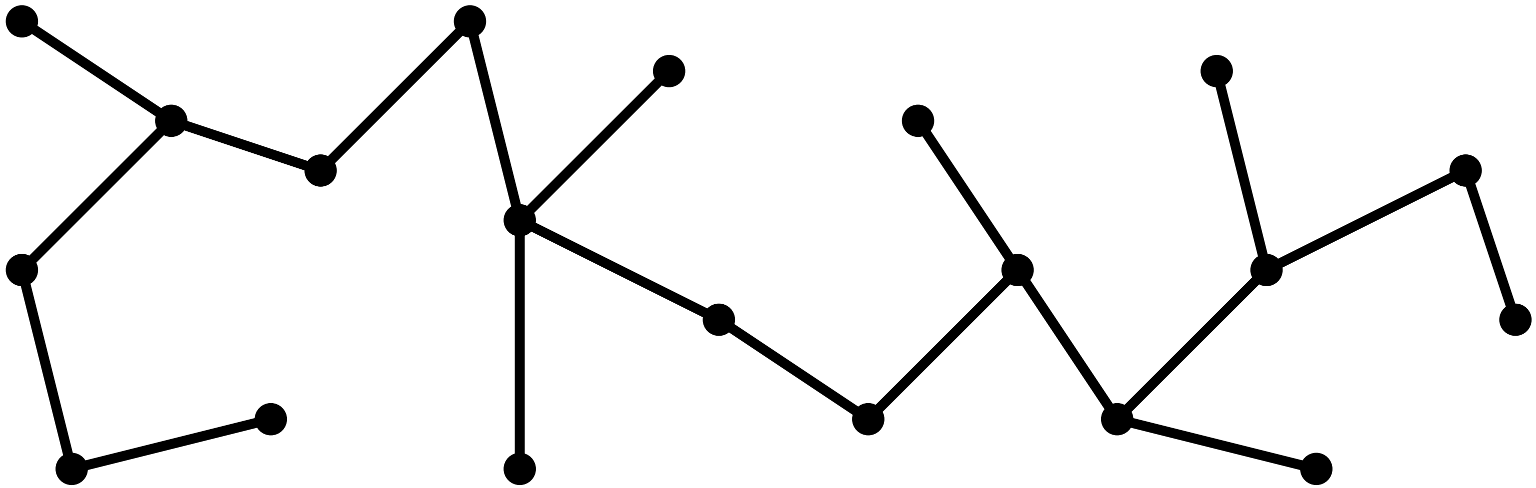


A cada passo, aumenta no máximo $2\,c_{ij}$,
onde j é o ponto de fora mais próximo a um já incluído i.

Teorema: O algoritmo acima é uma 2-aproximação.

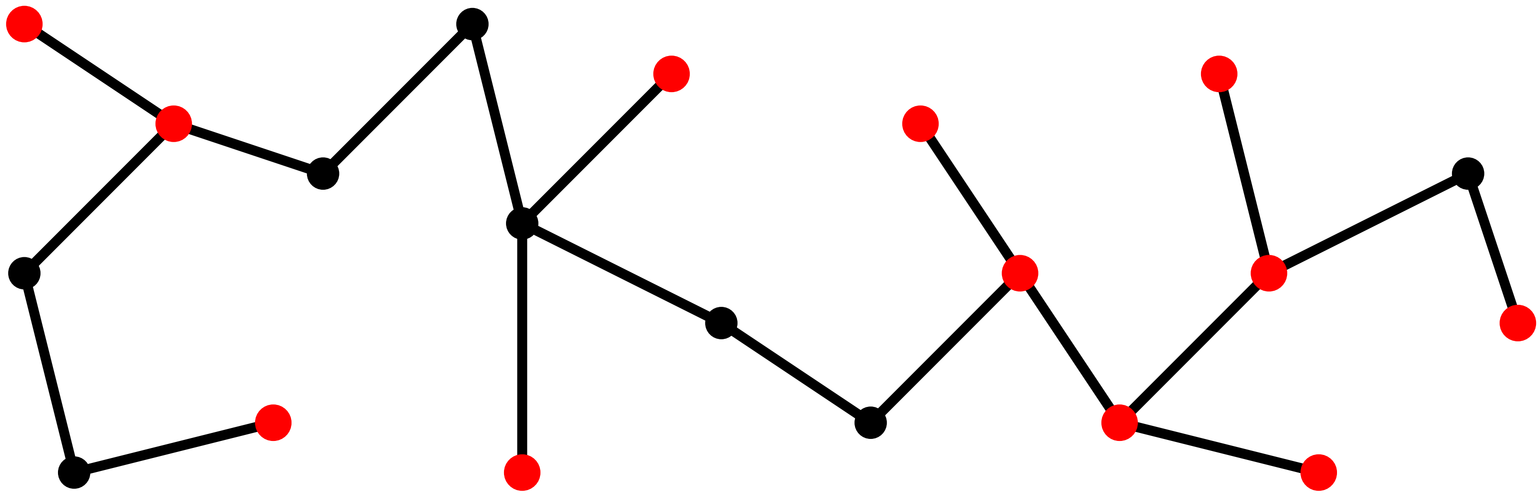
Algoritmo de Christofides

(1) Árvore geradora de comprimento mínimo



Algoritmo de Christofides

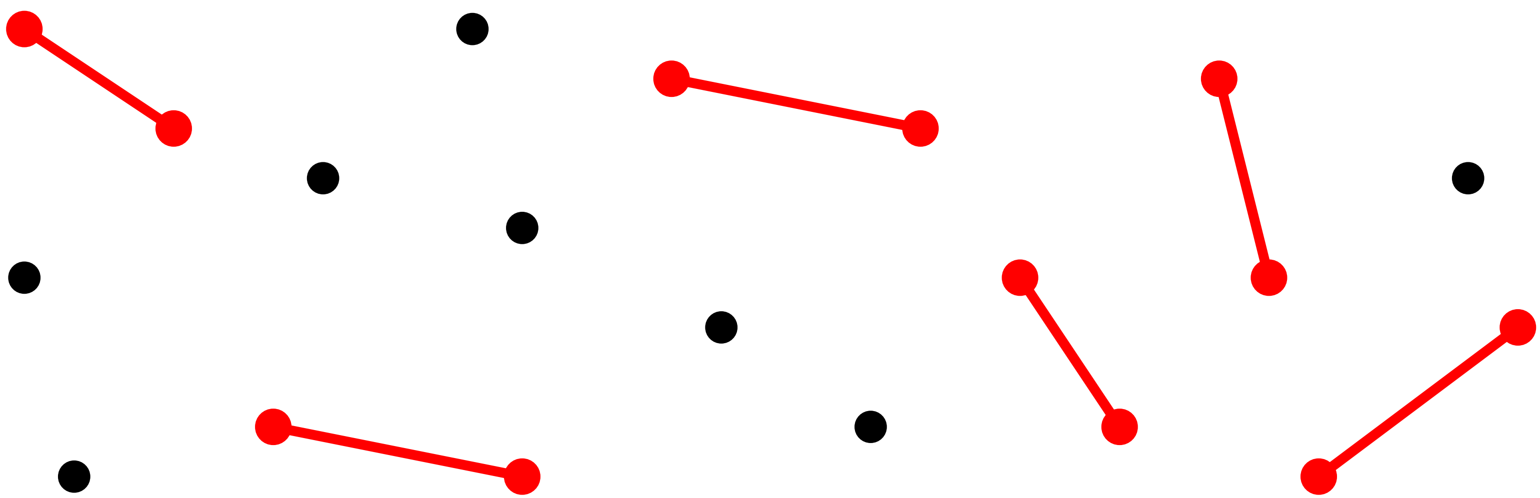
(1) Árvore geradora de comprimento mínimo



vértices de grau ímpar

Algoritmo de Christofides

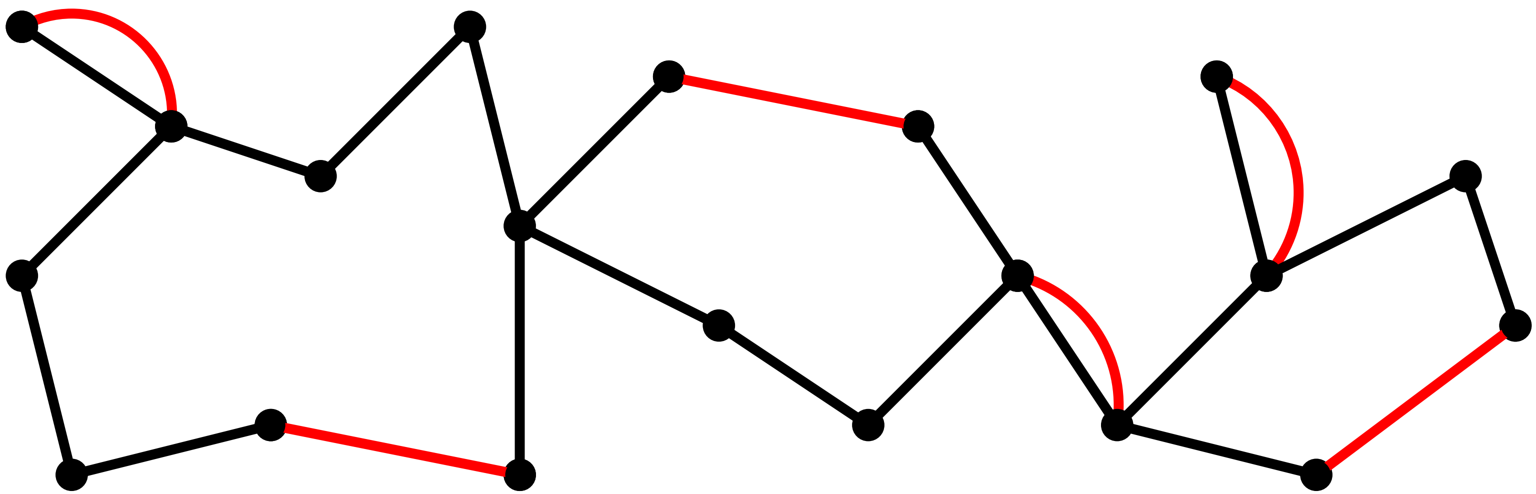
(1) Árvore geradora de comprimento mínimo



(2) Emparelhamento perfeito de comprimento mínimo no subgrafo induzido pelos vértices de grau ímpar (polinomial)

Algoritmo de Christofides

(1) Árvore geradora de comprimento mínimo

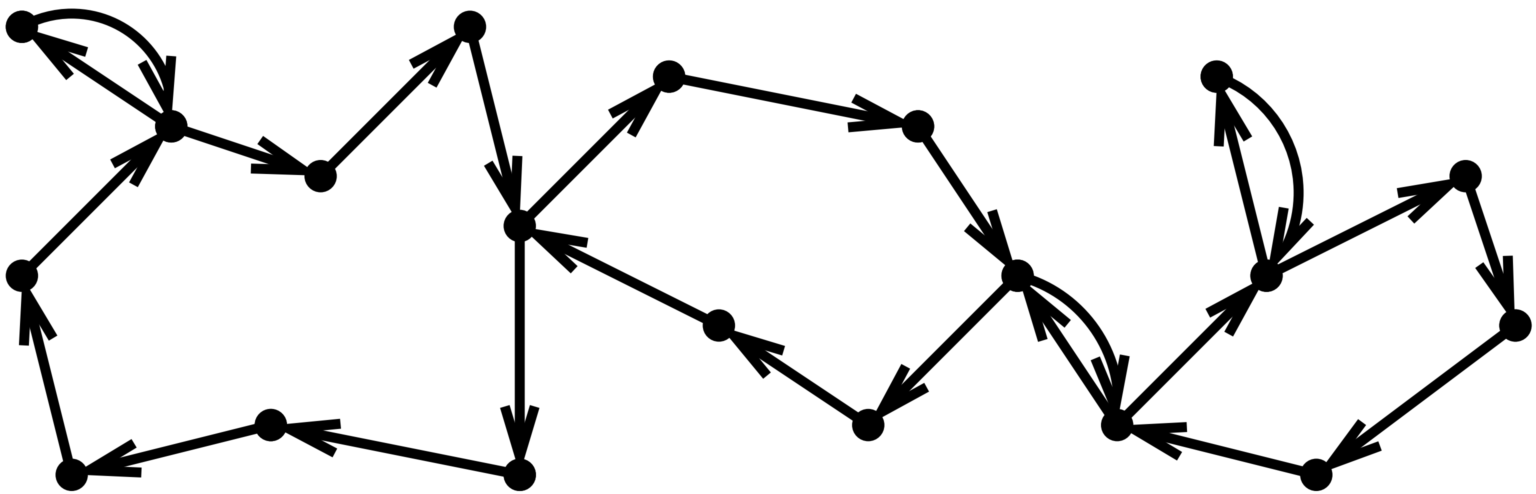


(2) Emparelhamento perfeito de comprimento mínimo no subgrafo induzido pelos vértices de grau ímpar (polinomial)

(3) Junte os dois obtendo T' euleriano

Algoritmo de Christofides

(1) Árvore geradora de comprimento mínimo



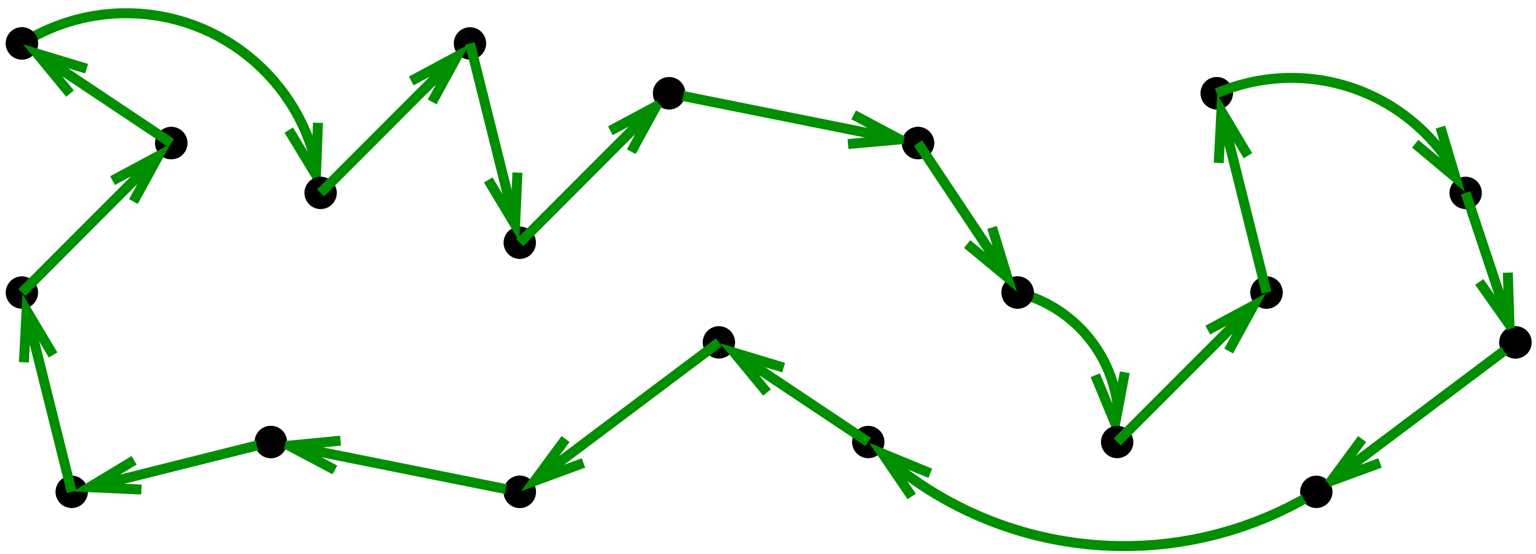
(2) Emparelhamento perfeito de comprimento mínimo no subgrafo induzido pelos vértices de grau ímpar (polinomial)

(3) Junte os dois obtendo T' euleriano

(4) Obtenha ciclo euleriano P de T'

Algoritmo de Christofides

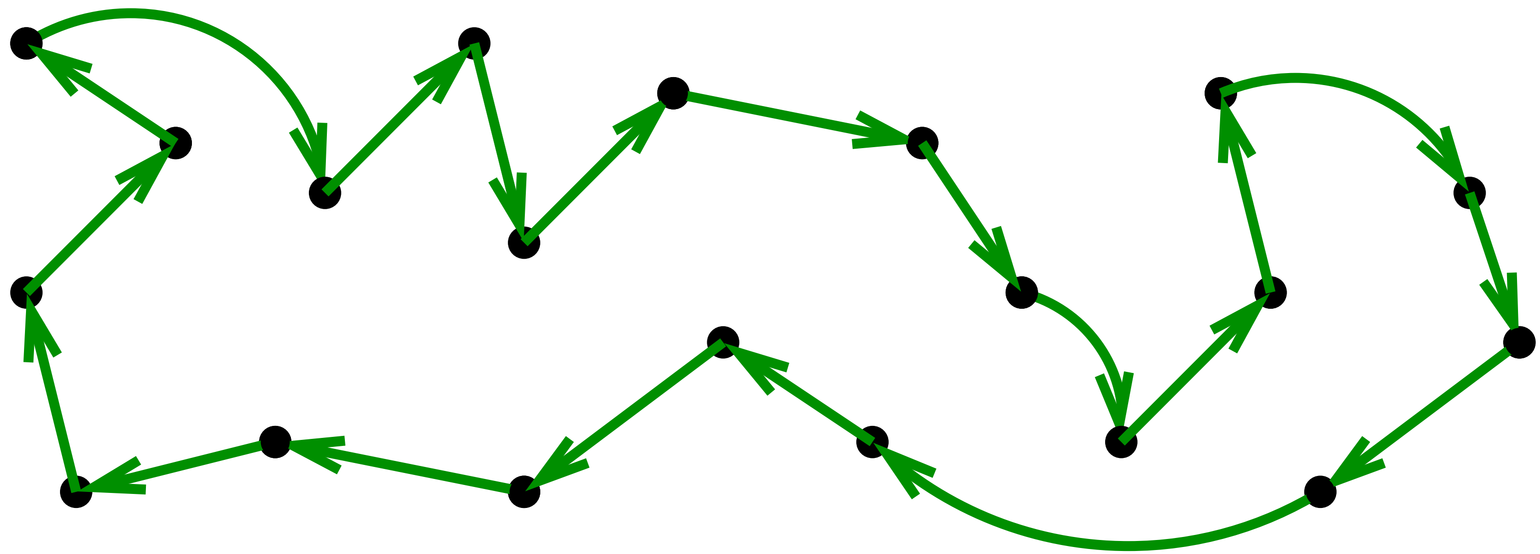
(1) Árvore geradora de comprimento mínimo



(2) Emparelhamento perfeito de comprimento mínimo no subgrafo induzido pelos vértices de grau ímpar (polinomial)
(3) Junte os dois obtendo T' euleriano
(4) Obtenha ciclo euleriano P de T'
(5) Obtenha de P circuito hamiltoniano C

Algoritmo de Christofides

(1) Árvore geradora de comprimento mínimo



(2) Emparelhamento perfeito de comprimento mínimo no subgrafo induzido pelos vértices de grau ímpar (polinomial)
(3) Junte os dois obtendo T' euleriano
(4) Obtenha ciclo euleriano P de T'
(5) Obtenha de P circuito hamiltoniano C
(6) Devolva C

Algoritmo de Christofides

Algoritmo TSPM-Christofides (G, l)

$T \leftarrow$ MST(G, l)
$I \leftarrow$ conjunto de vértices de grau ímpar de T
$M \leftarrow$ EDMONDS$(G[I], l)$
$T' \leftarrow T + M$
$P \leftarrow$ EULER(T')
$C \leftarrow$ ATALHO(P)
devolve C

($G[I]$: subgrafo de G induzido por I)

Algoritmo de Christofides

Algoritmo TSPM-Christofides (G, l)

$T \leftarrow$ MST(G, l)
$I \leftarrow$ conjunto de vértices de grau ímpar de T
$M \leftarrow$ EDMONDS$(G[I], l)$
$T' \leftarrow T + M$
$P \leftarrow$ EULER(T')
$C \leftarrow$ ATALHO(P)
devolve C

($G[I]$: subgrafo de G induzido por I)

Tempo de execução: n^3
(n: o número de vértices de G)

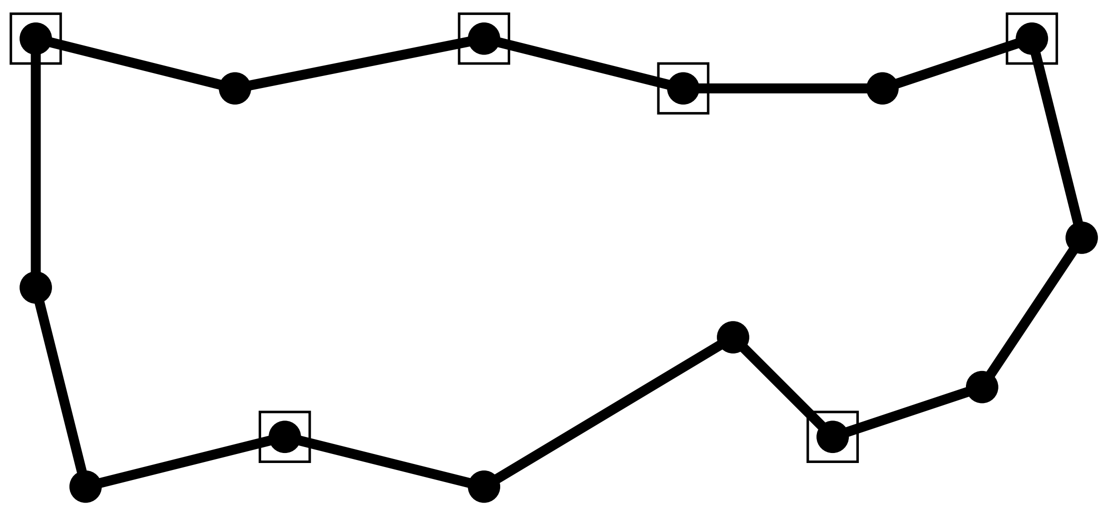
Algoritmo de Christofides

Uma segunda delimitação para OPT: $\mathrm{OPT} \geq 2\,l(M)$
onde M é e. p. de comprimento mínimo em $G[I]$

Algoritmo de Christofides

Uma segunda delimitação para OPT: $\quad \mathrm{OPT} \;\geq\; 2\,l(M)$
onde M é e. p. de comprimento mínimo em $G[I]$

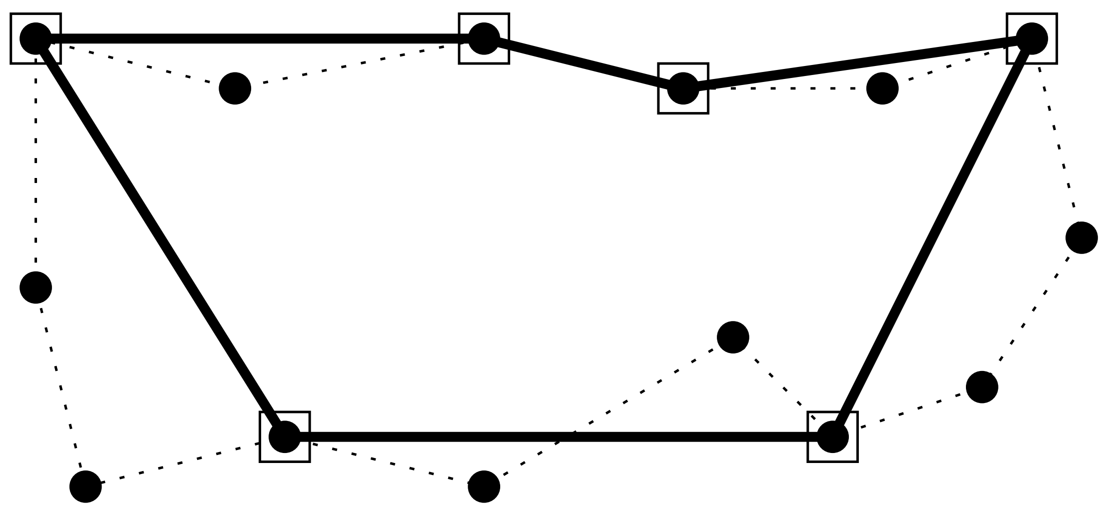
Pr: C^*: circuito hamiltoniano de comprimento mínimo
I: conj. vért. de grau ímpar de T ($|I|$ é par)



Algoritmo de Christofides

Uma segunda delimitação para OPT: $\mathrm{OPT} \geq 2\,l(M)$
onde M é e. p. de comprimento mínimo em $G[I]$

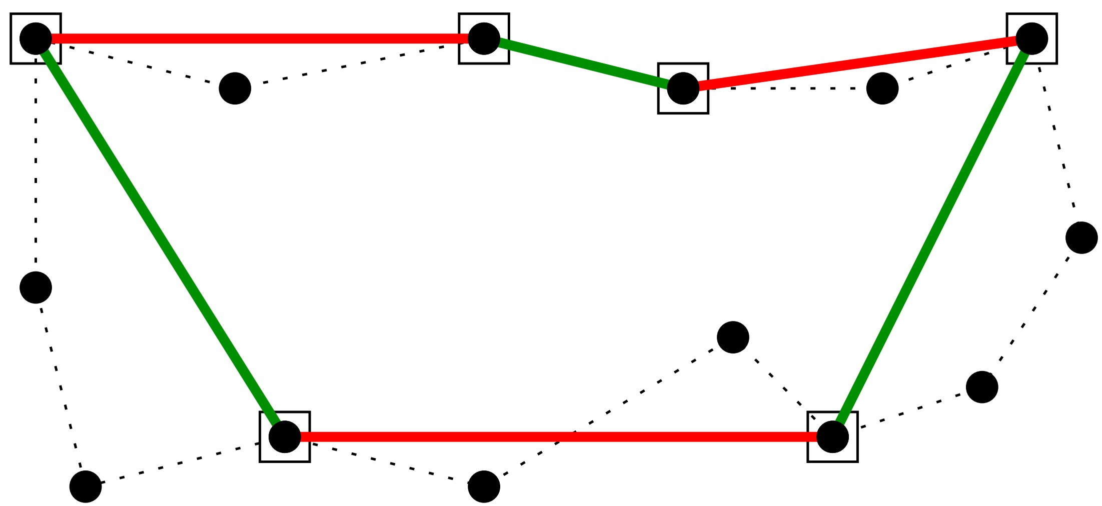
Pr: C^*: circuito hamiltoniano de comprimento mínimo
I: conj. vért. de grau ímpar de T ($|I|$ é par)
C': circuito induzido por C^* em I



Algoritmo de Christofides

Uma segunda delimitação para OPT: $\text{OPT} \geq 2\,l(M)$
onde M é e. p. de comprimento mínimo em $G[I]$

Pr: C^*: circuito hamiltoniano de comprimento mínimo
I: conj. vért. de grau ímpar de T ($|I|$ é par)
C': circuito induzido por C^* em I



C' determina dois e. p. em $G[I]$: M_1 e M_2

Algoritmo de Christofides

Uma segunda delimitação para OPT: $\mathrm{OPT} \geq 2\,l(M)$
onde M é e. p. de comprimento mínimo em $G[I]$.

Prova:

$$\begin{aligned} 2\,l(M) &\leq l(M_1) + l(M_2) \\ &= l(C') \\ &\leq l(C^*) && \text{(pela desigualdade triangular)} \\ &= \mathrm{OPT}\,. \end{aligned}$$

$\square$

Algoritmo de Christofides

Teorema: TSPM-Christofides é uma $1{,}5$-aproximação polinomial para o TSP métrico.

Algoritmo de Christofides

Teorema: TSPM-Christofides é uma $1{,}5$-aproximação polinomial para o TSP métrico.

Prova:

$$
\begin{aligned}
l(C) &\leq l(P) \\
&= l(T') \\
&= l(T) + l(M) \\
&\leq \mathrm{OPT} + \frac{1}{2}\,\mathrm{OPT} \\
&= \frac{3}{2}\,\mathrm{OPT}\,.
\end{aligned}
$$

$\square$

Algoritmo de Christofides

Teorema: TSPM-Christofides é uma $1{,}5$-aproximação polinomial para o TSP métrico.

Prova:

$$\begin{aligned} l(C) &\leq l(P) \\ &= l(T') \\ &= l(T) + l(M) \\ &\leq \mathrm{OPT} + \frac{1}{2}\,\mathrm{OPT} \\ &= \frac{3}{2}\,\mathrm{OPT}\,. \end{aligned}$$

$\square$

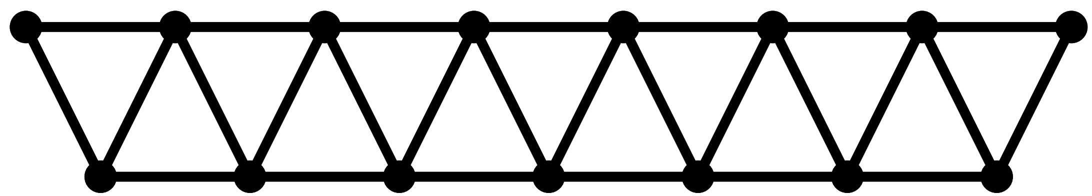
Melhor algoritmo de aproximação conhecido para o TSP métrico.

Algoritmo de Christofides

A análise é justa.

Algoritmo de Christofides

A análise é justa.

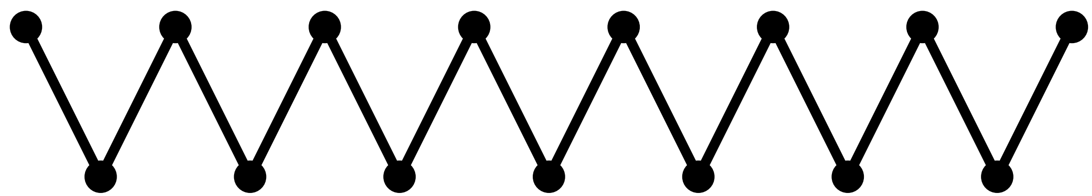


$2n + 1$ vértices

Algoritmo de Christofides

A análise é justa.

Christofides

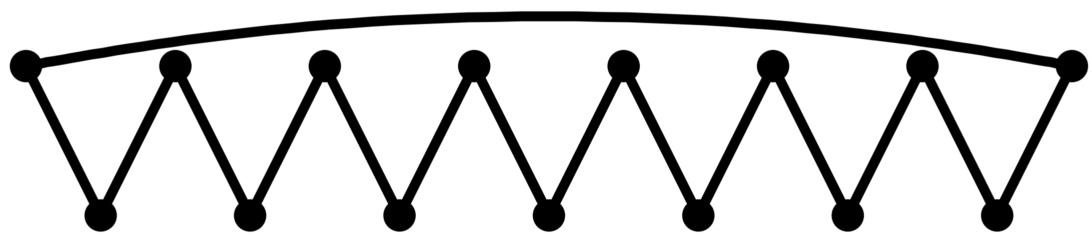


$2n + 1$ vértices

Algoritmo de Christofides

A análise é justa.

Christofides

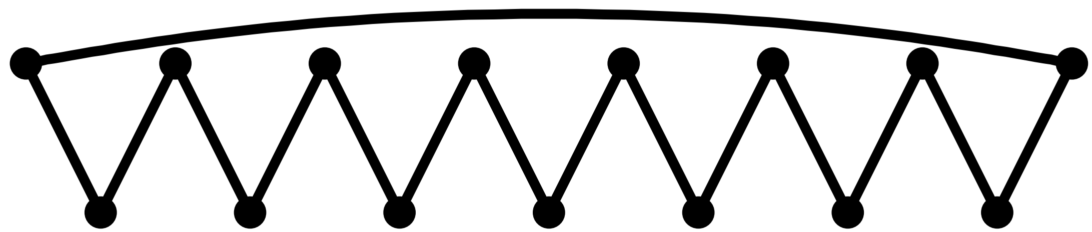


$2n+1$ vértices
Comprimento: $3n$

Algoritmo de Christofides

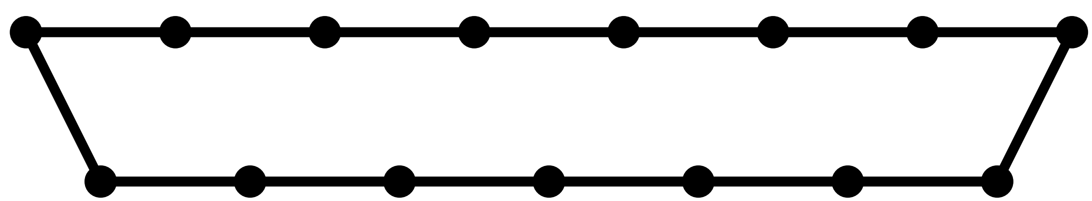
A análise é justa.

Christofides



$2n + 1$ vértices
Comprimento: $3n$

Circuito ótimo



Comprimento: $2n + 1$

TSP Euclideano

PTAS: esquema de aproximação polinomial

$(1+\epsilon)$-aproximação polinomial para todo $\epsilon > 0$ fixo

Arora'96 e Mitchel'96: PTAS para o TSP euclideano

Idéia:

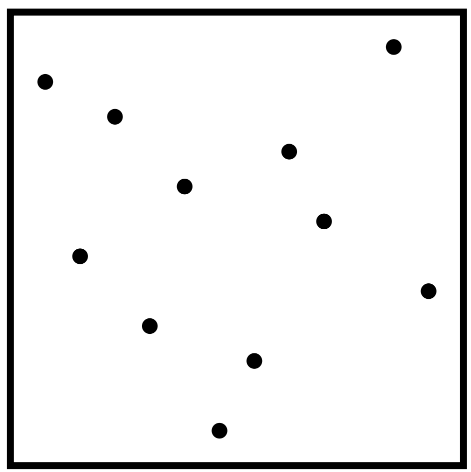
TSP Euclideano

PTAS: esquema de aproximação polinomial

$(1+\epsilon)$-aproximação polinomial para todo $\epsilon > 0$ fixo

Arora'96 e Mitchel'96: PTAS para o TSP euclideano

Idéia:



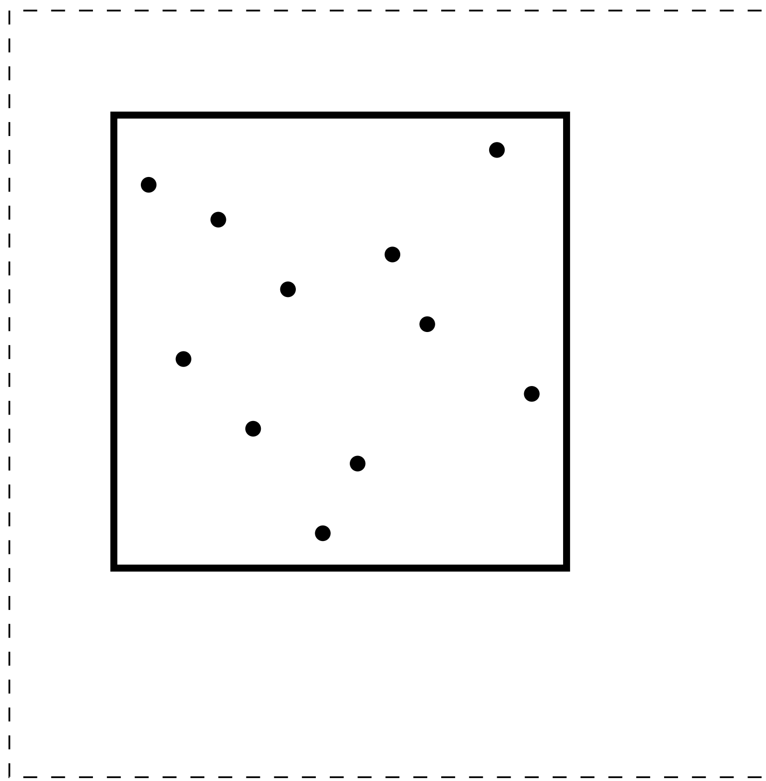
TSP Euclideano

PTAS: esquema de aproximação polinomial

$(1+\epsilon)$-aproximação polinomial para todo $\epsilon > 0$ fixo

Arora'96 e Mitchel'96: PTAS para o TSP euclideano

Idéia:



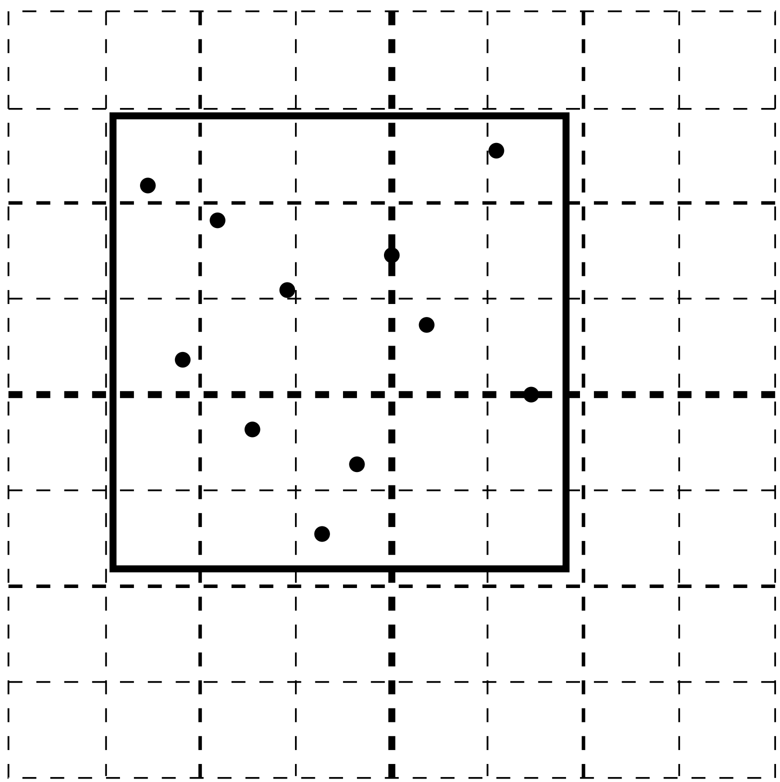
TSP Euclideano

PTAS: esquema de aproximação polinomial

$(1+\epsilon)$-aproximação polinomial para todo $\epsilon > 0$ fixo

Arora'96 e Mitchel'96: PTAS para o TSP euclideano

Idéia:



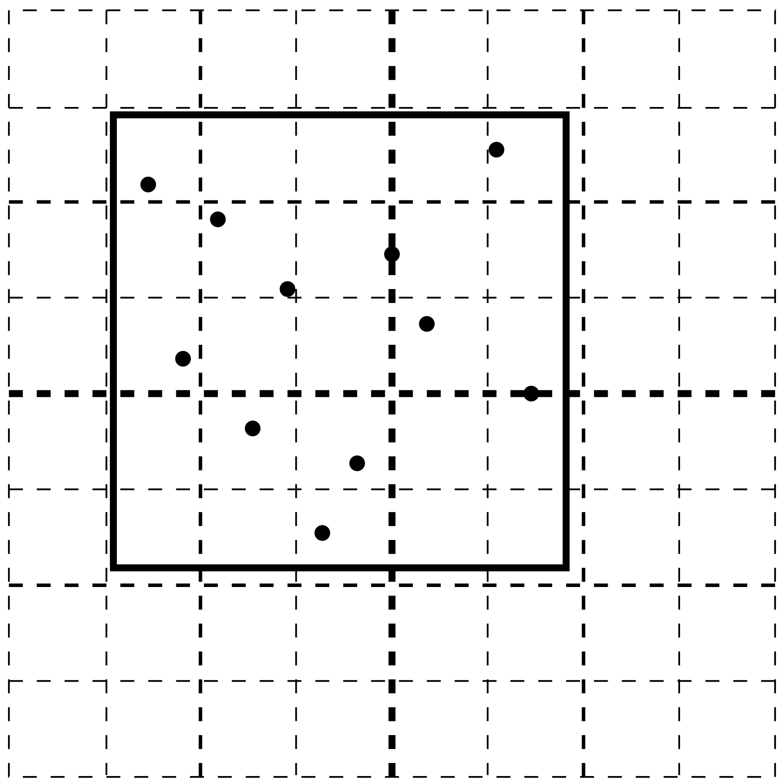
TSP Euclideano

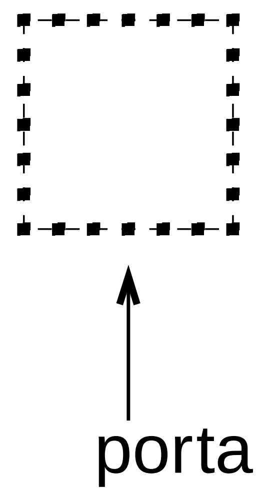
PTAS: esquema de aproximação polinomial

$(1+\epsilon)$-aproximação polinomial para todo $\epsilon > 0$ fixo

Arora'96 e Mitchel'96: PTAS para o TSP euclideano

Idéia:





TSP Euclideano

PTAS: esquema de aproximação polinomial

$(1+\epsilon)$-aproximação polinomial para todo $\epsilon > 0$ fixo

Arora'96 e Mitchel'96: PTAS para o TSP euclideano

Idéia:

Circuito que respeita portal: entra e sai dos quadrados da dissecção através de portais.

Algoritmo: Encontra um circuito hamiltoniano mais curto que respeita os portais por programação dinâmica.

Tal circuito está tão próximo quando se queira de um TSP tour.

Resultado de Inaproximabilidade

$\alpha : \mathbb{N} \rightarrow \mathbb{N}$ função polinomialmente computável

Teorema (Sahni e Gonzalez '76): Se existir $\alpha(n)$-aproximação polinomial p/o TSP(G, l), onde $n := |V_G|$, então P=NP.

Resultado de Inaproximabilidade

$\alpha : \mathbb{N} \rightarrow \mathbb{N}$ função polinomialmente computável

Teorema (Sahni e Gonzalez '76): Se existir $\alpha(n)$-aproximação polinomial p/o TSP(G, l), onde $n := |V_G|$, então P=NP.

Problema HC: Dado G, decidir se G tem ou não um circuito hamiltoniano.

- NP-completo [K72]

Resultado de Inaproximabilidade

Teorema (Sahni e Gonzalez '76): Se existir $\alpha(n)$-aproximação polinomial p/o TSP(G, l), onde $n := |V_G|$, então P=NP.

Pr:

$$\alpha(n)\text{-aproximação polinomial p/o TSP}$$
$$\Downarrow$$
$$\text{algoritmo polinomial p/o HC}$$

Resultado de Inaproximabilidade

Teorema (Sahni e Gonzalez '76): Se existir $\alpha(n)$-aproximação polinomial p/o TSP(G, l), onde $n := |V_G|$, então P=NP.

Pr:

$$\alpha(n)\text{-aproximação polinomial p/o TSP} \quad \leftarrow A$$

$$\Downarrow$$

$$\text{algoritmo polinomial p/o HC} \quad \leftarrow B$$

Resultado de Inaproximabilidade

Teorema (Sahni e Gonzalez '76): Se existir $\alpha(n)$-aproximação polinomial p/o TSP(G, l), onde $n := |V_G|$, então P=NP.

Pr:

$$\begin{array}{c} \alpha(n)\text{-aproximação polinomial p/o TSP} \quad \leftarrow A \\ \Downarrow \\ \text{algoritmo polinomial p/o HC} \quad \leftarrow B \end{array}$$

Algoritmo B (G)

$H \leftarrow$ grafo completo em V_G

para cada e em E_G faça $l_e \leftarrow 1$

para cada e em $E_H \setminus E_G$ faça $l_e \leftarrow \alpha(|V_G|)|V_G| + 1$

$C \leftarrow A(H, l)$

se $l(C) \leq \alpha(|V_G|)|V_G|$

então devolva "Sim"

então devolva "Não"

Resultado de Inaproximabilidade

A polinomial $\Rightarrow$ B polinomial

Resultado de Inaproximabilidade

A polinomial $\Rightarrow$ B polinomial

se existe circuito hamiltoniano em G,
então $\mathrm{OPT} = |V_G|$ e

$$l(C) \;\leq\; \alpha(|V_G|)\,\mathrm{OPT} \;=\; \alpha(|V_G|)|V_G|.$$

Resultado de Inaproximabilidade

A polinomial $\Rightarrow$ B polinomial

se existe circuito hamiltoniano em G,
então $\mathrm{OPT} = |V_G|$ e

$$l(C) \ \leq \ \alpha(|V_G|)\,\mathrm{OPT} \ = \ \alpha(|V_G|)|V_G|.$$

se não existe circuito hamiltoniano em G,
então

$$l(C) \ \geq \ \alpha(|V_G|)|V_G| + 1$$

pois qq circ. hamilt. usa $e \notin E_G$
(i.e., $l_e \ = \ \alpha(|V_G|)|V_G| + 1$).

$\square$

Para completar...