Sumário

Seja

$$(PLI) \equiv z = \min\{c(x) : x \in S\}.$$

Ideia geral: decompor S em conjuntos "mais pequenos" e resolver o PLI nesses conjuntos.

Proposição

Seja $S = S_1 \cup S_2 \cup \ldots \cup S_{|K|}$ uma decomposição de S e seja $z^k = \min\{c(x) : x \in S_k\}, k \in K$. Então,

$$z = \min_{k \in K}\{z^k\}.$$

Algoritmos de enumeração

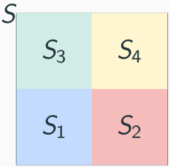
- Caso S_k não seja de resolução mais fácil pode ser decomposto, isto é,

$$S_k = S_{k1} \cup S_{k2} \cup \ldots.$$

- Este processo pode ser repetido até todos os problemas serem resolvidos.

$\Longrightarrow$ Pesquisa em árvore.

- Como decompor S em subconjuntos $S_1, \ldots, S_k$?
  - Os conjuntos $S_1, \ldots, S_k$ devem ser uma partição de S.



  - A forma mais simples é particionar S em dois subconjuntos S_1 e S_2.

Algoritmos de enumeração

Variáveis binárias

Seja x_i uma variável binária, isto é, $x_i \in \{0, 1\}$.

O conjunto S pode ser particionado em:

- $S_1 = \{x \in S : x_i = 0\}$
- $S_2 = \{x \in S : x_i = 1\}$

Variáveis inteiras

Seja x_i uma variável com valor fracionário, isto é, $x_i = \alpha \notin \mathbb{Z}$.

O conjunto S pode ser particionado em:

- $S_1 = \{x \in S : x_i \leq \lfloor \alpha \rfloor\}$
- $S_2 = \{x \in S : x_i \geq \lceil \alpha \rceil\}$

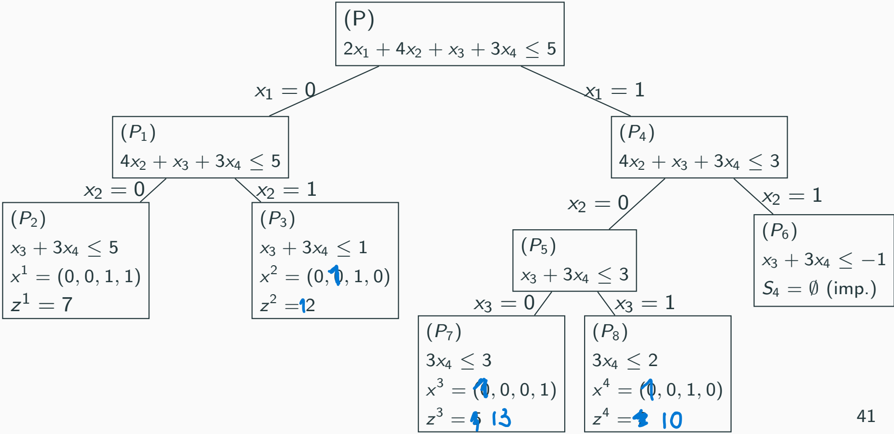
Podemos ir decompondo os conjuntos sucessivamente. $\Longrightarrow$ Árvore de enumeração.

Exemplo

$$(P) \equiv \max\ 8x_1 + 10x_2 + 2x_3 + 5x_4 \qquad \text{Problema saco-mochila}$$

$$\text{s.a:}\ \ 2x_1 + 4x_2 + x_3 + 3x_4 \leq 5$$

$$x_1, x_2, x_3, x_4 \in \{0, 1\}$$



Algoritmos de branch-and-bound

A cada nodo k está associado um problema P_k e um conjunto S_k, que é a RA de P_k.

- Se P_k é obtido de P_j através de uma ramificação diz-se que k é filho de j.
- Se P_k é obtido de P_j através de uma sequência de ramificações diz-se que k é descendente de j.
- Temos $v(P_j) \leq v(P_k)$, para todo o descendente k de j.
  - Relação válida para problemas de minimização ~~maximização~~.
- Temos $v(P_j) \geq v(P_k)$, para todo o descendente k de j.
  - Relação válida para problemas de maximização ~~minimização~~.

Nota: Cada vez que ramificamos estamos a adicionar restrições e, consequentemente, a piorar o valor da solução ótima dos subproblemas.

- Como se relaciona z com os limites nos valores de z^k?

Proposição

Seja:

- $S = S_1 \cup S_2 \cup \ldots \cup S_K$ uma decomposição de S;
- $z^k = \min\{c(x) : x \in S_k\}, k = 1, \ldots, K$;
- $\overline{z}^k$ um limite superior para z^k; e
- $\underline{z}^k$ um limite inferior para z^k.

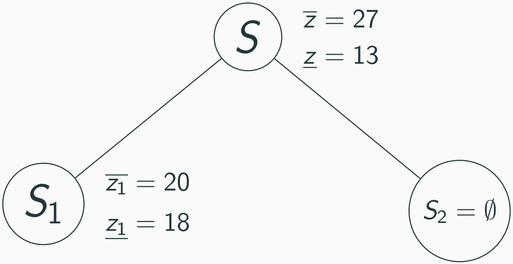
Então, $\overline{z} = \min_{k=1,\ldots,K}\{\overline{z}^k\}$ é um limite superior para z e
$\underline{z} = \overset{\text{min}}{\cancel{\max}}_{k=1,\ldots,K}\{\underline{z}^k\}$ é um limite inferior para z.

Algoritmos de enumeração

- Podemos usar a proposição anterior para reduzir o tamanho da árvore de enumeração.

- Um nodo k pode ser cancelado (não ramificado) em três situações:
  1. o problema é impossível, $S_k = \emptyset$;
  2. a solução obtida é admissível para o problema original P; ou
  3. é possível provar que P_k não contém a solução ótima (ou soluções melhores que a melhor solução conhecida para P).

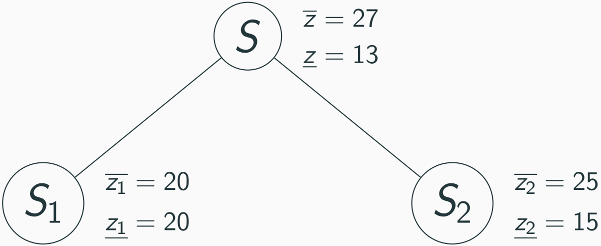
Problema de minimização



O problema P_2 é impossível, não é necessário ramificá-lo. $\Longrightarrow$ Corte por impossibilidade.

Algoritmos de enumeração

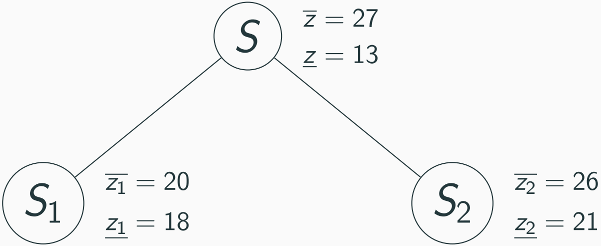
Problema de minimização



Como o limite superior e inferior de P_1 são iguais ($\overline{z}_1 = \underline{z}_1 = 20$) obtivemos o seu valor ótimo e por isso não é necessário ramificá-lo. $\Longrightarrow$ Corte por otimalidade.

Algoritmo de branch-and-bound

Problema de minimização



O valor ótimo de P_2 é pelo menos 21, enquanto o valor ótimo de S_1 é no máximo 20. Assim, como a partir de P_2 vamos sempre obter soluções piores que as de P_1 podemos não explorá-lo mais. $\Longrightarrow$ Corte por limite.

Algoritmo de branch-and-bound

- É um algoritmo de enumeração!
- Como resolver os subproblemas?
- ▶ É resolvida a sua relaxação linear.
- Notação:

$(P) \equiv z = \min\{c(x) : x \in X\}$

$(P_k) \equiv z_k = \min\{c(x) : x \in X_k\}$

L Lista de problemas por resolver

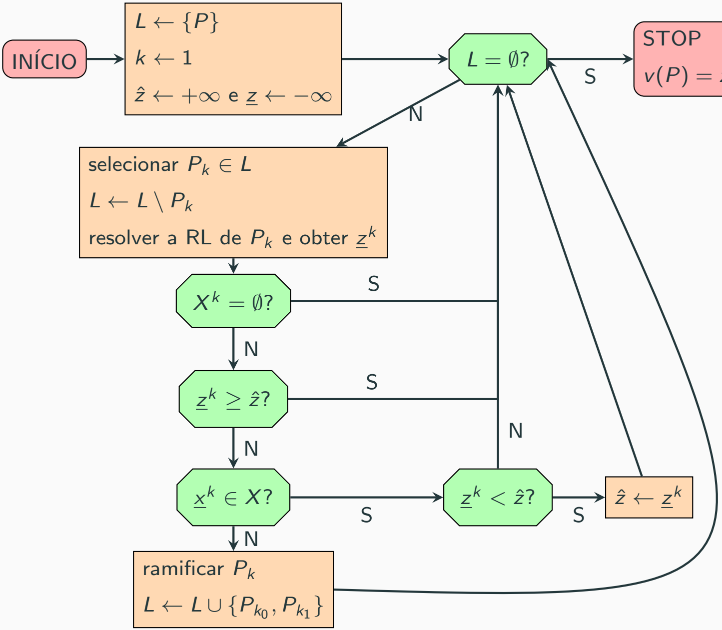
$\hat{z}$ valor da melhor SA conhecida - incumbente.

Caso não tenha sido determinada nenhuma SA, $\hat{z} = +\infty$.

$\underline{z}^k$ minorante de z^k.

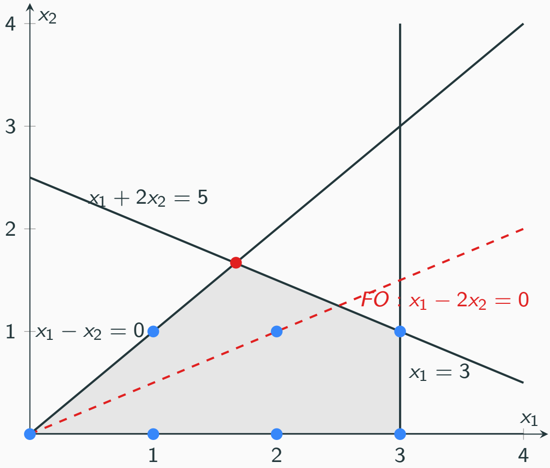
Temos $\underline{z}^k = +\infty$ se P_k impossível e $\underline{z}^k = z^k$ caso contrário.

Algoritmo de branch-and-bound



Exemplo

$$(P) \equiv \min z = x_1 - 2x_2$$
$$s.a: x_1 - x_2 \geq 0$$
$$x_1 + 2x_2 \leq 5$$
$$x_1 \leq 3$$
$$x_1, x_2 \geq 0 \text{ e inteiros}$$



Resolvendo a relaxação linear:

$x_{RL} = (\frac{5}{3}, \frac{5}{3})$ e $\underline{z} = \frac{5}{3} - 2 \times \frac{5}{3} = -\frac{5}{3}$

Exemplo

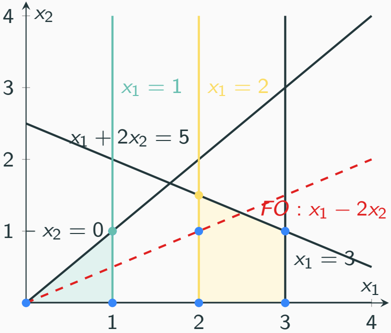
Atualiza tudo após resolver (P_1)

(P)
$\underline{z} = -\frac{5}{3}$
$\hat{z} = +\infty \; \boxed{= -1}$

$x_1 \leq \lfloor \frac{5}{3} \rfloor = 1$ | $x_1 \geq \lceil \frac{5}{3} \rceil = 2$

(P_1)
$\underline{z_1} = -1$
$\hat{z}_1 = -1$

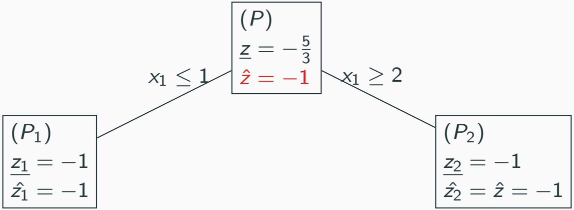
(P_2)
$\underline{z_2} = -1$
$\hat{z}_2 = -1$



- (P_1) : $x_1^* = (1, 1)$ e $\underline{z_1} = \overline{z_1} = -1$
- (P_2) : $x_2^* = (2, \frac{3}{2})$ e $\underline{z_2} = -1$

■ Não é preciso ramificar mais a partir de P_1 pois já encontramos uma SA. Podemos atualizar o majorante.

Exemplo



Como o minorante de P_2 é igual a $\hat{z}$ podemos cancelar o nodo pois não vai conter a SO. Todas as soluções de descendentes de P_2 vão ~~ser piores que~~ -1.

ter valor pior
ou igual a -1

Algoritmo de branch-and-bound

- Estratégias de ramificação e pesquisa
  - Como ramificar?
    - Escolher a variável fracionária com valor α mais próximo de $\lfloor \alpha \rfloor + \frac{1}{2}$ na relaxação linear (variável "mais fracionária").
    - Escolher a variável fracionária com o índice mais baixo.
  - Como selecionar o subproblema?
    - Pesquisa em profundidade.
      Escolher um dos últimos problemas criados. $\Longrightarrow$ Descer rapidamente na árvore de pesquisa de modo a encontrar rapidamente uma solução admissível.
    - Seleção do melhor nodo.
      Escolher o problema em aberto com maior valor de $\underline{z}^k$.

Algoritmo de branch-and-bound

- Estratégias de redução do número de nodos da árvore
  - Utilizar heurísticas para determinar soluções admissíveis (majorantes).
  - Se os coeficientes da função objetivo são inteiros, z só toma valores inteiros para soluções admissíveis. Neste caso os minorantes $\underline{z}^k$ podem ser substituídos por $\lceil \underline{z}^k \rceil$.
  - Tentar melhorar a qualidade dos minorantes.

■ O algoritmo de branch-and-bound pode ser utilizado como uma heurística.

$\Longrightarrow$ Por exemplo, parando o algoritmo quando uma SA for encontrada.