UNIVERSIDADE DE CAMPINAS - UNICAMP
INSTITUTO DE COMPUTAÇÃO - IC

Algoritmos de Aproximação

Flávio Keidi Miyazawa

Campinas, 2001-2018

Sumário I

Sumário II

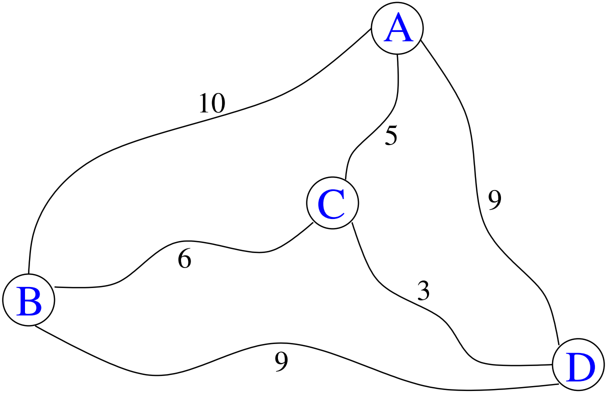
Problemas de Otimização Combinatória

- Domínio Finito (enumerável)
- Função de Otimização: Custos, Comprimentos, Quantidades
- Objetivo: Minimização ou Maximização

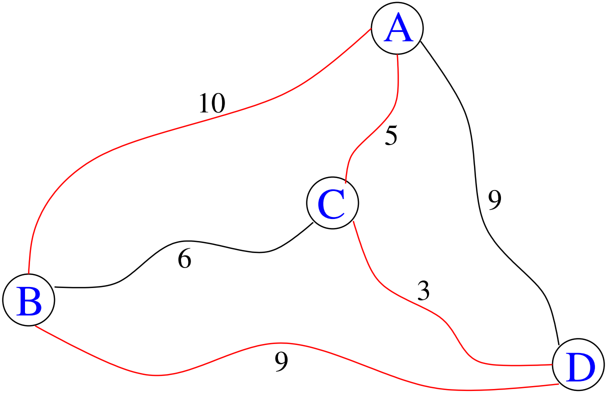
Exemplo: Problema do Caixeiro Viajante (TSP)

- *Entrada:*
  - Grafo não orientado: $G = (V, E)$,
  - Custos nas arestas: $c_e \geq 0, \forall e \in E$.
- *Objetivo:*
  Encontrar um *tour* (ciclo hamiltoniano) de custo mínimo que visita cada vértice exatamente uma vez.

Encontre o ciclo hamiltoniano de custo mínimo



ciclo hamiltoniano de custo mínimo: 27



Algoritmo Ingênuo para o TSP: Tentar todos os tours.
Complexidade: $O(n!)$, onde $n = |V|$.

Algoritmo *Bom* = Algoritmo Polinomial *(Cobham'64&Edmonds'65)*
Provavelmente não existam algoritmos eficientes para o TSP
Outros exemplos difíceis:

- Atribuição de Freqüências em Telefonia Celular
- Empacotamento de Objetos em Contêineres
- Escalonamento de Funcionários em Turnos de Trabalho
- Escalonamento de Tarefas em Computadores
- Classificação de Objetos
- Coloração de Mapas
- Projetos de Redes de Computadores
- Vários outros...

Complexidade Computacional

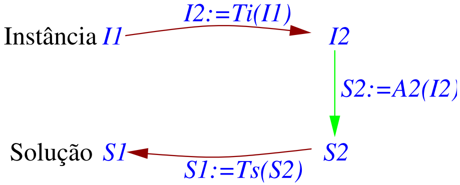
Problemas de Decisão

- Exemplo: Dado grafo G, existe um ciclo hamiltoniano em G ?
- Exemplo: Dado grafo completo $G = (V, E)$, função de peso nas arestas $c : E \to \mathbb{Z}^+$ e um inteiro positivo K, existe um ciclo hamiltoniano em G, de peso no máximo K ?
- Exemplo: Dado um mapa M e um inteiro K, posso colorir M com K cores sem conflitos ?
- Exemplo: Dado grafo completo $G = (V, E)$, pesos nas arestas $c : E \to \mathbb{Z}^+$, vértices s e t e um inteiro positivo K, existe um caminho em G, de s para t, de peso no máximo K ?

Redução polinomial entre problemas

P_1 é polinomialmente redutível a P_2 ($P_1 \preceq P_2$) se

- $\exists$ *Ti* que transforma instância I_1 de P_1 para instância I_2 de P_2
- $\exists$ *Ts* que transforma solução S_2 de I_2 para solução S_1 de I_1
- *Ti* e *Ts* têm complexidade de tempo polinomial



Conseqüências:

Se P_2 é "polinomial" então P_1 é "polinomial".
Se P_1 é "exponencial" então P_2 é pelo menos "exponencial".

Redução polinomial entre problemas de decisão

P_1 é polinomialmente redutível a P_2 ($P_1 \preceq P_2$) se

- $\exists$ T que transforma instância I_1 de P_1 para instância I_2 de P_2
- I_1 tem solução se e somente se I_2 tem solução
- T é polinomial

Conseqüências:

Se P_2 é "polinomial" então P_1 é "polinomial".
Se P_1 é "exponencial" então P_2 é pelo menos "exponencial".

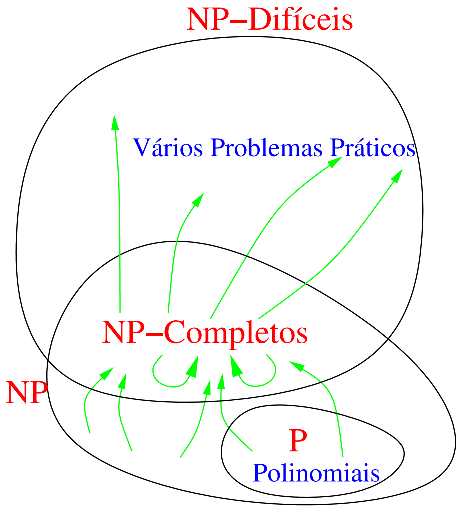
Classes de Complexidade:
P, NP, NP-Completo, NP-Difícil

$X \in$ P:
X pode ser decidido em tempo polinomial.

$X \in$ NP:
X tem certificado curto e verificável em tempo polinomial.

$X \in$ NP-Completo:
$X \in \text{NP}$ e $\forall Y \in NP,\ Y \preceq X$.

$X \in$ NP-Difícil:
$\forall Y \in \text{NP},\ Y \preceq X$, onde X não necessariamente pertence a NP.



Conjectura: P=NP ?

Prêmio de 1 milhão de US$.

Problemas em **P**

- Dado um conjunto de números S, existe $n \in S$ tal que $n = \sum_{i \in S \setminus \{n\}} i$?
- Decida se um dado grafo G é conexo ?
- Dado grafo completo $G = (V, E)$, pesos nas arestas $c : E \to \mathbb{Z}^+$, vértices s e t e um inteiro positivo K, existe um caminho em G, de s para t, de peso no máximo K ?
- Posso colorir um mapa com 4 cores sem conflitos ?

Problemas em **NP-Completo**

- Dado um conjunto de números S, existe $N \subseteq S$ tal que $\sum_{i \in N} i = \sum_{i \in S \setminus N} i$?
- Dado grafo G, existe um ciclo hamiltoniano em G ?
- Dado grafo completo $G = (V, E)$, pesos nas arestas $c : E \to \mathbb{Z}^*$ e um inteiro positivo K, existe um ciclo hamiltoniano em G, de peso no máximo K ?
- Dado grafo completo $G = (V, E)$, pesos nas arestas $c : E \to \mathbb{Z}$, vértices s e t e um inteiro positivo K, existe um caminho em G, de s para t, de peso no máximo K ?
- Posso colorir um mapa com 3 cores sem conflitos ?

Problemas NP-difíceis

- Vários problemas práticos são NP-difíceis
  Exemplos:
  - Problema do Caixeiro Viajante
  - Atribuição de Freqüências em Telefonia Celular
  - Empacotamento de Objetos em Contêineres
  - Escalonamento de Funcionários em Turnos de Trabalho
  - Escalonamento de Tarefas em Computadores
  - Classificação de Objetos
  - Coloração de Mapas
  - Projetos de Redes de Computadores
  - Vários outros...
- **P$\neq$NP $\Rightarrow$ não existem algoritmos eficientes para problemas NP difíceis**

Comparando tempos polinomiais e exponenciais

$f(n)$	$n = 20$	$n = 40$	$n = 60$	$n = 80$	$n = 100$
n	$2{,}0\times10^{-11}$seg	$4{,}0\times10^{-11}$seg	$6{,}0\times10^{-11}$seg	$8{,}0\times10^{-11}$seg	$1{,}0\times10^{-10}$seg
n^2	$4{,}0\times10^{-10}$seg	$1{,}6\times10^{-9}$seg	$3{,}6\times10^{-9}$seg	$6{,}4\times10^{-9}$seg	$1{,}0\times10^{-8}$seg
n^3	$8{,}0\times10^{-9}$seg	$6{,}4\times10^{-8}$seg	$2{,}2\times10^{-7}$seg	$5{,}1\times10^{-7}$seg	$1{,}0\times10^{-6}$seg
n^5	$2{,}2\times10^{-6}$seg	$1{,}0\times10^{-4}$seg	$7{,}8\times10^{-4}$seg	$3{,}3\times10^{-3}$seg	$1{,}0\times10^{-2}$seg
2^n	$1{,}0\times10^{-6}$seg	1,0seg	13,3dias	$1{,}3\times10^{5}$séc	$1{,}4\times10^{11}$séc
3^n	$3{,}4\times10^{-3}$seg	140,7dias	$1{,}3\times10^{7}$séc	$1{,}7\times10^{19}$séc	$5{,}9\times10^{28}$séc

Supondo um computador com velocidade de 1 Terahertz (mil vezes mais rápido que um computador de 1 Gigahertz).

Comparando tempos polinomiais e exponenciais

$f(n)$	Computador atual	100×mais rápido	1000×mais rápido
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5 + 6.64$	$N_5 + 9.97$
3^n	N_6	$N_6 + 4.19$	$N_6 + 6.29$

Fixando o tempo de execução

Algoritmos de Aproximação

- **Abordagem para tratar problemas NP-difíceis**
  - Problemas NP-difíceis precisam de alguma solução
- **Algoritmos eficientes (polinomiais)**
  - Maioria dos algoritmos de aproximação são rápidos
  - Adequados para instâncias grandes onde exatos são proibitivos
- **Garantia de "proximidade" com a solução ótima**
  - Análise formal da qualidade das soluções geradas
  - Análises formais indicam como medir qualidade de soluções viáveis quaisquer
- **Novas classes de complexidade em NP-difícil**
  - Problemas em NP-completo são polinomialmente equivalentes, mas
  - problemas de otimização apresentam dificuldades distintas sob o ponto de vista de aproximação
- **Agregar valor teórico a heurísticas (conhecidas)**
  - Área acadêmica valoriza análises formais

- **Inspiração para o desenvolvimento de heurísticas**
  – Heurísticas aproveitam propriedades estruturais de algoritmos aproximados
- **Desenvolvimento de teorias recentes**
  – Área que tem recebido grande atenção pela comunidade acadêmica
  – Novas técnicas no desenvolvimento de algoritmos
  – Teorias revelando inaproximabilidade de problemas
- **Desenvolvimento de algoritmos exatos**
  – Algoritmos exatos usam soluções viáveis para limitar busca
  – Algoritmos exatos baseados em enumerações e algoritmos de aproximação
- **Teoria dos Jogos**
  – Mesmo tipo de análise para o *Preço da Anarquia*
- **Análise de algoritmos online**
  – Algoritmos onde não há informação sobre próximos eventos

Experimentos Computacionais

- **Williamson'93**
  Problema do Emparelhamento: Grafos de até 131.000 vértices, em torno de 2% do ótimo
- **Goemans & Willamson'94**
  Problema do Corte Máximo: Grafos de até 200 vértices, em torno de 4% do ótimo
- **Homer & Peinado'94**
  Problema do Corte Máximo: Grafos de até 13.000 vértices
  Implementação distribuida do algoritmo de Goemans & Williamson
- **Hall'95**
  Problema da Árvore de Steiner: Grafos de até 1000 nós / 60.000 arestas em torno de 7% do ótimo

- **Hu & Wein**
  Problema da Floresta de Steiner: Grafos com até 64 vértices, em torno de 5% do ótimo
- **Johnson, Minkoff, Phillips'00**
  Problema da Árvore de Steiner com Penalidades: Grafos com até 76.000 vértices. Soluções próximas da ótima
- **Barahona & Chudak'99**
  Facility Location Problem: Técnica híbrida em grafos com 3000 facilidades e 3000 clientes, em torno de 1% do ótimo
- **Mihail, Mostrel, Dean & Shallcross'95**
  Survivable Network Design Problem: Software da Bellcore. Soluções próximas da ótima.

Grandes Desafios

- Instâncias em circuitos VLSI
  Johnson'90: Problema do Caixeiro Viajante em Grafos com 1.2 milhão de vértices (e aumentando)
  Outras instâncias grandes ocorrem em cristalografia
- **Limitantes caros**
  Para muitas instâncias não é nem possível se resolver programas lineares/semidefinidos em tempo razoável (muito usados para desenvolver algoritmos de aproximação)
- **Mundo real tem diversas instâncias grandes**
  Atualmente são resolvidos em partes.
  Exemplos: Projetos de circuitos VLSI, escalonamento de trabalhadores, localização de equipamentos em telecomunicações...

Principais Técnicas

- Métodos Combinatórios
- Programação Dinâmica
- Métodos baseados em Progamação Linear
- Métodos Probabilísticos
- Programação Semidefinida
- Técnicas de Inaproximabilidade

Alguns Problemas

- Escalonamento de Tarefas (Scheduling)
- Cobertura por Conjuntos (Set Cover)
- Problema da Mochila (Knapsack)
- Empacotamento de Barras (Bin Packing)
- Caixeiro Viajante (TSP)
- Floresta de Steiner
- Satisfatibilidade Máxima (Max-SAT)
- Corte Máximo (Max-CUT)
- Clique Máximo (Max-Clique)
- Localização de Facilidades

Medidas em Algoritmos de Aproximação

- Aproximação Absoluta
- Fator de Aproximação
- Fator de Aproximação Assintótico
- Fator de Aproximação Probabilístico

Aproximação Absoluta

Dado um algoritmo A e instância I,

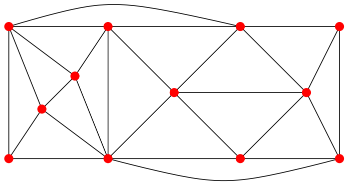
- OPT(I) é o tamanho da solução ótima
- $A(I)$ é o tamanho da solução gerada por A sobre I

A tem *aproximação absoluta k* se

$$|A(I) - \mathrm{OPT}(I)| \leq k \qquad \forall \text{ instância } I.$$

Coloração de Grafos Planares

Def.: *Um grafo $G = (V, E)$ é dito ser planar se é possível desenhá-lo no plano, tal que duas arestas só se interceptam em pontos de V que são suas extremidades em comum.*



Problema COLORAÇÃO-PLANAR: Dado um grafo planar simples $G = (V, E)$, colorir os vértices de G com o menor número de cores, onde vértices adjacentes devem ter cores distintas.

Teorema: COLORAÇÃO-PLANAR é NP-difícil.

Teorema: *Existe uma 1-aproximação absoluta para o problema da* COLORAÇÃO-PLANAR *(Appel & Haken'76)*

Uma 3-aproximação absoluta

Idéia do algoritmo: Indução com o seguinte teorema

Teorema: *Se G é planar simples, então G tem pelo menos um vértice de grau* ≤ 5

Prova. O Teorema de Euler para grafos planares conexos, diz que

$$|E| = |V| + |F| - 2,$$

onde F é o conjunto de faces de uma imersão de G no plano.
Exercício: Prove este teorema.

Usando este teorema, podemos provar que

$$|E| \leq 3|V| - 6 \quad \text{(Exercício)}$$

Usando a relação do número de arestas e a soma total de graus, o resultado segue (Exercício).

$\square$

ALGORITMO 6-CORES $G = (V, E)$

```
Se E = ∅ então,
    cor(v) := 1,      ∀v ∈ V.
senão, se G é bipartido, V = (X, Y) então
    cor(x):= 1, ∀x ∈ X   e   cor(y):= 2, ∀y ∈ Y.
senão,
    seja v ∈ V tal que grau(v) ≤ 5
    Pinte G' := G − v recursivamente.
    Seja c ∈ {1, ..., 6} cor não presente nos adjacentes de v
    cor(v) := c.
```

Teorema: *6-Cores é uma 3-aproximação absoluta.*

Prova. Se $E = \emptyset$ ou G é bipartido,

$$\text{6-Cores}(G) = \mathrm{OPT}(G)$$

caso contrário, $\mathrm{OPT}(G) \geq 3$

$$\text{6-Cores}(G) - \mathrm{OPT}(G) \leq 3.$$

$\square$

Uma 2-aproximação absoluta

Teorema: *Descreva um algoritmo 5-Cores que é uma 2-aproximação absoluta.*

Prova. Exercício.
Idéia: Baseado no algoritmo 6-Cores. Seja v o vértice removido.
• Se vizinhos de v usam ≤ 4 cores, pinte v com cor faltante.
• Caso contrário, considere uma imersão de G no plano.
S.P.G. seja $1, \ldots, 5$ os vizinhos de v, numerados no sentido horário em torno de v e seja i e j dois vizinhos não consecutivos (na ordem cíclica) de v.

Tente recolorir i com a cor de j trocando as duas cores a partir de i. Se for possível, você terá uma cor disponível para v. Caso contrário, procure repintar outros vértices. □

Coloração de Arestas

Problema COLORAÇÃO DE ARESTAS: Dado um grafo G, colorir as arestas de G com o menor número de cores, onde arestas adjacentes devem ter cores distintas.

Aplicações: Escalonamento com horários, coloração de fios em circuitos, etc.

Teorema: *(Holyer'81) O problema de Coloração de Arestas é NP-difícil.*

Teorema: *(Vizing'64) É possível colorir as arestas de G com $\Delta(G) + 1$ cores, onde $\Delta(G)$ é o maior grau de um vértice em G.*

Corolário: *Existe um algoritmo com aproximação absoluta 1 para o problema de Coloração de Arestas.*

Resultados Negativos: Sensibilidade a escala

Problema da Mochila

Problema MOCHILA: Dados itens $S = \{1, \ldots, n\}$ com peso v_i e tamanho s_i inteiros, $i = 1, \ldots, n$, e inteiro B, encontrar $S' \subseteq S$ que maximiza $\sum_{i \in S'} v_i$ tal que $\sum_{i \in S'} s_i \leq B$.

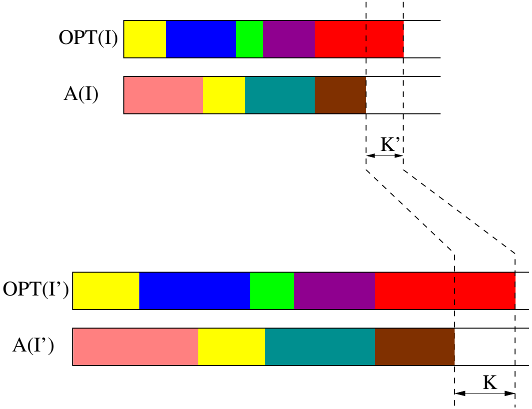
Teorema: *Se $P \neq NP$ então não existe algoritmo de tempo polinomial com aproximação absoluta k para o problema da Mochila, para qualquer k fixo.*

Prova. Suponha existir algoritmo A e k tal que

$$|\mathrm{OPT}(I) - A(I)| \leq k,$$

e k limitado por polinômio de n.

Seja $I := \{S = (1, \ldots, n), v = (v_1, \ldots, v_n), (s_1, \ldots, s_n), B\}$ uma instância qualquer.
Seja $I' := \{S = (1, \ldots, n), v = (v'_1, \ldots, v'_n), (s_1, \ldots, s_n), B\}$ uma instância onde $v'_i := (k+1) \cdot v_i$.



$$(k+1)\mathrm{OPT}(I) = \mathrm{OPT}(I')$$

Seja *Sol* a solução de I correspondente ao gerado por A em I'.

$$|\mathrm{OPT}(I') - A(I')| \leq k$$

$\Longrightarrow$

$$|(k+1)\cdot \mathrm{OPT}(I) - (k+1)Sol| \leq k$$

$\Longrightarrow$

$$(k+1)\cdot |\mathrm{OPT}(I) - Sol| \leq k.$$

$\Longrightarrow$

$$|\mathrm{OPT}(I) - Sol| \leq \frac{k}{k+1}.$$

$\Longrightarrow$

$$|\mathrm{OPT}(I) - Sol| \leq 0.$$

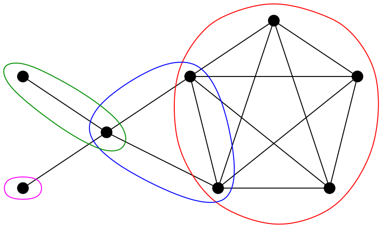
$\Longrightarrow$

$$\mathrm{OPT}(I) = Sol$$

$\square$

Problema do Clique Máximo

Def.: *Dado um grafo $G = (V, E)$, um conjunto $S \subseteq V$ é um clique se $\forall u, v \in S \Rightarrow \{u, v\} \in E$.*



Problema CLIQUE: Dado um grafo G, encontrar um clique em G de cardinalidade máxima.

Teorema: *(Karp'72)* CLIQUE *é um problema NP-difícil.*

Inaproximabilidade absoluta do Clique Máximo

Def.: *Seja G^m grafo com m cópias de G e vértices de cópias distintas ligados.*

Fato: *Se o maior clique de G tem tamanho k então, o maior clique de G^m tem tamanho $k \cdot m$.*

Teorema: *Se $P \neq NP$ então não existe algoritmo de tempo polinomial com aproximação absoluta k para o problema do Clique, para qualquer valor k limitado por polinômio de n.*

Prova.

Suponha existir algoritmo A e constante k tal que

$$|\mathrm{OPT}(G) - A(G)| \leq k \qquad \forall \text{ instância } I$$

Então

$$\begin{aligned}|\mathrm{OPT}(G^{k+1}) - A(G^{k+1})| &\leq k \\ |(k+1)\cdot\mathrm{OPT}(G) - A(G^{k+1})| &\leq k\end{aligned}$$

Seja *Sol* o maior clique em uma cópia feita por $A(G^{k+1})$.
Sol tem tamanho pelo menos $\frac{A(G^{k+1})}{k+1}$.
$\Longrightarrow$

$$|(k+1)\cdot\mathrm{OPT}(G) - (k+1)Sol| \leq k.$$

$\Longrightarrow$

$$|(k+1)(\mathrm{OPT}(G) - Sol)| \leq k.$$

$\Longrightarrow$

$$|\mathrm{OPT}(G) - Sol| \leq \frac{k}{k+1}.$$

$\Longrightarrow$

$$|\mathrm{OPT}(G) - Sol| \leq 0.$$

$\Longrightarrow$

$$Sol = \mathrm{OPT}(G)$$

Fator de Aproximação

Dado um algoritmo A e instância I,

- $\mathrm{OPT}(I)$ é o tamanho da solução ótima
- $A(I)$ é o tamanho da solução gerada pelo algoritmo A
- A é algoritmo de aproximação com fator α se

$$A(I) \leq \alpha \cdot \mathrm{OPT}(I) \qquad \forall I, \quad \text{para minimização}$$

$$A(I) \geq \alpha \cdot \mathrm{OPT}(I) \qquad \forall I, \quad \text{para maximização}$$

- A tem fator α justo se $\forall \epsilon > 0$ temos que

$$\exists I \quad \text{tal que} \quad A(I) > (\alpha - \epsilon)\mathrm{OPT}(I), \quad \text{para minimização}$$

$$\exists I \quad \text{tal que} \quad A(I) < (\alpha + \epsilon)\mathrm{OPT}(I), \quad \text{para maximização}$$

Escalonamento de Tarefas

Problema ESCALONAMENTO: Dados uma lista de tarefas $L = (J_1, \ldots, J_n)$, cada uma com tempo $t(J_i)$ e m máquinas idênticas, encontrar uma partição de L, $(M_1, \ldots, M_m)$, tal que $\max_i t(M_i)$ é mínimo.

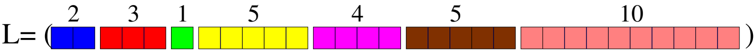
Teorema: ESCALONAMENTO é NP-difícil.

Algoritmo de Graham: Escalonar a próxima tarefa na máquina com menos tempo.

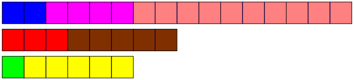
GRAHAM (m, n, t)

```
para j de 1 a m faça M_j ← ∅
para i de 1 a n faça
    seja k uma máquina tal que Σ_{i∈M_k} t_i é mínimo
    M_k ← M_k ∪ {i}
devolva {M_1, ..., M_m}
```

Exemplo:

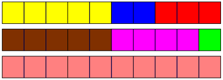


Algoritmo Graham



Graham(L) = 16

Escalonamento Ótimo

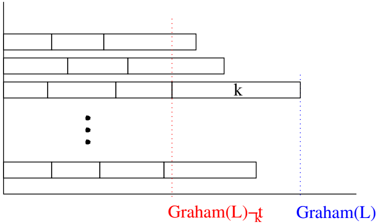


OPT(L) = 10

Teorema: $Graham(L) \leq 2\,\mathrm{OPT}(L)$.

Prova.

Seja J_k a última tarefa terminada no escalonamento.



$$\begin{aligned} \mathrm{OPT}(L) &\geq \max_i t(J_i) &&\geq t(J_k) \\ \mathrm{OPT}(L) &\geq \frac{1}{m}\sum_i t(J_i) &&\geq \mathrm{Graham}(L) - t(J_k) \end{aligned}$$

I.e., $2 \cdot \mathrm{OPT}(m, n, t) \geq \mathrm{Graham}(L)$.

$\square$

Algoritmo LPT - Longest Processing Time

LPT (m, n, t)

1 Ordene as tarefas tal que $t_1 \geq t_2 \geq \ldots \geq t_n$.

2 devolva Graham (m, n, t)

Teorema: $LPT(m, n, t) \leq \frac{3}{2}\mathrm{OPT}(m, n, t)$.

Prova.
Seja k a tarefa que terminou por último no escalonamento LPT.

Se $n \leq m$ ou $t_k =$LPT(m, n, t) então LPT$(m, n, t) = \mathrm{OPT}(m, n, t)$.

Senão, temos que $\mathrm{OPT}(m, n, t) \geq 2t_k$. I.e., $t_k \leq \frac{\mathrm{OPT}(m,n,t)}{2}$.

Sabemos que: $\mathrm{LPT}(L) \leq \frac{\sum_i t_i}{m} + t_k, \quad \frac{\sum_i t_i}{m} \leq \mathrm{OPT}(L)$ e $t_k \leq \frac{\mathrm{OPT}(m,n,t)}{2}$

Assim, $\mathrm{LPT}(L) \leq \mathrm{OPT}(L) + \frac{\mathrm{OPT}(m,n,t)}{2} = \frac{3}{2}\mathrm{OPT}(m, n, t)$ □

Esta análise é justa ?

Teorema: $LPT(m, n, t) \leq \frac{4}{3}\mathrm{OPT}(m, n, t)$.

Prova. Por contradição. Suponha que exista instância tal que $\frac{\mathrm{LPT}(L)}{\mathrm{OPT}(L)} > \frac{4}{3}$. Considere tal instância com menor número de tarefas.

Por ser contra-exemplo com menor número de tarefas, temos que t_n é a última tarefa a ser executada.

$$\mathrm{LPT}(L) \leq \frac{\sum_i t_i}{m} + t_n \quad \text{e} \quad \mathrm{OPT}(L) \geq \frac{\sum_i t_i}{m}$$

Assim, $\mathrm{LPT}(L) \leq \mathrm{OPT}(L) + t_n$.

Isto é, $\frac{4}{3} < \frac{\mathrm{LPT}(L)}{\mathrm{OPT}(L)} \leq 1 + \frac{t_n}{\mathrm{OPT}(L)}$.

O que nos dá: $\mathrm{OPT}(L) < 3t_n$.

Como t_n é a menor tarefa, todas as máquinas tem no máximo 2 tarefas. Segue que o escalonamento obtido por LPT é ótimo, dando uma contradição (Exercício). $\square$

Lema: *O fator $\frac{4}{3}$ do algoritmo LPT é justo.*

Prova. Exercício.

Considere $m = 4$ e $n = 9$ tarefas, onde $t_1 = t_2 = 7, \quad t_3 = t_4 = 6,$ $t_5 = t_6 = 5, \quad t_7 = t_8 = 4, \quad \text{e} \quad t_9 = 4$. Extrapole para uma instância com m maior e veja o limite de $\frac{\text{LPT}}{\text{OPT}}$. □

Notas adicionais

Algoritmo k-GRAHAM (m, n, t)

```
considere que t_1, ... t_k são as k maiores tarefas da instância
obtenha um escalonamento ótimo S_k das k primeiras tarefas
escalone as n − k tarefas restantes sobre S_k usando Graham
devolva o escalonamento obtido
```

Teorema: *k-Graham$(L) \leq (1+\xi)\,\mathrm{OPT}(L)$, onde $\xi = \frac{1-1/m}{1+k/m}$.*

Teorema: *Dado $\epsilon > 0$, existe um algoritmo com fator de aproximação $(1+\epsilon)$ para o problema de* ESCALONAMENTO.

Cobertura por Vértices

Problema COBERTURA POR VÉRTICES: Dados grafo $G = (V, E)$, encontrar $C \subseteq V$ tal que $\{u, v\} \cap C \neq \emptyset \quad \forall \{u, v\} \in E$ e $|C|$ é mínimo.

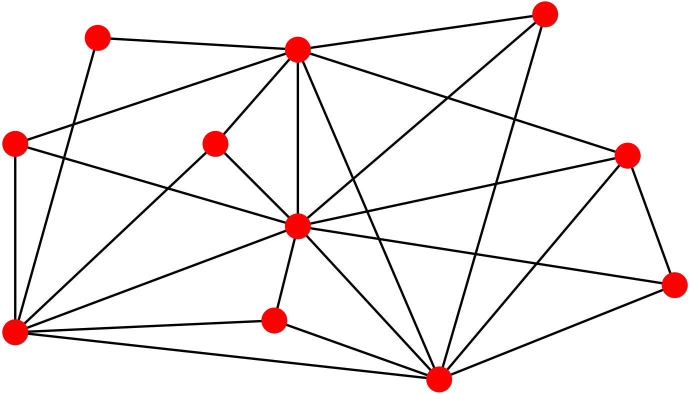
Aplicação: Dado um museu, dispor um número mínimo de guardas que cubram todas os corredores do museu.

Resultados Negativos

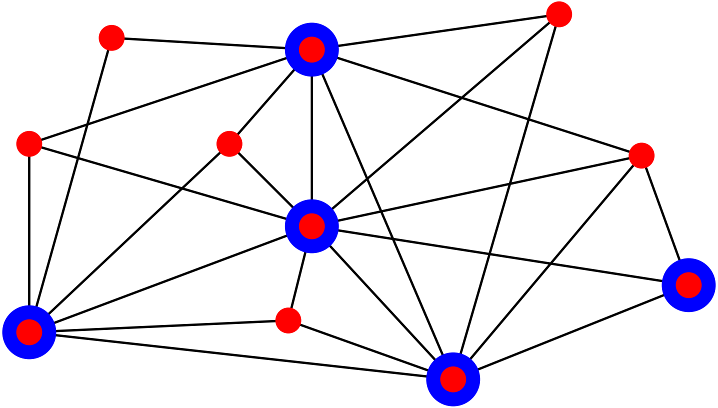
Teorema: COBERTURA POR VÉRTICES é NP-difícil.

Teorema: Se existir uma α-aproximação para o problema da cobertura por vértices com $\alpha < 7/6$ então P=NP *(Hastad'97)*.

Encontre uma cobertura por vértices:

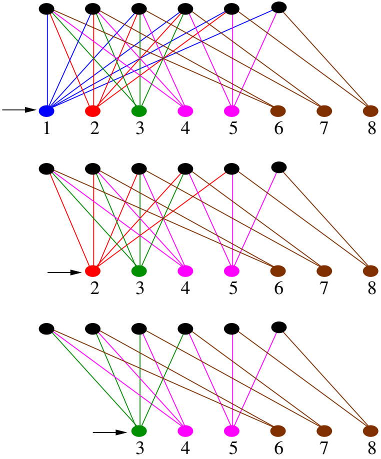


Cobertura por Vértices:



Lema: Algoritmo guloso que iterativamente remove (escolhe) o vértice de maior grau tem fator de aproximação $\Omega(\log(n))$

Prova. Exercício. Idéia: Considere o grafo bipartido abaixo e a remoção dos vértices na partição do lado de baixo.



Construa grafos com a mesma idéia e n grande. □

Algoritmo Guloso por Aresta

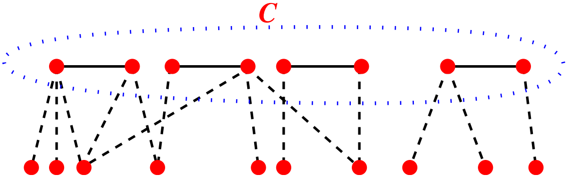
Idéia: Encontrar um emparelhamento maximal.

COBERTURA-VÉRTICES-GULOSO (G)

```
C ← ∅
enquanto E ≠ ∅
    escolha (i, j) ∈ E
    C ← C ∪ {i, j}
    remova todas as arestas adjacentes a i e j
retorne C
```

Teorema: Cobertura-Vértices-Guloso é uma 2-aproximação *(Gavril).*

Prova. Arestas escolhidas formam um emparelhamento maximal.



Qualquer cobertura por vértices deve ter cardinalidade $\geq |C|/2$. □

Cobertura por Conjuntos

Def.: Dada uma coleção $\mathcal{S}$ de subconjuntos de E dizemos que $\mathcal{S}$ *cobre* E, ou é uma *cobertura* de E, se $\cup_{S \in \mathcal{S}} S = E$.

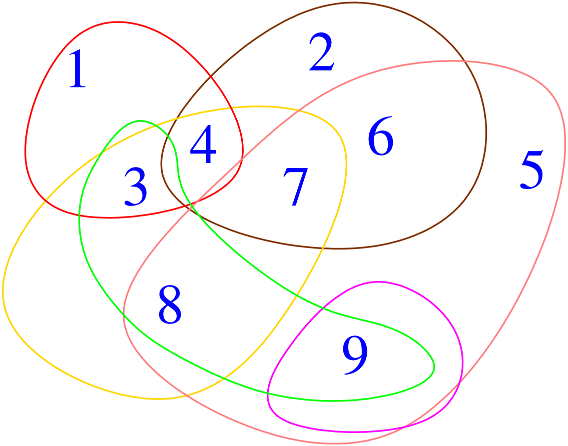
Problema COBERTURA POR CONJUNTOS: Dados conjunto E, subconjuntos $\mathcal{S}$ de E, custos $c(S)$, $S \in \mathcal{S}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $c(\mathcal{S}')$.

Resultados Negativos

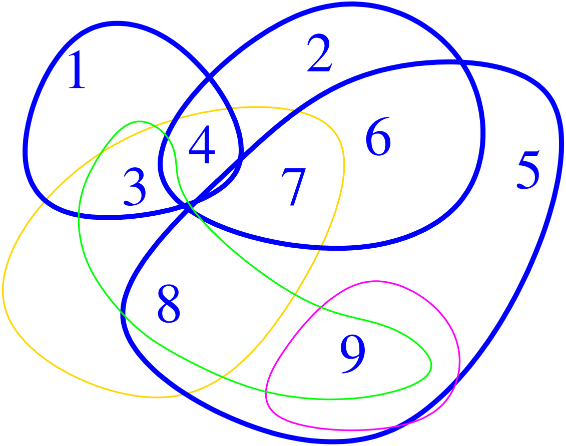
Teorema: COBERTURA POR CONJUNTOS é NP-difícil.

Teorema: *Não há algoritmo de aproximação com fator* $(1 - \epsilon) \ln |E|$, *para qualquer* $\epsilon > 0$, *para o problema da cobertura por conjuntos, a menos que* $\mathrm{P} = \mathrm{NP}$ *(Dinur, Steurer'14).*

Encontre uma cobertura por conjuntos:



Cobertura por conjuntos:



Deteção de Virus

Aplicação *(Williamson/IBM)***:** São determinadas algumas seqüências para cada vírus, cada seqüência com 20 ou mais bytes, em um total de 900 seqüências. O objetivo é encontrar o menor conjunto de seqüências que detecta todos os 500 vírus.

- Conjuntos: Cada seqüência determina um conjunto de vírus que contém a seqüência.
- Conjunto Base: Conjunto de todos os vírus.

Algoritmo Guloso: 190 subseqüências
Limitante para solução ótima: 185 subseqüências

Estratégia Gulosa:
Pegar sempre o conjunto de menor peso relativo.

MINCC-CHVÁTAL $(E, \mathcal{S}, c)$

se $E = \emptyset$
    então devolva $\emptyset$
    senão seja Z em $\mathcal{S}$ tal que $c_Z/|Z \cap E|$ é mínimo
        $E' \leftarrow E \setminus Z$
        $\mathcal{S}' \leftarrow \{ S \in \mathcal{S} : S \cap E' \neq \emptyset \}$
        seja c' a restrição de c a $\mathcal{S}'$
        $\mathcal{T}' \leftarrow$ MINCC-CHVÁTAL $(E', \mathcal{S}', c')$
        devolva $\{Z\} \cup \mathcal{T}'$

Lema: Se $Z \in \mathcal{S}$ é tal que $c(Z)/|Z|$ é mínimo, então

$$\frac{c_Z}{|Z|} \leq \frac{\mathrm{OPT}(E, \mathcal{S}, c)}{|E|}$$

Prova.
Seja $\mathcal{S}^*$ uma cobertura ótima.

$$\begin{aligned}
\frac{c_Z}{|Z|}|E| &\leq \frac{c_Z}{|Z|}\sum_{S\in\mathcal{S}^*}|S| \\
&\leq \sum_{S\in\mathcal{S}^*}\frac{c_S}{|S|}|S| \\
&= \sum_{S\in\mathcal{S}^*} c_S \\
&= c(\mathcal{S}^*) \\
&= \mathrm{OPT}(E, \mathcal{S}, c)\,.
\end{aligned}$$

$\square$

Teorema: O algoritmo MINCC-CHVÁTAL é uma H_n-aproximação polinomial para o MINCC $(E, \mathcal{S}, c)$, onde $n := |E|$ e

$$H_n := 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\,.$$

Prova. Prova por indução em $|E|$.

Se $|E| = 0$, o teorema é claramente válido.

Considere instância $(E, \mathcal{S}, c)$, $|E| > 0$.

$$\begin{aligned}
c(\{Z\} \cup \mathcal{T}') &= \\
&= c_Z + c'(\mathcal{T}') \\
&\leq \tfrac{|Z|}{n}\,\mathrm{OPT}(E, \mathcal{S}, c) + H_{|E'|}\,\mathrm{OPT}(E', \mathcal{S}', c') \\
&\leq \tfrac{|Z|}{n}\,\mathrm{OPT}(E, \mathcal{S}, c) + H_{n-|Z|}\,\mathrm{OPT}(E, \mathcal{S}, c) \\
&= \left(\frac{1}{n} + \frac{1}{n} + \cdots + \frac{1}{n} + H_{n-|Z|}\right)\mathrm{OPT}(E, \mathcal{S}, c) \\
&\leq \left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{n-|Z|+1} + H_{n-|Z|}\right)\mathrm{OPT}(E, \mathcal{S}, c) \\
&= H_n\,\mathrm{OPT}(E, \mathcal{S}, c)\,.
\end{aligned}$$

Teorema: O fator de aproximação do algoritmo MINCC-CHVÁTAL é justo.
Prova.

$$\begin{aligned} E &:= \{1, \ldots, n\}, \\ \mathcal{S} &:= \{E, \{1\}, \ldots, \{n\}\}, \\ c(E) &:= 1+\epsilon \quad \text{e} \\ c(\{i\}) &:= 1/i \quad \text{para cada } i. \end{aligned}$$

Cobertura de custo mínimo é $\{E\}$.
$\mathrm{OPT}(E, \mathcal{S}, c) = 1+\epsilon$.

MINCC-CHVÁTAL produz conjunto $\{\{1\}, \ldots, \{n\}\}$.
MINCC-CHVÁTAL$(E, \mathcal{S}, c) = H_n$. $\square$

Exercício

Mostre que o fator de aproximação do algoritmo que escolhe (e remove) a cada iteração o vértice de maior grau, é uma $O(\log n)$ *aproximação para o problema da cobertura por vértices.*

Esquemas de Aproximação

Def.: *Esquema de aproximação de tempo polinomial (PTAS):*
família de algoritmos $\{A_\epsilon\}$, $\epsilon > 0$, para um problema tal que A_ϵ é uma
$(1+\epsilon)$-aproximação para um problema de minimização,
$(1-\epsilon)$-aproximação para um problema de maximização.

Exemplo de complexidades de tempo: $O(n^{\frac{1}{\epsilon}})$, $O(n^2 5^{\frac{1}{\epsilon}})$ e $O(\frac{1}{\epsilon}n^3)$.

Def.: *Esquema de aproximação de tempo polinomial completo (FPTAS):*
PTAS $\{A_\epsilon\}$ tal que A_ϵ tem tempo polinomial em $\frac{1}{\epsilon}$.

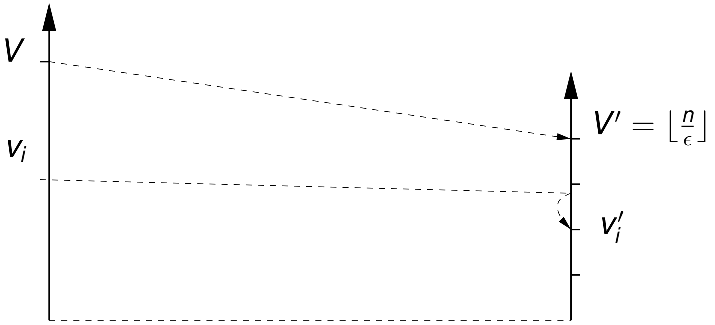
Exemplo de complexidades de tempo: $O(\frac{1}{\epsilon}n^2)$ e $O(\frac{1}{\epsilon^2}n^5)$.

Problema da Mochila

Problema MOCHILA: Dados itens $S = \{1, \ldots, n\}$ com valor v_i e tamanho s_i inteiros, $i = 1, \ldots, n$, e inteiro B, encontrar $S' \subseteq S$ que maximiza $\sum_{i \in S'} v_i$ tal que $\sum_{i \in S'} s_i \leq B$.

Estratégia:
Mudar escala dos valores e aplicar algoritmo exato.

Def.: *Seja $A(i, v) \equiv$ comprimento total mínimo de subconjunto de $\{1, \ldots, i\}$ com valor exatamente v (∞ se não existir tal conjunto).*



Mudança de escala e arredondamento

Perda em valor de cada item: $\frac{V}{n/\epsilon}$.

No pior caso: $n\frac{V}{n/\epsilon} = \epsilon\, V \leq \epsilon\, \mathrm{OPT}$

ALGORITMO MOCHILA-EXATO(B, n, s, v)

$V \leftarrow \max_i v_i$
para $i \leftarrow 0$ até n faça $A(i, 0) \leftarrow 0$
para $w \leftarrow 1$ até nV faça $A(0, w) \leftarrow \infty$
para $i \leftarrow 1$ até n faça
    para $w \leftarrow 1$ até nV faça
        se $v_i \leq w$ então
            $A(i, w) \leftarrow \min(A(i-1, w), s_i + A(i-1, w - v_i))$
        senão
            $A(i, w) \leftarrow A(i-1, w)$

Complexidade de Tempo: $O(n^2 \cdot V)$.

Solução na posição (n, w): $w = \max\{w' : A(n, w') \leq B\}$.

Itens \ Val.	0	1	2	...	$w-v_i$	...	w	...	nV
$\emptyset$	0	∞	∞	...	∞	...	∞	...	∞
$1\ldots 1$	0			...		...		...	
$1\ldots 2$	0			...		...		...	
$\vdots$	$\vdots$			...		...		...	
$1\ldots i-1$	0			...	$A(i-1,w-v_i)$	...	$A(i-1,w)$	...	
$1\ldots i$	0			...			$A(i,w)$	...	
$\vdots$	$\vdots$			...		...		...	
$1\ldots n$	0			...		...		...	

$$A(i,w) = \begin{cases} \min((A(i-1,w)\ ,\ s_i + A(i-1,w-v_i)) & \text{se } v_i \leq w \\ A(i-1,w) & \text{C. c.} \end{cases}$$

ALGORITMO MOCHILA-IK$_\epsilon(B, n, s, v)$

$V \leftarrow \max_i v_i$
$K \leftarrow \frac{\epsilon V}{n}$
$v'_i \leftarrow \left\lfloor \frac{v_i}{K} \right\rfloor \ \forall i$ % I.e., $v'_i = \left\lfloor \frac{n}{\epsilon} \frac{v_i}{V} \right\rfloor$
Execute Mochila-Exato(B, n, s, v')

Teorema: *Mochila-IK$_\epsilon$ é um FPTAS.*

Prova. Seja

S a solução encontrada por Mochila-IK$_\epsilon$,
O^* uma solução ótima de I e
OPT o valor de uma solução ótima.

$$Kv'_i \leq v_i \leq K(v'_i + 1)$$

$$v_i \leq K(v'_i + 1) \implies Kv'_i \geq v_i - K$$

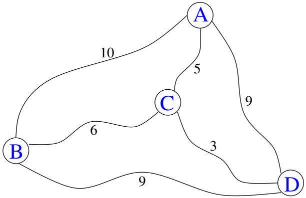
Portanto,

$$\begin{aligned}
\sum_{i\in S} v_i &\geq \sum_{i\in S} K\cdot v'_i \\
&\geq \sum_{i\in O^*} K\cdot v'_i \\
&\geq \sum_{i\in O^*} (v_i - K) = \sum_{i\in O^*} v_i - |O^*|K \\
&\geq \sum_{i\in O^*} v_i - nK = \sum_{i\in O^*} v_i - \epsilon V \\
&\geq \mathrm{OPT} - \epsilon\cdot\mathrm{OPT} = (1-\epsilon)\cdot\mathrm{OPT}.
\end{aligned}$$

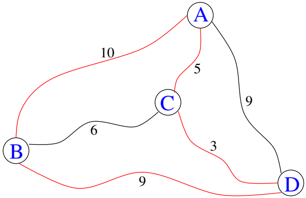
Complexidade de Tempo: $O(n^2V') = O(n^2\left\lfloor\frac{V}{K}\right\rfloor) = O(n^3\frac{1}{\epsilon})$. □

Caixeiro Viajante

Problema TSP: Dados um grafo G e um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e, determinar um circuito hamiltoniano C que minimize $c(C)$.



Grafo G



Circuito Hamiltoniano de G de custo 27

Aplicações:

- Perfuração e solda de circuitos impressos.
- Determinação de rotas de custo mínimo.

Resultados Negativos

Teorema: Dado um grafo G, o problema de decidir se G tem um circuito hamiltoniano é NP-completo.

Teorema: Não existe uma α-aproximação para o TSP, para qualquer valor α, computável polinomialmente, a menos que P=NP

Prova. Suponha existir α-aproximação para o TSP.

Dado $G = (V, E)$ seja $G' = (V', E', c')$ tal que

- $V' = V$
- $E' = V \times V$
- $c'(e) = \begin{cases} 1 & \text{se } e \in E, \\ \alpha|V| & \text{caso contrário} \end{cases}$

Se G tem circuito hamiltoniano, $\mathrm{OPT}(G') = |V|$

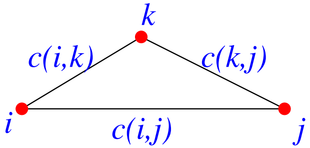
caso contrário, $\mathrm{OPT}(G') > \alpha|V|$.

$\Longrightarrow$ Decidimos existência de circuito hamiltoniano em G. $\square$

Caixeiro Viajante Métrico

Def.: Os custos de um grafo satisfazem desigualdade triangular se

$$c(i,j) \leq c(i,k) + c(k,j), \quad \forall i,j,k \in V$$



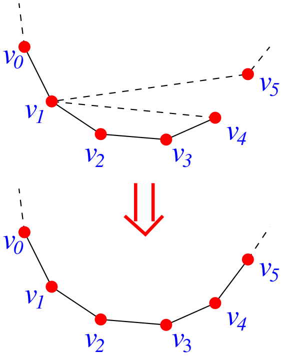
Problema TSP-Métrico: Dados um grafo G com desigualdade triangular e um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e, determinar um circuito hamiltoniano C que minimize $c(C)$.

Estratégia:

Encontrar ciclo gerador e fazer “shortcuts” (atalhos).

ALGORITMO ATALHO(P),
Trilha fechada $P = (v_0, \ldots, v_m)$

$w_0 \leftarrow v_0$
$n \leftarrow 0$
para i de 1 a m faça
    se $v_i \notin \{w_0, \ldots, w_n\}$
        então $n \leftarrow n + 1$
            $w_n \leftarrow v_i$
retorne $\{w_0, \ldots, w_n\}$



Teorema: *Uma árvore geradora de custo mínimo pode ser encontrada em tempo polinomial.*

Teorema: *G tem ciclo euleriano $\Leftrightarrow$ G é conexo e grau(v)=par, $\forall v \in V$.*

Teorema: *Encontrar um ciclo euleriano pode ser feito em tempo polinom*

TSPM-ROSENKRANTZ-STEARNS-LEWIS (G, c)

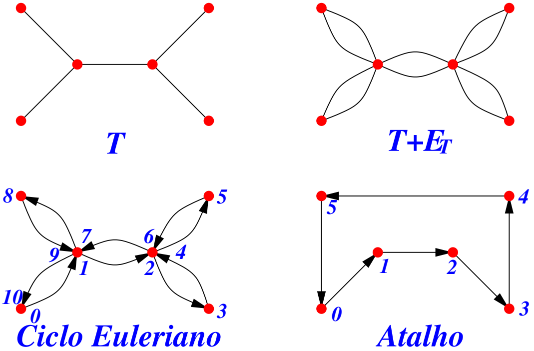
$T \leftarrow$ MINIMUM-SPANNING-TREE (G, c)

$T' \leftarrow T \dotplus E_T$

$P \leftarrow$ CICLO-EULERIANO (T')

$C \leftarrow$ ATALHO (P)

devolva C



Teorema: TSPM-*Rosenkrantz-Stearns-Lewis é uma 2-aproximação.*

Prova.

$$c(T) \leq \mathrm{OPT}(G) \implies c(T + E_T) \leq 2\mathrm{OPT}(G).$$

Atalhos não pioram o custo da solução. □

Algoritmo de Christofides

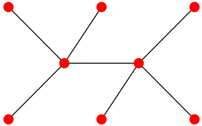
Estratégia:

Aumentar a árvore apenas o suficiente para encontrar ciclo euleriano.

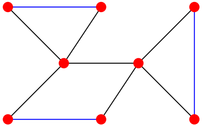
Teorema: *O problema de encontrar um emparelhamento perfeito de peso mínimo pode ser resolvido em tempo polinomial.*

TSPM-CHRISTOFIDES (G, c)

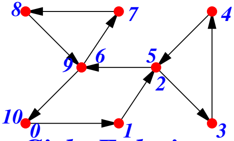
$T \leftarrow$ MINIMUM-SPANNING-TREE (G, c)
seja I o conjunto dos vértices de grau ímpar de T
$M \leftarrow$ EMPARELHAMENTO-PERFEITO-MÍNIMO $(G[I], c)$
$T' \leftarrow T \dotplus M$
$P \leftarrow$ CICLO-EULERIANO (T')
$C \leftarrow$ ATALHO (P)
devolva C



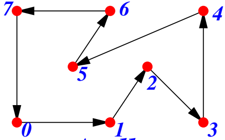
T



T+M



Ciclo Euleriano



Atalho

Teorema: TSPM-*Christofides é uma $\frac{3}{2}$-aproximação.*

Prova. Um circuito em I define dois emparelhamentos alternantes em I.

Assim, $c(M) \leq \frac{1}{2}\mathrm{OPT}(G)$.

Portanto $c(C) \leq c(T) + c(M) \leq \mathrm{OPT}(G) + \frac{1}{2}\mathrm{OPT}(G) = \frac{3}{2}\mathrm{OPT}(G)$. $\square$

Programação Linear e Inteira

Problema PL: Dados matriz $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$, vetores $c = (c_i) \in \mathbb{Q}^n$ e $b = (b_i) \in \mathbb{Q}^m$, encontrar vetor $x = (x_i) \in \mathbb{Q}^n$ (se existir) que

$$\begin{array}{ll}
\text{minimize} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{sujeito a} & \left\{ \begin{array}{lcl}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq & b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq & b_2 \\
\vdots & \vdots & \vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & = & b_m \\
x_i \in \mathbb{Q} & &
\end{array} \right.
\end{array}$$

Teorema: *PL pode ser resolvido em tempo polinomial.*

Problema PLI: Dados matriz $A = (a_{ij}) \in \mathbb{Q}^{m\times n}$, vetores $c = (c_i) \in \mathbb{Q}^n$ e $b = (b_i) \in \mathbb{Q}^m$, encontrar vetor $x = (x_i) \in \mathbb{Z}^n$ (se existir) que

$$
\begin{array}{ll}
\text{minimize} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{sujeito a} & \left\{\begin{array}{lcl}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq & b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq & b_2 \\
\vdots & \vdots & \vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & = & b_m \\
x_i \in \mathbb{Z} & &
\end{array}\right.
\end{array}
$$

Teorema: *PLI é um problema NP-difícil.*

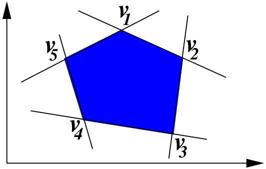
Def.: Chamamos o conjunto de pontos P,

$$P := \left\{ x \in \mathbb{Q}^n : \begin{array}{ccc} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq & b_1 \\ \vdots & \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq & b_m \end{array} \right\}$$

como sendo um poliedro.

Def.: Se $y_i \in P$ e $\alpha_i \in \mathbb{Q}$, $i = 1, \ldots, n$ dizemos que $y = \alpha_1 y_1 + \alpha_2 y_2 + \cdots + \alpha_n y_n$ é uma combinação convexa dos pontos $y_i's$ se $\alpha_i \geq 0$ e $\alpha_1 + \cdots + \alpha_n = 1$

Def.: Os vértices de P são os pontos de P que não podem ser escritos como combinação convexa de outros pontos de P



Teorema: *Uma solução que é vértice do PL pode ser encontrada em tempo polinomial.*

Algoritmos polinomiais para resolver PL:

- Algoritmo dos elipsóides e
- Método dos pontos interiores.

Algoritmos exponenciais para resolver PL:

- Método simplex (tempo médio polinomial).

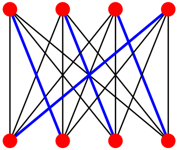
Formulação através de Variáveis Binárias

Considere um evento e e variáveis x_e formuladas como:

$$x_e = \begin{cases} 1 & \text{se evento } e \text{ ocorre,} \\ 0 & \text{se evento } e \text{ não ocorre.} \end{cases}$$

Emparelhamento em Grafos Bipartidos

Def.: Dado um grafo $G = (V, E)$, dizemos que $M \subseteq E$ é um emparelhamento de G se M não tem arestas com extremos em comum.



Problema EMPARELHAMENTO-BIPARTIDO: Dados um grafo bipartido $G = (V, E)$, e custos nas arestas $c : E \to \mathbb{Z}$, encontrar emparelhamento $M \subseteq E$ que maximize $c(M)$.

Aplicação: Encontrar atribuição de candidatos para vagas maximizando aptidão total.

Formulação para Emparelhamento em Grafos Bipartidos

Formulação em Programação Inteira

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{e \in E} c_e x_e \\ \text{sujeito a} & \left\{ \begin{array}{rcll} \displaystyle\sum_{e \in \delta(v)} x_e & \leq & 1 & \forall v \in V \\ x_e & \in & \{0,1\} & \forall e \in E \end{array} \right. \end{array}$$

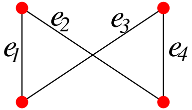
Relaxação Linear

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{e \in E} c_e x_e \\ \text{sujeito a} & \left\{ \begin{array}{rcl} \displaystyle\sum_{e \in \delta(v)} x_e & \leq & 1 \quad \forall e \in E \\ 0 \leq x_e & \leq & 1 \quad \forall e \in E \end{array} \right. \end{array}$$

Teorema: *Os vértices do poliedro relaxado do emparelhamento bipartido são inteiros.*

Exemplo:

Considere o seguinte grafo bipartido com custos unitários:



Programa Linear

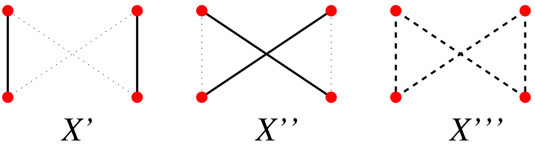
$$
\begin{array}{ll}
\text{maximize} & x_{e_1} + x_{e_2} + x_{e_3} + x_{e_4} \\
\text{sujeito a} & \left\{
\begin{array}{l}
x_{e_1} + x_{e_2} \leq 1, \\
x_{e_3} + x_{e_4} \leq 1, \\
x_{e_1} + x_{e_3} \leq 1, \\
x_{e_2} + x_{e_4} \leq 1, \\
0 \leq x_{e_1} \leq 1, \\
0 \leq x_{e_2} \leq 1, \\
0 \leq x_{e_3} \leq 1, \\
0 \leq x_{e_4} \leq 1
\end{array}
\right.
\end{array}
$$

Temos 4 variáveis binárias no programa linear correspondente:

$$X = (x_{e_1}, x_{e_2}, x_{e_3}, x_{e_4})$$

Os seguintes vetores são soluções ótimas do programa linear:

$$X' = (1, 0, 0, 1) \quad X'' = (0, 1, 1, 0) \quad X''' = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$$



- X' e X'' são vértices do poliedro do emparelhamento
- X''' é solução ótima, mas é combinação convexa de X' e X'' ($X''' = \frac{1}{2}X' + \frac{1}{2}X''$).

Fluxo Máximo e Corte Mínimo

Def.: Dado um grafo orientado $G = (V, E)$, capacidades $c : E \to \mathbb{Q}^+$ e vértices s e t, um fluxo de s para t é um vetor $f \in \mathbb{Q}^E$, tal que

$$\sum_u f(u, v) - \sum_w f(v, w) = 0 \quad \forall v \in V - \{s, t\},$$
$$0 \le f(e) \le c(e) \quad \forall e \in E,$$

O valor do fluxo f é o valor $\sum_v f(s, v)$

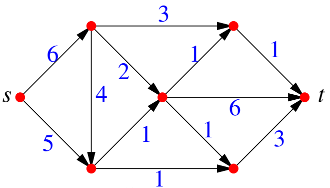
Problema FLUXO-MÁXIMO: Dado grafo orientado $G = (V, E)$, capacidades $c : E \to \mathbb{Q}^+$ e vértices s e t, encontrar um fluxo de valor máximo de s para t.

Teorema: *Se as capacidades c_e são inteiras, então os vértices do poliedro do fluxo são inteiros.*

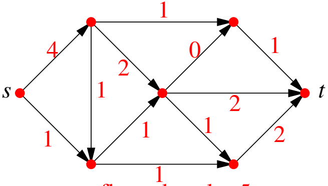
Def.: Dado $S \subseteq V$, $s \in S$ e $t \notin S$, a capacidade do corte $(S, \overline{S})$ é igual a $\sum_{(u,v)\in(S,\overline{S})} c(v, w)$.

Teorema: (Menger) Se $c_e = 1, \forall e$, então número máximo de caminhos aresta disjuntos é igual ao número mínimo de arestas necessárias para desconectar s e t.

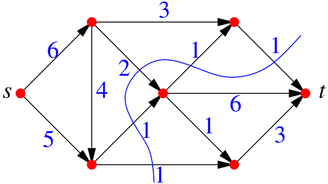
Teorema: *O valor do fluxo máximo de s para t é igual à capacidade do menor corte separando s e t.*



G(V,E,c)



fluxo de valor 5



Corte de capacidade 5

Corolário: *O número máximo de caminhos aresta disjuntos de s a t é igual ao número mínimo de arestas a serem removidas para desconectar s e t.*

Problema da Mochila

Problema MOCHILA: Dados itens $S = \{1, \ldots, n\}$ com valor v_i e tamanho s_i inteiros, $i = 1, \ldots, n$, e inteiro B, encontrar $S' \subseteq S$ que maximiza $\sum_{i \in S'} v_i$ tal que $\sum_{i \in S'} s_i \leq B$.

$$
\begin{array}{ll}
\text{maximize} & \displaystyle\sum_{i \in [n]} v_i x_i \\
\text{sujeito a} & \left\{ \begin{array}{rcll} \displaystyle\sum_{i \in [n]} s_i x_i & \leq & B & \\ x_i & \in & \{0, 1\} & \forall i \in [n] \end{array} \right.
\end{array}
$$

onde $[n] = \{1, \ldots, n\}$.

Coberturas, Empacotamentos e Partições

Seja E um conjunto e $\mathcal{C}$ uma coleção de subconjuntos de E.

Seja $\mathcal{C}_e := \{C \in \mathcal{C} : e \in C\}$ e $\mathcal{S} \subseteq \mathcal{C}$ tal que $x_C = 1 \Leftrightarrow C \in \mathcal{S}$.

- **$\mathcal{S}$ é uma cobertura se**

$$\sum_{C \in \mathcal{C}_e} x_C \geq 1 \quad \forall e \in E,$$

- **$\mathcal{S}$ é um empacotamento se**

$$\sum_{C \in \mathcal{C}_e} x_C \leq 1 \quad \forall e \in E,$$

- **$\mathcal{S}$ é uma partição se**

$$\sum_{C \in \mathcal{C}_e} x_C = 1 \quad \forall e \in E.$$

Problema da Cobertura por Conjuntos

Problema COBERTURA POR CONJUNTOS: Dados conjunto E, subconjuntos $\mathcal{S}$ de E, custos $c(S)$, $S \in \mathcal{S}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $c(\mathcal{S}')$.

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{S \in \mathcal{S}} c_S x_S \\
\text{sujeito a} & \left\{ \begin{array}{lcll} \displaystyle\sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \forall e \in E \\ x_S & \in & \{0,1\} & \forall S \in \mathcal{S} \end{array} \right.
\end{array}
$$

Problema da Localização de Facilidades

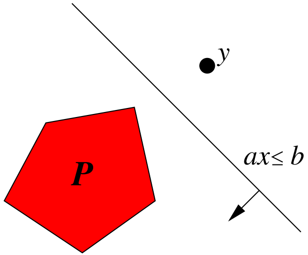
Problema LOCALIZAÇÃO DE FACILIDADES: Dados potenciais facilidades $F = \{1, \ldots, n\}$, clientes $C = \{1, \ldots, m\}$, custos f_i para "abrir" a facilidade i e custos $c_{ij} \in \mathbb{Z}$ para um cliente j ser atendido pela facilidade i. Encontrar facilidades $A \subseteq F$ minimizando custo para abrir as facilidades em A e atender todos os clientes

Aplicação: Instalar postos de distribuição de mercadorias.

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{i \in F} f_i y_i + \sum_{ij \in E} c_{ij} x_{ij} \\
\text{sujeito a} & \left\{ \begin{array}{rcll}
\displaystyle\sum_{ij \in E} x_{ij} & = & 1 & \forall j \in C, \\
x_{ij} & \leq & y_i & \forall ij \in E, \\
y_i & \in & \{0,1\} & \forall i \in F \\
x_{ij} & \in & \{0,1\} & \forall i \in F \text{ e } j \in C.
\end{array} \right.
\end{array}
$$

Otimização $\times$ Separação

Def.: **Problema da Separação:** Seja $P \subseteq \mathbb{R}^n$ um conjunto convexo e $y \in \mathbb{R}^n$, determinar se $y \in P$, caso contrário, encontrar desigualdade $ax \leq b$ tal que $P \subseteq \{x : ax \leq b\}$ e $ay > b$.



Teorema: *O problema de otimização de um programa linear pode ser resolvido em tempo polinomial se e somente se o problema da separação para o poliedro do programa linear pode ser resolvido em tempo polinomial.*

ALGORITMO PLANOS DE CORTE(P, c)

P poliedro (não necessariamente explícito),

c função objetivo

$Q \leftarrow$ {Poliedro inicial}

$y \leftarrow$OPT-LP(Q, c)

enquanto y pode ser separado de Q faça

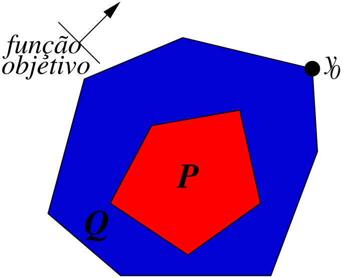
  seja $ax \leq b$ desigualdade de P que separa y

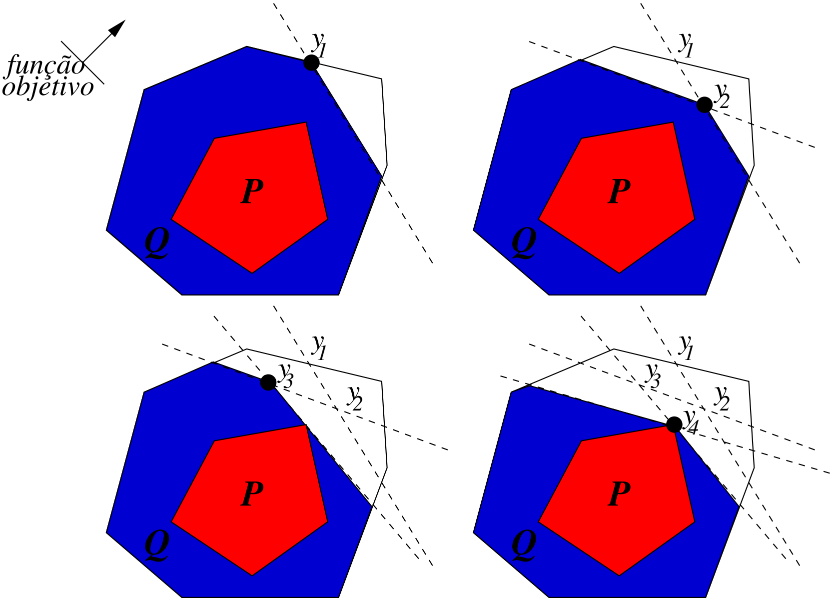
  $Q \leftarrow Q \cap \{x : ax \leq b\}$

  $y \leftarrow$OPT-LP(Q, c)

se $Q = \emptyset$ retorne $\emptyset$

senão retorne y.




função objetivo
y1
P
Q
y1
y2
P
Q
y1
y3
y2
P
Q
y1
y3
y2
y4
P
Q

Problema do Emparelhamento

Def.: Dado um grafo $G = (V, E)$, dizemos que $M \subseteq E$ é um emparelhamento de G se M não tem arestas com extremos em comum.

$$
\begin{array}{lllll}
\text{maximize} & \displaystyle\sum_{e \in E} c_e x_e & & & \\
\text{sujeito a} & \left\{ \begin{array}{rcll} \displaystyle\sum_{e \in \delta(v)} x_e & \leq & 1 & \forall v \in V, \\ \displaystyle\sum_{e \in E[S]} x_e & \leq & \dfrac{|S|-1}{2} & \forall S \subseteq V, \ \ |S| \text{ ímpar}, \\ x_e & \in & \{0,1\} & \forall e \in E. \end{array} \right. & & & ,
\end{array}
$$

onde $E[S] := \{e \in E : e \text{ tem ambos extremos em } S\}$.

Teorema: *Os vértices do poliedro relaxado do emparelhamento são inteiros (Edmonds'65).*

Teorema: *O problema da separação do poliedro relaxado do emparelhamento pode ser resolvido em tempo polinomial (Padberg & Rao'82).*

Problema do Caixeiro Viajante

Problema TSP: Dados um grafo $G = (V, E)$ e um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e, determinar um circuito hamiltoniano C que minimize $c(C)$.

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{e \in E} c_e x_e \\
\text{sujeito a} & \left\{
\begin{array}{rcll}
\displaystyle\sum_{e \in \delta(v)} x_e & = & 2 & \forall v \in V \\
\displaystyle\sum_{e \in \delta(S)} x_e & \geq & 2 & \forall S \subset V, \quad S \neq \emptyset \\
 & & & \text{(restrições de subcircuito)} \\
x_e & \in & \{0, 1\} & \forall e \in E
\end{array}
\right.
\end{array}
$$

onde $E[S] := \{e \in E : e \text{ tem ambos extremos em } S\}$.

Teorema: *O problema da separação relativo às restrições de subcircuito pode ser resolvido em tempo polinomial.*

Problema da Floresta de Steiner

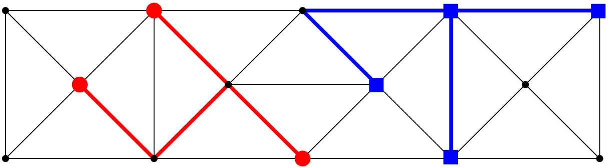
Seja G um grafo e $\mathcal{R}$ uma coleção de subconjuntos de V_G.

Def.: *Uma $\mathcal{R}$-floresta de G é qualquer floresta geradora F de G tal que para todo $R \in \mathcal{R}$, os elementos de R estão contidos em algum componente de F.*

Problema Floresta de Steiner: Dados um grafo G, um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e e uma coleção $\mathcal{R}$ de subconjuntos de V_G, encontrar uma $\mathcal{R}$-floresta F que minimize $c(F)$.

Aplicações:

- Roteamento em circuitos VLSI.
- Projeto de redes de conectividade.



Formulação:

$$
\begin{array}{lrcll}
\text{minimize} & \displaystyle\sum_{e\in E} c_e x_e & & & \\
\text{sujeito a} & \left\{ \begin{array}{rcll} \displaystyle\sum_{e\in\delta(S)} x_e & \geq & 1 & \forall S \subset V : \exists R \in \mathcal{R}, \emptyset \neq S \cap R \neq R \\ & & & \text{(desigualdades de corte)} \\ x_e & \in & \{0,1\} & \forall e \in E \end{array} \right. & & &
\end{array}
$$

Teorema: *O problema da separação relativo às desigualdades de corte pode ser resolvido em tempo polinomial.*

Método Primal: Arredondamento

Arredondamento da Solução Primal

Problema COBERTURA POR VÉRTICES: Dados grafo $G = (V, E)$ e custos $c : V \to \mathbb{Q}_{\geq}$, encontrar $S \subseteq V$ tal que $\{u, v\} \cap S \neq \emptyset$ $\forall \{u, v\} \in E$ e $c(S)$ é mínimo.

$$
(P) \qquad \begin{array}{lrcll}
\text{minimize} & cx & & & \\
\text{sujeito a} & x_i + x_j & \geq & 1 & \text{para cada } ij \text{ em } E\,, \\
 & x_i & \geq & 0 & \text{para cada } i \text{ em } V\,.
\end{array}
$$

MINCV-NT $(G = (V, E), c)$

1 seja $\hat{x}$ uma solução ótima racional de (P)

2 $S \leftarrow \{v \in V : \hat{x}_v \geq 1/2\}$

3 devolva S

Teorema: MinCV-NT é uma 2-aproximação (Nemhauser,Trotter'75; Hochbaum'83).

Prova. MinCV-NT produz uma cobertura de vértices:
$(S := \{v \in V : \hat{x}_v \geq 1/2\}$ e $x_i + x_j \geq 1 \quad \forall ij \in E) \Longrightarrow S$ é cobertura.

$$\begin{aligned} c(S) &= \sum_{i \in S} c_i \\ &\leq \sum_{i \in S} c_i 2\,\hat{x}_i \\ &= 2 \sum_{i \in S} c_i \hat{x}_i \\ &\leq 2 \sum_{i \in V} c_i \hat{x}_i \\ &= 2\, c\, \hat{x} \\ &\leq 2\, \mathrm{OPT} \end{aligned}$$

□

Generalização para Cobertura por Conjuntos

Problema COBERTURA POR CONJUNTOS: Dados conjunto E, subconjuntos $\mathcal{S}$ de E, custos $c(S) \in \mathbb{Q}_{\geq}$, $S \in \mathcal{S}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $\sum_{S \in \mathcal{S}'} c(S)$.

$$
\begin{array}{llllll}
 & \text{minimize} & cx & & & \\
(P) & \text{sob as restrições} & \displaystyle\sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \text{para cada } e \text{ em } E\,, \\
 & & x_S & \geq & 0 & \text{para cada } S \text{ em } \mathcal{S}\,.
\end{array}
$$

onde $\mathcal{S}_e = \{S \in \mathcal{S} : \ e \in S\}$

MINCC-HOCHBAUM $(E, \mathcal{S}, c)$

seja $\hat{x}$ uma solução ótima racional de (P)
para cada e em E faça $f_e \leftarrow |\{S \in \mathcal{S} : e \in S\}|$
$\beta \leftarrow \max_{e \in E} f_e$
$\mathcal{T} \leftarrow \{S \in \mathcal{S} : \hat{x}_S \geq 1/\beta\}$
devolva $\mathcal{T}$

Teorema: *O algoritmo* MinCC-Hochbaum *é uma* β*-aproximação polinomial para o* MinCC $(E, \mathcal{S}, c)$*, onde* β *é o número máximo de vezes que um elemento de* E *aparece em conjuntos de* $\mathcal{S}$*.*

Prova. O custo da cobertura $\mathcal{T}$ produzida pelo algoritmo é

$$\begin{aligned} c(\mathcal{T}) &= \sum_{S \in \mathcal{T}} c_S \\ &\leq \sum_{S \in \mathcal{T}} c_S \beta\, \hat{x}_S \\ &= \beta \sum_{S \in \mathcal{T}} c_S \hat{x}_S \\ &\leq \beta\, c\, \hat{x} \\ &\leq \beta\, \mathrm{OPT}(E, \mathcal{S}, c)\,, \end{aligned}$$

onde a primeira desigualdade vale pois $\beta\, \hat{x}_S \geq 1$ para todo S em $\mathcal{T}$. $\square$

Fator de Aproximação Assintótico

Adequado quando o limiar de aproximação é atingido apenas para instâncias "pequenas" do problema

Dado um algoritmo A para um problema de minimização e instância I,

- $\mathrm{OPT}(I)$ é o tamanho da solução ótima
- $A(I)$ é o tamanho da solução gerada pelo algoritmo A
- A têm um fator de aproximação assintótico α se

$$R_A^\infty := \inf \left\{ r \geq 1 : \ \exists N > 0, \frac{A(I)}{\mathrm{OPT}(I)} \leq r, \mathrm{OPT}(I) > N \right\} \leq \alpha$$

Definição análoga pode ser feita para problemas de maximização.

Notação: Dados conjunto $B \subseteq \{1, \ldots, n\}$ e uma função qualquer $s : \mathbb{N} \to \mathbb{R}$, denotamos por $s(B)$ o valor $\sum_{i \in B} s_i$.

Problema EMPACOTAMENTO (n, c) Dados inteiro positivo n e, para cada i em $\{1, \ldots, n\}$, um número racional s_i no intervalo $[0, 1]$, encontrar uma partição $\mathcal{B}$ de $\{1, \ldots, n\}$ tal que $s(B) \leq 1$ para todo B em $\mathcal{B}$ e $|\mathcal{B}|$ seja mínimo.

Teorema: *Não existe α-aproximação para o problema* EMPACOTAMENTO *com $\alpha < 3/2$, a menos que* P = NP.

Prova. Exercício. Dica: O problema de decisão da Partição é NP-completo. □

Fato: *Dado instância L para o problema do* EMPACOTAMENTO,

$$s(L) \leq \mathrm{OPT}(L)$$

Teorema: *(Fernandez de la Vega e Lucker'81) Existe algoritmo polinomial A_ϵ, $\epsilon > 0$, para o problema do* EMPACOTAMENTO *tal que*

$$A_\epsilon(L) \leq (1+\epsilon)\mathrm{OPT}(L) + 1,$$

para toda instância L.

Vamos provar o seguinte resultado:

Teorema: *(Fernandez de la Vega e Lucker'81) Existe algoritmo linear A_ϵ, $\epsilon > 0$, para o problema do* EMPACOTAMENTO *tal que*

$$A_\epsilon(L) \leq (1+\epsilon)\mathrm{OPT}(L) + \beta_\epsilon,$$

para toda instância L, onde β_ϵ só depende de ϵ.

NF (n, c)

$k \leftarrow 1$
$B_k \leftarrow \emptyset$
para i de 1 a n faça
    se $s_i \leq 1 - s(B_k)$
        então $B_k \leftarrow B_k \cup \{i\}$
        senão $k \leftarrow k + 1$
            $B_k \leftarrow \{i\}$
devolva $\{B_1, \ldots, B_k\}$

Teorema: NF *é uma* 2*-aproximação polinomial para* EMPACOTAMENTO*.*
Prova. Exercício. □

Lema: *Se* $\mathcal{B} = \{B_1, \ldots, B_k\}$ *é um empacotamento de L tal que* $s(B_i) \geq 1 - \frac{\epsilon}{2}$, *para* $i = 1, \ldots, k - 1$ *e* $\epsilon \in (0, 1)$ *então*

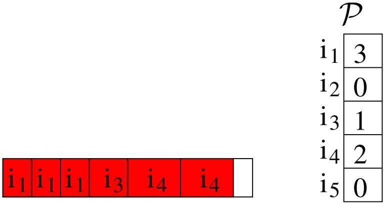
$$k \leq (1 + \epsilon) \cdot \mathrm{OPT}(L) + 1$$

Prova. Exercício. □

Restringindo itens no tamanho e nos tipos

L tem no máximo *k* tipos diferentes de items cada um com tamanho pelo menos ϵ onde *k* e ϵ são constantes.

Representação de um padrão em um vetor



k tipos diferentes em *L* cada item com tamanho pelo menos ϵ

$$\Downarrow$$

número de padrões diferentes é constante.

Algoritmo RST$_{k,\epsilon}(L)$

1. Produza todos os padrões $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_w$ possíveis de se empacotar items de L em um bin unitário.
2. Seja x^* uma solução ótima do seguinte LP com no máximo k variáveis não nulas:

$$\begin{array}{ll} & \text{minimize } \sum x_i \\ \text{s.a.} & \mathcal{P}_1 \cdot x_1 + \mathcal{P}_2 \cdot x_2 + \cdots + \mathcal{P}_w \cdot x_w = M, \\ & x_i \in \mathbb{Q}^* \end{array}$$

   onde M é o vetor de multiplicidade dos itens de L.
3. Obtenha um empacotamento $\mathcal{P}$ da lista L arredondando para cima as variáveis x^*
4. Retorne $\mathcal{P}$

Observação: Na prática, passo 2 pode ser feito pelo método de Geração de Colunas de maneira muito rápida.

Lema: $\mathrm{RST}_{k,\epsilon}(L) \leq \mathrm{OPT}(L) + k$
Prova. Como há no máximo k tipos diferentes em L, o LP

$$\text{minimize} \sum_{i=1}^{w} x_i$$

$$\text{s.a.} \quad \mathcal{P}_1 \cdot x_1 + \mathcal{P}_2 \cdot x_2 + \cdots + \mathcal{P}_w \cdot x_w = M,$$

$$x_i \in \mathbb{Q}^*$$

tem k linhas e portanto existe uma solução que é um ponto extremal com no máximo k variáveis não nulas. □

Restringindo itens no tamanho

L tem items com tamanho pelo menos ϵ.

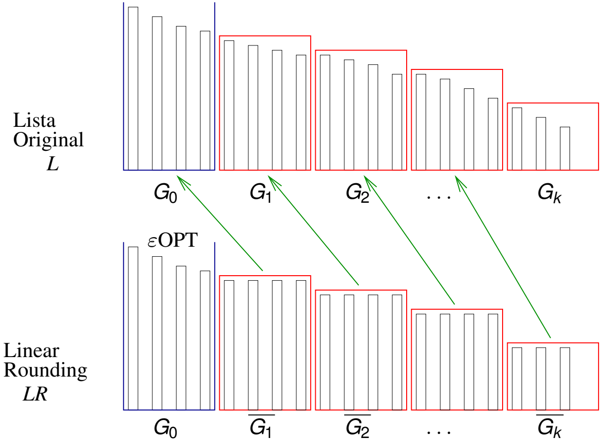
Idéia: Linear Rounding
Agrupar os itens por tamanho em k sublistas e arredondar os itens de cada sublista por cima.

Def.: *Temos $A \preceq B$ ($B \succeq A$) se $\exists$ função injetora $f : A \to B$ tal que $s(i) \leq s(f(i))$.*

Fato: *Se $A \preceq B$ então $\mathrm{OPT}(A) \leq \mathrm{OPT}(B)$*

Def.: *Seja $\overline{A}$ a lista com os itens da lista A arredondados para o maior item da lista A.*

Linear Rounding: Suponha todos itens em L são grandes ($s_i > \epsilon$)
Particione a lista em grupos de mesmo tamanho e
arredonde os itens de cada grupo, exceto os do maior grupo



Idéia: $\mathrm{OPT}(LR) \leq (1+\varepsilon)\mathrm{OPT}(L)$

Seja L ordenado de maneira não crescente e $L = (G_0 \| G_1 \| \ldots \| G_k)$ onde k e G_i são tais que $|G_i| = q,\ i = 0, \ldots, k-1$ e $|G_k| \leq q$.

Fato: $\overline{G_i} \preceq G_{i-1}$.

Fato: $\mathrm{OPT}(\overline{G_1} \| \overline{G_2} \| \ldots \| \overline{G_k}) \leq \mathrm{OPT}(L)$

Prova. Note que

$$
\begin{aligned}
(\overline{G_1} \| \overline{G_2} \| \ldots \| \overline{G_k}) &\preceq (G_0 \| G_1 \| \ldots \| G_{k-1}) \\
&\preceq (G_0 \| G_1 \| \ldots \| G_{k-1} \| G_k) \\
&= L
\end{aligned}
$$

□

Algoritmo $\text{RS}_\epsilon(L)$

1. Ordene L em ordem não crescente de tamanho.
2. Dado $q = \lceil n \cdot \epsilon^2 \rceil$, particione L em grupos $L = (G_0 \| G_1 \| \ldots \| G_k)$ onde k e G_i são tais que

$$|G_i| = q,\ \ i = 0, \ldots, k-1 \qquad \text{e} \qquad |G_k| \leq q$$

3. Seja $J \leftarrow (\overline{G_1} \| \overline{G_2} \| \ldots \| \overline{G_k})$.
4. $\mathcal{P}_{1..k} \leftarrow \text{RST}_{k,\epsilon}(J)$
5. $\mathcal{P}_0 \leftarrow \text{NF}(G_0)$
6. Retorne $\mathcal{P}_0 \| \mathcal{P}_{1..k}$

Lema: $RS_\epsilon(L) \leq (1+\epsilon)\cdot \mathrm{OPT}(L) + \beta_\epsilon$

Prova. Como $J \preceq L$

$$\begin{aligned}\mathrm{RST}_{k,\epsilon}(J) &\leq \mathrm{OPT}(J) + k_\epsilon \\ &\leq \mathrm{OPT}(L) + k_\epsilon\end{aligned}$$

$$\begin{aligned}\mathrm{NF}(G_0) &\leq q = \lceil n\cdot\epsilon^2\rceil \\ &\leq n\cdot\epsilon^2 + 1 \\ &\leq \epsilon\cdot s(L) + 1 \\ &\leq \epsilon\cdot \mathrm{OPT}(L) + 1\end{aligned}$$

Portanto

$$\begin{aligned}\mathrm{RS}(L) &= \mathrm{RST}_{k,\epsilon}(J) + \mathrm{NF}(G_0) \\ &\leq (\mathrm{OPT}(L) + k_\epsilon) + (\epsilon\cdot\mathrm{OPT}(L) + 1) \\ &= (1+\epsilon)\cdot\mathrm{OPT}(L) + \beta_\epsilon\end{aligned}$$

$\square$

Algoritmo $\mathcal{A}_\epsilon(L)$

1. Divida a lista L em L' e L'':
$$L' := \{i \in L : \ s(i) > \epsilon/2\} \quad \text{e} \quad L'' := L \setminus L'.$$
2. $\mathcal{P}'_1 \leftarrow \mathrm{RS}_{\epsilon/2}(L')$
3. Empacote os itens de L'' usando a estratégia NF testando nos bins gerados em $\mathcal{P}'_1$ (se necessário use bins novos).
4. Retorne o empacotamento gerado no passo anterior.

Teorema: $\mathcal{A}_\epsilon(L) \leq (1+\epsilon)\mathrm{OPT}(L) + \beta_\epsilon$

Prova. Seja $B_1, \ldots, B_k$ os bins gerados por $\mathcal{A}_\epsilon$. Temos dois casos:

Caso 1: NF gerou novos bins ao empacotar L''.
Então

$$s(B_i) \geq (1 - \epsilon/2), \qquad \text{para todo } i = 1, \ldots, k-1.$$

Portanto $\mathcal{A}_\epsilon(L) \leq (1+\epsilon)\mathrm{OPT} + 1$.

Caso 2: NF não gerou novos bins ao empacotar L''.
Logo

$$\mathcal{A}_\epsilon(L) = RS(L') \leq (1+\epsilon)\mathrm{OPT} + \beta_\epsilon$$

$\square$

Dualidade em Programação Linear

Considere o seguinte PL:

$$
\begin{array}{lc}
\text{minimize} & c_1x_1 + c_2x_2 + c_3x_3 \\
\text{sujeito a} & \left\{ \begin{array}{rcl}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 & \geq & b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 & = & b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & \leq & b_3 \\
x_1 \geq 0, \qquad\qquad x_3 \leq 0 & &
\end{array} \right.
\end{array}
$$

Vamos delimitar o valor ótimo do LP:

Multiplique as restrições por $y_1 \geq 0$, y_2 e $y_3 \leq 0$:

$$
\begin{array}{rcl}
y_1(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & \geq & y_1b_1 \\
y_2(a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & = & y_2b_2 \\
y_3(a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & \geq & y_3b_3
\end{array}
$$

Somando estas inequações obtemos:

$$
\begin{array}{lcl}
y_1(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) & \geq & y_1 b_1 \\
y_2(a_{21}x_1 + a_{22}x_2 + a_{23}x_3) & = & y_2 b_2 \\
y_3(a_{31}x_1 + a_{32}x_2 + a_{33}x_3) & \geq & y_3 b_3 \\
\hline
(y_1 a_{11} + y_2 a_{21} + y_3 a_{31})x_1 + & & \\
(y_1 a_{12} + y_2 a_{22} + y_3 a_{32})x_2 + & & \\
(y_1 a_{13} + y_2 a_{23} + y_3 a_{33})x_3 & \geq & (y_1 b_1 + y_2 b_2 + y_3 b_3) = yb
\end{array}
$$

Comparando com a função objetivo $cx = c_1x_1 + c_2x_2 + c_3x_3$, se

$$c_1x_1 \geq (y_1 a_{11} + y_2 a_{21} + y_3 a_{31})x_1$$

$$c_2x_2 = (y_1 a_{12} + y_2 a_{22} + y_3 a_{32})x_2$$

$$c_3x_3 \geq (y_1 a_{13} + y_2 a_{23} + y_3 a_{33})x_3$$

Nestas condições, temos $cx \geq yb$ e portanto yb é limitante de cx

Como $x_1 \geq 0$ e $x_3 \leq 0$, podemos simplificar as condições para ter $cx \geq yb$ como:

$$c_1 \geq y_1 a_{11} + y_2 a_{21} + y_3 a_{31}$$

$$c_2 = y_1 a_{12} + y_2 a_{22} + y_3 a_{32}$$

$$c_3 \leq y_1 a_{13} + y_2 a_{23} + y_3 a_{33}$$

Naturalmente, queremos dentre todos os valores de y, o que melhor delimita $cx \geq yb$, i.e., com yb máximo. Assim, obtemos o seguinte problema dual:

$$\begin{array}{ll}
\text{maximize} & y_1 b_1 + y_2 b_2 + y_3 b_3 \\
\text{sujeito a} & \left\{ \begin{array}{l}
y_1 a_{11} + y_2 a_{21} + y_3 a_{31} \leq c_1 \\
y_1 a_{12} + y_2 a_{22} + y_3 a_{32} = c_2 \\
y_1 a_{13} + y_2 a_{23} + y_3 a_{33} \geq c_3 \\
y_1 \geq 0, \qquad\qquad y_3 \leq 0
\end{array} \right.
\end{array}$$

Convenção para nomear estes sistemas:

Problema Primal

$$
\begin{array}{ll}
\text{minimize} & c_1 x_1 + c_2 x_2 + c_3 x_3 \\
\text{sujeito a} & \left\{ \begin{array}{lcl}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 & \geq & b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 & = & b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & \leq & b_3 \\
x_1 \geq 0, \qquad\qquad x_3 \leq 0 & &
\end{array} \right.
\end{array}
$$

Problema Dual

$$
\begin{array}{ll}
\text{maximize} & y_1 b_1 + y_2 b_2 + y_3 b_3 \\
\text{sujeito a} & \left\{ \begin{array}{l}
y_1 a_{11} + y_2 a_{21} + y_3 a_{31} \leq c_1 \\
y_1 a_{12} + y_2 a_{22} + y_3 a_{32} = c_2 \\
y_1 a_{13} + y_2 a_{23} + y_3 a_{33} \geq c_3 \\
y_1 \geq 0, \qquad\qquad y_3 \leq 0
\end{array} \right.
\end{array}
$$

Exercício

Faça exatamente a mesma análise feita anteriormente para se obter um dual (como limitante de um programa linear), mas em vez de partir de um problema de minimização, comece com um problema de maximização, e obtenha seu dual como um problema de minimização.

Exercício

Faça o programa dual da formulacao relaxada dos seguintes problemas:

- *Problema da Cobertura de Vertices*
- *Problema da Cobertura por Conjuntos*
- *Problema de Localizacao de Facilidades*
- *Problema de Steiner*

Problemas Primal e Dual

Problema Primal

$$
(P) \qquad
\begin{array}{lrcll}
\text{minimize} & c\,x & & & \\
\text{sujeito a} & (Ax)_i & \geq & b_i & \text{para cada } i \text{ em } M_1\,, \\
 & (Ax)_i & = & b_i & \text{para cada } i \text{ em } M_2\,, \\
 & (Ax)_i & \leq & b_i & \text{para cada } i \text{ em } M_3\,, \\
 & x_j & \geq & 0 & \text{para cada } j \text{ em } N_1\,, \\
 & x_j & \leq & 0 & \text{para cada } j \text{ em } N_3\,.
\end{array}
$$

Problema Dual

$$
(D) \qquad
\begin{array}{lrcll}
\text{maximize} & y\,b & & & \\
\text{sujeito a} & (yA)_j & \leq & c_j & \text{para cada } j \text{ em } N_1\,, \\
 & (yA)_j & = & c_j & \text{para cada } j \text{ em } N_2\,, \\
 & (yA)_j & \geq & c_j & \text{para cada } j \text{ em } N_3\,, \\
 & y_i & \geq & 0 & \text{para cada } i \text{ em } M_1\,, \\
 & y_i & \leq & 0 & \text{para cada } i \text{ em } M_3\,.
\end{array}
$$

Lema: *Seja (P) um programa primal de minimização e (D) seu problema dual de maximização, então $cx \geq yb$ para todo $x \in P$ e $y \in D$.*

Folgas Complementares

Dois vetores x e y, indexados por M e N respectivamente, têm folgas complementares se,

$x_j = 0$ ou $(yA)_j = c_j$ $\forall j \in N_1 \cup N_3$ (folgas complementares primais)
e
$y_i = 0$ ou $(Ax)_i = b_i$ $\forall i \in M_1 \cup M_3$ (folgas complementares duais).

Lema: *(das folgas complementares) Se (P) é um programa linear e (D) seu programa dual, x e y soluções viáveis de (P) e (D) então*

$$(cx = yb) \quad \Leftrightarrow \quad (x \text{ e } y \text{ satisfazem folgas complementares})$$

Teorema: *(da dualidade) Se (P) é um programa linear de minimização e (D) seu programa dual (de maximização), então vale exatamente uma das possibilidades:*

1. *(P) e (D) são viáveis e* OPT-LP(P)=OPT-LP(D).
2. *(P) é viável e (D) é inviável e* OPT-LP(P) $= -\infty$.
3. *(P) é inviável e (D) é viável e* OPT-LP(D) $= \infty$.
4. *(P) e (D) são inviáveis.*

Lema: *(de Farkas) Exatamente um dos programas restritos têm solução:*

$$(RP) \quad \begin{array}{rcll} \exists \;\; x & \in & \mathbb{Q}^N & \\ (Ax)_i & \geq & b_i & \forall i \in M, \\ x_i & \geq & 0 & \forall j \in N. \end{array} \qquad (RD) \quad \begin{array}{rcll} \exists \;\; y & \in & \mathbb{Q}^M & \\ yb & > & 0 & \\ (yA)_j & \leq & 0 & \forall j \in N, \\ y_i & \geq & 0 & \forall i \in M. \end{array}$$

Prova.

Considere o programa linear (P) e seu dual (D):

$$(P) \quad \begin{array}{rcll} \min & 0\,x & & \\ (Ax)_i & \geq & b_i & \forall i \in M, \\ x_i & \geq & 0 & \forall j \in N. \end{array} \qquad (D) \quad \begin{array}{rcll} \max & yb & & \\ (yA)_j & \leq & 0 & \forall j \in N, \\ y_i & \geq & 0 & \forall i \in M. \end{array}$$

Note que (P) é viável se e só se (RP) é víavel e o programa (D) é sempre viável pois $y = 0$ satisfaz as restrições.

Como (D) é viável, apenas as alternativas 1. ou 3. do Lema da Dualidade podem ocorrer.

Caso 1: (P) é viável e OPT-LP(P)=OPT-LP(D).

(P) viável $\Rightarrow$ (RP) é viável.

$\Big(\forall y \in (D),\ yb \leq \text{OPT-LP}(D) = \text{OPT-LP}(P) = 0\Big) \Rightarrow (RD)$ é inviável.

Caso 3: OPT-LP$(D) = \infty$ e (P) é inviável.

(P) inviável $\Rightarrow$ (RP) é inviável.

$\Big(\text{OPT-LP}(D) = \infty\Big) \Rightarrow \Big(\exists y \in (D) :\ yb > 0\Big) \Rightarrow (RD)$ é viável. $\square$

Método Dual: Arredondamento

Arredondamento pela Solução Dual

Se $\hat{x}$ e $\hat{y}$ são soluções ótimas de (P) e (D), então

$$\hat{x}_j > 0 \quad \Longrightarrow \quad (\hat{y}A)_j = c_j \qquad \text{(folgas complementares primais)}$$

Estratégia:

Resolver o programa dual e selecionar j onde $(\hat{y}A)_j = c_j$.

Problema COBERTURA POR VÉRTICES: Dados grafo $G = (V, E)$ e custos $c : V \to \mathbb{Q}_{\geq}$, encontrar $S \subseteq V$ tal que $\{u, v\} \cap S \neq \emptyset$ $\forall \{u, v\} \in E$ e $c(S)$ é mínimo.

Formulações Primal e Dual

$$(P) \quad \begin{array}{rcll} \min & cx & & \\ x_i + x_j & \geq & 1 & \forall ij \in E\,, \\ x_i & \geq & 0 & \forall i \in V\,. \end{array} \qquad (D) \quad \begin{array}{rcll} \max & y(E_G) & & \\ \displaystyle\sum_{e \in \delta(v)} y_e & \leq & c_v & \forall v \in V\,, \\ y_e & \geq & 0 & \forall e \in E\,. \end{array}$$

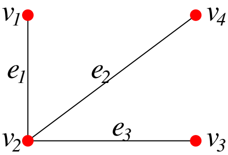
MINCV-HOCHBAUM (G, c)

1 seja $\hat{y}$ uma solução ótima de (D)

2 $C \leftarrow \{v \in V_G : \sum_{e \in \delta(v)} \hat{y}_e = c_v\}$

3 devolva C

Exemplo:



$$
(P)\quad \begin{array}{rl}
\min & x_{v_1}+x_{v_2}+x_{v_3}+x_{v_4} \\
& x_{v_1}+x_{v_2} \geq 1 \\
& x_{v_2} + x_{v_4} \geq 1 \\
& x_{v_2}+x_{v_3} \geq 1 \\
& x_v \geq 0
\end{array}
\qquad
(D)\quad \begin{array}{rll}
\max & y_{e_1}+y_{e_2}+y_{e_3} & \\
& y_{e_1} \leq 1 & (D_{v_1}) \\
& y_{e_1}+y_{e_2}+y_{e_3} \leq 1 & (D_{v_2}) \\
& y_{e_3} \leq 1 & (D_{v_3}) \\
& y_{e_2} \leq 1 & (D_{v_4}) \\
& y_e \geq 0 &
\end{array}
$$

Valor da solução ótima de (D) $\Rightarrow$ $\hat{y}(E) = 1$

$\hat{y} = (\hat{y}_{e_1} = 1, \quad \hat{y}_{e_2} = 0, \quad \hat{y}_{e_3} = 0)$

Desigualdades de (D) justas: (D_{v_1}) e (D_{v_2})

Solução aproximada: $\{v_1, v_2\}$

Teorema: *O algoritmo* MINCV-HOCHBAUM *produz uma cobertura por vértices.*

Prova. Se $uv \in E$ então ($\sum_{e \in \delta(u)} \hat{y}_e = c_u$ ou $\sum_{e \in \delta(v)} \hat{y}_e = c_v$) pois $\hat{y}$ é ótimo. □

Teorema: *O algoritmo* MINCV-HOCHBAUM *é uma* 2*-aproximação polinomial para o* MINCV*.*

Prova. Ao final do algoritmo, C é tal que

$$
\begin{aligned}
c(C) &= \sum_{v \in C} c_v = \sum_{v \in C} \sum_{e \in \delta(v)} \hat{y}_e \\
&\leq 2 \sum_{e \in E_G} \hat{y}_e \qquad (*) \\
&\leq 2\,\mathrm{OPT}(G, c)\,.
\end{aligned}
$$

(*) vale pois, $\forall e \in E$ temos que $\hat{y}_e$ aparece no máximo duas vezes na soma. □

Método Dual: Seleção de Dual Maximal

Def.: *Se $\tilde{y}$ é uma solução viável para um programa dual, $\tilde{y}$ é maximal se não existe solução viável y, com $y_i \geq \tilde{y}_i$ e $\sum_{i=1}^{n} y_i > \sum_{i=1}^{n} \tilde{y}_i$.*

Estratégia: Seleção através de Dual Maximal

1. Encontre uma solução dual viável, maximal
2. Selecione os objetos satisfeitos com igualdade no dual

Problema COBERTURA POR CONJUNTOS: Dados conjunto E, subconjuntos $\mathcal{S}$ de E, custos $c : \mathcal{S} \to \mathbb{Q}_{\geq}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $\sum_{S \in \mathcal{S}'} c(S)$.

$$
\begin{array}{llcll}
\min & \displaystyle\sum_{S \in \mathcal{S}} c_S x_S & & & \\
(P) & \displaystyle\sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \forall e \in E \\
 & x_S & \geq & 0 & \forall S \in \mathcal{S},
\end{array}
\qquad
\begin{array}{llcll}
\max & \displaystyle\sum_{e \in E} y_e & & & \\
(D) & \displaystyle\sum_{e \in S} y_e & \leq & c_S & \forall S \in \mathcal{S} \\
 & y_e & \geq & 0 & \forall e \in E
\end{array}
$$

onde $\mathcal{S}_e := \{S \in \mathcal{S} : e \in S\}$

MINCC-HOCHBAUM $(\mathcal{S}, E, c)$

1 seja $\tilde{y}$ uma solução viável maximal de (D)

2 $\mathcal{S}' \leftarrow \{S \in \mathcal{S} : \sum_{e \in S} \tilde{y}_e = c_S\}$

3 devolva $\mathcal{S}'$

Seja $\tilde{y}$ um vetor maximal de (D) e $\mathcal{S}'$ obtida de $\tilde{y}$ pela algoritmo MINCC-HOCHBAUM.

Lema: *$\mathcal{S}'$ é uma cobertura por conjuntos.*

Prova. Se $\mathcal{S}'$ não é cobertura por conjuntos, então existe $e \in E$ não coberto. Logo podemos "aumentar" $\tilde{y}_e$ até que alguma desigualdade dual fique justa. Contrariando a maximalidade de $\tilde{y}$. $\square$

Seja $\beta := \max\{|\mathcal{S}_e| : e \in E\}$, onde $\mathcal{S}_e = \{S \in \mathcal{S} : e \in S\}$.

Teorema: *MinCC-Hochbaum é uma β-aproximação para o problema da cobertura por conjuntos.*

Prova. Seja $\mathcal{S}'$ o conjunto devolvido pelo algoritmo e $\tilde{y}$ o vetor usado no passo 1 para gerar $\mathcal{S}'$.

$$
\begin{aligned}
c(\mathcal{S}') &= \sum_{S \in \mathcal{S}'} c(S) = \sum_{S \in \mathcal{S}'} \sum_{e \in S} \tilde{y}_e \\
&\leq \sum_{e \in E} |\mathcal{S}_e| \tilde{y}_e \\
&\leq \sum_{e \in E} \beta \tilde{y}_e = \beta \sum_{e \in E} \tilde{y}_e \\
&\leq \beta \sum_{e \in E} \hat{y}_e = \beta \, \text{OPT-LP} \\
&\leq \beta \, \text{OPT}
\end{aligned}
$$

□

Método Primal-Dual

- Baseado em Folgas Complementares e Problemas de Viabilidade
- Muitas vezes produz algoritmos combinatórios

Método Primal-Dual Clássico

$$(P)\quad \begin{array}{rrcll} \min & cx & & & \\ & (Ax)_i & \geq & b_i & \forall i \in M\,, \\ & x_i & \geq & 0 & \forall j \in N\,. \end{array} \qquad (D)\quad \begin{array}{rrcll} \max & yb & & & \\ & (yA)_j & \leq & c_j & \forall j \in N\,, \\ & y_i & \geq & 0 & \forall i \in M\,. \end{array}$$

Folgas Complementares

Se $\hat{x}$ e $\hat{y}$ são soluções ótimas de (P) e (D), então

$\hat{x}_j = 0$ ou $(\hat{y}A)_j = c_j$ (folgas complementares primais)

$\hat{y}_i = 0$ ou $(A\hat{x})_i = b_i$ (folgas complementares duais)

Estratégia:

Dado vetor y viável de (D),

- obter x viável de (P) satisfazendo folgas complementares com y
  ou
- obter y' tq. $y'' \leftarrow y + y'$ é viável em (D) e $y''b > yb$;
  repita processo com y''

Folgas Complementares

Se $\hat{x}$ e $\hat{y}$ são soluções ótimas de (P) e (D), então

$$\hat{x}_j = 0 \quad \text{ou} \quad (\hat{y}A)_j = c_j \qquad \text{(folgas complementares primais)}$$
$$\hat{y}_i = 0 \quad \text{ou} \quad (Ax)_i = b_i \qquad \text{(folgas complementares duais)}$$

Dado vetor y viável de (D),

$$I(y) := \{i \in M : \ y_i = 0\} \quad \text{e} \quad J(y) := \{j \in N : \ (yA)_j = c_j\} .$$

$$\text{(Viabil.)} \quad \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in M , \\ x_i & \geq & 0 & \forall j \in N . \end{array} \; + \; \text{(F.C.)} \quad \begin{array}{rcll} (Ax)_i & = & b_i & \forall i \in M \setminus I(y) , \\ x_j & = & 0 & \forall j \in N \setminus J(y) . \end{array}$$

Problema Restrito Primal

$$(RP) \quad \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in I(y) , \\ (Ax)_i & = & b_i & \forall i \in M \setminus I(y) , \\ x_j & \geq & 0 & \forall j \in J(y) , \\ x_j & = & 0 & \forall j \in N \setminus J(y) . \end{array}$$

Pelo Lema de Farkas, (RP) é viável ou (RD) é viável, não ambos (exercício).

$$(RP)\quad \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in I(y)\,, \\ (Ax)_i & = & b_i & \forall i \in M \setminus I(y)\,, \\ x_j & \geq & 0 & \forall j \in J(y)\,, \\ x_j & = & 0 & \forall j \in N \setminus J(y)\,. \end{array} \oplus \quad (RD)\quad \begin{array}{rcll} y'b & > & 0 & \\ (y'A)_j & \leq & 0 & \forall j \in J(y)\,, \\ y'_i & \geq & 0 & \forall i \in I(y)\,. \end{array}$$

$$I(y) := \{i \in M : \; y_i = 0\} \quad \text{e} \quad J(y) := \{j \in N : \; (yA)_j = c_j\}\,.$$

$$(D) \quad \begin{array}{rrcll} \max & yb & & & \\ & (yA)_j & \leq & c_j & \forall j \in N\,, \\ & y_i & \geq & 0 & \forall i \in M\,. \end{array}$$

$$(RD) \quad \begin{array}{rcll} y'b & > & 0 & \\ (y'A)_j & \leq & 0 & \forall j \in J(y)\,, \\ y'_i & \geq & 0 & \forall i \in I(y)\,. \end{array}$$

Lema: *Se y é viável para (D) e y' é viável para (RD), então, existe $\theta > 0$ tal que $y'' \leftarrow y + \theta\, y'$ é também é viável para (D).*

Prova. Exercício. □

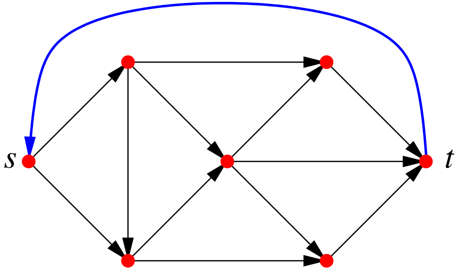
Método PRIMAL-DUAL (A, b, c)

seja y um vetor viável de (D)
enquanto $\mathrm{RP}(A, b, y)$ não tem solução faça
    seja y' uma solução do $\mathrm{RD}(A, b, y)$
    se $y + \theta y'$ em (D) para todo θ positivo
        então devolva y'
        senão seja θ máximo tal que $y + \theta y'$ é viável em (D)
            $y \leftarrow y + \theta y'$
seja x uma solução do $\mathrm{RP}(A, b, y)$
devolva x e y

Problema do Fluxo Máximo

Problema FLUXO-MÁXIMO: Dado um grafo orientado $G = (N, A)$, capacidades $c : A \to \mathbb{N}$ e vértices s e t, encontrar um fluxo de valor máximo de s para t.

Simplificação: adicionar aresta (t, s) sem restrição de capacidade.



Objetivo: Encontrar fluxo que respeita conservação de fluxo em todos os vértices e maximiza fluxo na aresta (t, s).

Por conveniência, vamos chamar a formulação do fluxo de Dual.

$$
(D) \quad \begin{array}{lrcll}
\max & y_{ts} & & & \\
& \displaystyle\sum_{e \in \delta^+(i)} y_e - \sum_{e \in \delta^-(i)} y_e & = & 0 & \forall i \in N, \\
& y_{ij} & \leq & c_{ij} & \forall ij \in A, \\
& y_{ij} & \geq & 0 & \forall ij \in A.
\end{array}
$$

Primal: encontrar $x = x' \| x''$, x' indexado por N e x'' indexado por A que

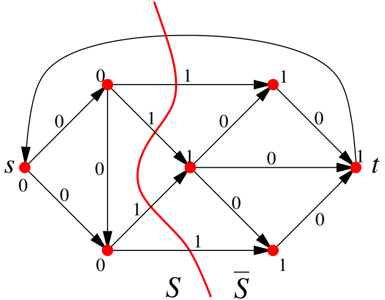
$$
(P) \quad \begin{array}{lrcll}
\min & \displaystyle\sum_{ij \in A} c_{ij} x''_{ij} & & & \\
& x'_t - x'_s & \geq & 1 & , \\
& x'_i - x'_j + x''_{ij} & \geq & 0 & \forall ij \in A, \\
& x''_{ij} & \geq & 0 & \forall ij \in A.
\end{array}
$$

Sobre o programa *(P)*

$$(P)\quad \begin{array}{lrcll} \min & \displaystyle\sum_{ij\in A} c_{ij}\, x''_{ij} & & & \\ & x'_t - x'_s & \geq & 1 & , \\ & x'_i - x'_j + x''_{ij} & \geq & 0 & \forall ij \in A, \\ & x''_{ij} & \geq & 0 & \forall ij \in A. \end{array}$$

Dado um corte $S \subset N$ que separa s de t (i.e. $s \in S$ e $t \notin S$), defina

$$x'_i = \begin{cases} 0 & \forall i \in S \\ 1 & \forall i \in N \setminus S \end{cases} \qquad x''_{ij} = \begin{cases} 1 & \forall ij \in \delta^+(S) \\ 0 & \forall ij \in A \setminus \delta^+(S) \end{cases}$$



Temos $x = x' \| x''$ viável para (P) definindo um corte com capacidade dada pela função objetivo.

Problemas Restritos

Dado y, fluxo viável seja

$$I := \{ij \in A : \; y_{ij} = 0\} \qquad \text{e} \qquad J := \{ij \in A : \; y_{ij} = c_{ij}\} \,.$$

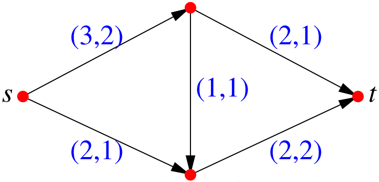
Restrito Primal: encontrar $x = x' \| x''$ viável em (P) satisfazendo F.C.

$$(RP) \quad \begin{array}{rcll} x'_t - x'_s & \geq & 1 & , \\ x'_i - x'_j + x''_{ij} & = & 0 & \forall ij \in A \setminus I, \\ x'_i - x'_j + x''_{ij} & \geq & 0 & \forall ij \in I, \\ x''_{ij} & \geq & 0 & \forall ij \in J, \\ x''_{ij} & = & 0 & \forall ij \in A \setminus J. \end{array}$$

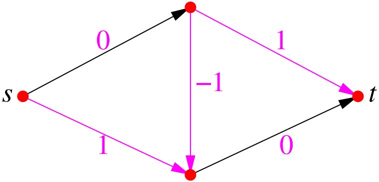
Restrito Dual: Se (RP) é inviável, pelo Lema de Farkas, (RD) é viável

$$(RD) \quad \begin{array}{rcll} y_{ts} & > & 0 & , \\ \displaystyle\sum_{e \in \delta^+(i)} y_e - \sum_{e \in \delta^-(i)} y_e & = & 0 & \forall i \in N, \\ y_{ij} & \leq & 0 & \forall ij \in J, \\ y_{ij} & \geq & 0 & \forall ij \in I. \end{array}$$

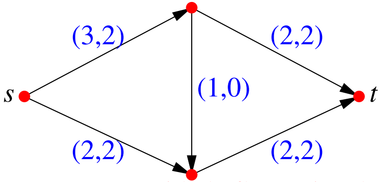
(RD) viável $\Rightarrow \exists$ caminho aumentador $P = (y')$ de s para t, representando um fluxo adicional.



(c,y)=(capacidade,fluxo viável)
Valor do fluxo inicial = 3



Caminho aumentador
Fluxo no caminho = 1



(c,y)=(capacidade,fluxo viável)
Valor do fluxo resultante = 4

(RD) inviável $\Rightarrow$ (RP) é viável

Seja $S := \{v \in N : \text{ existe caminho aumentador de } s \text{ para } v\}$ e

$$x'_i = \begin{cases} 0 & \forall i \in S \\ 1 & \forall i \in N \setminus S \end{cases} \qquad x''_{ij} = \begin{cases} 1 & \forall ij \in \delta^+(S) \\ 0 & \forall ij \in A \setminus \delta^+(S) \end{cases}$$

Temos que $x = x' \| x''$ é viável em (RP)

Método de Aproximação Primal-Dual

Generalização do Método Primal-Dual Clássico por F.C. aproximadas

$$(P)\quad \begin{array}{lrcll} \min & cx & & & \\ & (Ax)_i & \geq & b_i & \forall i \in M\,, \\ & x_j & \geq & 0 & \forall j \in N\,. \end{array} \qquad (D)\quad \begin{array}{lrcll} \max & yb & & & \\ & (yA)_j & \leq & c_j & \forall j \in N\,, \\ & y_i & \geq & 0 & \forall i \in M\,. \end{array}$$

Folgas Aproximadas

Dado $0 < \alpha \leq 1 \leq \beta$, x e y soluções viáveis de (P) e (D)

x e y tem folgas α-Aproximadas no Primal se

$$x_j = 0 \quad \text{ou} \quad \alpha\, c_j \leq (yA)_j \quad [\leq c_j\,]$$

x e y tem folgas β-Aproximadas no Dual se

$$y_i = 0 \quad \text{ou} \quad \beta\, b_i \geq (Ax)_i \quad [\geq b_i\,]$$

Lema: *Se x e y são não-negativos e satisfazem as condições de folgas α-aproximadas no primal e β-aproximadas no dual, então* $\alpha\, c x \leq \beta\, y b$.

Prova.

$$\begin{aligned}
\alpha\, c x &\leq \sum_{j \in N} \alpha\, c_j\, x_j \\
&\leq \sum_{j \in N} (y A)_j\, x_j \\
&= \sum_{i \in M} y_i\, (A x)_i \\
&\leq \sum_{i \in M} y_i\, \beta\, b_i \\
&= \beta\, y\, b
\end{aligned}$$

□

Lema: *(das Folgas Aproximadas) Se x e y são soluções viáveis de (P) e (D) satisfazendo folgas α e β aproximadas então*

x é uma $\frac{\beta}{\alpha}$-aproximação em (P) e
y é uma $\frac{\alpha}{\beta}$-aproximação em (D).

Estratégia:

Dado vetor y viável de (D), valores α e β, onde $0 < \alpha \leq 1 \leq \beta$

- encontrar x viável de (P) satisfazendo folgas α e β aproximadas com y

  ou
- encontrar y' tq. $y'' \leftarrow y + y'$ é viável em (D) e $y''b > yb$;
  repita processo com y''

Folgas Aproximadas

$$x_j = 0 \quad \text{ou} \quad (yA)_j \geq \alpha\, c_j \qquad \text{(folgas aproximadas primais)}$$

$$y_i = 0 \quad \text{ou} \quad (Ax)_i \leq \beta\, b_i \qquad \text{(folgas aproximadas duais)}$$

Dado vetor y viável de (D),

$$I(y) := \{i \in M : \; y_i = 0\} \qquad \text{e} \qquad J(y, \alpha) := \{j \in N : \; (yA)_j \geq \alpha\, c_j\}\,.$$

$$\text{(Viabil.)} \; \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in M \\ x_j & \geq & 0 & \forall j \in N \end{array} + \text{(F.A.)} \; \begin{array}{rcll} (Ax)_i & \leq & \beta\, b_i & \forall i \in M \setminus I(y) \\ x_j & = & 0 & \forall j \in N \setminus J(y, \alpha) \end{array}$$

Problema Restrito Aproximado Primal

$$(RAP) \quad \begin{array}{rcll} (Ax)_i & \geq & b_i & \forall i \in M\,, \\ (Ax)_i & \leq & \beta\, b_i & \forall i \in M \setminus I(y)\,, \\ x_j & \geq & 0 & \forall j \in J(y, \alpha)\,, \\ x_j & = & 0 & \forall j \in N \setminus J(y, \alpha)\,. \end{array}$$

(RAP) é inviável $\Rightarrow$ (RP) é inviável $\Rightarrow$ (RD) é viável $\Rightarrow$ (RAD) é viável.

$$
(RAP)\quad
\begin{array}{rcll}
(Ax)_i & \geq & b_i & \forall i \in M\,, \\
(Ax)_i & \leq & \beta\, b_i & \forall i \in M \setminus I(y)\,, \\
x_j & \geq & 0 & \forall j \in J(y,\alpha)\,, \\
x_j & = & 0 & \forall j \in N \setminus J(y,\alpha)\,.
\end{array}
\qquad
(RAD)\quad
\begin{array}{rcll}
y'b & > & 0 & \\
(y'A)_j & \leq & 0 & \forall j \in J(y,\alpha)\,, \\
y'_i & \geq & 0 & \forall i \in I(y)\,.
\end{array}
$$

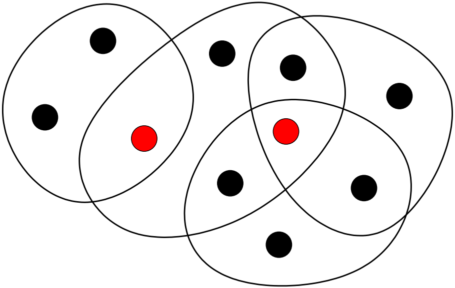
Método PRIMAL-DUAL (A, b, c)

seja y um vetor viável de (D)
enquanto $\mathrm{RAP}(A, b, y, \alpha, \beta)$ não tem solução faça
    seja y' uma solução do $\mathrm{RAD}(A, b, y, \alpha)$
    se $y + \theta y'$ é viável em (D) para todo θ positivo
        então devolva y'
        senão seja θ máximo tal que $y + \theta y'$ é viável em (D)
            $y \leftarrow y + \theta y'$
seja x uma solução do $\mathrm{RAP}(A, b, y, \alpha, \beta)$
devolva x e y

Problema da Transversal Mínima

Def.: *Dado conjunto E e coleção $\mathcal{S}$ finita de subconjuntos de E, um conjunto $T \subseteq E$ é uma transversal de $\mathcal{S}$ se $T \cap S \neq \emptyset, \forall S \in \mathcal{S}$.*

Problema MINTC: Dados conjunto E, coleção finita $\mathcal{S}$ de subconjuntos de E e custos $c : E \rightarrow \mathbb{Q}_{\geq}$, encontrar transversal T de $\mathcal{S}$ tal que $\sum_{e \in T} c_e$ é mínimo.



Formulações Primal e Dual:

$$(P)\quad \begin{array}{lrcll} \min & \displaystyle\sum_{e\in E} c_e x_e & & & \\ & \displaystyle\sum_{e\in S} x_e & \geq & 1 & \forall S \in \mathcal{S}, \\ & x_e & \geq & 0 & \forall e \in E. \end{array}$$

$$(D)\quad \begin{array}{lrcll} \max & \displaystyle\sum_{S\in \mathcal{S}} y_S & & & \\ & \displaystyle\sum_{S\in \mathcal{S}_e} y_S & \leq & c_e & \forall e \in E, \\ & y_S & \geq & 0 & \forall S \in \mathcal{S}, \end{array}$$

onde $\mathcal{S}_e = \{S \in \mathcal{S} : e \in S\}$.

Estratégia: Obter x e y viáveis, com x binário, satisfazendo:

Folgas aproximadas primais:

$$x_e = 1 \Rightarrow \alpha c_e \leq \sum_{S\in\mathcal{S}_e} y_S \leq c_e \quad \text{para todo } e \in E$$

Folgas aproximadas primais:

$$y_S > 0 \Rightarrow \beta \geq \sum_{e\in S} x_e \geq 1 \quad \text{para todo } S \in \mathcal{S}$$

Problemas Restritos Aproximados

Considere $\alpha := 1$ e $\beta := \max\{|S| : S \in \mathcal{S}\}$. Seja y, viável dual e

$$I(y) := \{S \in \mathcal{S} : y_S = 0\} \quad \text{e} \quad J(y) := \{e \in E : \sum_{S \in \mathcal{S}_e} y_S \geq c_e\}.$$

Restrito Aproximado Primal: encontrar x viável em (P) satisfazendo F.A.

$$(RAP) \begin{array}{rcll} \sum_{e \in E} x_e & \geq & 1 & \forall S \in \mathcal{S}, \\ \sum_{e \in S} x_e & \leq & \beta & \forall S \in \mathcal{S} \setminus I(y), \\ x_e & \geq & 0 & \forall e \in J(y), \\ x_e & = & 0 & \forall e \in E \setminus J(y). \end{array} \qquad (RAD) \begin{array}{rcll} \sum_{S \in \mathcal{S}} y'_S & > & 0 & , \\ \sum_{S \in \mathcal{S}} y'_S & \leq & 0 & \forall e \in J(y), \\ y'_S & \geq & 0 & \forall S \in I(y). \end{array}$$

Restrito Aproximado Dual: Se (RAP) é inviável então (RP) é inviável e pelo Lema de Farkas, (RAD) é viável

Algoritmo de Bar-Yehuda e Even

MinTC-BE $(E, \mathcal{S}, c)$

$J \leftarrow \{e \in E : c_e = 0\}$

$y_S \leftarrow 0, \ \forall S \in \mathcal{S}$

enquanto existe $R \in \mathcal{S}$ tal que $J \cap R = \emptyset$ faça

  aumente y_R até o máximo, mantendo válidas as restrições

$$\sum_{S \in \mathcal{S}_e} y_S \leq c_e, \ \forall e \in R$$

  seja f um elemento de R tal que $\sum_{S \in \mathcal{S}_f} y_S = c_f$

  $J \leftarrow J \cup \{f\}$

devolva J

Lema: *Seja J o conjunto devolvido por* MinTC-BE, *x o vetor característico de J e y o vetor dual construído pelo algoritmo. Então,*

1. *J é uma transvesal,*
2. *x satisfaz folgas aproximadas primais com $\alpha = 1$*
$$x_e = 0 \quad ou \quad 1 \cdot c_e \leq \sum_{S \in \mathcal{S}_e} y_S \leq c_e$$
3. *y satisfaz folgas aproximadas duais com $\beta = \max\{|S| : S \in \mathcal{S}\}$:*
$$y_e = 0 \quad ou \quad \beta \cdot 1 \geq \sum_{e \in S} x_e \geq 1$$

Prova.

Para verificar (1), note que se $R \in \mathcal{S}$ e $J \cap R = \emptyset$ então há uma folga para aumentar y_R.

Para verificar (2), note que um elemento $e \in J$ foi escolhido por satisfazer $\displaystyle\sum_{S \in \mathcal{S}_e} y_S = c_e$

Para verificar (3), note que $\displaystyle\sum_{e \in S} x_e \leq |S| \leq \beta$. $\square$

Teorema: *O algoritmo* MINTC-BE *é uma* β*-aproximação polinomial para o* MINTC $(E, \mathcal{S}, c)$, *onde* $\beta := \max_{S \in \mathcal{S}} |S|$.

Prova. Segue do Lema das Folgas Aproximadas. □

Exercício

Usando a abordagem vista para o problema da Transversal Mínima, apresente uma 2-aproximação para o problema da Cobertura por Vértices.

Problema de Localização de Facilidades

Problema LOCALIZAÇÃO *(facility location problem)*: Dados potenciais facilidades $F = \{1, \ldots, n\}$, clientes $C = \{1, \ldots, m\}$, custos f_i para "abrir" a facilidade i e custos $c_{ij} \in \mathbb{Z}$ para um cliente j ser atendido pela facilidade i. Encontrar facilidades $A \subseteq F$ minimizando o custo para abrir as facilidades em A e atender todos os clientes

Aplicações: Instalação de postos de distribuição de mercadorias, construção de redes de telecomunicação.

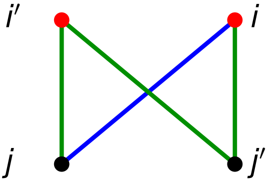
Teorema: *O problema* MINCC *é um caso particular do problema* LOCALIZAÇÃO*.*

Corolário: *(Raz & Safra'97) O problema de* LOCALIZAÇÃO *não pode ser aproximado em $\epsilon \log |E|$, para alguma constante $\epsilon > 0$, a menos que* P $=$ NP*.*

Problema de Localização de Facilidades Métrico

Custos satisfazem desigualdade triangular:

$$c_{ij} \leq c_{ij'} + c_{i'j'} + c_{i'j} \quad i, i' \in F \text{ e } j, j' \in C$$



Vamos ver uma 3-aproximação de Jain e Vazirani'01 usando técnica primal-dual:

Programa inteiro:

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{i \in F} f_i y_i + \sum_{ij \in E} c_{ij} x_{ij} \\
\text{sujeito a} & \left\{ \begin{array}{rcll}
\displaystyle\sum_{ij \in E} x_{ij} & \geq & 1 & \forall j \in C, \\
y_i - x_{ij} & \geq & 0 & \forall ij \in E, \\
x_{ij} & \in & \{0,1\} & \forall i \in F \text{ e } j \in C, \\
y_i & \in & \{0,1\} & \forall i \in F.
\end{array} \right.
\end{array}
$$

Programa Relaxado Primal:

$$
(P) \qquad \text{tal que} \qquad
\begin{array}{rcll}
\min & \displaystyle\sum_{i \in F} f_i y_i + \sum_{ij \in E} c_{ij} x_{ij} & & \\
& \displaystyle\sum_{ij \in E} x_{ij} & \geq & 1 \quad \forall j \in C\,, \\
& y_i - x_{ij} & \geq & 0 \quad \forall i \in F,\ j \in C\,, \\
& x_{ij} & \geq & 0 \quad \forall i \in F,\ j \in C\,, \\
& y_i & \geq & 0 \quad \forall i \in F\,.
\end{array}
$$

Programa Dual:

$$
(D) \qquad \text{tal que} \qquad
\begin{array}{rcll}
\max & \displaystyle\sum_{j \in C} \alpha_j & & \\
& \alpha_j - \beta_{ij} & \leq & c_{ij} \quad \forall i \in F,\ j \in C\,, \\
& \displaystyle\sum_{j \in C} \beta_{ij} & \leq & f_i \quad \forall i \in F\,, \\
& \alpha_j & \geq & 0 \quad \forall j \in C\,, \\
& \beta_{ij} & \geq & 0 \quad \forall i \in F,\ j \in C\,.
\end{array}
$$

Método Primal-Dual com folgas aproximadas primais:

$$\begin{array}{rcl} x_{ij} = 1 & \Rightarrow & \frac{1}{3}c_{ij} \leq \alpha_j - \beta_{ij} \leq c_{ij} \\ y_i = 1 & \Rightarrow & \frac{1}{3}f_i \leq \sum_{j \in C} \beta_{ij} \leq f_i \end{array}$$

e folgas complementares duais:

$$\begin{array}{rcl} \alpha_j > 0 & \Rightarrow & \sum_{i \in F} x_{ij} = 1 \\ \beta_{ij} > 0 & \Rightarrow & y_i = x_{ij} \end{array}$$

Isto nos dá uma 3-aproximação para o problema da LOCALIZAÇÃO. Entretanto, vamos usar um resultado mais forte para usá-lo no problema da k-MEDIANA.

Estratégia do algoritmo:

- Inicialmente comece as variáveis duais nulas.
- Aumente uniformemente as variáveis α's até que uma variável α_j tenha seu crescimento limitado por uma aresta ij ($\alpha_j - \beta_{ij} \leq c_{ij}$). Neste caso a variável β_{ij} também começa a aumentar.
- Uma facilidade é justa quando tem a soma dos valores β incidentes a ela igual ao seu custo. Neste ponto as variáveis duais associadas param de crescer.
- O conjunto das facilidades abertas é um subconjunto das facilidades justas.

FACILITY-JV (F,C,c)

Inicializações

$A \leftarrow C$ % Conjunto de clientes ativos

$S \leftarrow \emptyset$ % Conjunto de arestas especiais

$R \leftarrow \emptyset$ % Conjunto de arestas especiais inativas

$F_t \leftarrow \emptyset$ % Facilidades temporariamente abertas

para cada j em C faça $\alpha_j \leftarrow 0$

para cada $i \in F$ e $j \in C$ faça $\beta_{ij} \leftarrow 0$

Fase 1

enquanto $A \neq \emptyset$ faça
    aumente uniformemente α_j e β_{ij} para $j \in A$ e $ij \in S \setminus R$ até que
        (a) $\alpha_j = c_{ij}$ para algum $ij \in (F \setminus F_t \times A) \setminus S$ ou
        (b) $\alpha_j = c_{ij}$ para algum $ij \in (F_t \times A) \setminus S$ ou
        (c) $\sum_{j \in C} \beta_{ij} = f_i$ para algum $i \in F \setminus F_t$.
    se (a) foi satisfeita para algum ij então
        $S \leftarrow S \cup \{ij\}$
    senão se (b) foi satisfeita para algum ij então
        $\tau(j) \leftarrow i$ % testemunha de j
        $A \leftarrow A \setminus \{j\}$
    senão se (c) foi satisfeita para algum i então
        para cada j tal que $ij \in S$ faça $\tau(j) \leftarrow i$
        $A \leftarrow A \setminus \{j : ij \in S\}$
        $R \leftarrow R \cup \{ij : ij \in S\}$
        $F_t \leftarrow F_t \cup \{i\}$

Fase 2

$H \leftarrow (V_H, E_H)$ onde

$\quad V_H := F_t$

$\quad E_H := \{i'i'' : i', i'' \in V_H$ e $\exists j \in C,$ com $i'j \in S$ e $i''j \in S\}$

$I \leftarrow$ (Conjunto independente maximal de H)

para todo $j \in C$ faça

$\quad$ se $\tau(j) \in I$ então

$\quad\quad \phi(j) \leftarrow \tau(j)$ ($(\phi(j), j)$ é conexão direta)

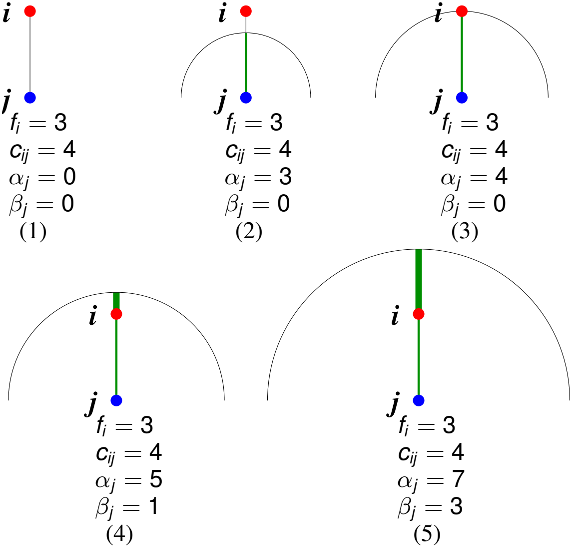
$\quad$ senão

$\quad\quad$ seja $i \in I$ tal que $(i, \tau(j)) \in H$

$\quad\quad \phi(j) \leftarrow i$ (ij é conexão indireta)

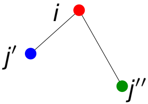
devolva (I, ϕ)

Aumentando α_j e β_{ij} de apenas um cliente



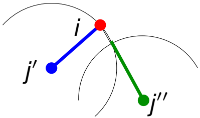
Aumentando α_j e β_{ij} de dois clientes

$f_i = 3$
$c_{ij'} = 3$
$c_{ij''} = 4$



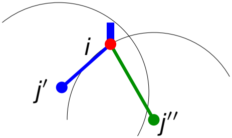
$\alpha_{j'} = 0$
$\alpha_{j''} = 0$
$\beta_{ij'} = 0$
$\beta_{ij''} = 0$

(1)



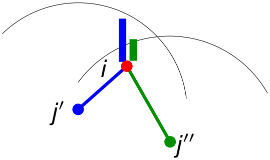
$\alpha_{j'} = 3$
$\alpha_{j''} = 3$
$\beta_{ij'} = 0$
$\beta_{ij''} = 0$

(2)



$\alpha_{j'} = 4$
$\alpha_{j''} = 4$
$\beta_{ij'} = 1$
$\beta_{ij''} = 0$

(3)



$\alpha_{j'} = 5$
$\alpha_{j''} = 5$
$\beta_{ij'} = 2$
$\beta_{ij''} = 1$

(4)

Para provar que FACILITY-JV é uma 3-aproximação, usaremos o teorema das folgas aproximadas com as seguintes condições:

Folgas aproximadas primais:

$$
\begin{array}{llcrcl}
x_{ij} = 1, & ij \text{ indireto} & \Rightarrow & \frac{1}{3}c_{ij} \leq & \alpha_j - \beta_{ij} & \leq c_{ij} \\
x_{ij} = 1, & ij \text{ direto} & \Rightarrow & & \alpha_j - \beta_{ij} & = c_{ij} \\
y_i = 1 & & \Rightarrow & & \sum_{j \in C} \beta_{ij} & = f_i
\end{array}
$$

e folgas complementares duais:

$$
\begin{array}{lcrcl}
\alpha_j > 0 & \Rightarrow & \sum_{i \in F} x_{ij} & = & 1 \\
\beta_{ij} > 0 & \Rightarrow & y_i & = & x_{ij}
\end{array}
$$

Note que o único lugar *frouxo* são as folgas para as arestas indiretas.

Lema: *(Folga aproximada primal 1)*

$$x_{ij} = 1 \text{ e } ij \text{ é direta} \Rightarrow \alpha_j - \beta_{ij} = c_{ij}$$

Prova.

Caso 1: $\beta_{ij} = 0$

Neste caso, α_j cresceu apenas para cobrir o custo c_{ij} pois i já estava aberta.

Caso 2: $\beta_{ij} > 0$

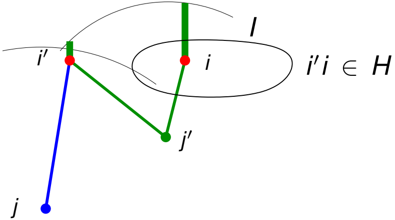
Neste caso, α_j pode passar de c_{ij}, mas quando isso ocorre, ij fica especial e β_{ij} cresce junto com α_j mantendo a igualdade $\alpha_j - \beta_{ij} = c_{ij}$.

□

Lema: *(Folga aproximada primal 2)*

$$x_{ij} = 1 \text{ e } ij \text{ é indireta} \Rightarrow \frac{1}{3} c_{ij} \leq \alpha_j - \beta_{ij} \leq c_{ij}$$

Prova. Neste caso, note que $\beta_{ij} = 0$ e $\alpha_j \leq c_{ij}$.
Agora, vamos mostrar que $c_{ij} \leq 3\alpha_j$.
Seja $i' := \tau(j)$. Deve existir $i' \in I$ e $j' \in C$ tal que



$$\begin{aligned} c_{ij} &\leq c_{i'j} + c_{i'j'} + c_{ij'} \\ &\leq \alpha_j + \alpha_{j'} + \alpha_{j'} \\ &\leq \alpha_j + \alpha_j + \alpha_j \quad \text{(pois raio de } j \text{ parou depois do de } j') \\ &= 3\alpha_j \end{aligned}$$

□

Lema: *(Folga aproximada primal 3)*

$$y_i = 1 \Rightarrow \sum_{j \in C} \beta_{ij} = f_i$$

Prova. Cada facilidade temporariamente aberta, em F_t, deve satisfazer a condição (c) (linha 11) pelo comando da linha 21:

$$\sum_{j \in C} \beta_{ij} = f_i$$

O resultado segue pois o conjunto I das facilidades abertas ($y_i = 1$) é subconjunto de F_t. □

Lema: *(Folga aproximada dual 1)*

$$\alpha_j > 0 \Rightarrow \sum_{i \in F} x_{ij} = 1.$$

Prova. Exercício. □

Lema: *(Folga aproximada dual 2)*

$$\beta_{ij} > 0 \Rightarrow y_i = x_{ij}$$

Prova. Exercício (basta considerar casos quando $y_i = 0$ e $y_i = 1$). □

Teorema: *(Jain e Vazirani'01)* FACILITY-JV *é uma 3-aproximação para o problema da* LOCALIZAÇÃO.

Prova. Segue do lema das folgas aproximadas e dos 5 lemas anteriores. □

Dual Fitting

Suponha (P) um problema primal do tipo $\big(\min cx \;\text{ sujeito a }\; x \in \mathcal{P}\big)$.

Suponha (D) um problema dual do tipo $\big(\max yb \;\text{ sujeito a }\; y \in \mathcal{D}\big)$.

Idéia:

- Obter uma solução inteira x para (P) e um vetor $\tilde{y}$ (guiado para ser um dual) mas $\tilde{y}$ não necessariamente viável para (D).
- Apesar de $\tilde{y}$ não ser viável, $\tilde{y}b$ “paga” o valor da solução primal.
- Obter fator f tal que $y \leftarrow \dfrac{\tilde{y}}{f}$ é viável para (D).
- Assim, $cx \leq \tilde{y}\,b = f\,\dfrac{\tilde{y}\,b}{f} = f\,y\,b \leq f\,\mathrm{OPT}$.

Problema COBERTURA POR CONJUNTOS: Dados conjunto E, subconjuntos $\mathcal{S}$ de E, custos $c : \mathcal{S} \to \mathbb{Q}_{\geq}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $\sum_{S\in\mathcal{S}'} c(S)$.

Relaxação e dual:

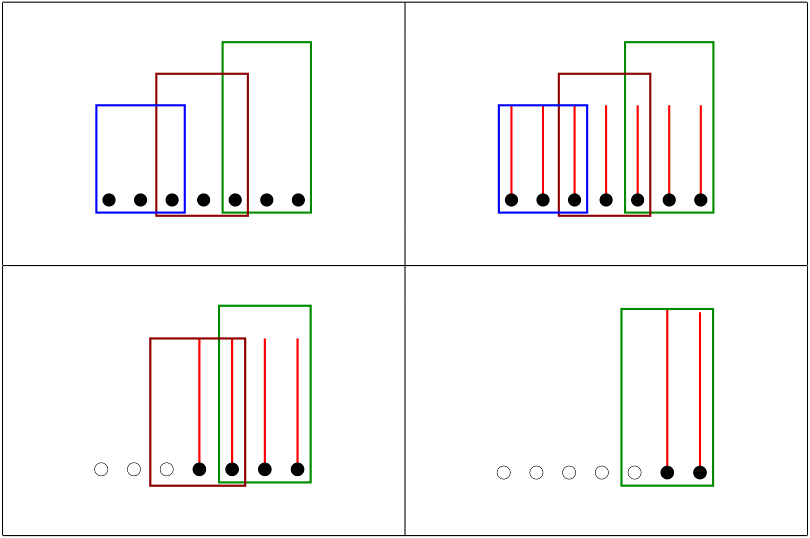
$$
(P)\quad
\begin{array}{lrcll}
\min & \displaystyle\sum_{S\in\mathcal{S}} c_S x_S & & & \\
 & \displaystyle\sum_{S\in\mathcal{S}_e} x_S & \geq & 1 & \forall e \in E \\
 & x_S & \geq & 0 & \forall S \in \mathcal{S},
\end{array}
$$

$$
(D)\quad
\begin{array}{lrcll}
\max & \displaystyle\sum_{e\in E} y_e & & & \\
 & \displaystyle\sum_{e\in S} y_e & \leq & c_S & \forall S \in \mathcal{S} \\
 & y_e & \geq & 0 & \forall e \in E
\end{array}
$$

onde $\mathcal{S}_e := \{S \in \mathcal{S} : \; e \in S\}$

Idéia: Aumentar (a partir de 0) variáveis $\tilde{y}$ de maneira a pagar por uma solução inteira. Depois obter fator f que transforma $\tilde{y}/f$ em dual viável.

Aumento uniforme de $\tilde{y}_e$, para todo elemento <u>ativo</u> $e \in A$



MinCC-Dual-Fitting $(\mathcal{S}, E, c)$

faça $\mathcal{S}' \leftarrow \emptyset$
faça $A \leftarrow E$ (conjunto de elementos ativos)
faça $\tilde{y}_e \leftarrow 0$ para todo $e \in E$
enquanto $A \neq \emptyset$ faça
  aumente uniformemente $\tilde{y}_e$ para todo $e \in A$ até que

$$\sum_{e \in R \cap A} \tilde{y}_e = c_R \quad \text{para algum} \quad R \in \mathcal{S} \setminus \mathcal{S}'$$

  $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{R\}$
  $A \leftarrow A \setminus R$
devolva $(\mathcal{S}', \tilde{y})$

Lema: *Seja $(\mathcal{S}', \tilde{y})$ solução devolvida por* MINCC-DUAL-FITTING. *Então,*

$$\sum_{S \in \mathcal{S}'} c_S = \sum_{e \in E} \tilde{y}_e$$

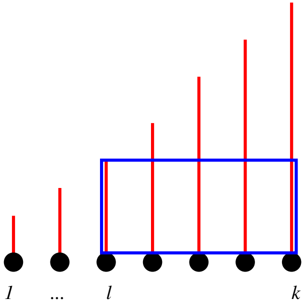
Prova. Exercício. □

Fato: *Note que $\tilde{y}$ pode não ser dual viável.*
Prova. Exercício □

Lema: *Seja $(\mathcal{S}', \tilde{y})$ devolvido por* MINCC-DUAL-FITTING *e $S = \{e_1, \ldots, e_k\} \in \mathcal{S}'$ e s.p.g., temos $\tilde{y}_{e_1} \leq \tilde{y}_{e_2} \leq \ldots \leq \tilde{y}_{e_k}$. Então*

$$\sum_{i=l}^{k} \tilde{y}_{e_l} = (k - l + 1)\tilde{y}_{e_l} \leq c(S)$$

Prova. Note que se $\tilde{y}_{e_l}$ aumentou até um valor t, todos os outros valores $\tilde{y}_{e_j}$ para $j = l, \ldots, k$ também chegaram a t, sem violar o custo de S.



□

Lema: *Se $\tilde{y}$ é devolvido por* MINCC-DUAL-FITTING *então* $\dfrac{\tilde{y}}{H_n}$ *é dual viável para* (D)*, onde* $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

Prova. Seja $R = \{e_1, \ldots, e_k\} \in \mathcal{S}'$ e s.p.g. $\tilde{y}_{e_1} \leq \tilde{y}_{e_2} \leq \ldots \leq \tilde{y}_{e_k}$.
Aplicando lema anterior para $l = 1, \ldots, k$ temos

$$\begin{cases} l = 1, & k\,\tilde{y}_{e_1} \leq c(R) & \Rightarrow & \dfrac{\tilde{y}_{e_1}}{c(R)} \leq \dfrac{1}{k} \\ l = 2, & (k-1)\,\tilde{y}_{e_2} \leq c(R) & \Rightarrow & \dfrac{\tilde{y}_{e_2}}{c(R)} \leq \dfrac{1}{k-1} \\ & & \vdots & \\ l = k, & \tilde{y}_{e_k} \leq c(R) & \Rightarrow & \dfrac{\tilde{y}_{e_k}}{c(R)} \leq 1 \end{cases}$$

Somando as desigualdades acima

$$\sum_{i=1}^{k} \frac{\tilde{y}_{e_i}}{c(R)} \leq H_k$$

$$\text{Assim, } \sum_{e \in R} \frac{\tilde{y}_e}{H_n} \leq c(R).$$

□

Teorema: *O algoritmo* MinCC-Dual-Fitting *é uma H_n-aproximação para o Problema da Cobertura por Conjuntos.*

Exercício

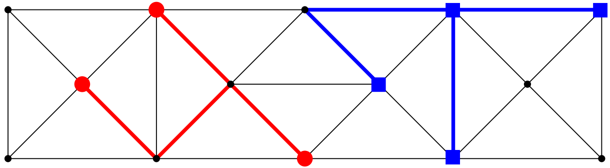
Reescreva o algoritmo MinCC-Dual-Fitting *como um algoritmo guloso.*

Problema da Floresta de Steiner

Seja G um grafo e $\mathcal{R}$ uma coleção de subconjuntos de V_G.

Def.: *Uma $\mathcal{R}$-floresta de G é qualquer floresta geradora F de G tal que para todo $R \in \mathcal{R}$, os elementos de R estão contidos em algum componente de F.*

Problema Floresta de Steiner: Dados um grafo G, um custo c_e em $\mathbb{Q}_\geq$ para cada aresta e e uma coleção $\mathcal{R}$ de subconjuntos de V_G, encontrar uma $\mathcal{R}$-floresta F que minimize $c(F)$.



Def.: *Um conjunto $S \subset V$ é chamado ativo se existe $R \in \mathcal{R}$ tal que*

$$R \cap S \neq \emptyset \qquad e \qquad R \setminus S \neq \emptyset$$

$\mathcal{S} := \{S \subset V : \ S \text{ é conjunto ativo}\}$ e
$\mathcal{S}_e := \{S \in \mathcal{S} : \ e \in \delta(S)\}$

Formulações (P) e (D):

$$(P) \quad \begin{array}{lrcll} \min & \sum_{e \in E} c_e x_e & & & \\ & \sum_{e \in \delta(S)} x_e & \geq & 1 & \forall S \in \mathcal{S}, \\ & x_e & \geq & 0 & \forall e \in E. \end{array}$$

$$(D) \quad \begin{array}{lrcll} \max & \sum_{S \in \mathcal{S}} y_S & & & \\ & \sum_{S \in \mathcal{S}_e} y_S & \leq & c_e & \forall e \in E, \\ & y_S & \geq & 0 & \forall S \in \mathcal{S}. \end{array}$$

Se existir x inteiro viável em (P) e y viável em (D) satisfazendo folgas α e β aproximadas, com $\alpha = 1$ e $\beta = 2$, temos

$$x_e = 1 \quad \Rightarrow \quad \sum_{S \in \mathcal{S}_e} y_S = c_e \qquad \text{(folgas aproximadas primais)}$$

$$y_S > 0 \quad \Rightarrow \quad \sum_{e \in \delta(S)} x_e \leq 2 \qquad \text{(folgas aproximadas duais)}$$

Então, pelo lema das folgas aproximadas, x é uma 2-aproximação. Mas

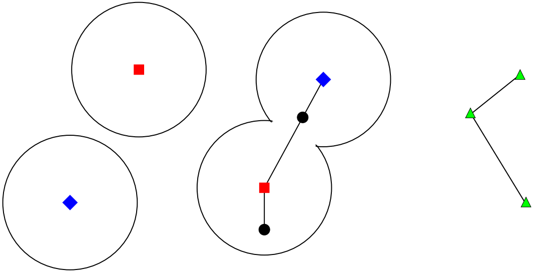
- Folgas aproximadas primais são fáceis de se obter.
- Folgas aproximadas duais são difíceis de se obter.

Estratégia: Usar uma nova folga aproximada dual mais flexível

Trocar
$$y_S > 0 \quad \Rightarrow \quad \sum_{e \in \delta(S)} x_e \leq 2$$

por
$$y_S > 0 \quad \Rightarrow \quad \frac{\displaystyle\sum_{S \in \mathcal{S}_F} \sum_{e \in \delta(S)} x_e}{|\mathcal{S}_F|} \leq 2$$

onde $\mathcal{S}_F$ é o conjunto dos componentes ativos da floresta sendo construída, em cada iteração do algoritmo.



Componentes ativos em uma iteração

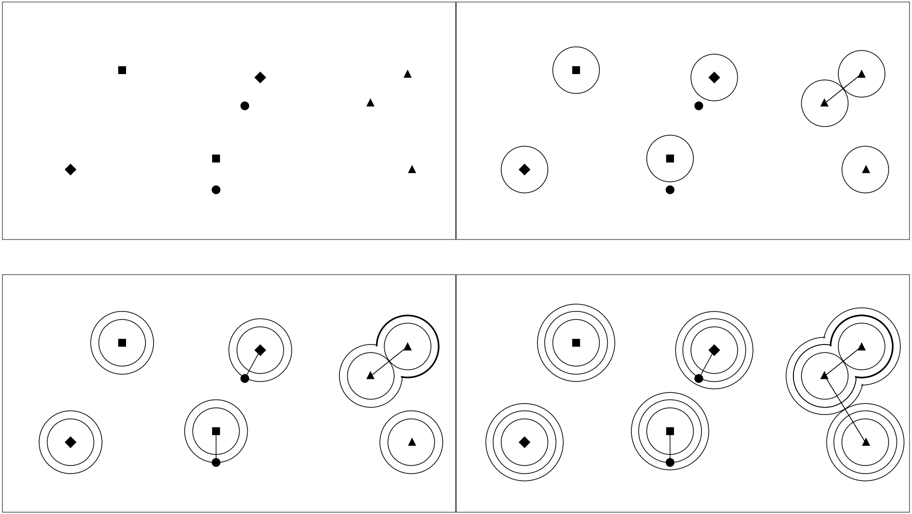
Observações sobre o algoritmo:

- Constrói dual viável que começa com 0.
- Usa número limitado de conjuntos ativos.
- Conjuntos ativos usados têm propriedade laminar.
- Folgas aproximadas primais válidas em cada iteração.
- Folgas aproximadas duais flexíveis válidas em cada iteração.
- No fim do algoritmo, as arestas desnecessárias são podadas.

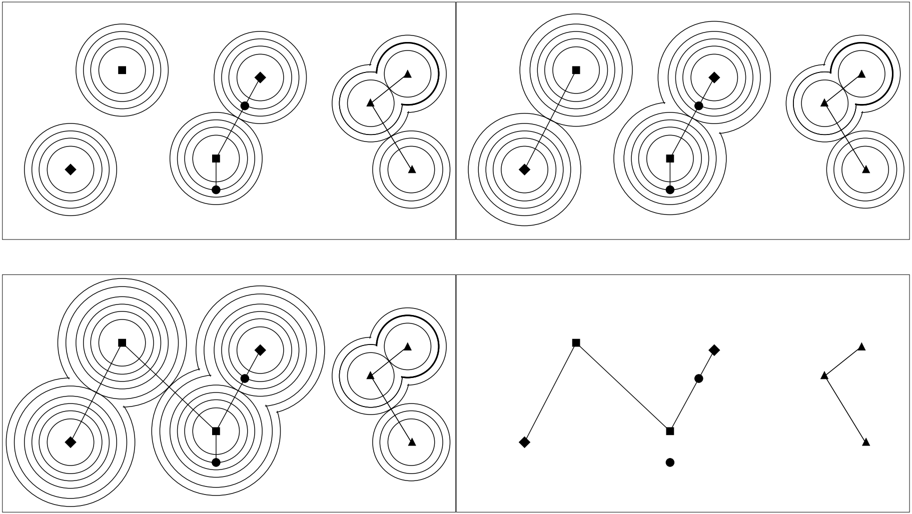
- F: Floresta sendo construída.
- $\mathcal{S}_F$: Componentes ativas de F
- Aresta externa: exatamente um extremo em uma componente de $\mathcal{S}_F$
- F_0: Floresta F logo antes da poda.
- F_1: Floresta final após a poda.

MINFS-GW $(G, c, \mathcal{R})$

```
F ← (V, ∅)
para cada S em 𝒮 faça y_S ← 0
enquanto 𝒮_F ≠ ∅ faça
    aumente uniformemente y_S até o máximo ∀S ∈ 𝒮_F mantendo
        Σ_{S∈𝒮_e} y_S ≤ c_e para cada aresta externa e
    seja f uma aresta externa tal que Σ_{S∈𝒮_f} y_S = c_f
    F ← F + f
F_0 ← F
seja F_1 uma 𝓡-floresta minimal de F_0
devolva F_1
```



Círculos representam o crescimento das variáveis duais.
Discos são steiner nodes (não têm requisitos de conectividade).

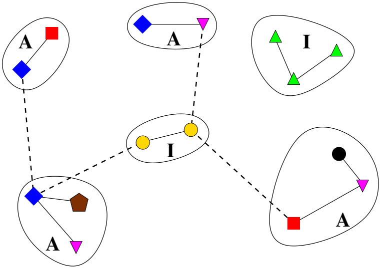


Componentes inativos não têm crescimento nas variáveis duais.
No final, é feita a poda das arestas desnecessárias.

Lema: *No início de cada iteração temos*

$$\frac{\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)|}{|\mathcal{S}_F|} \leq 2\,.$$

Prova. Seja $\mathcal{C}$ o conjunto de componentes de F no início da iteração. Seja $H = (\mathcal{C}, E_H)$ tal que $\{U, W\} \in E_H \Leftrightarrow \exists \{u, w\} \in F_1 : u \in U, w \in W$.



Como F_1 é minimal, todas as folhas de H são ativas.

Seja

$\mathcal{S}_F$ os componentes ativos de F e

$\mathcal{Z}_F$ os componentes inativos de F com grau não nulo.

Como H é floresta temos

$$\begin{aligned}
\sum_{S\in\mathcal{S}_F} |\delta_{F_1}(S)| + \sum_{S\in\mathcal{Z}_F} |\delta_{F_1}(S)| &= \sum_{S\in V_H} |\delta_H(S)| \\
&= 2|E_H| \\
&\leq 2(|\mathcal{S}_F| + |\mathcal{Z}_F| - 1) \\
&< 2|\mathcal{S}_F| + 2|\mathcal{Z}_F|
\end{aligned}$$

Portanto

$$\begin{aligned}
\sum_{S\in\mathcal{S}_F} |\delta_{F_1}(S)| &\leq 2|\mathcal{S}_F| + 2|\mathcal{Z}_F| - \sum_{S\in\mathcal{Z}_F} |\delta_{F_1}(S)| \\
&\leq 2|\mathcal{S}_F| .
\end{aligned}$$

□

Lema: *No início de cada iteração, vale a desigualdade*

$$\sum_{S \in \mathcal{S}} |\delta_{F_1}(S)|\, y_S \;\leq\; 2 \sum_{S \in \mathcal{S}} y_S\,,$$

Prova. Por indução no número de iterações:
Inicialmente $y = 0$ e a desigualdade vale.
Suponha que a desigualde vale no início de uma iteração.
Durante a iteração, y_S é acrescido de θ se e somente se $S \in \mathcal{S}_F$.
Assim, o lado esquerdo da desigualdade é acrescido de

$$\sum_{S \in \mathcal{S}_F} |\delta_{F_1}(S)|\theta$$

enquanto o lado direito é acrescido de

$$2|\mathcal{S}_F|\theta.$$

Pelo lema anterior, a desigualdade vale. □

Teorema: *O algoritmo* MinFS-GW *é uma 2-aproximação para o* MinFS*.*

Prova.

$$\begin{aligned}
c(F_1) &= \sum_{e \in F_1} c_e \\
&= \sum_{e \in F_1} \sum_{S \in \mathcal{S}_e} y_S && \text{(folgas 1-aprox. primais)} \\
&= \sum_{S \in \mathcal{S}} |\delta_{F_1}(S)|\, y_S && \text{(invertendo as somas)} \\
&\leq 2 \sum_{S \in \mathcal{S}} y_S && \text{(pelo lema anterior)} \\
&\leq 2\,\mathrm{OPT}(G, c, \mathcal{R})\,.
\end{aligned}$$

□

Exercício

Mostre que é possível melhorar um pouco a análise do teorema obtendo um fator de aproximação de $(2 - \frac{1}{n})$*, no lugar de 2.*

Axiomas de Probabilidade

Def.: *Um espaço de probabilidade tem 3 componentes:*

- Um espaço amostral Ω
- Uma família $\mathcal{F}$ de eventos, cada $E \in \mathcal{F}$ é t.q. $E \subseteq \Omega$.
- Função de probabilidade $\Pr : \mathcal{F} \to \mathbb{R}^+$

$E \in \mathcal{F}$ é dito ser simples ou elementar se $|E| = 1$

Iremos considerar apenas espaços discretos de probabilidade.

Def.: *Uma função de probabilidade é qualquer função* $\Pr : \mathcal{F} \to \mathbb{R}^+$ *t.q.*

- $\forall E \in \mathcal{F}$ temos $0 \leq \Pr(E) \leq 1$
- $\Pr(\Omega) = 1$
- Para toda seqüência finita ou enumerável de eventos mutuamente exclusivos $E_1, E_2, \ldots,$ temos

$$\Pr(E_1 \cup E_2 \ldots) = \Pr(E_1) + \Pr(E_2) + \ldots$$

Exemplo

Considere o lance de um dado de 6 lados.

- $\Omega = \{1, \ldots, 6\}$

Exemplo de eventos que podemos considerar

- E' = Evento do dado mostrar número par.
- E'' = Evento do dado mostrar número menor ou igual a 3.
- E''' = Evento do dado mostrar número primo.

Def.: *Dois eventos E e F são ditos serem independentes sse*

$$\Pr(E \cap F) = \Pr(E) \cdot \Pr(F)$$

e eventos $E_1, \ldots, E_k$ são mutuamente independentes sse $\forall I \subseteq \{1, \ldots, k\}$ temos

$$\Pr(\bigcap_{i \in I} E_i) = \Pi_{i \in I} \Pr(E_i).$$

Lema: *(Limitante da União) Dados eventos $E_1, E_2, \ldots$ temos*

$$\Pr(\bigcup_{i \geq 1} E_i) \leq \sum_{i \geq 1} \Pr(E_i)$$

Def.: *Uma variável aleatória discreta (v.a.) sobre* Ω *é uma função*

$X : \Omega \to \{\lambda_1, \ldots, \lambda_n\}$, *onde cada* λ_i *é um número real.*

Def.: *Um evento* $(X \in S)$ *é o conjunto* $X^{-1}(S)$ *para* $S \subseteq \{\lambda_1, \ldots, \lambda_n\}$.

Notação: Dado variável aleatória X e valor real a, o evento "$X = a$" representa o conjunto $\{e \in \Omega : X(e) = a\}$.
Analogamente para $(X \neq x)$, $(X < x)$, $(X \leq x)$, $(X > x)$ e $(X \geq x)$.

Def.: *A probabilidade do evento* $(X \in S)$ *é o número* $\Pr(X^{-1}(S))$, *denotado por* $\Pr(X \in S)$.

Assim,

$$\Pr(X = a) = \sum_{e \in \Omega:\ X(e) = a} \Pr(e).$$

Def.: *A esperança (ou valor esperado) da variável aleatória X é o número*

$$\mathbf{E}[X] := \sum_{i=1}^{n} \lambda_i \Pr(X=\lambda_i) \,.$$

Exemplo

Considere o lançamento de dois dados e X_i o valor do i-ésimo dado, $i = 1, 2$. Seja $X = X_1 + X_2$. Então

$$E[X] = 2\frac{1}{36} + 3 \cdot \frac{2}{36} + \cdots + 12 \cdot \frac{1}{36} = 7$$

LINEARIDADE DA ESPERANÇA

Teorema: *Para qualquer coleção finita de variáveis aleatórias discretas $X_1, \ldots, X_n$ com esperanças finitas*

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]$$

Observação: Note que não há restrições sobre a independência das variáveis aleatórias $X_1, \ldots, X_n$.

Lema: *Dados variável aleatória X e constante c, temos $E[c \cdot X] = c \cdot E[X]$.*

Exemplo

Considere o exemplo do lance de dois dados.

Seja X_1 *a v.a. do valor do primeiro dado.*
Seja X_2 *a v.a. do valor do segundo dado.*
Seja X *a v.a. da soma dos valores dos dois dados.*

Note que $X = X_1 + X_2$. *Assim,*

$$\begin{aligned} E[X] &= E[X_1 + X_2] \\ &= E[X_1] + E[X_2] \\ &= 2 \cdot \sum_{i=1}^{6} i \cdot \frac{1}{6} \\ &= 7 \end{aligned}$$

Variáveis Aleatórias de Bernoulli e Binomiais

Considere um experimento que tem probabilidade de sucesso p e falha de $1 - p$.

Seja Y uma v.a. tal que

$$Y = \begin{cases} 1 & \text{se experimento tem sucesso} \\ 0 & \text{caso contrário} \end{cases}$$

Então, Y é dita ser v.a. de Bernoulli ou v.a. Indicadora.

Lema: *Se Y é v.a. de Bernoulli com $\Pr(Y = 1) = p$, então $E[Y] = p$.*

Considere uma seqüência de n experimentos independentes, cada um com probabilidade de sucesso igual a p.

Se X é o número de sucessos nos n experimentos, dizemos que X tem distribuição binomial.

Def.: *Uma v.a. binomial (v.a.b.) X com parâmetros n e p, denotado por $B(n, p)$ é definida pela seguinte distribuição de probabilidade com $j = 0, 1, \ldots, n$:*

$$\Pr(X = j) = \binom{n}{j} p^j (1 - p)^{n-j}$$

Exercício

Mostre que $\sum_{j=0}^{n} \Pr(X = j) = 1.$

Lema: *Se X é uma v.a. binomial $B(n,p)$, então $E[X] = n \cdot p$.*

Prova. Seja X_i v.a. de bernoulli do i-ésimo experimento.

Então, $X = \sum_{i=1}^{n} X_i$ e portanto

$$\begin{aligned} E[X] &= \sum_{i=1}^{n} E[X_i] \\ &= n \cdot p \end{aligned}$$

□

DISTRIBUIÇÃO GEOMÉTRICA

Def.: *Uma v.a. X é dita ser geométrica (v.a.g.) com parâmetro p se tem distribuição*

$$\Pr(X = n) = (1 - p)^{n-1} \cdot p.$$

I.e., a probabilidade de jogar uma moeda $n - 1$ vezes com coroa (ou falha) e na n-ésima dar cara (sucesso).

Lema: *Se X é uma v.a.g. com parâmetro p, então*

$$E[X] = \frac{1}{p}.$$

Prova. Exercício. □

Lema: *(desigualdade de Markov) Se* $(\Omega, \Pr)$ *é um espaço discreto de probabilidade e* X *é uma variável aleatória sobre* Ω *cujos valores são todos não-negativos, então*

$$\Pr(X \geq \lambda) \leq \frac{1}{\lambda}\,\mathbf{E}[X]$$

para todo número positivo λ*.*

Prova. Se $\{\lambda_1, \ldots, \lambda_n\}$ é o conjunto de valores de X, então

$$\begin{aligned} \mathbf{E}[X] &= \sum_i \lambda_i \Pr(X = \lambda_i) \\ &\geq \sum_{\lambda_i \geq \lambda} \lambda \Pr(X = \lambda_i) \\ &= \lambda \Pr(X \geq \lambda)\,. \end{aligned}$$

□

Às vezes é útil reescrever a desigualdade de Markov assim:

$$\Pr(X \geq \lambda\,\mathbf{E}[X]) \leq \frac{1}{\lambda}.$$

Desigualdades úteis

Fato: *Se $m \geq 1$ e $|t| \leq m$ então*

$$\left(1 + \frac{t}{m}\right)^m \leq e^t.$$

Exemplo: Se $n \geq 1$ então

$$\left(1 + \frac{1}{n}\right)^n \leq e \quad \text{e} \quad \left(1 - \frac{1}{n}\right)^n \leq \frac{1}{e}.$$

Veja outras em
`http://dl.acm.org/citation.cfm?doid=242581.242585`

Algoritmos Aproximados Probabilísticos

Vamos supor a existência de uma função RAND. Dado $\rho \in [0, 1]$
RAND(ρ): devolve 1 com probabilidade ρ e 0 com probabilidade $1 - \rho$.

Def.: *Um algoritmo que usa a função* RAND *é chamado de probabilístico.*

Def.: *Um algoritmo probabilístico é polinomial se o número de chamadas de* RAND *e o consumo de tempo das demais operações é limitado por um polinômio no tamanho da instância.*

Def.: *Dado instância I e algoritmo probabilístico A, seja X_I a variável aleatória representando o valor da solução produzida por A sobre I. Dizemos que A é uma α-aproximação probabilística se*

$\mathbf{E}[X_I] \geq \alpha\,\mathrm{OPT}(I)$ *no caso de problema de maximização e*

$\mathbf{E}[X_I] \leq \alpha\,\mathrm{OPT}(I)$ *no caso de problema de minimização.*

Soluções Aproximadas com Alta Probabilidade

Seja $X_I \geq 0$ uma variável aleatória cujo valor é o valor da solução gerada pelo algoritmo A sobre I para um problema de minimização. Pela Desigualdade de Markov, temos

$$\begin{aligned}
\Pr(X_I \geq (\alpha+\epsilon)\mathrm{OPT}(I)) &\leq \frac{\mathbf{E}[X_I]}{(\alpha+\epsilon)\mathrm{OPT}(I)} \\
&\leq \frac{\alpha\,\mathrm{OPT}(I)}{(\alpha+\epsilon)\mathrm{OPT}(I)} \\
&= \frac{\alpha}{\alpha+\epsilon}\,.
\end{aligned}$$

Dado $\lambda \in (0,1)$, seja $k := \lceil \log_{\frac{\alpha}{\alpha+\epsilon}} \lambda \rceil$ e Y_I o melhor resultado obtido ao aplicar A sobre I k vezes. Então

$$\Pr(Y_I \geq (\alpha+\epsilon)\mathrm{OPT}(I)) \leq \lambda.$$

λ pequeno $\Rightarrow$ soluções dentro da aproximação com alta probabilidade.

Arredondamento Probabilístico

Problema COBERTURA POR CONJUNTOS: Dados conjunto E, subconjuntos $\mathcal{S}$ de E, custos $c : \mathcal{S} \to \mathbb{Q}_{\geq}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $\sum_{S \in \mathcal{S}'} c(S)$.

Relaxação:

$$
(P) \qquad \begin{array}{rrcll}
\min & \displaystyle\sum_{S \in \mathcal{S}} c_S x_S & & & \\
 & \displaystyle\sum_{S \in \mathcal{S}_e} x_S & \geq & 1 & \forall e \in E \\
 & x_S & \geq & 0 & \forall S \in \mathcal{S},
\end{array}
$$

onde $\mathcal{S}_e := \{S \in \mathcal{S} : \ e \in S\}$

Idéia: Resolver (P) e considerar cada x_S como uma probabilidade para obter S.

MinCC-AP1 $(\mathcal{S}, E, c)$

seja $\hat{x}$ solução (ótima) de (P)
seja $\mathcal{T} \leftarrow \emptyset$
para cada $S \in \mathcal{S}$ faça
    inclua S em $\mathcal{T}$ com probabilidade $\hat{x}_S$
devolva $\mathcal{T}$

Lema: *Se $\mathcal{T}$ é devolvido por* MinCC-AP1, *então* $E[\sum_{S \in \mathcal{T}} c_S] \leq \mathrm{OPT}$

Prova.

$$
\begin{aligned}
E[\sum_{S \in \mathcal{T}} c_S] &= \sum_{S \in \mathcal{S}} c_S \cdot \Pr(S \text{ pertencer a } \mathcal{T}) \\
&= \sum_{S \in \mathcal{S}} c_S \hat{x}_S \leq \mathrm{OPT}
\end{aligned}
$$

□

Lema: *Se $\mathcal{T}$ é devolvido por* MINCC-AP1 *e f é elemento de E, então* $\Pr(f$ *não ser coberto por* $\mathcal{T}) \leq \frac{1}{e} \approx 0,37$*.*

Prova. Seja $f \in E$ e s.p.g. considere $\mathcal{S}_f = \{S_1, \ldots, S_t\}$ os conjuntos que contêm f.

$$
\begin{aligned}
&\Pr(f \text{ não ser coberto por } \mathcal{T}) \\
&\qquad = \Pr(S_1 \notin \mathcal{T}) \cdot \Pr(S_2 \notin \mathcal{T}) \cdot \ldots \cdot \Pr(S_t \notin \mathcal{T}) \\
&\qquad = (1 - \hat{x}_{S_1}) \cdot (1 - \hat{x}_{S_2}) \cdot \ldots \cdot (1 - \hat{x}_{S_t}) \\
&\qquad \leq (1 - \frac{1}{t}) \cdot (1 - \frac{1}{t}) \cdot \ldots \cdot (1 - \frac{1}{t}) \\
&\qquad = (1 - \frac{1}{t})^t \\
&\qquad \leq \frac{1}{e}
\end{aligned}
$$

□

Aumentando as chances de obter cobertura
Versão Monte Carlo

MinCC-AP $(\mathcal{S}, E, c)$

seja $\mathcal{S}' \leftarrow \emptyset$ e $k = 2\log|E|$
para $i \leftarrow 1$ até k faça
    $\mathcal{T}_i \leftarrow$ MinCC-AP1 $(\mathcal{S}, E, c)$.
    $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathcal{T}_i$
devolva $\mathcal{S}'$

Lema: *Se $\mathcal{S}'$ é devolvido por* MinCC-AP *então* $E[c(\mathcal{S}')] \leq 2\log|E|\mathrm{OPT}$

Prova. Exercício. □

Lema: *Se $\mathcal{S}'$ é devolvido por* MinCC-AP *então* $\Pr(\mathcal{S}'$ *é cobertura de* $E) \geq 1 - \frac{1}{|E|}$.

Prova. Dado $f \in E$, temos que $\Pr(f$ não ser coberto por $\mathcal{T}_i) \leq \frac{1}{e}$.

Assim, $\Pr(f$ não ser coberto por $\mathcal{S}') \leq \left(\frac{1}{e}\right)^{2\log|E|} = \frac{1}{|E|^2}$.

Dado $e \in E$, seja $\mathcal{E}_e$ o evento do elemento e não ser coberto por $\mathcal{S}'$.

$$
\begin{aligned}
& \Pr(\mathcal{S}' \text{ não é cobertura}) \\
& \quad = \Pr\Big(\bigcup_{e \in E} \mathcal{E}_e\Big) \\
& \quad \leq \sum_{e \in E} \Pr(\mathcal{E}_e) \\
& \quad \leq \sum_{e \in E} \frac{1}{|E|^2} \\
& \quad = \frac{1}{|E|}
\end{aligned}
$$

$\square$

Versão Las Vegas

MINCC-AP-LAS-VEGAS $(\mathcal{S}, E, c)$

seja $\mathcal{S}' \leftarrow \emptyset$ e $i \leftarrow 1$
enquanto $\mathcal{S}'$ não é cobertura faça
  $\mathcal{T}_i \leftarrow$ MINCC-AP1 $(\mathcal{S}, E, c)$.
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathcal{T}_i$
  $i \leftarrow i + 1$
devolva $\mathcal{S}'$

Teorema: *Mostre que o número de passos esperado do algoritmo* MINCC-AP-LAS-VEGAS *é* $O(\log |E|)$.

Prova. Exercício. Dica: Divida as iterações em sequências de $2 \log |E|$ iterações: $S_1, S_2, \ldots$. Considere a possibilidade de parar na sequência S_i como uma variável geométrica. $\square$

Satisfatibilidade Máxima

Def.: *Uma literal é uma variável booleana ou sua negação.*

Def.: *Uma cláusula C sobre variáveis booleanas V, é uma disjunção de literais, todas associadas a variáveis distintas.*

Dada cláusula C sobre variáveis de V, denotaremos por

$C_0 \subseteq V$: é o conjunto das variáveis "complementadas"

$C_1 \subseteq V$: é o conjunto das variáveis "não-complementadas".

Exemplo: Se C é a cláusula $(a \vee \overline{b} \vee \overline{c})$ então,
$C_1 = \{a\}$ e $C_0 = \{b, c\}$.

Def.: *Uma valoração de V é um vetor x indexado por V com valores em $\{0, 1\}$.*

Def.: *A valoração x satisfaz C se $x_v = 1$ para algum $v \in C_1$ ou $x_v = 0$ para algum $v \in C_0$.*

Def.: *Uma fórmula está em Forma Normal Conjuntiva (FNC) se a fórmula é uma conjunção de cláusulas.*

Exemplo de fórmula booleana em FNC

$$\phi = (a \vee \overline{b} \vee \overline{c}) \wedge (\overline{a} \vee \overline{c}) \wedge (a \vee b \vee d) \wedge (d \vee \overline{c}) \wedge (d \vee \overline{a})$$

Problema MAXSAT $(V, \mathcal{C})$ Dada uma coleção $\mathcal{C}$ de cláusulas sobre um conjunto V de variáveis, encontrar uma valoração x de V que satisfaça o maior número possível de cláusulas de $\mathcal{C}$.

Teorema: *O problema* MAXSAT *é um problema NP-difícil.*

Algoritmo de Johnson

Atribui 0 ou 1 para as variáveis com probabilidade $\frac{1}{2}$

MAXSAT-JOHNSON $(V, \mathcal{C})$

1 para cada v em V faça
2 $\dot{x}_v \leftarrow$ RAND $(\frac{1}{2})$
3 devolva $\dot{x}$

onde RAND (p), para $p \in [0, 1]$, devolve 1 com probabilidade p e 0 com probabilidade $1 - p$.

Teorema: *Se X é a variável cujo valor é o número de cláusulas satisfeitas por uma valoração produzida por* MAXSAT-JOHNSON *e toda cláusula tem pelo menos k variáveis então*

$$\mathbf{E}[X] \geq \left(1 - \frac{1}{2^k}\right) \mathrm{OPT}(V, \mathcal{C}) .$$

Prova. Para cada cláusula C, defina uma variável aleatória Z_C como:

$Z_C := 1$ se valoração produzida satisfaz C e
$Z_C := 0$ em caso contrário.

$$\Pr(Z_C{=}0) \leq 1/2^k \quad \Rightarrow \quad \Pr(Z_C{=}1) \geq 1 - 1/2^k.$$

Assim,

$$\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}[\sum_{C \in \mathcal{C}} Z_C] \\
&= \sum_{C \in \mathcal{C}} \mathbf{E}[Z_C] \\
&\geq (1 - \frac{1}{2^k})|\mathcal{C}| \\
&\geq (1 - \frac{1}{2^k})\,\mathrm{OPT}(V, \mathcal{C})
\end{aligned}$$

$\square$

Teorema: *O algoritmo* MAXSAT-JOHNSON *é uma* 0,5*-aproximação probabilística polinomial para o* MAXSAT*.*

Prova. Direto do teorema anterior, para $k = 1$. $\square$

Algoritmo de arredondamento probabilístico

Dado valor $\rho \in (0,1)$, arredondar ρ com probabilidade ρ

$$
(P) \qquad
\begin{array}{rrcll}
\max & \displaystyle\sum_{C \in \mathcal{C}} z_C & & & \\
& \displaystyle\sum_{v \in C_0} (1 - x_v) + \sum_{v \in C_1} x_v & \geq & z_C & \forall\, C \in \mathcal{C}\,, \\
& 0 \;\leq\; z_C & \leq & 1 & \forall\, C \in \mathcal{C}\,, \\
& 0 \;\leq\; x_v & \leq & 1 & \forall\, v \in V\,.
\end{array}
$$

MAXSAT-GW $(V, \mathcal{C})$

```
seja (x̂, ẑ) uma solução ótima racional de (P)
para cada v em V faça
    ẋ_v ← RAND (x̂_v)
devolva ẋ
```

Teorema: *Se $(V, \mathcal{C})$ é instância do* MAXSAT*, cada cláusula com no máximo k variáveis e $X_{\mathcal{C}}$ é a variável aleatória cujo valor é o número de cláusulas satisfeitas por uma valoração produzida por* MAXSAT-GW $(V, \mathcal{C})$*, então*

$$\mathbf{E}[X_{\mathcal{C}}] \;\geq\; \left(1 - \left(1 - \frac{1}{k}\right)^{k}\right) \mathrm{OPT}(V, \mathcal{C})\,.$$

Prova. Seja $C \in \mathcal{C}$ e Z_C a variável com valor 1 se a valoração produzida satisfaz C e 0 caso contrário. Se t é o número de variáveis de C então

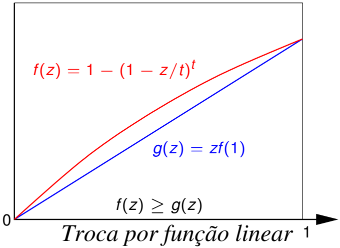
$$\begin{aligned}
\Pr(Z_C{=}1) &= 1 - \prod_{v \in C_0} \hat{x}_v \prod_{v \in C_1} (1 - \hat{x}_v) \\
&\geq 1 - \left(\frac{\sum_{v \in C_0} \hat{x}_v + \sum_{v \in C_1} (1 - \hat{x}_v)}{t}\right)^{t} \\
&= 1 - \left(\frac{t - \sum_{v \in C_0} (1 - \hat{x}_v) - \sum_{v \in C_1} \hat{x}_v}{t}\right)^{t}
\end{aligned}$$

$$\geq 1 - \left(\frac{t - \hat{z}_C}{t}\right)^t$$

$$= 1 - \left(1 - \frac{\hat{z}_C}{t}\right)^t$$

$$\geq \left(1 - \left(1 - \frac{1}{t}\right)^t\right)\hat{z}_C$$

$$\geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right)\hat{z}_C .$$



□

Teorema: *O algoritmo* MAXSAT-GW *é uma* 0,63*-aproximação probabilística polinomial para o* MAXSAT*.*

Prova. Segue do teorema anterior junto com a seguinte desigualdade

$$\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e} \quad \forall k \geq 1$$

□

Algoritmo Combinado

MaxSat-Johnson: Melhor performance para cláusulas grandes

MaxSat-GW: Melhor performance para cláusulas pequenas

Idéia: Pegar a melhor solução entre MaxSat-Johnson e MaxSat-GW

MaxSat-Combinado-GW $(V, \mathcal{C})$

```
ẋ ← MaxSat-Johnson (V)
ẍ ← MaxSat-GW (V, 𝒞)
seja ṡ o número de cláusulas de 𝒞 satisfeitas por ẋ
seja s̈ o número de cláusulas de 𝒞 satisfeitas por ẍ
se ṡ ≥ s̈
    então devolva ẋ
    senão devolva ẍ
```

Teorema: *O algoritmo* MAXSAT-COMBINADO-GW *é uma* 0,75*-aproximação probabilística polinomial para o* MAXSAT*.*

Prova. Seja $\mathcal{C}_k := \{C \in \mathcal{C} : |C| = k\}$ (cláusulas com k variáveis),
Seja $X_{\mathcal{C}}$ var. aleat. do valor da solução gerada por MAXSAT-COMBINADO-
Seja $\dot{X}_{\mathcal{C}}$ var. aleat. do valor da solução gerada por MAXSAT-JOHNSON.
Seja $\ddot{X}_{\mathcal{C}}$ var. aleat. do valor da solução gerada por MAXSAT-GW.

$$
\begin{aligned}
\mathbf{E}[X_{\mathcal{C}}] &\geq \mathbf{E}[\frac{1}{2}(\dot{X}_{\mathcal{C}} + \ddot{X}_{\mathcal{C}})] \;=\; \frac{1}{2}(\mathbf{E}[\dot{X}_{\mathcal{C}}] + \mathbf{E}[\ddot{X}_{\mathcal{C}}]) \\
&\geq \frac{1}{2}\sum_{k}\sum_{C\in\mathcal{C}_k}\left((1-2^{-k}) + (1-(1-k^{-1})^{k})\hat{z}_C\right) \\
&\geq \frac{1}{2}\sum_{k}\sum_{C\in\mathcal{C}_k}\left(1-2^{-k} + 1-(1-k^{-1})^{k}\right)\hat{z}_C \\
&\geq \frac{1}{2}\sum_{k}\sum_{C\in\mathcal{C}_k}\frac{3}{2}\,\hat{z}_C \\
&\geq \frac{3}{4}\,\mathrm{OPT}(V,\mathcal{C})\,.
\end{aligned}
$$

□

Desaleatorização

Método das Esperanças Condicionais (Erdős & Selfridge' 74):

Estratégia:

- Cálculo das esperanças condicionais eficientemente
- Iterativamente fazer a próxima escolha sem piorar o valor esperado.
- Determinação de todas as escolhas nos leva a um algoritmo determinístico com valor não pior que o valor esperado inicial.

Exemplo: Desaleatorização do Algoritmo de Johnson

MAXSAT-JOHNSON (V)

1 para cada v em V faça

2 $\quad \dot{x}_v \leftarrow$ RAND $(\frac{1}{2})$

3 devolva $\dot{x}$

$E[X]$

$E[X|x_1 = 0]$

$E[X|x_1 = 0, x_2 = 0]$

$E[X|x_1 = 0, x_2 = 0, x_3 = 0]$

$E[X|x_1 = 0, x_2 = 0, x_3 = 1]$

$E[X|x_1 = 0, x_2 = 1]$

$E[X|x_1 = 0, x_2 = 1, x_3 = 0]$

$E[X|x_1 = 0, x_2 = 1, x_3 = 1]$

$E[X|x_1 = 1]$

$E[X|x_1 = 1, x_2 = 0]$

$E[X|x_1 = 1, x_2 = 0, x_3 = 0]$

$E[X|x_1 = 1, x_2 = 0, x_3 = 1]$

$E[X|x_1 = 1, x_2 = 1]$

$E[X|x_1 = 1, x_2 = 1, x_3 = 0]$

$E[X|x_1 = 1, x_2 = 1, x_3 = 1]$

Escolhas de x_1, x_2 e x_3 tais que

$$E[X] \leq E[X|x_1] \leq E[X|x_1, x_2] \leq E[X|x_1, x_2, x_3]$$

MAXSAT-JOHNSON-DESALEATORIZADO $(V, \mathcal{C})$

$\mathcal{D} \leftarrow \mathcal{C}$
para cada v em V faça
    se ESPCOND $(v, 1, \mathcal{D}) \geq$ ESPCOND $(v, 0, \mathcal{D})$
        então $\dot{x}_v \leftarrow 1$
            para cada C em $\mathcal{D}$ faça
                se $v \in C_1$
                    então $\mathcal{D} \leftarrow \mathcal{D} \setminus \{C\}$
                    senão $C_0 \leftarrow C_0 \setminus \{v\}$
        senão $\dot{x}_v \leftarrow 0$
            para cada C em $\mathcal{C}$ faça
                se $v \in C_0$
                    então $\mathcal{D} \leftarrow \mathcal{D} \setminus \{C\}$
                    senão $C_1 \leftarrow C_1 \setminus \{v\}$
devolva $\dot{x}$

Procedimento para calcular esperança condicional:

v: variável booleana
i: valor atribuído para a variável v
$\mathcal{D}$: Conjunto de cláusulas

Procedimento ESPCOND $(v, i, \mathcal{D})$

$esp \leftarrow 0$
para cada C em $\mathcal{D}$ faça
    $k \leftarrow |C_1| + |C_0|$
    se $v \in C_i$
        então $esp \leftarrow esp + 1$
        senão se $v \in C_{1-i}$
            então $esp \leftarrow esp + (1 - 2^{-k+1})$
            senão $esp \leftarrow esp + (1 - 2^{-k})$
devolva esp

Teorema: *O algoritmo* MAXSAT-JOHNSON-DESALEATORIZADO *é uma 0,5-aproximação polinomial para o* MAXSAT.

Prova. Seja X a variável aleatória do número de cláusulas em $\mathcal{C}$ satisfeitas por uma valoração produzida pelo algoritmo MAXSAT-JOHNSO
Como probabilidade de $\dot{x}_v = 1$ é $\frac{1}{2}$ e probabilidade de $\dot{x}_v = 0$ é $\frac{1}{2}$, temos

$$\begin{aligned} \mathbf{E}[X] &= \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}1] + \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}0] \\ &\leq \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}i] + \frac{1}{2}\mathbf{E}[X|\dot{x}_v{=}i] \\ &= \mathbf{E}[X|\dot{x}_v{=}i] \;=\; \mathbf{E}[X|\dot{x}_v] \end{aligned}$$

onde i é o valor escolhido pelo algoritmo para a variável $\dot{x}_v$.
O mesmo raciocínio segue para outras variáveis. I.e.,

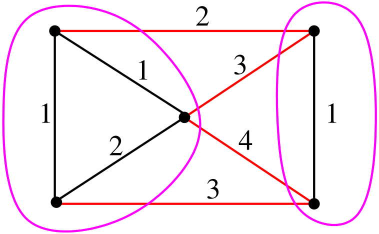
$$\mathbf{E}[X] \leq \mathbf{E}[X|\dot{x}_{v_1}] \leq \mathbf{E}[X|\dot{x}_{v_1}, \dot{x}_{v_2}] \leq \ldots \leq \mathbf{E}[X|\dot{x}_{v_1}, \ldots, \dot{x}_{v_n}] \,.$$

Como $0{,}5\,\mathrm{OPT}(V, \mathcal{C}) \leq \mathbf{E}[X]$ e o último termo é um valor determinístico, o teorema segue. $\square$

Programação Semidefinida

Problema MAXCUT: Dados um grafo $G = (V, E)$ e um peso w_e em $\mathbb{Q}_{\geq}$ para cada aresta e, encontrar um corte R que maximize $w(R)$.

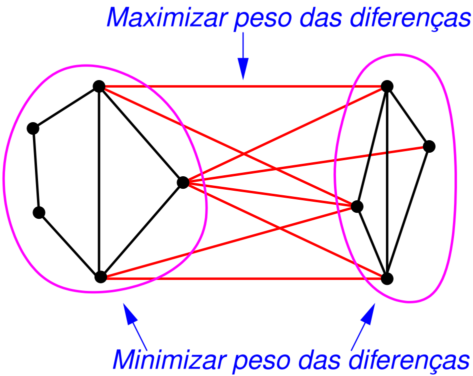
Exemplo:



Teorema: MAXCUT *é NP-difícil.*

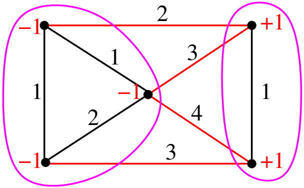
Teorema: *Se P≠NP então* MAXCUT *não aproximável em tempo polinomial em* $16/17 - \epsilon$ *(Håstad'97).*

Aplicação: Partição por similaridade



Formulação Quadrática: Encontrar x que

$$\begin{array}{lll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1 - x_i x_j) & \\ & x_i x_i = 1 & \forall i \in V \end{array}$$



Se $x_i x_j = -1$

$$w_{ij}(1 - x_i x_j) = w_{ij}(1 - (-1)) = 2w_{ij}$$

Se $x_i x_j = +1$

$$w_{ij}(1 - x_i x_j) = w_{ij}(1 - (+1)) = 0$$

Relaxação Vetorial

- Trocar (relaxar) x_i por vetor $Y_{i\star}$ (n-dimensional)

$$\boxed{x_i} \Longrightarrow \boxed{\qquad\qquad Y_{i\star} \qquad\qquad}$$

- I.e., trocar x por uma matriz Y.

$$\begin{array}{c} x \\ \begin{array}{|c|} \hline x_1 \\ \hline x_2 \\ \hline \vdots \\ \hline x_n \\ \hline \end{array} \end{array} \Longrightarrow \begin{array}{c} Y \\ \begin{array}{|c|} \hline \qquad Y_{1\star} \qquad \\ \hline Y_{2\star} \\ \hline \vdots \\ \hline Y_{n\star} \\ \hline \end{array} \end{array}$$

- $Y_{i\star} \cdot Y_{i\star}$: Vetor na esfera unitária

$$x_i \cdot x_j \Rightarrow Y_{i\star} \cdot Y_{j\star} = (YY^\top)_{ij}$$

$$(YY^\top)_{ij} = \left(\begin{array}{|c|} \hline \vdots \\ \hline Y_{i\star} \\ \hline \vdots \\ \hline \end{array} \bullet \begin{array}{|c|c|c|} \hline \ldots & Y_{j\star} & \ldots \\ \hline \end{array} \right)_{ij}$$

Formulação Quadrática:

$$(Q) \quad \begin{array}{lll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1 - x_i x_j) & \\ & x_i x_i = 1 & \forall i \in V \end{array}$$

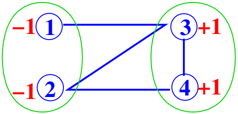
$$\Downarrow$$

Formulação Relaxada:

$$(R) \quad \begin{array}{lll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1 - (Y\,Y^\top)_{ij}) & \\ & (Y\,Y^\top)_{ii} = 1 & \forall i \in V \end{array}$$

Relaxação Vetorial

$$\begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ x = & \begin{bmatrix} -1 & -1 & +1 & +1 \end{bmatrix} \end{array}$$



$$Y = \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{bmatrix} -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 \end{bmatrix} \qquad Y^{T} = \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} -1 & -1 & +1 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$x_i x_j = (YY^T)_{ij} \qquad \text{e} \qquad (YY^T)_{ii} = 1$$

I.e.,

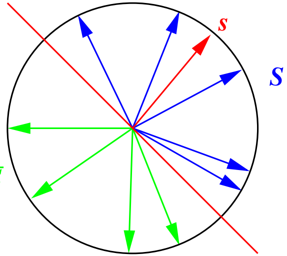
$$\text{Max } \frac{1}{2} \sum_{ij \in E} w_{ij}(1 - (Y\,Y^{\top})_{ij}) \geq \mathrm{OPT}(G, w)$$

Algoritmo de Goemans e Williamson

Idéia: Distribuir vetores na esfera unitária, considerando forças de repuls

MAXCUT-GW (G, w)

$\hat{Y} \leftarrow$ solução ótima de (R)
$s \leftarrow$ RANDESFERA(V)
$S \leftarrow \{\, i \in V : s\hat{Y}_{i\star} > 0 \,\}$
devolva $\delta(S)$

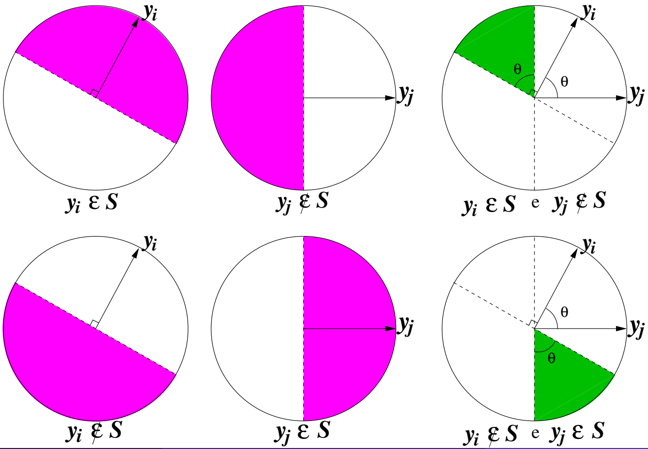


Lema: $\Pr(\, ij \in \delta(S)\,) \;\geq\; \frac{1}{\pi} \operatorname{arccos}((\hat{Y}\,\hat{Y}^\top)_{ij})\,.$

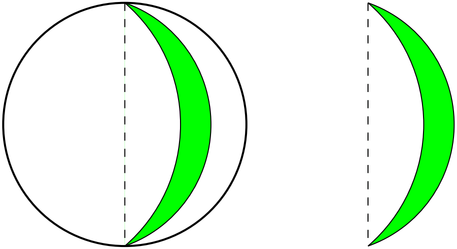
Prova.

$$\Pr(\, ij \in \delta(S)\,) = \Pr(\, sy_i > 0 \text{ e } sy_j \leq 0\,) + \Pr(\, sy_i \leq 0 \text{ e } sy_j > 0\,)\,,$$

onde $y_i = \hat{Y}_{i\star}$ e $y_j = \hat{Y}_{j\star}$.



Exemplo no $\mathbb{R}^3$ (fatia).



$$
\begin{aligned}
\Pr(\, ij \in \delta(S)\,) &= \frac{\theta}{2\pi} + \frac{\theta}{2\pi} = \frac{\theta}{\pi} \\
&= \frac{\arccos(y_i y_j)}{\pi} \\
&= \frac{1}{\pi} \arccos\left((\hat{Y}\,\hat{Y}^\top)_{ij}\right)
\end{aligned}
$$

Teorema: $\mathbf{E}[w(\delta(S))] \geq 0{,}878\,\mathrm{OPT}(G, w)$.

Prova.

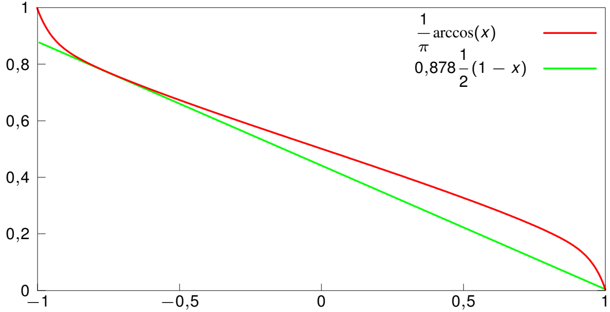
$$
\begin{aligned}
\mathbf{E}[w(\delta(S))] &= \sum_{ij \in E} w_{ij} \Pr(\, ij \in \delta(S)\,) \\
&\geq \sum_{ij \in E} w_{ij} \frac{1}{\pi} \operatorname{arccos}((\hat{Y}\hat{Y}^\top)_{ij}) \\
&\geq 0{,}878 \frac{1}{2} \sum_{ij \in E} w_{ij}(1 - (\hat{Y}\hat{Y}^\top)_{ij}) \quad \text{Troca por função linear} \\
&\geq 0{,}878\,\mathrm{OPT}(G, w) .
\end{aligned}
$$

□

Teorema: MAXCUT-GW *é uma* 0,878*-aproximação probabilística polinomial.*

Teorema: *O algoritmo* MAXCUT-GW *pode ser desaleatorizado (Mahajan e Ramesh' 95).*

$$\frac{1}{\pi}\arccos(x) \;\geq\; 0{,}878\,\frac{1}{2}(1-x)\,.$$



Troca por função linear

Programa Vetorial $\times$ Programa Semidefinido

Def.: *X é* **positiva semidefinida** $(X \succeq 0)$,
se $\exists Y$ quadrada tal que $X = Y Y^\top$.

Programa Vetorial:

$$\begin{array}{ll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1-(YY^\top)_{ij}) \\ & (YY^\top)_{ii} = 1 \qquad\qquad \forall i \in V \end{array}$$

$$\Downarrow$$

Programa Semidefinido:

$$\begin{array}{ll} \text{Max} & \frac{1}{2}\sum_{ij\in E} w_{ij}(1-X_{ij}) \\ & X_{ii} = 1 \qquad\qquad \forall i \in V \\ & X \succeq 0 \end{array}$$

Programas Semidefinidos

Programa Semidefinido: Encontrar matriz quadrada X tal que

$$\begin{array}{lrcll} \text{Min} & \sum_{ij \in V \times V} w_{ij}(X_{ij}) & & & \\ & A_{kij} X_{ij} & = & b_k & \forall k \in M \\ & X & \succeq & 0 & \end{array}$$

- Dado uma matriz positiva semidefinida X, é possível obter uma matriz Y tal que $X = YY^T$ em tempo polinomial (modelo com números reais).
- Soluções de um programa semidefinido não necessariamente racional.
- Podemos encontrar soluções arbitrariamente próximas da ótima através do Método do Elipsóide e do Algoritmo de Pontos Interiores.
- Obtenção de soluções ótimas ou bem próximas das ótimas (inevitável pois soluções ótimas podem ser irracionais) [Goemans & Williamson, Homer & Peinado, Poljak & Rendl]
- RandEsfera pode ser implementado de maneira aproximada por geradores de números pseudo-aleatórios em [0,1] (Knuth'98)

PTAS para Escalonamento de Tarefas

Problema ESCALONAMENTO: Dados uma lista de tarefas $L = (J_1, \ldots, J_n)$, cada uma com tempo $t(J_i)$ e m máquinas idênticas, encontrar uma partição de L, $(M_1, \ldots, M_m)$, tal que $\max_i t(M_i)$ é mínimo.

Teorema: *O problema* ESCALONAMENTO *é um problema fortemente* NP*-completo.*

Corolário: *Não existe FPTAS para* ESCALONAMENTO*, a menos que* $\mathrm{P} = \mathrm{NP}$*.*

Vamos apresentar um PTAS para ESCALONAMENTO

Def.: *Um algoritmo R_ϵ é dito ser $(1+\epsilon)$-restrito para o problema* ESCALONAMENTO *se dado ϵ, tempo T e uma instância (L, t, m) devolve:*

- $\emptyset$ *- se não existe um escalonamento com tempo T ou*
- $\mathcal{S}$ *- onde $\mathcal{S}$ é um escalonamento com $\mathrm{val}(\mathcal{S}) \leq (1+\epsilon)T$.*

Lema: *Seja $L := \max\{\max_j t(J_j)\,, \frac{\sum_j t(J_i)}{m}\}$, então*

$$\mathrm{OPT}(L) \in [L, 2L]$$

Prova. Note que o algoritmo ESCALONAMENTO-GRAHAM devolve um escalonamento limitado por $\max_j t(J_j) + \frac{\sum_j t(J_i)}{m} \leq 2L$ □

Idéia: Construir algoritmo $(1+\epsilon)$-restrito e fazer busca binária.

Algoritmo de Hochbaum e Shmoys'88

Este algoritmo usa um algoritmo $(1+\epsilon)$-restrito R_ϵ como subrotina.

ESCALONAMENTO-HS$_\epsilon$ (L, m, t)

$\mathcal{S} \leftarrow$ ESCALONAMENTO-GRAHAM(L, m, t)
$T' \leftarrow \max\{\max_j t(J_j)\,,\ \frac{\sum_j t(J_i)}{m}\}$
$T'' \leftarrow 2T'$
$\epsilon' \leftarrow \frac{\epsilon}{3}$
enquanto $T'' - T' > \epsilon' T'$ faça
    seja $T \leftarrow \frac{T'+T''}{2}$
    $\mathcal{S}' \leftarrow R_{\epsilon'}(L, m, t, T)$
    se $\mathcal{S}' \neq \emptyset$ então
        $T'' \leftarrow T$;
        $\mathcal{S} \leftarrow \mathcal{S}'$;
    senão
        $T' \leftarrow T$
devolva $\mathcal{S}$

Teorema: *(Hochbaum & Shmoys'88)* ESCALONAMENTO-HS *é um PTAS para o problema do Escalonamento.*

Prova. A cada iteração temos escalonamento $\mathcal{S}'$ e intervalo $[T', T'']$ que:

- $\text{val}(\mathcal{S}') \in [T', T''(1+\epsilon')]$,
- $T'' \leq T' + \epsilon T'$
- $T' \leq \text{OPT}$.

O algoritmo pára com escalonamento $\mathcal{S}$ que

$$\begin{aligned} \text{val}(\mathcal{S}) &\leq (1+\epsilon')T'' \\ &\leq (1+\epsilon')(T' + \epsilon' T') \\ &= (1+\epsilon')^2 T' \\ &\leq (1 + 2\epsilon' + \epsilon'^2)\text{OPT} \\ &\leq (1+\epsilon)\text{OPT} \end{aligned}$$

Complexidade de tempo: polinomial no tamanho da instância e em $\frac{1}{\epsilon}$.
$\square$

Algoritmo $(1+\epsilon)$-restrito

Subrotina: algoritmo $(1+\epsilon)$-restrito, RG_ϵ, para tarefas com tempo maior que ϵT.

$R_\epsilon(L, m, t, T)$
$G \leftarrow \{j \in L : t(j) > \epsilon T\}$
$P \leftarrow L \setminus G$
$\mathcal{S}_G \leftarrow RG_\epsilon(G, m, t, T)$
se $\mathcal{S}_G = \emptyset$ então devolva $\emptyset$
senão
    preencha $\mathcal{S}_G$ com itens em P usando ESCALONAMENTO-GRAHAM
    se $\mathrm{val}(\mathcal{S}) > (1+\epsilon)T$ então devolva $\emptyset$
    senão devolva $\mathcal{S}$

Lema: *O algoritmo R_ϵ é um algoritmo $(1+\epsilon)$-restrito.*

Prova. Exercício. □

Algoritmo $(1+\epsilon)$-restrito - Itens Grandes

Subrotina: algoritmo exato, $Exato_{\epsilon,k}$, para encontrar escalonamento em tempo T, quando há até k tempos diferentes.
Se não existir tal escalonamento devolve $\emptyset$.

Idéia: Definir tempos fixos e arredondar tempos de processamento para baixo.

$RG_\epsilon(G, m, t, T)$

para cada $j \in G$ faça
  seja i tal que $t(j) \in [\ \epsilon T + i\epsilon^2 T\ \ ,\ \ \epsilon T + (i+1)\epsilon^2 T\ )$
  $t'(j) \leftarrow \epsilon T + i\epsilon^2 T$
$k \leftarrow \lfloor 1/\epsilon^2 \rfloor$
$\mathcal{S} \leftarrow Exato_{\epsilon,k}(G, m, t', T)$
devolva a partição $\mathcal{S}$

Lema: *O algoritmo RG_ϵ é um algoritmo $(1+\epsilon)$-restrito.*

Prova.

Note que $t'(j) \in \{\epsilon T,\ \epsilon T + \epsilon^2 T, \ldots, \epsilon T + (k-1)\epsilon^2 T,\ \ \epsilon T + k\,\epsilon^2 T\}$.

Seja $\mathcal{S} \neq \emptyset$ um escalonamento gerado por $RG_\epsilon(G, m, t', T)$.

Note que

- há no máximo $\frac{1}{\epsilon}$ tarefas em cada processador;
- perda devido ao arredondamento é no máximo $\epsilon^2 T$.

Portanto o tempo total de processamento em um processador em $\mathcal{S}$ usando tempo de processamento t é no máximo $(1+\epsilon)T$. $\square$

Algoritmo exato para tarefas grandes e tempos restritos

Dado instância (G, m, t', T) e inteiro k, onde $t'(j) \geq \epsilon T$ e há no máximo k tempos distintos em t', encontrar escalonamento $\mathcal{S}$ com tempo no máximo T, se existir.
Daremos a versão de decisão deste algoritmo:
Decidir se um conjunto L de tarefas pode ser executado em tempo T usando no máximo m processadores

Estratégia: Programação Dinâmica.

- Número de possiblidades de se escalonar apenas em uma máquina é polinomial.
- Para m máquinas, teste todas as possibilidades em uma máquina e verifique para cada possibilidade se as demais tarefas podem ser executadas em $m-1$ máquinas.

DecisãoExato$_{\epsilon,l}(G, m, t', T)$

para $i = 1$ até k faça
    $s_i \leftarrow \epsilon T + (i-1)\epsilon^2 T$
    $n_i \leftarrow |\{j \in G : t'(j) = s_i\}|$
seja $\mathcal{Q} \leftarrow \{(a_1, \ldots, a_k) : \sum_{i=1}^{k} a_i s_i \leq T$ e $0 \leq a_i \leq n_i\}$
$M(0, \ldots, 0) \leftarrow 0$
para cada $(a_1, \ldots, a_k) \in \mathcal{Q}$ faça $M(a_1, \ldots, a_k) \leftarrow 1$
para $a_1 \leftarrow 0$ até n_1 faça
    para $a_2 \leftarrow 0$ até n_2 faça
        $\ldots$
            para $a_k \leftarrow 0$ até n_k faça
                se $(a_1, \ldots, a_k) \notin \mathcal{Q}$ então
                    seja $(b_1, \ldots, b_k) \in \mathcal{Q}$ tal que $M(a_1 - b_1, \ldots, a_k - b_k)$ é mínimo.
                    $M(a_1, \ldots, a_k) \leftarrow 1 + M(a_1 - b_1, \ldots, a_k - b_k)$
se $M(n_1, \ldots, n_k) \leq m$ devolva SIM
senão devolva NÃO

Lema: *Dados instância (G, m, t') e tempo T, onde G tem no máximo k tempos de processamentos distintos em t', algoritmo DecisãoExato$_{\epsilon,k}$ decide se G pode ser escalonado em m máquinas com tempo máximo T em tempo polinomial.*

Prova. Como $n_i \leq n$, $i = 1, \ldots, k$, temos

- $|\mathcal{Q}| = O(n^k)$
- Passo 4 e 6: tempo $O(k\, n^k)$
- Passos 11–13: tempo $O(k\, n^k)$
- Passos 7–13: tempo $O(k\, n^{2k})$

Portanto o algoritmo tem complexidade de tempo $O(k\, n^{2k})$, que é polinomial quando k é constante. □

Inaproximabilidade

Problema de Otimização

- $\mathcal{I}$: Conjunto de instâncias.
- Sol(I): Conjunto de soluções para cada $I \in \mathcal{I}$.
- $\mathrm{val}(I, S)$: Valor da solução $S \in$ Sol(I).

Problema de Minimização

Encontrar $S \in$ Sol(I) tal que $\mathrm{val}(I, S)$ é mínimo.

Problema de Maximização

Encontrar $S \in$ Sol(I) tal que $\mathrm{val}(I, S)$ é máximo.

Solução Ótima

Solução mínima (máxima) para problema de minimização (maximização).

Classes de Aproximabilidade

NPO - Extensão dos problemas de NP a problemas de otimização

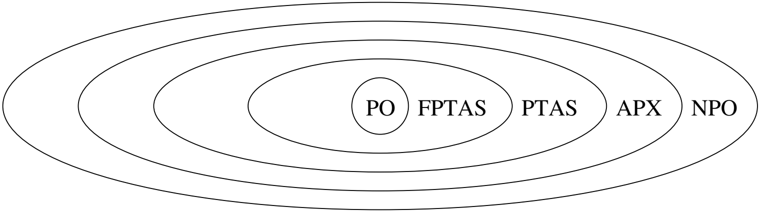
- $\exists$ função polinomial p tal que $\langle S \rangle \leq p(\langle I \rangle)$, $\forall I \in \mathcal{I}$, $\forall S \in$ Sol(I).
- Dado palavra X, $\exists$ algoritmo polinomial que decide se $X \in \mathcal{I}$.
- Dado objeto Y e $I \in \mathcal{I}$, $\exists$ algoritmo polinomial que decide se $Y \in$ Sol
- Dado $I \in \mathcal{I}$ e $S \in$ Sol(I), $\exists$ algoritmo polinomial que calcula $\mathrm{val}(I, S)$.

PO - Problemas de NPO para os quais existe algoritmo polinomial exato

APX - Problemas de NPO para os quais existe α-aproximação polinomial, para α constante.

Fato: $\mathit{PO} \subseteq \mathit{FPTAS} \subseteq \mathit{PTAS} \subseteq \mathit{APX} \subseteq \mathit{NPO}$

Possível configuração para estas classes:



Teorema: MOCHILA $\in$ *FPTAS.*

Teorema: ESCALONAMENTO $\in$ *PTAS.*

Teorema: *Os seguintes problemas de otimização em grafos planares são NP-difíceis e admitem PTAS (Baker'94): Conjunto independente, cobertura mínima, conjunto dominante, empacotamento de triângulos.*

Teorema: *Os problemas* EMPACOTAMENTO, MAXCUT, MAXSAT, MINCV, MINFS, TSPM *pertencem a APX.*

Teorema: MINCC, MINMCUT, MINTC, TSP *pertencem a NPO.*

NP-Completo no sentido forte

Max(I): maior número inteiro em valor absoluto que ocorre em $I \in \mathcal{I}$;
Max(I) $:= 0$ se nenhum número inteiro ocorre em I.

$\mathcal{I}_p := \{I \in \mathcal{I} : \text{Max}(I) \leq p(\langle I \rangle)\}$ para uma função polinomial p.

Π é NP*-completo no sentido forte* ou *fortemente* NP*-completo* se existe função polinomial p tal que Π_p é NP-completo.

Teorema: *O problema da Mochila não é fortemente NP-completo.*
Prova. Exercício. □

Teorema: *Os seguintes problemas em grafos planares são fortemente NP-completos: Conjunto independente, cobertura mínima, conjunto dominante, empacotamento de triângulos.*
Prova. Exercício. □

Teorema: *O problema* ESCALONAMENTO *é um problema fortemente NP-completo.*

Teorema: *(Garey & Johnson'78) Seja* $\Pi \in$*NPO fortemente* NP*-completo tal que* $\mathrm{val}(I, S)$ *é inteiro não-negativo* $\forall I \in \mathcal{I}$ *e* $\forall S \in \mathrm{Sol}(I)$. *Se* $p(n, m)$ *é uma função polinomial tal que*

$$\mathrm{OPT}(I) \leq p(\langle I \rangle, \mathrm{Max}(I)), \quad \forall I \in \mathcal{I}$$

e $\Pi \in$ FPTAS, *então* $\mathrm{P} = \mathrm{NP}$.

Prova. Seja Π problema de minimização (análogo para maximização). Seja A um FPTAS para Π.

Seja $\epsilon := \dfrac{1}{p(\langle I \rangle, \mathrm{Max}(I)) + 1}$. Em tempo polinomial temos

$$\begin{aligned} \mathrm{val}(I, A(\epsilon, I)) - \mathrm{OPT}(I) &\leq \epsilon\, \mathrm{OPT}(I) \\ &= \frac{\mathrm{OPT}(I)}{p(\langle I \rangle, \mathrm{Max}(I)) + 1} \\ &< 1\,, \end{aligned}$$

para toda instância I. I.e., $\mathrm{val}(I, A(\epsilon, I)) = \mathrm{OPT}(I)$. □

Este teorema é válido também para números racionais.

Teorema: *Se* PO = FPTAS, *então* P = NP.
Prova. Exercício. □

Teorema: *Se* FPTAS = PTAS, *então* P = NP.
Prova. Exercício. □

Teorema: *Se PTAS=APX, então P=NP.*
Prova. Exercício. □

Teorema: *Se APX=NPO, então P=NP.*
Prova. Exercício. □

Teorema: *Se* P = NP, *então* PO = NPO.
Prova. Veja Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela e Protasi'99. □

Completude para problemas de Otimização

Uma AP-*redução* de um problema de otimização Π a um problema de otimização Π' ($\Pi \leq_{\text{AP}} \Pi'$) é um terno (f, g, β) onde f e g são algoritmos e β é um racional positivo tais que:

(AP1) f recebe racional positivo δ e instância I de Π, e devolve uma instância $f(\delta, I)$ de Π';

(AP2) g recebe racional positivo δ, instância I de Π e um elemento S' em $\text{Sol}(f(\delta, I))$, e devolve $g(\delta, I, S')$ em $\text{Sol}(I)$;

(AP3) para todo número racional positivo δ, os algoritmos $f(\delta, \cdot)$ e $g(\delta, \cdot, \cdot)$ são polinomiais; e

(AP4) para toda instância I de Π, todo número racional positivo δ, e todo S' em $\text{Sol}(f(\delta, I))$, vale que se
$$(1 - \delta)\,\mathrm{OPT}(f(\delta, I)) \leq \mathrm{val}(f(\delta, I), S') \leq (1 + \delta)\,\mathrm{OPT}(f(\delta, I))\ ,$$
então
$$(1 - \beta\delta)\,\mathrm{OPT}(I) \leq \mathrm{val}(I, g(\delta, I, S')) \leq (1 + \beta\delta)\,\mathrm{OPT}(I)\ .$$

$$\begin{tikzcd}
\Pi & \text{Instância} I \arrow[r, "I' := f(\delta, I)"] & I' \arrow[d, "S' := A'(I')"] & \Pi' \\
(1 \pm \beta\delta)\text{-}\mathit{aproximação} & \text{Solução } S & S' \arrow[l, "S := g(\delta, I, S')"'] & (1 \pm \delta)\text{-}\mathit{aproximação}
\end{tikzcd}$$

Teorema: *Se* $\Pi_1 \leq_{\text{AP}} \Pi_2$ *e* $\Pi_2 \leq_{\text{AP}} \Pi_3$, *então* $\Pi_1 \leq_{\text{AP}} \Pi_3$.
Prova. Exercício. $\square$

Teorema: *Se* Π *está em* NPO, Π' *está em* APX *e* $\Pi \leq_{\text{AP}} \Pi'$, *então* Π *está em* APX.
Prova. Exercício. $\square$

Teorema: *Se $\Pi \in$ NPO, $\Pi' \in$ PTAS e $\Pi \leq_{\text{AP}} \Pi'$, então $\Pi \in$ PTAS.*

Prova. Seja A' um PTAS para Π' e (f, g, β) uma AP-redução de Π a Π'. Vamos fazer um esquema de aproximação A para Π.

$A(\epsilon, I)$

$\epsilon' \leftarrow \epsilon/\beta$

$I' \leftarrow f(\epsilon', I)$

$S' \leftarrow A'(\epsilon', I')$

$S \leftarrow g(\epsilon', I, S')$

devolva S

$$
\begin{tikzcd}
\Pi \quad \text{Instância } I \arrow[r, "I' := f(\epsilon', I)"] & I' \quad \Pi' \arrow[d, "S' := A'(I')"] \\
(1 \pm \epsilon)\text{-aproximação} \quad \text{Solução } S & S' \quad (1 \pm \epsilon')\text{-aproximação} \arrow[l, "S := g(\epsilon', I, S')"]
\end{tikzcd}
$$

□

Def.: *Um problema* Π *em* APX *é* APX*-completo se cada problema em* APX *pode ser* AP*-reduzido a* Π.

Teorema: *(Papadimitriou & Yannakakis'91 e Khanna, Motwani, Sudan & Vazirani'99) O problema* MAXSAT *é* APX*-completo.*

Def.: *Um problema* Π, *não necessariamente em* APX, *é* APX*-difícil se a existência de um esquema de aproximação polinomial para* Π *implica em* $\mathrm{P} = \mathrm{NP}$.

Teorema: *Os problemas* EMPACOTAMENTO, MAXCUT, MAXSAT, MINCC, MINCV, MINFS, MINMCUT, MINTC, TSPM *e* TSP *são* APX*-difíceis.*

Def.: *Um problema* Π *em* NPO *é* NPO*-completo se cada problema em* NPO *pode ser* AP*-reduzido a* Π.

Teorema: *(Orponen e Mannila'87) O problema* TSP *é* NPO*-completo.*

Limiares de aproximação

Def.: *O limiar de aproximação (approximation threshold) de um problema de minimização (maximização) é o maior (menor) limitante inferior (superior) de todos os α para os quais existe uma α-aproximação polinomial para o problema.*

Lema: *Se* $\mathrm{P} = \mathrm{NP}$ *então o limiar de aproximação de todo problema em* NPO *é 1.*

A tabela seguinte mostra alguns resultados sobre os limiares de aproximação para vários problemas, considerando P $\neq$ NP.

problema	limiar de aproximação
Mochila	$= 1$ (Ibarra & Kim'75) (o problema está em FPTAS)
Escalonamento	$= 1$ (Hochbaum & Shmoys'88) (o problema está em PTAS
Empacotamento	$\geq 3/2$ (Garey & Johnson'79)
MaxCut	$\leq 16/17$ (Håstad'97)
MaxSat	$\leq 7/8$ (Håstad'97)
MinCV	$\geq 7/6$ (Håstad'97)
TSPM	$\geq 131/130$ (Engebretsen & Karpinski'00)
MinCC $(E, \mathcal{S}, c)$	$> \epsilon \log \lvert E \rvert$, para alguma constante $\epsilon > 0$ (Raz & Safra'97)
MinTC $(E, \mathcal{S}, c)$	$> \epsilon \log \lvert E \rvert$, para alguma constante $\epsilon > 0$
	(equivalente ao MinCC (Ausiello, D'Atri & Protasi'80)
TSP (G, c)	$> f(\langle G, c \rangle)$, para toda função f computável
	em tempo polinomial (Sahni & Gonzalez'76)
Clique(V, E)	$< 1/\lvert V \rvert^{1-\epsilon}$ (Zuckerman'07) para todo $\epsilon > 0$

Provas Verificáveis Probabilisticamente

Classe NP

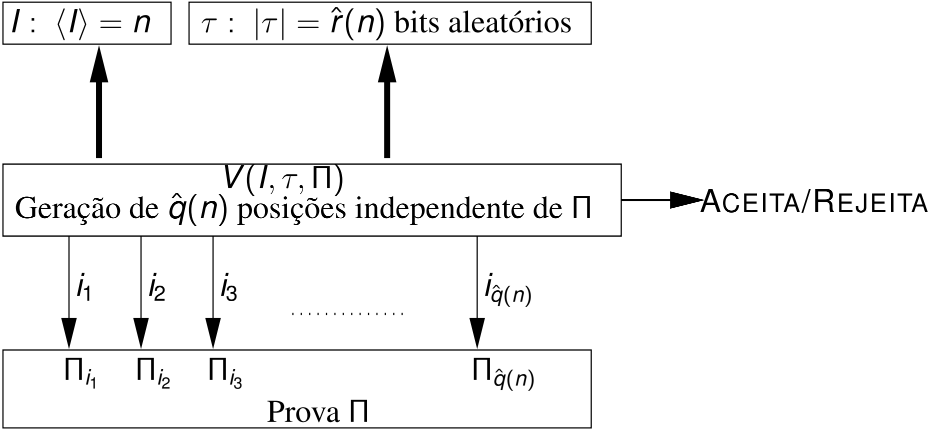
- Alfabeto $\Sigma = \{0, 1\}$
- Linguagem $L \subseteq \Sigma^*$
- $L \in$ NP $\Leftrightarrow \forall I \in L$ existe certificado "curto" C_I e algoritmo polinomial V que verifica que $I \in L$.

Sistemas de Provas PCP (Probabilistically Checkable Proofs)

Def.: *Dada instância I, $\langle I \rangle = n$, funções $r(n)$ e $q(n)$, e uma seqüência τ de bits aleatórios, dizemos que V é um verificador $(r(n), q(n))$-restrito se existem funções inteiras $\hat{r}(n) = O(r(n))$ e $\hat{q}(n) = O(q(n))$ tal que*

- *V, acessa I e $\hat{r}(n)$ primeiros bits de τ e*
- *determina $\hat{q}(n)$ posições de Π, $i_1, i_2, \ldots, i_{\hat{q}(n)}$*
- *acessa $\Pi_{i_1}, \Pi_{i_2}, \ldots, \Pi_{i_{\hat{q}(n)}}$ e responde* ACEITA *ou* REJEITA *deterministicamente.*

Verificador $(r(n), q(n))$-restrito



Def.: *Uma linguagem $L \in \Sigma^*$ está em $\mathrm{PCP}(r(n), q(n))$ se e só se existe um verificador $(r(n), q(n))$-restrito tal que*

- *Para todo $I \in L$, $\exists\ \Pi_I \in \Sigma^*$:*
  $\Pr_\tau(V(I, \tau, \Pi_I) = \text{ACEITA}) = 1$
- *Para todo $I \notin L$, $\forall\ \Pi \in \Sigma^*$*
  $\Pr_\tau(V(I, \tau, \Pi) = \text{ACEITA}) < \frac{1}{4}$

No lugar de $\frac{1}{4}$ poderíamos ter no lugar, qualquer valor constante β, onde $0 < \beta < 1$.

Nova caracterização de NP

Teorema: *(Arora, Lund, Motwani, Sudan & Szegedy'92)*
$\mathrm{PCP}(\log n, 1) = \mathrm{NP}$.

Inaproximabilidade do MAX3SAT

Problema MAX3SAT $(V, \mathcal{C})$ Dada uma coleção $\mathcal{C}$ de cláusulas sobre um conjunto V de variáveis, cada cláusula com exatamente 3 variáveis, encontrar uma valoração x de V que satisfaça o maior número possível de cláusulas de $\mathcal{C}$.

Teorema: *Se* P $\neq$ NP *então* MAX3SAT $\notin$ PTAS.

Prova.

Vamos mostrar que

$$\text{MAX3SAT} \in \text{PTAS} \Rightarrow \exists \text{ algoritmo polinomial para decidir } L,$$

para qualquer $L \in$ NP.

Seja $L \in \text{NP}$ e $I \in \Sigma^*$. Vamos considerar a pertinência $I \in L$.

Como $L \in \mathrm{PCP}(\log n, 1)$ existe verificador V, $(\log n, 1)$-restrito para L.

Dado I vamos mostrar como construir em tempo polinomial uma instância S_I para MAX3SAT tal que

$$\begin{array}{lcl} I \in L & \Rightarrow & S_I \text{ é satisfatível} \\ I \notin L & \Rightarrow & \text{no máximo } \frac{1}{4} \text{ das cláusulas de } S_I \text{ pode ser satisfeita} \\ & & \text{(a fração é independente de } I\text{)} \end{array}$$

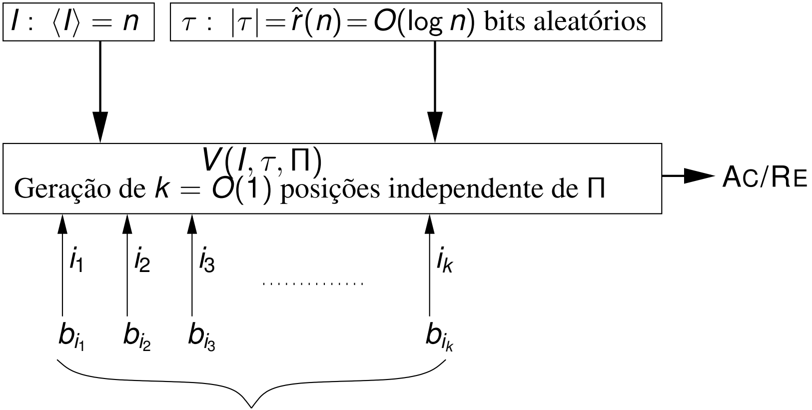
Dado sequência τ de bits aleatórios, onde $|\tau| = \hat{r}(n) = O(\log n)$, verificador V obtém endereços $i_1, i_2, \ldots, i_k$ em Π, onde k é constante.

V devolve ACEITA ou REJEITA considerando os k valores $\Pi_{i_1}, \Pi_{i_2}, \ldots, \Pi_{i_k}$ da prova Π.

Considere todas as atribuições de k bits para estes endereços para os quais V responde ACEITA.

Seja S_τ uma fórmula em 3-Sat que representa estas atribuições, S_τ com no máximo K cláusulas de 3 variáveis.

Verificador $(\log n, 1)$**-restrito**



S_τ:=Fórmula em 3-Sat para respostas ACEITO
de atribuições de $(b_{i_1}, \ldots, b_{i_k})$
\# cláusulas de $S_\tau \leq K = f(k) = O(1)$

Agora, considere todos os $m := 2^{\hat{r}(n)}$ valores possíveis para bits aleatórios, $\tau_1, \tau_2, \ldots, \tau_m$.

Considere a fórmula $S := S_{\tau_1} \bigwedge S_{\tau_2} \bigwedge \ldots \bigwedge S_{\tau_m}$

$$I \in L \quad \Rightarrow \quad \exists \Pi_I \text{ tal que } S_I \text{ é satisfatível}$$

$$I \notin L \quad \Rightarrow \quad \forall \Pi \text{ no máximo } \tfrac{1}{4} \text{ das fórmulas } S_{\tau_i}$$
$$\text{pode ser satisfeita simultaneamente}$$

Para S_{τ_i} não ser satisfeita, basta uma de suas cláusulas (de no máximo K) não ser satisfeita.

Vamos calcular uma fração máxima de cláusulas satisfeitas

$$S := \underbrace{S_{\tau_1} \bigwedge S_{\tau_2} \bigwedge \ldots \bigwedge S_{\tau_{m/4}}}_{\text{satisfeitas}} \bigwedge \underbrace{S_{\tau_{m/4+1}} \bigwedge \ldots \bigwedge S_{\tau_m}}_{\text{não satisfeitas}}$$

Assim, se $I \notin L$ no máximo $\dfrac{K \cdot \frac{m}{4} + (K-1) \cdot \frac{3m}{4}}{K \cdot m} = 1 - \dfrac{3}{4K}$ das cláusulas podem ser satisfeitas.

Assim,

- ou todos as cláusulas de S são satisfeitas,
- ou no máximo $\left(1 - \dfrac{3}{4K}\right)$ das cláusulas de S são satisfeitas.

Portanto, se tivermos uma α-aproximação para o MAX3SAT, com $\alpha > \left(1 - \frac{3}{4K}\right)$, podemos decidir a existência de uma prova Π para I.

$\square$

Teorema: *Se* P $\neq$ NP, *então* MAXSAT $\notin$ PTAS.

Inaproximabilidade do CLIQUE

Teorema: *(Zuckerman'07) Se existir uma $n^{1-\epsilon}$-aproximação polinomial para o problema* CLIQUE, *para qualquer* $\epsilon > 0$, *então* $\mathrm{P} = \mathrm{NP}$.

Este resultado usa o sistema PCP. Provaremos algo mais fraco.

Teorema: *Se existir uma α-aproximação polinomial para o problema* CLIQUE, *para qualquer constante $\alpha > 0$, então* $\mathrm{P} = \mathrm{NP}$.

Prova.
Vamos mostrar que
CLIQUE $\in$ APX $\Rightarrow \exists$ algoritmo polinomial para decidir L, onde $L \in$ NP.

Seja $L \in$ NP e $I \in \Sigma^*$. Vamos considerar a pertinência $I \in L$.

Como $L \in \mathrm{PCP}(\log n, 1)$ existe verificador V, $(\log n, 1)$-restrito para L.
Dado I vamos construir um grafo G_I para o problema CLIQUE tal que

$$\begin{array}{rcl} I \in L & \Rightarrow & \omega(G_I) = f(n) \\ I \notin L & \Rightarrow & \omega(G_I) < \frac{1}{4} f(n) \end{array}$$

A cada consulta de bits aleatórios τ, $|\tau| = \hat{r}(n) = O(\log n)$, V obtém k endereços $i_1, i_2, \ldots, i_k$

V consulta $\Pi_{i_1}, \Pi_{i_2}, \ldots, \Pi_{i_k}$ e devolve ACEITA ou REJEITA.

Seja $G_I = (V_I, E_I)$ grafo tal que

- V_I são todas as seqüências de $\hat{r}(n) + k$ bits $(\tau, b_{i_1}, b_{i_2}, \ldots, b_{i_k})$ tal que V consulta os endereços $i_1, i_2, \ldots, i_k$ e os valores $(b_{i_1}, b_{i_2}, \ldots, b_{i_k})$ para estas posições fazem V devolver ACEITO.
- Dados dois vértices v' e v'',

  $$v' = (\tau', b_{i'_1}, \ldots, b_{i'_k}) \quad \text{e} \quad v'' = (\tau'', b_{i''_1}, \ldots, b_{i''_k})$$

  $\{v', v''\} \in E_I$ se não há conflito no valor de dois bits de mesma posição.

Note que G_I pode ser construido em tempo polinomial.

Dado prova Π qualquer,

$$\begin{aligned} \omega(G_I) &\geq |\{\tau : V(I,\tau,\Pi) = \text{ACEITA}\}| \\ &= 2^{\hat{r}(n)} \Pr_\tau(V(I,\tau,\Pi) = \text{ACEITA}) \end{aligned}$$

Dado um clique C em G_I, $|C| = \omega(G_I)$, existe prova Π_C, consistente com todos os vértices de C

$$\begin{aligned} \omega(G_I) &\leq |\{\tau : V(I,\tau,\Pi_C) = \text{ACEITA}\}| \\ &= 2^{\hat{r}(n)} \Pr_\tau(V(I,\tau,\Pi_C) = \text{ACEITA}) \end{aligned}$$

Portanto

$$\omega(G_I) = 2^{\hat{r}(n)} \max_{\Pi} \Pr_\tau(V(I,\tau,\Pi) = \text{ACEITA})$$

Pela definição de sistemas PCP,

$$\max_{\Pi} \Pr_{\tau}(V(I, \tau, \Pi) = \text{ACEITA}) \begin{cases} = 1 & \text{se } I \in L \\ < \frac{1}{4} & \text{se } I \notin L \end{cases}$$

Assim,

- ou existe clique de tamanho $f(n)$,
- ou o clique máximo é menor que $\frac{1}{4}f(n)$,

onde $f(n) = 2^{\hat{r}(n)}$.

Portanto, se tivermos uma 4-aproximação para o CLIQUE podemos decidir a existência de uma prova Π para I.

Fortalecendo o verificador para probabilidade de $1/\alpha$, no lugar de $1/4$, provamos que não existe uma α-aproximação para o CLIQUE. $\square$

Técnica Métrica

Def.: *Dados grafo G e conjunto K de pares de vértices, um caminho de s a t é um K-caminho se $\{s, t\} \in K$.*

Def.: *Um conjunto M de arestas é um K-multicorte se não existe K-caminho no grafo $G - M$.*

Problema $\text{MINMCUT}(G, K, c)$ Dados grafo G, conjunto K de pares de vértices e custo $c_e \in \mathbb{Q}_{\geq}$ para cada $e \in E_G$, encontrar um K-multicorte M que minimize $c(M) = \sum_{e \in M} c_e$.

Teorema: *O problema* $\text{MINMCUT}(G, K, c)$ *é polinomial quando* $|K| = 1$ *ou* $|K| = 2$ *e* NP*-difícil quando* $|K| \geq 3$.

Seja $\mathcal{P}$ o conjunto de todos os K-caminhos. O seguinte programa linear é uma relaxação para MINMCUT.

Encontrar um vetor x indexado por E_G que

$$
(P) \quad
\begin{array}{rrcll}
\text{minimize} & \displaystyle\sum_{e \in E} c_e x_e & & & \\
& \displaystyle\sum_{e \in E_P} x_e & \geq & 1 & \text{para cada } P \text{ em } \mathcal{P}\,, \\
& x_e & \geq & 0 & \text{para cada } e \text{ em } E_G\,.
\end{array}
$$

Algoritmo de Garg, Vazirani e Yannakakis:

MINMCUT-GVY (G, K, c), $\quad K \neq \emptyset$.

1 seja $\hat{x}$ uma solução ótima racional de (P).

2 $k \leftarrow |K|$

3 $M \leftarrow$ CENTRAL $(G, k, K, c, \hat{x})$

4 devolva M

Apresentaremos o algoritmo CENTRAL e a prova do seguinte lema posteriormente.

Lema: *O algoritmo* CENTRAL *produz um K-multicorte M em tempo polinomial tal que* $\sum_{e \in M} c_e \leq (4 \ln 2k)\, c\, x$.

Teorema: *(Garg, Vazirani, Yannakakis'96) O algoritmo* MINMCUT-GVY *é uma* $(4 \ln 2k)$*-aproximação polinomial para o* MINMCUT (G, K, c)*, sendo* $k := |K| > 0$*.*

Prova.

$$c(M) = \sum_{e \in M} c_e \leq (4 \ln 2k)\, c\, \hat{x} \leq (4 \ln 2k)\, \mathrm{OPT}(G, K, c)\,.$$

A linha 1 de MINMCUT-GVY pode ser executada em tempo polinomial, pois temos algoritmo de separação para as desigualdades de (P). □

Algoritmo Central

O algoritmo CENTRAL separa pelo menos um par de $\{s, t\} \in K$ em cada chamada recursiva.

Para isto, o algoritmo encontra um corte (S, T) tal que

1. nenhum par em K está em S,
2. algum par em K tem extamente um vértice em S
3. $c(\delta(S))$ é razoavelmente pequeno.

Seja

$$x(s,u) := \min\{\sum_{e\in E_P} x_e : \ P \text{ é um caminho de } s \text{ a } u\}.$$

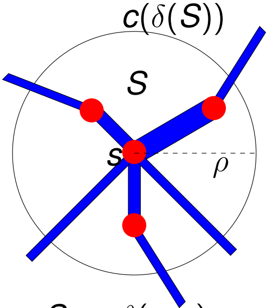
$$V(s,\rho) := \{v \in V_G : x(s,v) \le \rho\}\,.$$

Idéia: Imagine que as arestas do grafo são tubos, sendo x_e o comprimento e c_e a área da secção transversal do tubo e.

Denote por $\vartheta(s,\rho)$ o volume da parte da tubulação que dista no máximo ρ de s:

$$\vartheta(s,\rho) := c_A x_A + \sum_{uv\in\delta(S),\, u\in S} c_{uv}(\rho - x(s,u))\,,$$

onde $S := V(s,\rho)$ e c_A e x_A são as restrições de c e x, ao conjunto $A := E_{G[S]}$.

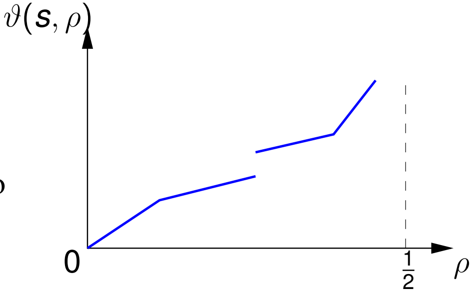


$S := \vartheta(s, \rho)$

c_e = seção transversal do tubo

x_e = comprimento do tubo

$c(\delta(S))$ = custo do corte



Descontinuidades podem ocorrer pela inclusão de toda uma aresta.

CENTRAL (G, k, K, c, x), $|K| \leq k$

se $K = \emptyset$
    então devolva $\emptyset$
    senão se $cx = 0$
        então devolva $\{e \in E_G : x_e > 0\}$
        senão sejam $\{s, t\} \in K$
            seja $v_1, \ldots, v_n$ tal que $x(s, v_1) \leq \cdots \leq x(s, v_n)$
            para i de 1 a n faça $p_i \leftarrow x(s, v_i)$
            $j \leftarrow 1 + \max\{i : p_i = 0\}$
            enquanto $\vartheta(s, p_j) > ((2k)^{2p_j} - 1)\frac{1}{k}cx$ faça $j \leftarrow j+1$
            $S \leftarrow V(s, p_{j-1})$
            $T \leftarrow V_G \setminus S$
            $B \leftarrow E_{G[T]}$
            $G_B \leftarrow (V_G, B)$
            $K_B \leftarrow K \setminus \{\{s', t'\} :$ S separa s' de $t'\}$
            $M_B \leftarrow$ CENTRAL (G_B, k, K_B, c_B, x_B)
            devolva $\delta(S) \cup M_B$

Lema: *Em uma chamada, temos* $p_{j-1} < \frac{1}{2}$*, e portanto* $x(s,u) < \frac{1}{2}$ *para todo* $u \in S$*.*

Prova.

Seja h o menor natural tal que $p_h \geq \frac{1}{2}$.

Temos que $2 \leq h \leq n$ já que $p_n \geq x(s,t) \geq 1$ e $p_1 = 0$.

Como $\vartheta(s, p_h) \leq cx$ e $k \geq 1$ temos

$$\begin{aligned} \vartheta(s, p_h) &\leq cx \\ &\leq \left((2k)^{2p_h} - 1\right)\frac{1}{k}cx \end{aligned}$$

assim, h não satisfaz condição da linha 9 e portanto $j \leq h$. □

Corolário: *Em uma chamada, para um par* $\{s,t\}$*, temos* $s \in S$ *e* $t \notin S$*. Além disso, se* $\{s_i, t_i\} \in K - \{s,t\}$*, então* $s_i \notin S$ *ou* $t_i \notin S$*.*

Prova. Exercício. □

Lema: *Ao fim da linha* 10 *do algoritmo* CENTRAL, *temos que*

$$c(\delta(S)) \leq (2\ln 2k)\left(c_A x_A + c_{\delta(S)} x_{\delta(S)} + \frac{1}{k} c x\right),$$

onde $A := E_{G[S]}$.

Prova. Note que após a linha 9, temos

$$\begin{aligned} p_{j-1} &< p_j\,, & (1)\\ \vartheta(s,p_j) &\leq ((2k)^{2p_j}-1)\frac{1}{k}cx\,. & (2)\\ \vartheta(s,p_{j-1}) &\geq ((2k)^{2p_{j-1}}-1)\frac{1}{k}cx \quad \text{e} & (3)\end{aligned}$$

De (3) e (2) temos que

$$\frac{\vartheta(s,p_j)+\frac{1}{k}cx}{\vartheta(s,p_{j-1})+\frac{1}{k}cx} \leq (2k)^{2(p_j-p_{j-1})}\,. \qquad (4)$$

Tomando-se o logaritmo natural do lado esquerdo de

$$\frac{\vartheta(s,p_j)+\frac{1}{k}cx}{\vartheta(s,p_{j-1})+\frac{1}{k}cx} \leq (2k)^{2(p_j-p_{j-1})}\,. \qquad (5)$$

obtemos

$$\begin{aligned}
\ln\left(\vartheta(s,p_j)+\tfrac{1}{k}cx\right)-\ln\left(\vartheta(s,p_{j-1})+\tfrac{1}{k}cx\right) &= \\
= \int_{p_{j-1}}^{p_j} \frac{d}{d\rho}\ln\left(\vartheta(s,\rho)+\tfrac{1}{k}cx\right)d\rho \\
= \int_{p_{j-1}}^{p_j} \frac{c(\delta(S))}{\vartheta(s,\rho)+\frac{1}{k}cx}\,d\rho\,,
\end{aligned}$$

(note que $\vartheta(s,\rho)$ é uma função linear com coeficiente $c(\delta(S))$).

Tomando-se o logaritmo natural do lado direito de

$$\frac{\vartheta(s,p_j)+\frac{1}{k}cx}{\vartheta(s,p_{j-1})+\frac{1}{k}cx} \leq (2k)^{2(p_j-p_{j-1})}. \tag{6}$$

obtemos

$$2(p_j-p_{j-1})\ln 2k = \int_{p_{j-1}}^{p_j} (2\ln 2k)\, d\rho\,.$$

Como o logaritmo é uma função crescente, concluímos de (6) que

$$\int_{p_{j-1}}^{p_j} \frac{c(\delta(S))}{\vartheta(s,\rho)+\frac{1}{k}cx}\, d\rho \leq \int_{p_{j-1}}^{p_j} (2\ln 2k)\, d\rho\,.$$

Então, para algum ρ no intervalo (p_{j-1},p_j) temos que

$$\frac{c(\delta(S))}{\vartheta(s,\rho)+\frac{1}{k}cx} \leq (2\ln 2k).$$

Assim,

$$\begin{aligned} c(\delta(S)) &\leq (2\ln 2k)(\vartheta(s,\rho) + \frac{1}{k}cx) \\ &\leq (2\ln 2k)(c_A x_A + \sum_{uv\in\delta(S),\ u\in S} c_{uv}(\rho - x(s,u)) + \frac{1}{k}cx) \\ &\leq (2\ln 2k)(c_A x_A + \sum_{uv\in\delta(S)} c_{uv}x_{uv} + \frac{1}{k}cx) \\ &= (2\ln 2k)(c_A x_A + c_{\delta(S)}x_{\delta(S)} + \frac{1}{k}cx) \end{aligned}$$

□

Teorema: *O algoritmo* CENTRAL (G, k, K, c, x) *produz um* K*-multicorte* M *em tempo polinomial tal que*

$$c(M) \leq (2 \ln 2k)(1 + \tfrac{1}{k}|K|)cx. \tag{7}$$

Prova. Por indução em $|K|$.

Se $K = \emptyset$ ou $cx = 0$, claramente (7) vale.
Suponha que $K \neq \emptyset$ e $cx > 0$.
Neste caso, o algoritmo devolve $M := \delta(S) \cup M_B$. Assim,

$$\begin{aligned} c(\delta(S) \cup M_B) &= c(\delta(S)) + c_B(M_B) \\ &\leq (2 \ln 2k)\Big(c_A x_A + c_{\delta(S)} x_{\delta(S)} + \frac{1}{k} cx + (1 + \frac{1}{k}|K_B|) c_B x_B\Big) \\ &= (2 \ln 2k)\Big(cx + \frac{1}{k} cx + \frac{1}{k}|K_B| c_B x_B\Big) \\ &\leq (2 \ln 2k)\Big(cx + \frac{1}{k} cx + \frac{1}{k}(|K| - 1) cx\Big) \\ &\leq (2 \ln 2k)(1 + \frac{1}{k}|K|) cx\,. \end{aligned}$$

Equilíbrio de Nash e Busca Local

- Internet: Rede gigantesca com grande quantidade de usuários e complexa estrutura sócio-econômica
- Usuários podem ser competitivos, cooperativos,...
- Situações envolvendo Teoria dos Jogos e Computação

Ref.: Cap. 12 - Local Search do livro *Algorithm Design* de Kleinberg e Tardos

Um jogo Multicast

- Jogadores podem construir links entre nós
- Há um nó origem
- Cada jogador representa um nó destino
- Cada jogador quer conectar o nó origem até seu nó destino
- Há cooperação na construção da rede. Isto é, o custo de um link é dividido igualmente entre os usuários que o utilizam

Definição

Dados

- Grafo direcionado $G = (V, E)$
- Custo positivo c_e para cada aresta e.
- Vértice fonte s
- k vértices destinos $t_1, \ldots, t_k$

Cada usuário i procura encontrar

- caminho orientado P_i do vértice s até t_i pagando menos

Custo para

- usuário i é $c(P_i) = \sum_{e \in P_i} \frac{c_e}{k_e}$, onde k_e número de caminhos usando e
- sistema é $c(P_1, \ldots, P_k) = \sum_i c(P_i)$ (custo social)

Jogo

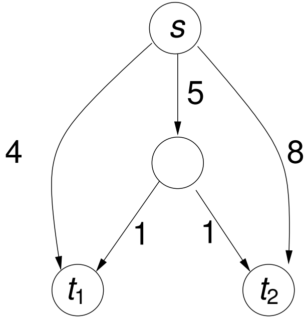
Regras do Jogo:

- Cada usuário fica estável ou muda sua rota (pagando menos) baseado apenas na configuração atual
- Em um estado do jogo com caminhos $(P_1, \ldots, P_k)$, denotamos por $E^+ \subseteq E$ as arestas usadas em pelo menos um caminho.
- O custo social é o custo dos caminhos escolhidos pelos jogadores:

$$c(P_1, \ldots, P_k) = \sum_{i=1}^{k} c(P_i) = \sum_{e \in E^+} c_e$$

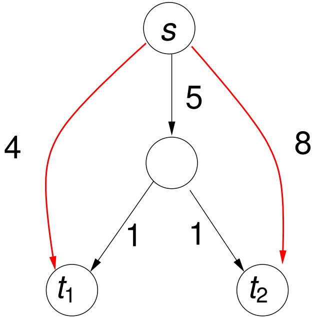
Exemplo 1

- Temos dois jogadores: 1 e 2
- Cada um tem duas alternativas: uma rota externa e uma interna.



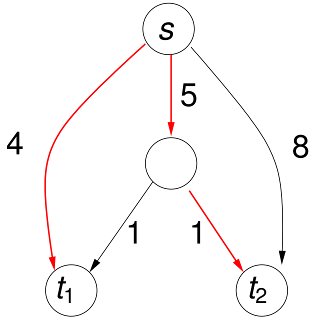
Exemplo 1

- Considere que inicialmente os jogadores usam as rotas externas.
- O jogador 1 paga 4 e o jogador 2 paga 8
- O custo social é igual a 12.



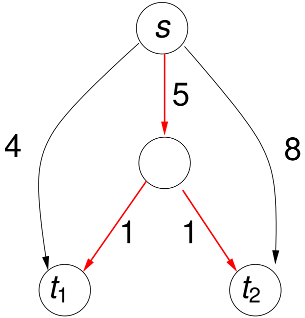
Exemplo 1

- O jogador 2 muda para a rota interna e seu custo cai para 6
- O custo social cai para 10



Exemplo 1

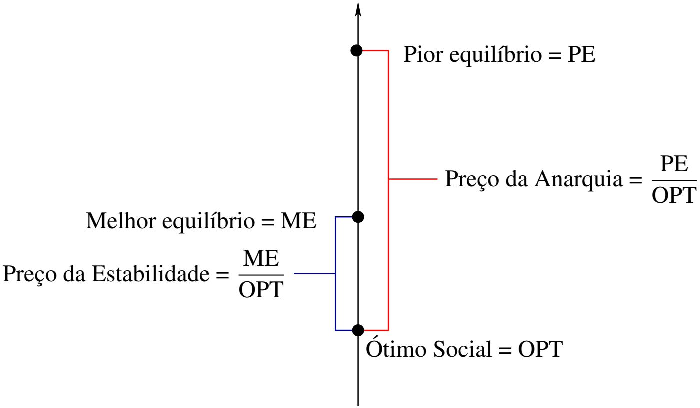
- O jogador 1 tem incentivo a mudar
- Cada jogador paga $2{,}5 + 1$, e estamos em um equilíbrio
- O custo social cai para 7 (solução final também é ótima)



Definições e Notação

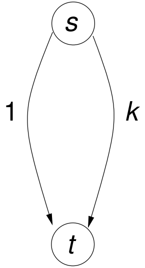
- A Estratégia do jogador i é o conjunto de rotas de s para t_i
- O estado do jogo em um momento é dado pelos k caminhos $(P_1, \ldots, P_k)$ no momento
- O ótimo social é o menor valor possível de uma solução (dos k caminhos), possivelmente não está em equilíbrio.
- Um usuário i está insatisfeito no estado atual, se ele pode mudar sua rota por outra de custo melhor
- Estado em Equilíbrio de Nash quando não há usuários insatisfeitos
- O Preço da Estabilidade razão entre a melhor solução em equilíbrio com o ótimo social
- O Preço da Anarquia razão entre a pior solução em equilíbrio com o ótimo social
- Melhor resposta: movimento para estratégia de maior ganho positivo

Preços da anarquia e da estabilidade para minimização



Exemplo 2

- Na rede abaixo há k jogadores todos com mesmo destino t
- Considere todos usando a aresta da direita
- Estamos em um equilíbrio com custo k (cada jogador paga 1).
- O ótimo social tem custo 1 (cada jogador paga $\frac{1}{k}$).



k Jogadores

Teorema: *O preço da anarquia deste jogo Multicast é k.*

Método da Função Potencial e o Preço da Estabilidade

Def.: *Uma* função potencial exata Φ *é uma função que*

- *mapeia cada vetor de estratégia* $\mathcal{P}$ *para um valor real tal que*
- *se* $\mathcal{P} = (P_1, \ldots, P_i, \ldots, P_k)$ *e*

  $P_i' \neq P_i$ *é uma estratégia alternativa para o jogador* i*, então*

$$\Phi(\mathcal{P}) - \Phi(\mathcal{P}') = c_i(P_i) - c_i(P_i'),$$

onde $\mathcal{P}' = (P_1, \ldots, P_i', \ldots, P_k)$

Fato: *Seja* Φ *uma função potencial exata para o jogo do Multicast com dinâmica de melhor resposta. Se jogador* i *muda sua estratégia de* P_i *para* P_i'*, e o vetor de estratégia muda de* $\mathcal{P}$ *para* $\mathcal{P}'$*, então*

$$\Phi(\mathcal{P}) > \Phi(\mathcal{P}').$$

Isto é, Φ *é estritamente decrescente após jogadas.*

Função Potencial para Multicast

Dado vetor de estratégias $\mathcal{P} = (P_1, \ldots, P_k)$, denote por

$$\Phi(\mathcal{P}) = \sum_e c_e \cdot H(k_e),$$

onde

$$H(t) = 1 + \frac{1}{2} + \cdots + \frac{1}{t} \quad \text{e} \quad H(0) = 0$$

k_e é o número de caminhos de $\mathcal{P}$ que usam e

Lema: Φ *é uma função potencial exata.*

Prova. Exercício □

Fato: $\Phi(\mathcal{P})$ *é limitado inferiormente.*

Prova. Exercício □

Lema: *O jogo Multicast com a dinâmica de melhor resposta converge para um equilíbrio de Nash.*

Prova. Exercício □

Preço da Estabilidade

Lema: *Se $\mathcal{P} = (P_1, \ldots, P_k)$ é um vetor de estratégia, então*

$$c(\mathcal{P}) \leq \Phi(\mathcal{P}) \leq H(k)c(\mathcal{P}).$$

Teorema: *O preço da estabilidade do jogo Multicast é no máximo $H(k)$.*

Prova. Seja:

OPT um vetor de estratégia ótimo (ótimo social)

$\mathcal{O}$ um vetor de estratégia em equilíbrio obtido a partir de OPT

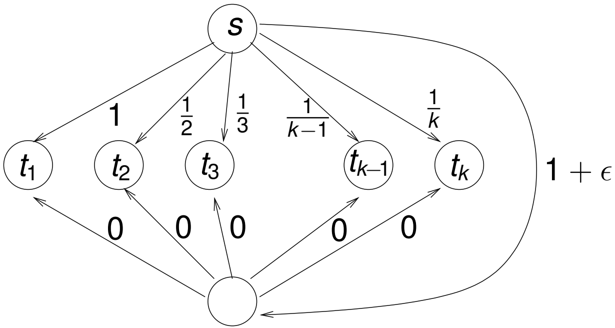
$\mathcal{P}$ um vetor de estratégia em Equilíbrio de Nash de menor custo

$$\begin{aligned} c(\mathcal{P}) &\leq c(\mathcal{O}) \\ &\leq \Phi(\mathcal{O}) \\ &\leq \Phi(\mathrm{OPT}) \\ &\leq H(k) \cdot c(\mathrm{OPT}) \end{aligned}$$

□

Lema: *O preço da estabilidade $H(k)$ do problema de Multicast é justo (melhor possível).*

Prova.



□

Observações

- Tempo de convergência do problema Multicast pode ser exponencial
- Encontrar ótimo social é um problema NP-difícil.

Exercício: Mostre que encontrar o ótimo social do problema Multicast é um problema NP-difícil. Sugestão: por Cobertura por Conjuntos.

Complexidade de se encontrar Equilíbrio de Nash em Jogos Potenciais

O quão difícil é encontrar um algoritmo polinomial para encontrar um equilíbrio de Nash ?

Classe PLS - *Polynomial Local Search Problems*

Em um problema de otimização (minimização) temos:

- conjunto de instâncias I
- para entrada $x \in I$, temos um conjunto de soluções viáveis $F(x)$
- para toda solução $s \in F(x)$ temos um custo $c_x(s)$
- um oráculo que diz se s pertence ou não à $F(x)$ e em caso positivo, computa $c_x(s)$

O problema consiste em dado $x \in I$, encontrar $s \in F(x)$ tal que $c_x(s)$ é mínimo.

A versão de maximização é análoga.

Um *problema de otimização local* é um problema de otimização onde

- há uma vizinhança $N_x(s) \subset F(x)$ para cada $x \in I$ e $s \in F(x)$
- e uma solução s de $F(x)$ é um mínimo local se $c_x(s) \leq c_x(s')$ para todo $s' \in N_x(s)$.

O objetivo é encontrar uma solução que é mínimo local.

Um problema de otimização local pertence à *PLS* se temos um oráculo que, para qualquer instância $x \in I$ e solução $s \in F(x)$, decide se s é ótimo local, e se não for, devolve $s' \in N_x(s)$ com $c_x(s') < c_x(s)$.

Def.: *(Johnson, Papadimitriou, Yannakakis'88) Um problema P é PLS-completo se está em PLS e se para todo problema Q de PLS, há uma redução polinomial de Q para P, tal que qualquer ótimo local de P corresponde a um ótimo local de Q.*

Há vários problemas em PLS-completo (Circuit-SAT com pesos, busca de ótimos locais relativos ao TSP, MAXCUT, SAT, etc)

Problema da Satisfatibilidade com Pesos (LSAT - Local SAT):

- Seja ϕ uma fórmula em Forma Normal Conjuntiva (FNC) $C_1 \wedge \ldots \wedge C_m$
- cada cláusula C_j com peso w_j.
- Uma atribuição lógica qualquer das variáveis é uma solução de ϕ.
- O valor de uma solução s é o peso total das cláusulas satisfeitas por s.
- A vizinhança de s são as atribuições obtidas trocando o valor de apenas uma variável de s.
- O objetivo é encontrar uma solução que é mínimo local.

Teorema: *(Krentel'89) O problema LSAT é um problema PLS-completo.*

Teorema: *(Fabrikant, Papadimitriou, Talwar'04) O problema de se encontrar um equilíbrio puro de Nash em jogos potenciais, onde a melhor resposta é computada em tempo polinomial, é PLS-completo.*