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/)

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

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

Métodos de Solução: *Branch-and-Bound*

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

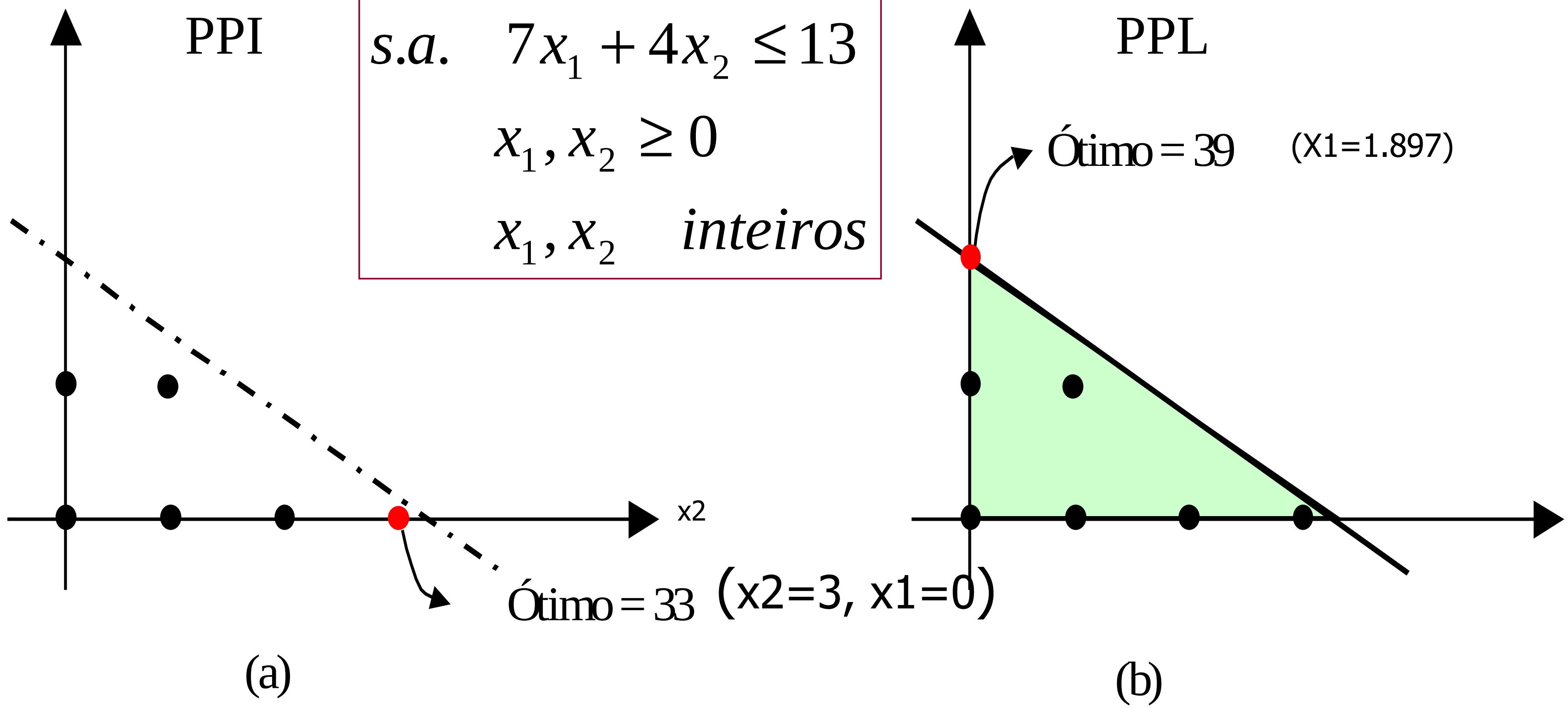
• Na Figura (a) têm-se os pontos que representam as soluções factíveis do problema (todos os pontos inteiros que satisfazem as restrições).

• O problema de programação linear (PPL) obtido ao desconsiderarmos as restrições de integralidade das variáveis inteiras é conhecido como a relaxação linear do PPI (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;

Métodos de Solução: *Branch-and-Bound*

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

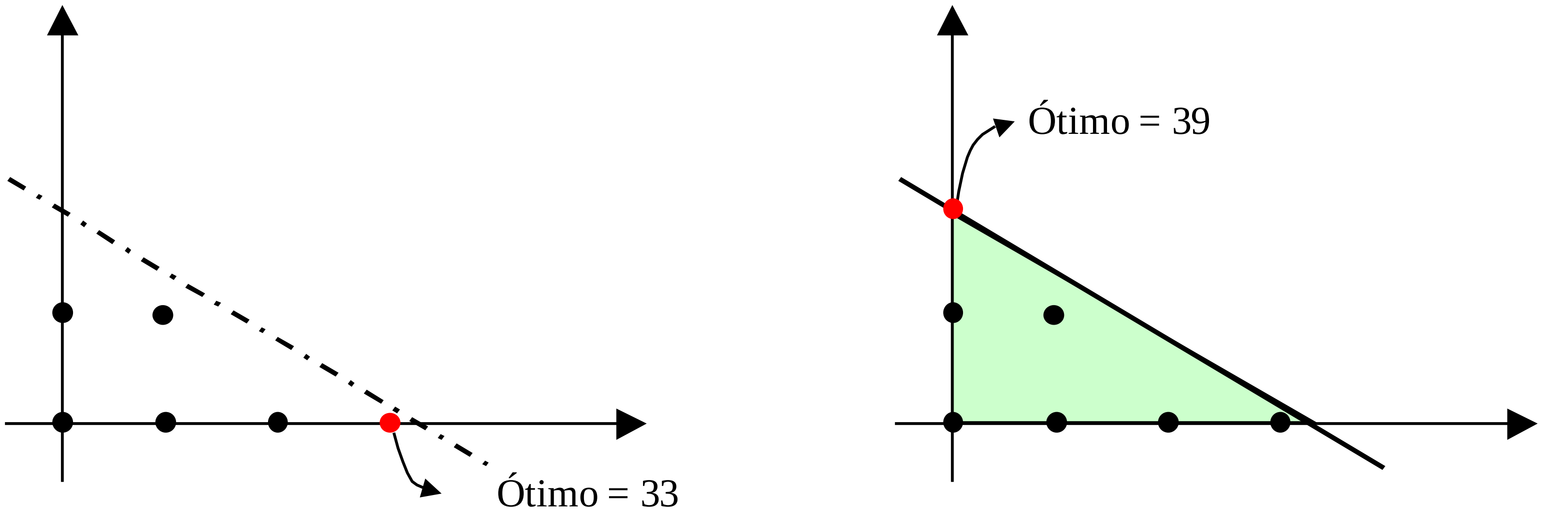


Métodos de Solução: *Branch-and-Bound*

- Como podemos observar a solução do PPL é sempre maior ou igual a solução do PPI, 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) (b)

Métodos de Solução: *Branch-and-Bound*

• **Princípio básico:** se a solução do PPL relaxado corresponde a uma solução do PPI, pois possui todas as variáveis inteiras, então esta solução é a solução ótima do PPI.

Prova: É fácil provar tal princípio, pois sabemos que $Z^*_{PPL} \geq Z^*_{PPI}$ para um problema de máximo, logo, $Z^*_{PPL} \geq Z_{PPI}$, ou seja, Z^*_{PPL} é maior ou igual a uma solução qualquer do PPI (Z_{PPI}).

Suponha que Z^*_{PPL} seja inteira, logo temos que, $Z^*_{PPL} = Z_{INT} \geq Z_{PPI}$, portanto o valor máximo para o PPI será exatamente igual a Z^*_{PPL} .

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 (TS).

(TS1 ou poda por infactibilidade) O problema relaxado é infactível.

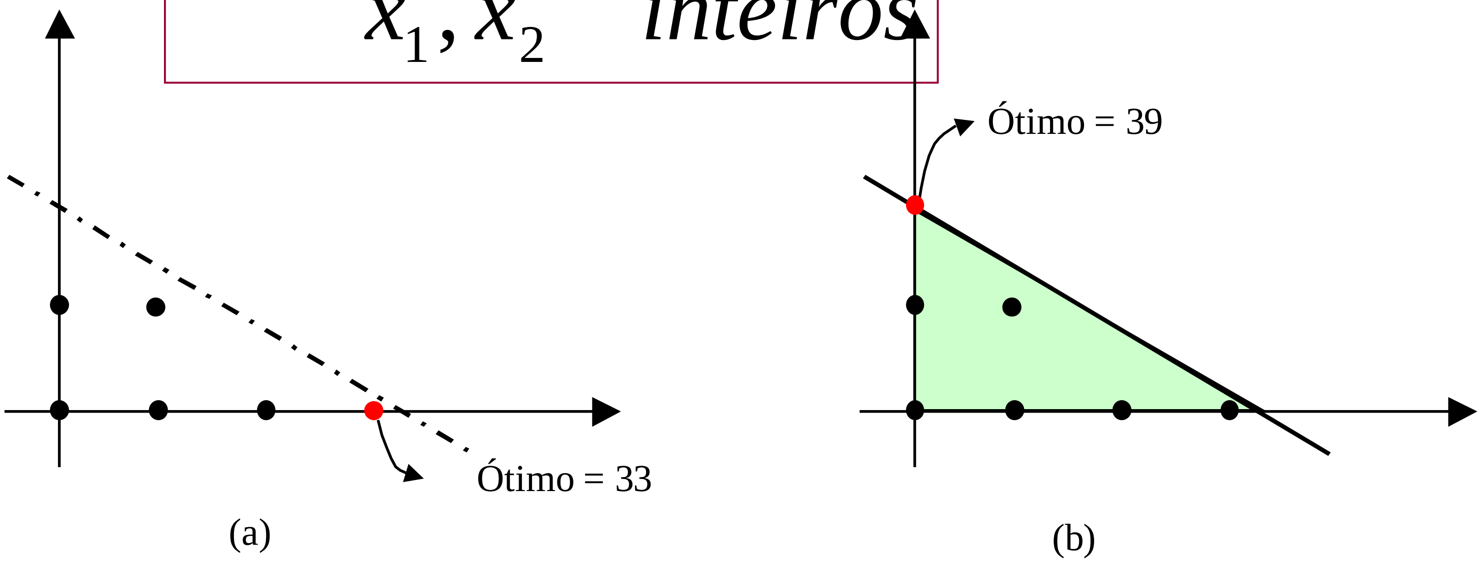
(TS2 ou poda por otimalidade) A solução ótima do problema relaxado é inteira .

(TS3 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.

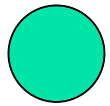
Exemplo: *Branch-and-Bound*

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



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

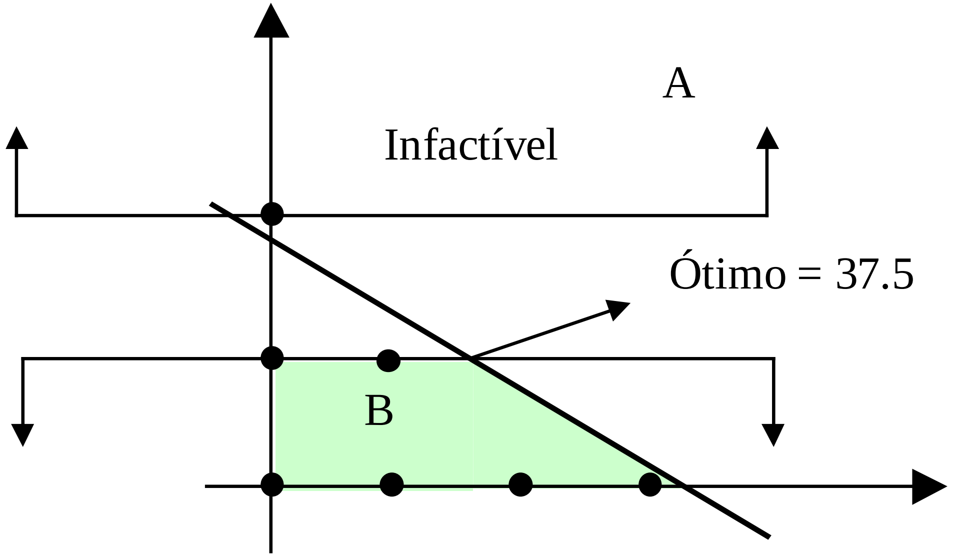


Z=39
x_1=1.86
x_2=0

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

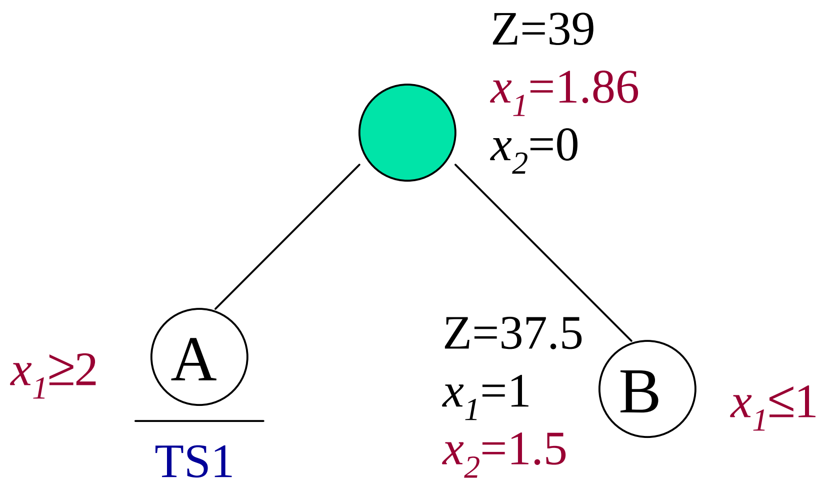


Exemplo: *Branch-and-Bound*

- Não encontramos solução factível ao resolver o problema A, então aplicando o o critério para poda podemos eliminá-lo ((TS 1) O problema relaxado é infactível).

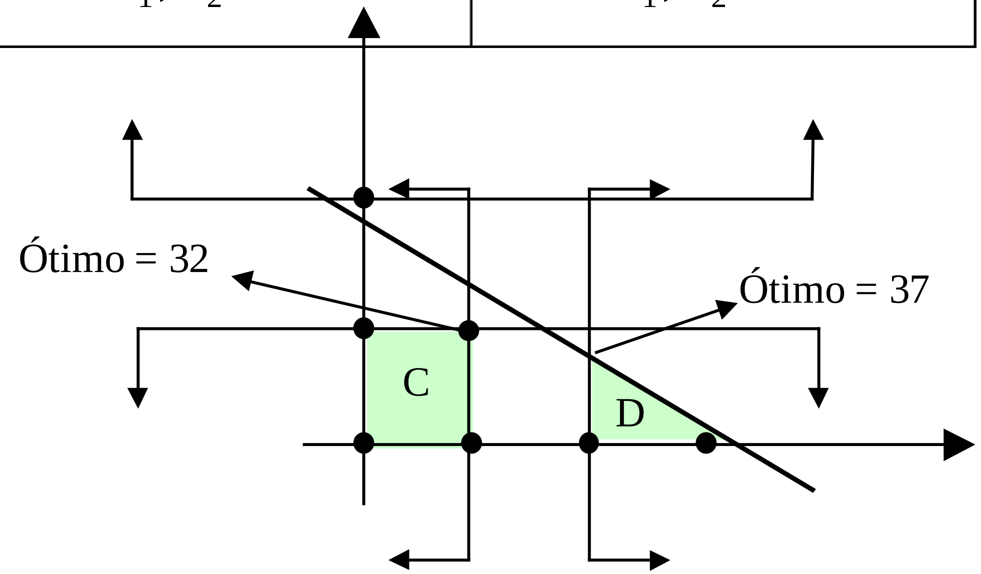
- Resolvendo o subproblema B temos Z = 37.5, x_1 =1 e x_2 =1.5

    - 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 $x_2 \geq 2$.



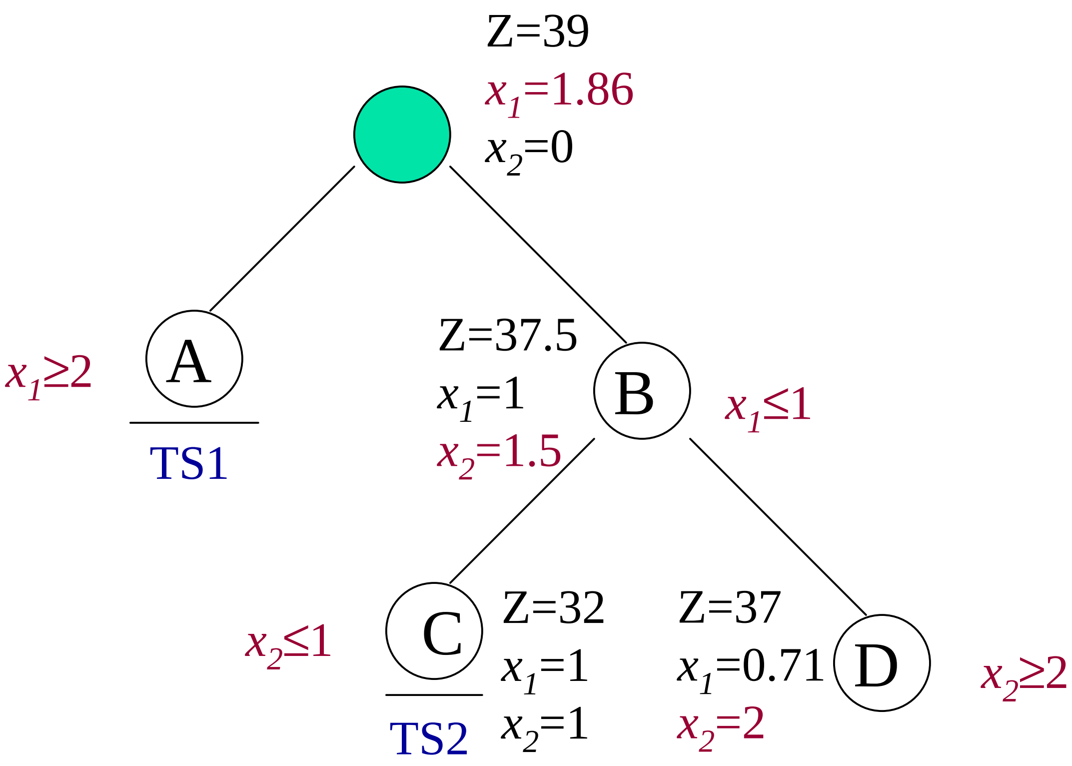
Exemplo: *Branch-and-Bound*

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



Exemplo: *Branch-and-Bound*

(TS2 - otimalidade) A solução ótima do problema relaxado é inteira.



Exemplo: *Branch-and-Bound*

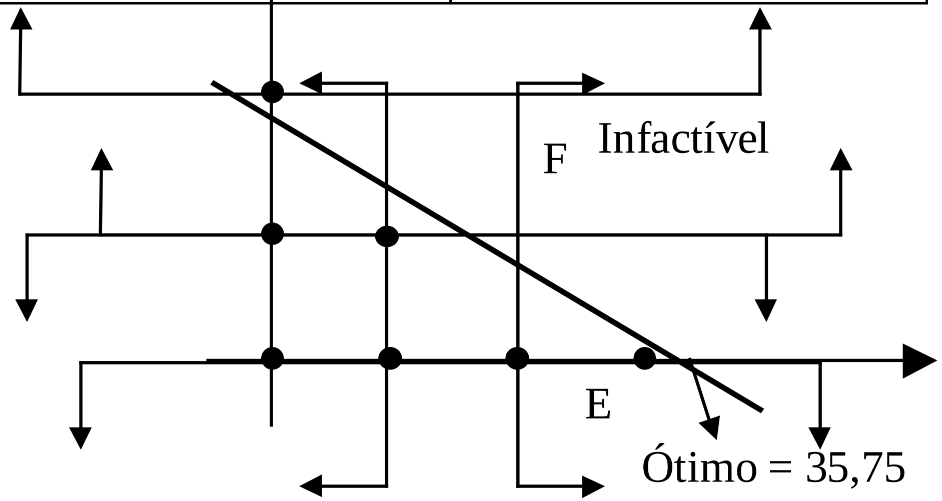
- A solução do subproblema C é igual a 32, $x_1=1$ e $x_2=1$, as duas variáveis são inteiras, logo considerando o teste de sondagem (TS2) este problema pode ser sondado por otimalidade.

- Resolvendo o subproblema D temos Z = 37, $x_1=0.71$ e $x_2=2$

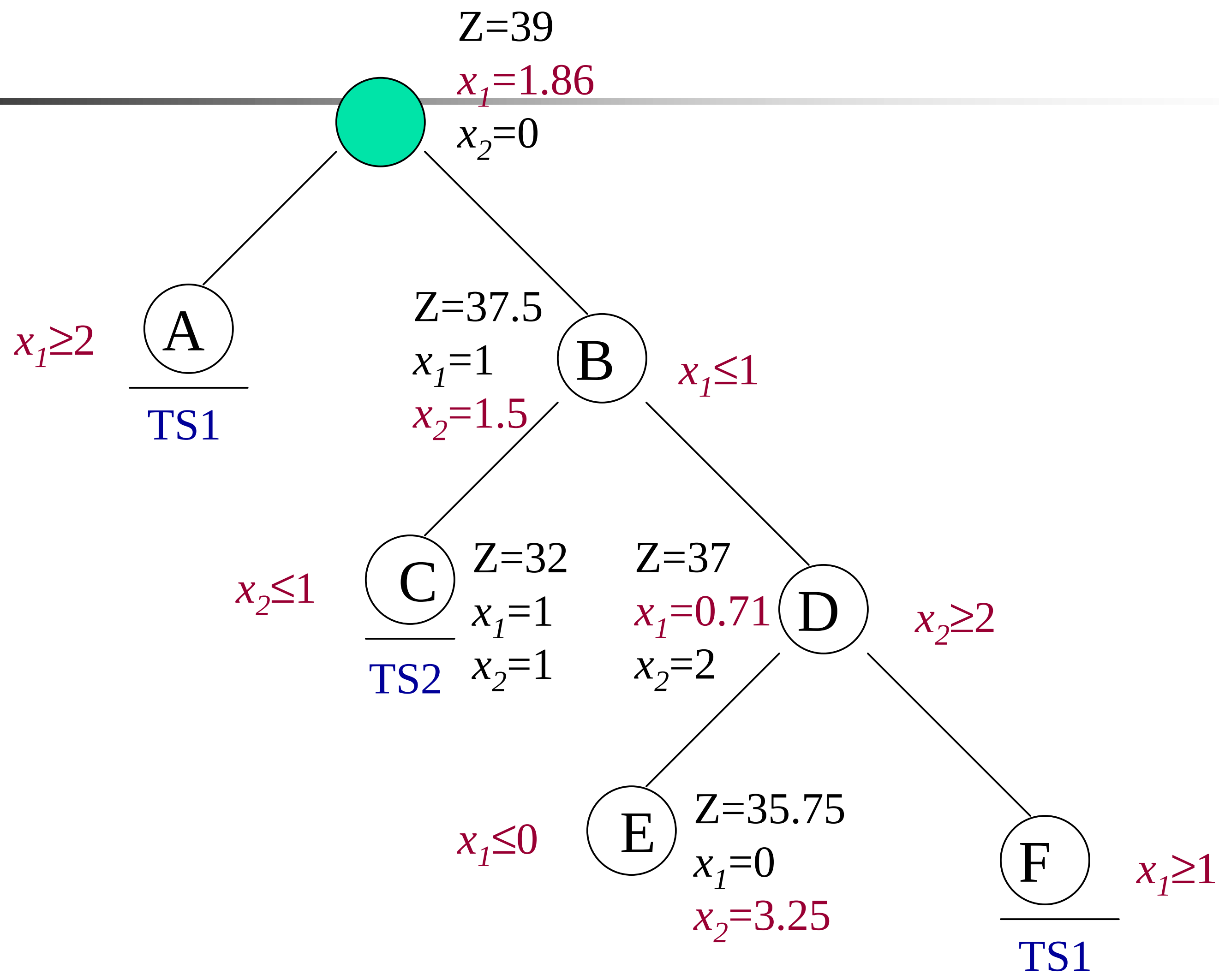
    - note que a variável x_1 novamente não é inteira, então particionamos o subproblema gerando dois novos subproblemas como mostramos a seguir

Exemplo: *Branch-and-Bound*

Suproblema E	Subproblema F
$\begin{array}{ll} max & 21x_1 + 11x_2 \\ s.a. & 7x_1 + 4x_2 \leq 13 \\ & x_1 \leq 1 \\ & x_2 \geq 2 \\ & x_1 \leq 0 \\ & x_1, x_2 \geq 0 \end{array}$	$\begin{array}{ll} max & 21x_1 + 11x_2 \\ s.a. & 7x_1 + 4x_2 \leq 13 \\ & x_1 \leq 1 \\ & x_2 \geq 2 \\ & x_1 \geq 1 \\ & x_1, x_2 \geq 0 \end{array}$



Exemplo: *Branch-and-Bound*

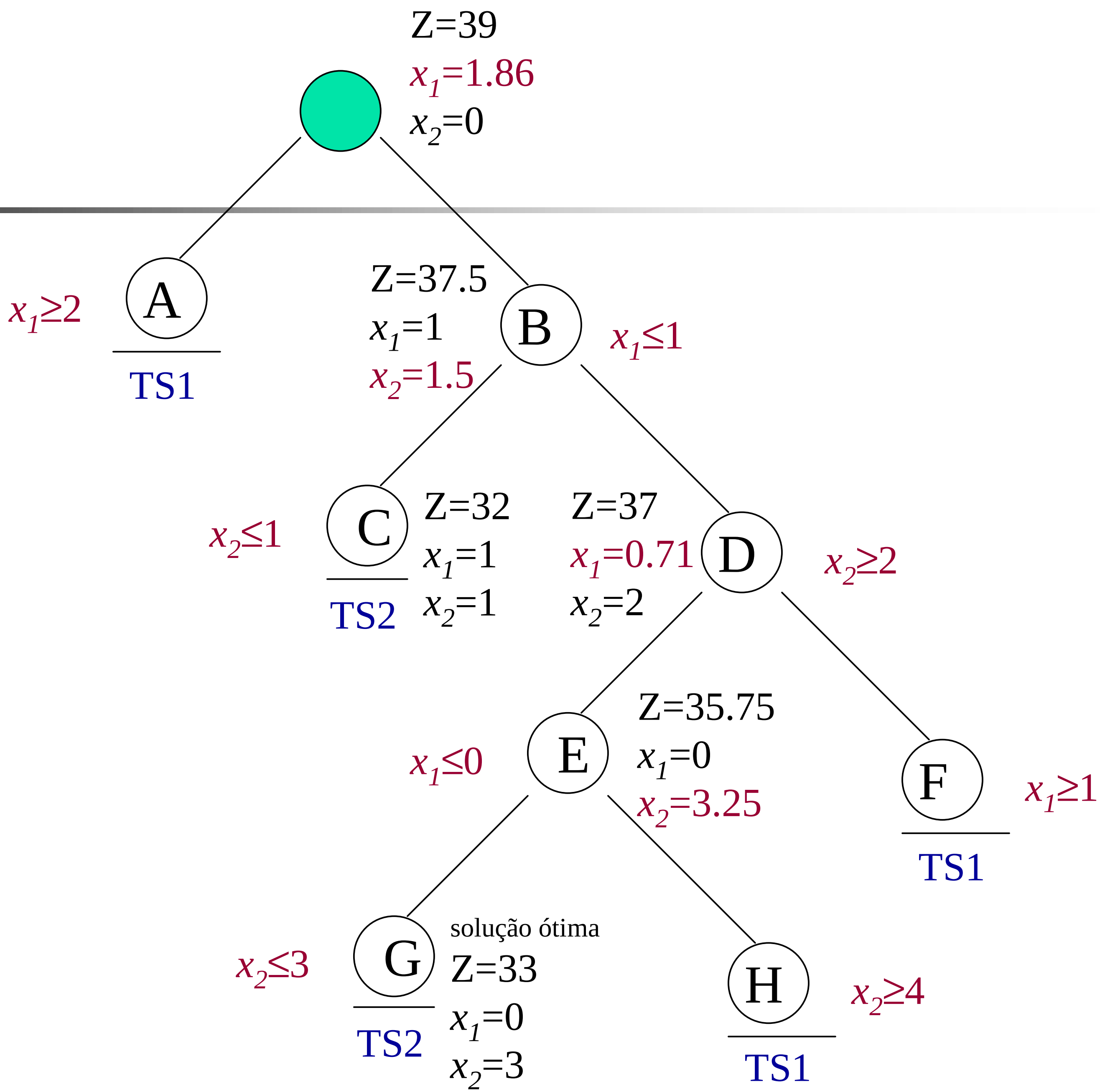


Exemplo: *Branch-and-Bound*

- O problema F é infactível, logo podemos usar TS1 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 TS2 este problema pode ser sondado.
- O subproblema H não tem solução factível e também pode ser sondado por TS1.
- 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.

Seleção de nós na árvore Branch-and-Bound (livro Pesquisa Operacional – pagina 249-251)

- Considere uma lista de nós ativos de uma árvore. Como escolher o próximo nó a ser examinado?

Existem duas regras alternativas:

- • Regras a priori (determinam previamente a ordem de escolha dos nós)
- • Regras adaptativas (determinam o nó a partir de informação dos nós ativos).

Seleção de nós na árvore Branch-and-Bound (livro Pesquisa Operacional – pagina 249-251)

- A regra a priori mais utilizada: busca em profundidade com backtracking (last-in, first-out – o último nó incluído na lista é o primeiro a ser examinado).
- Na busca em profundidade, se o nó corrente não é eliminado, o próximo nó a ser examinado é um de seus filhos.
- Backtracking - quando um nó é eliminado, retorna-se ao longo do caminho em direção ao nó raiz até encontrar o primeiro nó que tem filho a ser examinado.

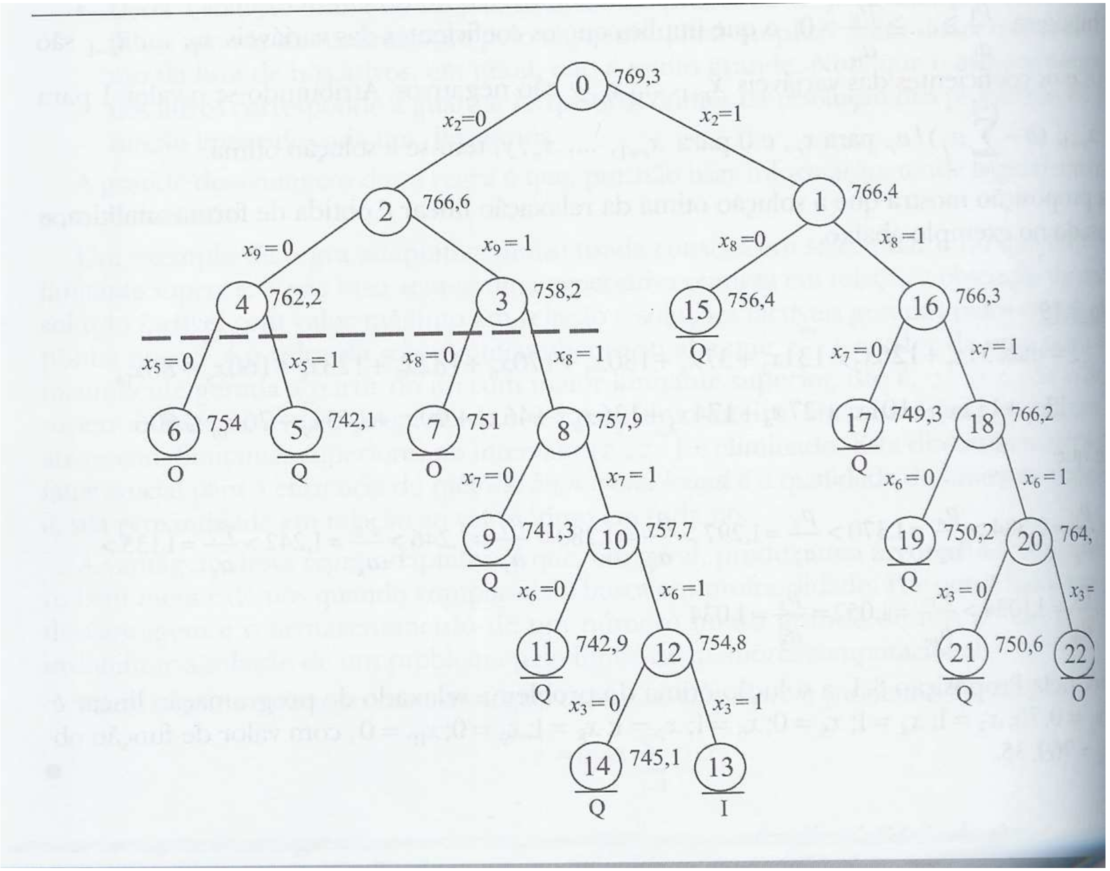
Seleção de nós na árvore Branch-and-Bound (livro Pesquisa Operacional – pagina 249-251)

- A busca em profundidade tem duas vantagens:
- 1. a experiência mostra que é mais provável encontrar soluções factíveis em níveis mais profundo em relação a raiz
- 2. a reotimização do nó filho, dada a solução ótima do nó pai, pode ser feita utilizando um algoritmo chamado dual simplex. O tamanho Maximo dos nós ativos não é muito grande.
- Desvantagem: não usa informação, tende a gerar uma arvore com muitos nós.

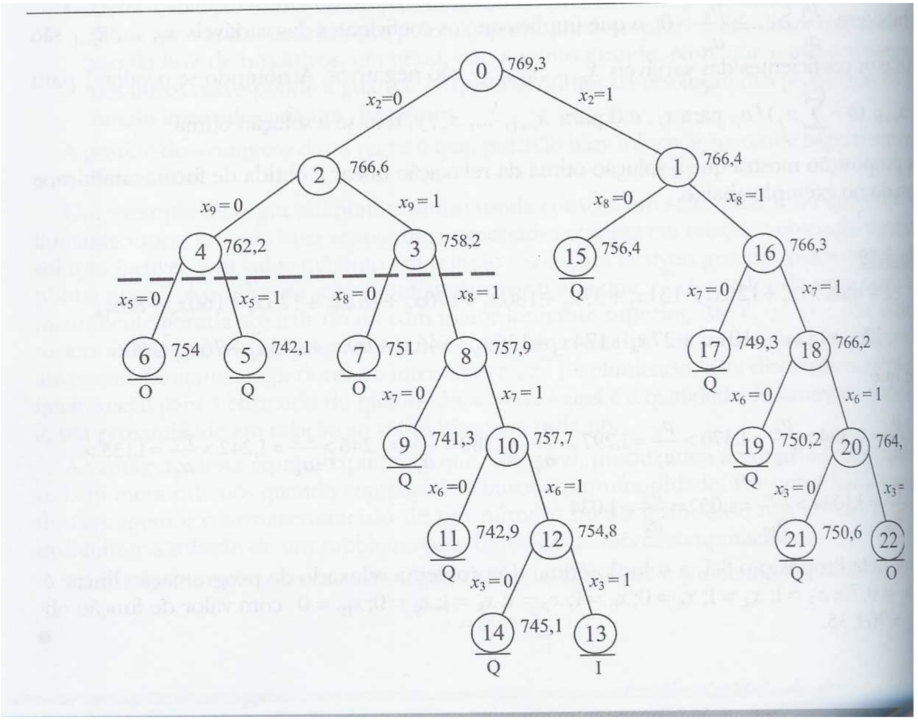
Seleção de nós na árvore Branch-and-Bound (livro Pesquisa Operacional – pagina 249-251)

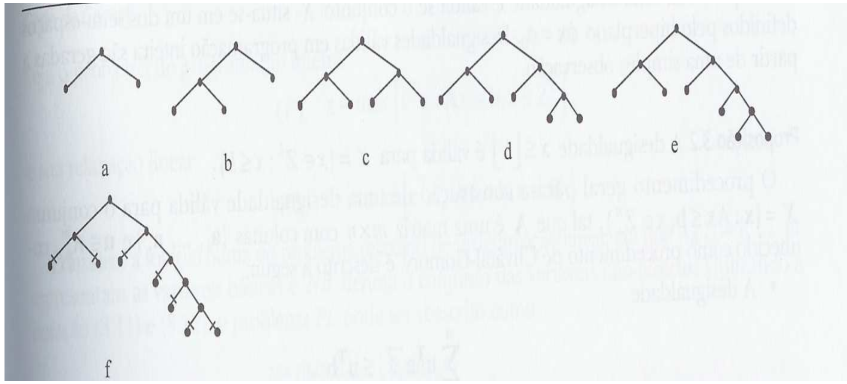
- Regra adaptativa: escolher o nó que tem o maior limitante superior (maximização).
- Vantagem: produz uma árvore com um número menor de nós (em relação a busca em profundidade) porém guarda muitos nós ativos o que pode inviabilizar a solução de um problema pelo limite de memória computacional.

Seleção de nós na árvore Branch-and-Bound –Problema da Mochila



Seleção de nós na árvore Branch-and-Bound –Problema da Mochila
EVOLUÇÃO DA BUSCA PELO MAIOR LIMITE SUPERIOR PARA O PROBLEMA DA MOCHILA (Podas -> Q-Qualidade, O-Otimalidade e I – Infactibilidade).

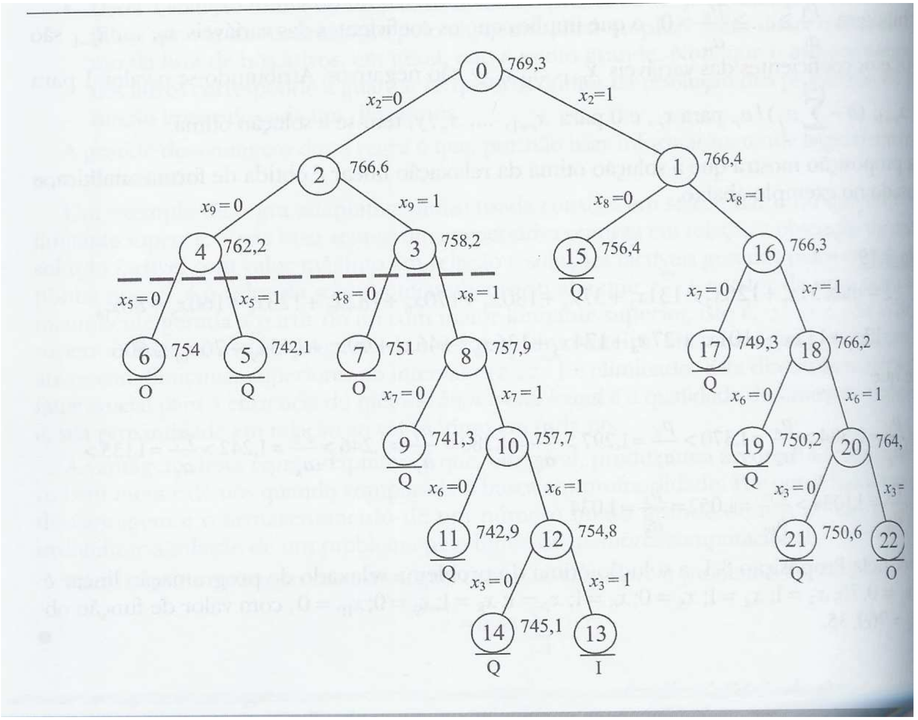


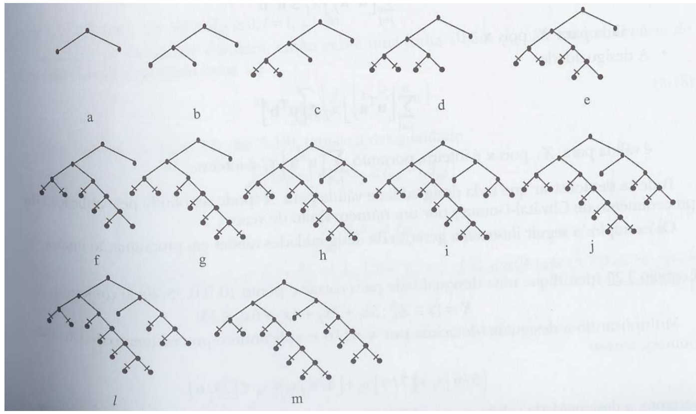


Solução ótima x1=x2=x3=1, x4=x5=0, x6=x7=x8=1, x9=x10=0,
Valor z=763

Seleção de nós na árvore Branch-and-Bound –Problema da Mochila

EVOLUÇÃO DA BUSCA EM PROFUNDIDADE PARA O PROBLEMA DA MOCHILA .





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 (algoritmo). 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.