Algoritmos de Aproximação e Problemas de Clustering

Trabalho de Formatura Supervisionado

Samuel Plaça de Paula
Supervisora: Profª Cristina Gomes Fernandes

Novembro de 2012

NP-completude

Problemas NP-completos

Admitem solução polinomial se, e só se, P = NP.

NP-completude

Problemas NP-difíceis

"Pelo menos tão difíceis" quanto qualquer problema NP-completo.

NP-completude

Problemas NP-difíceis

"Pelo menos tão difíceis" quanto qualquer problema NP-completo.

Ou seja, não admitem algoritmo polinomial a menos que P = NP.

NP-completude

Problemas NP-difíceis

"Pelo menos tão difíceis" quanto qualquer problema NP-completo.

Ou seja, não admitem algoritmo polinomial a menos que P = NP.

E daí?

Otimização Combinatória

Otimização combinatória nos permite modelar vários problemas reais.

Otimização Combinatória

Otimização combinatória nos permite modelar vários problemas reais.

Exemplos: caminho mínimo, árvore geradora de custo mínimo.

Otimização Combinatória

Otimização combinatória nos permite modelar vários problemas reais.

Exemplos: caminho mínimo, árvore geradora de custo mínimo.

Mas...

Há muitos problemas interessantes de Otimização Combinatória que são NP-difíceis.

Otimização combinatória

Problema de otimização combinatória

Conjunto de instâncias.

Otimização combinatória

Problema de otimização combinatória

Conjunto de instâncias.

Para instância I, conjunto de soluções viáveis $\mathrm{Sol}(I)$.

Otimização combinatória

Problema de otimização combinatória

Conjunto de instâncias.

Para instância I, conjunto de soluções viáveis $\mathrm{Sol}(I)$.

Para cada solução $S \in \mathrm{Sol}(I)$, valor $\mathrm{val}\,(S)$.

Otimização combinatória

Problema:

Dada uma instância I, encontrar S^* que atinja o **máximo** (ou **mínimo**) valor.

Otimização combinatória

Problema:

Dada uma instância I, encontrar S^* que atinja o **máximo** (ou **mínimo**) valor.

S^*: solução ótima
$\mathrm{val}\,(S^*)$: valor ótimo, ou $\mathrm{OPT}(I)$.

Problemas NP-difíceis

Programação Linear Inteira

Programação Linear é fácil.

Problemas NP-difíceis

Programação Linear Inteira

Programação Linear é fácil. (No sentido computacional, claro...)

Problemas NP-difíceis

Programação Linear Inteira

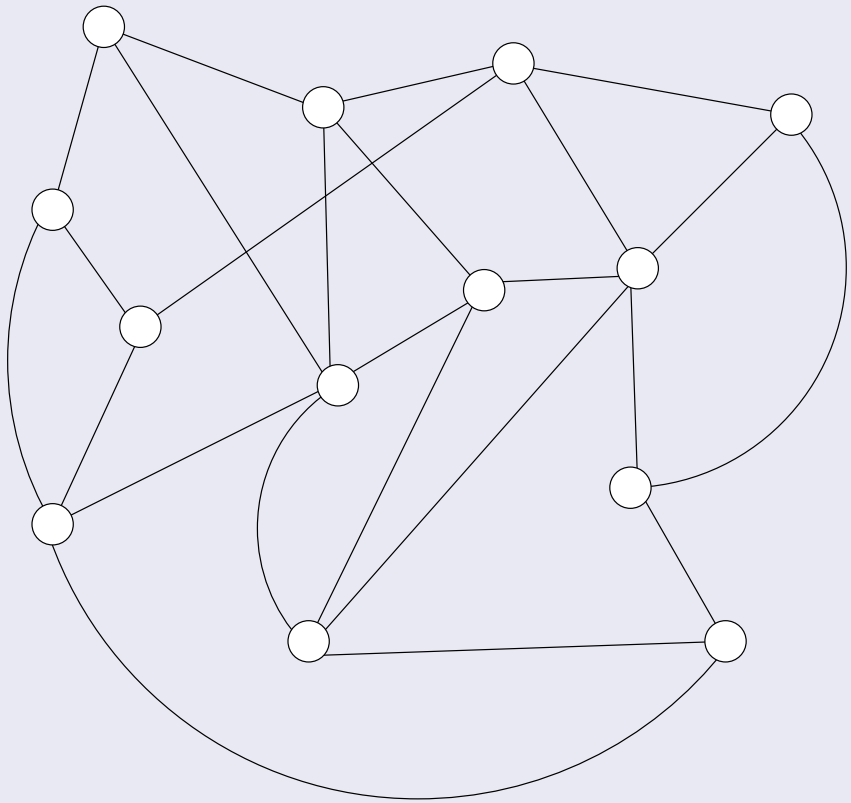
Programação Linear é fácil. (No sentido computacional, claro...)

Restrições de integralidade tornam o problema NP-difícil!

Problemas NP-difíceis

Caixeiro Viajante (TSP)

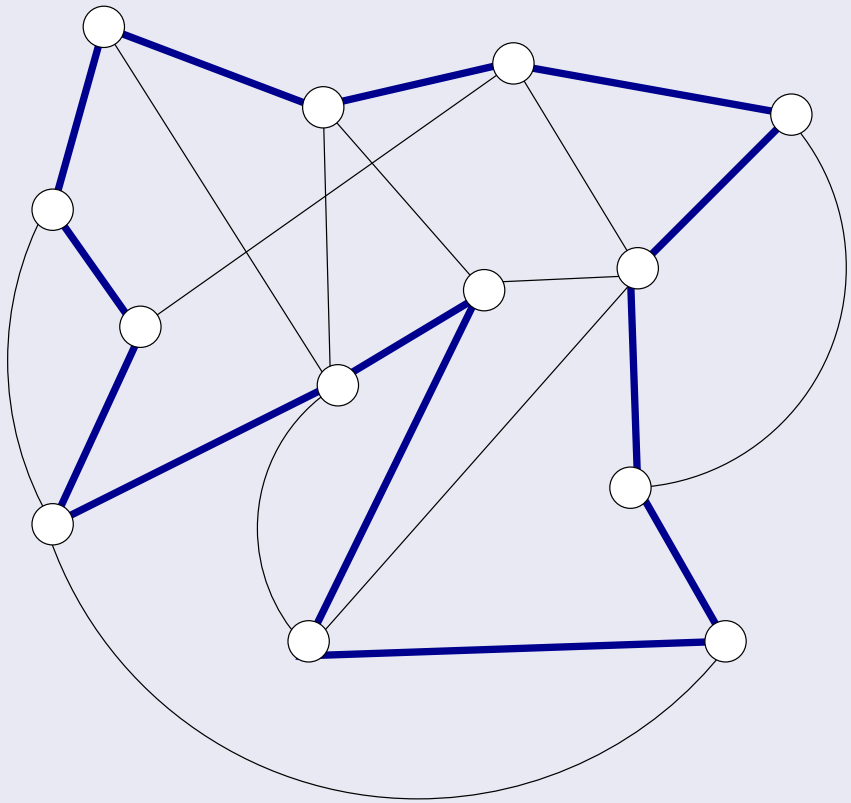
Encontrar circuito hamiltoniano de custo mínimo num grafo com custos nas arestas.



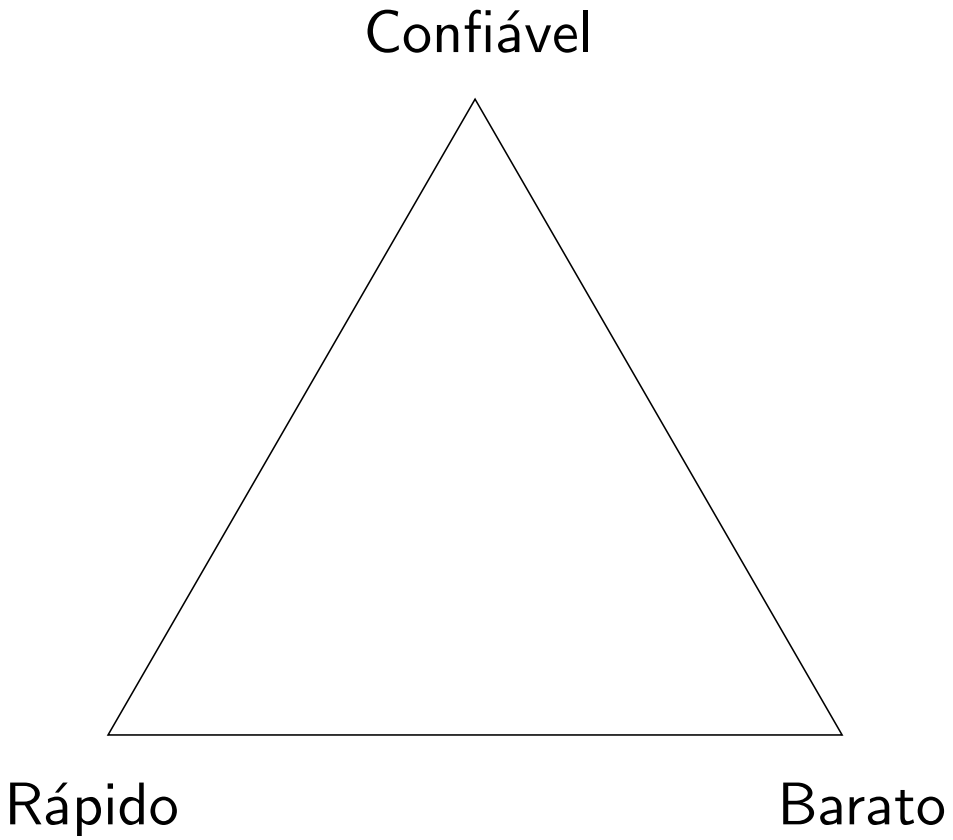
Problemas NP-difíceis

Caixeiro Viajante (TSP)

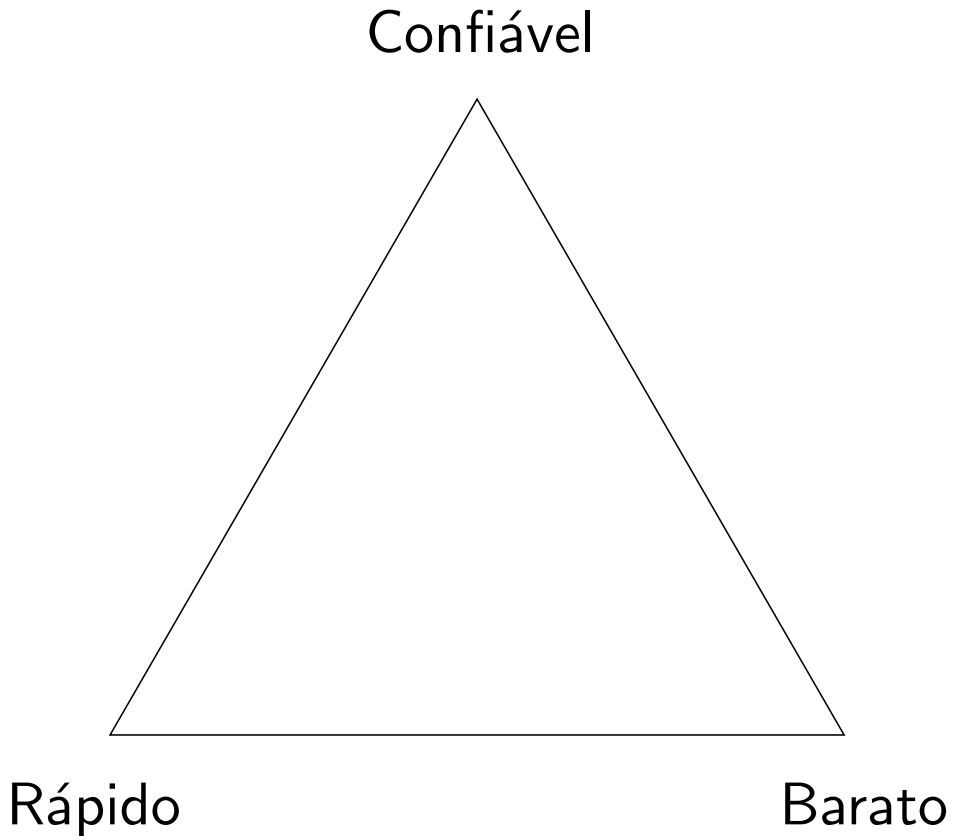
Encontrar circuito hamiltoniano de custo mínimo num grafo com custos nas arestas.



Escolhas...

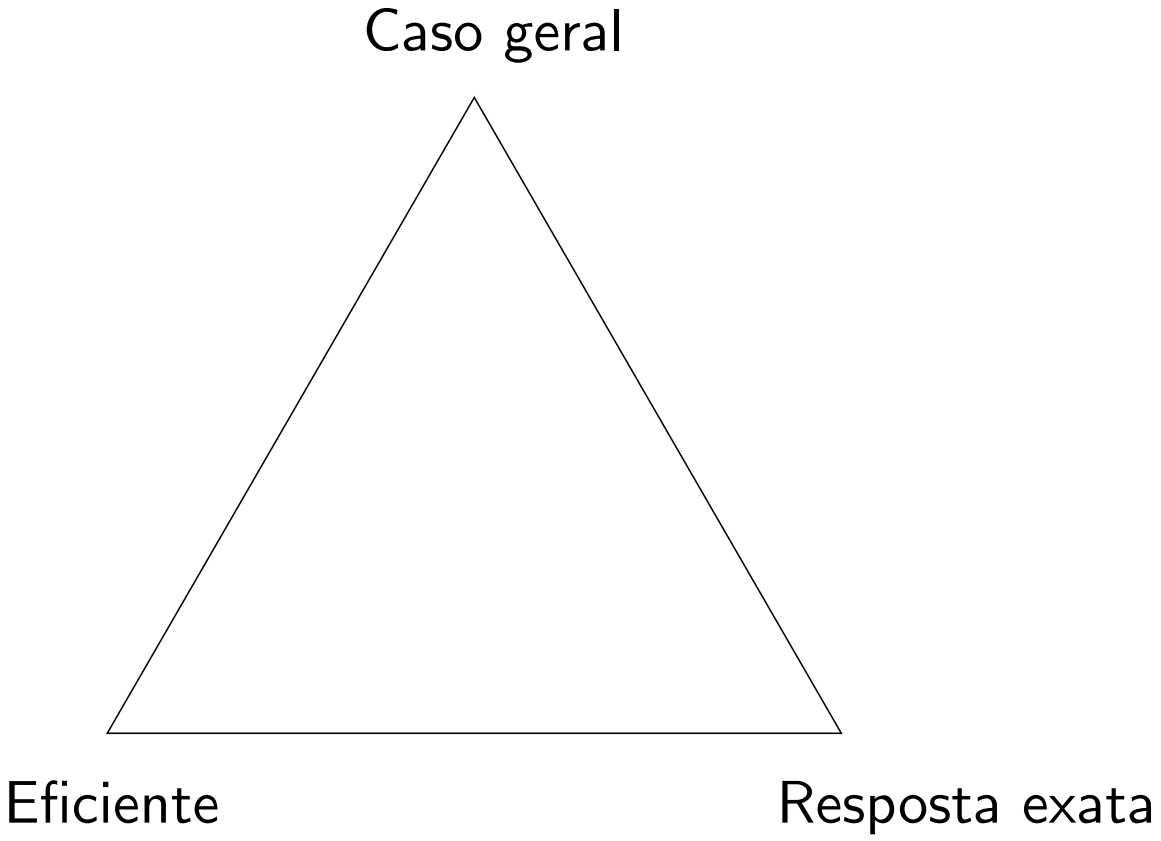


Escolhas...

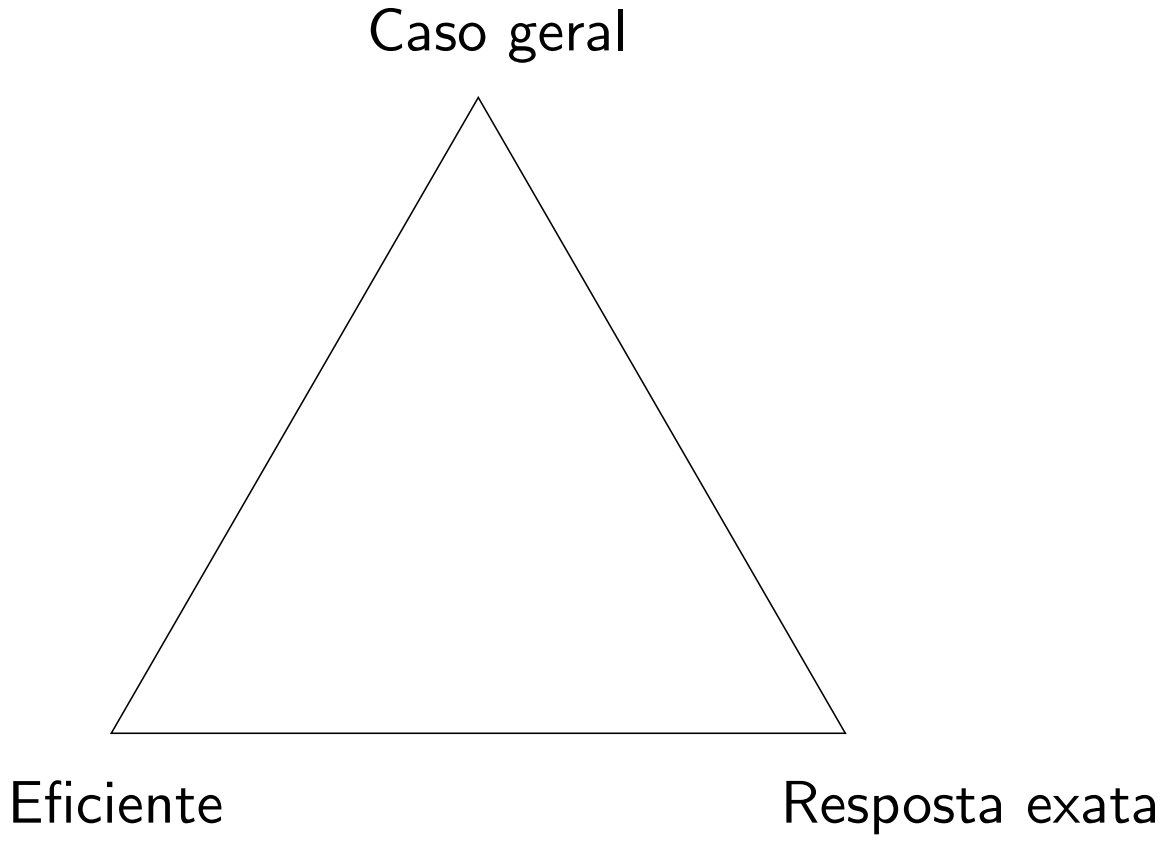


Ecolha dois!

Para problemas NP-difíceis...

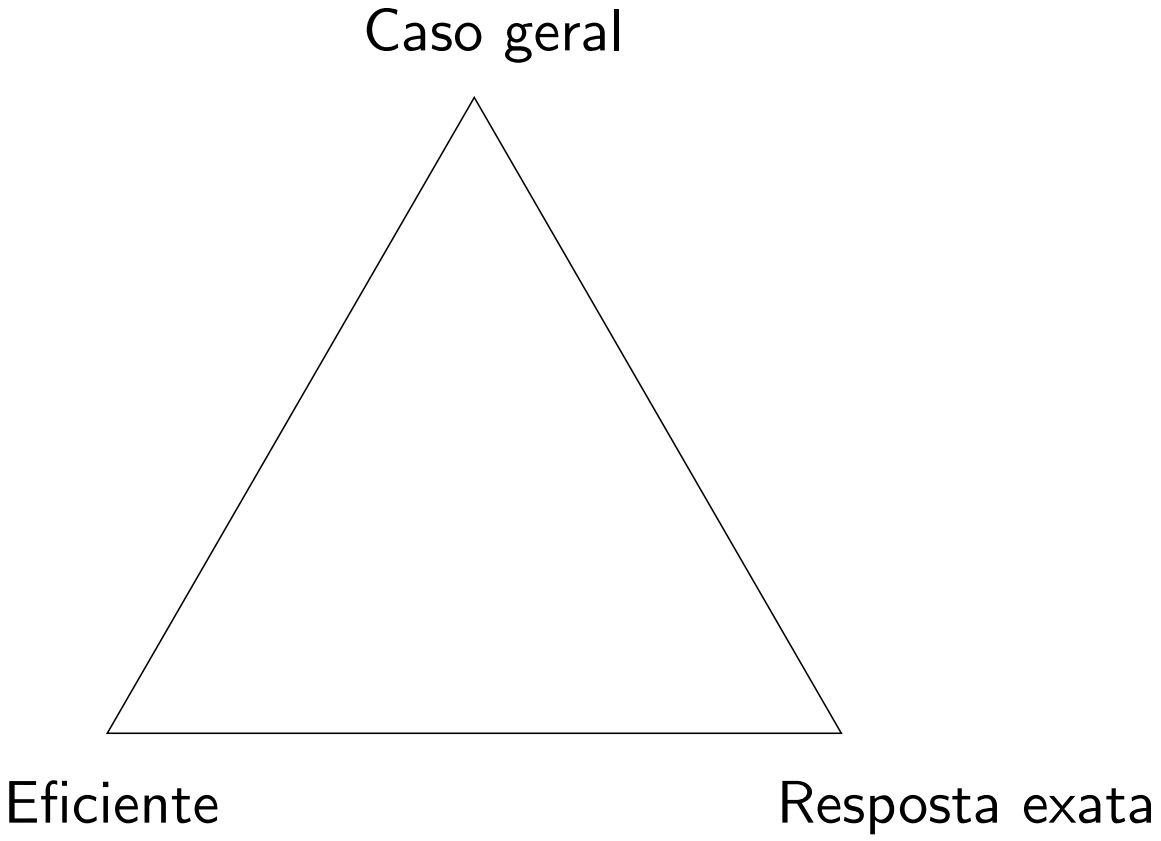


Para problemas NP-difíceis...



Ecolha dois!

Para problemas NP-difíceis...



Ecolha dois! ...ou prove que P = NP.

Abordagens

- Resolver caso geral
- Resolver em tempo polinomial
- Obter sempre a resposta exata

Abordagens

- ~~Resolver caso geral~~
- Resolver em tempo polinomial
- Obter sempre a resposta exata

Resolver casos especiais.

Abordagens

- Resolver caso geral
- ~~Resolver em tempo polinomial~~
- Obter sempre a resposta exata

Solvers de SAT e Programação Linear Inteira.

Abordagens

- Resolver caso geral
- Resolver em tempo polinomial
- ~~Obter sempre a resposta exata~~

Algoritmos de Aproximação!

Algoritmo de aproximação

Idéia

Uma α-aproximação para um problema é um algoritmo **eficiente** que devolve soluções sempre a um fator α do valor ótimo.

Algoritmo de aproximação

Mais precisamente...

Para um problema de **minimização**, uma α-aproximação é um algoritmo de tempo polinomial que, para qualquer instância I do problema, devolve uma solução $S \in \mathrm{Sol}(I)$ tal que:

$$\mathrm{val}\,(S) \leq \alpha \cdot \mathsf{OPT}(I).$$

Algoritmo de aproximação

Mais precisamente...

Para um problema de **minimização**, uma α-aproximação é um algoritmo de tempo polinomial que, para qualquer instância I do problema, devolve uma solução $S \in \mathrm{Sol}(I)$ tal que:

$$\mathrm{val}\,(S) \leq \alpha \cdot \mathsf{OPT}(I).$$

Nesse caso, $\alpha > 1$.

Algoritmo de aproximação

Mais precisamente...

Para um problema de **minimização**, uma α-aproximação é um algoritmo de tempo polinomial que, para qualquer instância I do problema, devolve uma solução $S \in \mathrm{Sol}(I)$ tal que:

$$\mathrm{val}\,(S) \leq \alpha \cdot \mathsf{OPT}(I).$$

Nesse caso, $\alpha > 1$.

A definição para problemas de maximização é análoga.

Aproximando o Caixeiro Viajante

TSP tem vários casos...

- Caso geral
- Caso métrico
- Caso euclideano

Caso geral do Caixeiro Viajante

Grafo qualquer, não direcionado, com custos nas arestas. Podemos supor grafo completo sem perda de generalidade.

Caso geral do Caixeiro Viajante

Grafo qualquer, não direcionado, com custos nas arestas. Podemos supor grafo completo sem perda de generalidade.

Muito difícil de aproximar :-(

Caso geral do Caixeiro Viajante

Grafo qualquer, não direcionado, com custos nas arestas. Podemos supor grafo completo sem perda de generalidade.

Muito difícil de aproximar :-(

α-aproximação é NP-difícil para α constante.

Caso geral do Caixeiro Viajante

Grafo qualquer, não direcionado, com custos nas arestas. Podemos supor grafo completo sem perda de generalidade.

Muito difícil de aproximar :-(

α-aproximação é NP-difícil para α constante.
De fato, para $\alpha \in O(2^n)$, onde n é o número de vértices!

Caso geral do Caixeiro Viajante

Grafo qualquer, não direcionado, com custos nas arestas. Podemos supor grafo completo sem perda de generalidade.

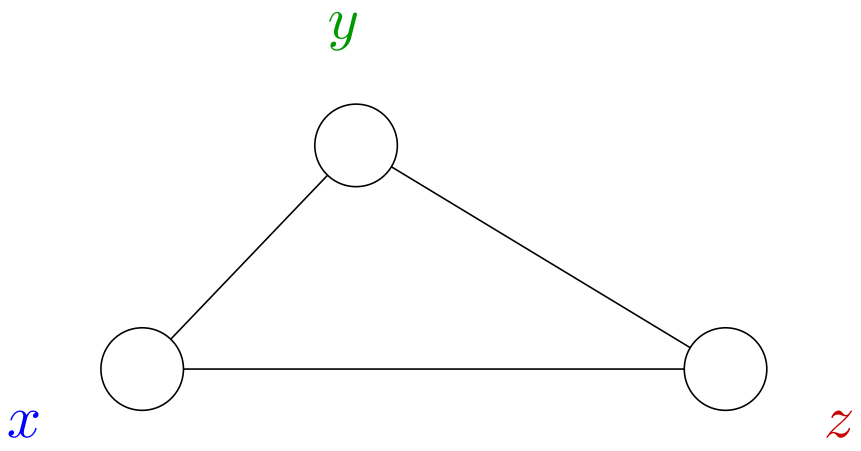
Muito difícil de aproximar :-(

α-aproximação é NP-difícil para α constante.
De fato, para $\alpha \in O(2^n)$, onde n é o número de vértices!

...mas nem tudo está perdido!

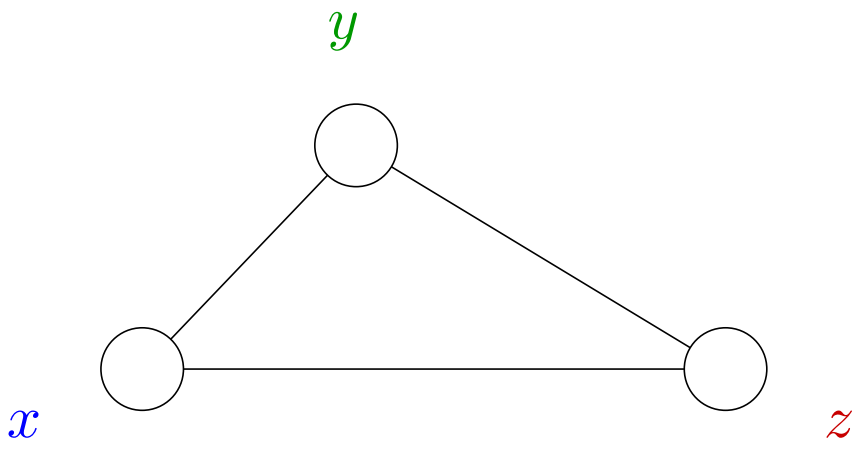
Caixeiro Viajante, caso métrico

Grafo completo, custos respeitando a **desigualdade triangular**.



Caixeiro Viajante, caso métrico

Grafo completo, custos respeitando a **desigualdade triangular**.



Dados três vértices diferentes x, y e z, sempre vale que

$$d(x, z) \leq d(x, y) + d(y, z)$$

Caixeiro Viajante, caso métrico

O caso métrico é, de certa forma, o caso que em geral nos interessa!

Caixeiro Viajante, caso métrico

O caso métrico é, de certa forma, o caso que em geral nos interessa!

(Pense nas distâncias ou custos dos problemas reais.)

Caixeiro Viajante, caso métrico

O caso métrico é, de certa forma, o caso que em geral nos interessa!

(Pense nas distâncias ou custos dos problemas reais.)

Ao mesmo tempo, ainda é um caso bem genérico, e portanto versátil.

Caixeiro Viajante, caso métrico

O caso métrico é, de certa forma, o caso que em geral nos interessa!

(Pense nas distâncias ou custos dos problemas reais.)

Ao mesmo tempo, ainda é um caso bem genérico, e portanto versátil.

Conhecemos uma $\frac{3}{2}$-aproximação para o Caixeiro Viajante métrico (Algoritmo de Christofides).

Caixeiro Viajante euclideano

Os vértices do grafo são pontos em um espaço euclideano, o grafo é completo e os custos das arestas são distâncias euclideanas.

Caixeiro Viajante euclideano

Os vértices do grafo são pontos em um espaço euclideano, o grafo é completo e os custos das arestas são distâncias euclideanas.

Por exemplo, temos uma coleção de pontos no $\mathbb{R}^2$ e suas distâncias em linha reta.

Caixeiro Viajante euclideano

Os vértices do grafo são pontos em um espaço euclideano, o grafo é completo e os custos das arestas são distâncias euclideanas.

Por exemplo, temos uma coleção de pontos no $\mathbb{R}^2$ e suas distâncias em linha reta.

Nesse caso, para qualquer $\varepsilon > 0$, temos uma $(1 + \varepsilon)$-aproximação.

Problemas de Clustering

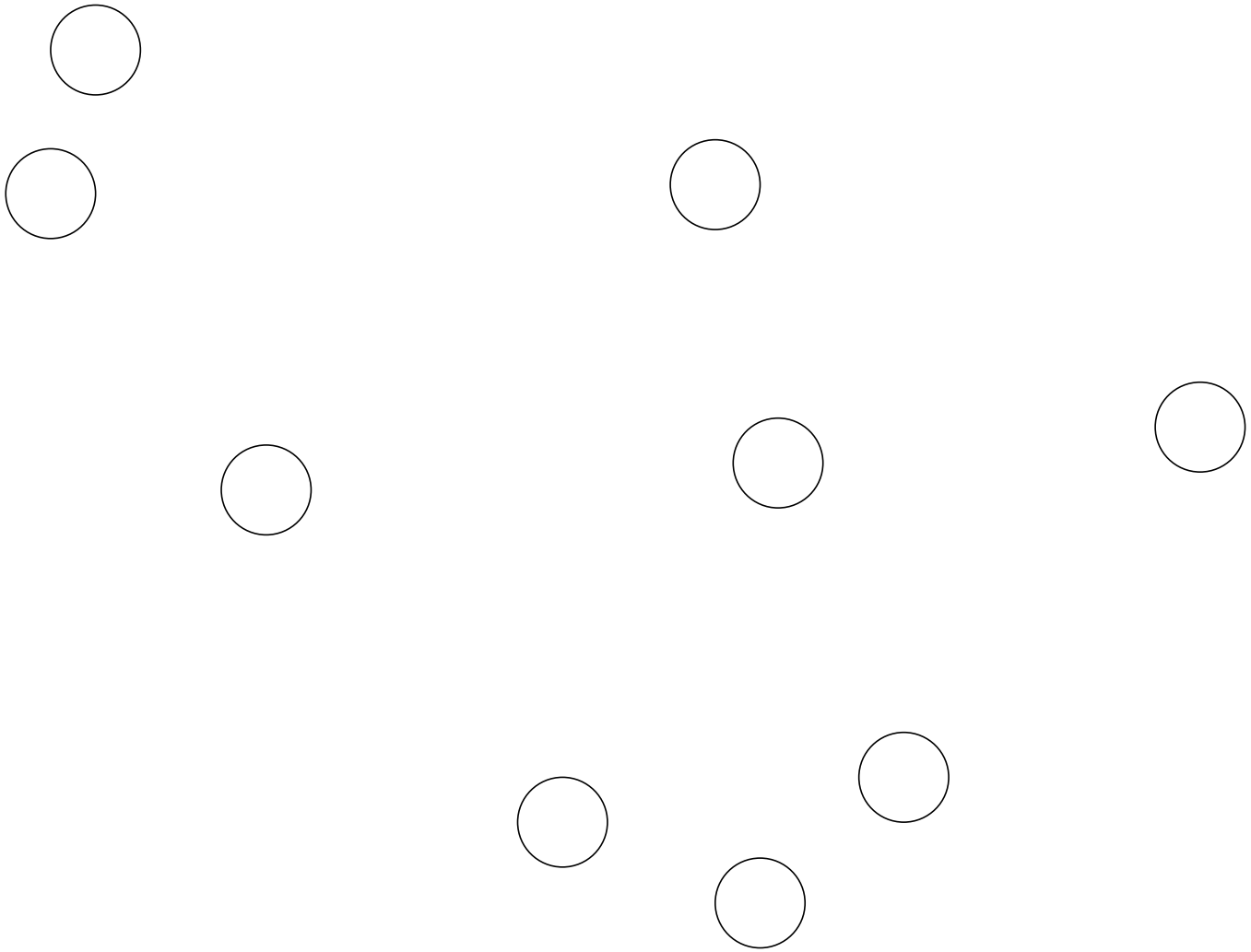
Dado um conjunto de elementos comparáveis entre si (via função distância ou correlação), particionar o conjunto de alguma forma.

Problemas de Clustering

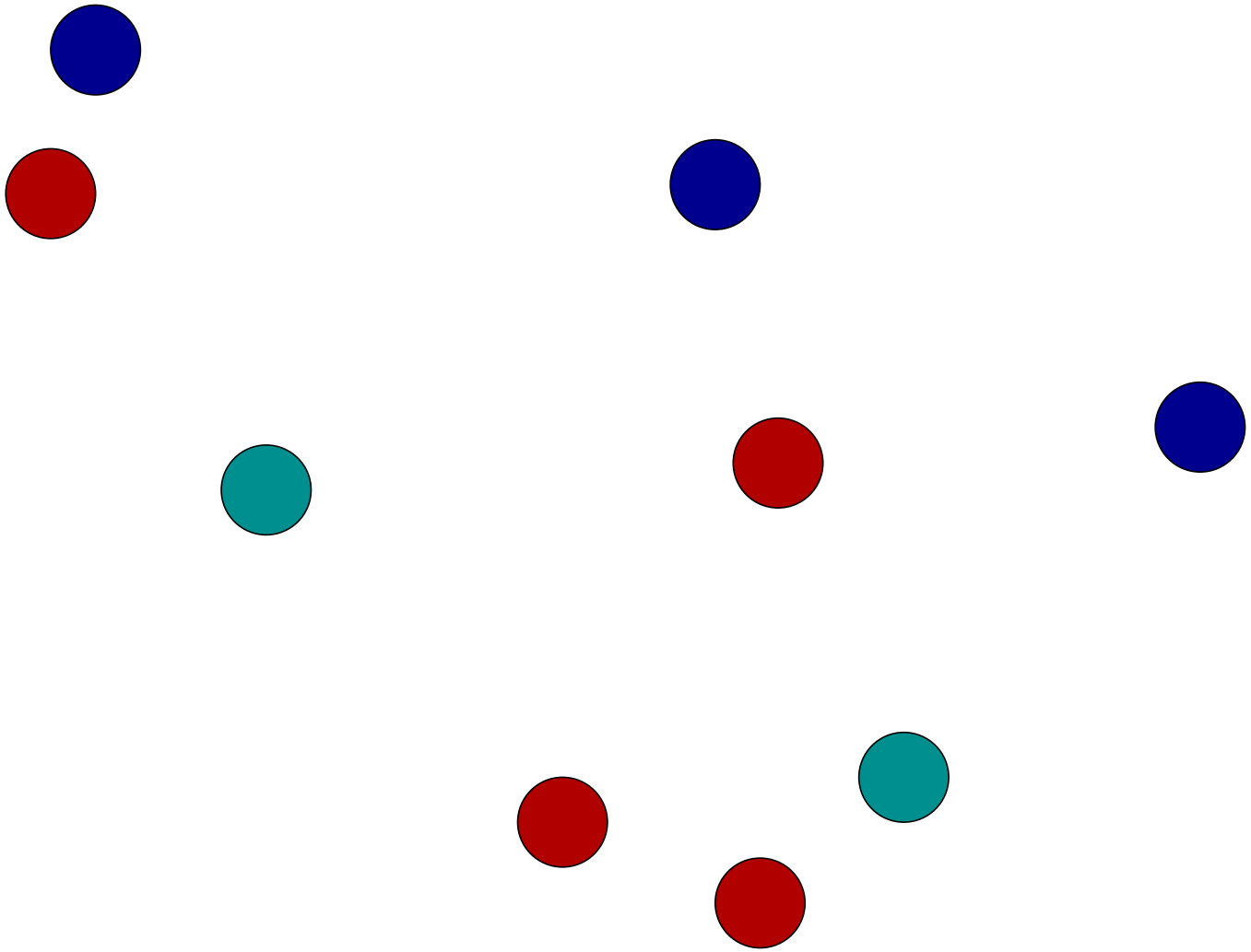
Dado um conjunto de elementos comparáveis entre si (via função distância ou correlação), particionar o conjunto de alguma forma.

Um bom particionamento (clustering) é um que tenha conjuntos (clusters) homogêneos.

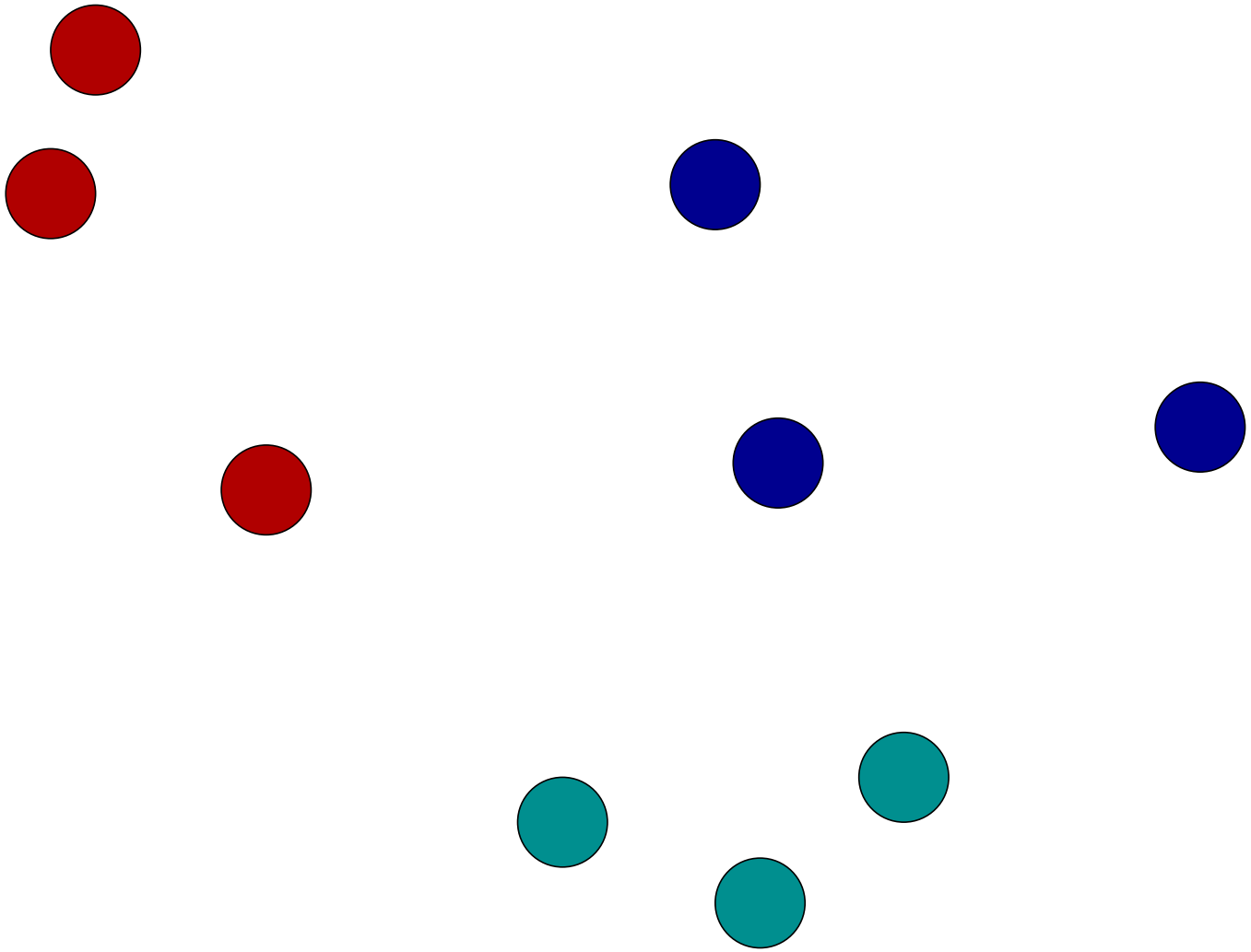
Conjunto de Pontos



Clustering ruim



Clustering "melhor"



Interpretações do clustering

Problemas de clustering têm várias interpretações, por exemplo:

Interpretações do clustering

Problemas de clustering têm várias interpretações, por exemplo:

- Categorização automática

Interpretações do clustering

Problemas de clustering têm várias interpretações, por exemplo:

- Categorização automática
- *Facility location*

O k-center

Definição do k-center

- Grafo $G = (V, E)$ com custos nas arestas
- Inteiro k

O k-center

Definição do k-center

- Grafo $G = (V, E)$ com custos nas arestas
- Inteiro k

Escolher k vértices (centros) de modo a minimizar a maior distância de um vértice ao centro mais próximo.

Algoritmo para o k-*center*

Algoritmo para o k-*center*

- Ordene as arestas de acordo com os custos, de modo que

$$c_1 \leq c_2 \leq \cdots \leq c_m.$$

Algoritmo para o k-*center*

- Ordene as arestas de acordo com os custos, de modo que

$$c_1 \leq c_2 \leq \cdots \leq c_m.$$

- Para cada i de 1 até m, construa o grafo G_i (o grafo G excluindo-se as arestas de custo maior que c_i.

Algoritmo para o k-*center*

- Ordene as arestas de acordo com os custos, de modo que

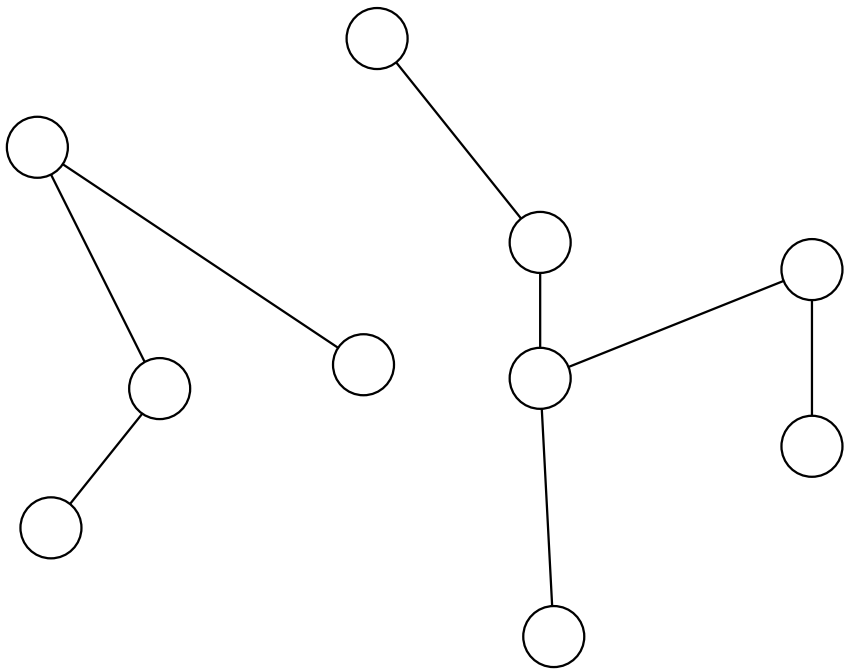
$$c_1 \leq c_2 \leq \cdots \leq c_m.$$

- Para cada i de 1 até m, construa o grafo G_i (o grafo G excluindo-se as arestas de custo maior que c_i.

- Use uma rotina que, dado G_i, nos diz que $OPT > c_i$ OU devolve uma solução de custo $\leq 2 \cdot c_i$.

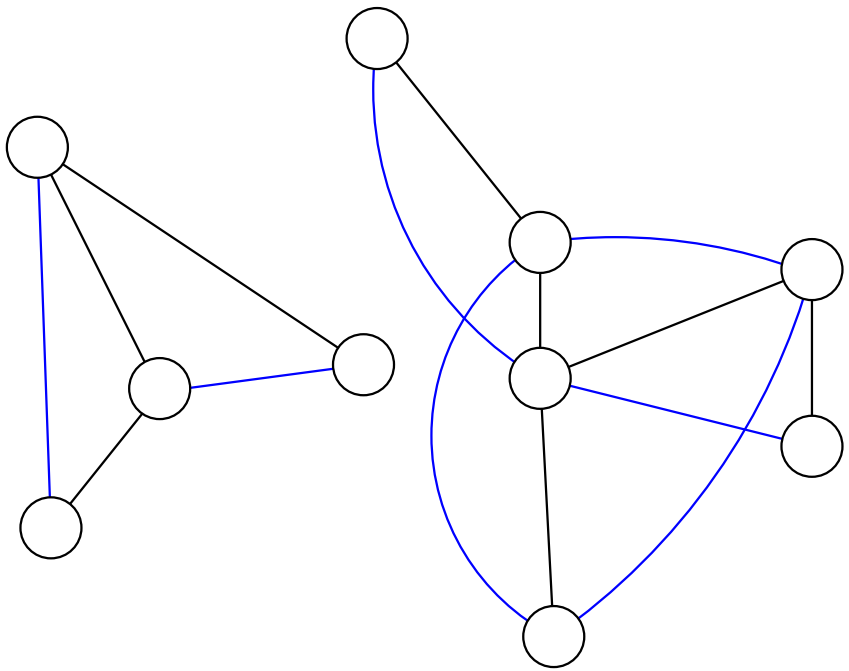
Rotina de decisão

Dado G_i, constrói-se G_i^2, em que vértices são vizinhos se estão a distância no máximo 2 em G_i.



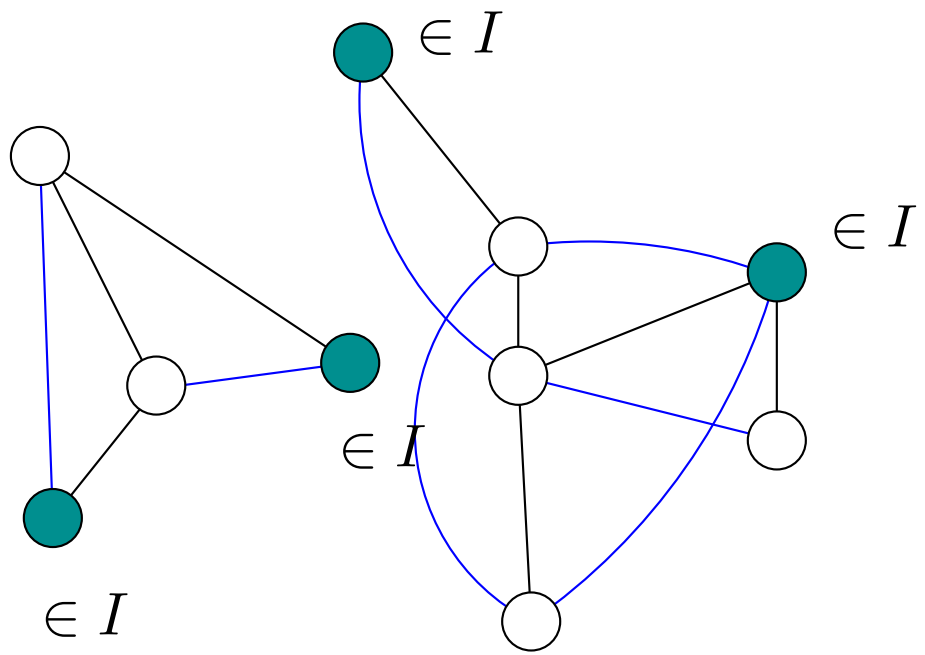
Rotina de decisão

Dado G_i, constrói-se G_i^2, em que vértices são vizinhos se estão a distância no máximo 2 em G_i.



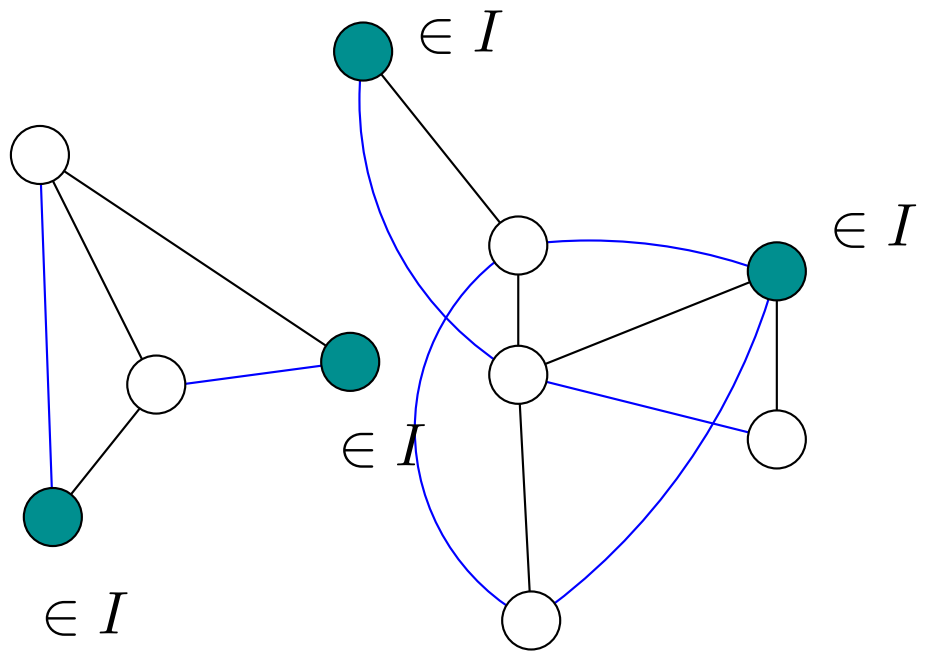
Rotina de decisão

Em G_i^2, obtém-se um conjunto independente maximal I.



Rotina de decisão

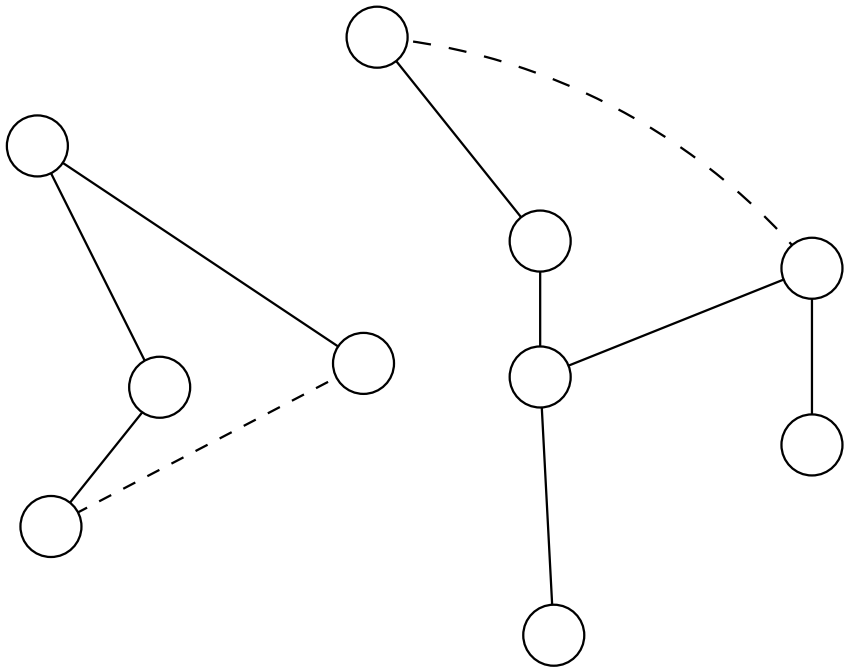
Em G_i^2, obtém-se um conjunto independente maximal I.



Uma solução com valor $\leq c_i$ precisa usar pelo menos $|I|$ centros.

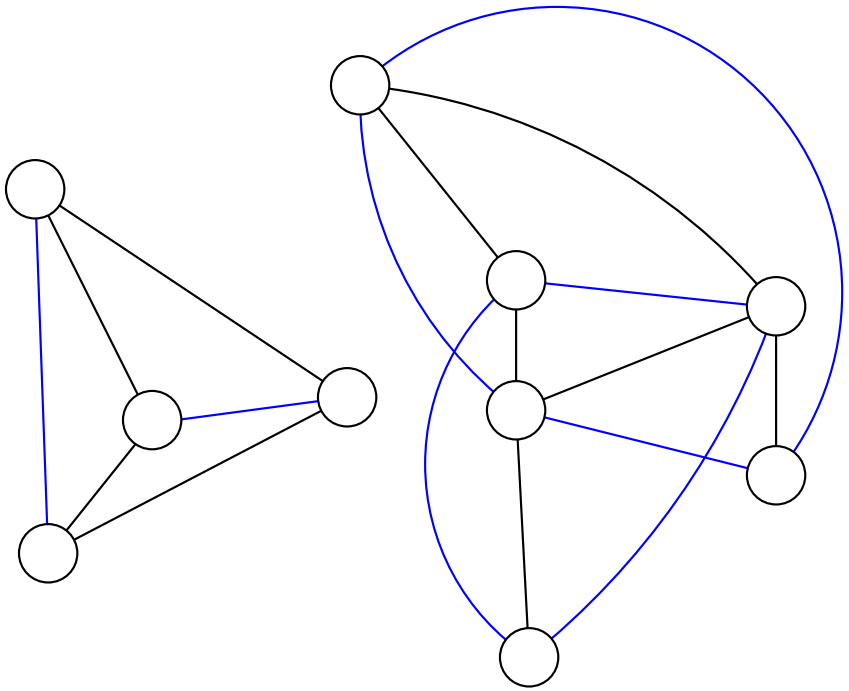
Rotina de decisão, próximo passo

Incrementamos o i, e agora o grafo G_i inclui mais arestas:



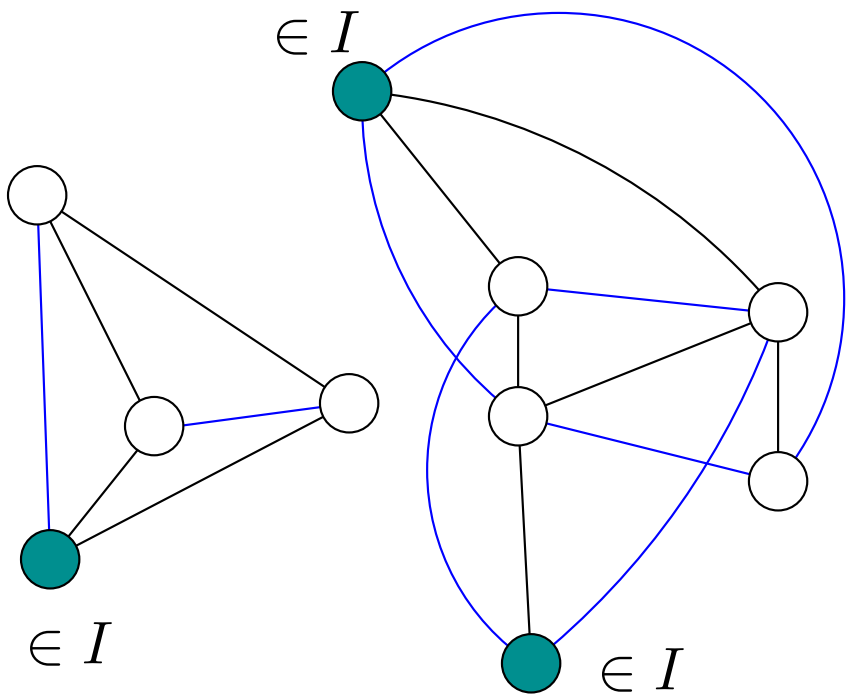
Construindo uma solução

Construímos G_i^2 e obtemos um novo conjunto independente maximal.



Construindo uma solução

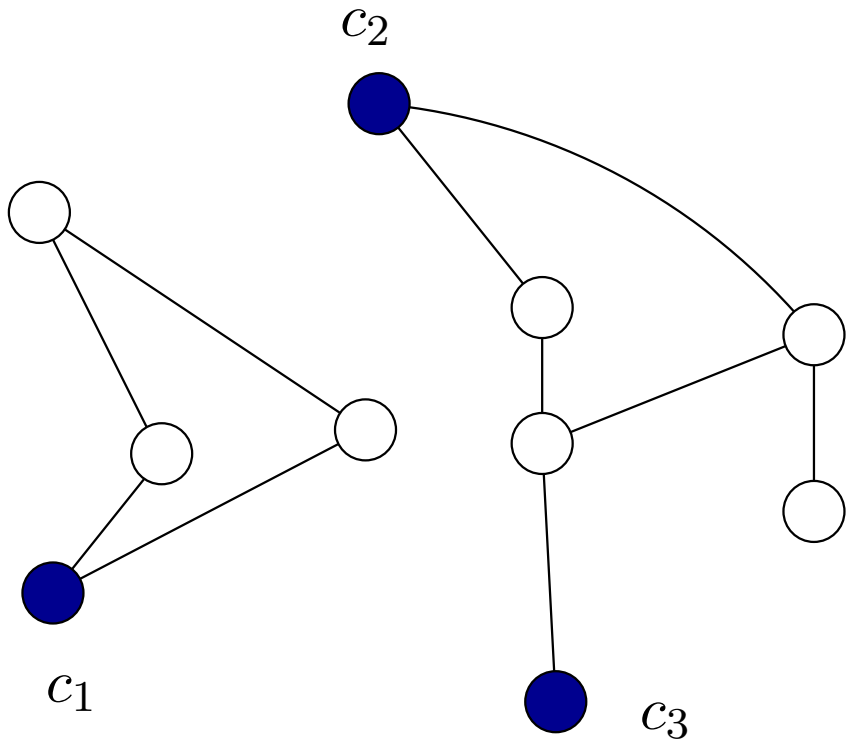
Construímos G_i^2 e obtemos um novo conjunto independente maximal.



Com $|I| = k$, tomamos I como conjunto de centros.

Solução obtida

Centros escolhidos:



Concluindo

O algoritmo devolve I como o conjunto de centros.

Concluindo

O algoritmo devolve I como o conjunto de centros.

Como I é maximal, todo vértice está a no máximo duas arestas "curtas" (de custo não superior a c_i) de distância de um vértice em I.

Concluindo

O algoritmo devolve I como o conjunto de centros.

Como I é maximal, todo vértice está a no máximo duas arestas "curtas" (de custo não superior a c_i) de distância de um vértice em I.

Pela desigualdade triangular, nenhum vértice está a distância maior que $2 \cdot c_i$ do centro mais próximo.

Concluindo

O algoritmo devolve I como o conjunto de centros.

Como I é maximal, todo vértice está a no máximo duas arestas "curtas" (de custo não superior a c_i) de distância de um vértice em I.

Pela desigualdade triangular, nenhum vértice está a distância maior que $2 \cdot c_i$ do centro mais próximo.

Como o ótimo não é menor que c_i (pela nossa rotina de decisão), vale que nenhum vértice está a distância maior que $2 \cdot \mathrm{OPT}$ do centro mais próximo.

Concluindo

Resultado...

Isso significa que nosso algoritmo é uma **2-aproximação**.

Concluindo

Resultado...

Isso significa que nosso algoritmo é uma **2-aproximação**.

Isso é o melhor que dá pra fazer pro k-*center* a menos que P $=$ NP!

Fim

É isso!

Muito obrigado pela atenção.

Fim

É isso!

Muito obrigado pela atenção.

Dúvidas?