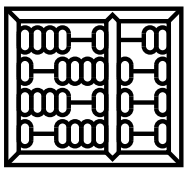
Introdução a Programação Linear e Inteira

Flávio Keidi Miyazawa



UNIVERSIDADE DE CAMPINAS - UNICAMP
INSTITUTO DE COMPUTAÇÃO - IC

Campinas, 2002-2016

Sumário I

Sumário II

Programação Linear

Anteriormente, vimos sobre Programação Linear

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

$$\begin{array}{ll} \text{minimize} & c_1 x_1 + c_2 x_2 + \cdots + c_m x_m \\ \text{sujeito a} & \left\{ \begin{array}{lcl} a_{11} x_1 + a_{12} x_2 + \cdots + a_{1m} x_n & \leq & b_1 \\ a_{21} x_1 + a_{22} x_2 + \cdots + a_{2m} x_n & \geq & b_2 \\ \vdots & \vdots & \vdots \\ a_{n1} x_1 + a_{n2} x_2 + \cdots + a_{nm} x_n & = & b_n \\ x_i \in \mathbb{Q} & & \end{array} \right. \end{array}$$

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

Programação Linear

Programação Linear levou a algoritmos eficientes para

- Emparelhamento de peso máximo em grafos bipartidos
- Problema do Transporte de custo mínimo
- Problema do Fluxo de custo mínimo
- Problema do *st*-Fluxo de valor máximo
- Problema do *st*-Corte de capacidade mínima
- Problema do caminho de custo mínimo de s a t (*)
- Problema dos k caminhos de s a t vértices disjuntos de custo mínimo (*)
- Problema dos k caminhos de s a t arestas disjuntos de custo mínimo (*)

(*) com custos não negativos.

Programação Linear Inteira

Os problemas resolvidos 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.

Programação Linear Inteira

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

$$\begin{array}{ll} \text{minimize} & c_1 x_1 + c_2 x_2 + \cdots + c_m x_m \\ \text{sujeito a} & \left\{ \begin{array}{lcl} a_{11} x_1 + a_{12} x_2 + \cdots + a_{1m} x_n & \leq & b_1 \\ a_{21} x_1 + a_{22} x_2 + \cdots + a_{2m} x_n & \geq & b_2 \\ \vdots & \vdots & \vdots \\ a_{n1} x_1 + a_{n2} x_2 + \cdots + a_{nm} x_n & = & b_n \\ x_i \in \mathbb{Z} & & \end{array} \right. \end{array}$$

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. □

Programação Linear Inteira

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 do método branch & bound.Cada nó da árvore representa um programa linear. Cada programa linear(nó) é ramificado enquanto houver valores fracionários em sua solução.

Programação Linear Inteira

- 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, $i = 1, \ldots, n$, 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 $S = \{1, \ldots, n\}$ 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 $\sum_{i \in S'} s_i \leq B$.

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.

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

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

Formulação do Problema da Mochila

Exemplo: Considere $n = 6$, $B = 100$ com os seguintes dados

i	1	2	3	4	5	6
s_i	25	37	42	39	52	29
v_i	2	7	4	3	8	5

$$\text{maximize} \quad 2x_1 + 7x_2 + 4x_3 + 3x_4 + 8x_5 + 5x_6$$

$$\text{sujeito a} \quad \begin{cases} 25x_1 + 37x_2 + 42x_3 + 39x_4 + 52x_5 + 29x_6 \leq 100 \\ x_i \in \{0,1\} \quad \text{para } i = 1,\ldots,6 \end{cases}$$

De maneira compacta:

$$\text{maximize} \quad \sum_{i=1}^{6} v_i x_i$$

$$\text{sujeito a} \quad \begin{cases} \sum_{i=1}^{6} s_i x_i \leq B \\ x_i \in \{0,1\} \quad \text{para } i = 1,\ldots,6 \end{cases}$$

Coberturas, Empacotamentos e Partições

Coberturas, Empacotamentos e Partições são condições que ocorrem frequentemente 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**

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

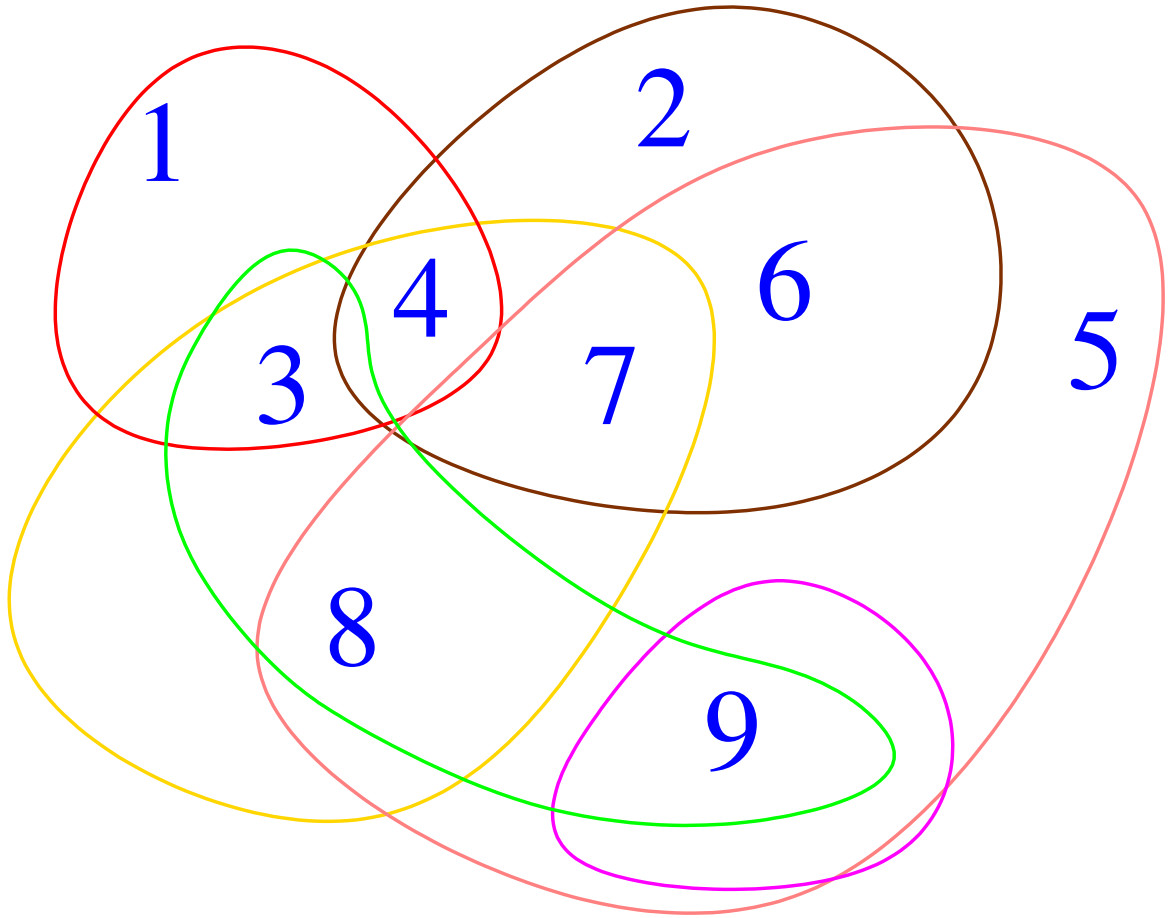
Problema da Cobertura por Conjuntos

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

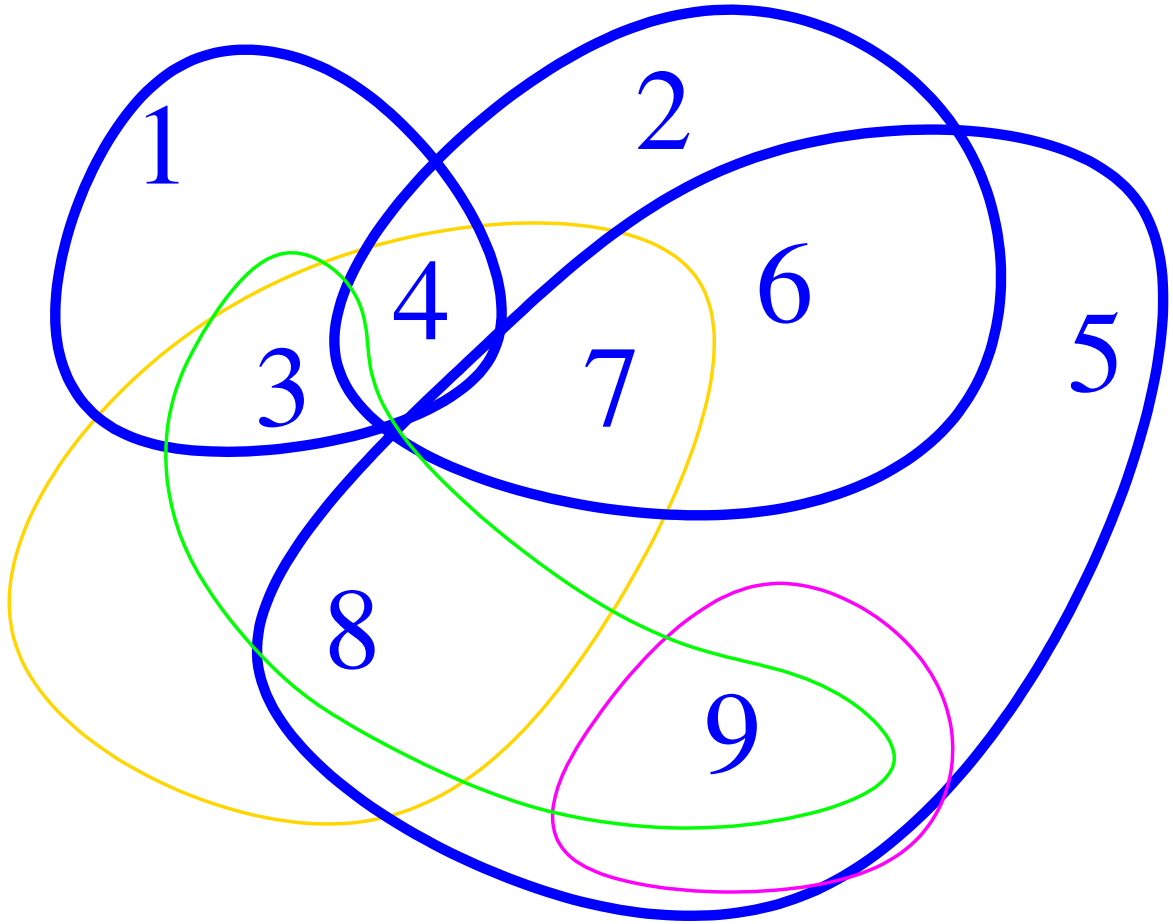
Problema COBERTURA POR CONJUNTOS: Dados conjunto E, subconjuntos $\mathcal{S}$ de E, custos $c(S)$, $S \in \mathcal{S}$, encontrar cobertura $\mathcal{S}' \subseteq \mathcal{S}$ que minimiza $c(\mathcal{S}')$.

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 por conjuntos:

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

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

Formulação do Problema da Cobertura por 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.

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S x_S \\ \text{sujeito a} \quad & \begin{cases} \sum_{S \in \mathcal{S}_e} x_S \geq 1 & \forall e \in E \\ x_S \in \{0,1\} & \forall S \in \mathcal{S}, \end{cases} \end{aligned}$$

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 $F = \{1, \ldots, n\}$, clientes $C = \{1, \ldots, m\}$, 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.

$$\text{minimize} \quad \sum_{i \in F} f_i y_i + \sum_{ij \in E} c_{ij} x_{ij}$$

$$\text{sujeito a} \quad \begin{cases} \sum_{ij \in E} x_{ij} = 1 & \forall j \in C, \\ x_{ij} \leq y_i & \forall ij \in E, \\ y_i \in \{0,1\} & \forall i \in F \\ x_{ij} \in \{0,1\} & \forall i \in F \text{ e } j \in C. \end{cases}$$

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 Floresta de Steiner

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

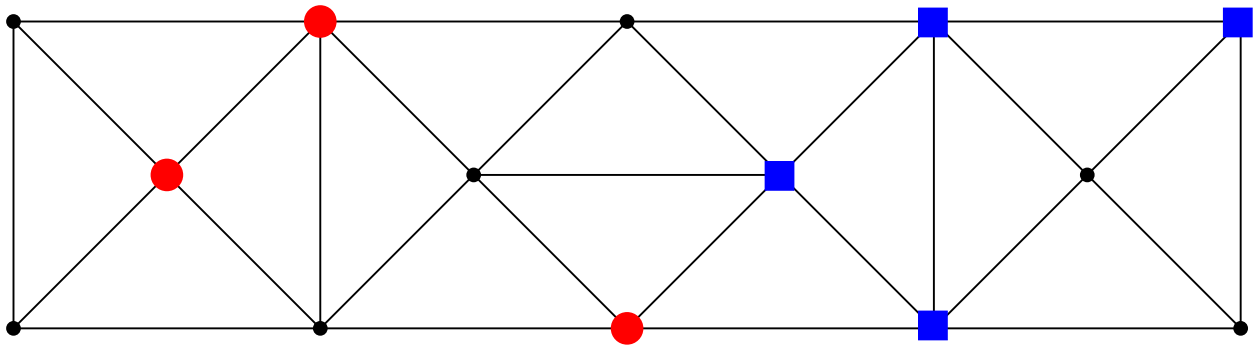
Def.: *Uma $\mathcal{R}$-floresta de G é qualquer floresta geradora F de G tal que todo elemento de $\mathcal{R}$ está contido em algum componente de F.*

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 $c(F)$.

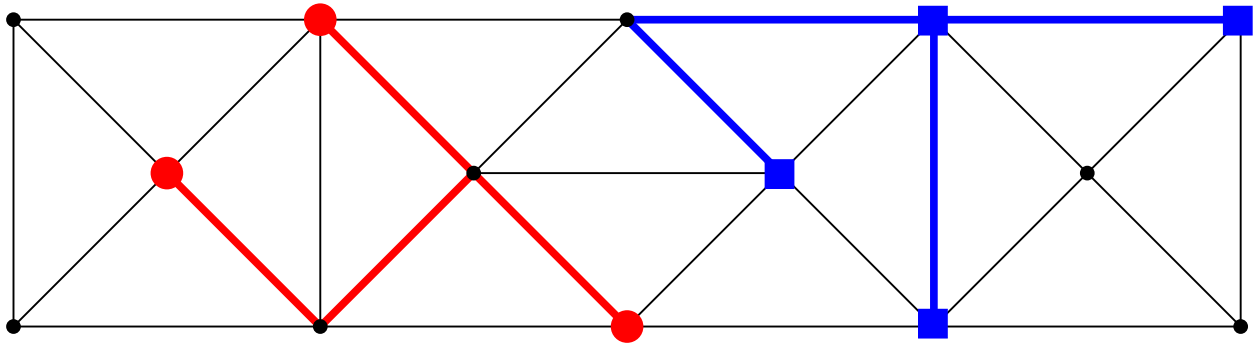
Aplicações:

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

Exemplo: Considere o seguinte grafo com possíveis ligações. Por simplicidade, definimos restrições de conectividade para nós representados pelos mesmos símbolos. Assim, devemos encontrar uma solução de custo mínimo que conecte todos os círculos e que conecte todos os quadrados.



Possível solução



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.

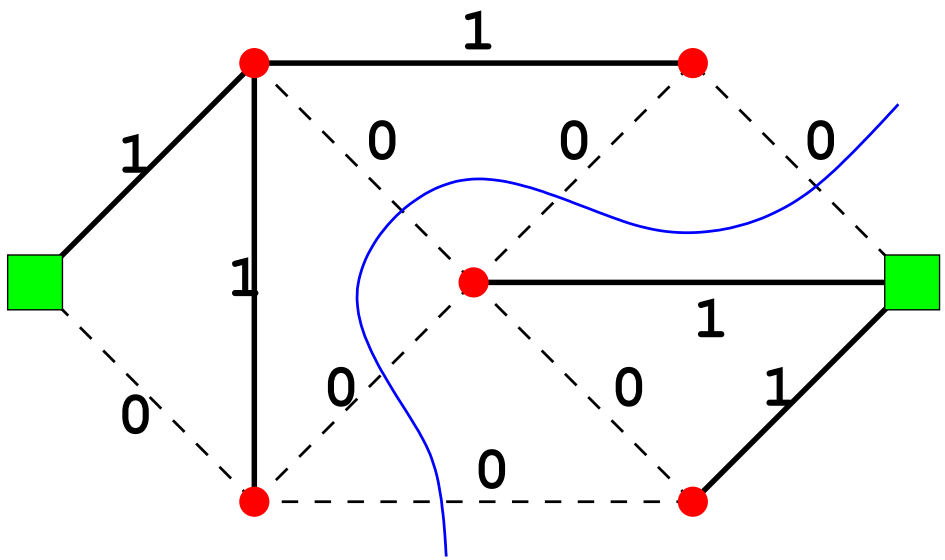
Vamos definir soluções através de variáveis binárias x_{ij} tal que
$x_{ij} = 1$ indica que a aresta ij pertence à solução
$x_{ij} = 0$ indica que a aresta ij não pertence à solução
Note que se S é inválido

- 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 separadore
- pelo menos uma aresta do corte separador deve estar na solução

Exemplo de atribuição para x_e (para cada aresta e) que não é solução viável.



Note que pelo menos uma das arestas que pertence ao corte (aresta que corta linha azul) deve pertencer à solução.

Vc se lembra de como encontrar um corte que separa dois vértices s e t de capacidade mínima ?

Formulação:

$$\text{minimize} \quad \sum_{e \in E} c_e x_e$$

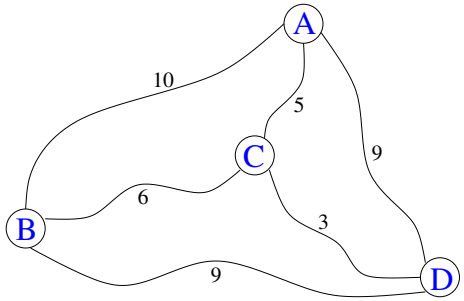
$$\text{sujeito a} \quad \begin{cases} \sum_{e \in \delta(S)} x_e \geq 1 & \forall S \subset V : \\ & S \text{ separa algum conjunto de termi} \\ & \text{(desigualdades de corte)} \\ x_e \in \{0,1\} & \forall e \in E \end{cases}$$

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

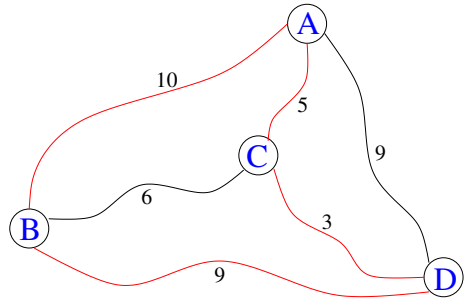
A primeira restrição impõe que todo corte separador deve ter pelo menos uma aresta que pertença à solução.

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.
- Seqüênciamento de DNA (Genoma).
- Outras: `http://www.math.princeton.edu/tsp/apps`
- D.S. Johnson: TSP Chalenging: `www.research.att.com/~dsj/cht`
- CONCORDE: Applegate, Bixby, Chvátal, Cook'01: Branch & cut para TSP

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

$$\text{minimize} \quad \sum_{e \in E} c_e x_e$$

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

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:

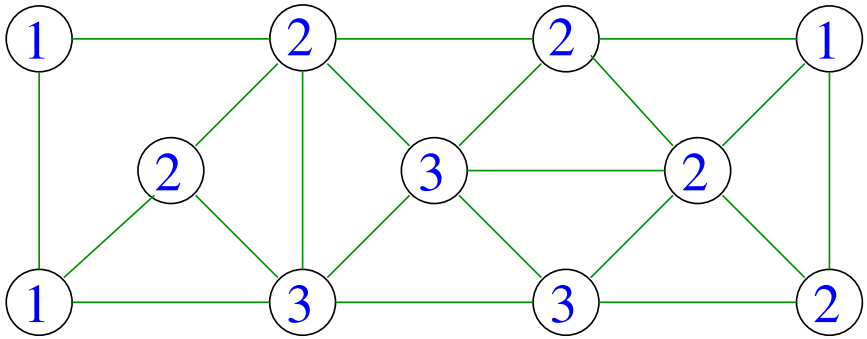
1. A formulação que fizemos do TSP foi para grafos não orientados. Faça uma formulação para o TSP quando as arestas são orientadas (estamos procurando por um circuito hamiltoniano orientado de peso mínimo).
2. Faça uma formulação para encontrar o caminho mínimo de um vértice s e um vértice t, em um grafo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

Survivable Network Design Problem

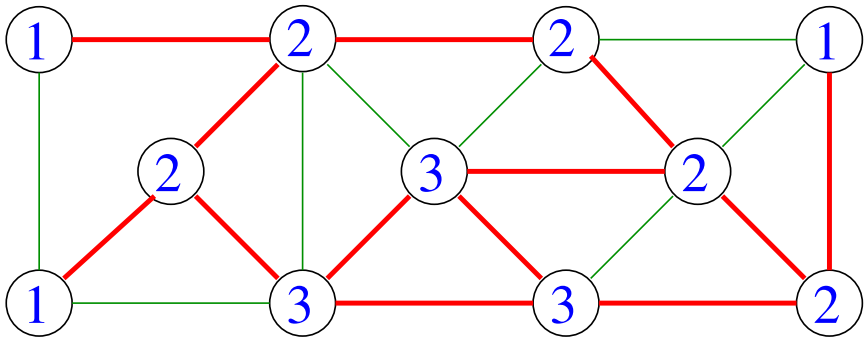
- Neste problema temos vários pontos em uma rede de telecomunicações e alguns pontos que devem ser conectados com requisitos de conectividade.
- Requisitos de conectividade são dados por uma função $f : V \times V \to \mathbb{N}$ que indica o grau de conectividade entre pares de nós.
- Dado dois nós u e v, a função $f(u, v)$ indica a quantidade de rotas alternativas que devem existir entre u e v. Assim, se $f(u, v) = 3$ indica que devem cair no mínimo 3 links na rede para que os nós u e v fiquem desconectados.
- Um subgrafo de G que satisfaz os requisitos de conectividade de f é dita ser uma rede f-conectada de G.
- Para ver mais sobre este problema, veja Mechthild Stoer'92.

Problema SNDP: Dados um grafo $G = (V, E)$, um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e e função $f : V \times V \to \mathbb{N}$ encontrar rede f-conectada em G de custo mínimo.

Exemplo: Para simplificar a função f, considere o seguinte grafo onde f é dada como: $f(u, v) = \min\{label(u), label(v)\}$.
Se o vértice u tem label 3 e o vértice v tem label 2, então $f(u, v) = 2$.



Eis uma solução que satisfaz os requisitos de conectividade da função f.



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
Estratégia é a mesma do Problema da Floresta de Steiner.
Usaremos desigualdades de corte para garantir f-conectividade.

Formulação:

$$\text{minimize} \sum_{e \in E} c_e x_e$$

$$\text{sujeito a} \begin{cases} \sum_{e \in \delta(S)} x_e \geq K_S & \forall S \subset V, K_S = \max\{f(u,v) : u \in S,\ v \in V \setminus S\} \\ & \text{(desigualdades de corte)} \\ x_e \in \{0,1\} & \forall e \in E \end{cases}$$

As desigualdades de corte garantem que toda solução inteira deve ter pelo menos os requisitos de conectividade necessários.

Truques de modelagem

- **Desigualdades excludentes**

  Deve valer apenas uma entre duas desigualdades:
  ou $a'x \leq \alpha'$ ou $a''x \leq \alpha''$ devem valer, não ambas.

  Use nova variável binária Δ com valor 0 se vale a primeira desigualdade e 1 caso a segunda deva ocorrer.

$$\begin{array}{rrcl} a'x & -M\cdot\Delta & \leq & \alpha' \\ a''x & -M\cdot(1-\Delta) & \leq & \alpha'' \\ & \Delta & \in & \{0,1\} \end{array}$$

  onde M é um termo suficientemente grande.

Obs.: Programas que apresentam valores de M grande podem provocar soluções "muito fracionárias".

- **Alternativas no lado direito de igualdades**
  Se deve valer a igualdade $ax = \alpha$ onde $\alpha \in \{\alpha_1, \ldots, \alpha_k\}$, usamos novas variáveis binárias $\Delta_1, \ldots, \Delta_k$ onde $\Delta_i = 1$ se e somente se a igualdade satisfeita é $ax = \alpha_i$.

$$ax = \sum_{i=1}^{k} \alpha_i \Delta_i \qquad \sum_{i=1}^{k} \Delta_i = 1$$
$$\Delta_i \in \{0, 1\}$$

- **Penalidades em desigualdades**
  As vezes permitimos que uma desigualdade $ax \leq \alpha$ seja violada, mas quanto maior a violação, penalizamos com fator P:

$$\text{minimize } \ldots + yP$$

$$ax \leq \alpha + y$$

$$y \geq 0$$

Formulações com variáveis inteiras

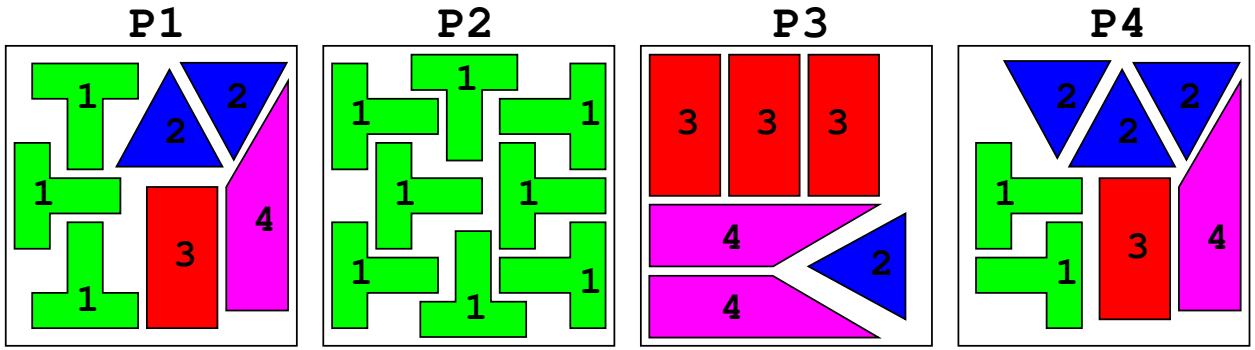
Problema de Empacotamento

Uma empresa deve vender objetos $O = \{1, 2, \ldots, m\}$, cada objeto i com demanda d_i que devem ser recortados a partir de placas que devem ter uma configuração previamente estabelecida.

Há k configurações possíveis, a config. j tem a_{ij} items 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 em 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_i.
- 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:

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

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

Formulação:

$$
\begin{array}{ll}
\text{minimize} & x_1 + x_2 + \cdots + x_k \\
\text{sujeito a} & \left\{ \begin{array}{lcl}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1k}x_k & \geq & d_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2k}x_k & \geq & d_2 \\
\vdots & \vdots & \vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nk}x_k & \geq & d_n \\
x_i \in \mathbb{Z}^* &&
\end{array} \right.
\end{array}
$$

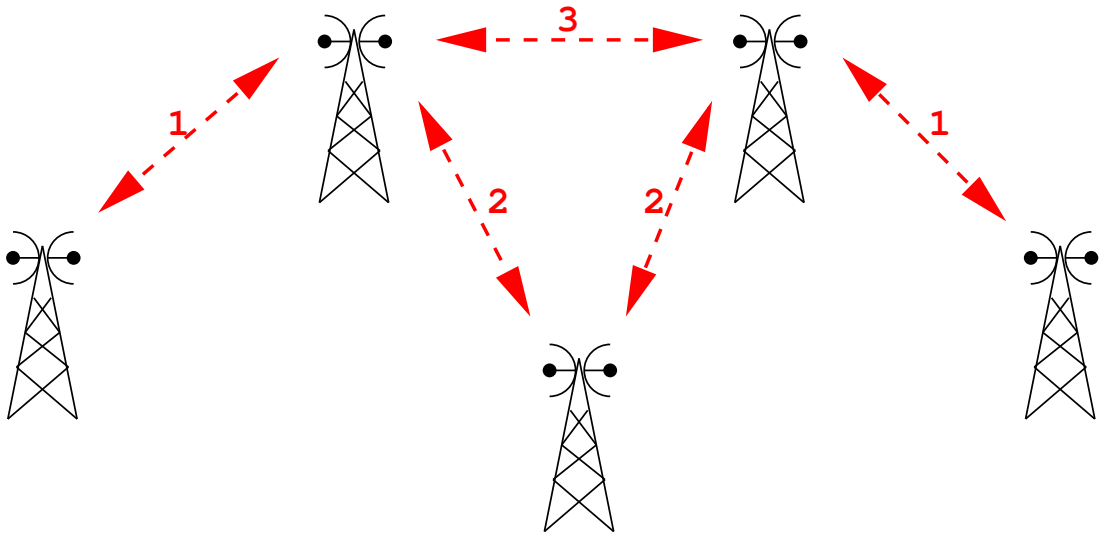
Outras aplicações: Corte de barras, alocação de comerciais de TV, alocação em páginas de memória, corte de placas (vidro, madeira, chapas, tecido, espuma, etc), empacotamento em contêineres, etc.

Problema ATRIBUIÇÃO DE FREQUÊNCIAS *(Radio link frequency assignment problem)*:

São dados conjunto de antenas $A = \{a_1, a_2, \ldots, a_n\}$, conjunto de frequências $F = \{1, \ldots, K\}$, K é uma variável, e uma função distância $d : A \times A \to \mathbb{N}$.

Uma atribuição de frequências para as antenas é uma função $f : A \to F$ tal que a distância das frequências atribuídas para as antenas a_i e a_j é pelo menos $d(a_i, a_j)$.

O objetivo é encontrar uma atribuição de frequências viável que minimize o valor de K.



Formulação:

- Variáveis f_i indicando a frequência que iremos atribuir à antena i;
- todas as frequências devem ocorrer no intervalo $\{1, \ldots, K\}$:

$$1 \leq f_i \leq K, \quad \forall i \in A.$$

- O objetivo é minimizar K.
- A atribuição deve satisfazer a função de distância:

$$|f_i - f_j| \geq d(i,j), \qquad i \in A \quad e \quad j \in A$$

Ops, módulo não é restrição linear. Vamos transformar para linear.
Usaremos variável binária Δ_{ij} para indicar as alternativas do módulo:

- se $\Delta_{ij} = 0$ então deve ocorrer $f_i - f_j \geq d(i,j)$

e

- se $\Delta_{ij} = 1$ então deve ocorrer $f_j - f_i \geq d(i,j)$.

Podemos trocar a restrição não linear

$$|f_i - f_j| \geq d(i,j)$$

pelas seguintes restrições lineares

$$\begin{array}{rrcl} f_i - f_j & +M\cdot\Delta_{ij} & \geq & d(i,j) \\ f_j - f_i & +M\cdot(1-\Delta_{ij}) & \geq & d(i,j) \\ & \Delta_{ij} & \in & \{0,1\} \end{array}$$

onde M é um número suficientemente grande.
Se $\Delta_{ij} = 0$, a primeira restrição nos dá

$$f_i - f_j \geq d(i,j) \text{ e consequentemente } f_i \geq f_j$$

a segunda restrição é sempre válida:

$$f_j - f_i + M \geq d(i,j)$$

o que é sempre verdade se M for suficientemente grande.
Análise análoga pode ser feita quando $\Delta_{ij} = 1$.

Obtemos a seguinte formulação inteira:

$$
\begin{array}{ll}
\text{minimize} & K \\
\text{sujeito a} & \left\{
\begin{array}{rcll}
f_i - f_j + M \cdot \Delta_{ij} & \geq & d(i,j) & \forall i \neq j, i \in A, j \in A \\
f_j - f_i + M \cdot (1 - \Delta_{ij}) & \geq & d(i,j) & \forall i \neq j, i \in A, j \in A \\
\Delta_{ij} & \in & \{0,1\} & \forall i \neq j, i \in A, j \in A \\
f_i & \geq & 1 & \forall i \in A \\
f_i & \leq & K & \forall i \in A \\
f_i & \in & \mathbb{Z} & \forall i \in A \\
K & \in & \mathbb{Z} &
\end{array}
\right.
\end{array}
$$

Obs.: A formulação deste problema foi simplificada. Algumas vezes há custos (interferência) envolvida na atribuição que dependem da distância entre frequências para cada par de antenas. Algumas antenas tem restrições de frequências, por estarem próximas a antenas que estão fora do controle da atribuição (e.g., pertencem a outras empresas). Para ver mais sobre este problema, veja Borndörfer, Eisenblätter, Grötschel, Martin'96 (ZIB).

Problema ESCALONAMENTO COM PRECEDÊNCIA: Dados uma lista de tarefas $L = \{1, \ldots, n\}$, cada uma com tempo inteiro p_i, uma ordem parcial $\prec$ entre tarefas e uma função de prioridade $w : L \to \mathbb{R}^+$,

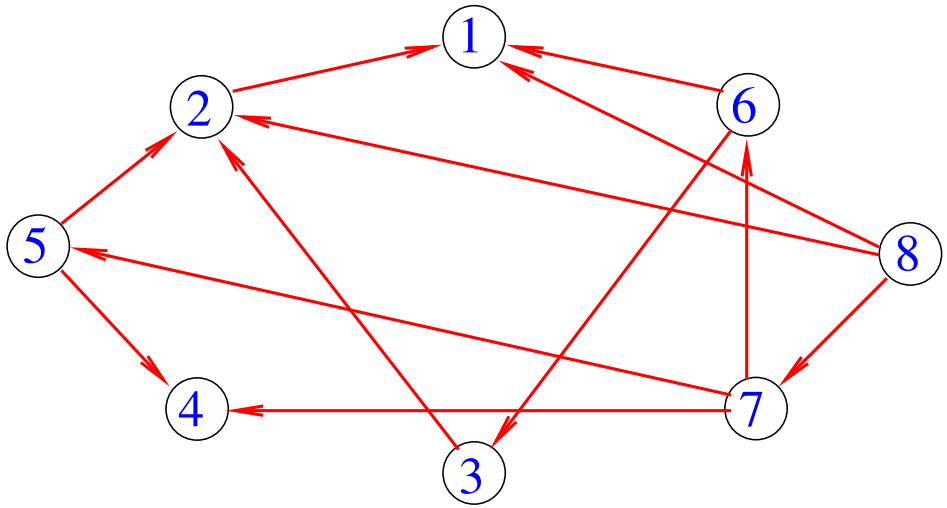
encontrar um escalonamento de L, obedecendo precedência, em um processador tal que $\sum_i w_i f_i$ é mínimo, onde f_i é o tempo que a tarefa i é finalizada.

Um escalonamento obedece precedências se para todo par de tarefas (i, j) para o qual vale $i \prec j$ então a tarefa i termina antes da tarefa j começar. A execução das tarefas não podem ser feitas por partes.

Teorema: *O* ESCALONAMENTO COM PRECEDÊNCIA *é um problema NP-difícil.*

Problemas de escalonamento em geral: Escalonamento de processos em máquinas paralelas, escalonamento de pessoal em turnos de trabalho, etc

Exemplo de precedência através de um grafo orientado:
($i \leftarrow j$ significa que i deve ser executado antes de j)



Ex

de escalonamentos viáveis:

1 - 2 - 3 - 4 - 5 - 6 - 7 - 8
4 - 1 - 2 - 5 - 3 - 6 - 7 - 8
1 - 4 - 2 - 3 - 6 - 5 - 7 - 8

Vamos ver duas formulações para este problema

Formulação 1:

- s_i variável inteira indicando o tempo de início da tarefa i
- f_i variável inteira indicando o tempo que a tarefa i é finalizada
- Δ_{ij} variável binária indicando se a tarefa i vem antes de j
  Δ_{ij} tem valor 1 se tarefa i vem antes de j, 0 caso contrário.

Função objetivo: Minimizar o tempo de finalização com prioridades:

$$\text{minimize } \sum_{i=1}^{n} w_i f_i$$

Tempos não negativos:

$$s_i \geq 0, \;\; f_i \geq 0, \text{ para } 1 \leq i \leq n$$

Término é tempo de início + tempo de processamento: $s_i + p_i = f_i$.

$$f_i - s_i = p_i$$

Formulação 1:

Quando $i \prec j$

$$f_i \leq s_j$$

Quando não há restrição entre i e j,

$$\begin{array}{lrcl} f_i & -M(1-\Delta_{ij}) & \leq & s_j \\ f_j & -M\Delta_{ij} & \leq & s_i \end{array}$$

onde M é um valor suficientemente grande.

Formulação 1:

$$\text{minimize} \quad \sum_{i=1}^{n} w_i f_i$$

$$\text{sujeito a} \begin{cases} -s_i + f_i & & = & p_i & \forall i \in [n] \\ -s_i + f_i - M(1-\Delta_{ij}) & & \leq & 0 & \forall i < j : \ ij \notin \mathcal{P} \text{ e } ji \notin \mathcal{P} \\ -s_i + f_j - M\Delta_{ij} & & \leq & 0 & \forall i < j : \ ij \notin \mathcal{P} \text{ e } ji \notin \mathcal{P} \\ -s_j + f_i & & \leq & 0 & \forall ij : i \prec j \\ & s_i & \geq & 0 & \forall i \in [n] \\ & \Delta_{ij} & \in & \{0,1\} & \forall ij \in [n] \times [n] \end{cases}$$

onde $[n] = \{1, \ldots, n\}$ e M é um valor suficientemente grande.

Formulação 2:

Vamos supor que o tempo é medido em unidades inteiras.
e é possível terminar todas as tarefas em um tempo máximo T.

- f_j tempo que a tarefa j terminou.
- x_{jt} vale 1 se a tarefa j termina no tempo t, e vale 0 caso contrário

Uma tarefa só pode terminar em um determinado tempo:

$$\sum_{t=1}^{T} x_{jt} = 1, \text{ para } 1 \leq j \leq n$$

$x_{jt} = 1$ se e somente se $f_j = t$:

$$f_j = \sum_{t=1}^{T} t\, x_{jt}, \text{ para } 1 \leq j \leq n$$

Função objetivo: Minimizar tempos de finalização com prioridades:

$$\text{minimize } \sum_{j=1}^{n} w_j f_j$$

Nenhuma tarefa pode terminar antes do seu tempo de processamento.

$$x_{jt} = 0, \text{ para } t < p_j \text{ e } 1 \leq j \leq n$$

Obs.: Uma estratégia nesta formulação é considerar o seguinte:

$$\sum_{s=t_i}^{t_f} x_{js} = 1$$

equivale a dizer que j terminou o processamento no tempo $[t_i, \ldots, t_f]$.
Se $j \prec k$ então j deve terminar pelo menos p_k unidades de tempo antes de k terminar.

$$\sum_{s=1}^{t} x_{js} \geq \sum_{s=1}^{t+p_k} x_{ks}, \text{ para } j \prec k \text{ e } t = 1, \ldots, T - p_k$$

Em cada tempo t, deve haver apenas um job sendo executado

$$\sum_{j=1}^{n} \sum_{s=t}^{\min\{t+p_j-1, T\}} x_{js} \leq 1, \text{ para } t = 1, \ldots, T$$

Note que se $\sum_{s=t}^{\min\{t+p_j-1, T\}} x_{js} = 1$ significa que o tempo t está sendo usado pela tarefa j.

$$\text{minimize} \quad \sum_{j=1}^{n} w_j f_j$$

$$\text{sujeito a} \begin{cases} \displaystyle\sum_{t=1}^{T} x_{jt} = 1 & \forall 1 \le j \le n \\ f_j - \displaystyle\sum_{t=1}^{T} t\, x_{jt} = 0 & \forall 1 \le j \le n \\ x_{jt} = 0 & \forall t < p_j \text{ e } 1 \le j \le n \\ \displaystyle\sum_{s=1}^{t} x_{js} - \sum_{s=1}^{t+p_k} x_{ks} \ge 0 & \forall ij : j \prec k \text{ e } t = 1, \ldots, T - p_k \\ \displaystyle\sum_{j=1}^{n} \sum_{s=t}^{\min\{t+p_j-1, T\}} x_{js} \le 1 & t = 1, \ldots, T \\ x_{jt} \in \{0, 1\} & \forall 1 \le t \le T \text{ e } 1 \le j \le n \end{cases}$$

Apesar de mais variáveis e mais complicada, esta formulação tem apresentado melhores resultados que a primeira formulação

Exercícios:

- Em uma formulação linear inteira, uma variável x pode assumir valor 0 ou valores acima de uma constante K. Como você restringe para que isto ocorra ?
- Seu programa linear inteiro deve ter restrições $ax \leq \alpha_1, \ \ldots \ ax \leq \alpha_m$, mas no máximo t, $1 \leq t \leq m$ restrições devem realmente ser satisfeitas. Como você pode formular isto ?
- Faça uma formulação alternativa para o problema de atribuição de frequências, usando variáveis binárias x_{if} com valor 1 se e somente se a antena i recebeu a frequência f (suponha que a maior frequência é K).

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., $P_I := conv(I)$).

Queremos otimizar em P_I, mas muitas vezes temos facilmente 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 empacotame

Formulação em PLI:

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

Relaxação do PLI:

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

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.

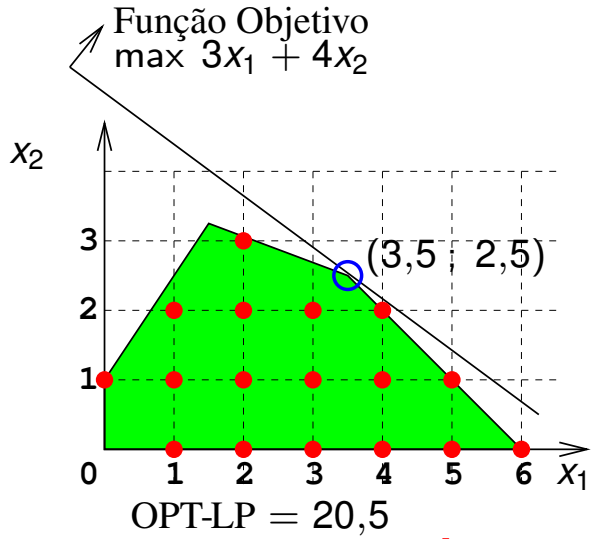
Branch & Bound e Programação Linear Inteira

- Modelar o problema como problema de programação linear inteira
- Relaxar a formulação para programação linear
- Enumerar o espaço de soluções através do método branch & bound
  - Cada nó da árvore representa um programa linear
  - Cada programa linear (nó) é ramificado enquanto houver valores fracionários em sua solução

Considere o programa linear inteiro:

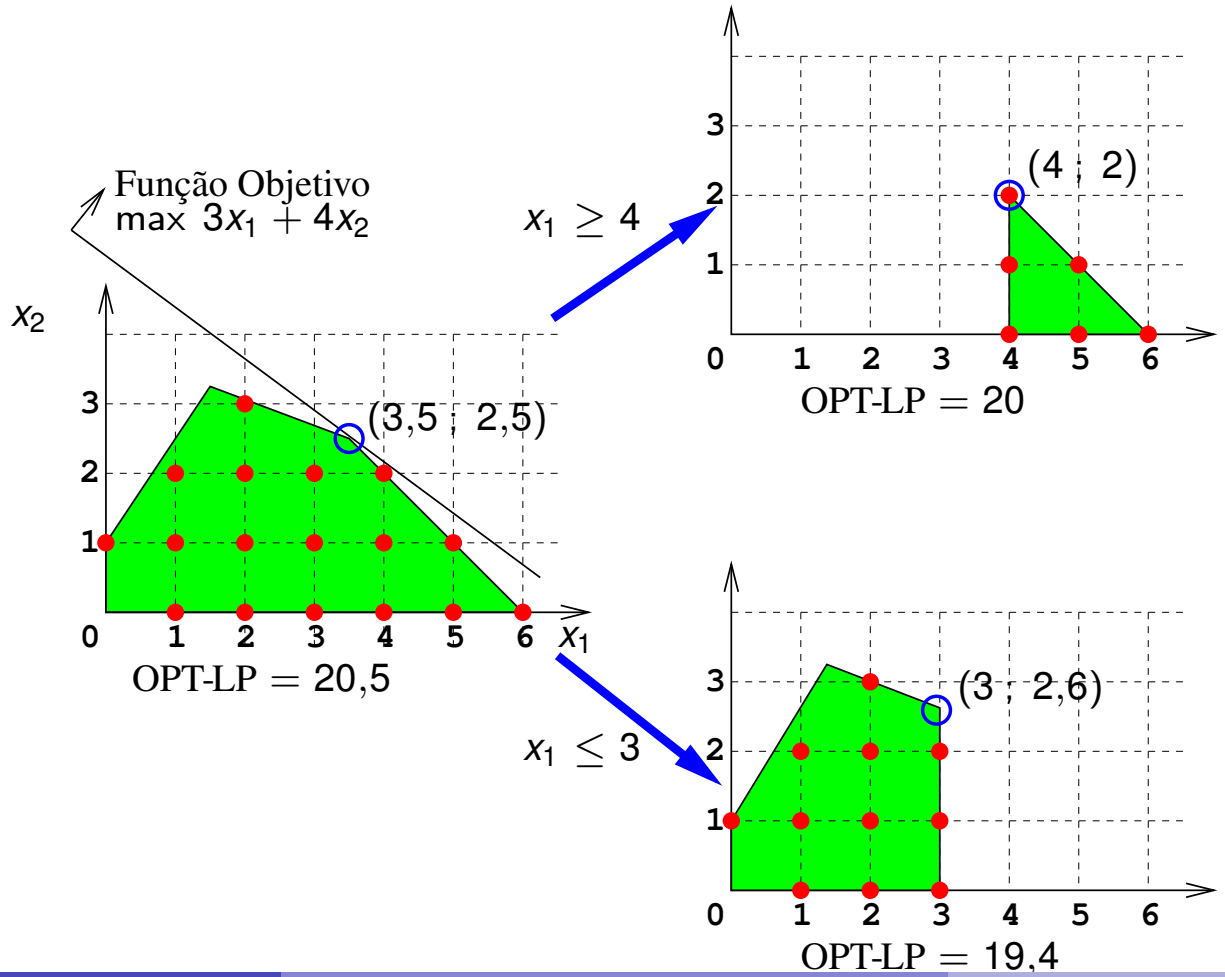
$$\begin{array}{ll} \text{maximize} & 3x_1 + 4x_2 \\ \text{sujeito a} & \left\{ \begin{array}{rrcl} -3x_1 & +2x_2 & \leq & 2 \\ x_1 & +3x_2 & \leq & 11 \\ x_1 & +x_2 & \leq & 6 \\ x_1 \geq 0, & x_2 \geq 0 & & \\ x_i \in \mathbb{Z}^* & & & \end{array} \right. \end{array}$$

Representação gráfica do programa relaxado e os pontos inteiros:

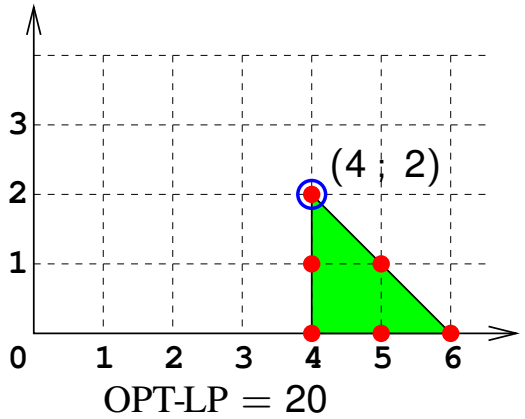


Vamos resolver o programa inteiro através do Branch (& Bound)

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:

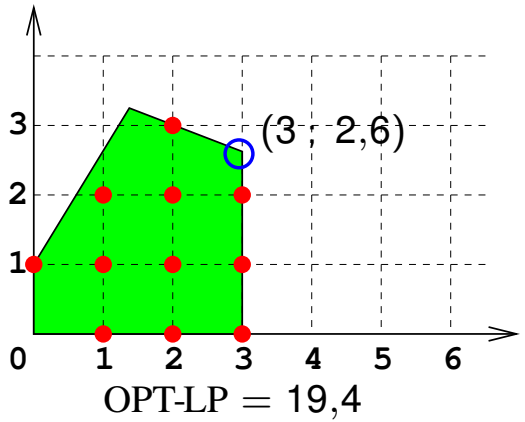


Um dos programas já tem solução inteira e não precisamos continuar sua ramificação:



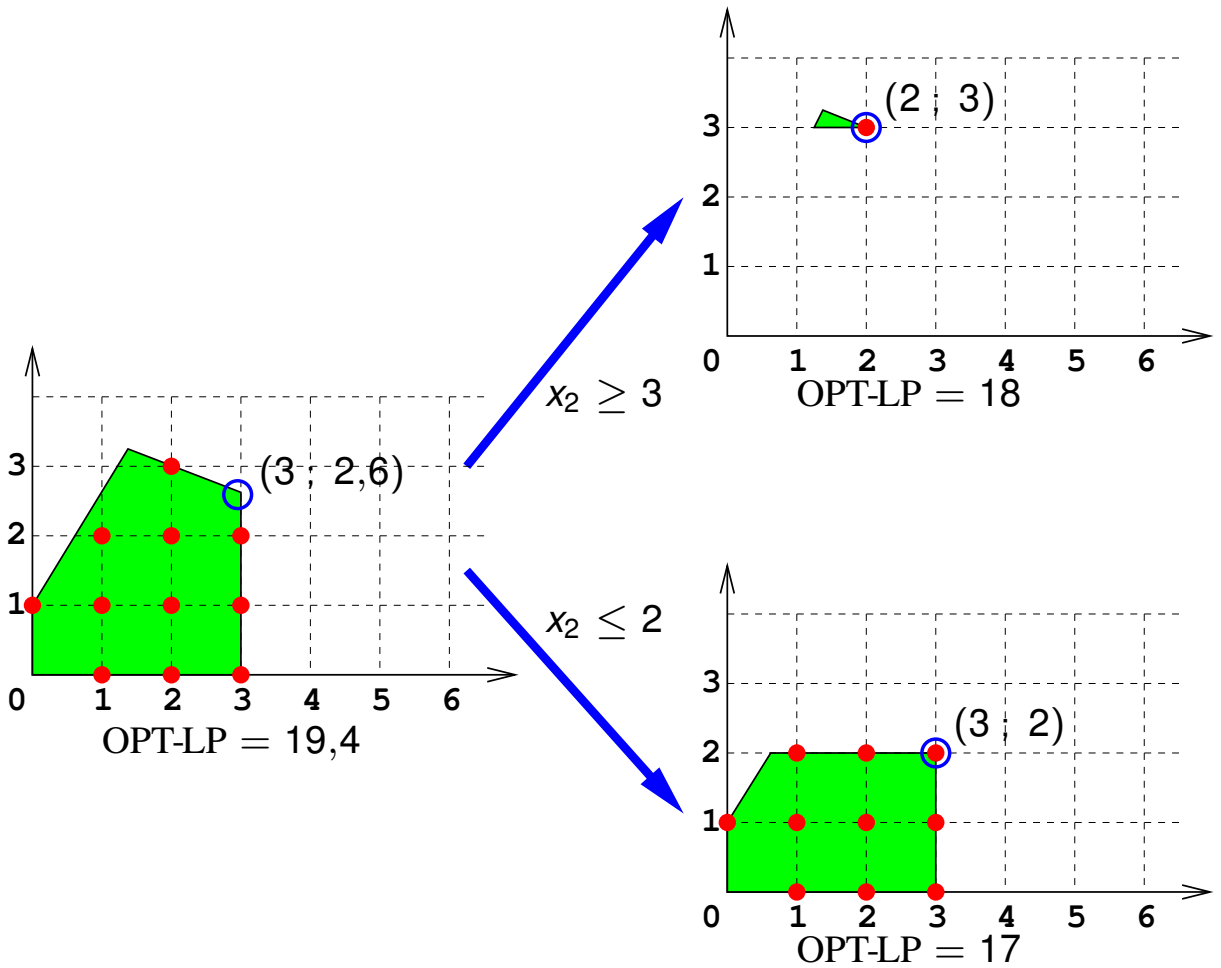
OPT-LP = 20

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



OPT-LP = 19,4

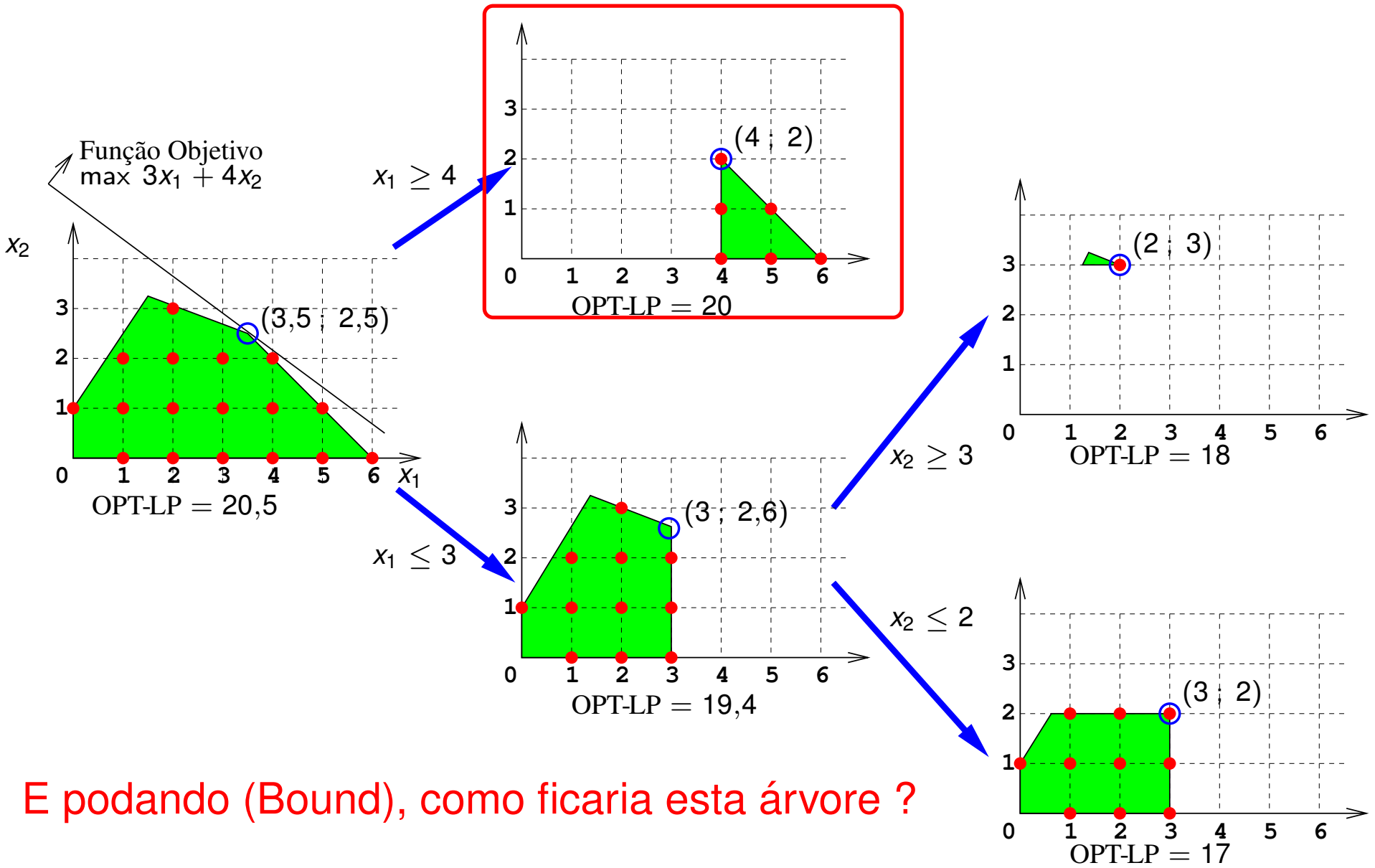
Ramificando em x_2, para os casos $x_2 \leq 2$ e $x_2 \geq 3$, obtemos os seguintes programas



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

Árvore completa de Branch (sem Bound).

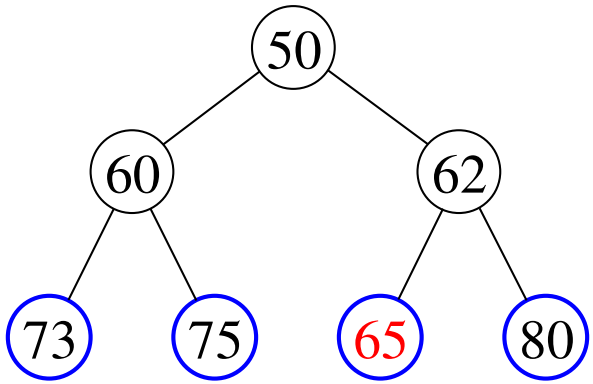
E a solução ótima do problema



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

Estratégias comuns envolvidas na árvore de B&B

- **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)
- **Usando apenas um programa linear:**
  Na maioria das vezes usamos apenas um programa linear.
  Resolva o programa linear associado a um nó, acertando previamente os limitantes das variáveis daquele nó.
  A resolução de programas lineares anteriormente resolvidos com pequenas modificações é em geral mais rápida.

Estratégias comuns envolvidas na árvore de B&B

- **Representando a árvore de Branch & Bound:**
  Algumas vezes não precisamos armazenar 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.

BRANCH-BOUND-MINIMIZAÇÃO-SIMPLIFICADO (P^o) % PL relaxado P^o

```
1   LB← solução ótima de P^o (halt se P^0 é inviável)
2   x* ← Heurística sobre P^o (∅ se não obteve solução viável/não há )
3   UB← val(x*) (onde val(∅) = +∞)
4   Ativos← {P^o}
5   enquanto Ativos≠ ∅ faça
6       escolha P ∈ Ativos
7       Ativos ← Ativos \ {P}
8       se P é viável, % se inviável, ramo é podado
9           x ← solução ótima de P
10          se val(x) > LB então atualiza LB (se necessário)
11          se val(x) < UB % se val(x) ≥ UB , ramo é podado
12              se x é inteiro então
13                  UB ← val(x)
14                  x* ← x
15              senão
16                  crie dois subproblemas P' e P'' a partir de P
17                  Ativos←Ativos ∪ {P' , P''}
18  devolva x*
```

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.
Em alguns casos, é possível resolver (P, c) rapidamente.

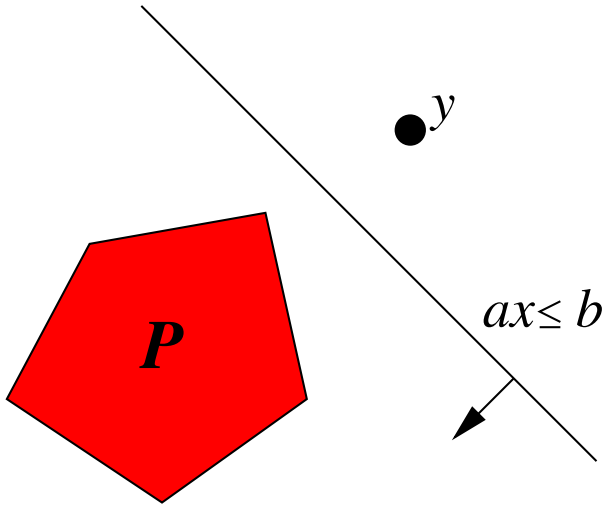
Estratégia:

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 $\times$ 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: *(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 $Q \leftarrow \{$Poliedro inicial$\}$

2 $y \leftarrow$OPT-LP(Q, c)

3 enquanto y pode ser separado de Q faça

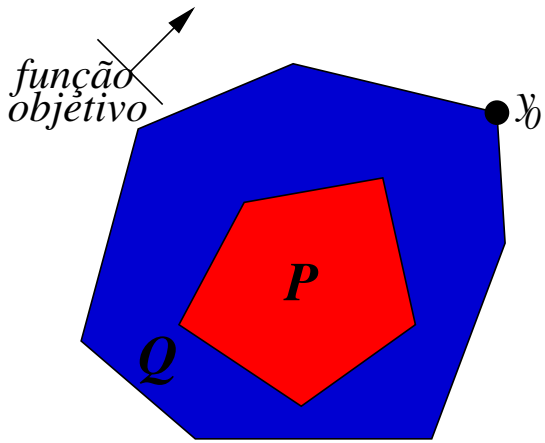
4 seja $ax \leq b$ desigualdade de P que separa y

5 $Q \leftarrow Q \cap \{x : ax \leq b\}$

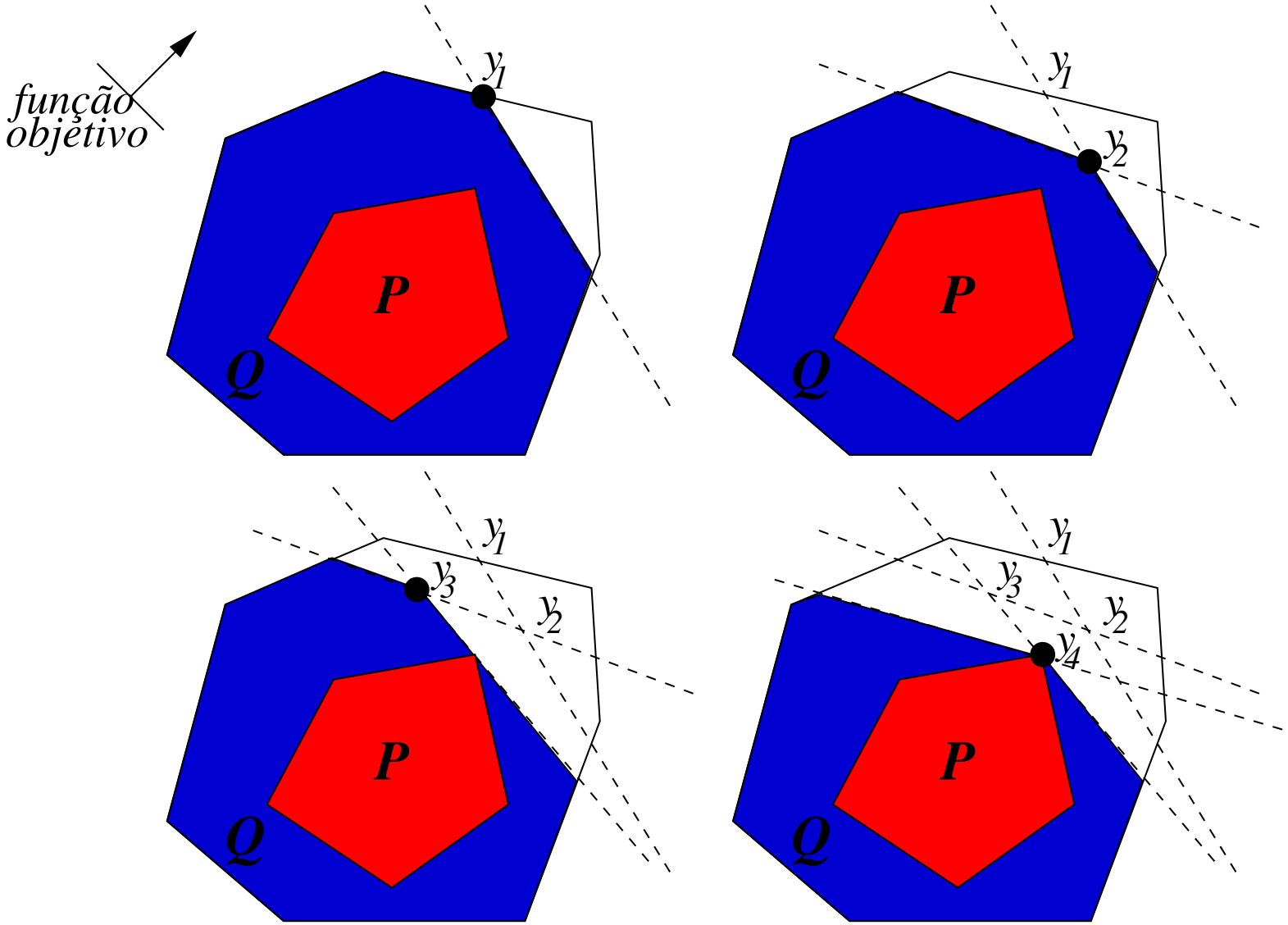
6 $y \leftarrow$OPT-LP(Q, c)

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

8 senão retorne y,



onde OPT-LP(Q, c) é a solução ótima do programa linear (Q, c)



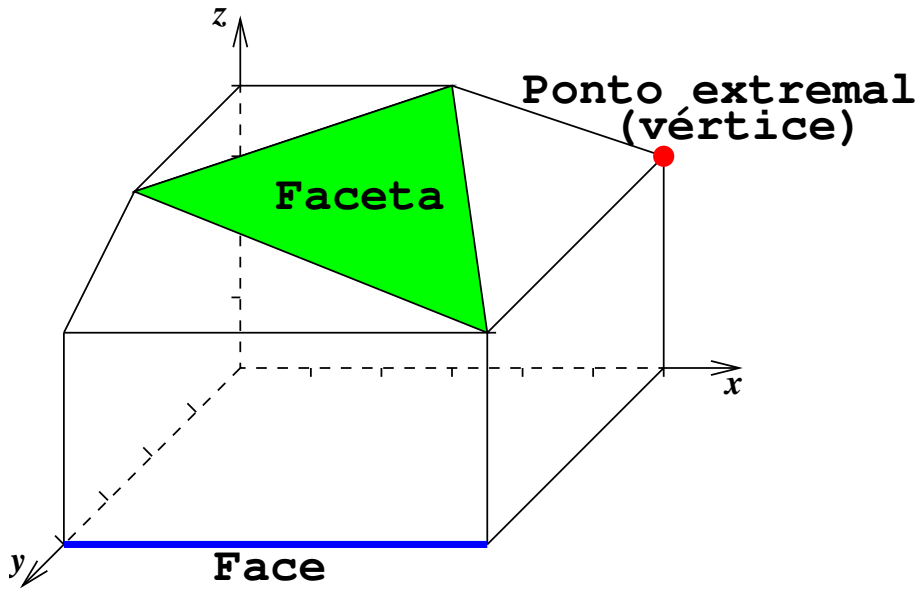
Que tipo de desigualdade são as melhores para serem inseridas ?

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 $F = P \cap \{x : ax \leq b\}$
- 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 $ax \leq \lfloor t \rfloor$

Exercício: Transforme um poliedro descrito por desigualdades e igualdades ($\leq, =, \geq$) em um descrito apenas por desigualdades do tipo $\leq$?

Cortes de Chvátal

Seja P um poliedro

1. Uma desigualdade $ax \leq t$, sendo a um vetor inteiro, pertence ao fecho elementar de P, denotado por $e^1(P)$ se existem desigualdades $d_1 x \leq b_1, \ldots, d_n x \leq b_n$ de P e valores não negativos $\lambda_1, \ldots, \lambda_m$ tal que $\sum_{i=1}^{m} \lambda_i d_i = a$ e $\lfloor \sum_{i=1}^{m} \lambda_i b_i \rfloor \leq t$.
2. Para $k > 1$, definimos $e^k(P) := e^1(P \cup e^{k-1}(P))$.
3. O fecho $c(P)$ é definido como $\cup_{k=1}^{\infty} e^k(P)$.

Teorema: *(Chvátal'73) Se P é um poliedro limitado e I o conjunto de pontos inteiros em P, então conv(I) pode ser obtido após um número finito k de operações do fecho.*

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:

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

Relaxação linear

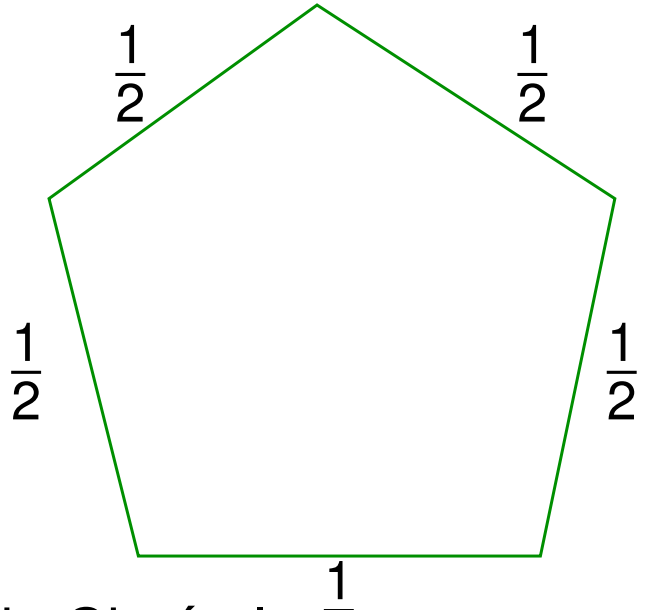
$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} c_e x_e \\ (P) \quad \text{sujeito a} & \begin{cases} \sum_{e \in \delta(v)} x_e \leq 1 & \forall v \in V, \\ x_e \geq 0 & \forall e \in E. \end{cases} \end{array}$$

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:

1. Seja $U \subseteq V$ tal que $|U| \geq 3$ é impar.
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:

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

$$\text{maximize} \quad \sum_{e \in E} c_e x_e$$

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

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.*

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:

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

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 ?

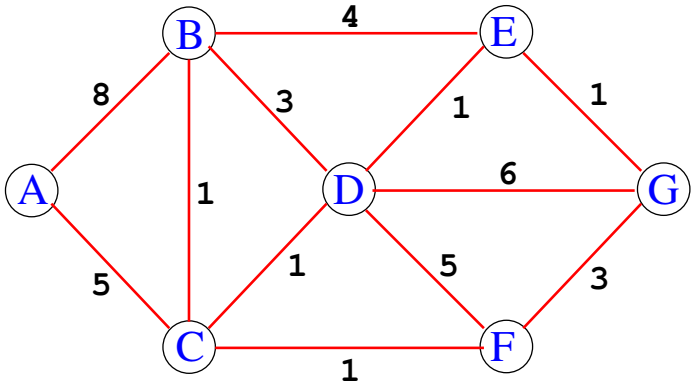
Para testar se dois vértices s e t estão separados por um corte $(S, \overline{S})$ que viola a desigualdade de corte, i.e., está ocorrendo

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

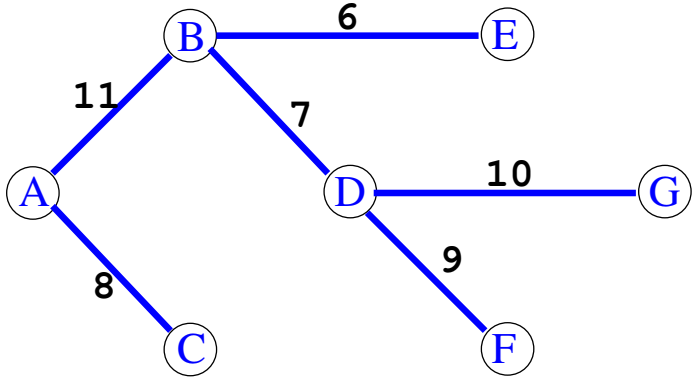
podemos executar o algoritmo para st-corte mínimo.
Note que o algoritmo deve ser executado não para o grafo original, mas onde cada aresta e tem capacidade x_e.
Se o valor do st-corte mínimo for menor que 2, então o corte gerado viola a desigualdade de corte e podemos inseri-la como desigualdade válida.

- 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:

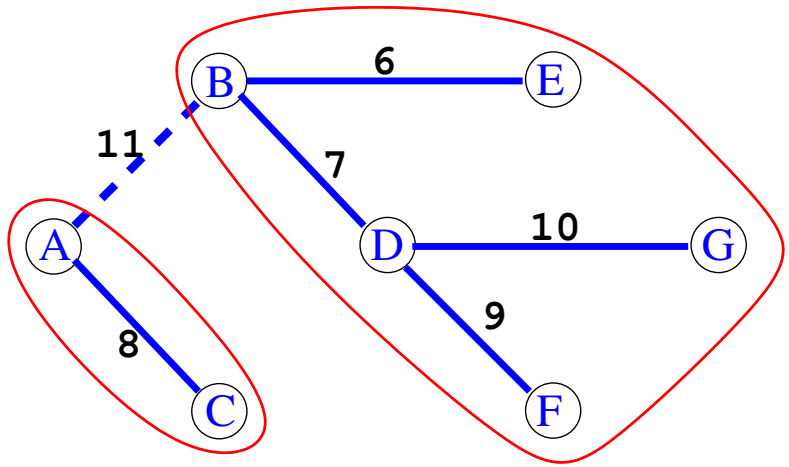


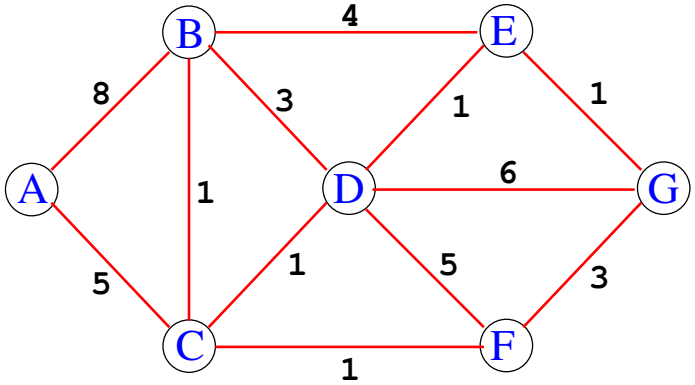
Grafo $G = (V, E)$



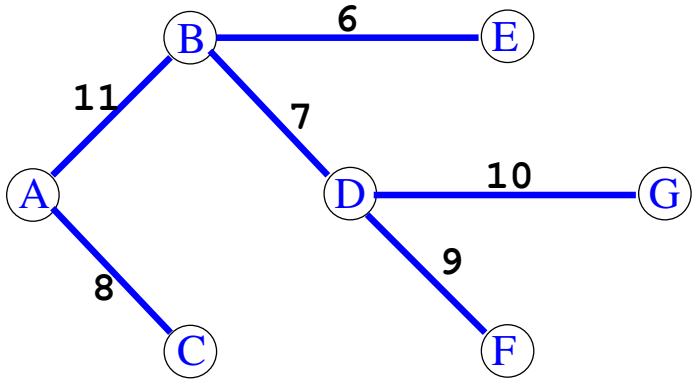
Árvore T de Cortes de Gomory-Hu de G

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



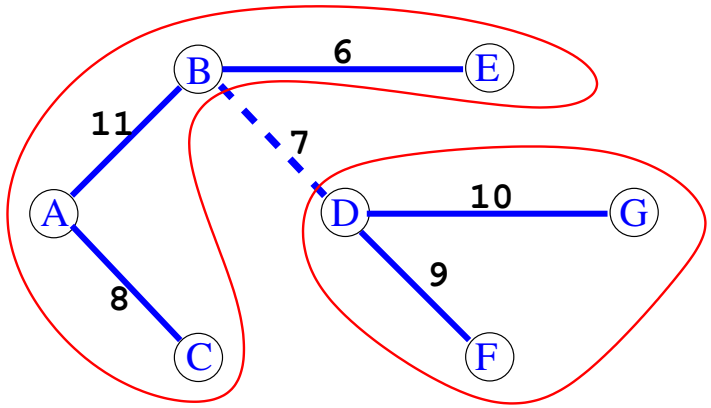


Grafo $G = (V, E)$



Árvore T de Cortes de Gomory-Hu de G

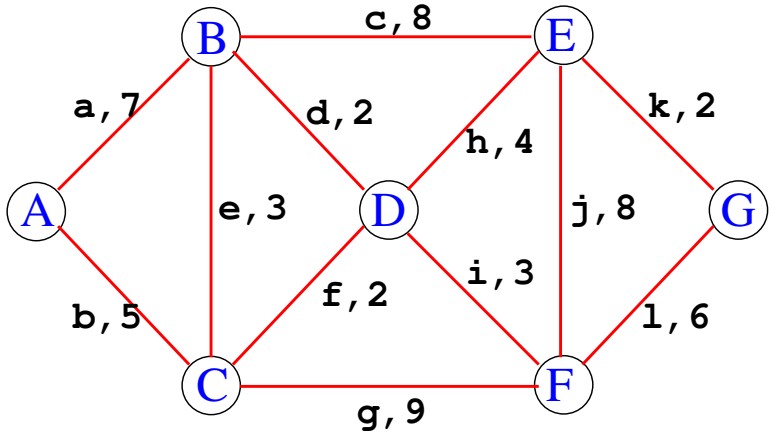
Corte mínimo que separa u e v em G é dado pelas componentes obtidas da remoção da aresta de capacidade mínima no caminho entre u e v em T.



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

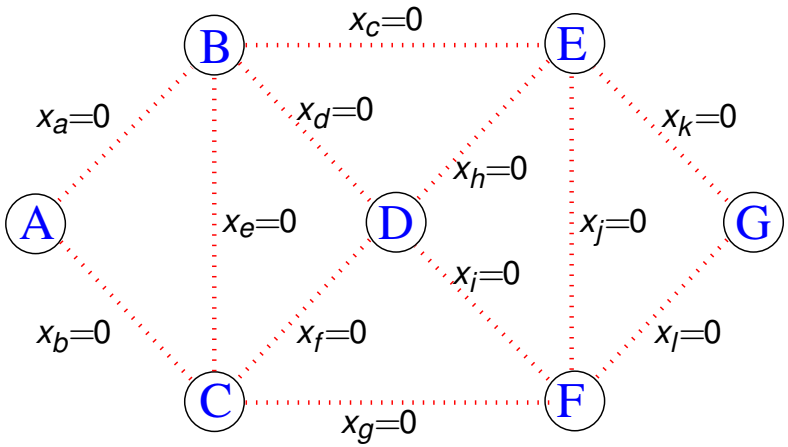
Separação para o TSP

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

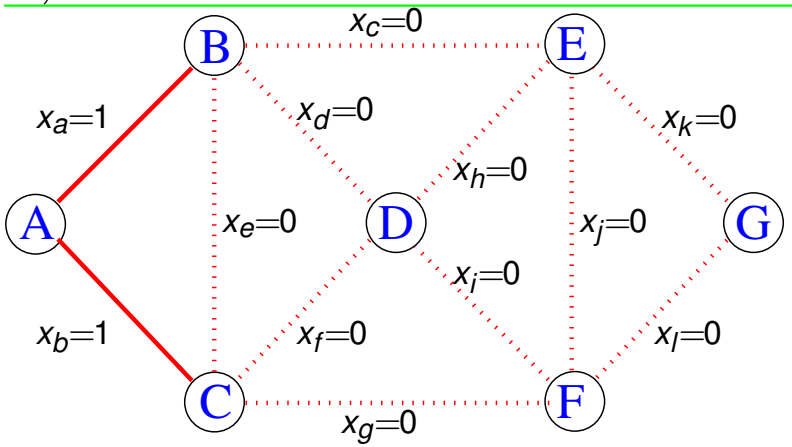
Para não sobrecarregar, em cada passo seguinte apresentaremos apenas os valores obtidos da solução ótima fracionária sem colocar os custos das arestas e seus nomes.



$$\begin{aligned} \min \quad & c_a x_a + c_b x_b + \cdots + c_l x_l \\ \text{s.a.} \quad & \{\ 0 \leq x_a \leq 1, \ldots, 0 \leq x_l \leq 1 \end{aligned}$$

Solução ótima = 0

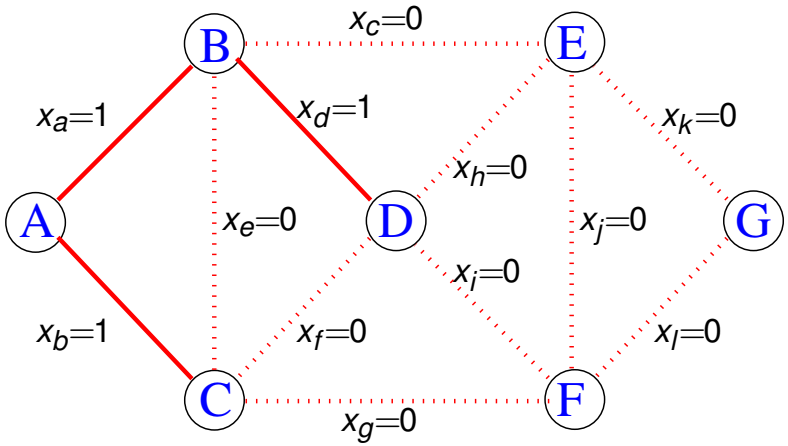
Apenas restrições $0 \leq x_e \leq 1$, $\forall e \in E$



$$\begin{aligned} \min \quad & c_a x_a + c_b x_b + \cdots + c_l x_l \\ \text{s.a.} \quad & \{\ \begin{array}{r} x_a + x_b = 2 \\ 0 \leq x_a \leq 1, \ldots, 0 \leq x_l \leq 1 \end{array} \end{aligned}$$

Solução ótima = 12

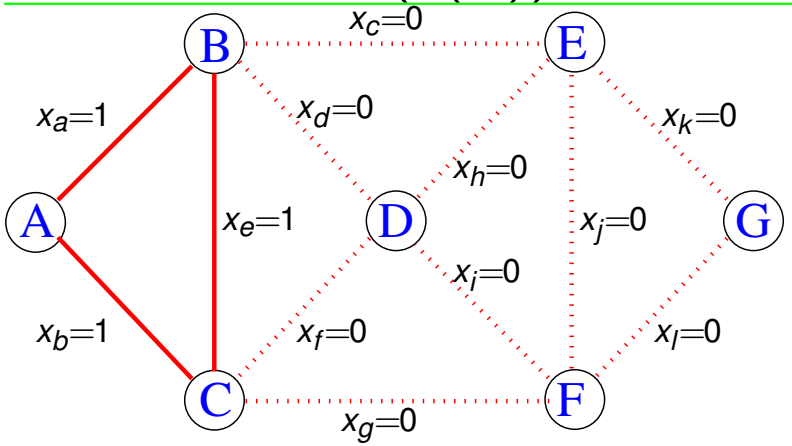
Adicionando $x(\delta(A)) = 2$



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

$$\min \quad c_a x_a + c_b x_b + \cdots + c_l x_l$$
$$\text{s.a.} \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ 0 \le x_a \le 1, \ldots, 0 \le x_l \le 1 \end{cases}$$

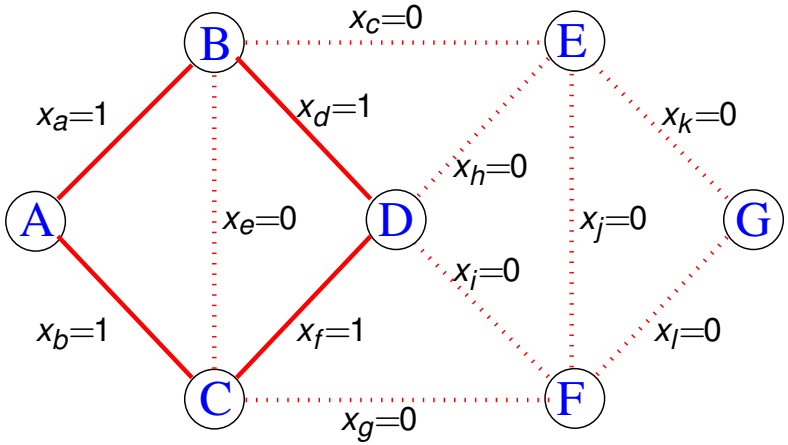
Solução ótima = 14



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

$$\min \quad c_a x_a + c_b x_b + \cdots + c_l x_l$$
$$\text{s.a.} \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ 0 \le x_a \le 1, \ldots, 0 \le x_l \le 1 \end{cases}$$

Solução ótima = 15

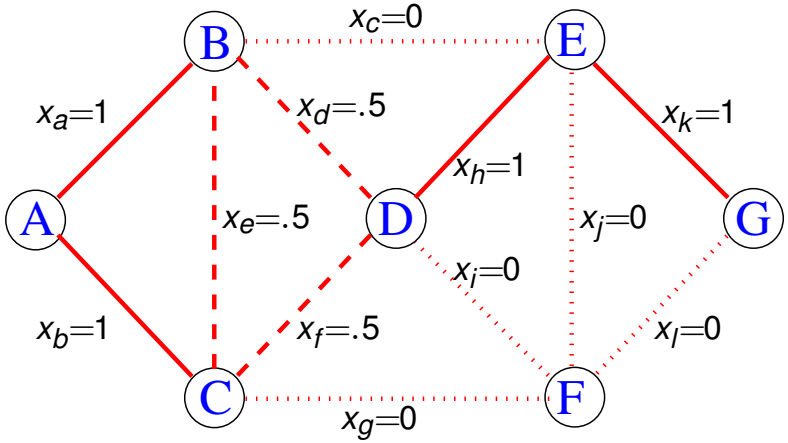


$$\min \quad c_a x_a + c_b x_b + \cdots + c_l x_l$$

$$\text{s.a.} \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ 0 \le x_a \le 1, \ldots, 0 \le x_l \le 1 \end{cases}$$

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

Solução ótima = 16

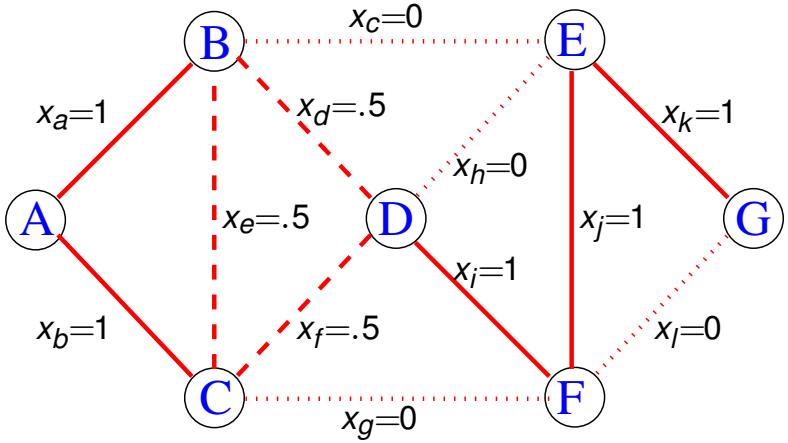


$$\min \quad c_a x_a + c_b x_b + \cdots + c_l x_l$$

$$\text{s.a.} \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ 0 \le x_a \le 1, \ldots, 0 \le x_l \le 1 \end{cases}$$

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

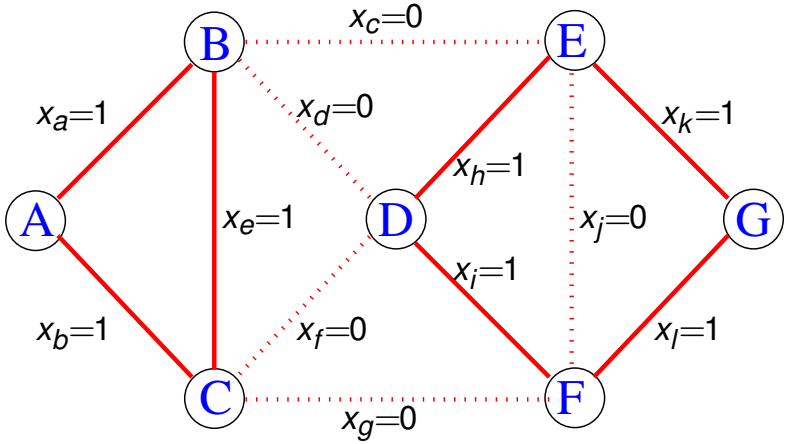
Solução ótima = 21,5



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

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

Solução ótima = 28.5

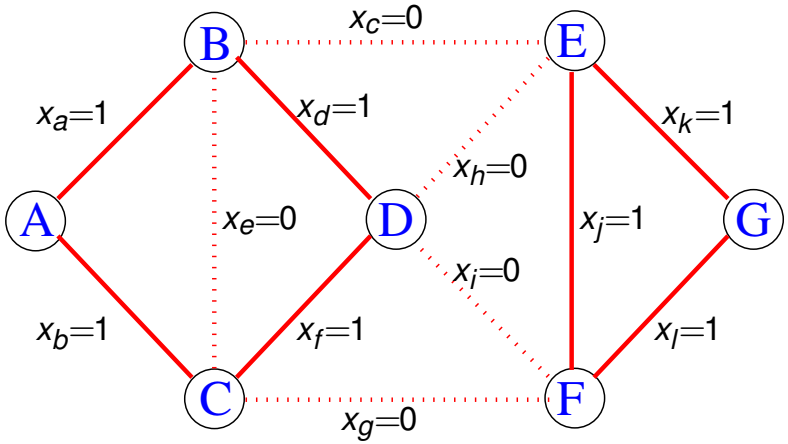


Adicionando $x(\delta(G)) = 2$

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

Solução ótima = 30

Todas as restrições $x(\delta(v)) = 2, \quad \forall v \in V$ foram inseridas.
Em geral, restrições estruturais que ocorrem pouco são inseridas todas de uma vez.
Agora vamos inserir as desigualdades de corte:



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

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

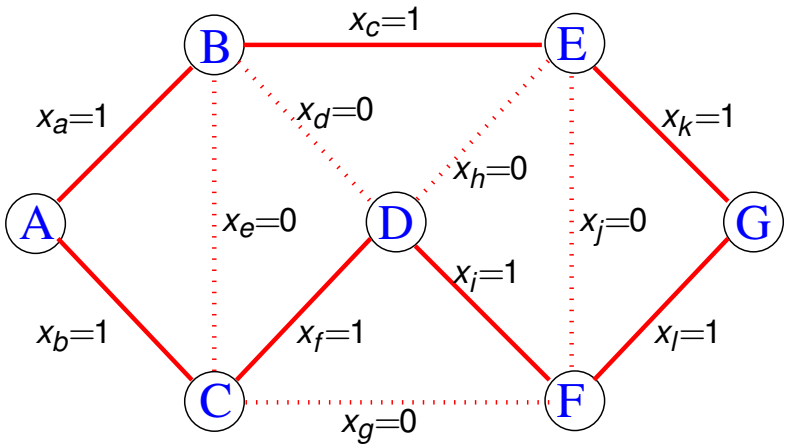
onde $S = \{A, B, C\}$

$$\min \quad c_a x_a + c_b x_b + \cdots + c_l x_l$$

$$\text{s.a.} \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ x_g + x_i + x_j + x_l = 2 \\ x_k + x_l = 2 \\ x_c + x_d + x_f + x_g \geq 2 \end{cases}$$

$$0 \leq x_a \leq 1, \ldots, 0 \leq x_l \leq 1$$

Solução ótima = 32



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

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

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

$$\min \quad c_a x_a + c_b x_b + \cdots + c_l x_l$$

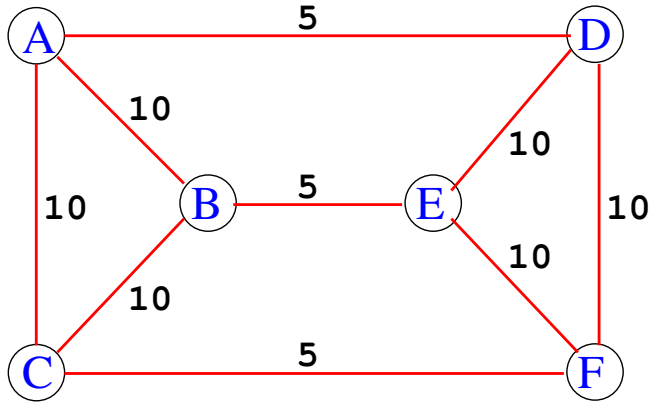
$$\text{s.a.} \begin{cases} x_a + x_b = 2 \\ x_a + x_c + x_d + x_e = 2 \\ x_b + x_e + x_f + x_g = 2 \\ x_d + x_f + x_h + x_i = 2 \\ x_c + x_h + x_j + x_k = 2 \\ x_g + x_i + x_j + x_l = 2 \\ x_k + x_l = 2 \\ x_c + x_d + x_f + x_g \geq 2 \\ x_c + x_g + x_h + x_i \geq 2 \\ 0 \leq x_a \leq 1, \ldots, 0 \leq x_l \leq 1 \end{cases}$$

Solução ótima = 33

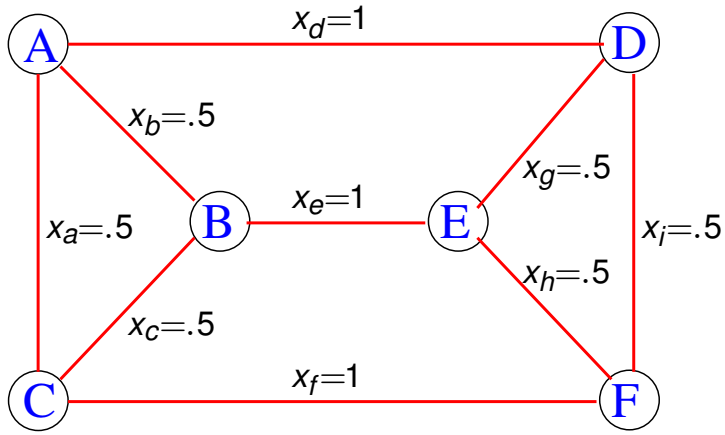
Chegamos em uma solução inteira!!! Solução ótima para TSP de valor 33.

Ex.: Solução ótima fracionária do programa do TSP

Grafo G (grafo envelope) e custos nas arestas:

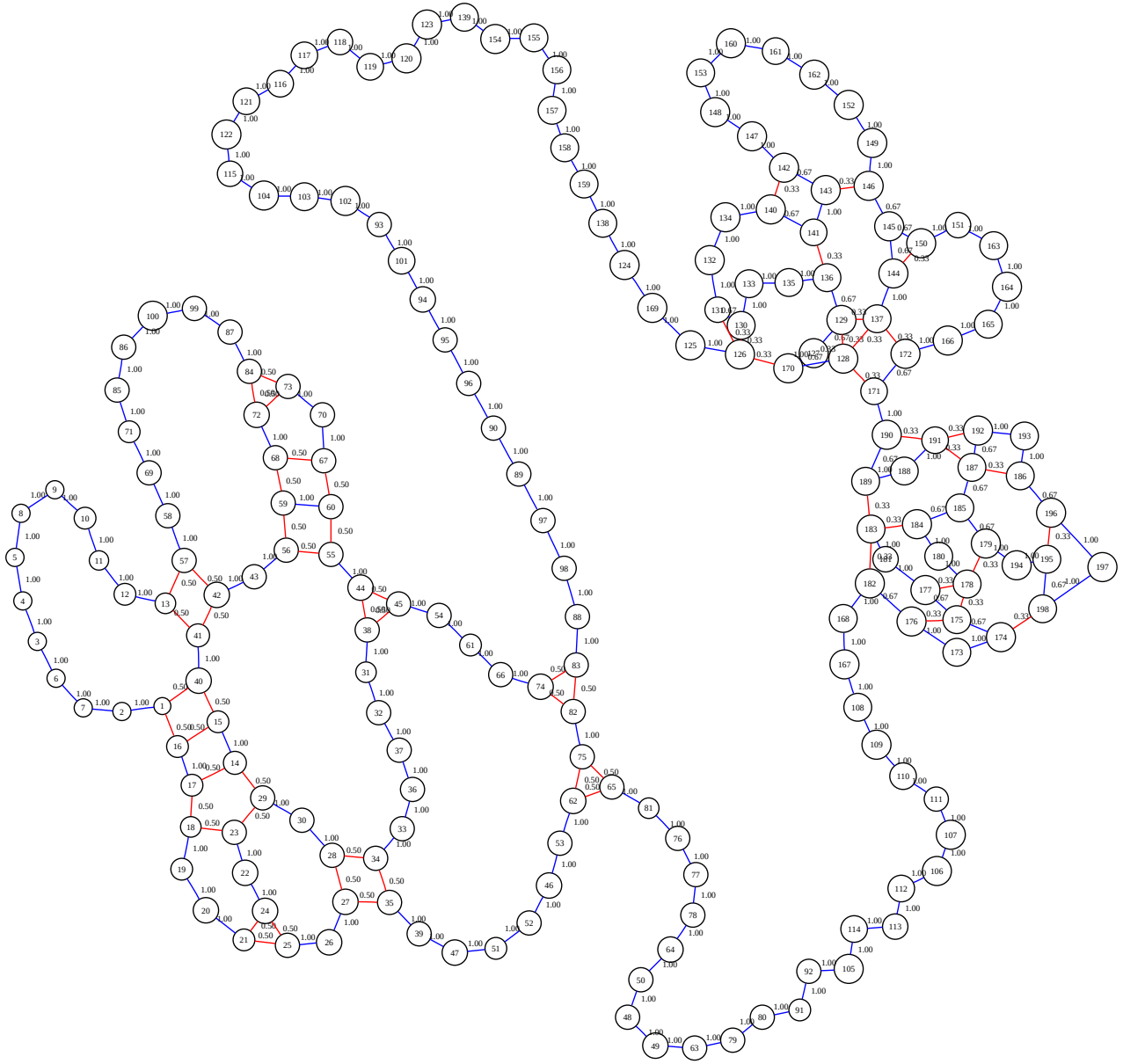


Solução ótima fracionária de peso 45 (LP com restrições de grau e de co



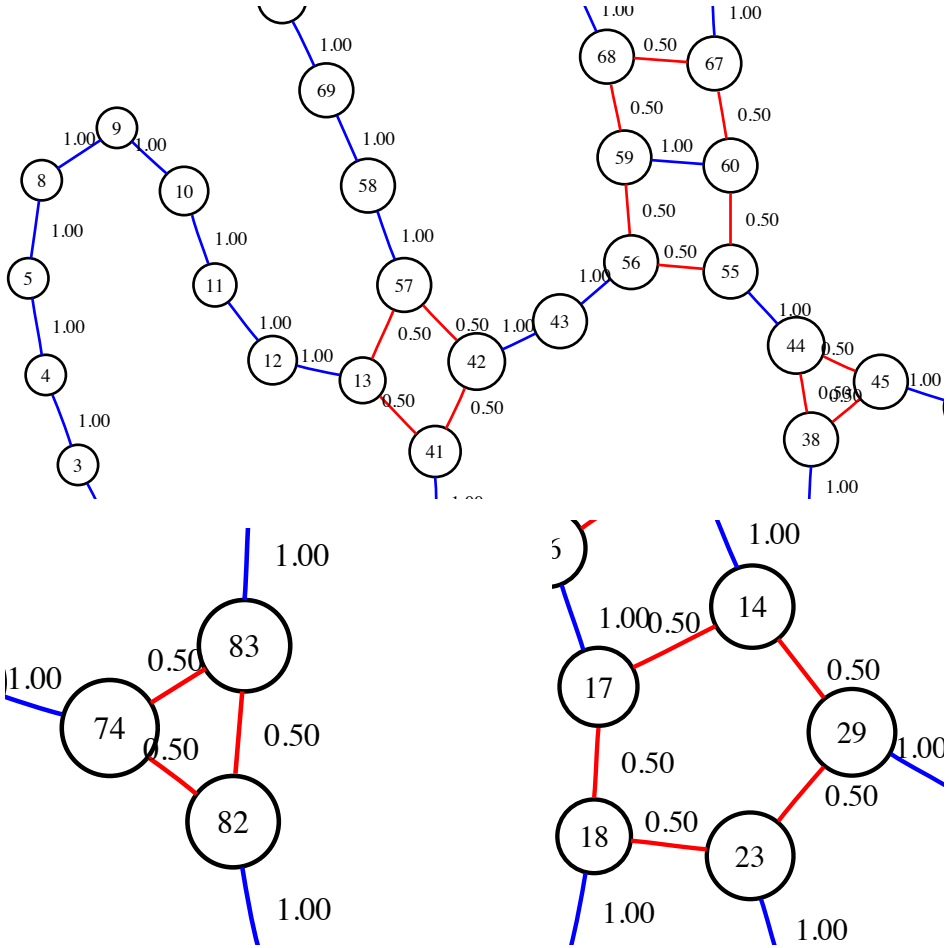
Chvátal'75: apresenta separação de soluções como acima (comb inequa

Ex.: Solução ótima fracionária do TSP `d198.tsp`



Há poucas variáveis fracionárias (grafo original é completo)

Ex.: Solução ótima fracionária do TSP `d198.tsp`



Branch & Cut

- 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 sofistica

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.

- Manutenção de um *Pool* (depósito) de desigualdades
  - Inserção no Pool de novo plano de corte
    Sempre que um novo plano de corte for encontrado, inserir no pool.
  - Ativação das desigualdades do Pool
    Uma desigualdade inserida no pool, pode ser um plano de corte para um nó. Assim, percorremos as desigualdades do pool, inserindo aquelas que são planos de corte para o nó corrente.
  - Eliminação das desigualdades do Pool
    Convém manter uma idade para cada desigualdade do pool. Cada vez que o pool é percorrido, aumenta se a idade das desigualdades que não foram ativadas. Posteriormente, eliminamos as desigualdades velhas. Permite que o pool não cresca exageradament
  - Manutenção de um pool de desigualdades por nó

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^o) % programa relaxado P^o

1 LB $\leftarrow -\infty$ % Lower bound
2 $x^* \leftarrow$ Heurística sobre P^o ($\emptyset$ se não encontrar solução viável)
3 UB $\leftarrow \text{val}(x^*)$ (onde $\text{val}(\emptyset) = +\infty$)
4 Ativos $\leftarrow \{P^o\}$
5 enquanto Ativos $\neq \emptyset$ faça
6 escolha $P \in$ Ativos e remova P de Ativos
8 $x \leftarrow$ solução ótima de P ($x \leftarrow \emptyset$ se P é inviável)
9 se –*Tailing On*– então
10 enquanto $x \neq \emptyset$ e consegue separar x faça
10 insira planos de corte separando x
11 $x \leftarrow$ solução ótima de P ($x \leftarrow \emptyset$ se P é inviável)
12 se $\text{val}(x) > LB$ então atualiza LB (se necessário)
13 se $\text{val}(x) < UB$ % se $\text{val}(x) \geq UB$, ramo é podado
14 se x é inteiro então
15 $UB \leftarrow \text{val}(x)$;
15 $x^* \leftarrow x$
17 senão
18 crie dois subproblemas P' e P'' a partir de P
19 Ativos $\leftarrow$ Ativos $\cup \{P' , P''\}$
20 devolva x

- Uma apresentação sobre o método branch & cut aplicado ao problema da árvore de Steiner (caso particular do Problema da Floresta de Steiner) pode ser encontrada em Ferreira & Wakabayashi'96, Capítulo 4.
  `http://www.ime.usp.br/~yw/livros/livro-new.ps.gz`