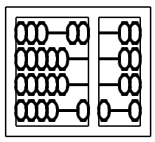


Universidade Estadual de Campinas - UNICAMP
Instituto de Computação - IC
MO824 - Tópicos em Otimização Combinatória

Branch-and-Bound

Cid Carvalho de Souza

Outubro de 2006

Branch-and-Bound: introdução

$$z = \max\{cx : x \in S\}$$

Idéia do algoritmo: (paradigma de *divisão e conquista*)

- particionar o conjunto de soluções S em subconjuntos disjuntos;
- resolver o problema para estas instâncias menores;
- combinar as soluções dos subproblemas para obter a solução do problema original.

Proposição 1

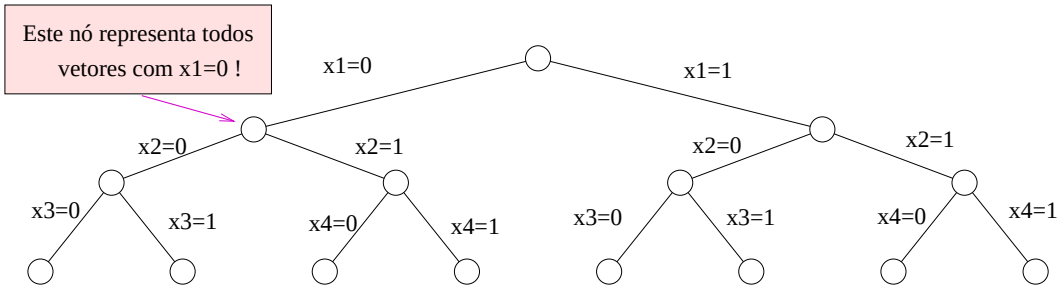
Seja $\{S_1, S_2, \ldots, S_K\}$ uma decomposição de S, i.e. $S = S_1 \cup S_2 \cup \ldots \cup S_K$. Se $z^k = \max\{cx : x \in S_k\}$, $k = 1, \ldots, K$, então $z = \max_k z^k$.

Representação do fluxo do algoritmo

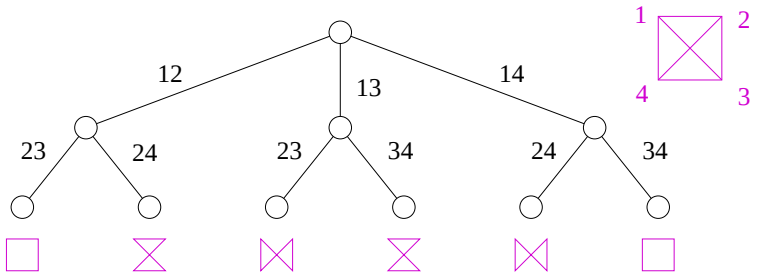
árvore de enumeração !

Branch-and-Bound: introdução (cont.)

- Exemplo 1: $S = \{0,1\} = \mathbb{B}^3$



- Exemplo 2: S é o conjunto de conjuntos de arestas que formam ciclos hamiltonianos no K^4.



Branch-and-Bound: introdução (cont.)

Enumeração implícita

- a árvore de enumeração pode ficar muito grande !
- Idéia: enumerar a árvore de forma *implícita*.
- Como ? *usar limitantes* para podar ramos da árvore que não contém solução ótima.

Proposição 2

Seja $\{S_1, S_2, \ldots, S_K\}$ uma decomposição de S e $z^k = \max\{cx : x \in S_k\}$, $k = 1, \ldots, K$. Seja ainda $\overline{z}^k$ ($\underline{z}^k$) um limitante superior (inferior) para z^k. Então os valores de $\overline{z}$ e $\underline{z}$ abaixo são limitantes superior e inferior de z, respectivamente

$$\overline{z} = \max_k\{\overline{z}^k\} \qquad \text{e} \qquad \underline{z} = \max_k\{\underline{z}^k\}.$$

Podando a Árvore de Enumeração

Definição

ramificação ou *branching* em um nó.

Definição

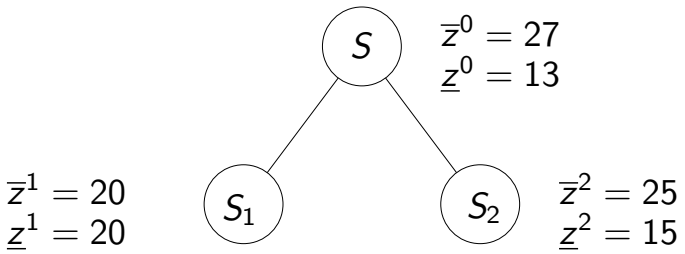
um *nó amadurecido* (*pruned*) não mais sofrerá *branching*.

Objetivo: amadurecer os nós da árvore usando limitantes !

Em que casos isto pode ser feito ?

- Caso 1: poda por *otimalidade*
- Caso 2: poda por *limitante*
- Caso 3: poda por em inviabilidade

Podando a Árvore de Enumeração por *otimalidade*

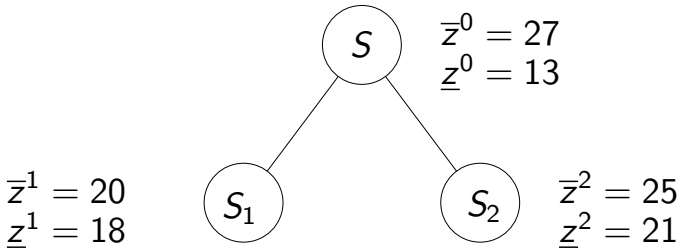


$$\overline{z} = \max\{\overline{z}^1, \overline{z}^2\} = \max\{20, 25\} = 25$$
$$\underline{z} = \max\{\underline{z}^1, \underline{z}^2\} = \max\{20, 15\} = 20$$

Em S_1 é conhecida uma solução de custo 20 $(= \underline{z}^1)$ e ela é *ótima* pois o custo de qualquer solução de S_1 é ≤ 20 $(= \overline{z}^1)$.

Logo S_1 deve ser amadurecido !

Podando a Árvore de Enumeração por *limitante*



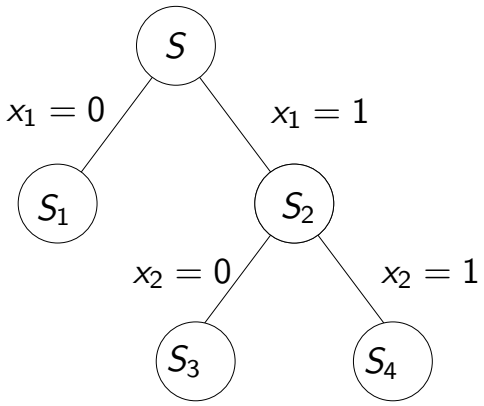
$$\overline{z} = \max\{\overline{z}^1, \overline{z}^2\} = \max\{20, 25\} = 25$$
$$\underline{z} = \max\{\underline{z}^1, \underline{z}^2\} = \max\{18, 21\} = 21$$

A solução de maior custo em S_1 tem custo limitado a 20 $(= \overline{z}^1)$ mas, em S_2 já é conhecida uma solução de custo 21 $(= \underline{z}^2)$.

Logo S_1 deve ser amadurecido !

Podando a Árvore de Enumeração por *inviabilidade*

Mochila 0–1: $8x_1 + 5x_2 + 3x_3 + 3x_4 \leq 12$



O nó S_4 representa o subconjunto de todas as soluções que contêm o item 1 *e* o item 2. Mas estes subconjunto é vazio já que a soma dos pesos destes itens ultrapassa a capacidade da mochila.

Logo S_4 deve ser amadurecido !

Podando a Árvore de Enumeração

Portanto, NÃO podem ser amadurecidos os nós onde $\overline{z}^i \neq \underline{z}^i$ E $\overline{z}^i > \underline{z}$ E $S^i \neq \emptyset$

- Nós ativos são aqueles que **NÃO** foram amadurecidos **E** que ainda não foram ramificados.
- Cálculos de limitantes:
  - primais: heurísticas e soluções viáveis obtidas durante a enumeração
  - duais: relaxações

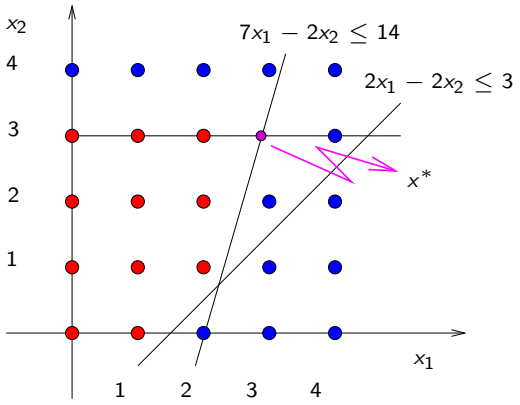
Algoritmo de *Branch-and-Bound*

Considerações no projeto do algoritmo

- Limitantes:
  fáceis de calcular mas fracos $\times$ fortes mas computacionalmente caros
- Decomposição do espaço de soluções:
  - Decompor em duas ou em mais partes ?
  - Fixar a regra de decomposição ou deixá-la em função dos limitantes obtidas no correr da execução ?
- Ordem de percurso da árvore:
  - Profundidade ?
  - Largura ?
  - Usar nó com melhor limitante dual ?

Exemplo de *B&B* usando PL

$$
\begin{array}{ll}
\max & z = 4x_1 - x_2 \\
s.a. & 7x_1 - 2x_2 \leq 14, \\
& 2x_1 - 2x_2 \leq 3, \\
& x_2 \leq 3, \\
& x_1 \text{ e } x_2 \text{ inteiros}
\end{array}
$$



Limitantes

- solução do LP: $x_1^* = 20/7$, $x_2^* = 3$.
- Limitante superior: $\overline{z} = z_{LP} = 59/7$.
- Limitante inferior: $\underline{z} = -\infty$ (convenção !)

Branching

Escolher variável fracionária x_j^* e fazer:

$$
\begin{aligned}
S_1 &\doteq S \cap \{x \in \mathbb{R}^n : x_j \leq \lfloor x_j^* \rfloor\} \\
S_2 &\doteq S \cap \{x \in \mathbb{R}^n : x_j \geq \lceil x_j^* \rceil\}
\end{aligned}
$$

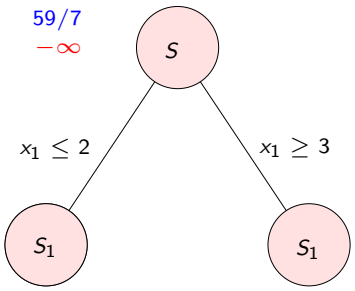
Por quê esta escolha ?

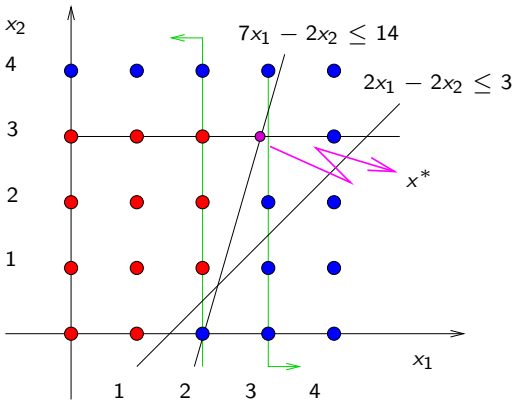
- $S_1 \cup S_2 = S$ e $S_1 \cap S_2 = \emptyset$: partição de S !
- x^* não é viável nem em S_1 e nem em S_2.
  Logo, na ausência de degenerescência, $\max\{\overline{z}_1, \overline{z}_2\} < \overline{z}$.
  Ou seja, o limitante superior vai cair !

No exemplo ...

$$
\begin{aligned}
S_1 &= S \cap \{x \in \mathbb{R}^2 : x_1 \leq 2\} \\
S_2 &= S \cap \{x \in \mathbb{R}^2 : x_1 \geq 3\}
\end{aligned}
$$

Árvore de enumeração





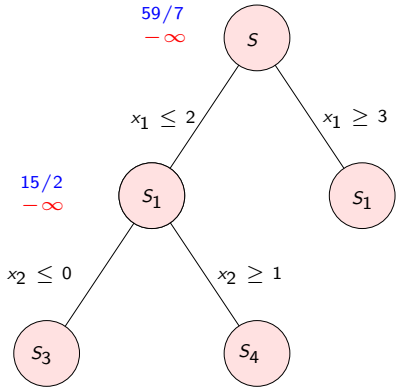
Cálculo dos limitantes

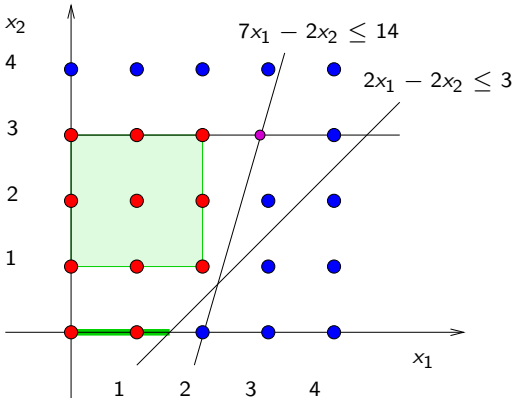
- Nó S_1: (usar o dual simplex !)
  $z_1^{LP} = \max\{4x_1 - x_2 : 2x_1 - 2x_2 \leq 3, 7x_1 - 2x_2 \leq 14, x_2 \leq 3, x_1 \leq 2\}$,
  $x_1^* = 2$ e $x_2^* = 1/2$, $z_1^{LP} = 15/2$.
- Nó S_2: podado por inviabilidade !

Novo *branching*

$$S_3 = S \cap \{x \in \mathbb{R}^2 : x_2 \leq 0\}$$
$$S_4 = S \cap \{x \in \mathbb{R}^2 : x_2 \geq 1\}$$

Árvore de enumeração





Cálculo dos limitantes

- Nó S_4:
  $z_4^{LP} = \max\{4x_1 - x_2 : 2x_1 - 2x_2 \leq 3, 7x_1 - 2x_2 \leq 14, x_2 \leq 3, x_1 \leq 2, x_2 \geq 1\}$,

  $x_1^* = 2$ e $x_2^* = 1$, $z_4^{LP} = 7$.
  Solução inteira viável: poda por otimalidade !
  Atualiza $\underline{z}$ para 7.
- Nó S_3:
  $z_3^{LP} = \max\{4x_1 - x_2 : 2x_1 - 2x_2 \leq 3, 7x_1 - 2x_2 \leq 14, x_2 \leq 3, x_1 \leq 2, x_2 \leq 0\}$,

  $x_1^* = 3/2$ e $x_2^* = 0$, $z_3^{LP} = 6 = \overline{z}^3$.
  Mas, neste caso, $\overline{z}^3 < \underline{z} = 7$
  S_3 deve ser podado por limitante !

  Lista de nós ativos está vazia: algortimo pàra !

Aspectos práticos de implementação

Informações a armazenar

- só aquelas referentes aos nós ativos;
- para cada nó:
  - limites superiores e inferiores *das variáveis*;
  - o limitante superior $\overline{z}^k$;
  - usualmente guarda-se a base ótima para acelerar a reotimização;
- o limitante inferior global $\underline{z}$ (para fazer poda por limitante);

Cálculo dos limitantes

- superior: programação linear;
- inferior: heurísticas (arredondamento);

Branching

Usualmente feito sobre a variável *"mais fracionária"*.
Alternativa: estimar custo para *"tornar"* a variável inteira.

Escolha do próximo nó a explorar

- DFS: vantagem é achar solução inteira mais rápido.
  Nota: LPs são resolvidos mais rápido pois as bases ótimas ficam na memória (dual simplex);
- *Best bound*: os nós ativos são mantidos em uma fila de prioridades, explorando sempre o nó com melhor limitante dual !

> **O objetivo desta estratégia é minimizar o número de nós explorados. Note que ela nunca explorará nós com limitantes duais piores do que o valor da solução ótima !**

Usando sistemas comerciais de PLI

Características

1. Pré-processamento;
2. Diferentes tipos de algoritmos de Programação Linear:
   - Pontos interiores: bom para a resolução do primeiro LP de modelos grandes;
   - Simplex: Primal e Dual (reotimização);
3. Escolha (limitada) de regras de *branching* e seleção de nós da árvore de enumeração;
4. Possibilidade de uso de *prioridades* para *branching de variáveis*;
5. *Generalized Upper Bound branching*;
6. Heurísticas primais: diversas (assunto em moda !);
7. Fixação de variáveis por custos reduzidos.

GUB branching

Quando e como se aplica:

- O modelo contém restrições (GUB) da forma: $\sum_{i=1}^{k} x_i = 1$ com $x_i \in \{0,1\}$ para todo $i = 1,\ldots,k$;
- Considere a solução fracionária ótima da relaxação corrente dada por x^* e assuma que $0 < x^*_\ell < 1$ para algum $\ell \in \{1,\ldots,k\}$;
- *Branching usual*: $S_1 = S \cap \{x_\ell = 0\}$ e $S_2 = S \cap \{x_\ell = 1\}$
  Problema: árvore de enumeração "desequilibrada";
- *GUB branching*: seja $r \doteq \min\{t : \sum_{i=1}^{t} x^*_i > 1/2\}$.
  Defina: $S_1 = S \cap \{\sum_{i=1}^{r} x_i = 1\}$ e $S_2 = S \cap \{\sum_{i=r+1}^{k} x_i = 1\}$;
- O *GUB branching* costuma funcionar bem para modelos com muitas restrições do tipo GUB. Exemplo: *set partitioning*.

Fixação de variáveis

O caso dos PLIs 0-1

- O problema: $z^* = \max\{cx : Ax = b, x \in \mathbb{B}^n\}$;
- Solução básica da relaxação linear:

$$\begin{aligned} x_B &= B^{-1}b - B^{-1}N_0 x_{N_0} - B^{-1}N_1 x_{N_1}, \\ z &= c_B^{-1}b + \underbrace{(c_{N_0} - c_B^{-1}N_0)x_{N_0}}_{\sum_{j\in N_0}\overline{c}_j x_j} + \underbrace{(c_{N_1} - c_B^{-1}N_1)x_{N_1}}_{\sum_{j\in N_1}\overline{c}_j x_j}, \end{aligned}$$

  onde $x_{N_0} = 0$ e $x_{N_1} = \mathbb{1}$ são as variáveis não-básicas;
- Seja $\underline{z}$ um limitante primal (inferior) para z^*:
  - $j \in N_0$: se $z + \overline{c}_j < \underline{z}$ então toda solução com custo melhor (maior) que $\underline{z}$ satisfaz $x_j = 0$;
  - $j \in N_1$: se $z - \overline{c}_j < \underline{z}$ então toda solução com custo melhor (maior) que $\underline{z}$ satisfaz $x_j = 1$.

Fixação de variáveis: explicação

- Reescrevendo o problema em termos das variáveis não-básicas:

$$\begin{aligned} \max \quad & z = \underbrace{c_B^{-1} b}_{z_0} + \underbrace{(c_{N_0} - c_B^{-1} N_0) x_{N_0}}_{\sum_{j \in N_0} \overline{c}_j x_j} + \underbrace{(c_{N_1} - c_B^{-1} N_1) x_{N_1}}_{\sum_{j \in N_1} \overline{c}_j x_j}, \\ \text{s.a.} \quad & x_B + B^{-1} N_0 x_{N_0} + B^{-1} N_1 x_{N_1} = B^{-1} b \\ & \mathbf{0}_n \leq (x_B, x_{N_0}, x_{N_1}) \leq \mathbb{1}_n. \end{aligned}$$

- Se $j \in N_0$ então $x_j = 0$ e $\overline{c}_j \leq 0$.
  Se x_j aumentar de valor, algumas variáveis $k \in N_0$ ($\overline{c}_k \leq 0$) subirão de valor e algumas variáveis $\ell \in N_1$ ($\overline{c}_\ell \geq 0$) descerão de valor.
  Como as variáveis referentes a base B tem custo reduzido nulo, o custo da solução deverá ser, no máximo, igual a $z_0 + \overline{c}_j$.
  Portanto, se $z_0 + \overline{c}_j \leq \underline{z}$, não pode haver uma solução de custo melhor (maior) que $\underline{z}$ satisfazendo $x_j = 1$.
- O caso $j \in N_1$ é análogo. $\square$

Algumas opções interessantes dos resolvedores

- alterar a estratégia de *pricing* do simplex (escolha da variável que entrará na base);
- usar *strong branching*.

Dificuldades ...

O que fazer se após um tempo *"aceitável"* de computação,

- não houver solução viável disponível ? ou
- o *"gap"* entre os limitantes primal e dual for grande ? ou
- o algoritmo pàra por falta de memória devido ao grande número de nós ativos ?

Alternativas:

Melhorar os limitantes !

- Primais: heurísticas.
- Duais: mudar o modelo !