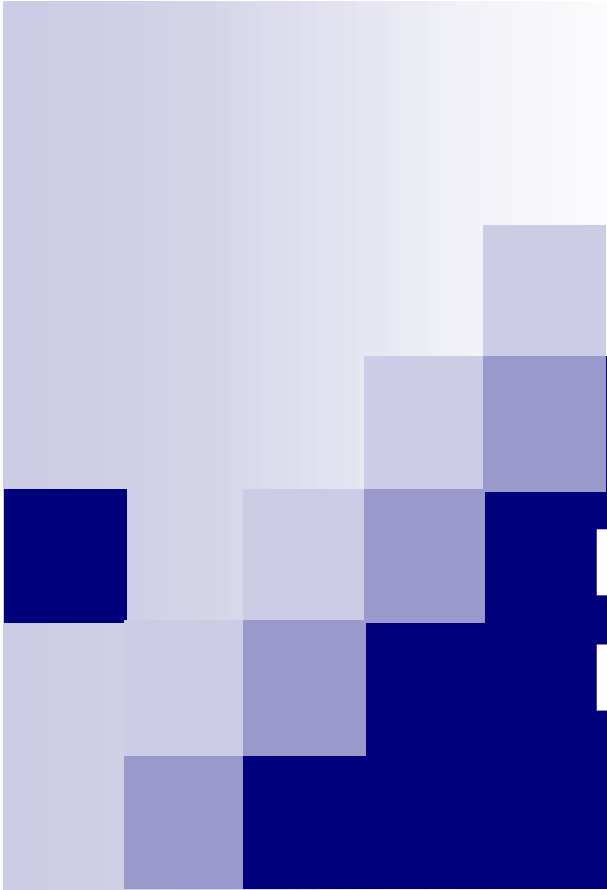


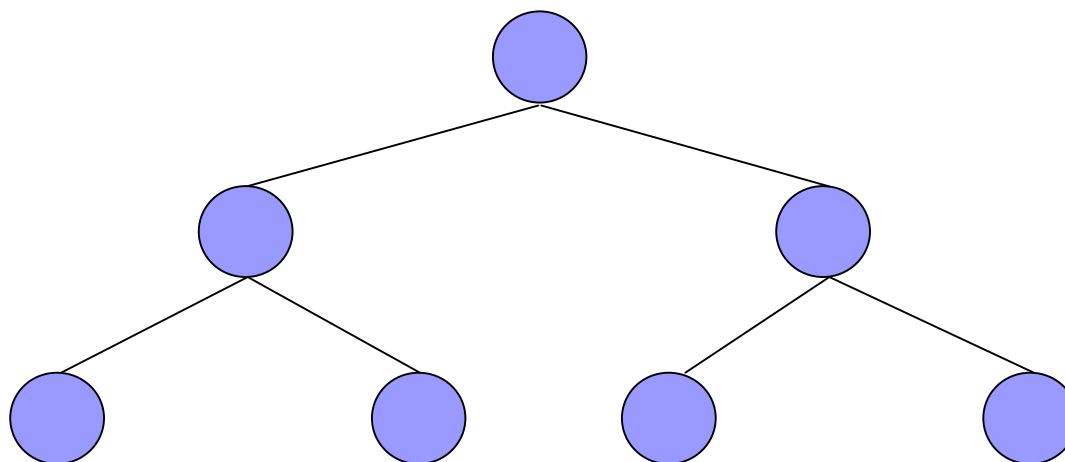


Pesquisa Operacional / Programação Matemática

Otimização discreta

Branch-and-bound (Implementação)

Criando a árvore do B&B



Qual explorar primeiro ?

E depois ?

Note que a ordem pode influir (e muito!)

Regras de seleção

- Regras *a priori*

- ☐ determinam previamente a ordem de escolha dos nós;

- Regras adaptativas

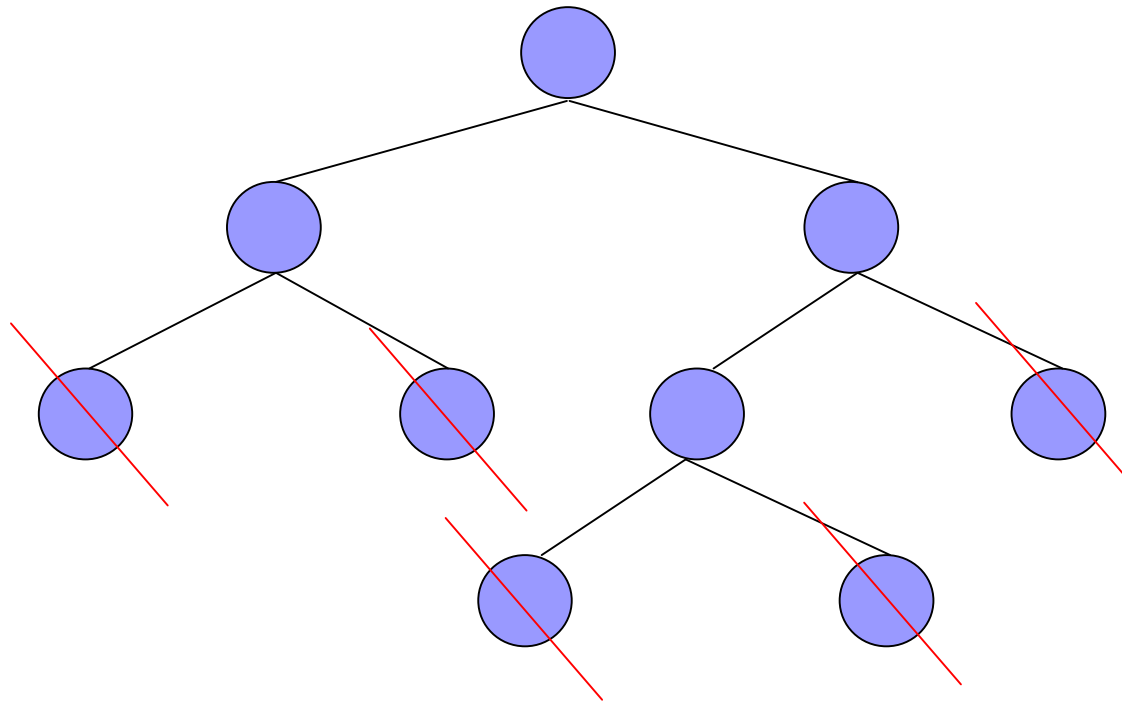
- ☐ dependem das informações dos nós.

Regras *a priori*

- Busca em profundidade com *backtracking*
 - *last-in, first-out*: o último nó a ser incluído na lista é o primeiro a ser examinado
 - *backtracking*: se o nó é podado, retorna-se ao longo do caminho em direção ao nó raiz, até encontrar um nó aberto.
 - ordem: pode-se definir, por exemplo, que o filho à esquerda sempre é examinado primeiro.

Regras *a priori*

- Busca em profundidade com *backtracking*



Regras *a priori*

- Busca em profundidade com *backtracking*
- vantagens:
 - Nós factíveis são mais facilmente encontrados em níveis mais profundos da árvore (qual a vantagem de se encontrar nós factíveis logo ?)
 - Pode-se usar *re-otimização* em nós filhos.
- desvantagem:
 - tende a gerar árvores maiores (com muitos nós).

Regras adaptativas

- Melhor limitante

- Selecionar a cada momento, o nó que tem melhor limitante (e que eventualmente, pode fornecer a melhor solução inteira).

Regras adaptativas

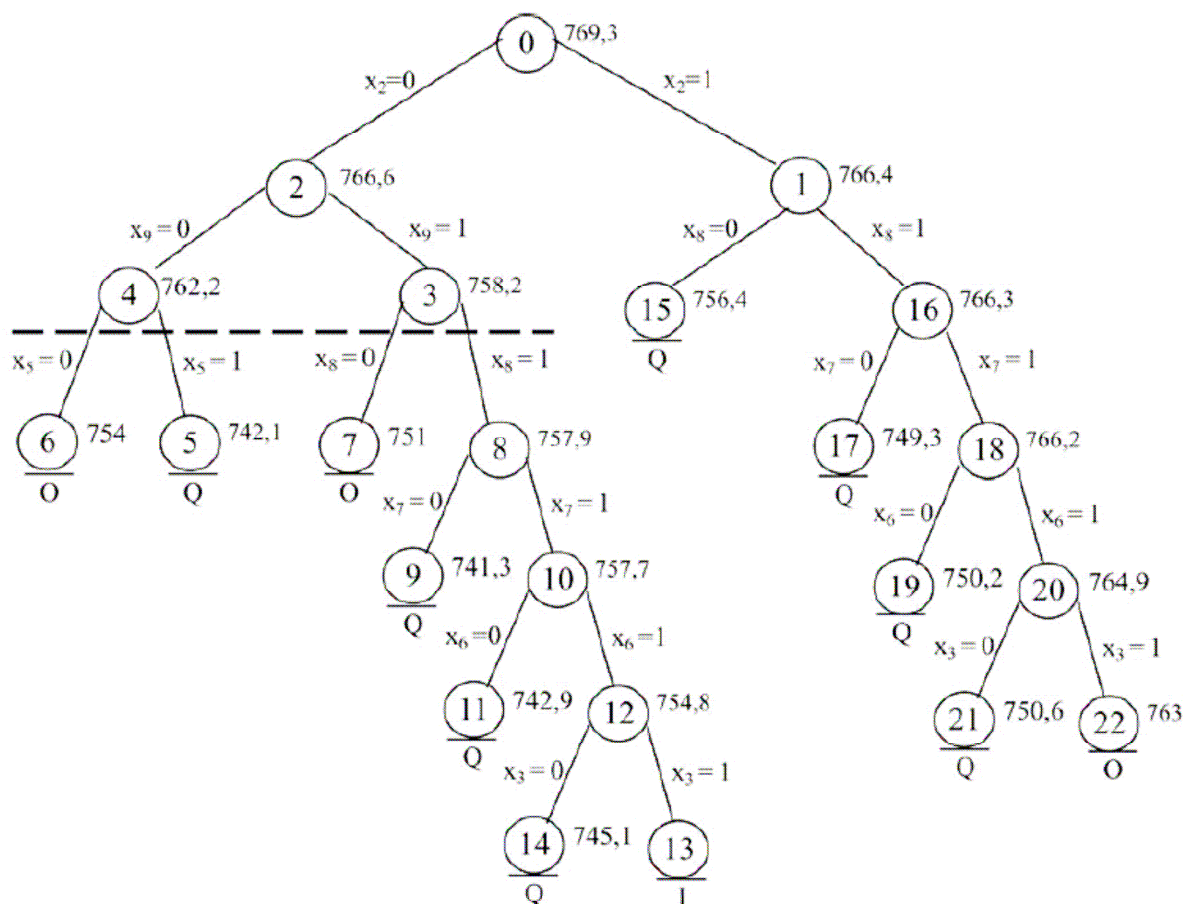
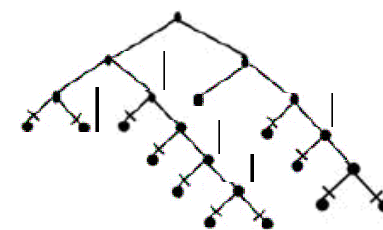
- Melhor limitante
- vantagem:
 - menos nós explorados no final.
- desvantagem:
 - grande número de nós ativos a cada momento (limites de memória ?)

Exemplo

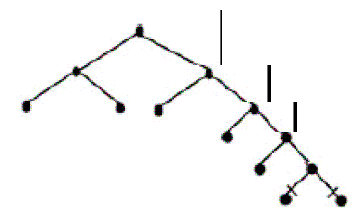
$$z = \max 31x_1 + 126x_2 + 131x_3 + 37x_4 + 180x_5 + 170x_6 + 182x_7 + 123x_8 + 160x_9 + 80x_{10}$$

$$13x_1 + 111x_2 + 101x_3 + 27x_4 + 174x_5 + 136x_6 + 146x_7 + 99x_8 + 145x_9 + 76x_{10} \leq 606$$

Solução



busca em profundidade



melhor limitante

Outras estratégias

- Busca em largura

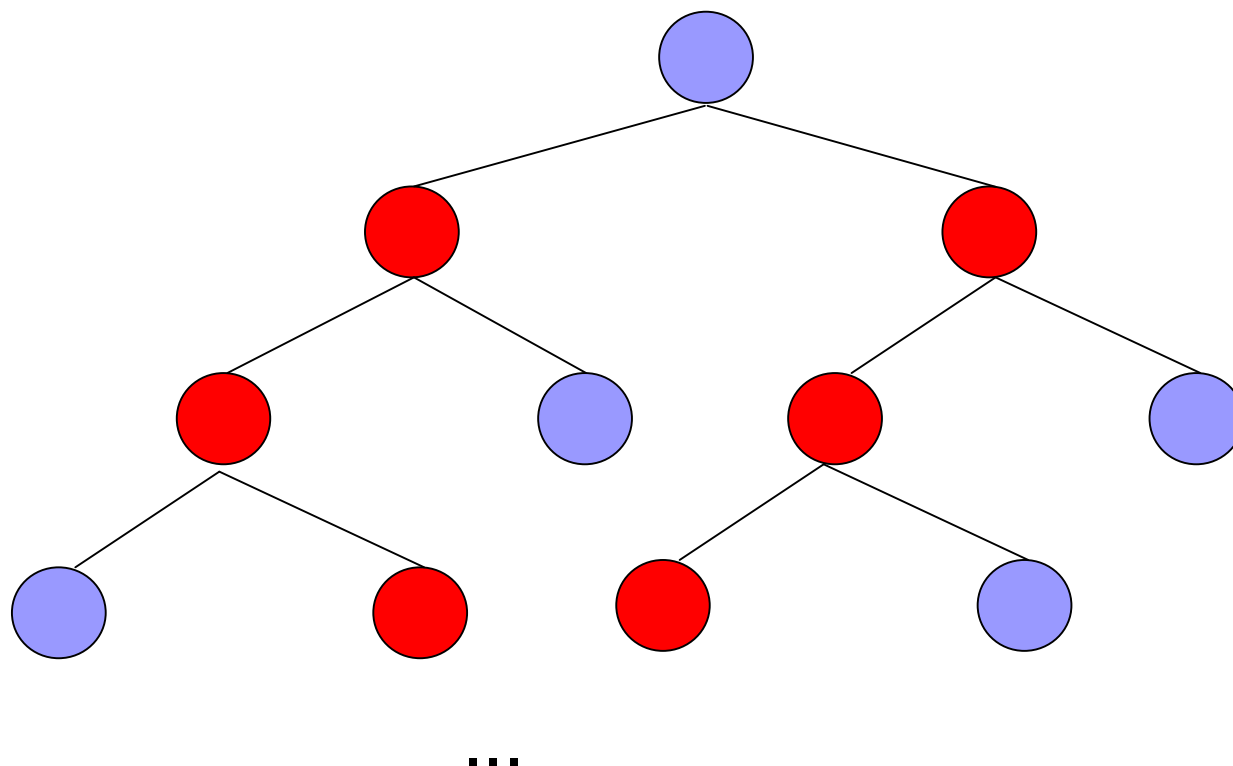
todos os nós em um dado nível são considerados, antes de passar-se para o nível seguinte.

- Pode ser pouco interessante em um algoritmo exato, mas pode ser conveniente em heurísticas;

(ex.: beam search)

■ Exemplo de heurística

- (Escolhe apenas os nós mais promissores para continuar a busca)



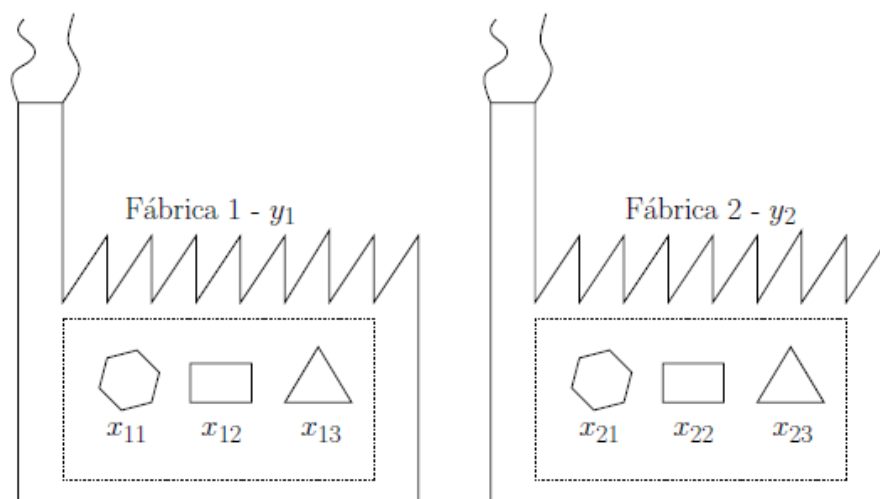
Escolha da variável a ramificar

- Pode influir bastante na velocidade de convergência do algoritmo.
 - (idéia: tipicamente, se fixamos algumas variáveis - *as certas* - em valores inteiros, as outras naturalmente se tornam inteiras).
 - difícil saber quais as variáveis *certas* para ramificar.

Escolha da variável a ramificar

■ Prioridades:

- Variáveis que definem se uma fábrica deve ou não ser construída devem ser ramificadas antes de variáveis que definem quais máquinas comprar para a fábrica.



Escolha da variável a ramificar

■ Outras estratégias:

- escolher a variável que, tornada inteira, mais modifica a função objetivo.

- Generalized upper bound constraints

$$\sum_{j \in Q_i} x_j = 1, \quad i = 1 \dots, p$$

ramificação:

$$\sum_{j \in Q_i^1} x_j = 1 \quad \sum_{j \in Q_i \setminus Q_i^1} x_j = 1$$

Exemplo

- Material adicional - Site