
Programação Inteira

Flávio Keidi Miyazawa

© 2003

IC - UNICAMP

Tópicos

- **Complexidade Computacional**
- **Programação Linear**
- **Programação Linear Inteira**
 - Modelagem de Problemas
 - Método Branch & Bound
 - Método por Planos de Cortes
 - Método Branch & Cut

Problemas de Otimização

Def.: *Um problema de otimização consiste em:*

Dado conjunto S , e função $f : S \rightarrow \mathbb{R}$, encontrar $x \in S$ tal que $f(x)$ é mínimo.

I.e.,

Encontrar x que

minimize $f(x)$

sujeito a $x \in S$

Nosso principal interesse será para o caso onde S é um domínio finito e enumerável.

Problemas de Otimização Combinatória

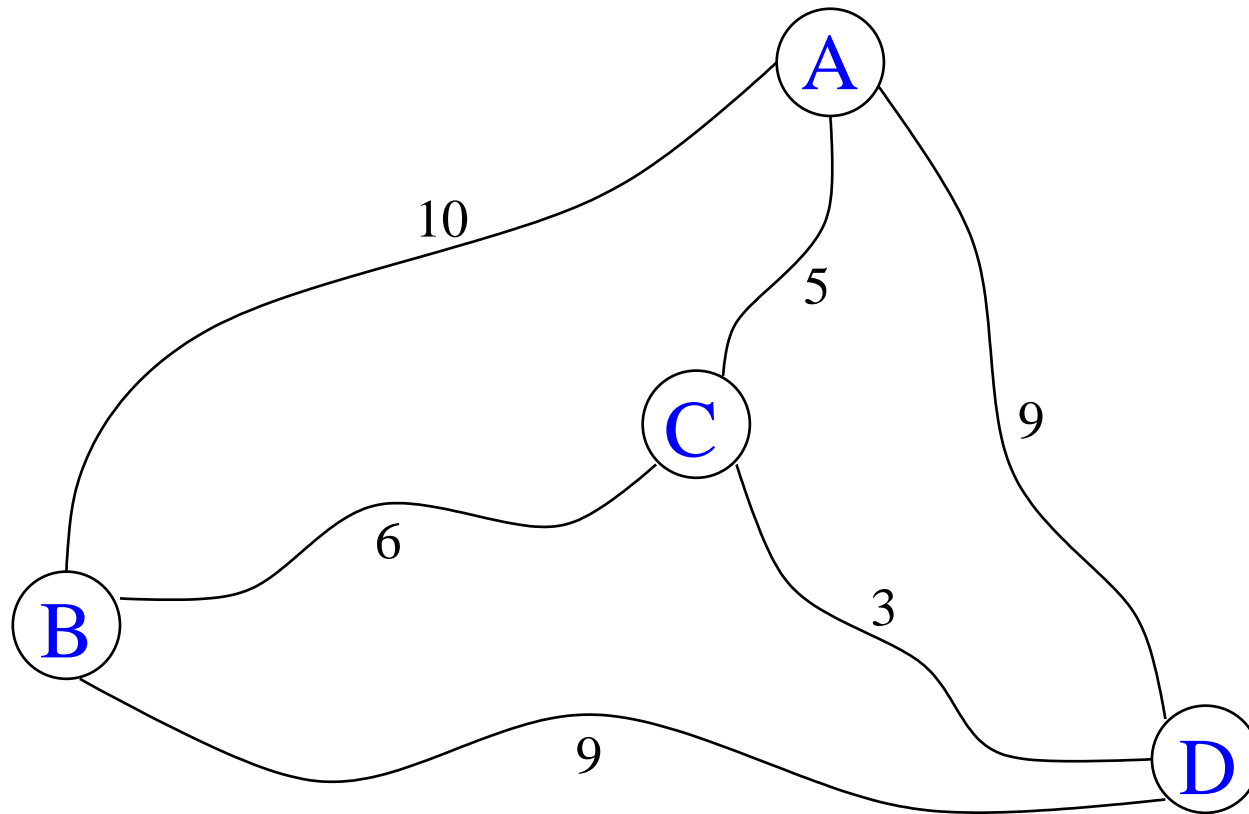
- Domínio Finito (enumerável)
- Função de Otimização: Custos, Comprimentos, Quantidades
- Objetivo: Minimização ou Maximização

Exemplo: Problema do Caixeiro Viajante (TSP)

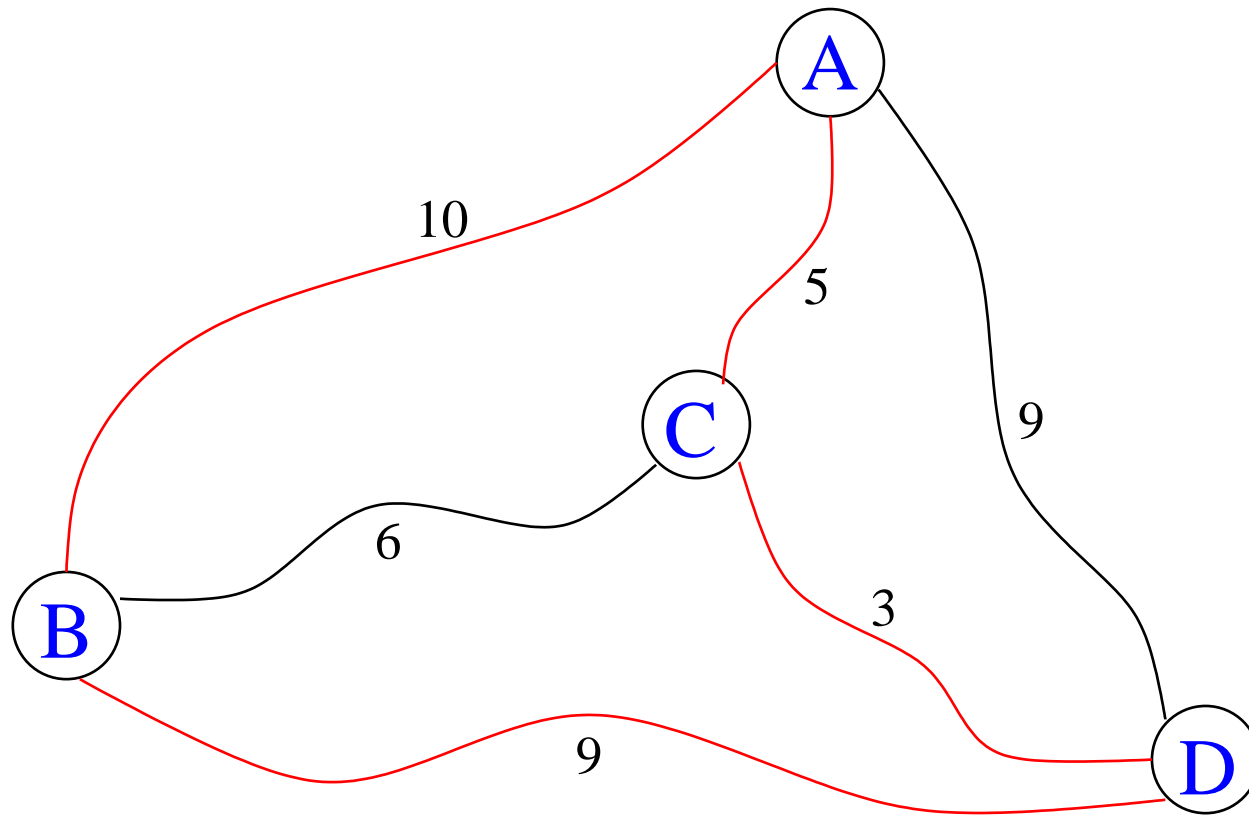
- *Entrada:*
 - Grafo não orientado: $G = (V, E)$,
 - Custos nas arestas: $c_e \geq 0, \forall e \in E$.
- *Objetivo:*

Encontrar um *tour* (circuito hamiltoniano) de custo mínimo que visita cada vértice exatamente uma vez.

Encontre o circuito hamiltoniano de custo mínimo



Circuito 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 rápidos para o TSP

Outros exemplos difíceis:

- Atribuição de Frequências 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

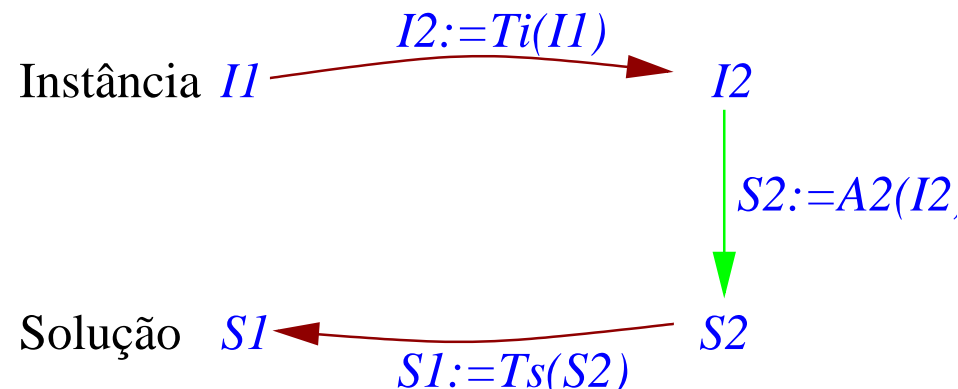
Simplificação dos problemas e sua formalização

- **Exemplo:** Dado grafo $G = (V, E)$, existe um circuito hamiltoniano em G ?
- **Exemplo:** Dado grafo completo $G = (V, E)$, $c : E \rightarrow \mathbb{Z}^+$ e um inteiro positivo K , existe um circuito 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 $G = (V, E)$, $c : E \rightarrow \mathbb{Z}^+$, 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 ($P_1 \leftarrow P_2$ ou $P_1 \preceq P_2$) se

- $\exists T_i$ que transforma instância I_1 de P_1 para instância I_2 de P_2
- $\exists T_s$ que transforma solução S_2 de I_2 para solução S_1 de I_1
- T_i e T_s 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 ($P_1 \leftarrow P_2$ ou $P_1 \preceq P_2$) 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”.

Exemplo de reduções entre problemas

- Redução do problema de encontrar o máximo de um conjunto para o problema de ordenação do conjunto.

Basta ordenar o conjunto, com os maiores primeiro e retornar o primeiro elemento

- Redução do problema de colorir um grafo com K cores para o problema de colorir com $K + 1$ cores.

Dado um grafo G (para o problema das K -cores), construímos um grafo G' inserindo um novo vértice em G e todas as arestas ligando o vértice novo com os antigos. G é K colorível se e somente se G' é $K + 1$ colorível

- Redução do problema de ordenação de um conjunto para o problema de encontrar um caminho hamiltoniano em grafo orientado.

Dado um conjunto C (a ordenar), construir um grafo orientado G cujo conjunto de vértices é C e existe aresta $u \rightarrow v$ se $u \leq v$. Um caminho hamiltoniano em G nos dá uma ordenação de C .

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.

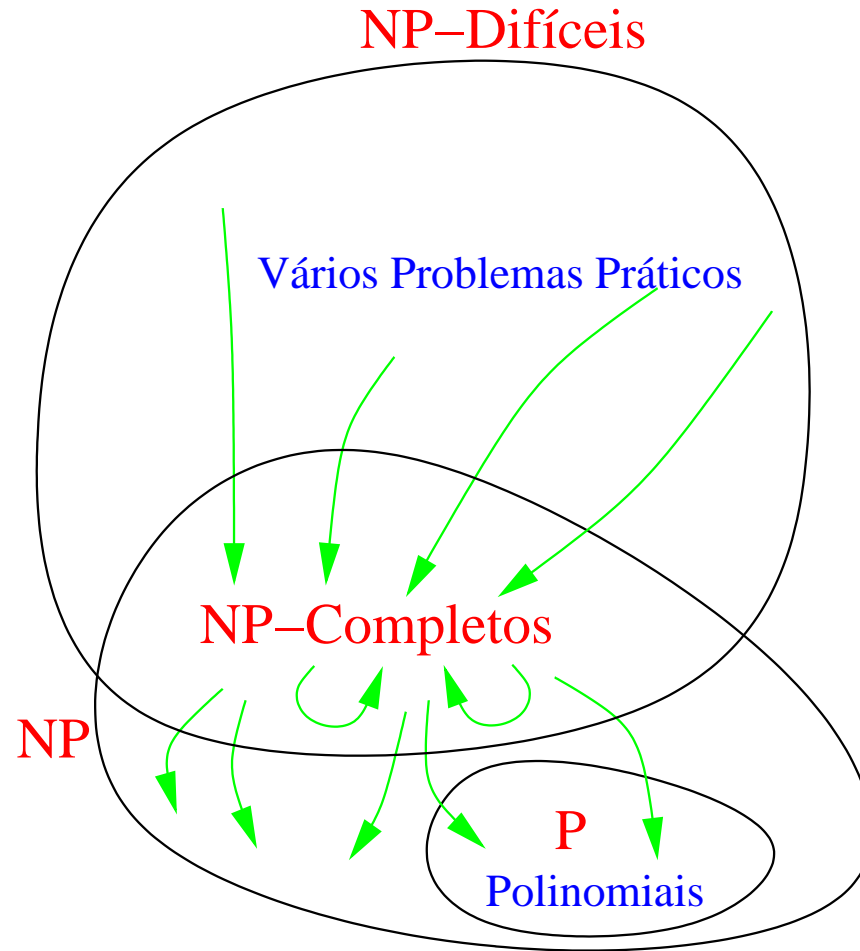
$X \in NP$ -Completo:

$X \in NP$ e $\forall Y \in NP, Y \preceq X$.

$X \in NP$ -Difícil:

$\forall Y \in NP, Y \preceq X$, onde X não necessariamente pertence a NP.

Uma provável configuração das classes



Conjectura (Edmonds'65): $P \neq NP$?

Prêmio de 1 milhão de US\$.

Problemas em **P**

- Dado um conjunto de números S , existe $k \in S$ tal que $k = \sum_{i \in S \setminus \{k\}} i$?
- Dado um grafo $G = (V, E)$, G é conexo ?
- Dado grafo completo $G = (V, E)$, $c : E \rightarrow \mathbb{Z}^+$, 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 ?

O primeiro problema NP-Completo

Def.: *Uma fórmula booleana está na forma normal conjuntiva (FNC) se é uma conjunção (conectivos \wedge) de cláusulas, cada cláusula contendo apenas conectivos \vee entre literais.*

Exemplo: $f(x, y, z) = (\bar{x} \vee y \vee z) \wedge (\bar{y} \vee x) \wedge (x \vee z)$

Problema SAT: Dado uma fórmula booleana F na forma FNC, decidir se F tem uma atribuição para suas variáveis que a torne verdadeira.

Exemplo:

$f(x, y, z) = (\bar{x} \vee y \vee z) \wedge (\bar{y} \vee x) \wedge (x \vee z)$ é satisfeita para $x=0$, $y=0$ e $z=1$.

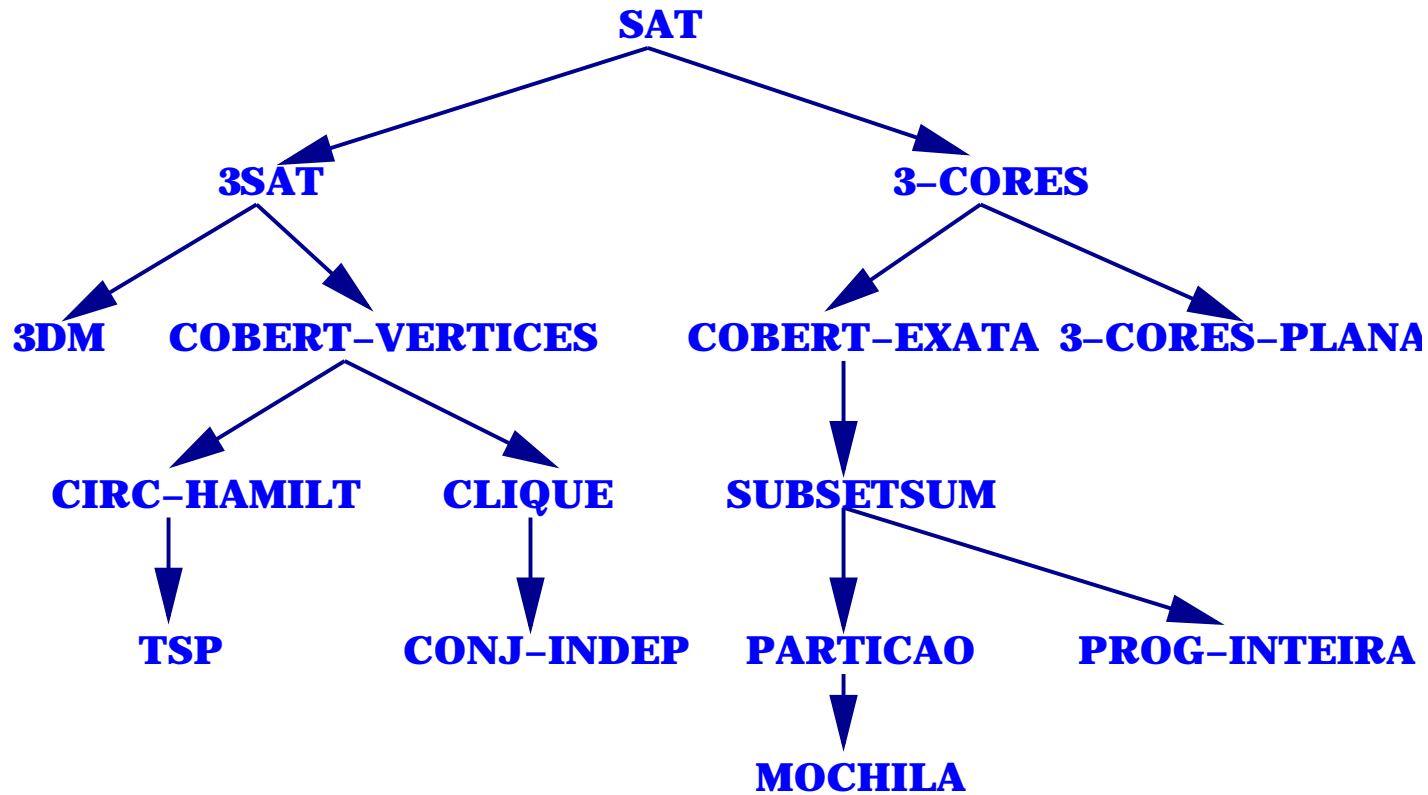
Verificação: Podemos verificar em tempo polinomial se uma atribuição satisfaz uma fórmula: **SAT \in NP**

Teorema: (Cook'71) O problema SAT é NP-completo.

Problemas em NP-Completo

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

Algumas reduções de problemas NP-completos



As primeiras reduções foram apresentadas por Karp [Kar72].

Problemas NP-difíceis

- Vários problemas práticos são NP-difíceis

Exemplos:

- Problema do Caixeiro Viajante
 - Atribuição de Frequências 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**
 - Para uma lista grande de problemas NP-difíceis, veja Crescenzi e Kann [CK00].

Comparando tempos polinomiais e exponenciais

$f(n)$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
n	.00001seg	.00002seg	.00003seg	.00004seg	.00005seg	.00006seg
n^2	.0001seg	.0004seg	.0009seg	.0016seg	.0025seg	.0036seg
n^3	.001seg	.008seg	.027seg	.064seg	.125seg	.216seg
n^5	.1seg	3.2seg	24.3seg	1.7min	5.2min	13.0min
2^n	.001seg	1.0seg	17.9min	12.7dias	35.7anos	366sec.
3^n	.059seg	58min	6.5anos	3855sec	2×10^8 sec	1.3×10^{13} sec

Quando uma instrução é efetuada em 0.000001 segundos.

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

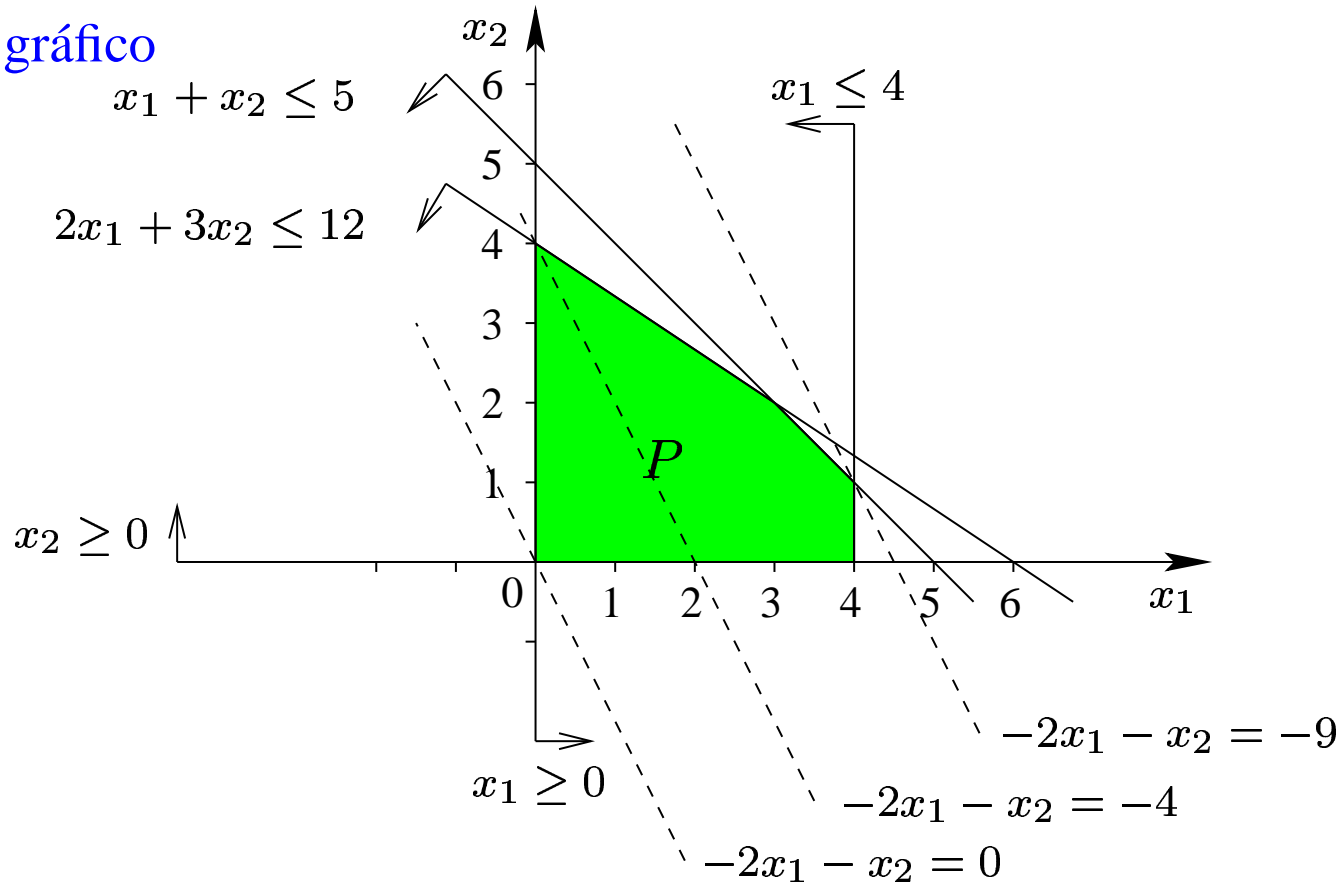
Para entender mais sobre a teoria de NP-completude, veja [Pap94, GJ79, CLR90].

Programação Linear

Programação linear

- dá a resolução exata para muitos problemas
- dá a resolução aproximada para muitos problemas
- faz parte dos principais métodos para obter soluções ótimas
- faz parte dos principais métodos para obter soluções aproximadas
- dá excelentes delimitantes para soluções ótimas
- pode ser executada muito rapidamente
- há diversos programas livres e comerciais

Exemplo gráfico



minimize $-2x_1 - x_2$
 sujeito a $\begin{cases} x_1 + x_2 \leq 5 \\ 2x_1 + 3x_2 \leq 12 \\ x_1 \leq 4 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$

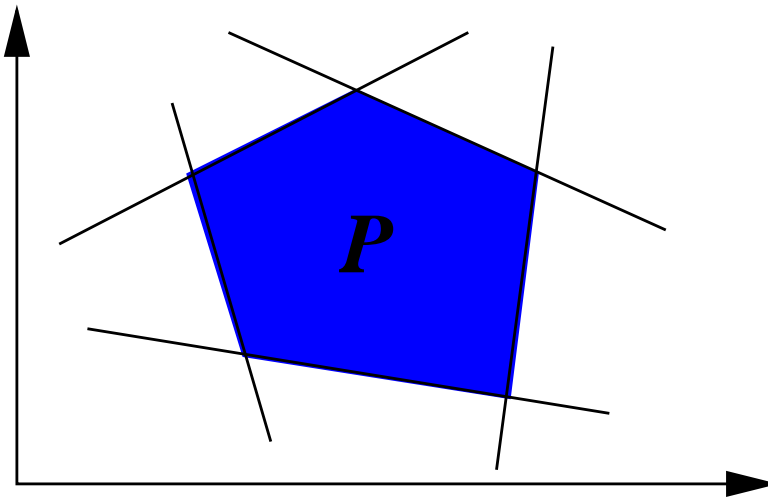
Solução ótima: $x_1 = 4$ e $x_2 = 1$

Valor da solução ótima = -9

Def.: Chamamos o conjunto de pontos P ,

$$P := \left\{ x \in \mathbb{Q}^n : \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_n \leq b_1 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_n \geq b_n \end{array} \right\}$$

como sendo um **poliedro**.



Def.: Se $y_i \in P$ e $\alpha_i \in \mathbb{Q}$, $i = 1, \dots, n$ dizemos que

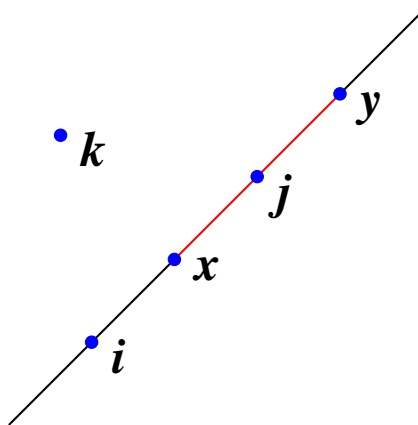
$y = \alpha_1 y_1 + \alpha_2 y_2 + \dots + \alpha_n y_n$ é uma **combinação convexa** dos pontos y_i 's se $\alpha_i \geq 0$ e $\alpha_1 + \dots + \alpha_n = 1$

Exemplo: Os pontos que são combinação convexa de dois pontos x e y são:

$$\text{conv}(\{x, y\}) := \{\alpha_1 x + \alpha_2 y : \alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_1 + \alpha_2 = 1\}$$

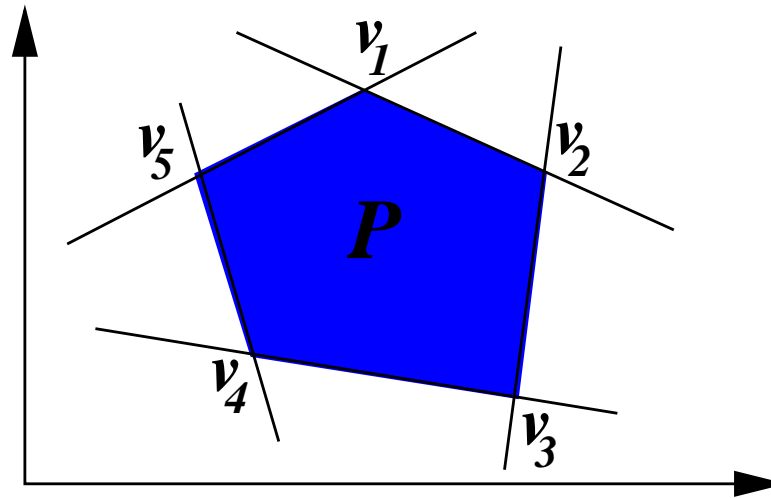
substituindo $\alpha_2 = 1 - \alpha_1$, temos

$$\text{conv}(\{x, y\}) := \{\alpha x + (1 - \alpha)y : 0 \leq \alpha \leq 1\}$$



j é combinação convexa de x e y , mas i e k não são

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



Vértices de P : v_1, v_2, v_3, v_4, v_5

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

Algoritmos polinomiais para resolver PL:

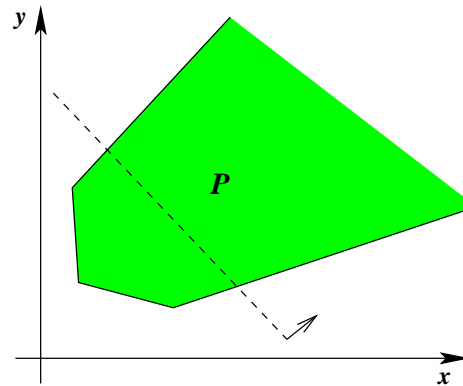
- Algoritmo dos elipsóides (Kachian'79) e
- Método dos pontos interiores (Karmarkar'84).

Algoritmos exponenciais para resolver PL:

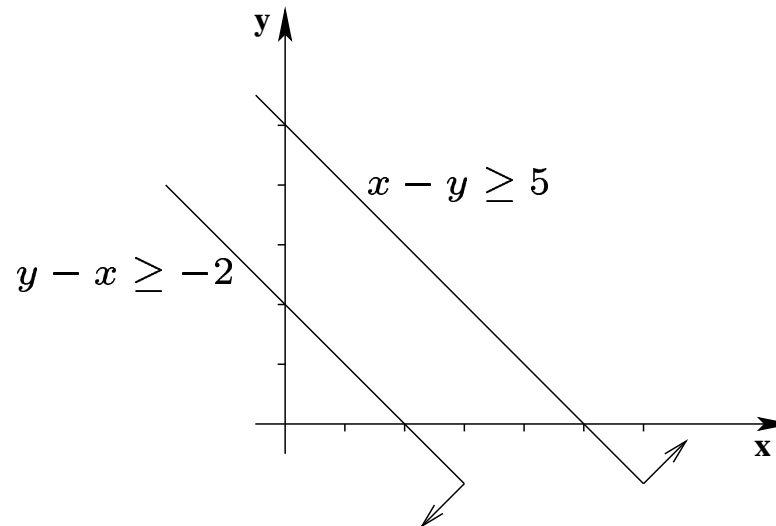
- Método simplex (Dantzig'47) (tempo médio polinomial).

Nem sempre encontramos uma solução ótima

- Sistema ilimitado



- Sistema inviável



Resolvedores de Sistemas Lineares

Programas comerciais

- CPLEX / ILOG: www.ilog.com/products/cplex/
- XPRESS: www.dashoptimization.com/
- OSL / IBM: www.research.ibm.com/osl

Programas livres

- GLPK / Gnu: www.gnu.org/software/glpk/glpk.html
- SoPlex / Zib: www.zib.de/Optimization/Software/Soplex

Fluxo de Custo Mínimo

Considere um sistema de produção de bens onde há centros consumidores e produtores.

- Todos os bens produzidos devem ser todos consumidos.
- Os bens produzidos chegam até os consumidores através de rotas.
- Cada rota tem uma capacidade máxima de escoamento (direcionada).
- Cada rota tem seu custo para transportar cada unidade do bem.
- **Objetivo:** Transportar os bens dos produtores para os consumidores minimizando custo para transportar os bens.

Outras aplicações:

- Transferência de dados em rede de computadores, detecção de “gargalos” da rede na transferência.
- Projeto de vias de tráfego no planejamento urbano.
- Detecção da capacidade de transmissão entre pontos de redes de telecomunicações.

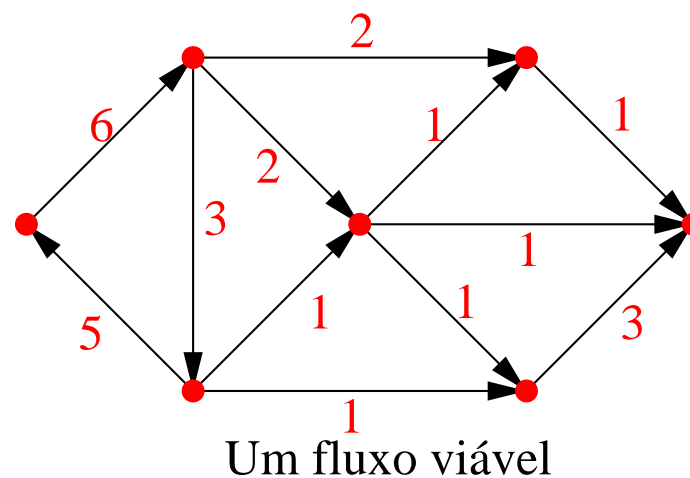
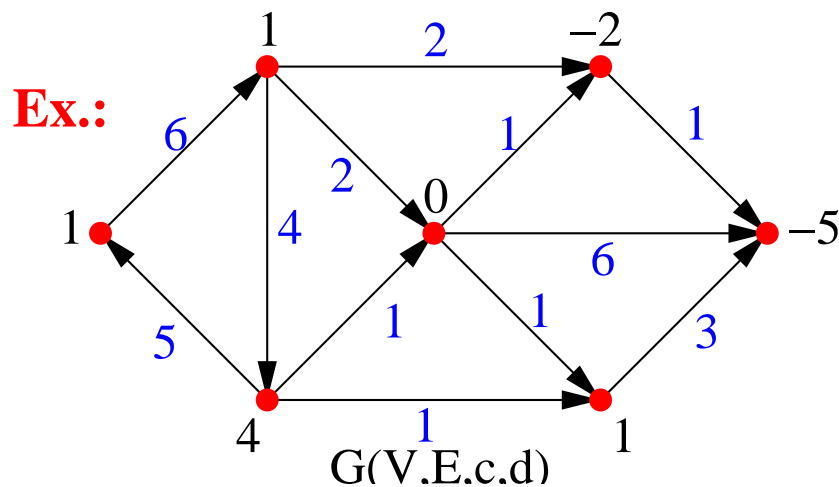
Definições:

Seja $G = (V, E)$ um grafo orientado com função de capacidades nas arestas $c : E \rightarrow \mathbb{Q}^+$, demandas $b : V \rightarrow \mathbb{Q}$ e custos nas arestas $w : E \rightarrow \mathbb{Q}^+$.

Def.: Dado um vértice $i \in V$, se $b_i < 0$ dizemos que i é um *consumidor* e se $b_i > 0$ dizemos que i é um *produtor*.

Def.: Dizemos que $x : E \rightarrow \mathbb{Q}^+$ é um *fluxo* em G se

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b_i \quad e \quad 0 \leq x_e \leq c_e$$



Def.: Dado um fluxo $x : E \rightarrow \mathbb{Q}^+$ (respeitando c e b) definimos o **custo do fluxo** x como sendo $\sum_{e \in E} w_e x_e$.

Problema FLUXO DE CUSTO MÍNIMO: Dados um grafo orientado $G = (V, E)$, capacidades $c : E \rightarrow \mathbb{Q}^+$, demandas $b : V \rightarrow \mathbb{Q}^+$ e uma função de custo nas arestas $w : E \rightarrow \mathbb{Q}^+$, encontrar um fluxo $x : E \rightarrow \mathbb{Q}^+$ de custo mínimo.

Formulação do Fluxo de custo mínimo

Encontrar x tal que

$$\min \sum_{e \in E} w_e x_e$$

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b_v$$

$$0 \leq x_e \leq c_e$$

Teorema: Se as capacidades nas arestas e as demandas dos vértices são inteiros (i.e., $c : E \rightarrow \mathbb{Z}^+$ e $b : V \rightarrow \mathbb{Z}^+$) então os vértices do poliedro do fluxo são inteiros.

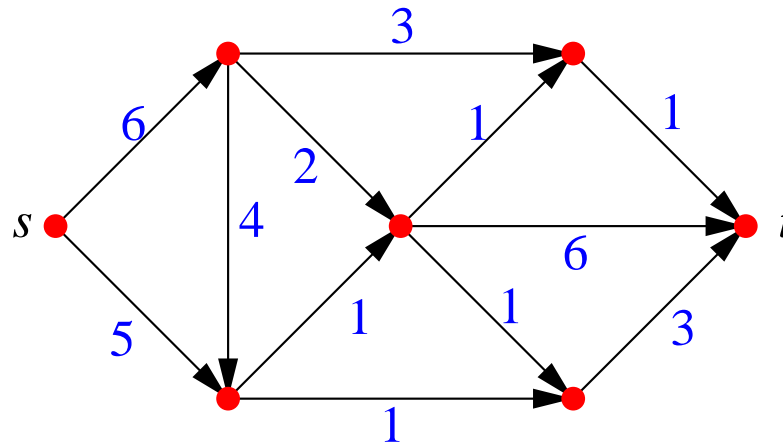
Subproblemas

- Fluxo máximo de um vértice s a um vértice t (st -fluxo)
- Problema do corte de capacidade mínima
- Problema do caminho mínimo (com pesos não negativos)
- Emparelhamento de peso máximo em grafos bipartidos

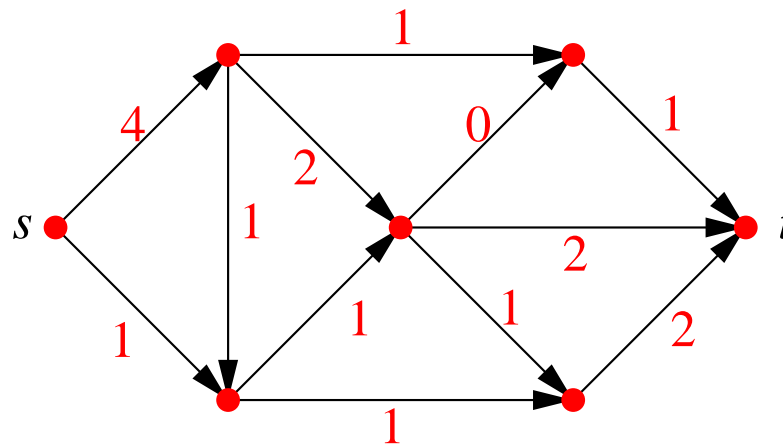
Fluxo Máximo de s para t (st -fluxo máximo)

- Não temos custo para transportar os bens.
- Cada rota tem uma capacidade máxima de escoamento (direcionada).
- Só temos um produtor (vértice s) de produção arbitrariamente grande.
- Só temos um consumidor (vértice t) de consumo arbitrariamente grande.
- Nós internos apenas repassam bens.
- **Objetivo:** Maximizar o transporte de bens de s para t , respeitando restrições de capacidade do fluxo.

Exemplo:



$G(V,E)$ e capacidades



fluxo de valor 5

Seja $G = (V, E)$ um grafo orientado, capacidades $c : E \rightarrow \mathbb{Q}^+$ e vértices s e t .

Def.: Dizemos que $x : E \rightarrow \mathbb{Q}^+$ é um *st-fluxo* se

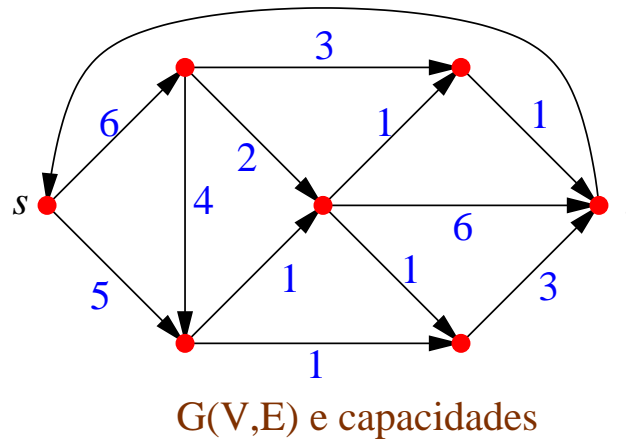
$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = 0 \quad \forall v \in V \setminus \{s, t\}$$
$$0 \leq x_e \leq c_e \quad \forall e \in E$$

Def.: Dado um *st-fluxo* x para $G = (V, E, c, s, t)$, definimos o valor do fluxo x como sendo $x(\delta^+(s)) - x(\delta^-(s))$.

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

Formulação Linear

- Para colocar no formato do problema de fluxo de custo mínimo, adicione uma aresta $t \rightarrow s$.



$$\begin{array}{l}
 \text{maximize} \quad x_{ts} \\
 \text{sujeito a} \quad \left\{ \begin{array}{l}
 \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = 0 \quad \forall v \in V \\
 0 \leq x_e \leq c_e \quad \forall e \in E.
 \end{array} \right.
 \end{array}$$

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

st-Corte Mínimo

Seja $G = (V, E)$ um grafo orientado com capacidades $c : E \rightarrow \mathbb{Q}^+$ e vértices s e t .

Def.: Um *st*-corte é um conjunto $S \subseteq V$ tal que $s \in S$ e $t \in \bar{S}$ (onde $\bar{S} := V \setminus S$). Também é denotado pelo conjunto de arestas $\delta(S)$.

Def.: Definimos a *capacidade de um st-corte* S , como sendo a soma

$$c(\delta(S)) := \sum_{e \in \delta(S)} c_e.$$

Problema *st*-CORTE MÍNIMO: Dado um grafo orientado $G = (V, E)$, capacidades $c : E \rightarrow \mathbb{Q}^+$ e vértices s e t , encontrar um *st*-corte de capacidade mínima.

Aplicações: Projeto de redes de conectividade, Classificação de dados (data mining), particionamento de circuitos VLSI, etc

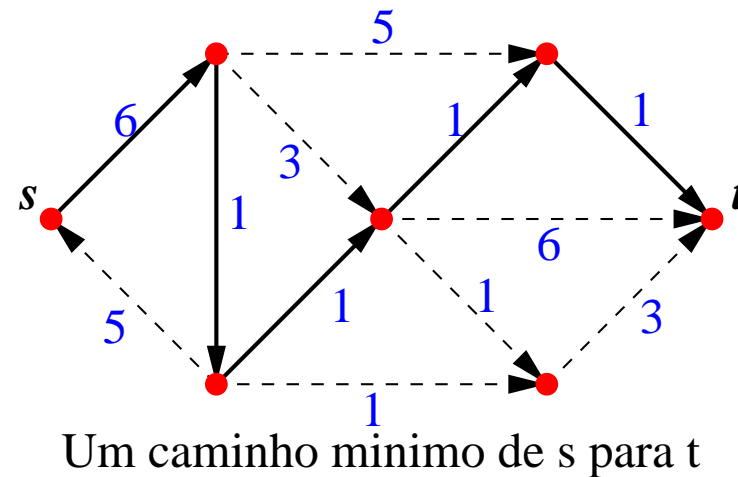
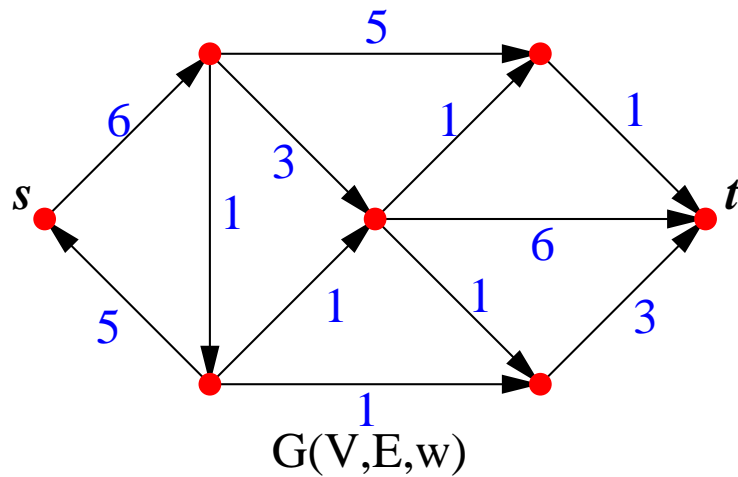
Exercícios:

- Faça um programa que, dado um st -fluxo de valor máximo, encontra um st -corte mínimo em tempo linear.
- Dado um grafo *não orientado*, com capacidades nas arestas, indicando a capacidade de fluxo que pode ir de um extremo ao outro de uma aresta, em qualquer uma das direções. O problema do fluxo máximo de um vértice s a um vértice t é encontrar um fluxo que sai de s e chega em t , respeitando a capacidade nas arestas e a lei de conservação do fluxo em cada vértice interno. Reduza este problema ao problema de encontrar um st -fluxo máximo em grafo orientado.
- Dado um grafo não orientado com capacidades nas arestas, e dois vértices s e t , resolva o problema de encontrar um corte de capacidade mínima separando s e t .

Um excelente livro sobre problemas em fluxo é dado em Ahuja et al [AMO93].

Problema do Caminho Mínimo

Problema CAMINHO MÍNIMO: Dado um grafo orientado $G = (V, E)$, custos nas arestas $w : E \rightarrow \mathbb{Q}^+$ e vértices s e t , encontrar um caminho de s para t de custo mínimo.



Aplicações: Determinação de rotas de custo mínimo, segmentação de imagens, Escolha de centro distribuidor, reconhecimento de fala, etc

Exemplo de aplicação em segmentação de imagens



Formulação para o Problema do Caminho Mínimo

Considere a seguinte formulação:

$$(P) \quad \begin{array}{l} \text{minimize} \\ \text{sujeito a} \end{array} \quad \left\{ \begin{array}{l} \sum_{e \in E} w_e x_e \\ \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = 0 \quad \forall v \in V \setminus \{s, t\} \\ \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e = 1 \\ \sum_{e \in \delta^+(t)} x_e - \sum_{e \in \delta^-(t)} x_e = -1 \\ 0 \leq x_e \leq 1 \quad \forall e \in E. \end{array} \right.$$

- Formulação é caso particular do problema do fluxo de custo mínimo

Corolário: *Se x é um ponto extremal ótimo de (P) , então x é inteiro. I.e., $x_e \in \{0, 1\} \forall e \in E$.*

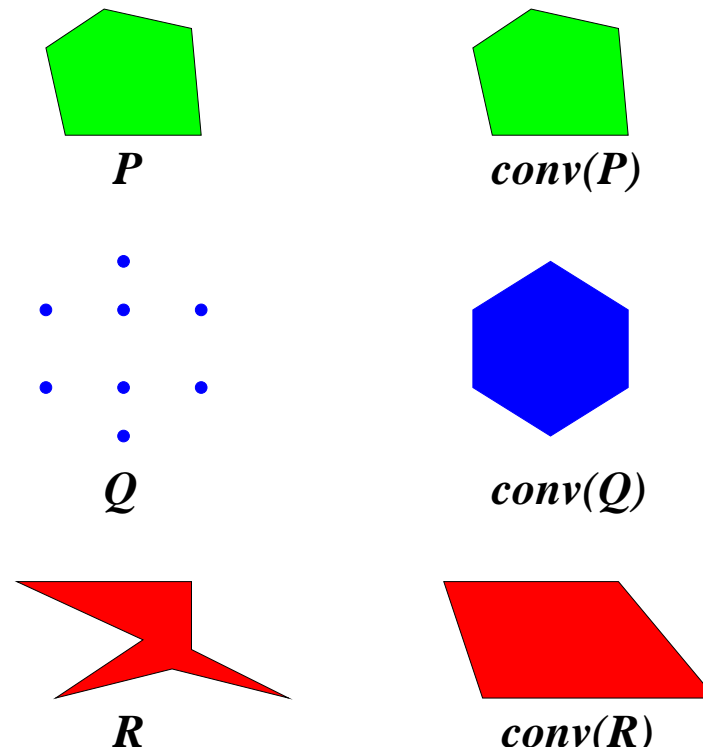
Teorema: *Se x é um ponto extremal ótimo de (P) , então as arestas $e \in E$ onde $x_e = 1$ formam um caminho mínimo ligando s a t .*

Definições

Def.: Dado um conjunto de pontos S , definimos o *fecho convexo*, denotado por $conv(S)$ o conjunto dos pontos formados pela combinação convexa dos pontos de S .

Exemplo: Dado dois pontos $x, y \in \mathbb{R}^n$, qualquer ponto que esteja no segmento de reta que liga x a y está em $conv(\{x, y\})$.

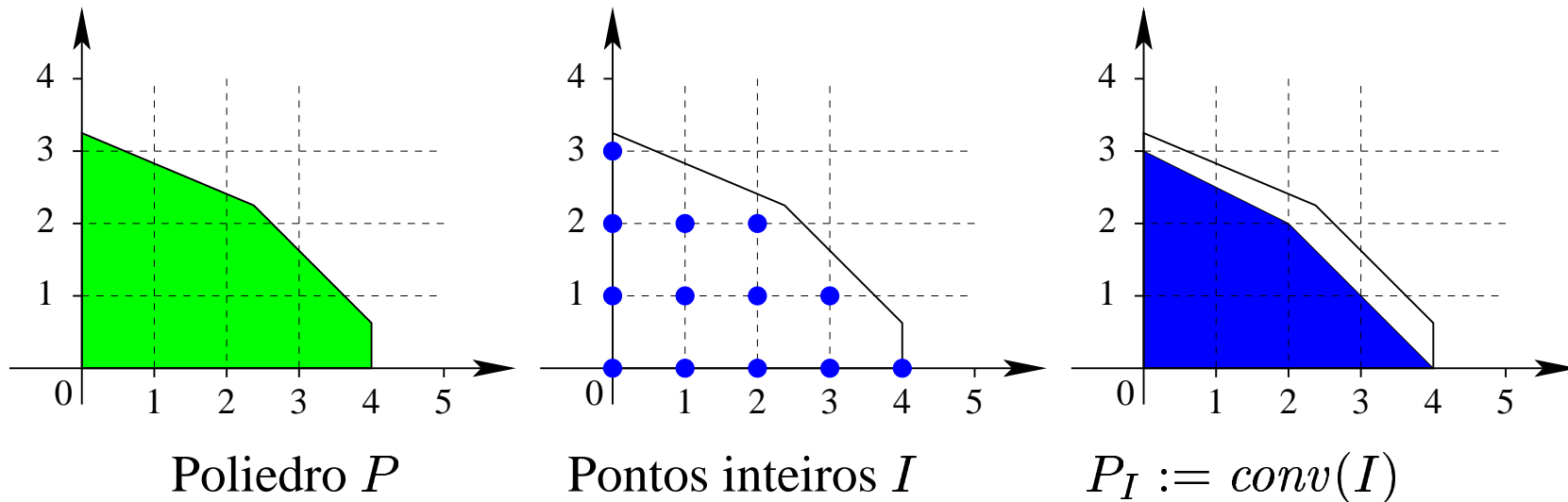
Exemplo:



Def.: Dado um poliedro P , definimos seu *fecho inteiro*, P_I , como sendo o fecho convexo dos vetores inteiros de P . I.e.,

$$P_I := \text{conv}\{x \in P : x \text{ é inteiro}\}$$

Exemplo: Exemplo de poliedro P , conjunto I dos pontos inteiros em P e o fecho convexo de I .

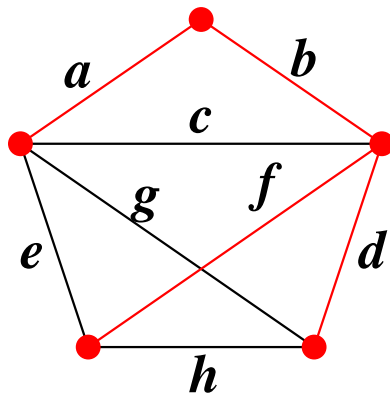


Def.: Dado um conjunto finito e ordenado E , dizemos que $\chi^A := (\chi_e^A : e \in E)$ é o **vetor de incidência** de A , $A \subseteq E$; onde $\chi_e^A = 1$ se $e \in A$ e 0 caso contrário.

Exemplo: Se $E = (a, b, c, d, e, f, g)$ e $A = \{a, c, e, f\}$ e χ^A é um vetor de incidência de A em E

$$\chi^A = (1, 0, 1, 0, 1, 1, 0).$$

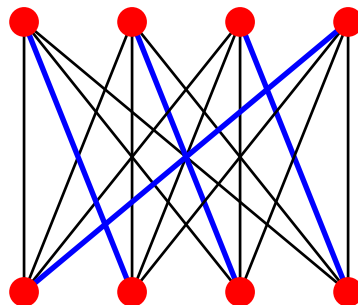
Exemplo: Seja $G = (V, E)$ o grafo abaixo com ordenação das arestas $E = (a, b, c, d, e, f, g, h)$ e T o subgrafo definido pelas arestas em vermelho.



Então o vetor de incidência das arestas de T em E é $\chi^T = (1, 1, 0, 1, 0, 1, 0)$.

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 $c : E \rightarrow \mathbb{Z}$, encontrar emparelhamento $M \subseteq E$ que maximize $c(M)$.

Aplicações: Encontrar atribuição de candidatos para vagas maximizando aptidão total, atribuição de motoristas de veículos, formação de equipes (eg. com um chefe e subordinados), etc.

Formulação para Emparelhamento em Grafos Bipartidos

Formulação em Programação Inteira

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

A formulação nos dá pontos no espaço \mathbb{R}^E , cada ponto um emparelhamento.

O poliedro dos emparelhamentos bipartidos de um grafo $G = (V, E)$ é definido como

$$P_{Emp}(G) := \text{conv}\{\chi^M \in \mathbb{R}^E : M \text{ é um emparelhamento em } G\},$$

onde χ^M é o vetor de incidência do emparelhamento M .

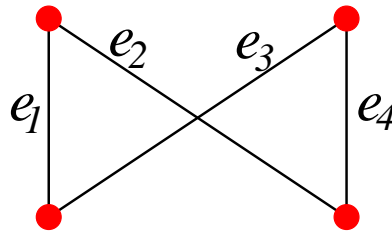
Relaxação Linear

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

Teorema: $LP_{Emp} = P_{Emp}$.

I.e., os vértices do poliedro relaxado do emparelhamento bipartido são inteiros.

Exemplo: Considere o seguinte grafo bipartido com custos unitários:



Programa Linear

$$\text{maximize } x_{e_1} + x_{e_2} + x_{e_3} + x_{e_4}$$

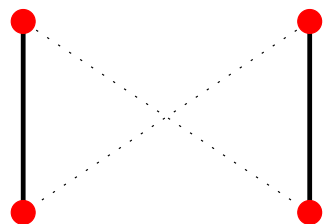
$$\text{sujeito a } \left\{ \begin{array}{l} x_{e_1} + x_{e_2} \leq 1, \\ x_{e_3} + x_{e_4} \leq 1, \\ x_{e_1} + x_{e_3} \leq 1, \\ x_{e_2} + x_{e_4} \leq 1, \\ 0 \leq x_{e_1} \leq 1, \\ 0 \leq x_{e_2} \leq 1, \\ 0 \leq x_{e_3} \leq 1, \\ 0 \leq x_{e_4} \leq 1 \end{array} \right.$$

Temos 4 variáveis binárias no programa linear correspondente:

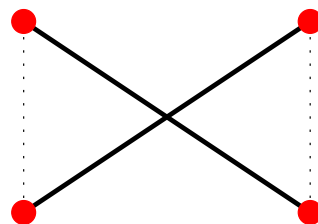
$$X = (x_{e_1}, x_{e_2}, x_{e_3}, x_{e_4})$$

Os seguintes vetores são soluções ótimas do programa linear:

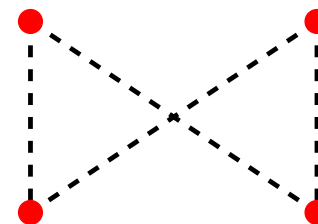
$$X' = (1, 0, 0, 1) \quad X'' = (0, 1, 1, 0) \quad X''' = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$$



X'



X''



X'''

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

Programação Linear Inteira

Os problemas resolvidos anteriormente por programação linear

- puderam ser resolvidos de maneira eficiente (tempo polinomial)

É possível usar programação linear na resolução de problemas NP-difíceis ?

SIM!!!

- Vários problemas tem sido bem resolvidos através de métodos em PLI.

Problema PLI: Dados matriz $A = (a_{ij}) \in \mathbb{Q}^{n \times n}$, 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

$$\begin{array}{l} \text{minimize} \\ \text{sujeito a} \end{array} \quad \begin{cases} c_1x_1 + c_2x_2 + \cdots + c_mx_m \\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_n \geq b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_n = b_n \\ x_i \in \mathbb{Z} \end{cases}$$

Qual a diferença com programação linear ?

Sobre a complexidade computacional, as diferenças são grandes.

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

Prova. Exercício: Reduza um dos problemas NP-difíceis vistos anteriormente para a forma de um problema PLI. □

Estratégias para resolver problemas NP-difíceis através de PLI:

- por arredondamento
 - modelar o problema como problema de programação linear inteira
 - relaxar a formulação para programação linear
 - resolver o programa linear
 - arredondar as variáveis para cima/baixo, de acordo com o problema.
- pelo método branch & bound
 - modelar o problema como problema de programação linear inteira
 - relaxar a formulação para programação linear
 - enumerar o espaço de soluções através de uma árvore de ramificação: Cada nó da árvore representa um programa linear. Cada programa linear (nó) é ramificado enquanto houver valores fracionários em sua solução.

- pelo método de planos de corte
 - modelar o problema como problema de programação linear inteira
 - relaxar a formulação para programação linear
 - repetidamente inserir uma desigualdade válida que separa o ponto fracionário.
 - parar quando obtiver uma solução inteira
- pelo método branch & cut
 - combinação do método branch & bound e
 - método de planos de corte

O ponto de partida de todas estas estratégias é modelar como um problema de programação linear inteira

Modelagem de Problemas

Modelagem através de variáveis 0/1

- Um dos passos mais importantes para se resolver um problema por programação linear inteira é a escolha da formulação.

Como no problema de emparelhamento, vamos formular alguns problemas NP-difíceis através de variáveis 0/1.

Em geral, a idéia principal é

- definir variáveis 0/1, digamos $x_i, i = 1, \dots, n$, tal que
- se $x_i = 1$ então o objeto i pertence à solução
- se $x_i = 0$ então o objeto i não pertence à solução

Formulação do Problema da Mochila

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

Vamos definir soluções através de variáveis binárias x_i , $i \in S$ tal que $x_i = 1$ indica que o elemento i pertence à solução e $x_i = 0$ indica que o elemento i não pertence à solução.

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

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

Coberturas, Empacotamentos e Partições

Coberturas, Empacotamentos e Partições são condições que ocorrem freqüentemente na formulação de problemas.

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

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

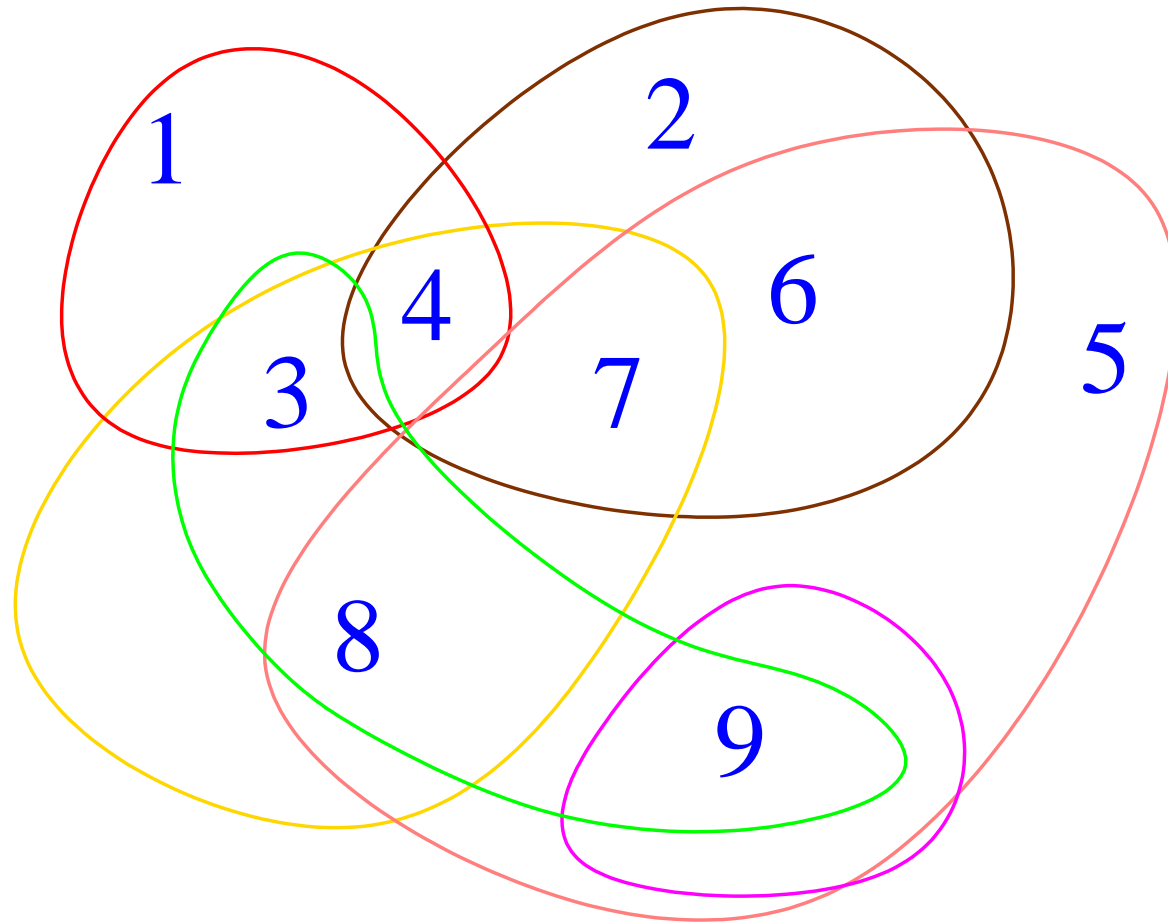
Problema da Cobertura por Conjuntos

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

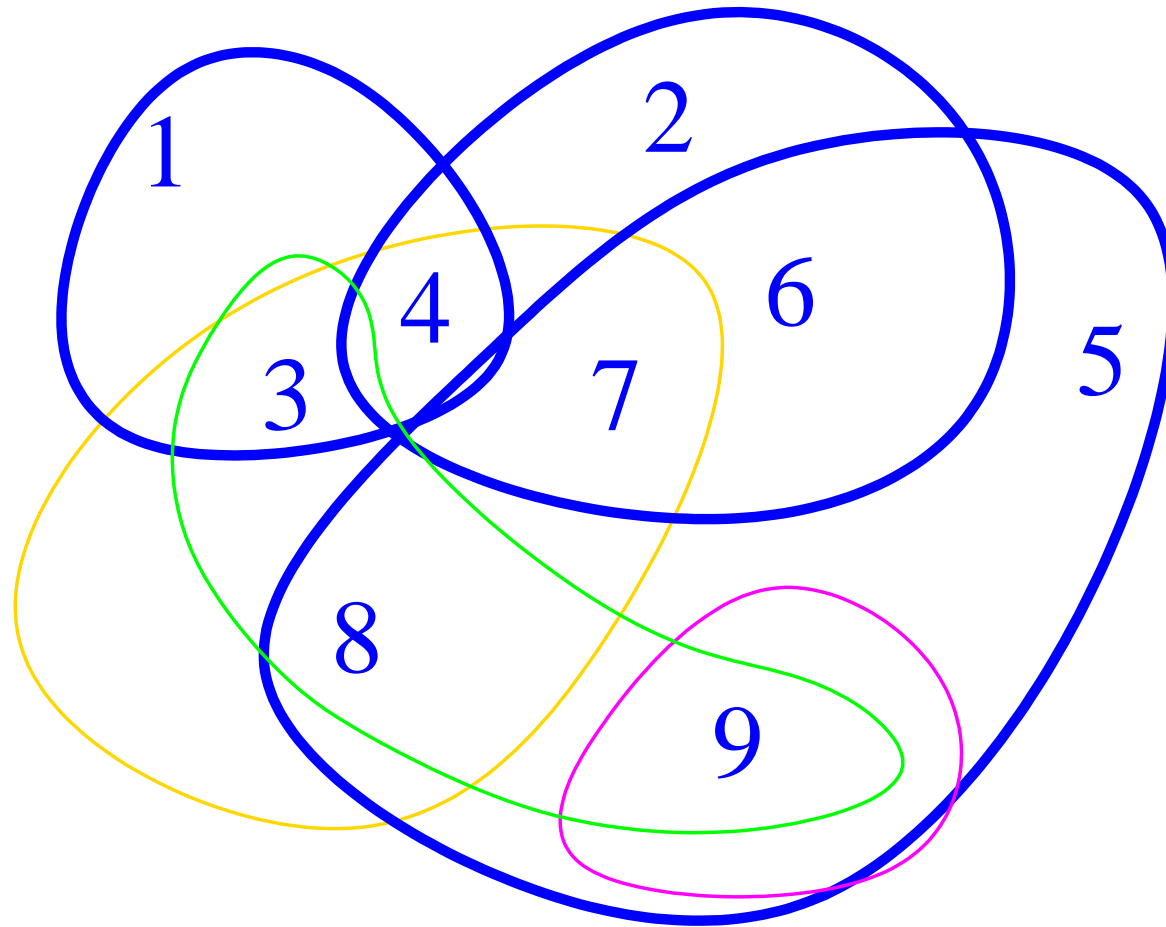
Problema COBERTURA POR CONJUNTOS: Dados conjunto E , subconjuntos \mathcal{S} de E , custos $c(S)$, $S \in \mathcal{S}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $c(\mathcal{S}')$.

Teorema: COBERTURA POR CONJUNTOS é NP-difícil.

Encontre uma cobertura por conjuntos:



Cobertura por conjuntos:



Deteção de Virus

Reportado por Williamson'98 de estudo feito na IBM:

- Para cada vírus determinamos algumas seqüências de bytes (assinaturas), cada seqüência com 20 ou mais bytes.
- Total de 9000 seqüências para todos os virus.
- O objetivo é encontrar o menor conjunto de seqüências que detecta todos os 5000 vírus.

Caso particular do problema de cobertura de conjuntos:

- **Conjuntos:** Cada seqüência determina um conjunto de vírus que contém a seqüência.
- **Conjunto Base:** Conjunto de todos os vírus.
- **Objetivo:** Encontrar uma cobertura de conjuntos de cardinalidade mínima.

Solução encontrada: 180 seqüências para detectar todos os 5000 virus.

Formulação do Problema da Cobertura de Conjuntos

Vamos definir soluções através de variáveis binárias x_S , $S \in \mathcal{S}$ tal que $x_S = 1$ indica que o conjunto S pertence à solução e $x_S = 0$ indica que o conjunto S não pertence à solução.

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

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

A primeira restrição diz que para qualquer elemento do conjunto E , pelo menos um dos conjuntos que o cobrem (conjuntos \mathcal{S}_e) deve pertencer a solução.

Problema da Localização de Recursos

Problema LOCALIZAÇÃO DE RECURSOS (FACILITY LOCATION PROBLEM): Dados potenciais recursos $F = \{1, \dots, n\}$, clientes $C = \{1, \dots, m\}$, custos f_i para instalar o recurso i e custos $c_{ij} \in \mathbb{Z}$ para um cliente j ser atendido pelo recurso i . Encontrar recursos $A \subseteq F$ minimizando custo para instalar os recursos em A e atender todos os clientes

Aplicação: Instalar postos de distribuição de mercadorias, centros de atendimento, instalação de antenas em telecomunicações, etc.

Note que neste problema temos de determinar quais os recursos que iremos instalar e como conectar os clientes aos recursos instalados.

Temos dois tipos de custos envolvidos:

- Se resolvermos instalar o recurso, devemos pagar pela sua instalação.
- Devemos pagar um preço pela conexão estabelecida entre um cliente e o recurso instalado mais próximo.

Vamos definir soluções através de variáveis binárias y_i $i \in F$ tal que $y_i = 1$ indica que o recurso i foi escolhido para ser instalado e $y_i = 0$ indica que o recurso i não vai ser instalado.

e variáveis x_{ij} , tal que

$x_{ij} = 1$ indica que o cliente j será conectado ao recurso i

$x_{ij} = 0$ indica que o cliente j não será conectado ao recurso i .

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

A primeira restrição indica que todo cliente deve ser conectado a algum recurso.

A segunda restrição indica que um cliente só deve ser conectado a um recurso instalado.

Problema da Árvore de Steiner

Seja G um grafo e T (terminais) um subconjunto de vértices de V_G .

Def.: Uma *Árvore de Steiner* de G é qualquer árvore H de G tal que todo elemento de T está em H .

Problema Árvore de Steiner: Dados um grafo G , um custo c_e em \mathbb{Q}_{\geq} para cada aresta e e um conjunto T de V_G , encontrar uma Árvore de Steiner H que minimize $c(H)$.

Aplicações:

- Roteamento em circuitos VLSI (VLSI Layout and routing).
- Projeto de redes de conectividade.
- Determinação de amplificadores de sinal em redes óticas.
- Construção de árvores filogenéticas.

Formulação:

Em muitos problemas que envolvem alguma estrutura de conectividade, é comum usarmos variáveis binárias para as arestas de conexão, pois

1. em geral as arestas tem custos que estamos querendo minimizar/maximizar
2. permitem facilmente escrever restrições relativas às condições de conectividade a que devem respeitar.

Vamos definir soluções através de variáveis binárias x_{ij} tal que

$x_{ij} = 1$ indica que a aresta ij pertence à solução

$x_{ij} = 0$ indica que a aresta ij não pertence à solução

Note que se S é inválido

- podemos dividir o conjunto de vértices em duas partes, I e J tal que
 - $i \in I, j \in J$ e
 - nenhuma aresta de S liga I e J .
 - Isto nos dá um corte que separa i e j .

Se em algum momento temos uma atribuição para x que ainda não é solução, então

- poderemos encontrar um conjunto de arestas que formam este corte separador
- pelo menos uma aresta do corte separador deve estar na solução

Formulação:

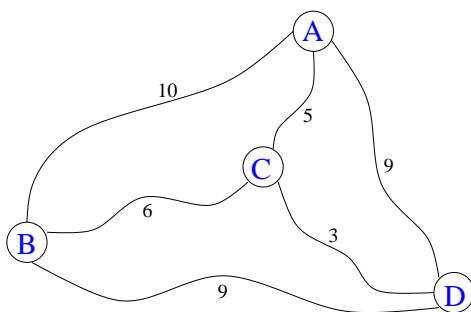
$$\begin{array}{l}
 \text{minimize} \\
 \text{sujeito a}
 \end{array}
 \left\{ \begin{array}{l}
 \sum_{e \in E} c_e x_e \\
 \sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \subset V : \\
 \quad \quad \quad S \text{ separa terminais.} \\
 \quad \quad \quad \text{(desigualdades de corte)} \\
 x_e \in \{0, 1\} \quad \forall e \in E
 \end{array} \right.$$

dizemos que S separa um conjunto de terminais se existe $R \in \mathcal{R}$ tal que $S \cap R \neq \emptyset$ e $R \setminus S \neq \emptyset$

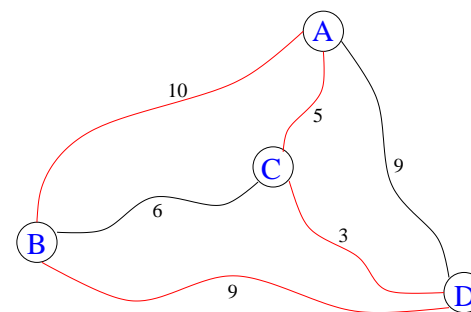
A primeira restrição impõe que todo corte separador deve ter pelo menos uma aresta que pertença à solução.

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 $c(C)$.



Grafo G



Circuito Hamiltoniano de G de custo 27

Aplicações:

- Perfuração e solda de circuitos impressos.
- Determinação de rotas de custo mínimo.
- Seqüenciamento de DNA (Genoma).
- Outras: <http://www.math.princeton.edu/tsp/apps>
- D.S. Johnson: TSP Challenging: www.research.att.com/~dsj/chtsp
- CONCORDE: Applegate, Bixby, Chvátal, Cook'01: Branch & cut para TSP
Soluções para instâncias de até 15112 vértices.

Vamos definir soluções através de variáveis binárias x_{ij} tal que
 $x_{ij} = 1$ indica que a aresta ij pertence ao circuito da solução
 $x_{ij} = 0$ indica que a aresta ij não pertence ao circuito da solução

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

A primeira restrição diz que para todo vértice há duas arestas da solução que incidem no vértice (uma para entrar e outra para sair).

A segunda restrição diz que a solução deve ser conexa.

Note que se não colocarmos as desigualdades da segunda restrição, a solução gerada poderia ser um conjunto de circuitos.

Exercícios:

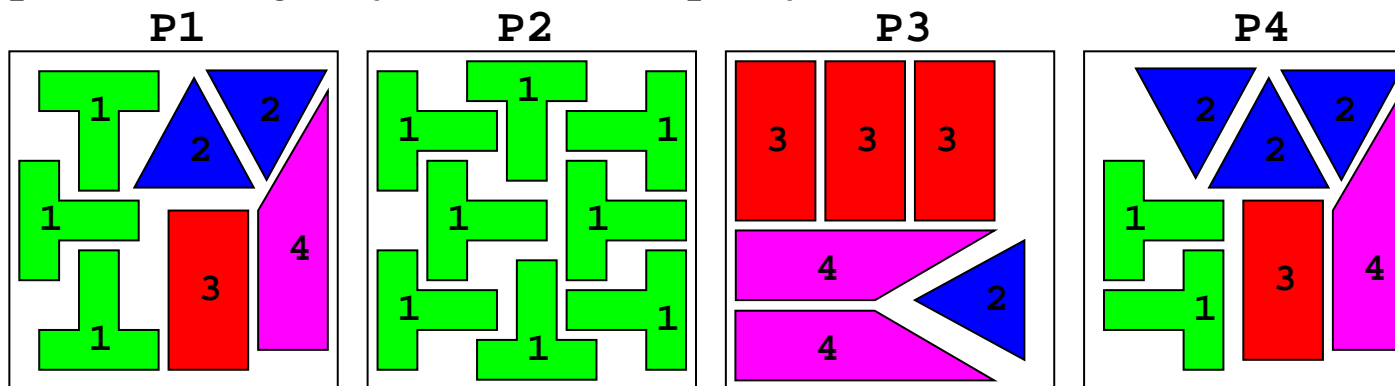
1. A formulação que fizemos do TSP foi para grafos não orientados. Faça uma formulação para o TSP quando as arestas são orientadas (estamos procurando por um circuito hamiltoniano orientado de peso mínimo).
2. Faça uma formulação para encontrar o caminho mínimo de um vértice s e um vértice t , em um grafo com pesos positivos nas arestas usando desigualdades de corte.

Formulações com variáveis inteiras

Problema de Empacotamento

Uma empresa deve vender objetos $O = \{1, 2, \dots, m\}$, cada objeto i com demanda d_i que devem ser recortados a partir de placas que devem ter uma configuração previamente estabelecida. Há um total de k configurações possíveis e cada configuração j tem a_{ij} itens do objeto i . O objetivo é encontrar as quantidades que a empresa deve cortar de cada configuração para suprir a demanda, cortando o menor número de placas possível.

Exemplo de configurações com a disposição dos itens dentro de cada placa.



Ex.: A placa P3 tem 0 itens do objeto 1, 1 item do objeto 2, 3 itens do objeto 3 e 2 itens do objeto 4

Formulação do problema de empacotamento

Vamos definir soluções através de variáveis inteiras $x_j \geq 0$, que dá a quantidade de placas na configuração j que devemos cortar.

Restrições para cada objeto i :

- As placas a serem cortadas devem suprir a demanda do objeto d_i .
- A quantidade de placas na configuração j é x_j .
- Cada configuração j tem a_{ij} itens do tipo i .

Assim, para satisfazer a demanda d_i , devemos impor que

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ik}x_k \geq d_i$$

Considere as configurações apresentadas na figura anterior. Suponha que $d_1 = 950$, $d_2 = 1150$, $d_3 = 495$ e $d_4 = 450$. A formulação ficaria a seguinte:

Formulação:

$$\begin{array}{r}
 \text{minimize} \\
 \\
 \text{sujeito a}
 \end{array}
 \left\{ \begin{array}{l}
 x_1 + x_2 + x_3 + x_4 \\
 3x_1 \quad +8x_2 \quad \quad \quad +2x_4 \geq 950 \\
 2x_1 \quad \quad \quad +1x_3 \quad +3x_4 \geq 1150 \\
 1x_1 \quad \quad \quad +3x_3 \quad +1x_4 \geq 495 \\
 1x_1 \quad \quad \quad +2x_3 \quad +1x_4 \geq 450 \\
 x_i \in \mathbb{Z}^*
 \end{array} \right.$$

Cada linha i contém os dados de um objeto i e cada coluna j contém os dados da configuração j .

Programa Inteiro e Programa Relaxado

Seja P um poliedro (relaxado) e I o conjunto de pontos inteiros em P seja P_I o correspondente poliedro inteiro em P (i.e., $P_I := \text{conv}(I)$).

Queremos otimizar em P_I , mas em geral só temos P .

Informações de P e P_I :

- As soluções de P_I e P podem estar muito próximas.
- Todos os pontos de P_I pertencem a P .
- Se a função objetivo é de minimização, a solução ótima de P é menor ou igual à solução ótima de P_I .
- Se a função objetivo é de maximização, a solução ótima de P é maior ou igual à solução ótima de P_I .

Resolvendo PLI por Arredondamento

Este método consiste em

- modelar o problema como problema de programação linear inteira
- relaxar a formulação para programação linear
- resolver o programa linear
- arredondar as variáveis para cima/baixo, de acordo com o problema.

Vejamos este método aplicado ao exemplo do problema do empacotamento.

Formulação em PLI:

$$\begin{array}{ll}
 \text{minimize} & x_1 + x_2 + x_3 + x_4 \\
 \text{sujeito a} & \left\{ \begin{array}{l}
 3x_1 + 8x_2 + 2x_4 \geq 950 \\
 2x_1 + 1x_3 + 3x_4 \geq 1150 \\
 1x_1 + 3x_3 + 1x_4 \geq 495 \\
 1x_1 + 2x_3 + 1x_4 \geq 450 \\
 x_i \geq 0, \quad x_i \in \mathbb{Z}
 \end{array} \right.
 \end{array}$$

Relaxação do PLI:

$$\begin{array}{l}
 \text{minimize} \\
 \text{sujeito a}
 \end{array}
 \left\{ \begin{array}{l}
 x_1 + x_2 + x_3 + x_4 \\
 3x_1 + 8x_2 + 2x_4 \geq 950 \\
 2x_1 + x_3 + 3x_4 \geq 1150 \\
 1x_1 + 3x_3 + 1x_4 \geq 495 \\
 1x_1 + 2x_3 + 1x_4 \geq 450 \\
 x_i \geq 0
 \end{array} \right.$$

Resolvendo este programa linear, obtemos uma solução de valor 437,656 e valores $x_1 = 0$, $x_2 = 26,406$, $x_3 = 41,875$ e $x_4 = 369,375$.

O número de placas é inteiro \Rightarrow qualquer solução ótima usa pelo menos 438 placas.

Arredondando as variáveis fracionárias para cima, obtemos uma solução de valor $x_1 = 0$, $x_2 = 27$, $x_3 = 42$ e $x_4 = 370$, que nos dão uma solução de 439 placas.

Assim, se a nossa solução não for ótima, estamos usando uma placa a mais que a solução ótima.

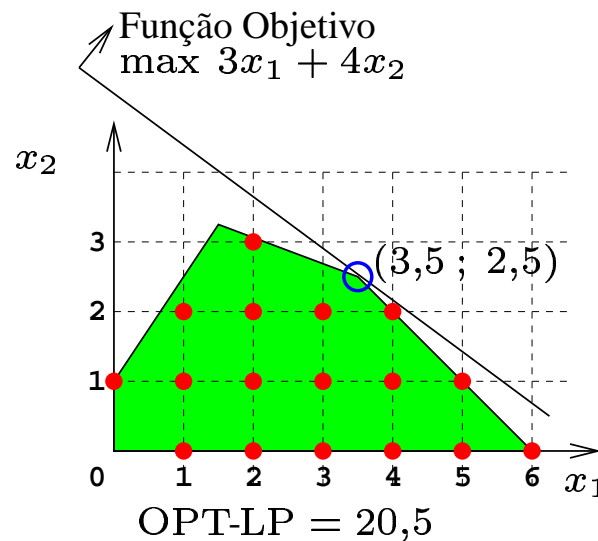
Branch & Bound e Programação Linear Inteira

- Modelar o problema como problema de programação linear inteira
- Relaxar a formulação para programação linear
- Enumerar o espaço de soluções, através de uma árvore: Cada nó da árvore representa um programa linear. Cada programa linear (nó) é ramificado enquanto houver valores fracionários em sua solução.
- Se um nó leva a atribuições inviáveis, ou a soluções não promissoras, podar o nó.

Considere o programa linear inteiro:

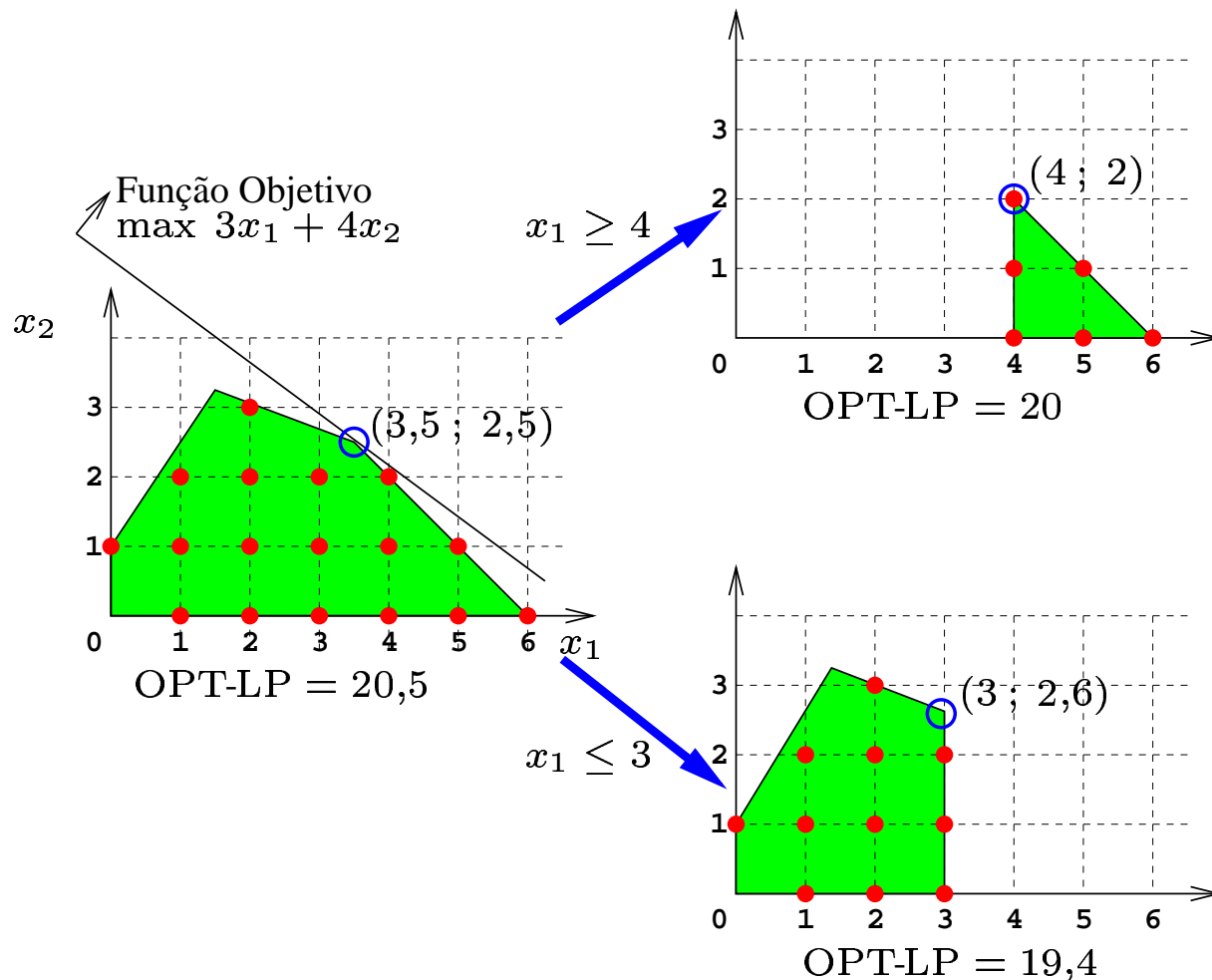
$$\begin{array}{ll}
 \text{minimize} & 3x_1 + 4x_2 \\
 \text{sujeito a} & \left\{ \begin{array}{l} -3x_1 + 2x_2 \leq 2 \\ x_1 + 3x_2 \leq 11 \\ x_1 + x_2 \leq 6 \\ x_1 \geq 0, \quad x_2 \geq 0 \\ x_i \in \mathbb{Z}^* \end{array} \right.
 \end{array}$$

Representação gráfica do programa relaxado e os pontos inteiros:

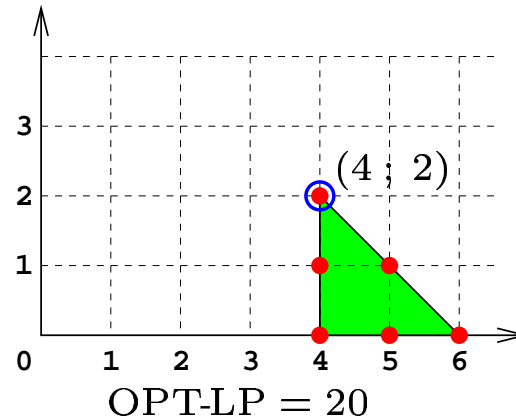


Vamos resolver o programa inteiro através do Branch (& Bound)

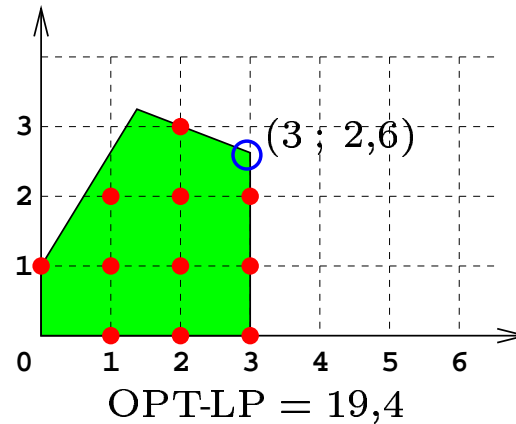
A primeira solução ótima fracionária corresponde ao ponto $(x_1 = 3,5 ; x_2 = 2,5)$. Escolhendo a variável x_1 (que tem valor fracionário 3,5) para fazer branching, separamos o problema em duas partes. Uma inserindo a restrição $x_1 \leq 3$ e outra inserindo a restrição $x_1 \geq 4$. Os dois subproblemas estão representados a seguir:



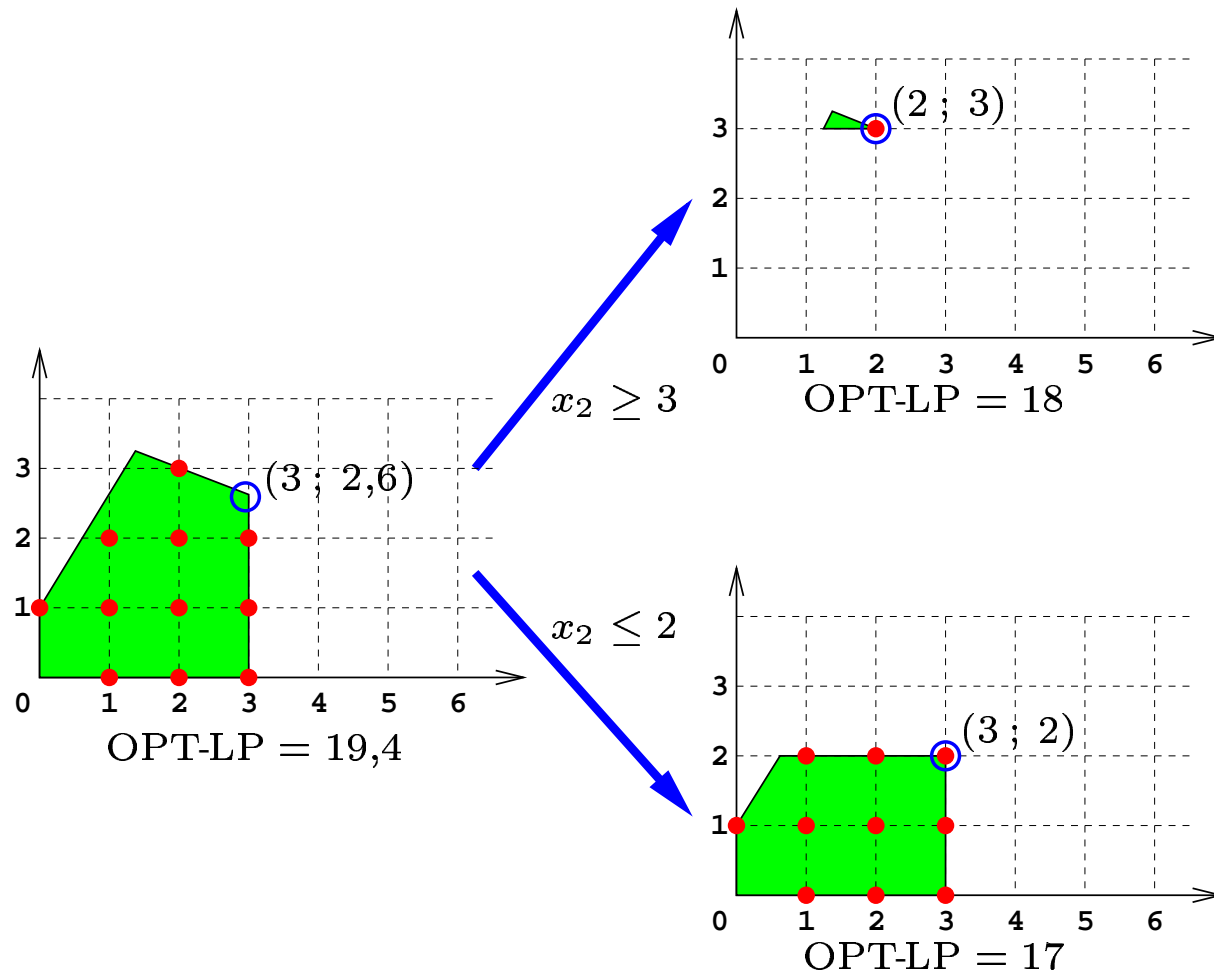
Um dos programas já tem solução inteira e não precisamos continuar sua ramificação:



O outro programa tem solução ótima em $(3 ; 2,6)$ e portanto fracionário na variável x_2 . Ainda é necessário ramificar este nó:

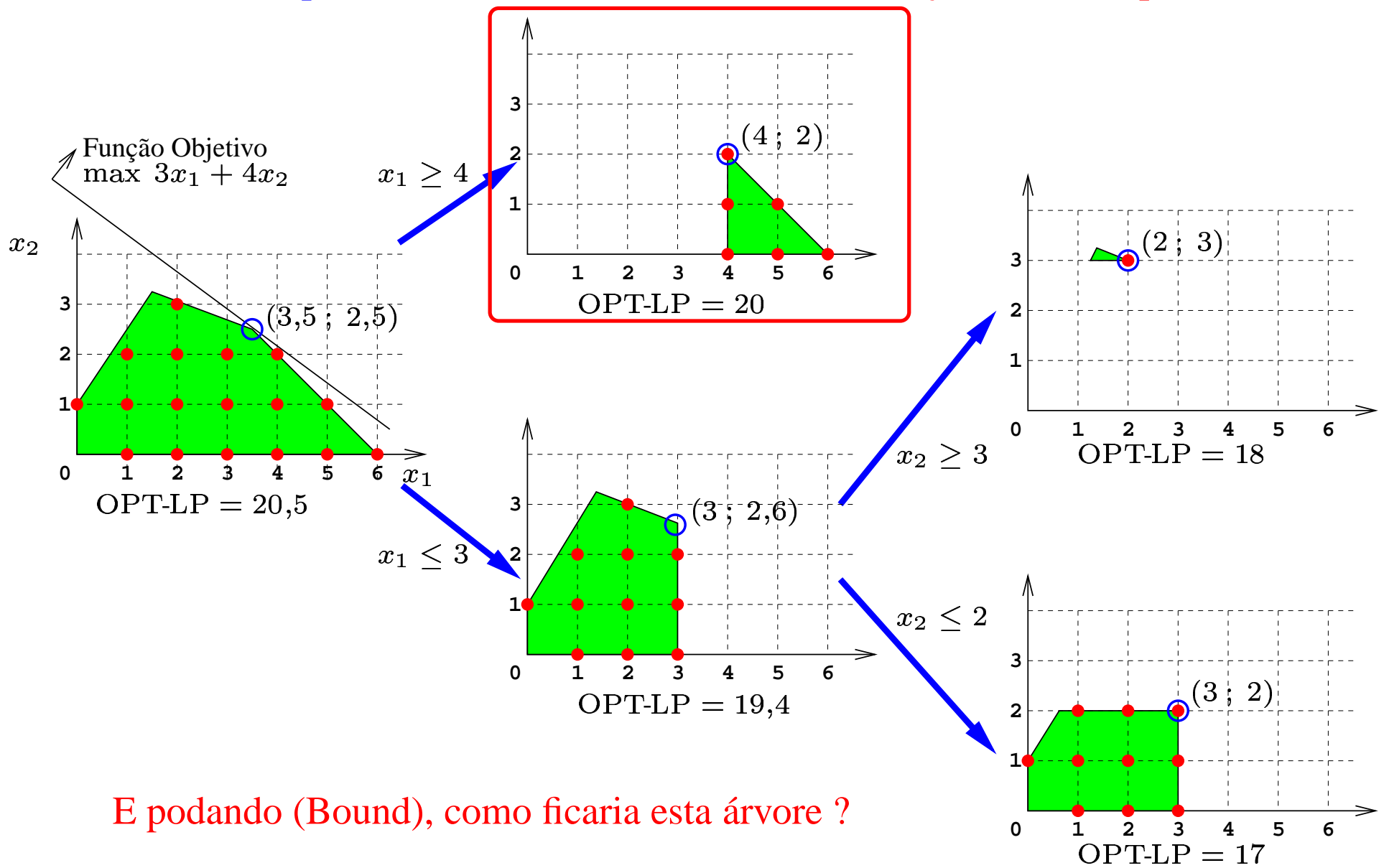


Ramificando em x_2 , para os casos $x_2 \leq 2$ e $x_2 \geq 3$, obtemos os seguintes programas



Desta vez, os dois novos programas tem soluções inteiras e podemos parar o processo.

Árvore completa de Branch (sem Bound). E a solução ótima do problema



E podando (Bound), como ficaria esta árvore ?

Estratégias comuns envolvidas na árvore de B&B

- **Escolhendo o nó a ramificar**

Usamos o nó que tem a maior distância entre o limitante inferior e superior.

Por exemplo, em um problema de minimização, pegue o nó que tem o menor limite inferior.

Isto permite diminuir a diferença entre limitantes superior e inferior.

- **Escolha da variável para ramificar:**

- Variável “mais fracionária”: Parte fracionária mais próxima de 0,5

- Variável “menos fracionária”: Parte fracionária mais distante de 0,5

- **Ramificação por restrição:**

Encontre uma restrição válida $ax \leq b$ e divida em duas partes:

1. Um ramo tem inserido a desigualdade $ax \leq b'$ e

2. outro ramo tem inserido a desigualdade $ax \geq b''$.

Ex.: Encontre uma desigualdade do TSP do tipo $x(\delta(S)) \geq 2$ e

1. coloque a restrição $x(\delta(S)) = 2$ em um ramo e

2. coloque a restrição $x(\delta(S)) \geq 3$ em outro ramo.

(naturalmente há um esforço para encontrar tal desigualdade)

- **Representando a árvore de Branch & Bound:**

Em geral não armazenamos a árvore, mas uma lista dos nós ativos.

Cada vez que escolhemos um nó ativo para ramificar, removemos este do conjunto e inserimos seus ramos (novos nós).

Estrutura de dados comum para armazenar nós: Fila de prioridade

- **Guarde a melhor solução viável**

- **Poda da árvore:**

Use o valor da melhor solução encontrada combinada com estratégias de limitantes superiores para podar ramos não promissores.

Boas estratégias de poda podem evitar crescimento rápido da árvore.

- **Percorrendo a árvore de B&B**

- Se não temos solução viável: aplicar busca em profundidade por ramos mais promissores

- Se temos solução viável: misture escolha do melhor nó com busca em profundidade

- **Branch & Bound para programas com variáveis em $\{0, 1\}$:**

Ramos da enumeração consistem em fixar variáveis para 0 ou 1.

- **Fixação de variáveis por implicações lógicas:**

A fixação do valor de algumas variáveis pode levar a fixar outras.

Considere o problema do TSP resolvido através de B & B com LP.

Removendo as arestas fixadas em 0 e contraindo arestas fixadas em 1, podemos obter um grafo tal que:

- se houver vértice de grau dois, podemos incluir as arestas incidentes na solução do nó.

- se houver uma aresta cuja remoção desconecta o grafo, podemos podar o ramo deste nó.

Método de Planos de Corte

Seja (P, c) um programa linear, onde P é um poliedro e c é a função objetivo.

Suponha que P é descrito por número grande de desigualdades.

Ainda assim, podemos obter uma solução ótima para (P, c) .

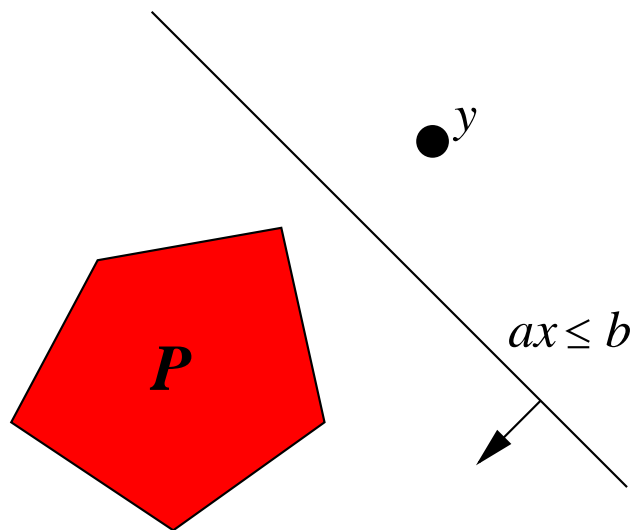
Estratégia:

1. Não usar P , mas começar com um poliedro inicial Q (descrito por poucas desigualdades) que contém P
2. Repetidamente encontrar uma solução ótima y de Q e caso $y \notin P$, adicionamos a Q uma desigualdade válida (chamada de plano de corte) que separa (corta) y de P sem perder nenhuma solução de P .
3. Parar quando encontrar uma solução ótima de P .

Precisamos repetir o passo 2 muitas vezes ?

Otimização × 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 $ay > b$.



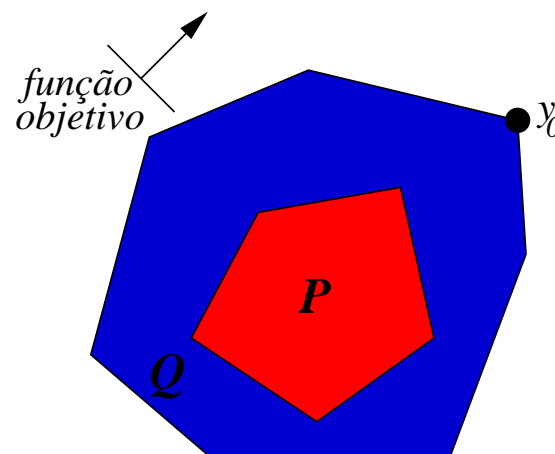
Teorema: (Grötschel, Lovász, Schrijver'81) *O problema de otimização de um programa linear pode ser resolvido em tempo polinomial se e somente se o problema da separação para o poliedro do programa linear pode ser resolvido em tempo polinomial.*

ALGORITMO PLANOS DE CORTE (P, c) Dantzig, Fulkerson e Johnson'54

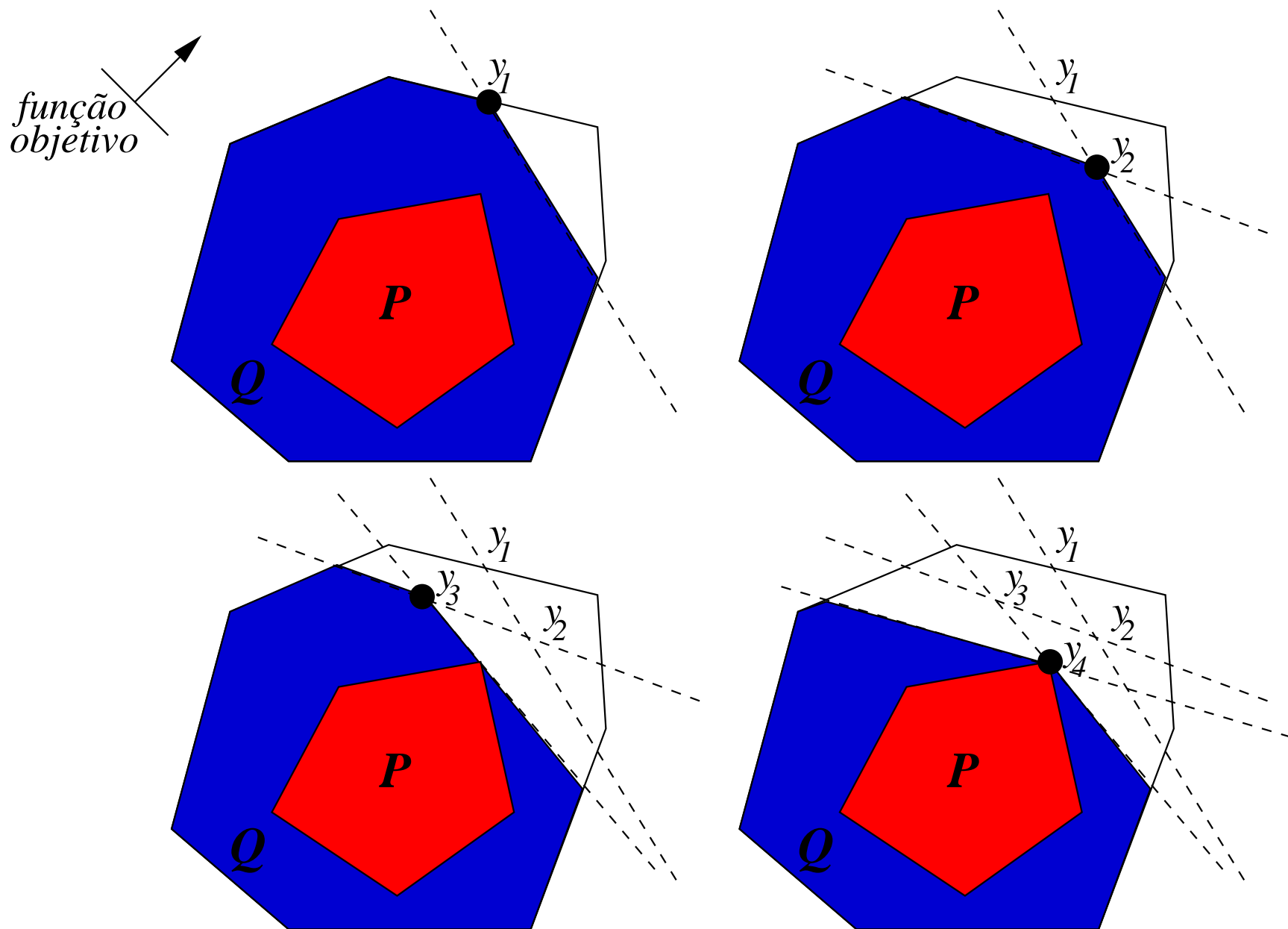
P poliedro (não necessariamente explícito),

c função objetivo

- 1 $Q \leftarrow \{\text{Poliedro inicial}\}$
- 2 $y \leftarrow \text{OPT-LP}(Q, c)$
- 3 enquanto y pode ser separado de Q faça
- 4 seja $ax \leq b$ desigualdade de P que separa y
- 5 $Q \leftarrow Q \cap \{x : ax \leq b\}$
- 6 $y \leftarrow \text{OPT-LP}(Q, c)$
- 7 se $Q = \emptyset$ retorne \emptyset
- 8 senão retorne y ,



onde $\text{OPT-LP}(Q, c)$ retorna solução ótima do programa linear (Q, c)



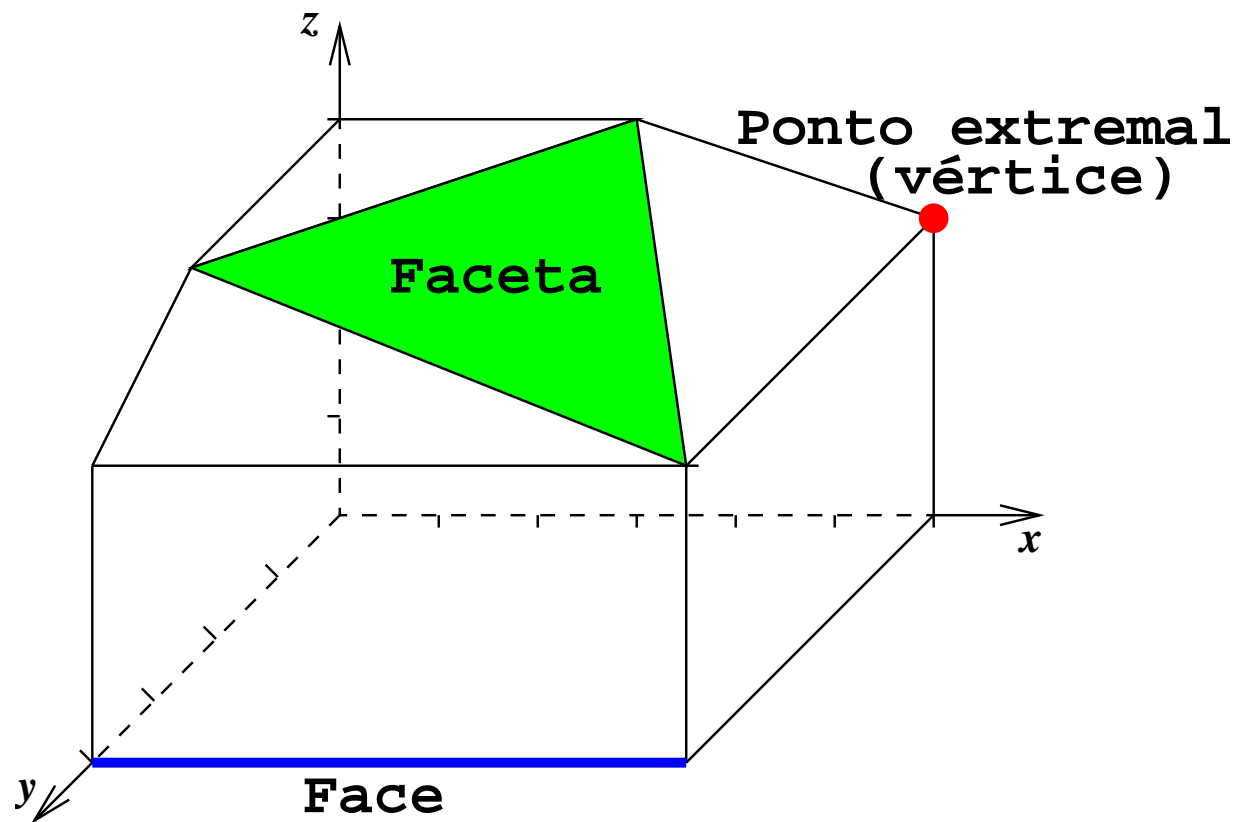
Que tipo de desigualdade são as melhores para serem inseridas ?

Desigualdades Válidas, Faces e Facetas

Seja P um poliedro.

- $ax \leq b$ é uma **desigualdade válida** para P se $P \subseteq \{x : ax \leq b\}$.
- Um conjunto F é dito ser uma **face** de P se existe desigualdade válida $ax \leq b$ tal que $F = P \cap \{x : ax \leq b\}$
- Uma face F de P é dita ser **própria** se $F \neq P$.
- Uma face F de P é dita ser **não-trivial** se $\emptyset \neq F \neq P$.
- Uma face não-trivial F é dita ser uma **faceta** de P se F não está contida em nenhuma outra face própria de P .

Exemplo de Faces e Facetas



Observe que as melhores desigualdades válidas são aquelas que induzem facetas.

Geração de Planos de Corte para programa linear inteiro

Estratégia:

1. Formular o problema como problema de programação linear inteira
2. Relaxar a formulação inteira para programação linear
3. Repetidamente inserir planos de corte

Como obter os planos de corte ?

- Gerando desigualdades válidas a partir do programa linear relaxado
- Gerando desigualdades pelas propriedades das soluções do problema

Gerando desigualdades a partir do programa linear

Cortes de Chvátal:

1. Suponha que queremos soluções inteiras, $x \in \mathbb{Z}^m$ para um poliedro $P := \{x : Ax \leq b\}$, $A \in \mathbb{Q}^{n \times m}$ e $b \in \mathbb{Q}^n$.
2. Idéia: Combinar as linhas do sistema $Ax \leq b$ para obter uma desigualdade $ax \leq t$ tal que o vetor a é inteiro e t não.
3. Inserir a nova desigualdade $ax \leq \lfloor t \rfloor$

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.

Queremos obter $P_{Emp}(G) := conv\{\chi^M \in \mathbb{R}^E : M \text{ é um emparelhamento}\}$.

Formulação Inteira do problema do Emparelhamento:

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

Relaxação linear

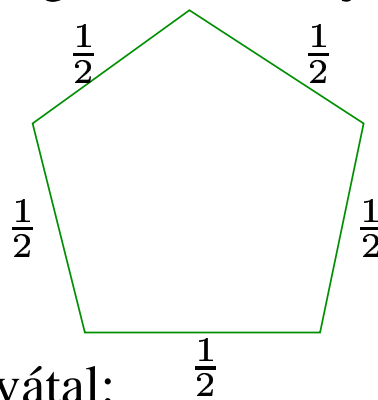
$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} c_e x_e \\ (P) \text{ sujeito a} & \left\{ \begin{array}{ll} \sum_{e \in \delta(v)} x_e \leq 1 & \forall v \in V, \\ x_e \geq 0 & \forall e \in E. \end{array} \right. \end{array}$$

Sabemos que quando G é bipartido $P = P_{Emp}$.

E quando G não é bipartido, a igualdade ocorre ?

Não.

Se P tem custos unitários, a seguinte atribuição é ótima com valor 2,5



Vamos gerar um corte de Chvátal: $\frac{1}{2}$

1. Seja $U \subseteq V$ tal que $|U| \geq 3$ é ímpar.
2. Some as desigualdades $\sum_{e \in \delta(u)} x_e \leq 1$ para todo $u \in U$.
3. Obtemos: $2 \sum_{e \in E(U)} x_e + \sum_{e \in \delta(u)} x_e \leq |U|$,
onde $E(U)$ são arestas com ambos extremos em U .
4. Ignorando o segundo termo: $2 \sum_{e \in E(U)} x_e \leq |U|$
5. Dividindo por dois: $\sum_{e \in E(U)} x_e \leq \frac{|U|}{2}$
6. Lado esquerdo é inteiro e o direito é fracionário: $\sum_{e \in E(U)} x_e \leq \lfloor \frac{|U|}{2} \rfloor$
7. Novo corte (que separa atribuição acima): $\sum_{e \in E(U)} x_e \leq \frac{|U|-1}{2}$

Considere o novo programa linear (P') com os cortes de Chvátal:

$$\begin{array}{l} \text{maximize} \\ (P') \text{ sujeito a} \end{array} \left\{ \begin{array}{l} \sum_{e \in E} c_e x_e \\ \sum_{e \in \delta(v)} x_e \leq 1 \quad \forall v \in V, \\ \sum_{e \in E[S]} x_e \leq \frac{|S| - 1}{2} \quad \forall S \subseteq V, |S| \text{ ímpar}, \\ x_e \geq 0 \quad \forall e \in E \end{array} \right.$$

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

Teorema: (Edmonds'65) $P' = P_{Emp}$.

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

Veja código em

<http://elib.zib.de/pub/Packages/mathprog/index.html>

Corolário: O problema do emparelhamento máximo em grafos quaisquer pode ser resolvido em tempo polinomial.

Outros métodos para geração de planos de corte para programas lineares inteiros

- Cortes de Gomory (Gomory'58)
- Cortes de Schrijver (Schrijver'80)

Estes métodos de planos de corte (Chvátal, Gomory e Schrijver)

1. são independentes do problema
2. garantem uma solução ótima inteira em tempo finito
3. nem sempre são bem sucedidos como no problema de emparelhamento
4. em geral são lentos e apresentam pequena melhora em cada iteração

Como gerar planos de corte bons ?

Use as propriedades estruturais das soluções do seu problema em particular para conseguir desigualdades válidas boas, se possível que induzem facetas

Planos de corte e Conectividade

Muitos problemas vistos envolvem conectividade:

- Floresta de Steiner
- Caixeiro Viajante
- Survivable Network Design

Por exemplo, no TSP, uma das desigualdades válidas foi a seguinte:

$$\sum_{e \in \delta(S)} x_e \geq 2, \quad \forall \emptyset \neq S \subset V$$

I.e., para todo conjunto de vértices S , $\emptyset \neq S \subset V$, o número de arestas escolhidas na solução (arestas e com $x_e = 1$) que pertencem ao corte $\delta(S)$ deve ser pelo menos 2.

Como testar se um ponto x satisfaz todas as desigualdades de corte acima ?

Como testar se dois vértices s e t estão separados por um corte (S, \bar{S}) que viola a desigualdade de corte, i.e., está ocorrendo

$$\sum_{e \in \delta(S)} x_e < 2 \quad s \in S, \quad t \in \bar{S}$$

podemos executar o algoritmo para st -corte mínimo.

Note que o algoritmo deve ser executado não para o grafo original, mas onde cada aresta e tem capacidade x_e .

Se o valor do st -corte mínimo for menor que 2, então o corte gerado viola a desigualdade de corte e podemos inseri-la como desigualdade válida.

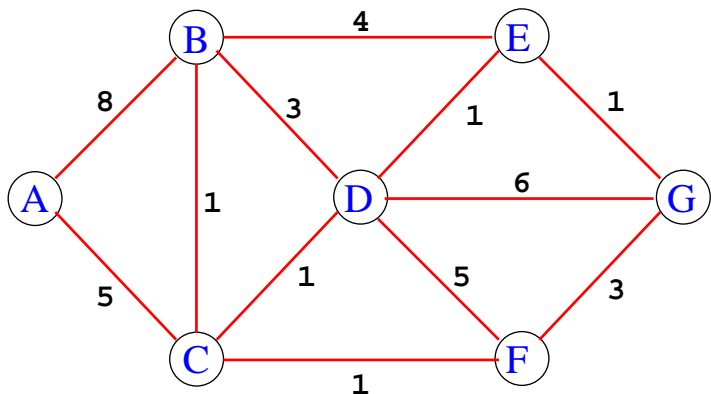
- Estratégia Direta:

Testar para todos os pares de vértices: $O(n^2)$ execuções do st -corte mínimo.

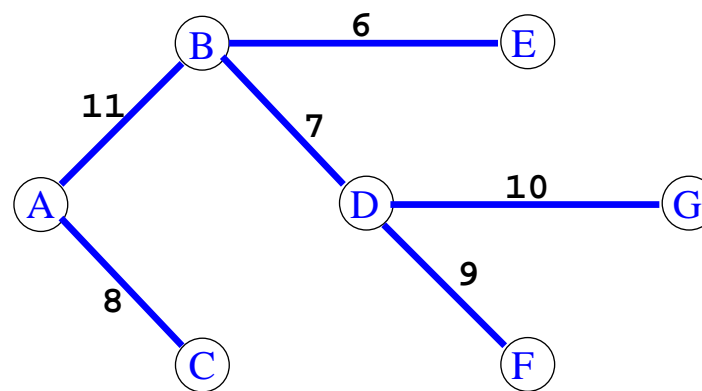
- Árvore de Cortes de Gomory-Hu (Gomory, Hu'61):

Todos os cortes representados por uma árvore usando $n - 1$ execuções do st -corte mínimo

Árvore de Cortes de Gomory-Hu:



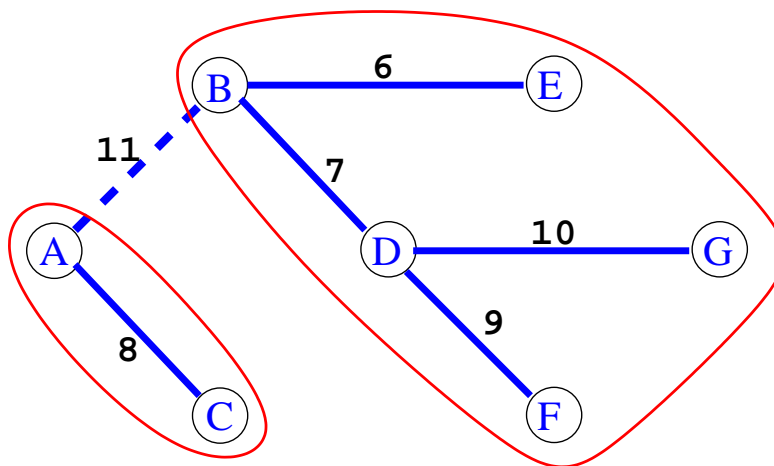
Grafo $G = (V, E)$

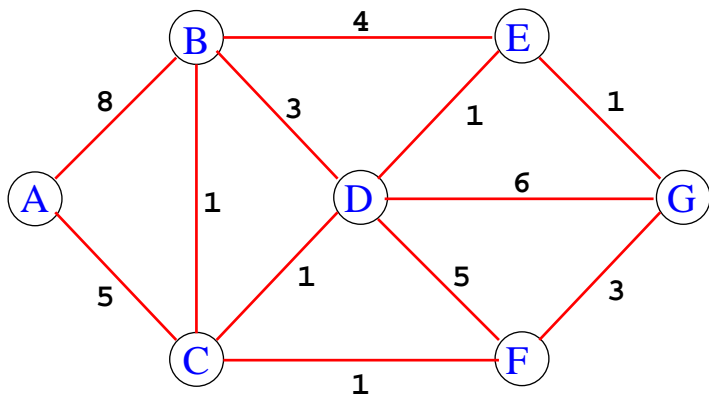


Árvore T de Cortes de Gomory-Hu de G

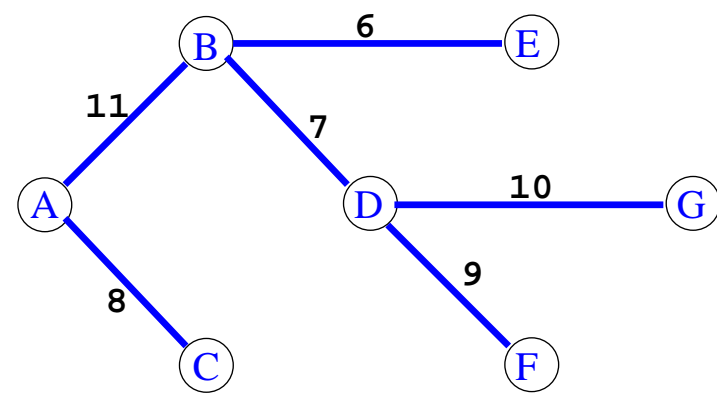
Cada aresta na árvore de cortes de Gomory-Hu representa um corte.
 A capacidade do corte é o peso da aresta em T .

Exemplo do corte de capacidade 11 representado pela aresta (A, B) em T



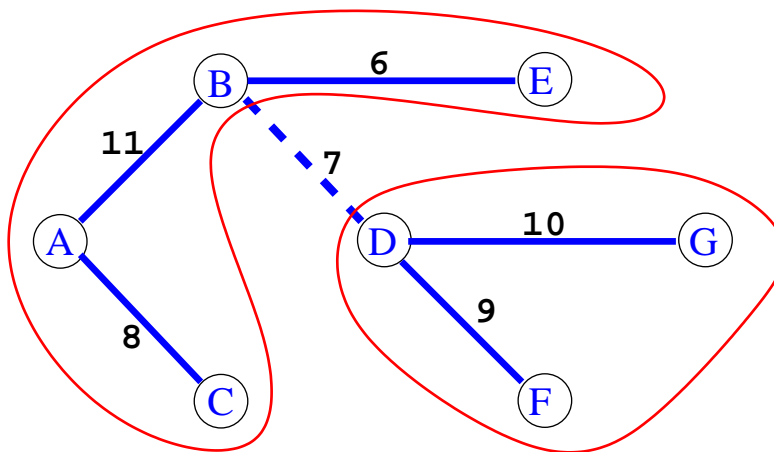


Grafo $G = (V, E)$



Árvore T de Cortes de Gomory-Hu de G

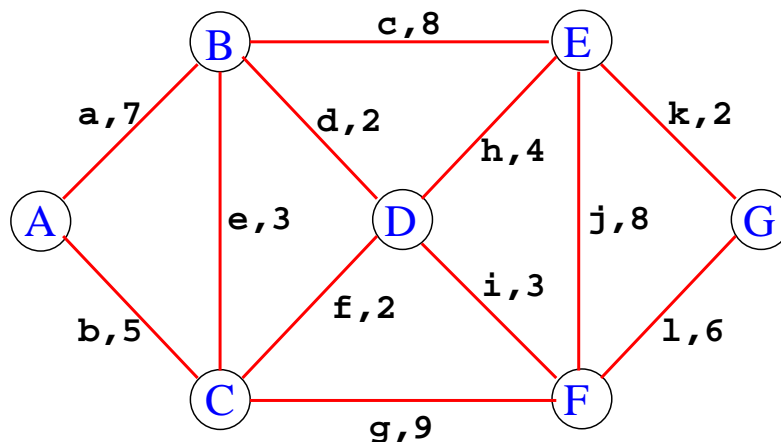
Corte mínimo que separa u e v em G é dado pelas componentes geradas pela remoção da aresta de capacidade mínima no caminho único entre u e v em T .



Exemplo de corte mínimo que separa os vértices A e G (7 é o mínimo em $\{11, 7, 10\}$).

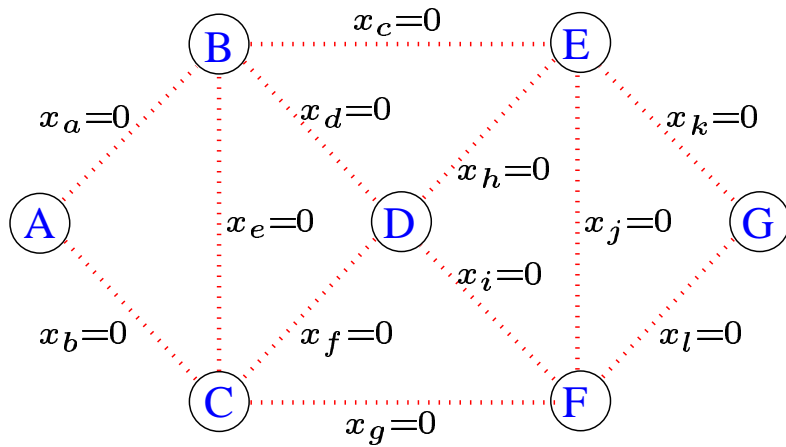
Separação para o TSP

Vamos ver um exemplo de como se comporta a inserção de desigualdades para o TSP para o seguinte grafo:



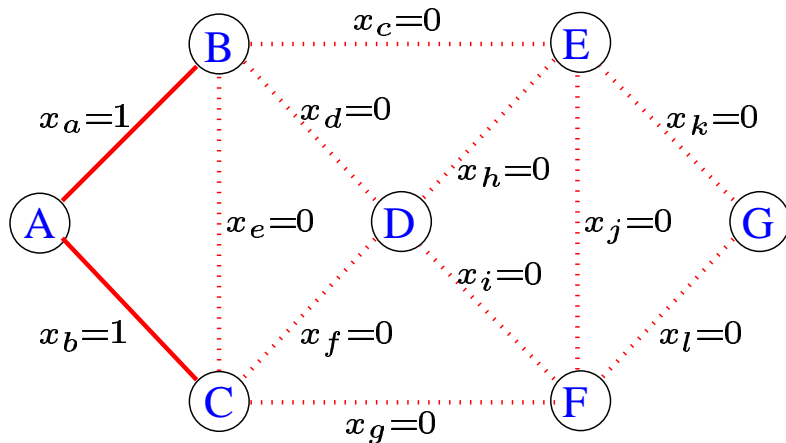
Grafo com nome de arestas e seus custos

Para não sobrecarregar, em cada passo seguinte apresentaremos apenas os valores obtidos da solução ótima fracionária sem colocar os custos das arestas e seus nomes.



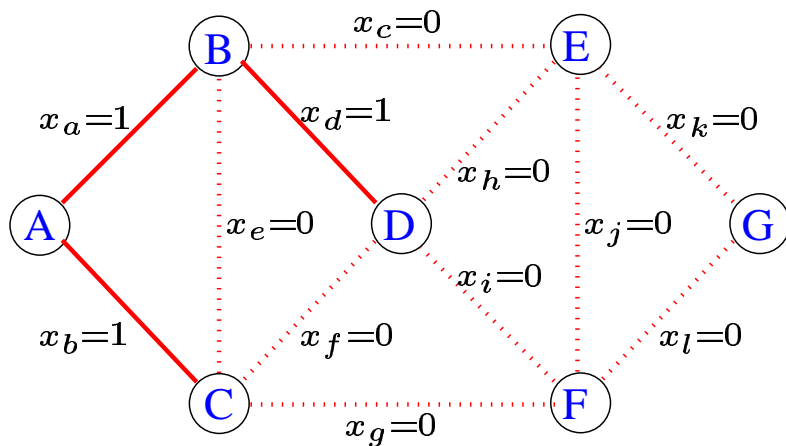
$$\begin{aligned} \min \quad & c_a x_a + c_b x_b + \dots + c_l x_l \\ \text{s.a.} \quad & \left\{ \begin{array}{l} 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{array} \right. \\ & \text{Solução ótima} = 0 \end{aligned}$$

Apenas restrições $0 \leq x_e \leq 1, \forall e \in E$

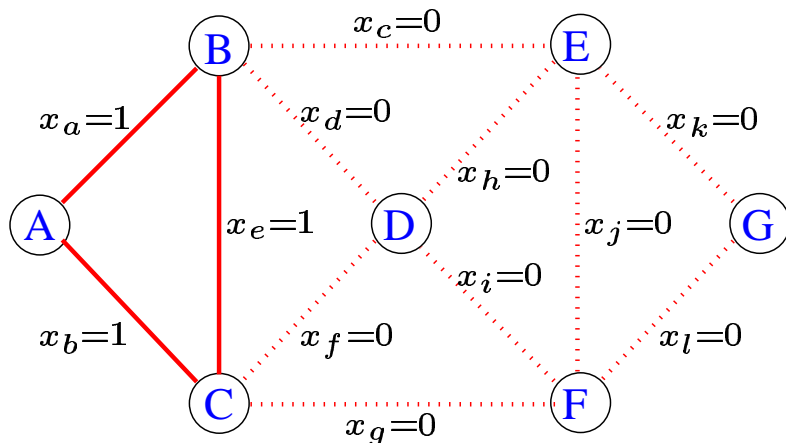


$$\begin{aligned} \min \quad & c_a x_a + c_b x_b + \dots + c_l x_l \\ \text{s.a.} \quad & \left\{ \begin{array}{l} x_a + x_b = 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{array} \right. \\ & \text{Solução ótima} = 12 \end{aligned}$$

Adicionando $x(\delta(A)) = 2$



Adicionando $x(\delta(B)) = 2$



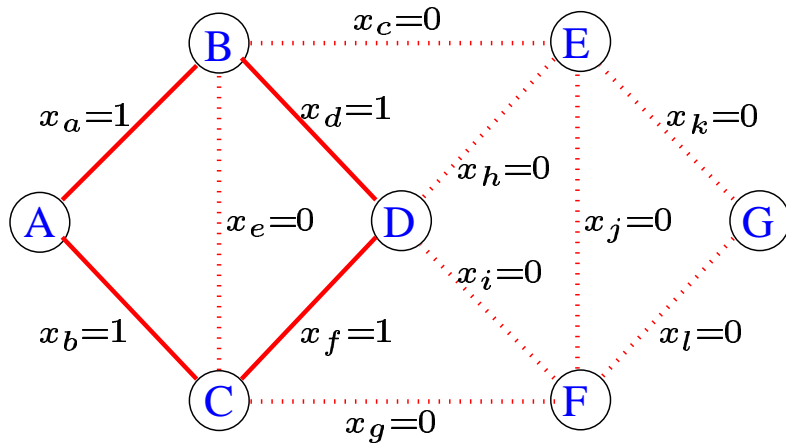
Adicionando $x(\delta(C)) = 2$

$$\begin{aligned} \min & \quad C_a x_a + C_b x_b + \dots + C_l x_l \\ \text{s.a.} & \quad \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{cases} \end{aligned}$$

Solução ótima = 14

$$\begin{aligned} \min & \quad C_a x_a + C_b x_b + \dots + C_l x_l \\ \text{s.a.} & \quad \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{cases} \end{aligned}$$

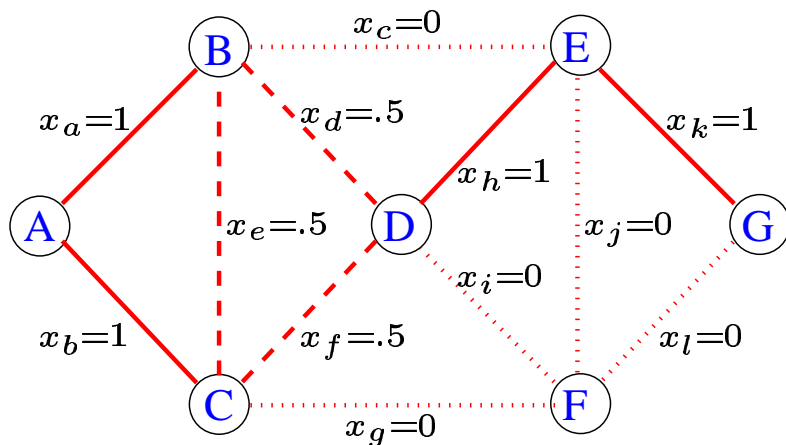
Solução ótima = 15



Adicionando $x(\delta(D)) = 2$

$$\begin{aligned} \min \quad & c_a x_a + c_b x_b + \dots + c_l x_l \\ \text{s.a.} \quad & \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{cases} \end{aligned}$$

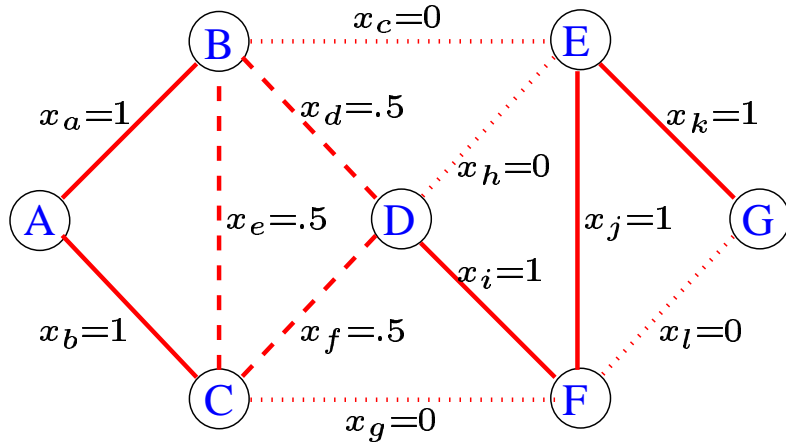
Solução ótima = 16



Adicionando $x(\delta(E)) = 2$

$$\begin{aligned} \min \quad & c_a x_a + c_b x_b + \dots + c_l x_l \\ \text{s.a.} \quad & \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{cases} \end{aligned}$$

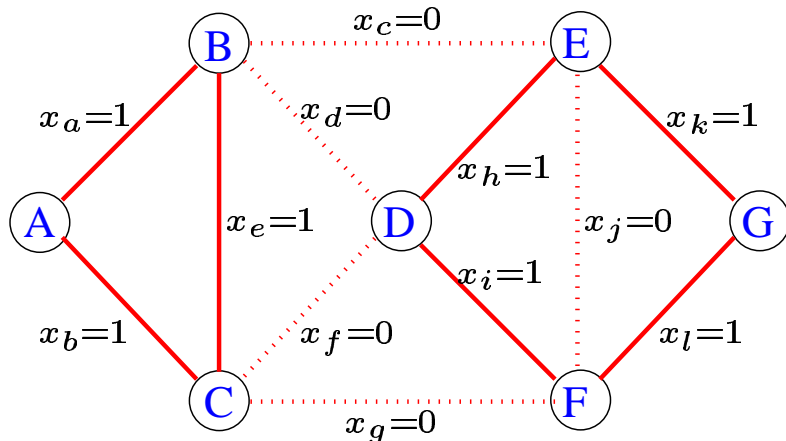
Solução ótima = 21,5



Adicionando $x(\delta(F)) = 2$

$$\begin{aligned} \min \quad & C_a x_a + C_b x_b + \dots + C_l x_l \\ \text{s.a.} \quad & \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ x_g + x_i + x_j + x_l = 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{cases} \end{aligned}$$

Solução ótima = 28.5



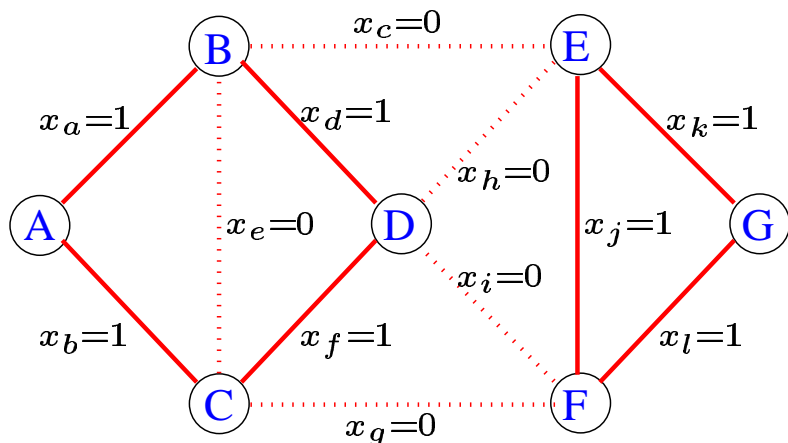
Adicionando $x(\delta(G)) = 2$

$$\begin{aligned} \min \quad & C_a x_a + C_b x_b + \dots + C_l x_l \\ \text{s.a.} \quad & \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ x_g + x_i + x_j + x_l = 2 \\ x_k + x_l = 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{cases} \end{aligned}$$

Solução ótima = 30

Todas as restrições $x(\delta(v)) = 2$ para todo vértice v foram inseridas.

Agora vamos inserir as desigualdades de corte:



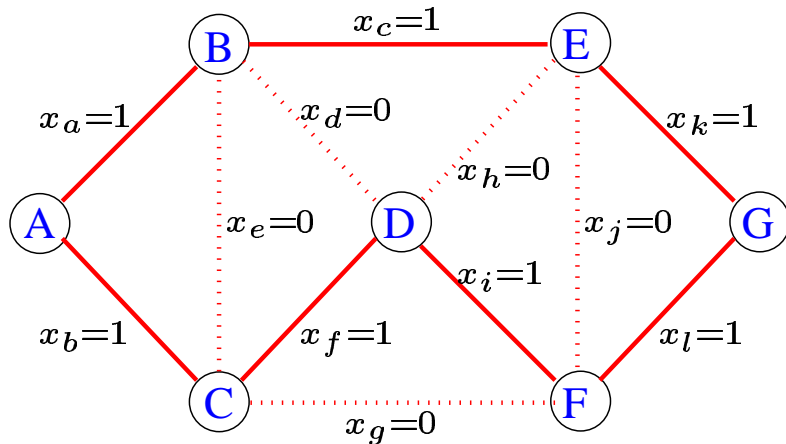
Adicionando desigualdade do corte mínimo que separa A e D (corte de valor 0)

$$x(\delta(S)) \geq 2$$

onde $S = \{A, B, C\}$

$$\begin{array}{l} \min \quad c_a x_a + c_b x_b + \dots + c_l x_l \\ \text{s.a.} \quad \left\{ \begin{array}{l} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ x_g + x_i + x_j + x_l = 2 \\ x_k + x_l = 2 \\ x_c + x_d + x_f + x_g \geq 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{array} \right. \end{array}$$

Solução ótima = 32



Adicionando desigualdade do corte mínimo que separa A e E (corte de valor 0)

$$x(\delta(S)) \geq 2$$

onde $S = \{A, B, C, D\}$

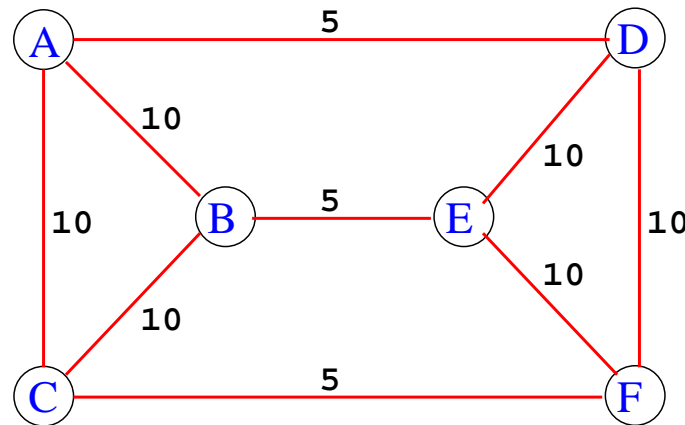
$$\begin{aligned} \min \quad & c_a x_a + c_b x_b + \dots + c_l x_l \\ \text{s.a.} \quad & \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ x_g + x_i + x_j + x_l = 2 \\ x_k + x_l = 2 \\ x_c + x_d + x_f + x_g \geq 2 \\ x_c + x_g + x_h + x_i \geq 2 \\ 0 \leq x_a \leq 1, \dots, 0 \leq x_l \leq 1 \end{cases} \end{aligned}$$

Solução ótima = 33

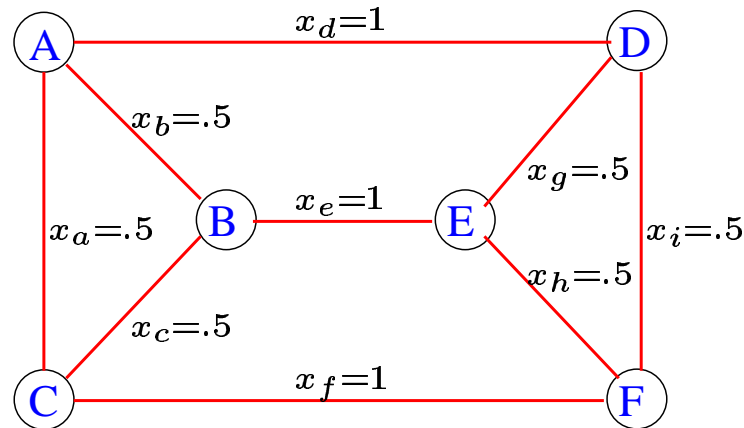
Chegamos em uma solução inteira!!! Solução ótima para TSP de valor 33.

Exemplo de solução ótima fracionária do programa relaxado do TSP

Grafo G (grafo envelope) e custos nas arestas:



Solução ótima fracionária de peso 45 (LP com restrições de grau e de corte):



Chvátal'75: apresenta separação de soluções como acima (comb inequalities)

Branch & Cut

Um algoritmo Branch & Cut é uma

- combinação do método Branch & Bound e
- geração de planos de corte durante a geração da árvore de B & B.

Parece simples, mas um algoritmo Branch & Cut envolve muitas sofisticacões

Estratégia:

- ★ **investir em cada nó para limitar crescimento da árvore e**
- ★ **delimitar problema**

- por estratégias de separação,
- uso de heurísticas primais em cada nó,
- métodos de ramificação,
- pré-processamento em cada nó
- gerenciamento de desigualdades válidas

Estratégias consideradas na separação

- **Uso de heurísticas para encontrar planos de corte**
as vezes vale mais a pena usar uma heurística para obter planos de corte em tempo rápido que algoritmos que garantidamente determinam a existência de classe de planos de corte, mas a um alto custo computacional.
- **Redução do número de desigualdades no nó**
Em sistemas grandes, remover as desigualdades que não estão justas
Uma desigualdade $a \cdot x \leq \alpha$ é justa se a solução ótima do LP x^* satisfaz a desigualdade com igualdade.
Isto diminui o número de restrições e permite que a resolução do sistema fique mais rápida.
- ***Tailing off***
Desigualdade de separação de alguns pontos pode gerar melhorias pequenas e sobrecarregar muito o sistema.
Em vez de separar estes pontos, ir direto para a ramificação.

- **Seleção de desigualdades**

- **Escolha desigualdades de classes diferentes**

A escolha de desigualdades de diferentes classes é mais efetiva que concentrar em desigualdades de mesma classe.

- **Selecione as desigualdades mais violadas**

Para cada desigualdade válida encontrada $a \cdot x \leq \alpha$, calcule $a \cdot x' - \alpha$, onde x' é a solução da relaxação atual. Quanto maior for a diferença, maior a chance de crescimento do limitante inferior.

- **Selecione desigualdades que cobrem todo espaço de variáveis**

Um sistema linear “se adapta” a cada nova desigualdade, mudando o mínimo. Quando introduzimos desigualdades com grande quantidade de variáveis não nulas, a chance disso ocorrer é menor.

Estratégias de delimitação

- **Heurísticas Primais**

Aproveite informações da solução do programa relaxado em cada nó para obter soluções viáveis.

- **Pre-processamento dos nós**

Fixação de variáveis por implicações lógicas. Considere o problema do TSP resolvido através de B & B com LP.

Removendo as arestas fixadas em 0 e contraindo arestas fixadas em 1, podemos obter um grafo tal que:

- se houver vértice de grau dois, podemos incluir as arestas incidentes na solução do nó.

- se houver uma aresta cuja remoção desconecta o grafo, podemos podar o ramo deste nó.

Estratégias usadas na ramificação

- Ramificação por variáveis

Escolha da variável com parte fracionária mais próxima de 0,5.

Escolha da variável com parte fracionária mais distante de 0,5.

- Ramificação por restrição

Ex.: Digamos que encontramos uma desigualdade que vale $x(\delta(S)) = 2,5$

1. coloque a restrição $x(\delta(S)) = 2$ em um ramo e

2. coloque a restrição $x(\delta(S)) \geq 3$ em outro ramo.

BRANCH-CUT-SIMPLIFICADO (P, c, \min)

```
1   $y^* \leftarrow \{\text{Solução heurística}, \emptyset \text{ se não obteve solução}\}$ 
2   $U \leftarrow \text{val}(y^*)$ 
3  Ativos  $\leftarrow \{P\}$ 
4  enquanto Ativos  $\neq \emptyset$  faça
5      escolha  $Q \in$  Ativos e remova  $Q$  de Ativos
6       $y \leftarrow \text{OPT-LP}(Q, c, \min)$ 
7      enquanto  $y \neq \emptyset$  e encontra plano  $ax \leq \gamma$  que separa  $y$  faça
8           $Q \leftarrow Q \cap \{x : ax \leq \gamma\}$ 
9           $y \leftarrow \text{OPT-LP}(Q, c, \min)$ 
10     se  $\text{val}(y) < U$  então
11         se  $y$  é inteiro então
12              $U \leftarrow \text{val}(y);$ 
13              $y^* \leftarrow y$ 
14         senão
15             seja  $y_i$  uma variável fracionária em  $y$  e  $\alpha$  seu valor
16              $Q' \leftarrow Q \cap \{x : x_i \leq \lfloor \alpha \rfloor\}$ 
17              $Q'' \leftarrow Q \cap \{x : x_i \geq \lceil \alpha \rceil\}$ 
18             Ativos  $\leftarrow$  Ativos  $\cup \{Q', Q''\}$ 
19     devolva  $y^*$ .
```

Apêndice I: Grafos

Def.: Um subgrafo $H = (V_H, E_H)$ de $G = (V, E)$ é um grafo tal que $V_H \subseteq V$, $E_H \subseteq E$ e $E_H \subseteq V_H \times V_H$.

Def.: Um subgrafo $H = (V_H, E_H)$ de $G = (V, E)$ é gerador se $V_H = V$.

Def.: Uma passeio P em $G = (V, E)$ é uma seqüência $(v_0, e_1, v_1, e_2, \dots, v_{k-1}, e_{k-1}, v_k)$ onde $v_i \in V$ e $e_i = \{v_{i-1}, v_i\} \in E$. Chamamos v_0 e v_k de origem e destino de P , resp.

Def.: Uma trilha é um passeio que não repete arestas.

Def.: Uma caminho é um passeio que não repete vértices.

Def.: Um ciclo é uma trilha com os vértices de origem e destino iguais.

Def.: Um ciclo euleriano de G é um ciclo que contém todas as arestas de G .

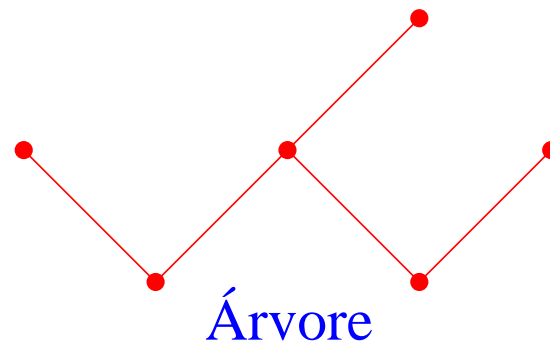
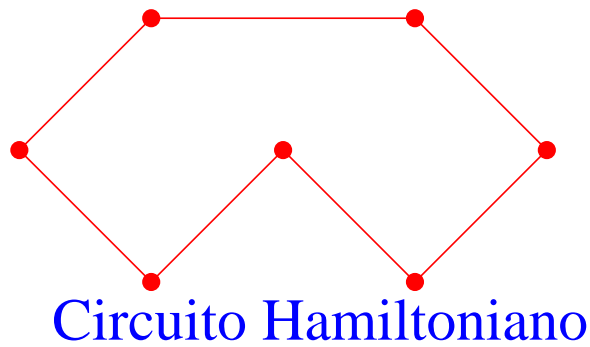
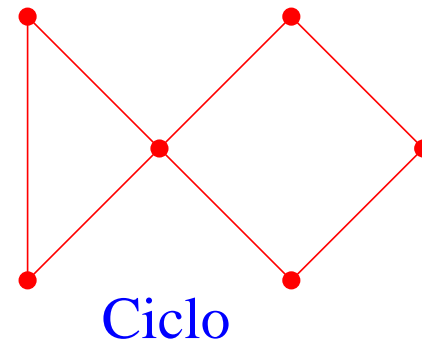
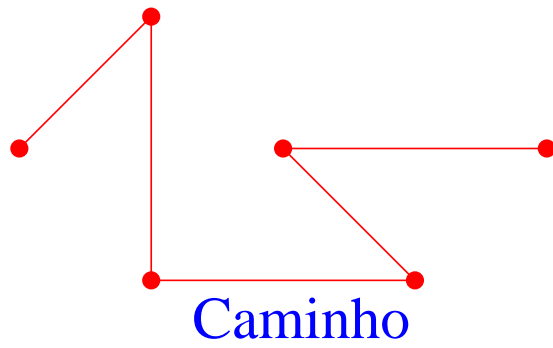
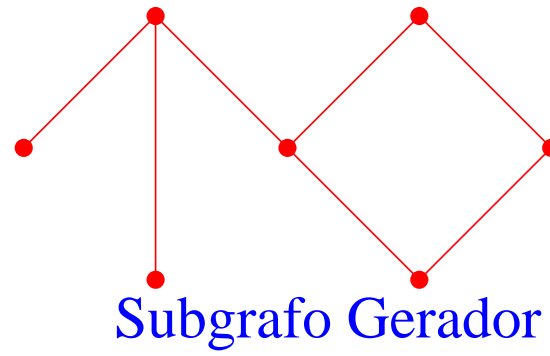
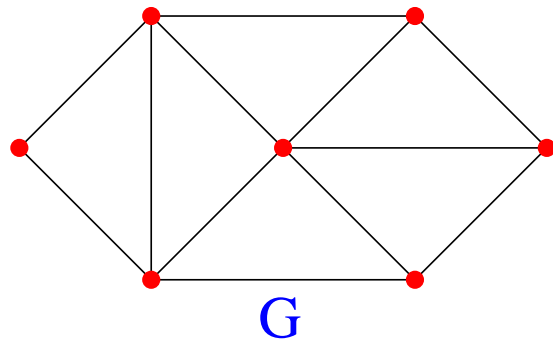
Def.: Um circuito é um ciclo que não repete vértices.

Def.: Um circuito hamiltoniano de G é um circuito que contém todos os vértices de G .

Def.: Um grafo $G = (V, E)$ é conexo se para todo $u, v \in V$, existe caminho ligando u e v .

Def.: Uma floresta $F = (V, E)$ é um grafo que não contém circuitos.

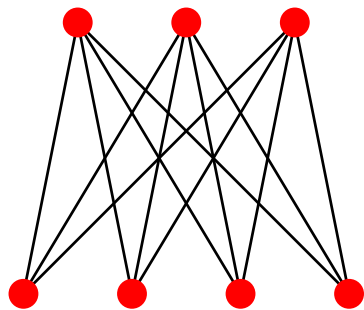
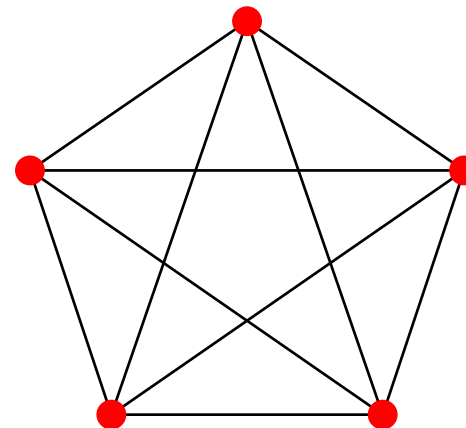
Def.: Uma árvore $F = (V, E)$ é um grafo conexo sem circuitos.



Def.: Um grafo $G = (V, E)$ é bipartido se existe partição (X, Y) de V tal que $E \subseteq X \times Y$.

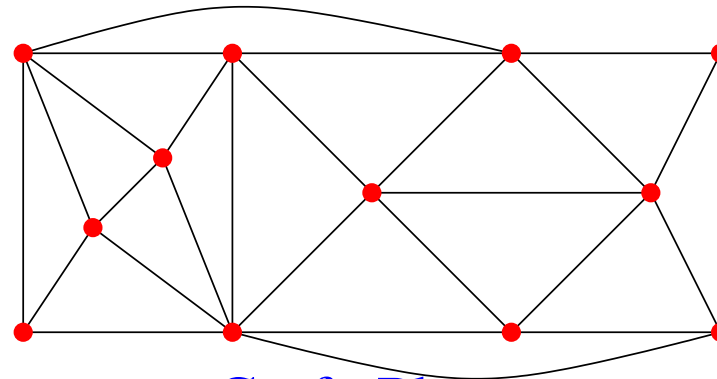
Def.: Um grafo $G = (V, E)$ é bipartido completo se existe partição (X, Y) de V tal que $E = X \times Y$. Denotado por $K_{|X| \times |Y|}$.

Def.: Um grafo $G = (V, E)$ é completo se $E = V \times V$. Denotado por $K_{|V|}$.

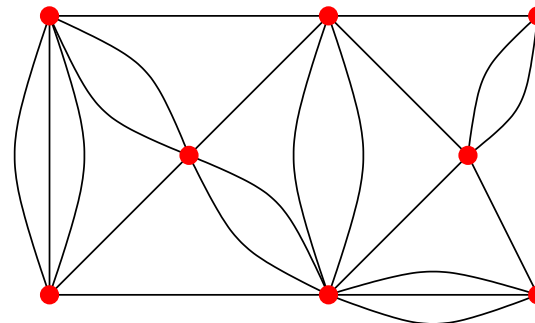
 $K_{3 \times 4}$  K_5

Def.: Um grafo $G = (V, E)$ é planar se pode ser desenhado no plano com arestas interceptando apenas nos extremos.

Def.: Um multigrafo $G = (V, E)$ é uma extensão de grafos para admitir várias cópias de uma aresta.



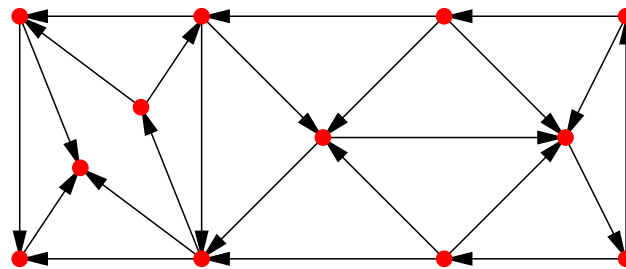
Grafo Planar



Multigrafo

Def.: Uma *aresta orientada* é um par ordenado (u, v) representada por $u \rightarrow v$ para vértices u e v . Dizemos que u é o início (ou cabeça) e v é o fim (calda) da aresta.

Def.: Um grafo $G = (V, E)$ é dito ser orientado se suas arestas são orientadas.



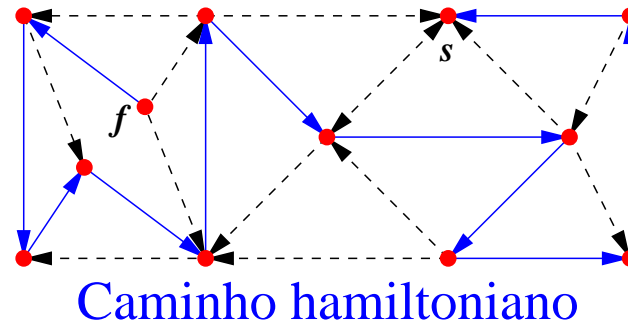
Grafo Orientado

Def.: Um vértice é dito ser fonte se não há arestas entrando no vértice.

Def.: Um vértice é dito ser sorvedouro se não há arestas saindo do vértice.

Algumas definições anteriores se aplicam para grafos orientados.

Exemplo de caminho hamiltoniano orientado de f para s



Caminho hamiltoniano

Def.: Dado um grafo não orientado $G = (V, E)$ e um vértice $v \in V$, denotamos por $\delta(v)$ o conjunto das arestas que tem um extremo em v .

Def.: Dado um grafo não orientado $G = (V, E)$ e conjunto $S \subseteq V$, denotamos por $\delta(S)$ o conjunto das arestas que tem exatamente um extremo em S .

Def.: Dado um grafo orientado $G = (V, E)$ e um vértice $v \in V$, denotamos por $\delta^+(v)$ o conjunto das arestas de E que tem a calda em v e denotamos por $\delta^-(v)$ o conjunto das arestas de E que tem o início em v .

Def.: Dado um grafo orientado $G = (V, E)$ e conjunto $S \subseteq V$, denotamos por $\delta^+(S)$ o conjunto das arestas de E que tem a calda em S e o início em $V \setminus S$ e denotamos por $\delta^-(S)$ o conjunto das arestas de E que tem o início em S e a calda em $V \setminus S$.

Para entender mais sobre grafos, veja Bondy and Murty [BM76].

Apêndice II: Definições e notação sobre conjuntos

Def.: Dado um conjunto A e uma função f sobre A , $f : A \rightarrow \mathbb{R}$, e um elemento $e \in A$, denotamos por $f_e = f(e)$.

Def.: Dado um conjunto A e uma função numérica $f : A \rightarrow \mathbb{R}$, denotamos por $f(B) := \sum_{e \in B} f(b)$ para todo $B \subseteq A$.

Exemplo: Se $G = (V, E)$ é um grafo, v é um vértice de V , e x é uma função numérica nas arestas $x : E \rightarrow \mathbb{R}$ então vale

$$x(\delta(v)) = \sum_{e \in \delta(v)} x_e = \sum_{e \in \delta(v)} x(e)$$

(soma dos valores da função x aplicada nas arestas que incidem em v).

Bibliografia

Referências

- [AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice-Hall, 1993.
- [BM76] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications*. Macmillan/Elsevier, 1976.
- [CCD⁺01] M.H. Carvalho, M.R. Cerioli, R. Dahab, P. Feofiloff, C.G. Fernandes, C.E. Ferreira, K.S. Guimarães, F.K. Miyazawa, J.C. Pina Jr., J. Soares, and Y. Wakabayashi. *Uma introdução sucinta a algoritmos de aproximação*. Editora do IMPA - Instituto de Matemática Pura e Aplicada, Rio de Janeiro–RJ, 2001. M.R. Cerioli, P. Feofiloff, C.G. Fernandes, F.K. Miyazawa (editors).

- [Chv83] V. Chvátal. *Linear Programming*. Freeman, 1983.
- [CK00] P. Crescenzi and V. Kann. *A Compendium of NP Optimization Problems*, 2000. URL:
<http://www.nada.kth.se/~viggo/wwcompendium/>.
- [CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1990.
- [FW96] C.E. Ferreira and Y. Wakabayashi. *Combinatória poliédrica e planos de corte faciais*. X Escola de Computação, UNICAMP, Campinas–SP, 1996.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, CA, 1979.
- [Kar72] R.M. Karp. Reducibility among combinatorial problems. In

- R.E. Miller and J.M. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, 1972.
- [LLRS90] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley, 1990. Reprint of the 1985 original.
- [Pap94] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, 1994.
- [PS82] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, 1982.
- [Vaz00] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2000.
- [Wol98] L.A. Wolsey. *Integer Programming*. Wiley, 1998.