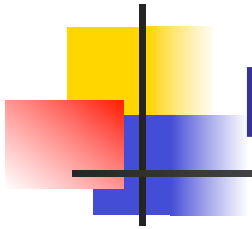
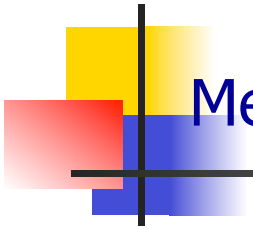


Programação Inteira

Algoritmo Branch-and-Bound (ou enumeração implícita)



Métodos de Solução: *Branch-and-Bound*

- O método *Branch-and-Bound* (B&B) baseia-se na idéia de desenvolver uma enumeração inteligente das soluções candidatas à solução ótima inteira de um problema.

- Apenas uma fração das soluções factíveis é realmente examinada.

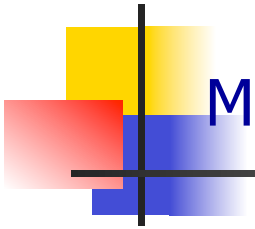
- O termo *branch* refere-se ao fato de que o método efetua partições no espaço das soluções e o termo *bound* ressalta que a prova da otimalidade da solução utiliza-se de limites calculados ao longo da enumeração.

Métodos de Solução: *Branch-and-Bound*

Exemplo

$$
\begin{aligned}
max \quad & 21x_1 + 11x_2 \\
s.a. \quad & 7x_1 + 4x_2 \leq 13 \\
& x_1, x_2 \geq 0 \\
& x_1, x_2 \quad inteiros
\end{aligned}
$$

O problema é um problema de programação linear inteira, pois as variáveis devem ser inteiras.



Métodos de Solução: *Branch-and-Bound*

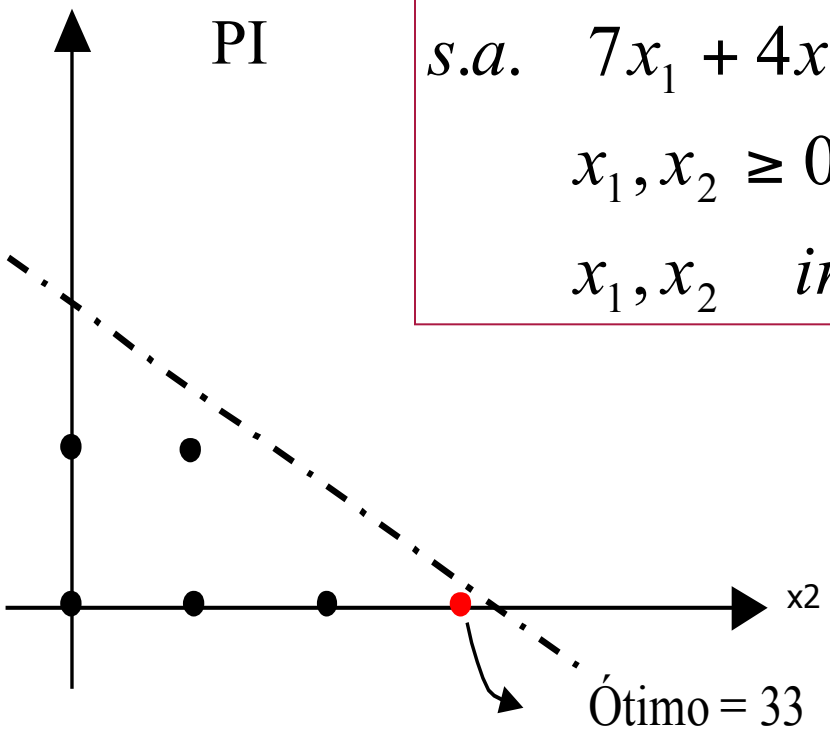
•Na Figura (a) têm-se os pontos que representam as soluções factíveis do problema (todos os pontos inteiros que satisfazem as restrições).

• O problema de programação linear (RL) obtido ao desconsiderarmos as restrições de integralidade das variáveis inteiras é conhecido como a relaxação linear do PI (ver Figura (b)).

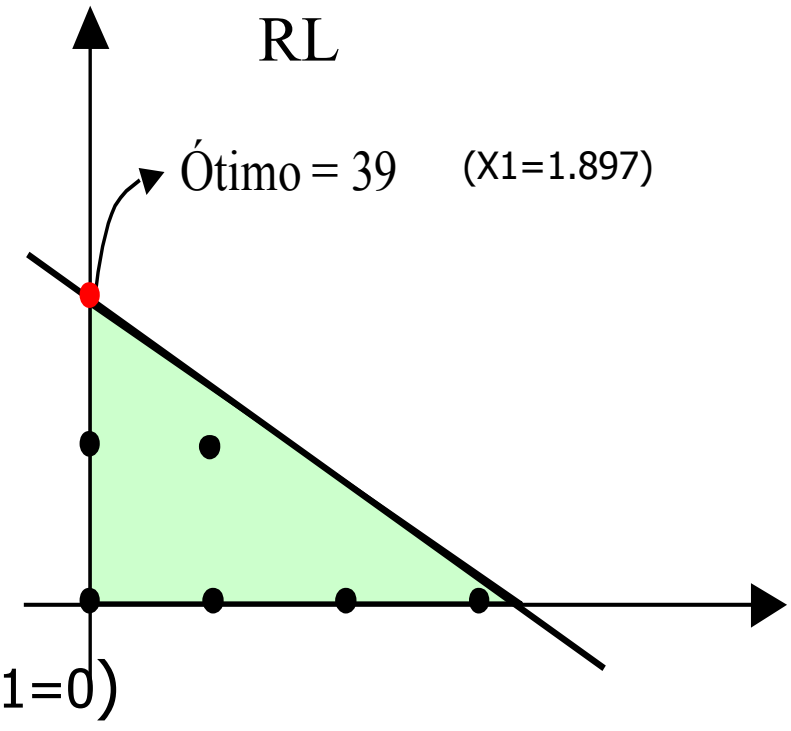
• Existem outros tipos de relaxação, como por exemplo a Relaxação Lagrangiana: relaxa-se algumas restrições, consideradas complicadas, incorporando uma penalidade na função objetivo (não falaremos no curso de PM);

Métodos de Solução: *Branch-and-Bound*

$$max \quad 21x_1 + 11x_2$$
$$s.a. \quad 7x_1 + 4x_2 \leq 13$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \quad inteiros$$



(a)



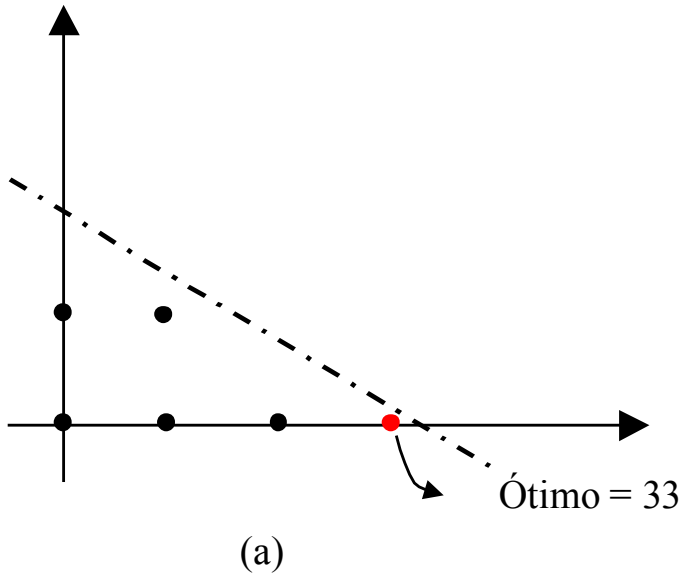
(b)

Métodos de Solução: *Branch-and-Bound*

• Como podemos observar a solução do problema linear (RL) é sempre maior ou igual a solução do PI, pois o problema relaxado é composto por todas as soluções inteiras e também as soluções reais do problema, logo é formado por um conjunto de soluções factíveis mais abrangente.

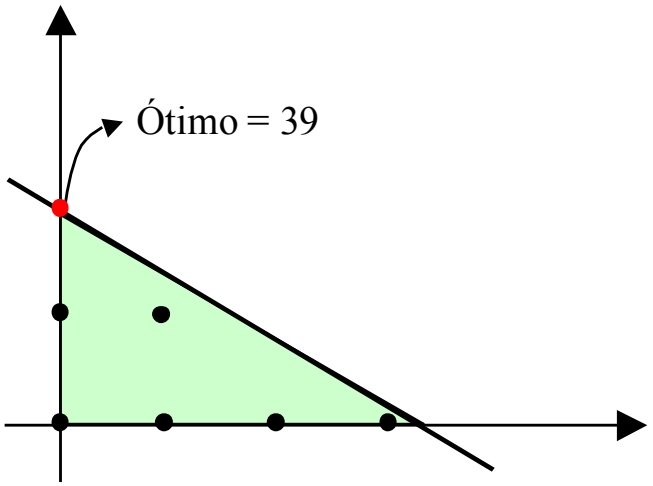
• Assim temos que, para um problema de maximização $Z^*_{PPL} \geq Z^*_{PPI}$, ou seja, a solução ótima da relaxação linear de um problema inteiro (Z^*_{PPL}) é sempre maior ou igual a solução ótima do problema inteiro (Z^*_{PPI}).

Métodos de Solução: *Branch-and-Bound*



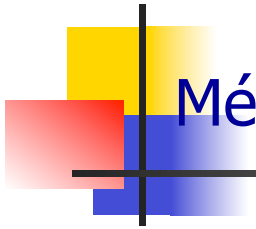
(a)

Problema inteiro



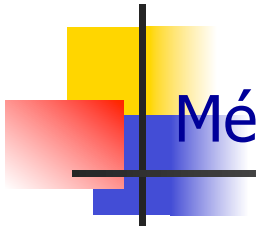
(b)

Problema relaxado



Métodos de Solução: *Branch-and-Bound*

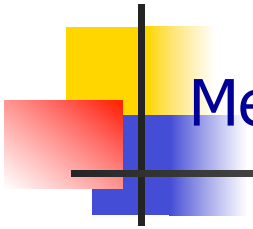
- **Princípio básico:** se a solução do RL relaxado corresponde a uma solução do PI, pois possui todas as variáveis inteiras, então esta solução é a solução ótima do PI.



Métodos de Solução: *Branch-and-Bound*

• **Idéia Geral:** relaxar o problema de programação inteira e dividir o problema relaxado em vários problemas até encontrar soluções inteiras ou não factíveis, o ótimo é a melhor solução encontrada.

O algoritmo *B&B* é baseado na idéia de "dividir para conquistar", ou seja, trabalhamos em problemas menores e mais fáceis de resolver em busca da solução ótima.



Métodos de Solução: *Branch-and-Bound*

- A divisão do problema é interrompida quando uma das condições a seguir é satisfeita. Essas condições são chamadas de testes de sondagem ou Poda do nó.

(I ou poda por infactibilidade) O problema relaxado é infactível.

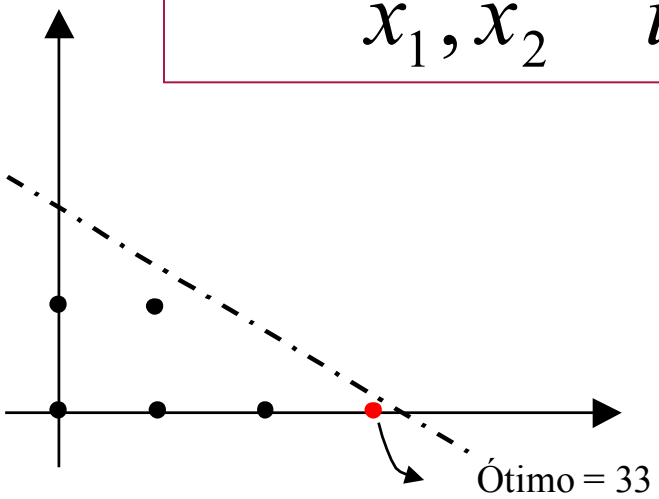
(O ou poda por otimalidade) A solução ótima do problema relaxado é inteira .

(Q ou poda por qualidade) O valor de qualquer solução factível do problema relaxado é pior que o valor da melhor solução factível atual (solução incumbente).

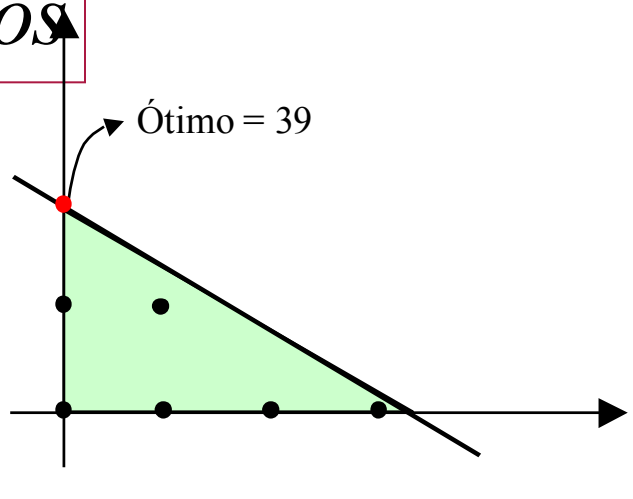
- Quando uma dessas três condições ocorre, o subproblema pode ser descartado (sondado ou podado), pois todas as suas soluções factíveis estão implicitamente enumeradas.

Exemplo: *Branch-and-Bound*

$$
\begin{aligned}
max \quad & 21x_1 + 11x_2 \\
s.a. \quad & 7x_1 + 4x_2 \leq 13 \\
& x_1, x_2 \geq 0 \\
& x_1, x_2 \quad inteiros
\end{aligned}
$$



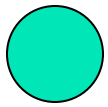
(a)



(b)

Exemplo: *Branch-and-Bound*

- Resolvendo o problema relaxado tem-se que:
  - Valor ótimo da solução: 39
  - Valores das variáveis x_1=1.86 e x_2=0

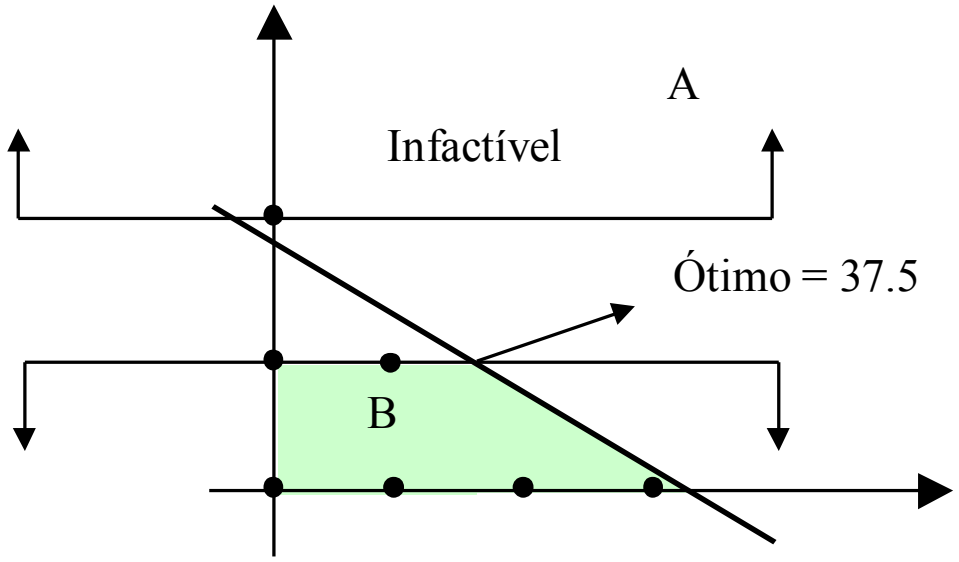


Z=39
x_1=1.86
x_2=0

- Logo o valor de x_1 não é inteiro, então dividimos o problema em dois subproblemas:
  - um onde consideramos o valor de $x_1 \geq 2$, que vamos chamar de subproblema A
  - outro consideramos $x_1 \leq 1$, chamado de subproblema B.

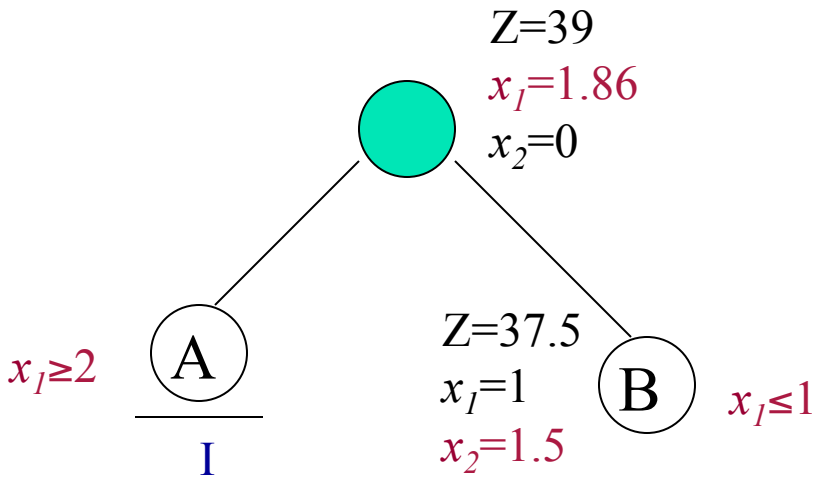
Exemplo: *Branch-and-Bound*

Suproblema A	Subproblema B
$max \quad 21x_1 + 11x_2$	$max \quad 21x_1 + 11x_2$
$s.a. \quad 7x_1 + 4x_2 \leq 13$	$s.a. \quad 7x_1 + 4x_2 \leq 13$
$x_1 \geq 2$	$x_1 \leq 1$
$x_1, x_2 \geq 0$	$x_1, x_2 \geq 0$



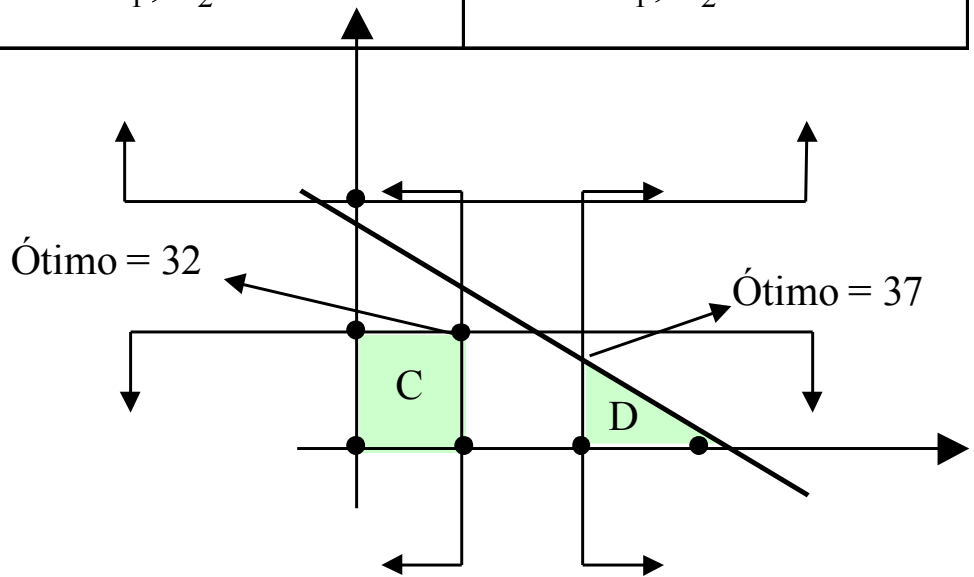
Exemplo: *Branch-and-Bound*

- Não encontramos solução factível ao resolver o problema A, então aplicando o o critério para poda podemos eliminá-lo ((I) O problema relaxado é infactível).
- Resolvendo o subproblema B temos Z = 37.5, x_1 =1 e x_2 =1.5
  - Agora x_2 não é inteiro, logo particionamos o problema em dois considerando o subproblema C com a variável $x_2 \leq 1$ e o subproblema D com $x_2 \geq 2$.



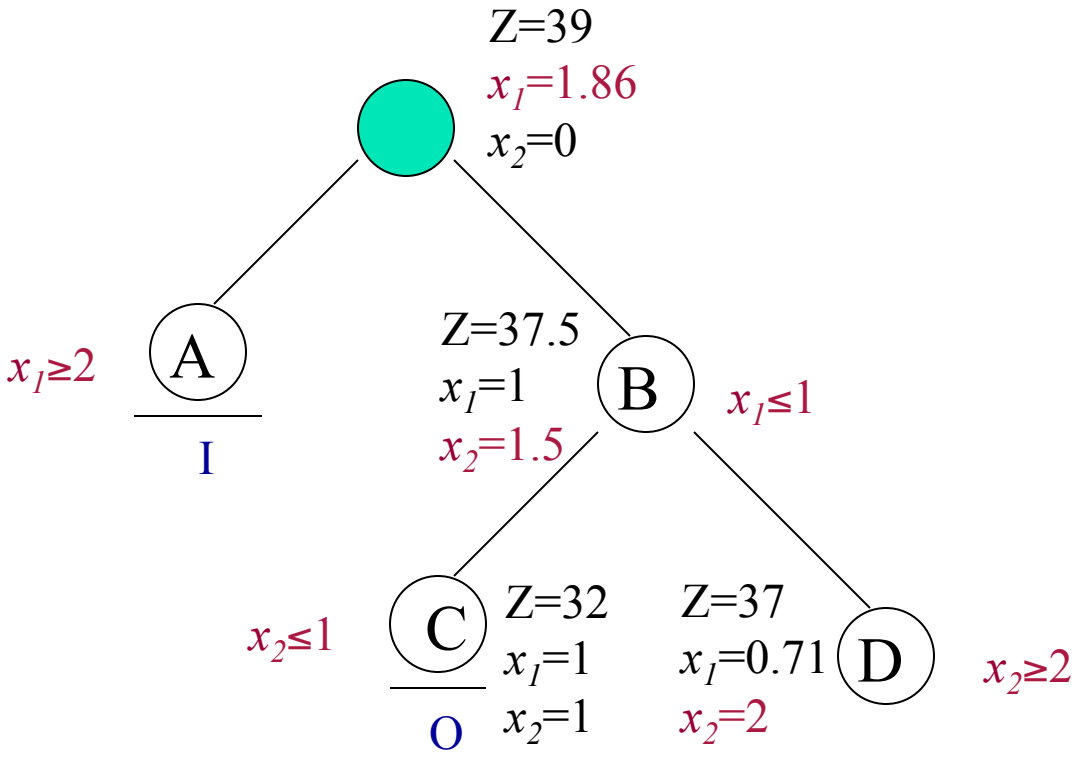
Exemplo: *Branch-and-Bound*

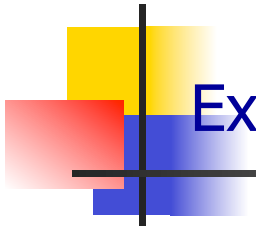
Suproblema C	Subproblema D
$max \quad 21x_1 + 11x_2$	$max \quad 21x_1 + 11x_2$
$s.a. \quad 7x_1 + 4x_2 \leq 13$	$s.a. \quad 7x_1 + 4x_2 \leq 13$
$x_1 \leq 1$	$x_1 \leq 1$
$x_2 \leq 1$	$x_2 \geq 2$
$x_1, x_2 \geq 0$	$x_1, x_2 \geq 0$



Exemplo: *Branch-and-Bound*

(O - otimalidade) A solução ótima do problema relaxado é inteira.



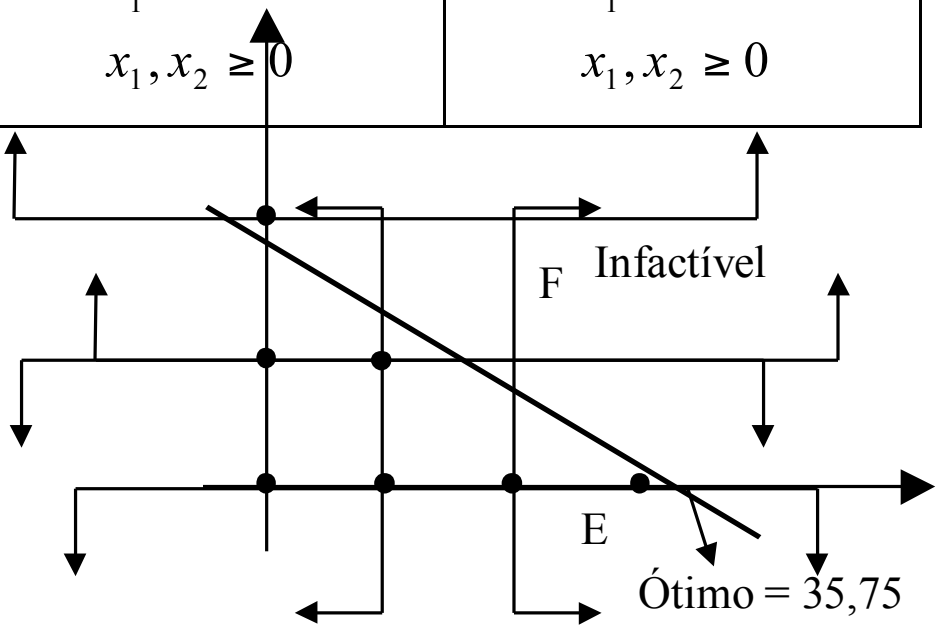


Exemplo: *Branch-and-Bound*

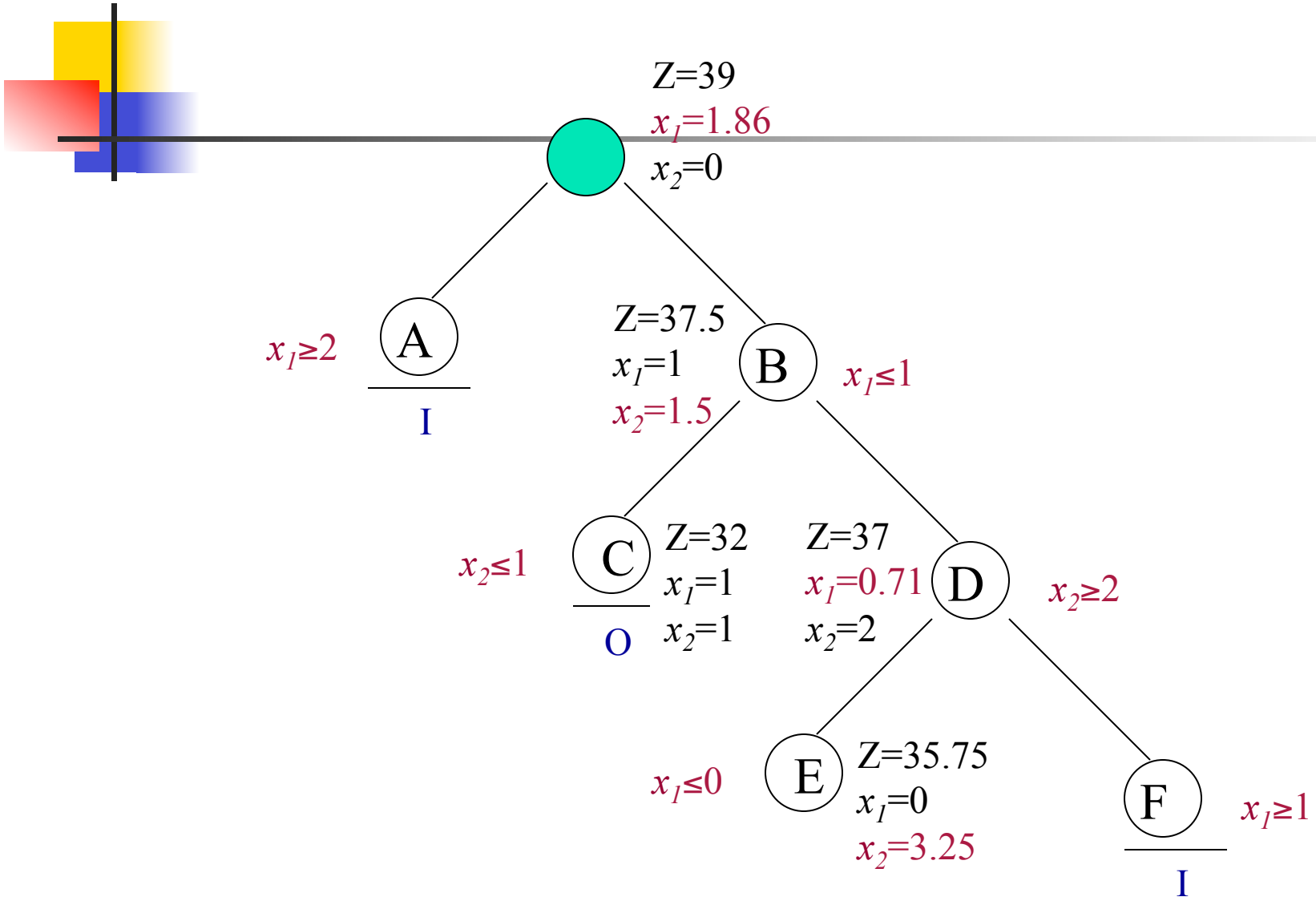
- A solução do subproblema C é igual a 32, x_1=1 e x_2=1, as duas variáveis são inteiras, logo considerando o teste de sondagem (O) este problema pode ser sondado por otimalidade.
- Resolvendo o subproblema D temos Z = 37, x_1=0.71 e x_2=2
  - note que a variável x_1 novamente não é inteira, então particionamos o subproblema gerando dois novos subproblemas como mostramos a seguir

Exemplo: *Branch-and-Bound*

Suproblema E	Subproblema F
$max \quad 21x_1 + 11x_2$	$max \quad 21x_1 + 11x_2$
$s.a. \quad 7x_1 + 4x_2 \le 13$	$s.a. \quad 7x_1 + 4x_2 \le 13$
$x_1 \le 1$	$x_1 \le 1$
$x_2 \ge 2$	$x_2 \ge 2$
$x_1 \le 0$	$x_1 \ge 1$
$x_1, x_2 \ge 0$	$x_1, x_2 \ge 0$



Exemplo: *Branch-and-Bound*

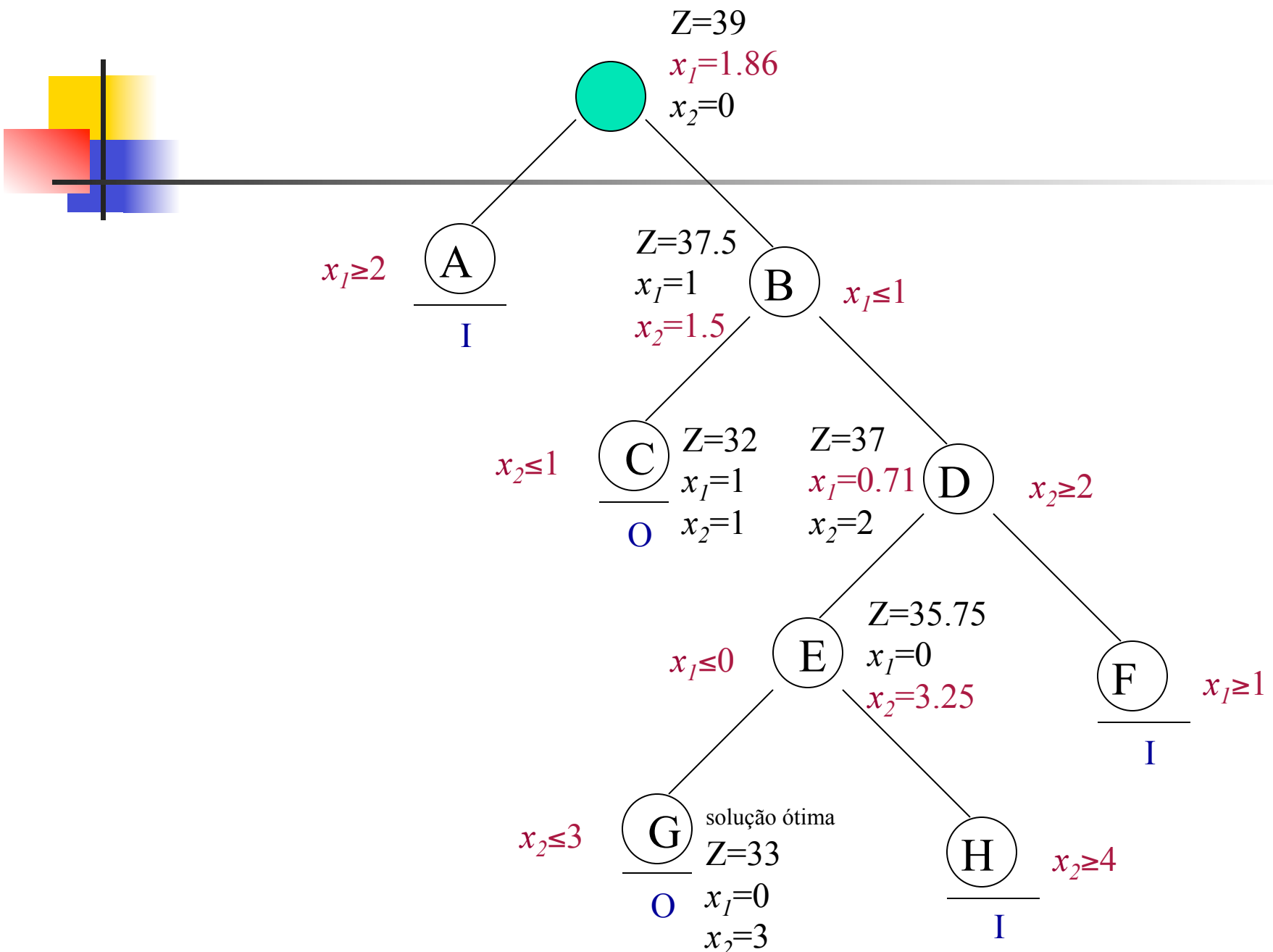


Exemplo: *Branch-and-Bound*

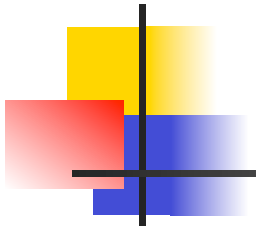
- O problema F é infactível, logo podemos usar I e eliminá-lo
- O subproblema E tem solução igual a 35.75 e x_1=0 e x_2=3.25

Suproblema G	Subproblema H
$max \quad 21x_1 + 11x_2$	$max \quad 21x_1 + 11x_2$
$s.a. \quad 7x_1 + 4x_2 \leq 13$	$s.a. \quad 7x_1 + 4x_2 \leq 13$
$x_1 \leq 1$	$x_1 \leq 1$
$x_2 \geq 2$	$x_2 \geq 2$
$x_1 \leq 0$	$x_1 \leq 0$
$x_2 \leq 3$	$x_2 \geq 4$
$x_1, x_2 \geq 0$	$x_1, x_2 \geq 0$

Exemplo: *Branch-and-Bound*

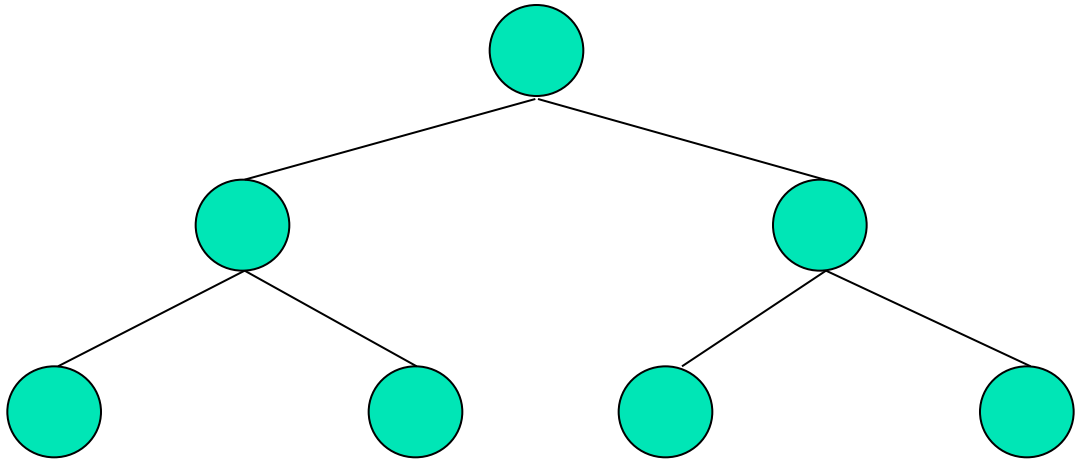


Exemplo: *Branch-and-Bound*



- Resolvendo o subproblema G obtemos Z = 33, x_1=0 e x_2=3 , logo a solução é inteira, portanto aplicando o (O) este problema pode ser sondado.
- O subproblema H não tem solução factível e também pode ser sondado por infactibilidade 9I).
- Temos que nenhum nó pode ser ramificado, logo, a melhor solução inteira encontrada é dada pelo problema G e é a solução ótima do problema.
- Na resolução do Exemplo através do método *B&B* podemos observar que muitas soluções não precisaram ser avaliadas explicitamente. Isso fica mais claro quando se resolve problemas maiores.

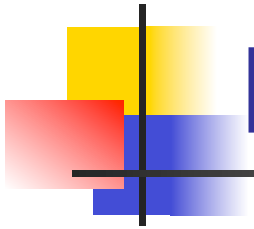
Criando a árvore do B&B



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.

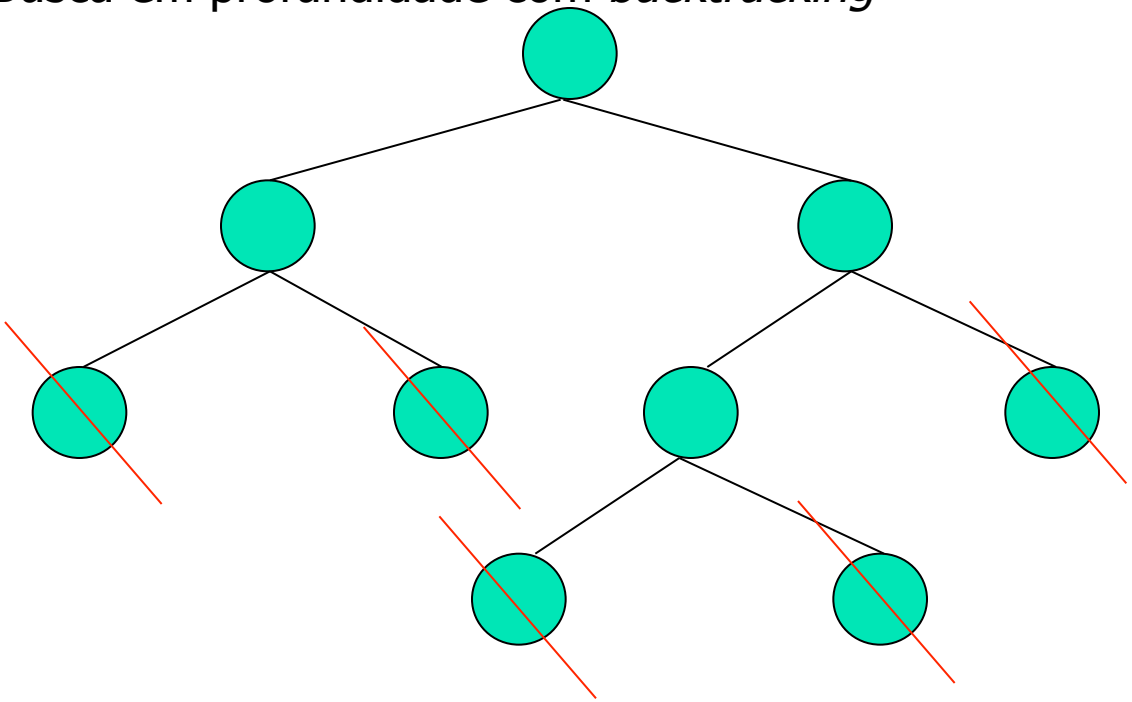


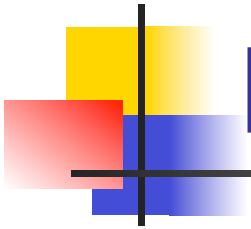
Regras *a priori*

- 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*

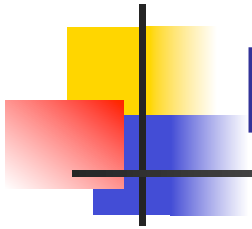
- Busca em profundidade com *backtracking*





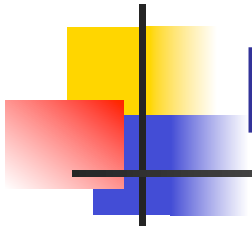
Regras *a priori*

- 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

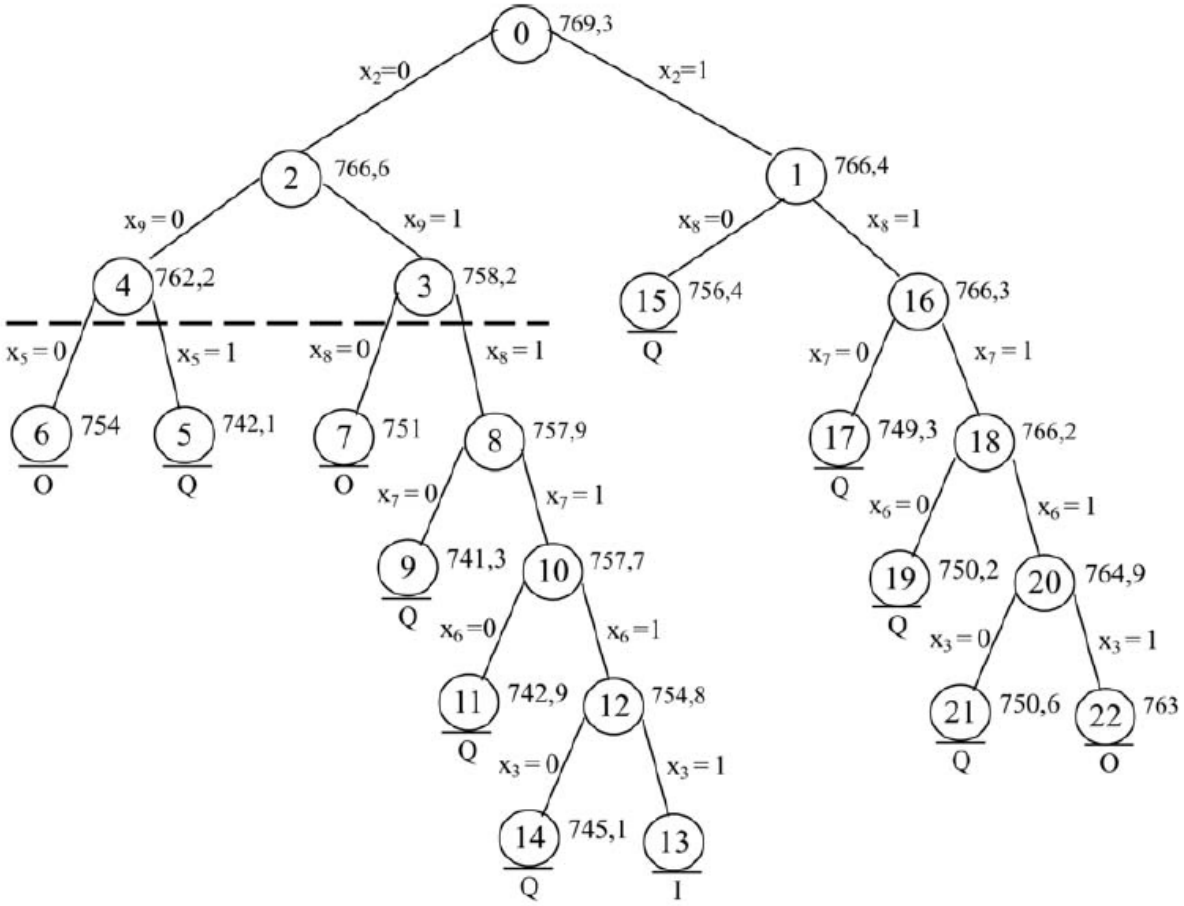
- vantagem:
  - menos nós explorados no final.
- desvantagem:
  - grande número de nós ativos a cada momento (limites de memória ?)

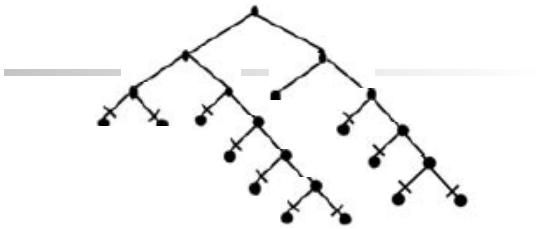
Exemplo (Livro Página 249)

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

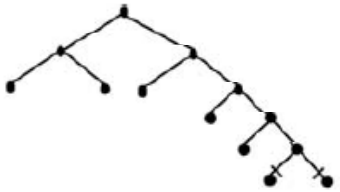
$$13x_1 + 111x_2 + 101x_3 + 27x_4 + 174x_5 + 136x_6 + 146x_7 + 99x_8 + 145x_9 + 76x_{10} \leq 606$$

Solução

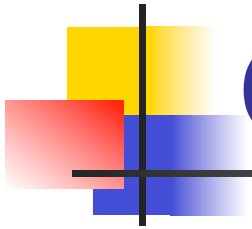




busca em profundidade



melhor limitante

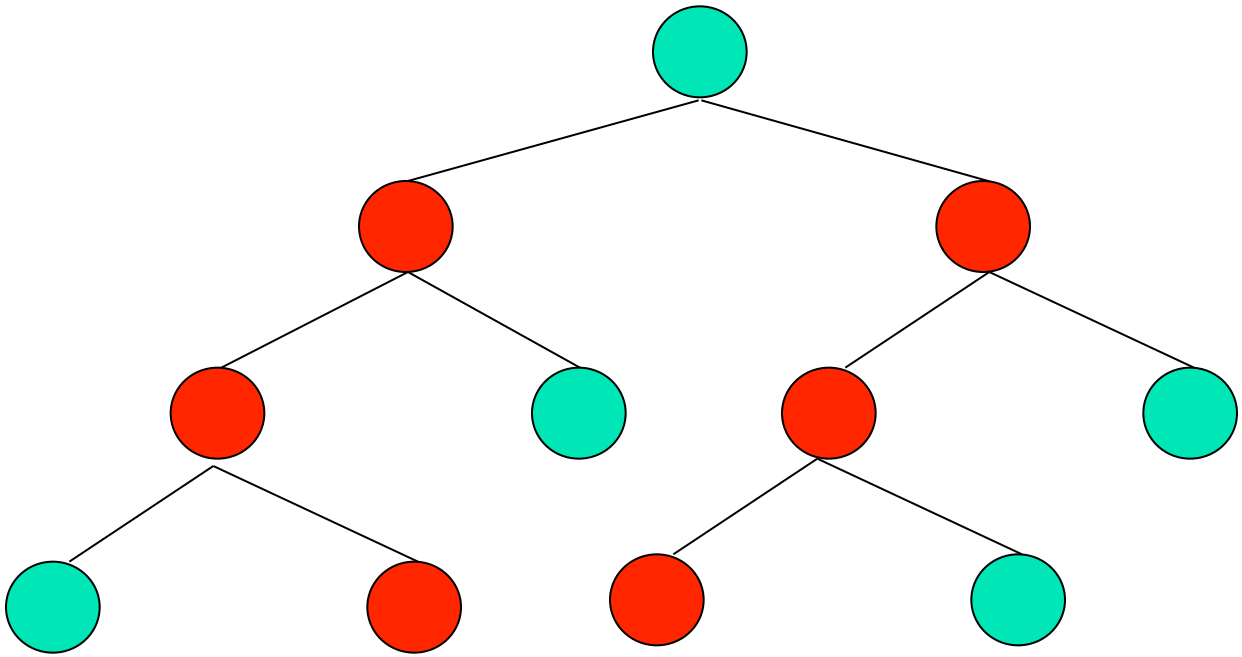


Outras estratégias

- Busca em largura

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

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



...

Exercício .

- Encontre a solução ótima para o problema de programação inteira abaixo. Especifique qual o motivo de cada poda dos nós. A primeira relaxação linear deve ser resolvida utilizando o algoritmo simplex O primeiro filho que será acrescentado com xi<= deverá ser resolvido pelo simplex tabelas.

$$\text{Max } Z = x_1 + 2x_2$$

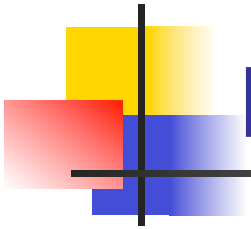
Sujeito a:

$$2x_1 + 2x_2 \leq 6$$

$$x_1 + 2x_2 \leq 5$$

$$4x_1 + 2x_2 \leq 8$$

$x_1 \geq 0$, $x_2 \geq 0$ e inteiras.



Programação Inteira

- Referências:
- **Notas de aulas do Prof. Silvio Alexandre de Araujo**

http://www.dcce.ibilce.unesp.br/~saraujo/

Material da Professora Gladys Castillo do Departamento de Matemática da

Universidade de Aveiro (http://www.mat.ua.pt/io/)

Notas de aula do Prof. Alysson Machado Costa

www.icmc.usp.br/~alysson