ALGORITMOS DE APROXIMAÇÃO

Marcus Ritt

Outline

P ≠ NP

Otimização

- Reduzir tempo por reduzir qualidade
- Mas: qualidade pessimista garantida
- Frequentemente: resolver problemas NP-completos em tempo polinomial

Notação

- Algoritmo de aproximação A
- Solução $A(x) = y$ aproximada para instância x
- Valor $\varphi(x, y)$ da solução
- Solução ótima y^*.
- Valor $\mathrm{OPT}(x) = \varphi(x, y^*)$ da solução ótima.

Um *problema de otimização* $\Pi = (\mathcal{P}, \varphi, \text{opt})$ é uma relação binária $\mathcal{P} \subseteq I \times S$ com instâncias $x \in I$ e soluções $y \in S$, junto com

- uma função de otimização (função de objetivo) φ: $\mathcal{P} \to \mathbb{N}$ (ou $\mathbb{Q}$).
- um objetivo: Encontrar mínimo ou máximo

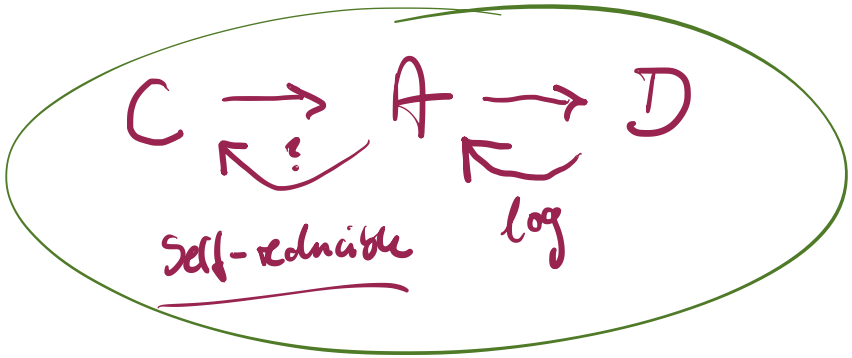
$$\text{OPT}(x) = \text{opt}\{\varphi(x, y) \mid (x, y) \in \mathcal{P}\}$$

junto com uma solução y^* tal que $f(x, y^*) = OPT(x)$.

O par $(x, y) \in \mathcal{P}$ caso y é uma solução para x.
Uma instância x de um problema de otimização possui soluções $S(x) = \{y \mid (x, y) \in \mathcal{P}\}$.

Três subproblemas

- Construção: Dado x, encontra a solução ótima y^* e seu valor $\mathrm{OPT}(x)$.
- Avaliação: Dado x, encontra valor ótimo $\mathrm{OPT}(x)$.
- Decisão: Dado x e k, decide se $\mathrm{OPT}(x) \geq k$ (maximização) ou $\mathrm{OPT}(x) \leq k$ (minimização).



- *Aproximação absoluta* garante
$D(x, y) = |\mathsf{OPT}(x) - \varphi(x, y)| \leq D$

Raro

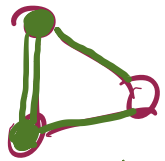
Coloração de grafos planares

- *Aproximação relativa* garante *erro relativo*
$E(x, y) = D(x, y)/\max\{\mathsf{OPT}(x), \varphi(x, y)\} \leq \epsilon \leq 1$ todos x.
  - Algoritmo ϵ-aproximativo.
  - Classes de problemas com ϵ-aproximação em tempo polinomial: APX.
- Alternativa: *taxa de aproximação*
$R(x, y) = 1/(1 - E(x, y)) \geq 1$.

Algoritmo guloso por vértices

$\Delta(G)$: grau máx.

$G=(V,A)$
$C \subseteq V$ t.q.
$a \in A$: $a \cap C$.
$(a=\{u,v\})$

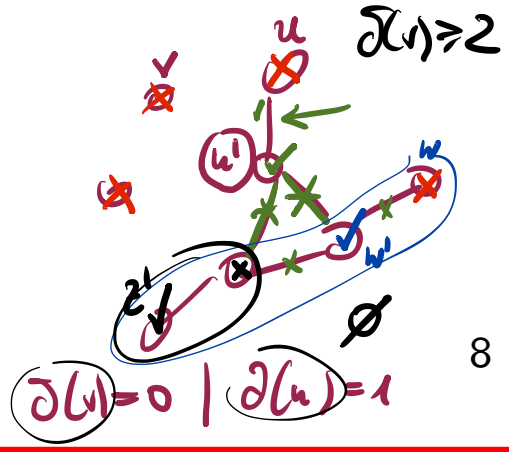


$|C|=2$

min. $|C|$

$\Rightarrow$ NP-completo.

```
VC-GV(G) :=
    (C, G) := Reduz(G)
    if V = ∅ then
        return C
    else
        escolhe v ∈ V : deg(v) = Δ(G)
        return C ∪ {v} ∪ VC-GV(G − v)
    end if
```



Proposição 2.1

O algoritmo VC-GV é uma $O(\log |V|)$-aproximação.

$\frac{k}{OPT} \leq \log |V|$

Demonstração $G = G_0 \xrightarrow{S(G_0)} G_1 \xrightarrow{S(G_1)} G_2 \xrightarrow{S(G_2)} G_3 \cdots \triangleright G_k$ | $|C^*| = OPT(G) = OPT$

C^* C^* C^*

Obs. $\sum_{v \in C^*} \delta_{G_i}(v) \geq \|G_i\|$ | $\|G\| = \#$ arestas.

$\bar{\delta}_{G_i}(G_i) = \frac{\sum_{v \in C^*} \delta_{G_i}(v)}{|C^*|} \geq \frac{\|G_i\|}{|C^*|} = \|G_i\| / OPT.$

$\Delta(G_i) \geq \|G_i\| / OPT = \bar{\delta}_{G_i}(G_i)$

Demonstração

Considere as las OPT iterações $G_0, G_1, \ldots, G_{OPT}$

$$\sum_{0 \le i < OPT} \Delta(G_i) \geq \sum \|G_i\| / OPT \geq \sum \|G_{OPT}\| / OPT = \|G_{OPT}\|$$

$$= \|G\| - \sum \Delta(G_i)$$

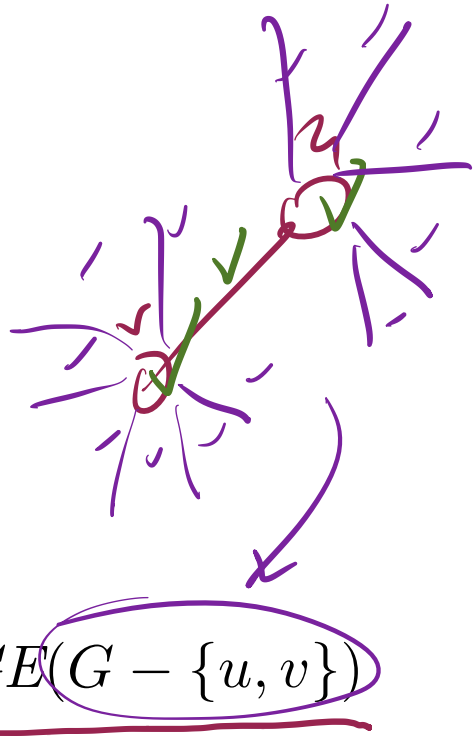
$$\sum_{0 \le i < OPT} \Delta(G_i) \geq \|G\|/2. \Rightarrow \text{nas próximas iterações.}$$

Depois: $OPT \lceil \lg \|G\| \rceil = O(OPT \log |V|)$ iterações
já tem mais arestas.

$$\frac{OPT \log |V|}{OPT} \text{ vértices} = \log |V|.$$

Algoritmo guloso por arestas

$VC\text{-}GE(G) :=$
  $(C, G) := Reduz(G)$
  $if\ E = \emptyset\ then$
    $return\ C$
  $else$
    $escolhe\ e = \{u, v\} \in E$
    $return\ C \cup \{u, v\} \cup VC\text{-}GE(G - \{u, v\})$
  $end\ if$



Proposição 2.2

Algoritmo VC-GE é uma 2-aproximação para VC.

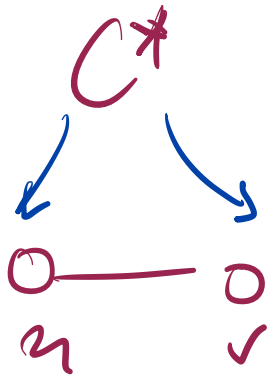
$\phi(G)$

Demonstração:

$|C| \geq \phi(G)/2$

cob.

$2OPT \geq 2|C| \geq \phi(G)$ ∎



Demonstração

Satisfatibilidade máxima, Maximum SAT

Instância Uma fórmula $\varphi \in \mathcal{L}(V)$ sobre variáveis $V = \{v_1, \ldots, v_m\}$, $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ em FNC.

Solução Uma atribuição de valores de verdade $a : V \to \mathbb{B}$.

Objetivo Maximiza o número de cláusulas satisfeitas

$$|\{C_i \mid |[C_i|]_a = 1\}| .$$

$v_1\ v_2\ \ldots\ v_k$
$0\ 1\ 0\ 0\ 1\ 0\ 0$

$$
\begin{array}{l}
SAT{-}R(\varphi) := \\
\quad seja\ \varphi = \varphi(v_1, \ldots, v_k) \\
\quad for\ all\ i \in [1, k]\ do \\
\quad\quad escolhe\ v_i = 1\ com\ probabilidade\ 1/2 \\
\quad end\ for
\end{array}
$$

Teorema 3.1

$[\![C]\!]$ var. aleatória $\to 1$, C sat. 0, c.c.

Algoritmo SAT-R é 2-aproximação. (valor esperado)

Demonstração: cláusula $C = \ell_1 \vee \ell_2 \vee \ell_3$ com ℓ literais

$x_1 \vee \bar{x}_2 \vee x_3$

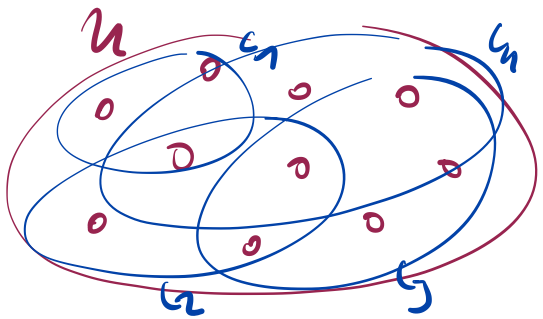
$E[[\![C]\!]] = Pr([\![C]\!]=1) = 1 - 2^{-\ell}$ $\quad \ell \geq 1$

$1 - 2^{-3} = 7/8.$

de cláusulas sat. $T = \sum_{i \in [n]} [\![C_i]\!] \geq n/2.$

$E[T] = E\left[\sum_{i \in [n]} [\![C_i]\!]\right] = \sum_{i \in [n]} E[[\![C_i]\!]] \geq n/2 \geq OPT/2.$

NP-completo.



Considere o problema de cobertura por conjuntos $C_1, \ldots, C_4$

$$\text{minimiza} \quad \sum_{i \in [n]} w_i x_i, \tag{1}$$

$$\text{sujeito a} \quad \sum_{i \in [n] \mid u \in C_i} x_i \geq 1, \qquad \forall u \in U,$$

$$x_i \in \{0, 1\}, \qquad \forall i \in [n].$$

P1

P2 Rel. lin. $0 \leq x_i \leq 1$.

$x_3 = 0.7$

Seja f_e a frequência de um elemento e, i.e. o número de conjuntos que contém e e f a maior frequência. Um algoritmo de arredondamento simples é dado por

Teorema 4.1

A seleção dos conjuntos com $x_i \geq 1/f$ na relaxação linear de (1) é uma f-aproximação do problema de cobertura de conjuntos.

Demonstração

$R = V \Rightarrow$ AGM $\in P$.

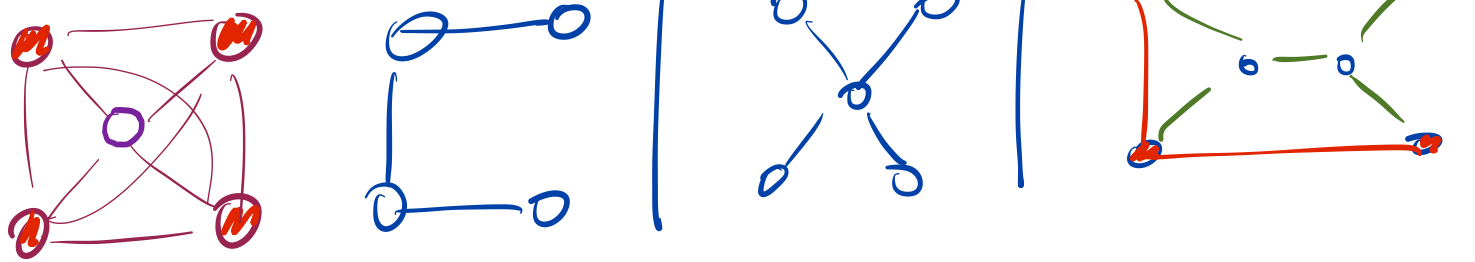
ST NP-C.

- Seja $G = (V, A)$ um grafo completo, não-direcionado com custos $c_a \geq 0$ nos arcos
- Encontra o subgrafo conexo mínimo que inclui um dado conjunto de *vértices necessários* ou *terminais* $R \subseteq V$.
- Esse subgrafo sempre é uma árvore.
- O conjunto $V \setminus R$ forma os *vértices Steiner*.
- Para um conjunto de arcos A, define o custo $c(A) = \sum_{a \in A} c_a$.

$S \subseteq V \setminus R$?

AGM: $S \cup R$.

R



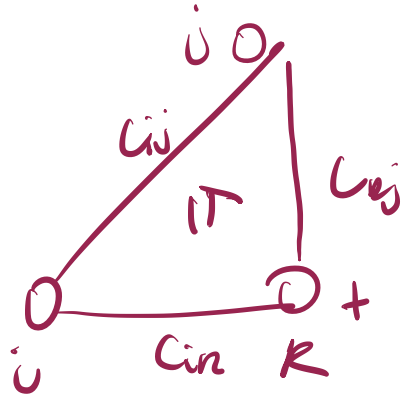
TSP ~~TSP~~ não possui apr. constante → TSP métrico: possui 3/2-apr.

Definição 5.1

Os custos são *métricos* se eles satisfazem a desigualdade triangular, i.e.

$$c_{ij} \leq c_{ik} + c_{kj}$$

para qualquer tripla de vértices i, j, k.



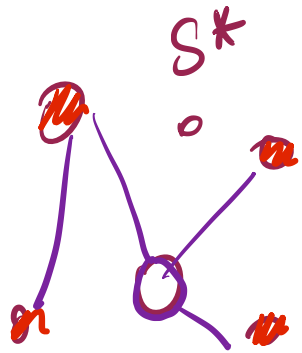
Teorema 5.1

Existe uma redução preservando a aproximação de ASM para a versão métrica do problema.

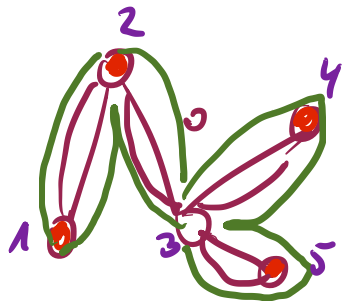
Teorema 5.2

O AGM sobre R é uma 2-aproximação para o problema do ASM.

Demonstração: Sol. ótima ASM. S^* : $c(A) \leq 2c(S^*)$



Duplica as arestas

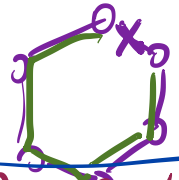


Cic. Euleriano C

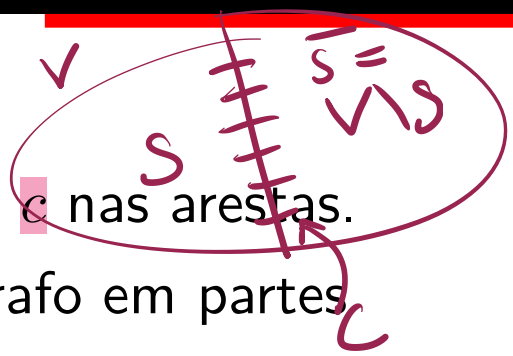
$c(C) = 2c(S^*)$

$c(C') \leq c(C)$

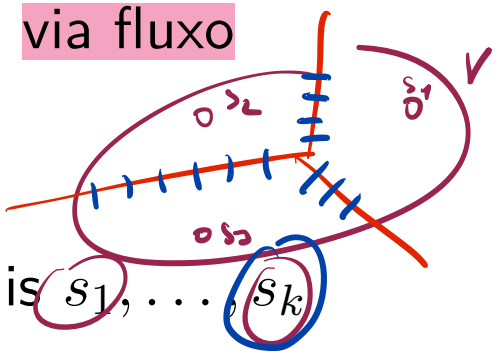
C' \ aresta é uma geradora sobre R



$c(A) \leq c(C') \leq c(C) = 2c(S^*)$

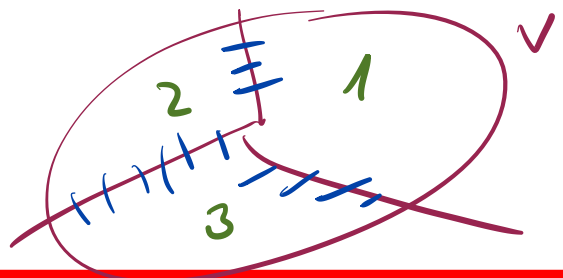


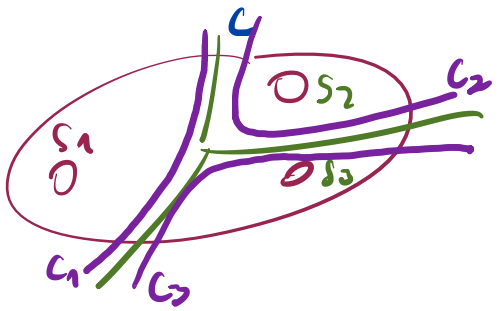
- Dado: grafo conexo $G = (V, A, c)$ com pesos c nas arestas.
- Corte C: conjunto de arestas que separa o grafo em partes $S \dot\cup V \setminus S$.
- Dado $s, t \in V$: corte mínimo separando s e t: via fluxo máximo em tempo polinomial.



Generalizações desse problema são:

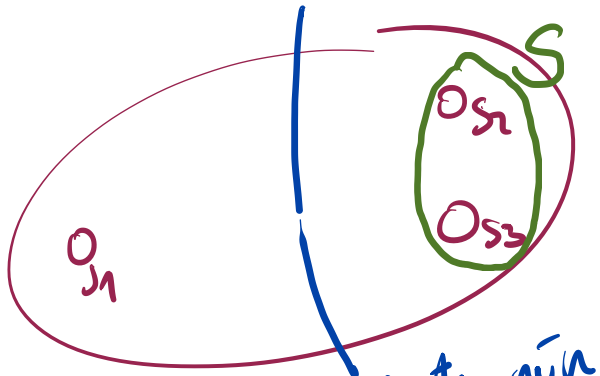
- Corte múltiplo mínimo (CMM): Dado terminais $s_1, \ldots, s_k$ determine o menor corte C que separa todos.
- k-corte mínimo (k-CM): Mesmo problema, sem terminais definidos. (Observe que todos k componentes devem ser não vazios).





$C = C_1 \cup C_2 \cup C_3$

- *Corte isolante*: separa um terminal dos demais.
- *Idéia*: A união de cortes isolantes para todo s_i é um corte múltiplo.
- Para calcular o corte isolante para um dado terminal s_i
  - Identifica os restantes terminais num único vértice S.
  - Calcula um corte mínimo entre s_i e S.



Algoritmo CMM:

Para cada $i \in [1, k]$*: Calcula o corte isolante* C_i *de* s_i*.*

Remove o maior desses cortes e retorna a união dos restantes.

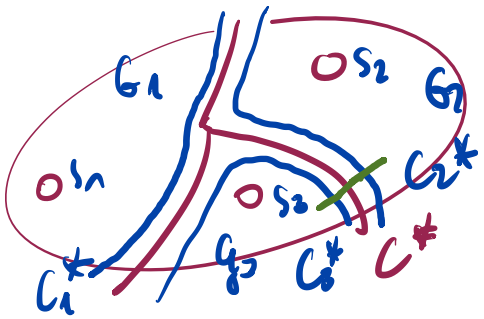
Teorema 6.1

Algoritmo CMM é uma $2 - 2/k$-aproximação.

$C_1, C_2, \ldots, C_k$

Demonstração: corte mín. C^* ; corte obtido C pelo alg.

a.d. $c(C) \leq (2-2/k)\, c(C^*)$



$C_i^* \leq C^*$

$C^* = \bigcup_{i \in [k]} C_i^*$

$\sum_{i \in [k]} c(C_i^*) = 2c(C^*)$.

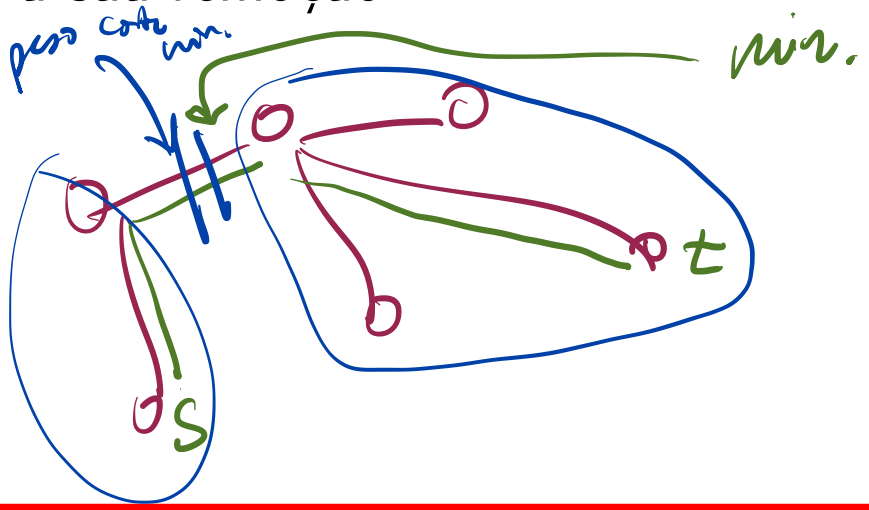
$c(C_1) \leq c(C_i^*)$

$$c(C) \leq (1-1/k) \sum_{i \in [k]} c(C_i) \leq (1-1/k) \sum_{i \in [n]} c(C_i^*) \leq 2(1-1/k)\, c(C^*)$$

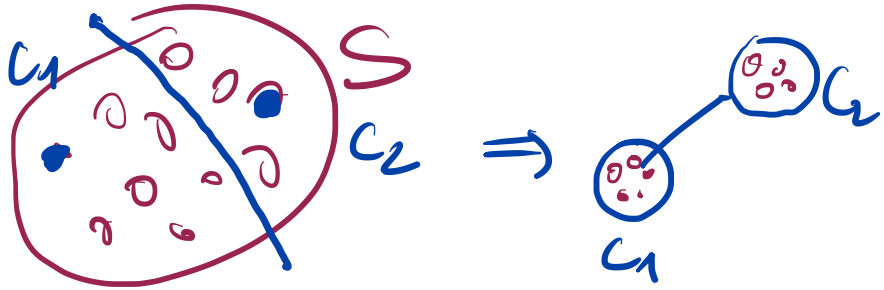
- Problema: Como saber a onde cortar?

Fato 6.1

Existem somente $n-1$ cortes diferentes num grafo. Eles podem ser organizados numa árvore de *Gomory-Hu* (AGH) $T = (V, T)$. Cada aresta dessa árvore define um corte associado em G pelos dois componentes após a sua remoção.



Calcular uma AGH



1. Escolhe um vértice gordo e dois vértices do grafo original que ele representa.
2. Calcula um corte mínimo entre esses vértices.
3. Separa o vértice gordo de acordo com o corte mínimo encontrado.

n-1 → árvore.

n-1 chamadas FM/CM.

Algoritmo kCM:

Calcula uma AGH T em G.

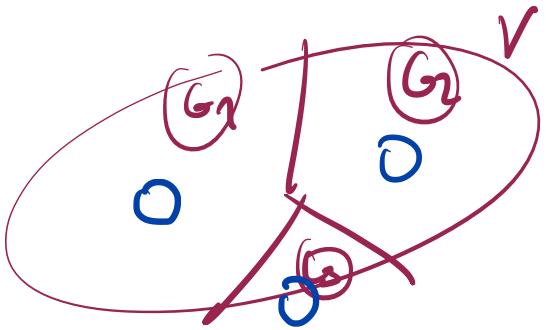
Forma a união dos $k-1$ cortes mais leves definidos por $k-1$ arestas em T.

Teorema 6.2

Algoritmo kCM é uma $2 - 2/k$-aproximação.

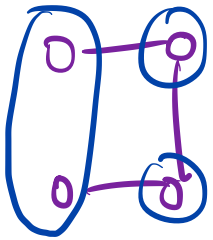
q.d. $c(C) \leq (2-2/k)\, c(C^*)$

Demonstração: $C^* = \bigcup_{i \in [k]} C_i^*$ corte mín. ; C obtido pelo alg.



Comp. $G_1, G_2, \ldots, G_k$

Contrai G_i para um único vértice. Inclui as arestas da AGM entre os comp. Remova arestas até obter uma árvore T.

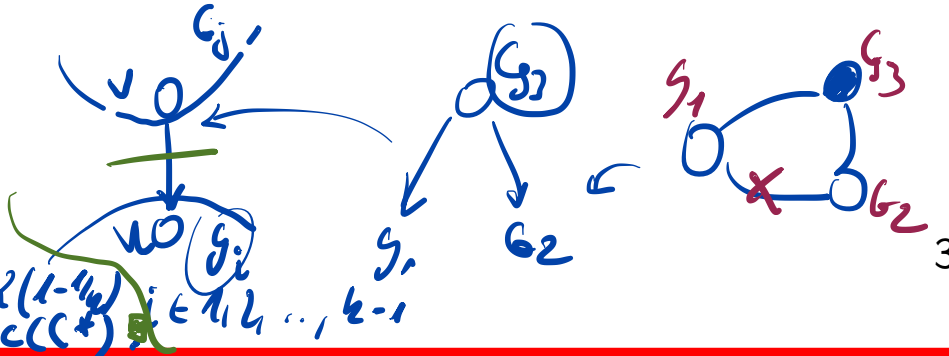


Seja C_k^* o corte isolante de menor peso : $\Rightarrow$ orienta T com raiz G_k

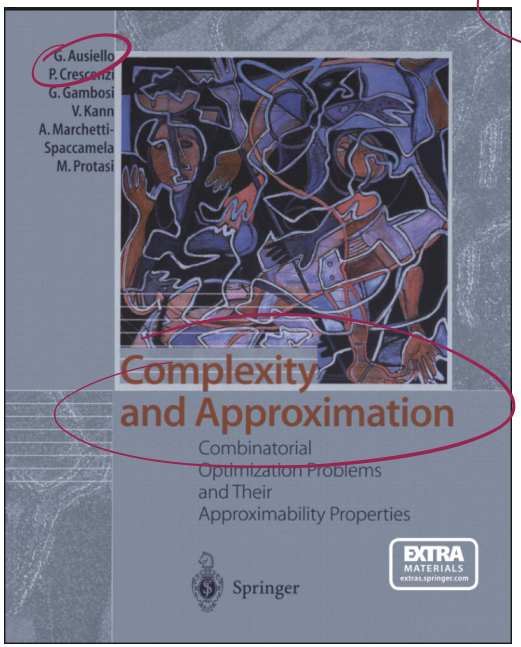
$c(C_i^*) \geq c(v,u)$

$c(C) \leq \sum_{a \in T} c(a) \leq \sum_{i=1}^{k-1} c(C_i^*)$

$\leq (1-1/k) \sum_{i \in [k]} c(C_i^*) \leq 2(1-1/k)\, c(C^*)$



Livros


G. Ausiello
P. Crescenzi
G. Gambosi
V. Kann
A. Marchetti-Spaccamela
M. Protasi
Complexity and Approximation
Combinatorial Optimization Problems and Their Approximability Properties
EXTRA MATERIALS
extras.springer.com
Springer


The Design of Approximation Algorithms
David P. Williamson
David B. Shmoys