Teoria da Complexidade

Cid C. de Souza / IC–UNICAMP

Universidade Estadual de Campinas
Instituto de Computação

1º semestre de 2012

Revisado por Zanoni Dias

Autor

Prof. Cid Carvalho de Souza
Universidade Estadual de Campinas (UNICAMP)
Instituto de Computação
Av. Albert Einstein nº 1251
Cidade Universitária Zeferino Vaz
13083-852, Campinas, SP, Brasil
Email: `cid@ic.unicamp.br`

Direitos autorais

- Este material só pode ser reproduzido com a autorização do autor.
- Os alunos dos cursos do Instituto de Computação da UNICAMP bem como os seus docentes estão autorizados (e são bem vindos) a fazer uma cópia deste material para estudo individual ou para preparação de aulas a serem ministradas nos cursos do IC/UNICAMP.
- Se você tem interesse em reproduzir este material e não se encontra no caso acima, por favor entre em contato comigo.
- Críticas e sugestões são muito bem vindas!

Campinas, agosto de 2010.

Cid

Tratamento de problemas $\mathcal{NP}$-difíceis: Heurísticas

▷ Heurísticas são algoritmos que geram soluções viáveis para quais *não se pode dar garantias de qualidade*. Ou seja, não se sabe o quão *distante* a solução gerada está de uma solução ótima (5%?, 10%?, 50%?, 100%?, ...).

▷ Tipos de heurísticas:
- *construtivas*: normalmente adotam estratégias *gulosas* para construir as soluções. Tipicamente são aplicadas a problemas onde é fácil obter uma solução viável.
- *de busca local*: partem de uma solução inicial e, através de transformações bem definidas, visitam outras soluções até atingir um *critério de parada* pré-definido.

Heurísticas Construtivas (TSP)

▷ Exemplo 1: TSP em um grafo não orientado completo.

```
Vizinho-Mais-Próximo(n, d)     (* d: matriz de distâncias *)
    Para i = 1 até n faça visitado[i] ← Falso;
    visitado[1] ← Verdadeiro;
    ciclo ← {}, comp ← 0 e k ← 1;
    Para i = 1 até n − 1 faça
        j* ← argmin{d[k, j] : visitado[j] = Falso};
        visitado[j*] ← Verdadeiro;
        ciclo ← ciclo ∪ {(k, j*)};     comp ← comp + d[k, j*];
        k ← j*;
    fim-para
    ciclo ← ciclo ∪ {(k, 1)};     comp ← comp + d[k, 1];
Retorne comp.
```

▷ Complexidade: $O(n^2)$

Heurísticas Construtivas (TSP)

▷ Exemplo 2: heurística para o TSP $\equiv$ algoritmo de Kruskal para AGM.

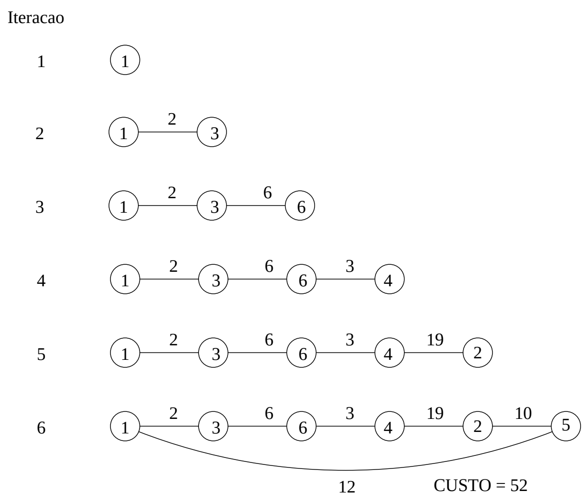
```
TSP-Guloso(n, d)     (* d: matriz de distâncias *)
    ℒ ← lista das arestas ordenadas crecentemente pelo valor de d;
    Para i = 1 até n faça grau[i] ← 0;   componente[i] = i   fim-para
    k ← 0;     ciclo ← {};     comp ← 0;
    Enquanto k ≠ n faça
        (u, v) ← Remove-primeiro(ℒ);
        Se (grau[u] ≤ 1 e grau[v] ≤ 1 e componente(u) ≠ componente(v))
        ou (grau[u] = grau[v] = 1 e k = n − 1) então
            ciclo ← ciclo ∪ {(u, v)};     comp ← comp + d[u, v];
            Unir-componentes(u, v);
            grau[u] + +;     grau[v] + +;     k + +;
        fim-se
    fim-enquanto
Retorne comp.
```

▷ Complexidade: $O(n^2 \log n)$ (usar *compressão de caminhos* para *união de conjuntos disjuntos*).

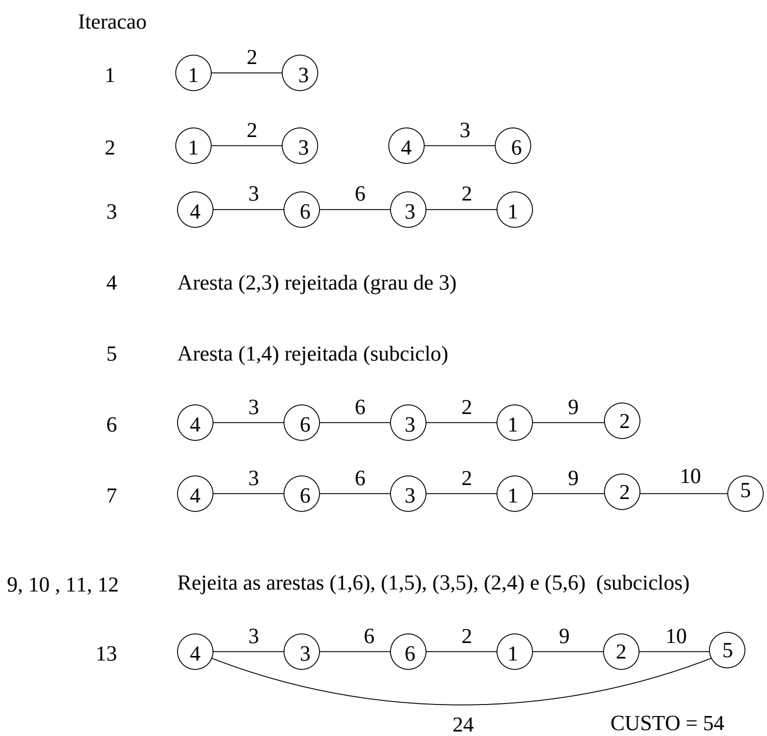
Aplicação das heurísticas para o TSP:

$$d = \begin{bmatrix} - & 9 & 2 & 8 & 12 & 11 \\ 9 & - & 7 & 19 & 10 & 32 \\ 2 & 7 & - & 29 & 18 & 6 \\ 8 & 19 & 29 & - & 24 & 3 \\ 12 & 10 & 18 & 24 & - & 19 \\ 11 & 32 & 6 & 3 & 19 & - \end{bmatrix}$$

Vizinho-Mais-Próximo



TSP-GULOSO



Heurísticas Construtivas (Mochila)

▷ Exemplo 2: Problema da Mochila.

```
Mochila-guloso(c, w, W)
    Ordenar itens segundo a razão c_i/w_i;
    (* assuma que c_1/w_1 ≥ c_2/w_2 ≥ ... ≥ c_n/w_n *)

    W̄ ← W;     S ← {};
    Para i = 1 até n faça
        Se w_i ≤ W̄ então
            W̄ ← W̄ − w_i;
            S ← S ∪ {i};
        fim-se
    fim-para
Retorne S.
```

▷ Complexidade: $O(n \log n)$.

Heurísticas Construtivas (Mochila)

▷ Aplicação da heurística `Mochila-guloso`.

$$\begin{array}{ll} \text{maximize} & 8x_1 + 16x_2 + 20x_3 + 12x_4 + 6x_5 + 10x_6 + 4x_7 \\ \text{Sujeito a} & 3x_1 + \ \ 7x_2 + \ \ 9x_3 + \ \ 6x_4 + 3x_5 + \ \ 5x_6 + 2x_7 \leq 17, \\ & x \in \mathbb{B}^7. \end{array}$$

Observação: $\frac{8}{3} \geq \frac{16}{7} \geq \frac{20}{9} \geq \frac{12}{6} \geq \frac{6}{3} \geq \frac{10}{5} \geq \frac{4}{2}$

▷ Solução gulosa: $S = \{1, 2, 4\}$, custo $= 36$.

▷ Solução ótima: $S = \{1, 2, 6, 7\}$, custo $= 38$.

Heurísticas Construtivas

- Soluções gulosas podem ser *arbitrariamente ruins*!
- `Mochila-guloso` é arbitrariamente ruim.
- Instância: $W = n$, $c_1 = 3/n$, $w_1 = 2/n$ e, para todo $i = 2, \ldots, n$, $c_i = n - (1/n)$ e $w_i = n - (1/n)$.
  Observação: $\frac{c_1}{w_1} \geq \frac{c_2}{w_2} = \ldots = \frac{c_n}{w_n}$.
- Solução gulosa: $S = \{1\}$, custo $= 3/n$.
- Solução ótima: $S = \{2\}$, custo $= n - (1/n)$.
- $\lim_{n\to\infty} \frac{(3/n)}{n-(1/n)} = 0$.
  Ou seja, aumentando o valor de n nesta instância, a solução gulosa pode se afastar tanto quanto eu quiser da solução ótima!

Heurísticas Construtivas

- `Vizinho-Mais-Próximo` para o TSP é arbitrariamente ruim.
- Instância: matriz simétrica de distâncias d onde, para $i < j$, tem-se:

$$d[i,j] = \begin{cases} n^2, & \text{se } i = n-1 \text{ e } j = n, \\ 1, & \text{se } j = i+1, \\ 2, & \text{caso contrário.} \end{cases}$$

- Solução gulosa: ciclo $= \{1, 2, \ldots, n-1, n\}$ e comp $= n^2 + n$.
- Solução ótima: ciclo $= \{1, 2, \ldots, n-3, n, n-2, n-1\}$ e comp $= n + 3$.
- $\lim_{n\to\infty} \frac{n+3}{n^2+n} = 0$.
  Novamente, aumentando o valor de n nesta instância, a solução gulosa pode se afastar tanto quanto eu quiser da solução ótima!

Heurísticas de Busca Local

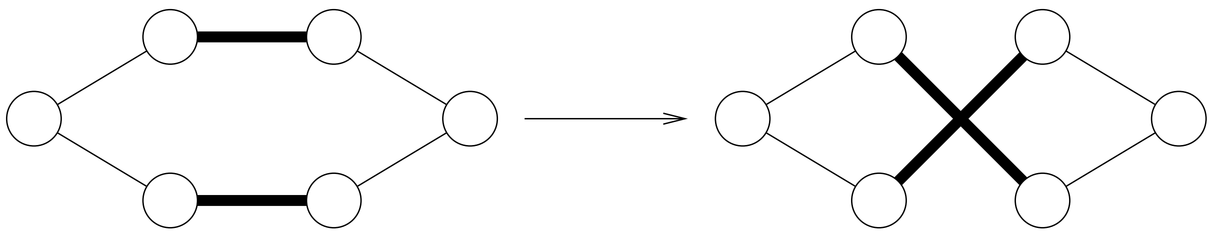
- Como nos algoritmos de *branch & bound* e *backtracking*, as soluções são representadas por tuplas.
- Sendo $\mathcal{F}$ o conjunto de todas as possíveis tuplas e $t \in \mathcal{F}$, a *vizinhança* da solução t, $N(t)$, é o subconjunto de tuplas de $\mathcal{F}$ que podem ser obtidas ao se realizar um conjunto de transformações pré-determinadas sobre t.
- *Complexidade da vizinhança*: número de tuplas na vizinhança.
- Exemplo 1:
  A tupla é um vetor binário de tamanho n.
  $N_1(t)$: conjunto de todas as tuplas obtidas de t "flipando" uma de suas componentes.
  Complexidade: $\Theta(n)$.

Heurísticas de Busca Local

- Exemplo 2:
  A tupla é um vetor representando uma permutação de $\{1, \ldots, n\}$.
  $N_2(t)$: conjunto de todas as tuplas obtidas trocando-se as posições de dois elementos da permutação.
  Complexidade: $\Theta(n^2)$.
- Algoritmo de busca local (problema de minimização):
  - Encontrar uma solução inicial t.
  - Encontrar t' em $N(t)$ com menor custo.
  - Se o custo de t' é menor que o custo de t, fazer $t \leftarrow t'$ e repetir o passo anterior. Se não, retorne t e pare.

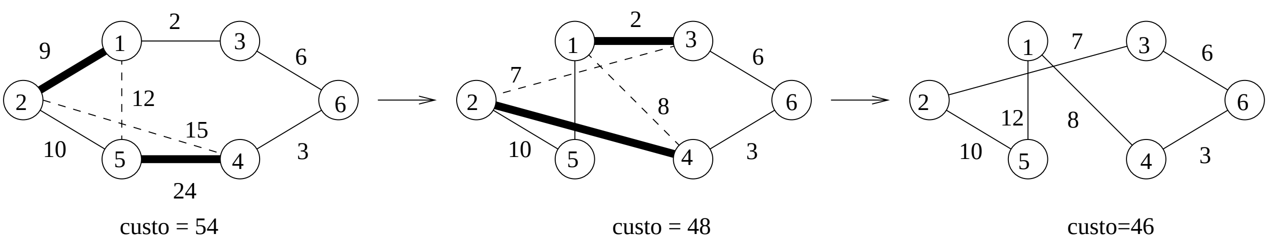
Heurísticas de Busca Local (TSP)

- ▷ Heurística da 2-troca para o TSP (Lin e Kernighan, 1973).
- ▷ Ciclo representado por uma permutação dos n vértices.
- ▷ Vizinhança: substituir pares de arestas.



- ▷ Complexidade: $\Theta(n^2)$.

Heurísticas de Busca Local (TSP)



- ▷ Tuplas: vetor de permutações de 1 até n.
- ▷ Vizinhança:
  inverte seqüência entre posições i e $j \pmod{n}$ $(j \geq i+2)$.
- ▷ No exemplo:
  $(1,3,6,\underline{4},5,2,\underline{1}) \Longrightarrow (\underline{1},3,6,4,\underline{2},5,1) \Longrightarrow (1,4,6,3,2,5,1)$

C. de Souza Teoria da Complexidade

Heurísticas de Busca Local (Partição de Grafos)

- Entrada: grafo não orientado $G = (V, E)$, com $|V| = 2n$, e custos c_{ij} para toda aresta $(i, j) \in E$.
- Saída: um subconjunto $V' \subseteq V$, com $|V'| = n$ e que minimize o valor de $\sum_{i \in V'} \sum_{j \notin V'} c_{ij}$.
- Solução representada por um vetor a de $2n$ posições, com os valores de 1 até $2n$. Nas n primeiras posições estão os vértices de V' e nas n seguintes os vértices de $\overline{V'}$.
- Vizinhança: todas as trocas possíveis de pares de vértices $(a[i], a[j])$, onde $1 \leq i \leq n$ e $(n+1) \leq j \leq 2n$.
- Complexidade: $\Theta(n^2)$.

Heurísticas de Busca Local (Partição de Grafos)

- Exemplo: grafo completo com 6 vértices (K_6).

$$c = \begin{bmatrix} - & 9 & 2 & 8 & 12 & 11 \\ 9 & - & 7 & 19 & 10 & 32 \\ 2 & 7 & - & 29 & 18 & 6 \\ 8 & 19 & 29 & - & 24 & 3 \\ 12 & 10 & 18 & 24 & - & 19 \\ 11 & 32 & 6 & 3 & 19 & - \end{bmatrix}$$

Solução inicial:
$a = \{1, 4, 6, 2, 3, 5\}$.

vizinhos	(1, 2)	(1, 3)	(1, 5)	(4, 2)	(4, 3)	(4, 5)	(6, 2)	(6, 3)	(6, 5)
ganho	−29	−12	−7	$\underline{-66}$	−15	−40	−22	−43	−32

- Nova solução: $a = \{1, 2, 6, 4, 3, 5\}$.

vizinhos	(1, 4)	(1, 3)	(1, 5)	(2, 4)	(2, 3)	(2, 5)	(6, 4)	(6, 3)	(6, 5)
ganho	37	34	23	66	51	26	44	59	54

- *Ótimo Local!*

Heurísticas de Busca Local

- ▷ Pode ser vantajoso que a busca local passe por soluções inviáveis!
- ▷ Nesses casos a função objetivo é composta de duas parcelas:

$$g(.) = f(.) + \alpha h(.),$$

onde f é função original, h é uma função que mede quão inviável é a solução e α é um fator de penalização.

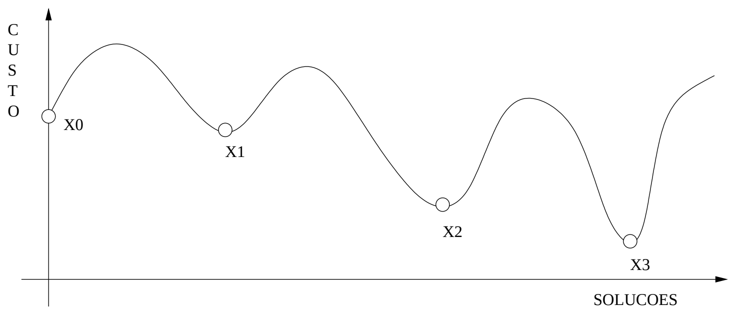
- ▷ Exemplo: no problema da partição de grafos, considere a vizinhança onde só um vértice muda de V' para $\overline{V'}$ ou vice-versa.
- ▷ Penalizar as soluções inviáveis usando $\alpha > 0$ grande e definindo:

$$h(V', \overline{V'}) = ||V'| - |\overline{V'}||^2.$$

- ▷ Se acabar em uma solução inviável, aplicar um algoritmo guloso que rapidamente restaura a viabilidade.

Heurísticas de Busca Local

- ▷ Busca local retorna solução que é ótimo local.



- ▷ Escapando de ótimos locais: mover para melhor vizinho mesmo se o custo for pior.
- ▷ *Metaeurísticas*: Busca Tabu, Simulated Annealing, Algoritmos genéticos, etc.

Heurísticas de Busca Local (Busca Tabu)

- Inserir na busca local uma lista de movimentos **tabu** que impedem, por algumas iterações, que um determinado movimento seja realizado.
  Objetivo: evitar que uma solução seja revisitada.
  Exemplo: no caso da equipartição de grafos, pode-se impedir que a troca de dois vértices por t iterações.
- Repetir a busca local básica por α iterações ou se nenhuma melhora foi obtida nas últimas β iterações.
- Os parâmetros α e β são fixados *a priori*.
- Parâmetros a ajustar: tamanho da lista tabu t, α e β.

Tratamento de problemas $\mathcal{NP}$-difíceis: Aproximações

- *Algoritmos aproximados* encontram uma solução **com garantia de qualidade** em tempo polinomial.
- Nomenclatura:

P	problema $\mathcal{NP}$-difícil
H	algoritmo aproximado
I	instância de P
$z^*(I)$	valor ótimo da instância I
$z^H(I)$	valor da solução obtida por H para a instância I

- *Aproximação absoluta*: para algum $k \in \mathbb{Z}_+$ tem-se que

$$|z^*(I) - z^H(I)| \leq k, \text{ para todo } I.$$

Aproximação Absoluta

- Exemplo 1: alocação de arquivos em discos (MFA). Dados n arquivos de tamanhos $\{\ell_1, \ldots, \ell_n\}$ e **dois** discos de capacidade L, qual o maior número de arquivos que podem ser armazenados nos discos?
- *Teorema*: MFA $\in \mathcal{NP}$-completo. (**Exercício**)
- Algoritmo: supor que $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_n$.

`Aprox-MFA`(n, ℓ);
    $L' \leftarrow L$; $j \leftarrow 1$;
    Enquanto $L' \geq \ell_j$ **faça**
        $L' \leftarrow L' - \ell_j$; `Colocar`$(j, 1)$; $j++$;
    fim-enquanto;
    $L' \leftarrow L$;
    Enquanto $L' \geq \ell_j$ **faça**
        $L' \leftarrow L' - \ell_j$; `Colocar`$(j, 2)$; $j++$;
    fim-enquanto;
Retornar $j - 1$.

Aproximação Absoluta

Teorema: $|z^*(I) - z^H(I)| \leq 1$.
<u>*Prova*</u>: Seja p o número de arquivos que o algoritmo `Aprox-MFA` consegue armazenar em um grande disco com capacidade $2L$.
Além disso, seja $j = \text{argmax}\{\sum_{i=1}^{j} \ell_i \leq L\} \leq p$.

1. $p \geq z^*(I)$; (a)
2. $\sum_{i=1}^{p} \ell_i \leq 2L$; (b)
3. $\sum_{i=j+1}^{p-1} \ell_i \leq \sum_{i=j+2}^{p} \ell_i \leq L$, devido a (b) e à definição de j que implica que $\sum_{i=1}^{j+1} \ell_i > L$. (c)

$$\ldots \overset{(c)}{\Longrightarrow} z^H(I) \geq p - 1 \overset{(a)\wedge(c)}{\Longrightarrow} z^H(I) \geq z^*(I) - 1. \qquad \square$$

Aproximação Absoluta

- Exemplo 2: coloração de *grafos planares* (CGP).
- Resultados conhecidos:
  - CGP $\in \mathcal{NP}$-completo.
  - Todo grafo planar tem pelo menos um vértice de grau menor do que 6.
  - Um *grafo é bipartido* se e somente se ele não tem ciclos ímpares.

```
6-cores(G);     (* G = (V,E) *)
    Se |V| = 0 então Retornar 0;  Se |E| = 0 então Retornar 1;
    Se G é bipartido então Retornar 2;
    se não
        Escolher v com grau(v) ≤ 5;   G' ← G − v;   k ← 6-cores(G');
        Seja x ∈ {1,2,3,4,5,6} uma cor diferente daquela dos vizinhos de v;
        Se (x > k) então  k ← k + 1;  x ← k;  fim-se;
        cor[v] ← x;
    fim-se
Retornar  k.
```


Aproximação Absoluta

- **Teorema**: $|z^*(I) - z^H(I)| \leq 3$.

  Prova: Se $|V| = 0$, $|E| = 0$ ou o grafo é bipartido então a coloração feita por `6-cores` é ótima e o resultado é imediato. Caso contrário, G tem pelo menos um *ciclo ímpar*. Logo qualquer coloração precisará de pelo menos três cores. Como o número de cores usadas por `6-cores` é ≤ 6 e a solução ótima requer pelo menos 3 cores, tem-se que

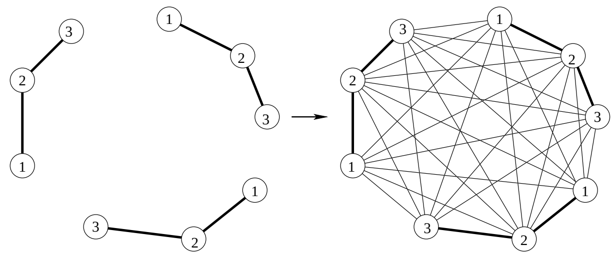
$$|z^*(I) - z^H(I)| \leq |3 - 6| = 3.$$

$\square$

- **Observações**:
  - Todo grafo planar admite uma 4-coloração.
  - São poucos problemas que tem aproximação absoluta.

Aproximação Absoluta $\times$ Questão $\mathcal{P} = \mathcal{NP}$

- **Teorema**: Não existe uma aproximação absoluta para `CLIQUE` com complexidade polinomial a menos que $\mathcal{P} = \mathcal{NP}$.
  Prova: Suponha que $\mathcal{P} \neq \mathcal{NP}$ e que existe um algoritmo polinomial H para `CLIQUE` tal que $|z^*(I) - z^H(I)| \leq k \in \mathbb{Z}_+$.
  Seja G^{k+1} o grafo composto de $k+1$ cópias de G mais todas as arestas ligando pares de vértices em diferentes cópias.



Observação: se α é o tamanho da maior clique de G então a maior clique de G^{k+1} tem $\alpha(k+1)$ vértices.

Aproximação Absoluta $\times$ Questão $\mathcal{P} = \mathcal{NP}$

- *Prova*: (cont.)
  Executando-se H para G^{k+1} tem-se que

$$z^*(G^{k+1}) - z^H(G^{k+1}) \leq k \Longrightarrow z^H(G^{k+1}) \geq (k+1)z^*(G) - k.$$

Se C é a clique encontrada por H em G^{k+1}, existe uma cópia de G tal que $C' = V \cap C$ e $|C'| \geq |C|/(k+1)$. Logo

$$|C'| \geq \frac{(k+1)z^*(G) - k}{k+1} = z^*(G) - \frac{k}{k+1}.$$

Portanto, $|C'| \geq z^*(G)$, ou seja C' é uma clique máxima de G.
Absurdo! $\square$

α-Aproximação

- Um algoritmo H para um problema P é uma α-*aproximação* se
  - P é um problema de **minimização** e $\dfrac{z^H(I)}{z^*(I)} \leq \alpha \;\; \forall\, I$,

  ou

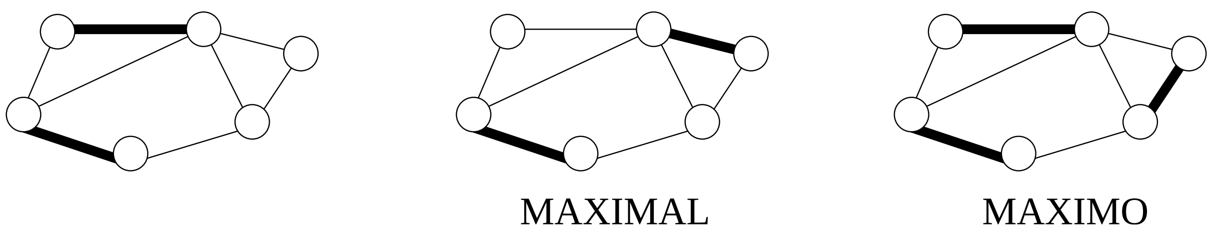
  - P é um problema de **maximização** e $\dfrac{z^*(I)}{z^H(I)} \leq \alpha \;\; \forall\, I$.

  Observação: α é sempre maior ou igual a 1.
- Um algoritmo H é uma α-*aproximação relativa* para um problema P se

$$\left| \frac{z^*(I) - z^H(I)}{z^*(I)} \right| \leq \alpha, \;\; \text{para todo } I.$$

α-Aproximação

- Exemplo 1: Cobertura mínima de vértices (CV).
- *Definições*: emparelhamento em grafos.



- Algoritmo:

CV-2-Aprox(G); (* $G = (V, E)$ *)
  $C \leftarrow \{\}$;
  Construir um emparelhamento maximal M^* em G;
  Para todo $(u, v) \in M^*$ **faça** $C \leftarrow C \cup \{u, v\}$;
Retornar C.

▷ **Teorema**: $\frac{z^H(I)}{z^*(I)} \leq 2$.

<u>*Prova*</u>:

Parte I: C é uma cobertura de vértices pois, se existisse uma aresta (u, v) não coberta então M^* não seria maximal.

Parte II: $|C| \leq 2z^*(I)$.
Se C' e M' são respectivamente uma cobertura e um emparelhamento qualquer de G então $|C'| \geq |M'|$. Logo:

$$z^*(I) \geq |M^*| = \frac{|C|}{2} = \frac{z^H(I)}{2}.$$

$\square$

▷ Exemplo 2: *bin packing* unidimensional.
Dados n arquivos de tamanhos $\{t_1, \ldots, t_n\}$ e disquetes de capacidade de armazenamento C, qual o menor número de disquetes necessários para fazer o *backup* de todos os arquivos?
Observação: supor que $t_i \leq C$ para todo $i = 1, \ldots, n$.

▷ Algoritmo básico:

```
Bin-Aprox(t, n, C);
   Preprocessamento(t, n);    Disquetes-em-uso ← {};    k ← 0;
   Para i = 1 até n faça
      j ← Escolher-disquette(Disquetes-em-uso, i);
      Se j = 0 então    (* arquivo não cabe nos disquetes em uso *)
         k + +;   Disquetes-em-uso ← Disquetes-em-uso ∪ {k};   j ← k;
      fim-se
      Armazenar(i, j);
   fim-para
Retornar k.
```

α-Aproximação

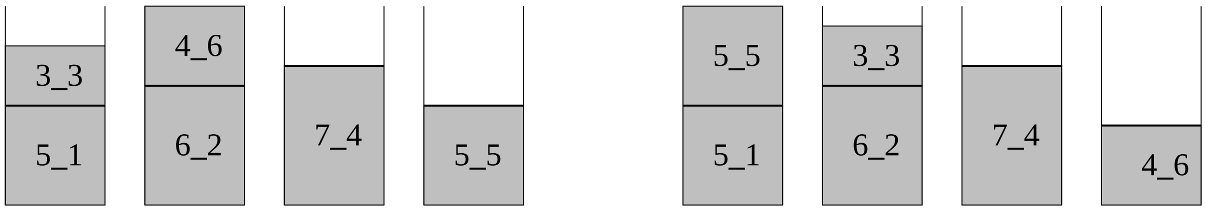
▷ Descrição dos procedimentos do algoritmo `Bin-Aprox`:
- `Preprocessamento`: retorna uma nova permutação dos arquivos.
- `Escolher-disquete`: retorna o número do disquete em uso onde será armazenado o arquivo *i* ou *zero* caso não encontre disquete com capacidade residual de armazenamento suficiente.
- `Armazenar`: registra que o arquivo *i* será alocado ao *j*-ésimo disquete, atualizando a sua capacidade residual de armazenamento.

α-Aproximação

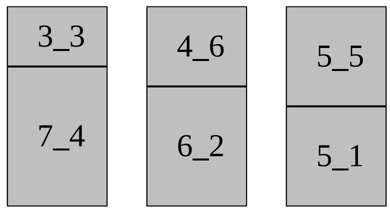
▷ Estratégias alternativas:
- *First Fit* (FF): `Preprocessamento` mantém ordem dos arquivos de entrada e `Escolher-disquete` procura o disquete em uso de *menor índice* aonde cabe o arquivo corrente.
- *Best Fit* (BF): `Preprocessamento` mantém ordem dos arquivos de entrada e `Escolher-disquete` procura o disquete em uso de *menor capacidade residual de armazenamento* aonde cabe o arquivo corrente.
- *First Fit Decrease* (FFD): variante do algoritmo FF onde o `Preprocessamento` ordena os arquivos em ordem decrescente de tamanho.
- *Best Fit Decrease* (BFD): variante do algoritmo BF onde o `Preprocessamento` ordena os arquivos em ordem decrescente de tamanho.

α-Aproximação

▷ Exemplo de aplicação dos algoritmos para *bin packing*:
$C = 10,\ n = 6,\ t = \{5_1, 6_2, 3_3, 7_4, 5_5, 4_6\}$ (notação: $i_j \Longrightarrow t_j = i$).



FF BF



FFD = BFD = OTIMO

α-Aproximação

▷ **Teorema**: FF é um algoritmo 2-aproximado para *bin packing*.

Prova: seja b o valor retornado por FF e b^* o valor ótimo. Suponha que os disquetes estão ordenados decrescentemente pela sua capacidade residual. Note que a capacidade residual dos $b-1$ primeiros disquetes da solução de FF é $\leq C/2$. Caso contrário, se dois disquetes tivessem capacidade residual $\geq C/2$ os seus arquivos teriam sido armazenados em um único disquete. Como o total armazenado no disquete b é maior que a capacidade residual dos demais disquetes, tem-se

$$S = \sum_{i=1}^{n} t_i \geq b\,\frac{C}{2}.$$

Como $b^* \geq \lceil \frac{S}{C} \rceil \geq \frac{S}{C}$, a equação acima implica que $b^* \geq \frac{1}{2}\,b$.
$\square$

α-Aproximação

- **Teorema**: para toda instância I do *bin packing* tem-se que

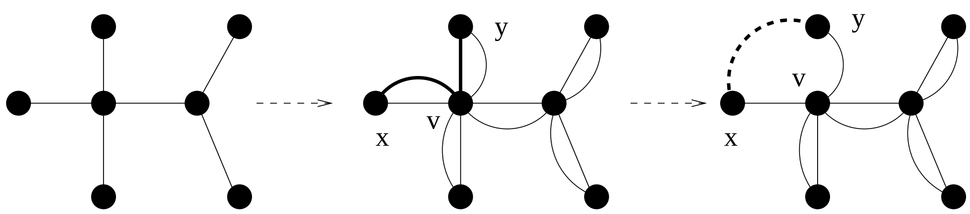
$$z^{xx}(I) \leq \frac{17}{10}\, z^*(I) + 2 \quad \text{e} \quad z^{xxD}(I) \leq \frac{11}{9}\, z^*(I) + 2,$$

onde $xx \in \{FF, BF\}$.

- Exemplo 3: 2-aproximação para o TSP-*métrico*, ou seja, quando as distâncias obedecem à *desigualdade triangular*.

> `TSP-Aprox`(G); (* $G = (V, E)$ e completo *)
> Construir T, uma árvore geradora minima de G;
> Construir o grafo C duplicando-se todas as arestas de T;
> **Enquanto** houver vértices de grau > 2 em C **faça**
> $v \leftarrow$ vértice de grau > 2 tal que existem vértices distintos x e y com (x, v) e (y, v) em C;
> Faça $C \leftarrow (C \cup (x, y)) - \{(x, v), (y, v)\}$; (*)
> **fim-enquanto**
> **Retorne** C;

α-Aproximação



- **Teorema**: `TSP-Aprox` é uma 2-aproximação para o TSP-*métrico*.

<u>*Prova*</u>: se z^* é o custo mínimo de um ciclo hamiltoniano em G,

$$\text{custo}(T) \leq z^* \Rightarrow 2\ \text{custo}(T) \leq 2z^*.$$

Por outro lado, devido aos custos obedecerem à desigualdade triangular, o comando (*) só pode diminuir o custo de C ao longo das iterações. Logo

$$\text{custo}(C) \leq 2\ \text{custo}(T) \leq 2z^*. \qquad \square$$

α-Aproximação $\times$ Questão $\mathcal{P} = \mathcal{NP}$

- **Teorema**: Não existe uma α-aproximação para TSP (genérico) com complexidade polinomial a menos que $\mathcal{P} = \mathcal{NP}$.
  Prova: Suponha que $\mathcal{P} \neq \mathcal{NP}$ e que existe um algoritmo polinomial H tal que $\frac{z^H(I)}{z^*(I)} \leq \alpha \in \mathbb{Z}_+$.
  Seja G o grafo dado como entrada do problema de decisão do ciclo hamiltoniano (HAM). Construa o grafo G' completando com as arestas que faltam. Atribua custo um às arestas originais e custo αn àquelas que foram inseridas no passo anterior.
  Se G tem um ciclo hamiltoniano, então o valor ótimo do TSP é $z^*(G) = n$. Como H é α-aproximado para o TSP

$$\frac{z^H(G)}{z^*(G)} \leq \alpha \;\Rightarrow\; z^H(G) \leq \alpha\, n.$$

α-Aproximação $\times$ Questão $\mathcal{P} = \mathcal{NP}$

- *Prova* (cont.):

  Assim, quando G tem um ciclo hamiltoniano, o ciclo encontrado por H para o TSP só terá arestas originais de G!

  Por outro lado, se G não tem ciclo hamiltoniano, $z^H(G) \geq 1 + \alpha\, n$.

  Portanto, G tem um ciclo hamiltoniano se somente se $z^H(G) \leq \alpha\, n$, ou seja, H resolve HAM em tempo polinomial.
- Absurdo, já que, por hipótese, $\mathcal{P} \neq \mathcal{NP}$. □