Aula do Curso de Projeto e Análise de Algoritmos (PAA)

# Lidando com Problemas NP-Difíceis (Parte 2)

**Mário César San Felice**

(com materiais da Profa. Carla Negri Lintzmayer do CMCC-UFABC
e do Prof. Flávio Keidi Miyazawa do IC-Unicamp)

Universidade Federal de São Carlos
Departamento de Computação


14 de Fevereiro de 2025

# Abordagens para Problemas NP-Difíceis

Se P $\neq$ NP, então não podemos ter algoritmos para problemas NP-Difíceis que, simultaneamente,

1. encontrem soluções ótimas
2. em tempo polinomial
3. para qualquer entrada.

Nos resta não exigir todas essas propriedades do mesmo algoritmo!

Algumas abordagens possíveis são

- métodos exatos
- heurísticas e metaheurísticas
- algoritmos de aproximação
- parametrização e casos particulares

# Problema da Mochila

## Mochila Binária

**Entrada:** conjunto de $n$ itens, cada item $i$ tem peso $w_i$ e valor $v_i$, e capacidade $W$ da mochila.

**Soluções viáveis:** conjuntos de itens $S \subseteq \{1, \ldots, n\}$ com $\sum_{i \in S} w_i \leq W$.

**Função objetivo:** soma dos valores dos itens em $S$.

**Objetivo:** encontrar uma solução de valor máximo.

# Heurística construtiva para a Mochila

1: **função** MochilaGuloso($n$, $w$, $v$, $W$)
2: ordene e renomeie os itens para que $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n}$
3: seja $q$ um inteiro tal que $\sum_{i=1}^{q} w_i \leq W$ e $\sum_{i=1}^{q+1} w_i > W$
4: **devolve** $v_1 + v_2 + \ldots + v_q$

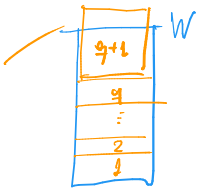
O que acontece no seguinte cenário?

ordem fração espaço livre = B

| $W = B$ | | razão |
|---|---|---|
| $v_1 = 2$ | $w_1 = 1$ | $\frac{v_1}{w_1} =$ 2 |
| $v_2 = B$ | $w_2 = B$ | $\frac{v_2}{w_2} =$ 1 |

# Algoritmo de aproximação para a Mochila

1: **função** MochilaAprox($n$, $w$, $v$, $W$)
2: ordene e renomeie os itens para que $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n}$
3: seja $q$ um inteiro tal que $\sum_{i=1}^{q} w_i \leq W$ e $\sum_{i=1}^{q+1} w_i > W$
4: **devolve** $\max\{v_1 + v_2 + \ldots + v_q, v_{q+1}\}$




Análise:

$ALG = \max\{v_1 + \ldots + v_q, v_{q+1}\}$

$\geq \frac{(v_1 + v_2 + \ldots + v_q) + (v_{q+1})}{2} \geq OPT_{Frac}/2$

$\geq OPT/2$

máx ≥ média

De onde vemos que esse algoritmo é uma $\frac{1}{2}$-aproximação.

# Algoritmos de aproximação

Relaxando a restrição de exigir soluções ótimas, com um detalhe.

Geram soluções viáveis em tempo polinomial e dão garantia no custo da solução gerada com relação ao custo da solução ótima.

Seja $A$ um algoritmo polinomial para um problema de minimização, $A(I)$ o custo da solução de $A$ para a entrada $I$ e $OPT(I)$ o custo da solução ótima.

$A$ é uma $\alpha$-aproximação se, para toda instância $I$,

$$ALG(I) \leq \alpha\, OPT(I)$$

Se o problema for de maximização, temos a definição

$$ALG(I) \geq \alpha\, OPT(I)$$

# Problema do Caixeiro Viajante

## Caixeiro Viajante (TSP)

**Entrada:** $G = (V, E)$ e função $w$ de peso nas arestas.

**Soluções viáveis:** circuitos hamiltonianos de $G$.

**Função objetivo:** soma dos pesos das arestas do circuito.

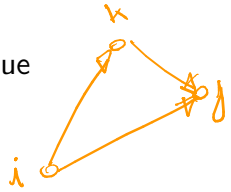
**Objetivo:** encontrar um circuito de menor custo.

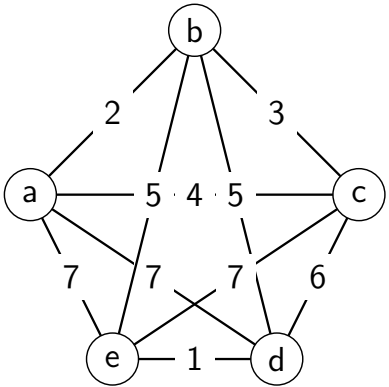
Não é possível obter uma aproximação constante para o TSP, por isso vamos focar no caso métrico deste problema.

# Algoritmo de aproximação para o TSP Métrico

Se um grafo $G = (V, E)$ com peso $w$ nas arestas é métrico

- então ele é completo e para todo trio $u, v, x \in V$ temos que $w(uv) \leq w(ux) + w(xv)$

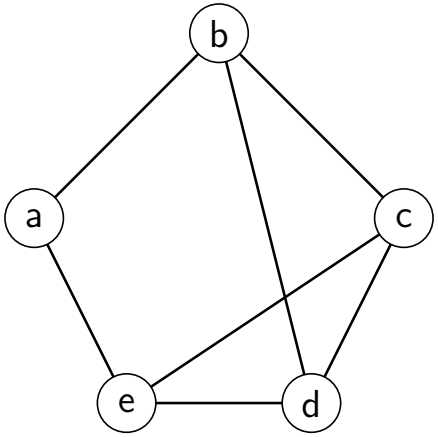



O TSP Métrico é o TSP onde o grafo de entrada é métrico.

# Algoritmo de aproximação para o TSP Métrico

Um atalho (short-cut) é a substituição de um caminho entre vértices $u$ e $v$, $(u, x_1, x_2, \ldots, x_k, v)$, pela aresta $uv$.

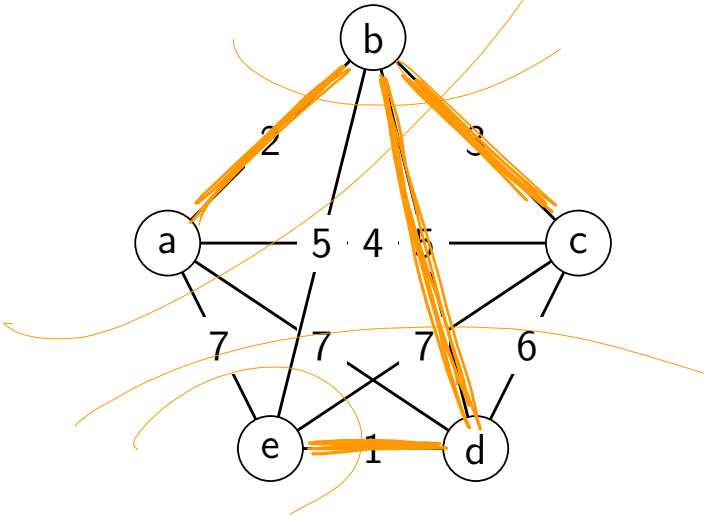



$(a, b, c, d) \rightarrow (a, d)$

Note que em um grafo métrico, $w(uv) \leq w(u, x_1, \ldots, x_k, v)$.

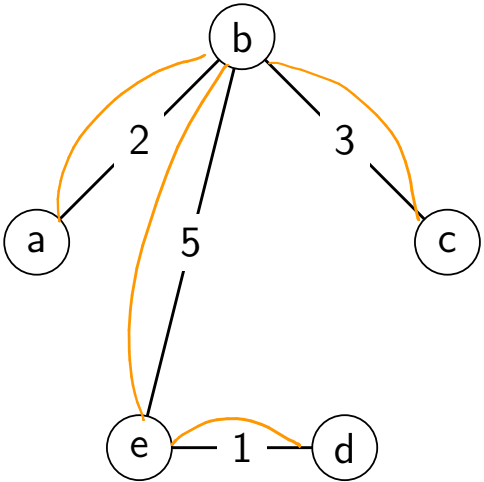
# Algoritmo de aproximação para o TSP Métrico

Encontre uma árvore geradora mínima (MST) $T$ de $G$.



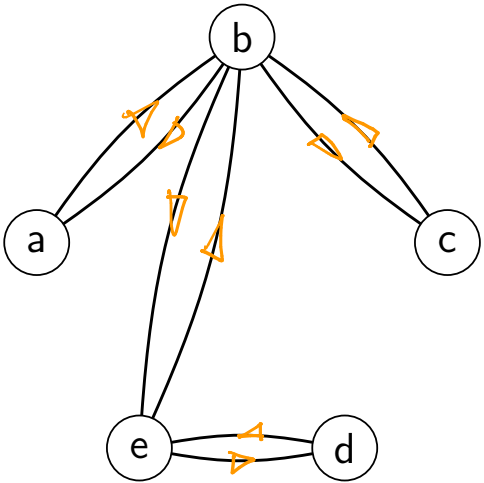
# Algoritmo de aproximação para o TSP Métrico

Duplique as arestas de $T$.

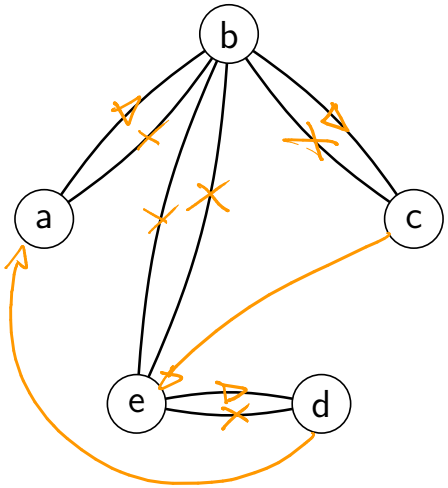


# Algoritmo de aproximação para o TSP Métrico

Encontre um ciclo Euleriano $\mathcal{E}$.



# Algoritmo de aproximação para o TSP Métrico

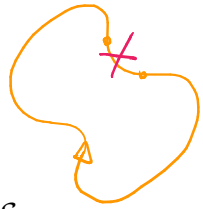



Percorra $\mathcal{E}$ fazendo short-cut se for repetir vértice.

$$\mathcal{E} = (a, b, c, b, e, d, e, b, a)$$

$$\mathcal{C} = (a, b, c, e, d, a)$$

# Algoritmo de aproximação para o TSP Métrico



1: **função** TSPMetricoAprox($G = (V, E)$, $w$)
2: $T \leftarrow MST(G, w)$
3: $T^2 \leftarrow T$ com arestas duplicadas
4: $\mathcal{E} \leftarrow$ ciclo euleriano em $T^2$
5: $\mathcal{C} \leftarrow$ circuito hamiltoniano obtido com short-cuts sobre $\mathcal{E}$
6: **devolve** $\mathcal{C}$

Análise:

$$ALG = c(\mathcal{C}) \leq c(\mathcal{E}) = c(T^2) = 2c(T) \leq 2\,OPT$$

De onde vemos que esse algoritmo é uma 2-aproximação.

# Problema do Corte Máximo

## Corte Máximo em Grafos

**Entrada:** um grafo $G = (V, E)$.

**Soluções viáveis:** um corte de $G$, i.e., um conjunto $S$ tal que $\emptyset \neq S \subset V$.

**Função objetivo:** número de arestas que sai de $S$, i.e., $|\delta(S)|$.

**Objetivo:** encontrar solução de custo máximo.

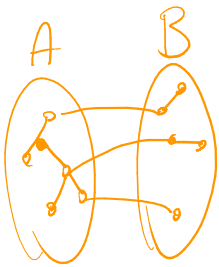
# Algoritmo de aproximação para o Corte Máximo

Vizinhança de um corte $S$:

- cortes que tem apenas um vértice a mais ou a menos que $S$

Seja $c(v) = \#$ de arestas incidentes a $v$ que atravessam o corte
e $d(v) = \#$ de arestas incidentes a $v$ que não atravessam o corte

1: **função** CorteMaximoBuscaLocal($G = (V, E)$)
2: $S \leftarrow$ um corte inicial; $\overline{S} \leftarrow V \setminus S$
3: **enquanto** houver vértice $v$ tal que $c(v) < d(v)$ **faça**
4: **se** $v \in S$ **então** $S \leftarrow S \setminus \{v\}$; $\overline{S} \leftarrow \overline{S} \cup \{v\}$
5: **senão** $\overline{S} \leftarrow \overline{S} \setminus \{v\}$; $S \leftarrow S \cup \{v\}$
6: **devolve** $S$

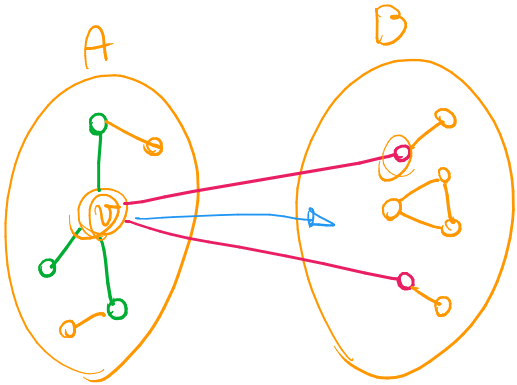


# Exemplo de funcionamento do algoritmo

$d(v) = 3$

$\vee$

$c(v) = 2$



# Análise

Vamos mostrar que esse algoritmo de busca local termina em tempo polinomial e tem razão de aproximação constante.

$$c(v) \geq d(v)$$
$$+c(v) \quad +c(v)$$
$$\Rightarrow 2c(v) \geq c(v) + d(v)$$

$$2\sum_{v \in V} c(v) \geq 2|E| \geq 2\,OPT \Rightarrow \sum_{v \in V} c(v) \geq OPT$$

$$2ALG = \sum_{v \in V} c(v) \Rightarrow ALG \geq OPT/2$$

Quiz: como generalizar essa heurística para a versão ponderada do Corte Máximo?

# Problema da Cobertura por Vértices

## Cobertura por Vértices

**Entrada:** grafo $G = (V, E)$ com peso $w$ nos vértices.

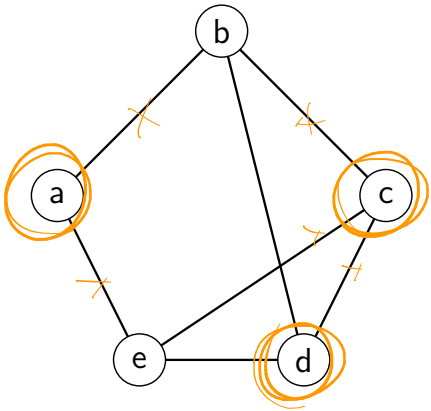
**Soluções viáveis:** subconjunto $S \subseteq V$ tal que para toda aresta $uv \in E$, $u \in S$ ou $v \in S$.

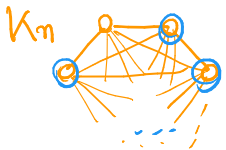
**Função objetivo:** soma dos pesos dos vértices em $S$.

**Objetivo:** encontrar solução de custo mínimo.

Esse problema é um caso particular da Cobertura por Conjuntos.

# Problema da Cobertura por Vértices





Quiz: como podemos reduzir o problema da cobertura por vértices ao problema da cobertura por conjuntos?

# Formulação em PLI para a Cobertura por Vértices

Sejam $x_v$ variáveis binárias que indicam se o vértice $v$ foi escolhido.

$$\begin{aligned} \text{minimizar} \quad & \sum_{v \in V} w_v x_v \\ \text{sujeito a} \quad & x_u + x_v \geq 1 \quad \forall uv \in E \\ & x_u \in \{0,1\} \quad \forall u \in V \end{aligned}$$

Programação linear é muito utilizada para encontrar soluções aproximadas.

Assim, considere o PL fruto da relaxação da integralidade do modelo anterior.

# Algoritmo de aproximação para a Cobertura por Vértices

1: **função** VertexCoverAprox($G = (V, E)$, $w$)
2: monte e resolva o PL, obtendo $x^*$
3: **para** $v \in V$ **faça**
4: $x_v = 1$, se $x_v^* \geq 0.5$
5: $x_v = 0$, se $x_v^* < 0.5$
6: **devolve** $\sum_{v \in V} w_v x_v$

Note que a solução construída é viável: como toda aresta satisfaz $x_u^* + x_v^* \geq 1$, algum dentre $x_u^*$ e $x_v^*$ deve ser pelo menos 0.5.

# Análise

Se $x_v = 1$, então $x_v^* \geq 0.5 \rightarrow x_v = 1 \leq 2x_v^*$.

Se $x_v = 0$, então também vale que $x_v = 0 \leq 2x_v^*$.

$$ALG = \sum_{v \in V} w_v x_v$$

$$\leq \sum_{v \in V} w_v 2x_v^*$$

$$= 2 \sum_{v \in V} w_v x_v^* = 2OPT_{PL} \leq 2OPT$$

De onde vemos que esse algoritmo é uma 2-aproximação.

Desafio: generalizar nosso algoritmo para tratar o problema da Cobertura por Conjuntos.

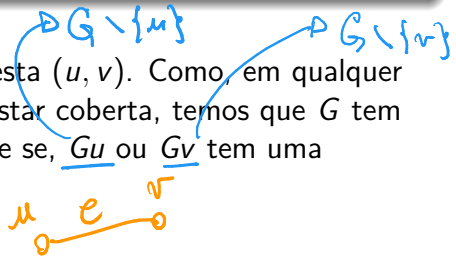
# Um problema parametrizado

## $k$ Cobertura por Vértices

**Entrada:** grafo $G = (V, E)$ e inteiro positivo $k$.

**Questão:** $G$ possui uma solução $S$ tal que $|S| \leq k$ e, para toda aresta $uv \in E$, $u \in S$ ou $v \in S$?

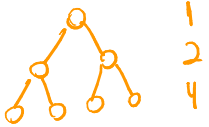
Subestrutura: Dado um grafo $G$ e uma aresta $(u, v)$. Como, em qualquer solução do problema a aresta $(u, v)$ deve estar coberta, temos que $G$ tem uma cobertura de tamanho $k$ se, e somente se, $Gu$ ou $Gv$ tem uma cobertura de tamanho $k - 1$.



# Algoritmo parametrizado para a $k$ Cobertura por Vértices

1: **função** VertexCoverK($G = (V, E)$, $k$)
2: **se** $G$ não possui arestas **então**
3: **devolve** SIM
4: **se** $k == 0$ **então**
5: **devolve** NÃO
6: escolha uma aresta $uv$ arbitrariamente
7: **se** VertexCoverK($G - u$, $k - 1$) retornar SIM **então**
8: **devolve** SIM
9: **se** VertexCoverK($G - v$, $k - 1$) retornar SIM **então**
10: **devolve** SIM
11: **devolve** NÃO

$T(k) = 2T(k-1) + cn$




$2^k$

Esse algoritmo executa em tempo $O(|V| \cdot 2^k)$: no pior caso, efetua duas chamadas recursivas, mas a árvore de busca tem altura no máximo $k$.

# Complexidade parametrizada

Relaxando a restrição de resolver qualquer instância.

Abordagem relativamente recente, que busca analisar se os problemas NP-Difíceis admitem algoritmos cujo tempo não polinomial depende apenas de alguns aspectos do problema:

- "Dada uma entrada $I$ e um inteiro não negativo $k$, $I$ tem alguma propriedade que depende de $k$?"

Esse valor $k$ é chamado de parâmetro e, na prática, ele pode ser pequeno com relação ao tamanho de $I$.

# Complexidade parametrizada

Esse valor $k$ é chamado de parâmetro e, na prática, ele pode ser pequeno com relação ao tamanho de $I$.

A ideia é encontrar algoritmos que são exponenciais apenas com relação a $k$ e polinomiais com relação ao tamanho $n$ de $I$,

- i.e., $O(n^c f(k))$.

Se for possível, o problema pertence à classe FPT, de problemas tratáveis por parâmetro fixo.