Programação Inteira - método de partição e avaliação

Investigação Operacional

J.M. Valério de Carvalho
vc@dps.uminho.pt

Departamento de Produção e Sistemas
Escola de Engenharia, Universidade do Minho

11 de dezembro de 2014

Programação Inteira (PI) - método de partição e avaliação

antes

- Além de planos de corte, os *solvers* de PI usam partição e avaliação.

Guião

- Estratégia do método: pesquisar subdomínios da relaxação linear$^{(*)}$, criados por uma regra de partição,
- e analisá-los, determinando o óptimo com Programação Linear (PL), para avaliar se podem conter a solução inteira óptima.
- Os subdomínios são representados por nós de uma árvore,
- que é pesquisada até encontrar a solução inteira óptima ou concluir que não existem soluções inteiras admissíveis.
- Outras designações: *branch-and-bound*, *divide-and-conquer*.

depois

- Os modelos de problemas reais complexos envolvem normalmente variáveis inteiras ou binárias.

$^{(*)}$ a *relaxação linear* do modelo de PI é o modelo de PL que resulta de relaxar (retirar) as restrições de integralidade.

Programação Inteira (PI)

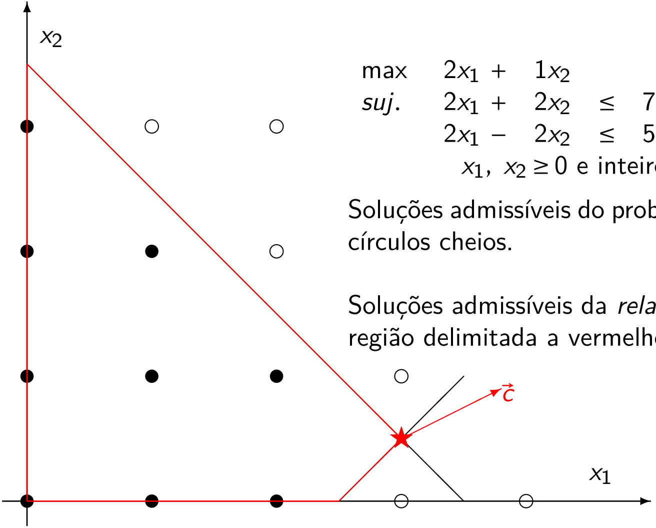
- problema de programação linear inteiro puro:

$$\begin{aligned} \max z_I &= cx \\ \text{suj. a} \quad & Ax = b \\ & x \geq 0 \text{ e inteiro} \end{aligned}$$

- problema de programação linear inteiro misto:

$$\begin{aligned} \max z_I &= cx + dy \\ \text{suj. a} \quad & Ax + Dy = b \\ & x \geq 0 \text{ e inteiro}, y \geq 0 \end{aligned}$$

- *Restrições de integralidade:* declaram que as variáveis de decisão x são inteiras.
- As restrições x binário também são restrições de integralidade.



$$
\begin{array}{llllll}
\max & 2x_1 & + & 1x_2 & & \\
suj. & 2x_1 & + & 2x_2 & \leq & 7 \\
 & 2x_1 & - & 2x_2 & \leq & 5 \\
 & & & x_1,\ x_2 \geq 0 \text{ e inteiros} & &
\end{array}
$$

Soluções admissíveis do problema inteiro: círculos cheios.

Soluções admissíveis da *relaxação linear*: região delimitada a vermelho.

A solução óptima da *relaxação linear* é o ponto $(x_1, x_2)^t = (3, 0.5)^t$, assinalado com $\bigstar$.

Algoritmo de partição e avaliação

- A Partição cria subdomínios, cada um deles representado por um nó de uma árvore.

Algoritmo de partição e avaliação (informal)

- Input: Problema de Programação Inteira (PI)
- Output: x_I^* (solução óptima inteira)

Início: $x_I^* = \emptyset$; ListaNós = {relaxação linear do modelo PI};
repetir
    seleccionar e retirar nó de ListaNós;
    determinar solução óptima de Prog.Linear do nó (x_{PL});
    se (x_{PL} é inteira e melhor do que x_I^*)
        actualizar x_I^*;
    se (x_{PL} é fraccionária e com valor melhor do que x_I^*)
        fazer Partição e adicionar nós a ListaNós;
até (ListaNós vazia);

Conteúdo

- Método de Partição e Avaliação
  - Partição
  - Avaliação
- Regras de pesquisa: em largura e em profundidade
- Aplicação a um exemplo
- Limite superior e limite inferior para o valor do óptimo
- Notas

Partição do domínio

- A Partição visa eliminar soluções fraccionárias (não desejadas).

- A partição de um domínio cria *subdomínios* disjuntos;
- cada subdomínio é obtido adicionando uma *restrição de partição* e é representado por um *nó* de uma *árvore,*
- que resulta da partição sucessiva dos subdomínios.
- *Regra de partição:* a criação dos *nós filhos* de um *nó pai* deve sempre obedecer ao seguinte:
  - a solução fraccionária do domínio do nó pai não pertence ao subdomínio de nenhum nó filho,
  - nenhuma solução inteira (admissível) é eliminada.

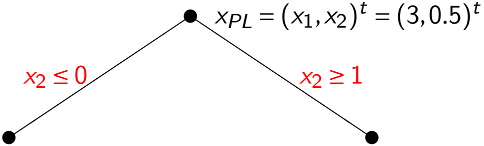
Exemplo: regra de partição dicotómica: $x_j \leq \lfloor x_j \rfloor$ e $x_j \geq \lceil x_j \rceil$:

- dado um nó pai com uma solução fraccionária,
- seleccionar a variável x_j fraccionária com menor índice,
- criar 2 nós filhos, usando as restrições de partição $x_j \leq \lfloor x_j \rfloor$ e $x_j \geq \lceil x_j \rceil$.

Exemplo: partição no nó raiz da árvore de pesquisa

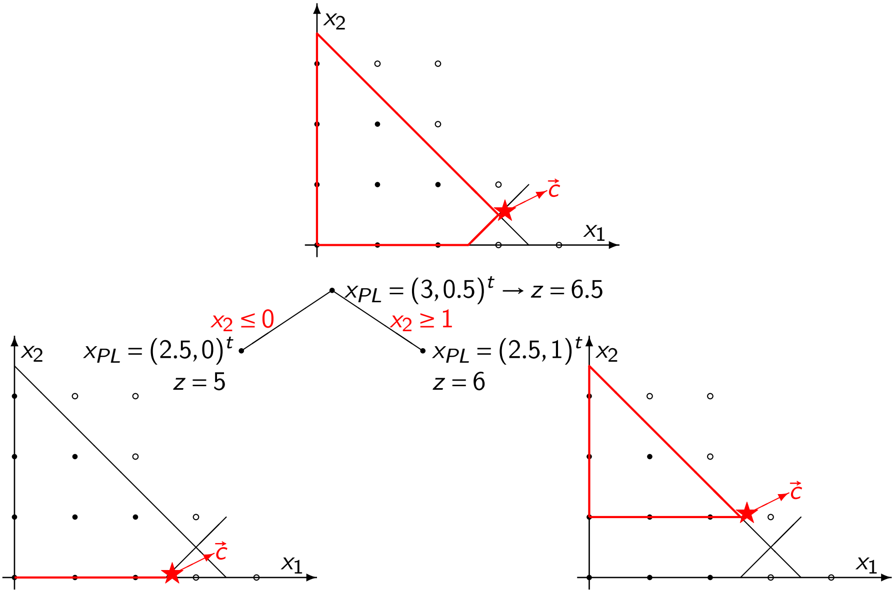
Solução óptima da relaxação linear (domínio original) é fraccionária

- seleccionar a variável fraccionária $x_2 = 0.5$,
- criar 2 nós filhos, cujos domínios são, respectivamente:
  - nó filho à esquerda: a parte do domínio original em que $x_2 \leq \lfloor 0.5 \rfloor$, e
  - nó filho à direita: a parte do domínio original em que $x_2 \geq \lceil 0.5 \rceil$.



$\lfloor \cdot \rfloor$ e $\lceil \cdot \rceil$: operadores arredondamento para baixo e para cima, resp..

Exemplo: sol. óptimas de PL do nó raiz e dos nós filhos



Pesquisa da árvore e solução incumbente

- À medida que se desce na árvore, da *raiz* para *nós* a um maior *nível de profundidade*, as restrições de partição conjugam-se, e o subdomínio dos nós é cada vez menor (pode ser o conjunto vazio).
- Na *pesquisa da árvore*, à medida que se visitam os nós, tipicamente encontram-se soluções inteiras sucessivamente melhores.

- O *lpsolve* mostra colunas com: MILP feasible (primeira solução inteira admissível), MILP better (uma ou mais soluções inteiras sucessivamente melhores), result (solução inteira óptima).

Solução incumbente:

- é a melhor solução inteira encontrada até um dado passo da pesquisa (x_{SI}) com valor de função objectivo z_{SI}.

- No diapositivo seguinte, apresenta-se o procedimento de avaliação para problemas de maximização. Em problemas de minimização, os operadores de comparação de valor ($<, >, \leq, \geq$) são ao contrário.

Avaliação do nó (problemas de maximização)

- Na Avaliação, opta-se por abandonar o nó ou por fazer partição.

Início) Determinar a sol. óptima de PL do domínio do nó ($x_{PL} \rightarrow z_{PL}$) :

- se for inteira, é a melhor solução inteira existente no domínio do nó;
- se for fraccionária, pode haver na *subárvore (descendentes do nó)* uma solução inteira, mas terá um valor de função objectivo $\leq z_{PL}$.

Opção 1) abandonar o nó (podar a subárvore) se:

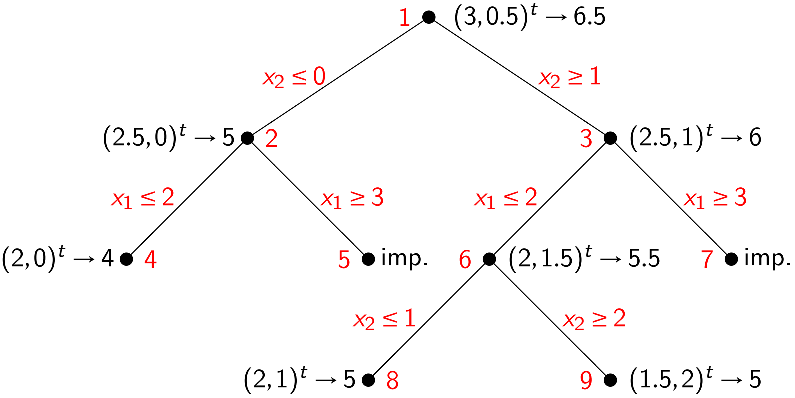
- o problema for impossível (domínio do nó é o conjunto vazio);
- a solução x_{PL} for inteira (actualizar incumbente se $z_{PL} > z_{SI}$);
- a solução x_{PL} for fraccionária e não puder haver na subárvore uma solução inteira melhor do que a solução incumbente ($z_{PL} \leq z_{SI}$);

Opção 2) fazer partição (explorar a subárvore) se:

- a solução x_{PL} for fraccionária e ainda puder haver na subárvore uma solução inteira melhor do que a solução incumbente ($z_{PL} > z_{SI}$);

Exemplo: problema de maximização

Que nós podemos abandonar? porquê?



Legenda

$x_{PL} = (x_1, x_2)^t \to z_{PL}$: sol.óptima de PL do nó $\to$ valor óptimo da f. obj.

Exemplo: avaliação do nó:

1. solução fraccionária; explorar nó
2. solução fraccionária; explorar nó
3. solução fraccionária; explorar nó
4. solução inteira com valor 4; actualizar incumbente; abandonar nó
5. problema impossível (domínio vazio)
6. solução fraccionária; explorar nó
7. problema impossível (domínio vazio)
8. solução inteira com valor 5; actualizar incumbente; abandonar nó
9. solução fraccionária; valor igual ao incumbente; abandonar nó

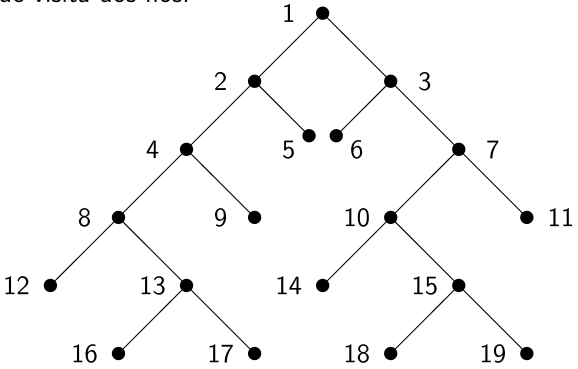
Solução óptima inteira: $x_I^* = (2,1)^t$ com valor óptimo $z_I^* = 5$.

Regra de pesquisa da árvore

- Regra de pesquisa: determina a ordem de visita dos nós, e.g.,
  - Pesquisa em largura
  - Pesquisa em profundidade
- A ordem de visita dos nós conduz a diferentes decisões de poda de subárvores, pelo que o conjunto de nós *efectivamente* visitados depende da regra de pesquisa,
- mesmo que a regra de partição seja igual.

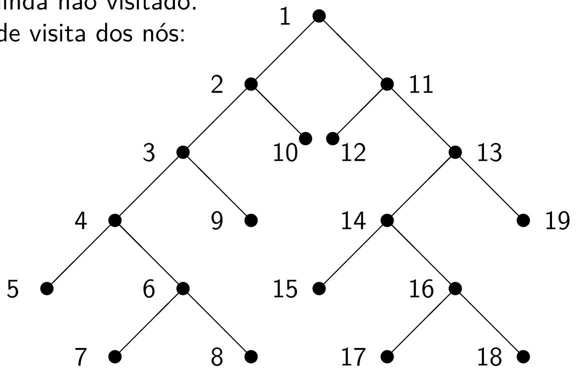
Pesquisa em largura,

- BFS *(breadth first search)* ou FIFO *(first in first out):* visitam-se em primeiro lugar todos os nós de um dado nível de profundidade antes de passar aos nós do nível seguinte.
- Depois de cada nó, é visitado o primeiro nó não visitado do mesmo nível de profundidade. Quando tal nó não existe, é seleccionado o primeiro nó anteriormente criado, mas ainda não visitado.
- Ordem de visita dos nós:



Pesquisa em profundidade,

- DFS *(depth first search)* ou LIFO *(last in first out):* desce-se para visitar em primeiro lugar os nós a um maior nível de profundidade em relação à raiz da árvore.
- Depois de cada nó, é visitado o primeiro nó filho mais à esquerda. Quando tal nó não existe, é necessário recuar *(backtrack)*, diminuindo o nível de profundidade, e encontrar o primeiro nó à direita ainda não visitado.
- Ordem de visita dos nós:



Exemplo: resolução

- Problema de Programação Inteira (PI):

$$\begin{array}{llcll} \max & 2x_1 + & 1x_2 & & \\ suj. & 2x_1 + & 2x_2 & \leq & 7 \\ & 2x_1 - & 2x_2 & \leq & 5 \\ & \multicolumn{4}{l}{x_1,\ x_2 \geq 0 \text{ e inteiros}} \end{array}$$

A relaxação linear do modelo de programação inteira é o conjunto X_1, de pontos $x = (x_1, x_2)^t$:

$$X_1 = \{x \in \mathrm{IR}^2 : 2x_1 + 2x_2 \leq 7, 2x_1 - 2x_2 \leq 5, x_1, x_2 \geq 0\}$$

Regras a usar no método de partição e avaliação

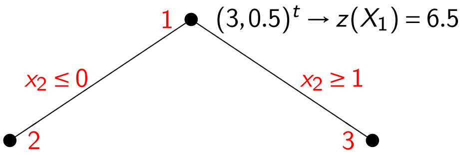
- Regra de pesquisa: largura (BFS)
- Regra de partição: seleccionar a variável x_j fraccionária com menor índice, e usar as restrições de partição: $x_j \leq \lfloor x_j \rfloor$ e $x_j \geq \lceil x_j \rceil$.

Exemplo: nó 1 (nó raiz)

ListaNós $= \{1\}$;

Seleccionar nó 1; ListaNós $= \emptyset$

- Solução óptima de X_1 é fraccionária: $x_{PL}(X_1) = (3, 0.5)^t, z(X_1) = 6.5$
- ainda não há soluções inteiras, fazer partição.
- Partição no nó 1:
  - domínio do nó filho à esquerda: $X_2 = \{x \in X_1 : x_2 \leq 0\}$
  - domínio do nó filho à direita: $X_3 = \{x \in X_1 : x_2 \geq 1\}$
- ListaNós $= \{2,3\}$ (adicionar nós 2 e 3 à lista de nós a explorar).

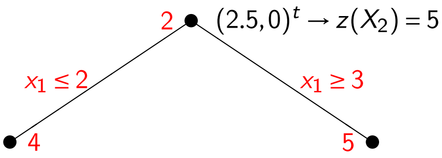


Exemplo: nó 2

ListaNós $= \{2,3\}$;

Seleccionar nó 2; ListaNós $= \{3\}$

- Solução óptima de X_2 é fraccionária: $x_{PL}(X_2) = (2.5, 0)^t$, $z(X_2) = 5$
- ainda não há soluções inteiras, fazer partição.
- Partição no nó 2:
  - domínio do nó filho à esquerda: $X_4 = \{x \in X_2 : x_1 \leq 2\}$
  - domínio do nó filho à direita: $X_5 = \{x \in X_2 : x_1 \geq 3\}$
- ListaNós $= \{3,4,5\}$ (adicionar nós 4 e 5 à lista de nós a explorar).

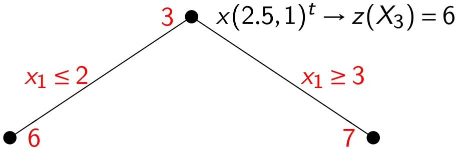


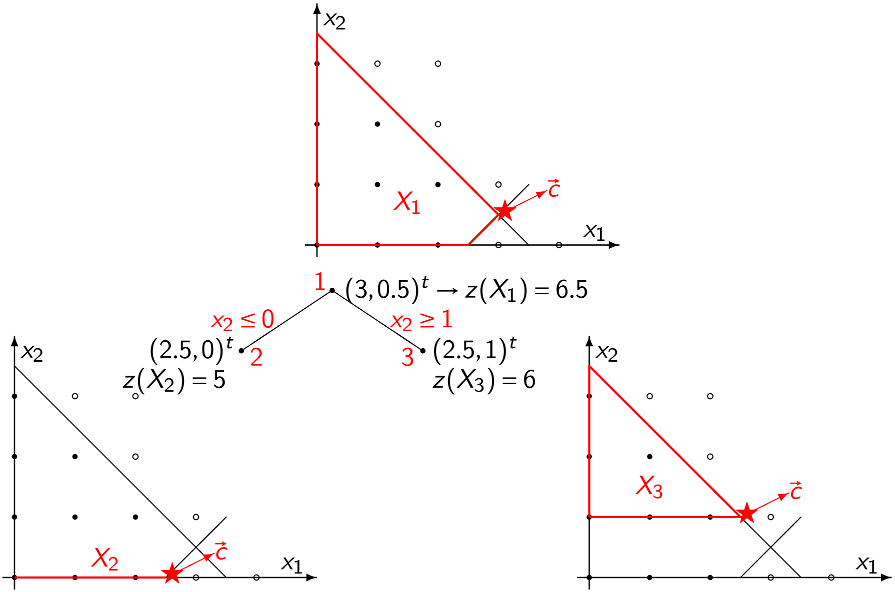
Exemplo: nó 3

ListaNós = {3,4,5};

Seleccionar nó 3; ListaNós = {4,5}

- Solução óptima de X_3 é fraccionária: $x_{PL}(X_3) = (2.5, 1)^t$, $z(X_3) = 6$
- ainda não há soluções inteiras, fazer partição.
- Partição no nó 3:
  - domínio do nó filho à esquerda: $X_6 = \{x \in X_3 : x_1 \leq 2\}$
  - domínio do nó filho à direita: $X_7 = \{x \in X_3 : x_1 \geq 3\}$
- ListaNós = {4,5,6,7} (adicionar nós 6 e 7 à lista de nós a explorar).





x_2
X_1
$\vec{c}$
x_1
1
$(3, 0.5)^t \rightarrow z(X_1) = 6.5$
$x_2 \leq 0$
$x_2 \geq 1$
$(2.5, 0)^t$
2
$z(X_2) = 5$
3
$(2.5, 1)^t$
$z(X_3) = 6$
x_2
X_2
$\vec{c}$
x_1
x_2
X_3
$\vec{c}$
x_1

Exemplo: nós 4 e 5

ListaNós $= \{4{,}5{,}6{,}7\}$

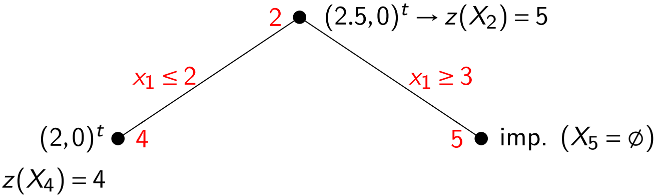
Seleccionar nó 4; ListaNós $= \{5{,}6{,}7\}$

- Solução óptima de X_4 é inteira: $x_{PL}(X_4) = (2{,}0)^t$, $z(X_4) = 4$
- actualizar incumbente; abandonar nó 4

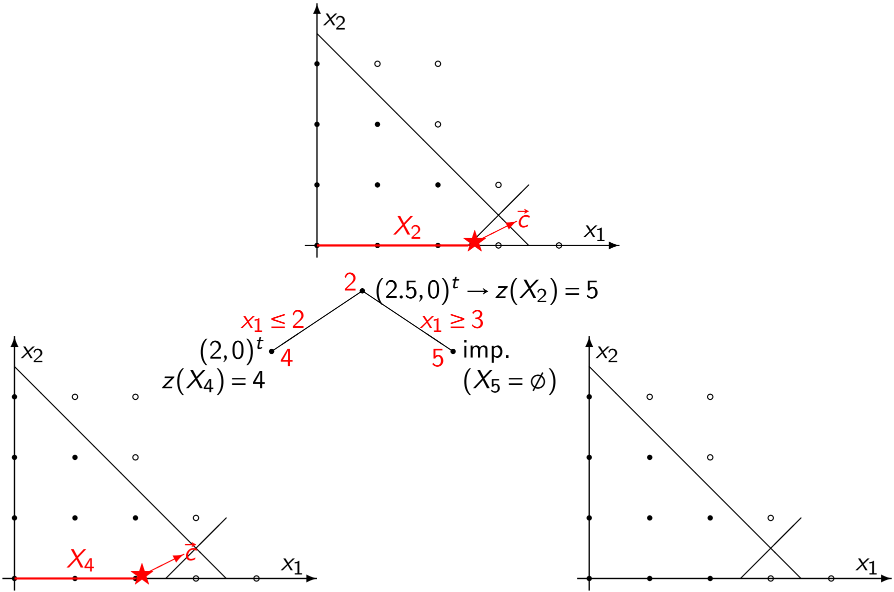
ListaNós $= \{5{,}6{,}7\}$

Seleccionar nó 5; ListaNós $= \{6{,}7\}$

- Problema impossível: $X_5 = \emptyset$ (domínio vazio); abandonar nó 5



Exemplo: síntese dos nós 2, 4 e 5

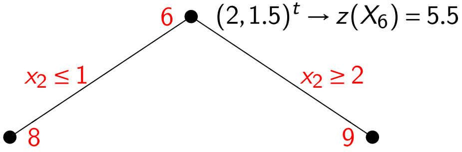


Exemplo: nós 6 e 7

ListaNós = {6,7}

Seleccionar nó 6; ListaNós = {7}

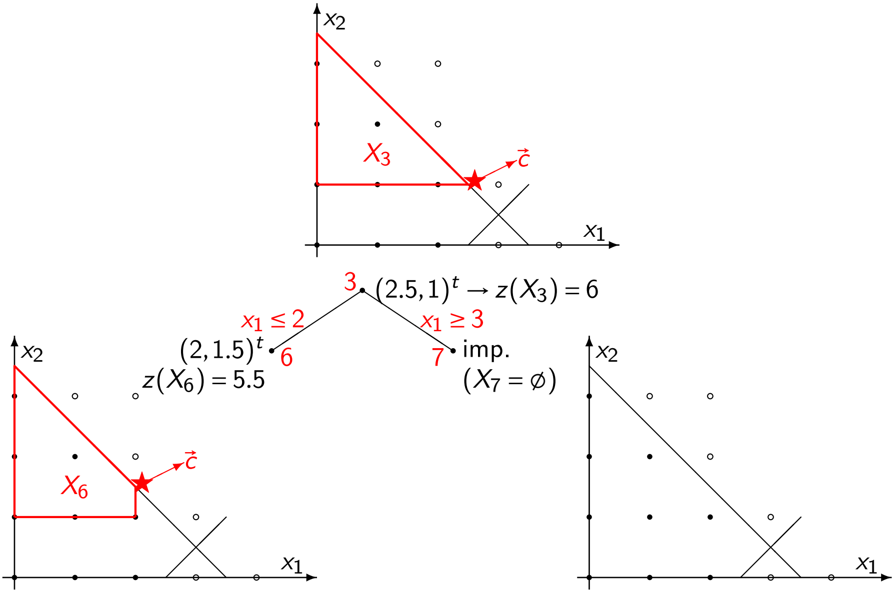
- Solução óptima de X_6 é fraccionária: $x_{PL}(X_6) = (2, 1.5)^t$, $z(X_6) = 5.5$
- X_6 pode conter solução inteira melhor do que incumbente.
- Partição no nó 6:
  - domínio do nó filho à esquerda: $X_8 = \{x \in X_6 : x_2 \leq 1\}$
  - domínio do nó filho à direita: $X_9 = \{x \in X_6 : x_2 \geq 2\}$
- ListaNós = {7,8,9} (adicionar nós 8 e 9 à lista de nós a explorar).



Seleccionar nó 7; ListaNós = {8,9}

- Problema impossível: $X_7 = \emptyset$ (domínio vazio); abandonar nó 7

Exemplo: síntese dos nós 3, 6, 7



Exemplo: nós 8 e 9

ListaNós $= \{8,9\}$

Seleccionar nó 8; ListaNós $= \{9\}$

- Solução óptima de X_8 é inteira: $x_{PL}(X_8) = (2,1)^t$, $z(X_8) = 5$
- actualizar incumbente; abandonar nó 8

ListaNós $= \{9\}$

Seleccionar nó 9; ListaNós $= \emptyset$

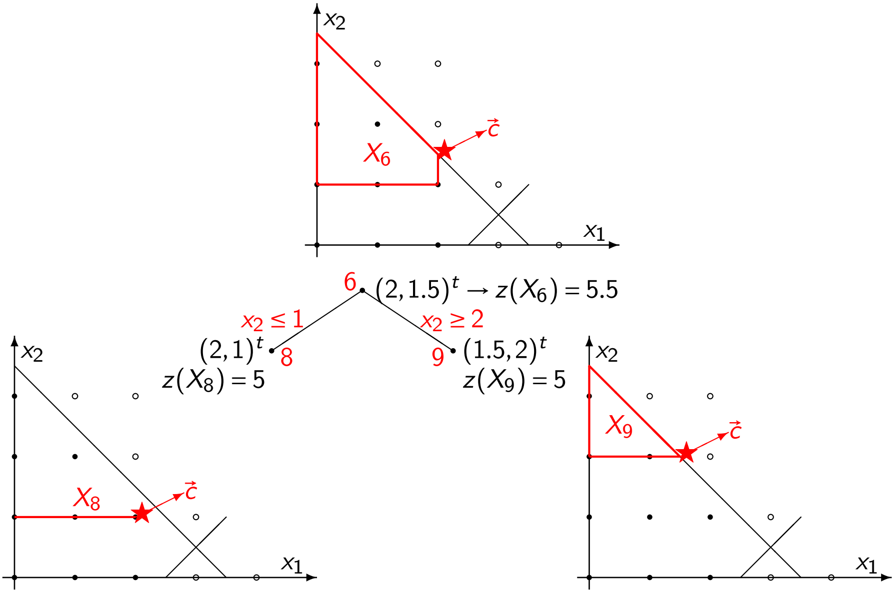
- Solução óptima de X_9 é fraccionária: $x_{PL}(X_9) = (1.5,2)^t$, $z(X_9) = 5$
- X_9 não poderá conter nenhuma solução melhor do que incumbente; abandonar nó 9.

ListaNós $= \emptyset$

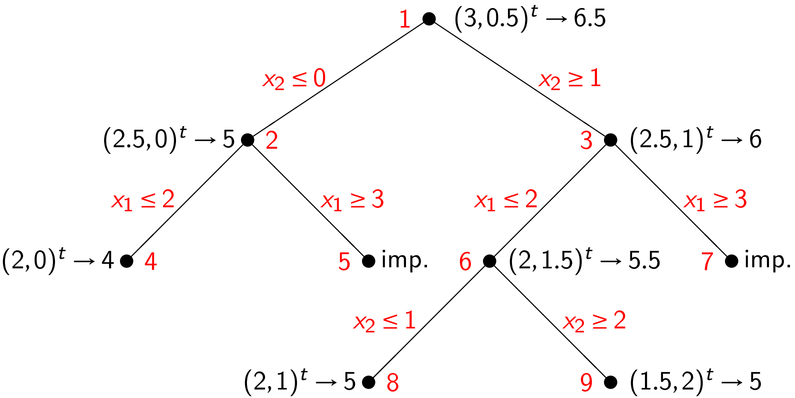
Enumeração terminada

- Solução óptima inteira: $x_I^* = (2,1)^t$ com valor óptimo $z_I^* = 5$.

Exemplo: síntese do nós 6, 8, 9



Exemplo: árvore de pesquisa (síntese)



Solução óptima inteira: $x_I^* = (2, 1)^t$ com valor óptimo $z_I^* = 5$.

Limite inferior e limite superior para o valor do óptimo

- Ao longo da pesquisa, obtém-se informação que permite conhecer o intervalo que contém o valor da solução óptima inteira (z_I^*):

$$z_I^* \in [LI, LS]$$

  sendo LI (Limite Inferior) e LS (Limite Superior).

- Isso permite avaliar quão próximo do óptimo é o valor da solução incumbente.
- Em cada nó da árvore, pode haver uma actualização de um ou dos dois limites do intervalo.

Limite superior (problema de maximização)

- O LS é apenas um valor, e não está associado a nenhuma solução inteira admissível.
- Exemplo: todas as soluções inteiras pertencem à relaxação linear do modelo (o domínio da raiz da árvore).
- O valor do óptimo da relaxação linear (z_{RL}) é um *limite superior*, calculado na raiz da árvore, para o valor do óptimo inteiro (z_I^*) :

$$z_I^* \leq z_{RL}$$

- Para problemas de minimização, é o oposto (limite inferior).

Exemplo: limite superior (raiz da árvore)

$$
\begin{array}{llcll}
\max & 2x_1 & + & 1x_2 & & \\
\textit{suj.} & 2x_1 & + & 2x_2 & \leq & 7 \\
 & 2x_1 & - & 2x_2 & \leq & 5 \\
\end{array}
$$
$$x_1,\ x_2 \geq 0 \text{ e inteiros}$$

	x_1	x_2	s_1	s_2	
x_1	1	0	1/4	1/4	3
x_2	0	1	1/4	-1/4	1/2
	0	0	3/4	1/4	6 1/2

- O valor do óptimo inteiro $z_I^* \leq z_{RL} = 6\ \frac{1}{2}$,
- ou melhor, $z_I^* \leq 6$, dado que z_I^* é um valor inteiro, porque os coeficientes da função objectivo são inteiros e as variáveis de decisão têm que ser inteiras.

Limite inferior (prob. de maximização) e heurísticas

- O valor de qualquer solução inteira admissível (*e.g.*, a solução incumbente ou uma *solução heurística*) é um *limite inferior* para z_I^*.
- Para problemas de minimização, é o oposto (limite superior).

Heurística: algoritmo que visa descobrir uma solução inteira admissível.

- Exemplo 1: uma a uma, e por ordem crescente de índices, atribuir à variável de decisão o máximo valor inteiro (dados os recursos disponíveis) que optimiza a função objectivo.
- Exemplo 2: arredondar (para cima/para baixo) os valores das variáveis fraccionárias, para construir uma solução inteira (pode não descobrir uma solução admissível!).

- Há heurísticas de baixa complexidade (*e.g.*, linear ou quadrática em função da dimensão dos dados do problema) ou muito elaboradas.
- A escolha da melhor heurística a usar depende do problema, e pode ser justificada por testes computacionais.

Exemplo: heurística

$$\begin{array}{lrcrcl} \max & 2x_1 & + & 1x_2 & & \\ suj. & 2x_1 & + & 2x_2 & \leq & 7 \\ & 2x_1 & - & 2x_2 & \leq & 5 \\ & & & \multicolumn{3}{l}{x_1,\ x_2 \geq 0 \text{ e inteiros}} \end{array}$$

- Heurística: uma a uma, por ordem crescente de índices, atribuir à variável de decisão o máximo valor inteiro (dados os recursos disponíveis) que optimiza a função objectivo.

Aplicação da heurística ao exemplo:

- passo 1: Atribuir a x_1 o valor 2. O valor 3 iria violar a segunda restrição. Quando se fixa $x_1 = 2$, restam 3 unidades do recurso 1 e 1 unidade do recurso 2.
- passo 2: Atribuir a x_2 o valor 1 (depois de fixar x_1, já só havia 3 unidades do recurso 1).
- Solução heurística: $x^{heur} = (2, 1)^t$, com valor $z_{heur} = 5$.
- Limite inferior para o valor do óptimo: $z_{heur} = 5 \leq z_I^*$.

Diferença (*gap*) de integralidade

Diferença (gap) de integralidade (em %): $(LS - LI)/LI$

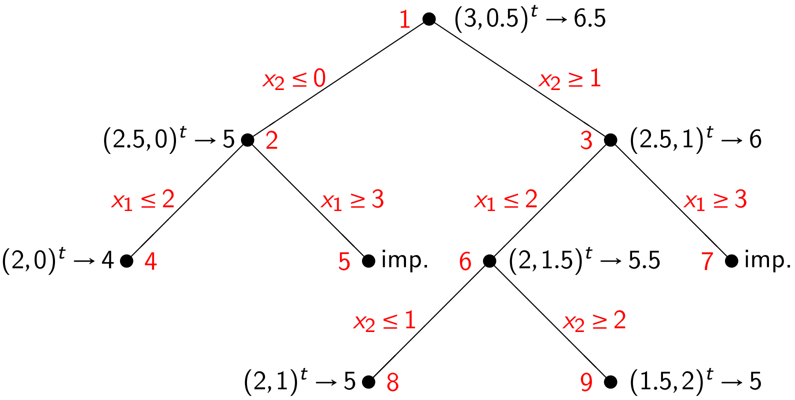
Exemplo: na raiz da árvore de pesquisa,

- LS=6 e a solução heurística (inteira e admissível) tem valor 5 = LI.
- O valor do óptimo inteiro $z_I^* \in [5,6]$.
- O *gap* = (6 - 5) / 5 = 20%.
- Há a garantia de que a solução inteira admissível conhecida está a menos de 20% do óptimo (ou, em valor absoluto, a 1 unidade do óptimo).

- Quando gap = 0, o LS (num problema de maximização) prova que a solução incumbente (com o valor LI) é a solução óptima.
- Podemos querer interromper a pesquisa quando o valor do *gap* é suficientemente pequeno.

Exemplo: Actualização de LI e LS (prob. de maximização)

Quais os LS e LI ao longo da pesquisa da árvore?



Exemplo: actualização de LI e LS (prob. maximização)

Os valores são actualizados nos seguintes nós (sem usar heurísticas):

1. LS=6, LI=$-\infty$
2.
3.
4. LS=6, LI=4
5.
6.
7. LS=5, LI=4
8. LS=5, LI=5
9.

Solução óptima inteira: $x^* = (2,1)^t$ com valor óptimo $z_I^* = 5$.

$^{(*)}$ não era necessário analisar o nó 9, porque, no nó 7 (depois de abandonar os nós 4, 5, 6 e 7), concluímos que o $LS = 5 = \lfloor 5.5 \rfloor$ (indicado pelo nó 6) e no nó 8 encontrámos uma solução inteira que tem o valor 5 (é portanto a solução óptima!), e a pesquisa pode terminar.

Notas - I

- O uso de um *modelo forte* (cuja relaxação linear descreve de uma forma mais próxima o domínio de soluções inteiras) produz um LS mais próximo do valor do óptimo (problemas de maximização).
- Heurísticas que gerem melhores soluções heurísticas ajudam também a obter uma pequena diferença de integralidade
- (quando se usa uma heurística em cada nó de árvore, ela deve ser eficiente).
- Quando as diferenças de integralidade são pequenas, o número de nós da árvore explorados é tipicamente menor.

Notas - II

- No pior caso, pode ser necessário visitar um número de nós exponencialmente grande em relação ao número de variáveis de decisão (Problema de PI é NP-completo).
- Em PI, não é conhecida nenhuma "boa" caracterização para provar que uma dada solução inteira é óptima (além do *gap*=0).
- A solução óptima inteira pode até ser encontrada cedo na pesquisa, mas provar que é óptima pode requerer um tempo muito elevado.
- É necessário pesquisar a árvore, fazendo uma enumeração *implícita* (muitos nós não são visitados, porque são podados na Avaliação).

Conclusão

- Nas últimas décadas, desenvolvimentos teóricos (modelos mais fortes) e o aumento da capacidade de computação têm permitido resolver instâncias de dimensão cada vez maior.
- A programação inteira é hoje em dia usada rotineiramente para encontrar soluções para problemas reais complexos em muitas indústrias e serviços.
- Geralmente, é possível obter, em tempos razoáveis, soluções inteiras óptimas, ou então soluções com diferenças *(gaps)* de integralidade inferiores a 5%, muitas vezes de 1% ou 2%.
- Pequenos valores de *gap* asseguram a qualidade das soluções!

Resultados de aprendizagem

- Conhecer a estratégia do método de partição e avaliação (para problemas gerais de maximização e minimização):
  - (partição) conhecer a regra de partição dicotómica
  - (avaliação) conhecer os casos em que a subárvore é podada
  - conhecer o conceito de solução incumbente
- Aplicar o método usando as regras de pesquisa em largura e em profundidade.
- Aplicar heurísticas simples para obter soluções inteiras admissíveis
- Conhecer os conceitos de limite superior e limite inferior para o valor do óptimo:
  - de um nó
  - do problema, e determinar a diferença de integralidade

Apêndice

- Inserção da restrição de partição no quadro simplex

Inserção da restrição de partição no quadro simplex

	x_1	x_2	s_1	s_2	
x_1	1	0	1/4	1/4	3
x_2	0	1	1/4	-1/4	1/2
	0	0	3/4	1/4	6 1/2

- Restrição de partição a inserir: $x_2 \geq 1$.
- Fazer eliminação de Gauss, para a restrição ser expressa em termos das variáveis não-básicas:

$$\begin{aligned}
x_2 &\geq 1 \\
\left(\frac{1}{2} - \frac{1}{4}s_1 + \frac{1}{4}s_2\right) &\geq 1 \\
-\frac{1}{4}s_1 + \frac{1}{4}s_2 &\geq \frac{1}{2} \\
\frac{1}{4}s_1 - \frac{1}{4}s_2 &\leq -\frac{1}{2}
\end{aligned}$$

Quadro após inserção de $x_2 \geq 1$ e re-optimização

Restrição a inserir:

$$\frac{1}{4}s_1 - \frac{1}{4}s_2 \leq -\frac{1}{2}$$

	x_1	x_2	s_1	s_2	s_3	
x_1	1	0	1/4	1/4	0	3
x_2	0	1	1/4	-1/4	0	1/2
s_3	0	0	1/4	-1/4	1	-1/2
	0	0	3/4	1/4	0	6 1/2

	x_1	x_2	s_1	s_2	s_3	
x_1	1	0	1/2	0	1	2.5
x_2	0	1	0	0	-1	1
s_2	0	0	-1	1	-4	2
	0	0	1	0	1	6

Nota:

Esta é a solução óptima do nó 3.

Fim