Programação Inteira

Transparências de apoio à leccionação de aulas teóricas

Slide 1

Versão 1

José Fernando Oliveira, Maria Antónia Carravilla – FEUP

Técnicas exactas para resolução de problemas de optimização

Slide 2

Técnicas exactas

Slide 3

- Enumeração explícita — por definição de problema de Optimização Combinatória, *gerando* e *avaliando* todas as *soluções admissíveis* é possível obter a *solução óptima*.
- Enumeração implícita — não se gerando todas as soluções admissíveis, elas são no entanto consideradas e *implicitamente avaliadas*.

  Exemplos: Método de pesquisa em árvore com enumeração e limitação ("branch and bound"); limites superiores e inferiores ao valor da solução óptima.
- Formulação dos problemas em modelos de programação inteira (variáveis de decisão assumem valores no conjunto dos números inteiros), ou mesmo binária (variáveis apenas com dois valores possíveis: 0 ou 1), e consequente resolução com algoritmos apropriados.

  Nota: Estas formulações podem também ser usadas para obter limites para o valor da solução óptima através de **relaxações**.

Técnicas exactas e relaxações

Relaxação — Não consideração de uma ou mais restrições do problema original PO, transformando-o num problema mais simples de resolver PR, sabendo-se que os valores óptimos das funções objectivo obedecem à seguinte relação (assumindo um problema de minimização):

$$f^{\star}_{PR} \leq f^{\star}_{PO}$$

Slide 4

(tradução: ao tirar restrições a solução só pode melhorar, ou ficar na mesma).

Relaxação linear – transformação de um problema em números inteiros num problema com variáveis reais (deixa-se cair a restrição "e inteiros" ou "$\in \mathcal{N}_0$" $\longrightarrow$ utilização do método simplex para a sua resolução em vez dos muito mais complexos (e extraordinariamente mais demorados) métodos de pesquisa em árvore).

Método de "branch and bound"

Baseia-se na ideia de uma enumeração inteligente das soluções candidatas a solução óptima inteira de um problema, efectuando *sucessivas partições* do espaço das soluções e *cortando a árvore de pesquisa* através da consideração de limites calculados ao longo da enumeração.

Slide 5

Representação gráfica

Slide 6

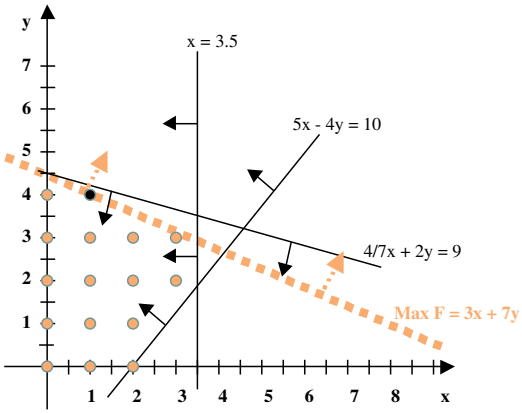
Considere-se o seguinte problema de Programação Inteira:
Maximizar:

$$F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcl} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \text{ e } \textit{inteiras} \end{array}$$

e a sua representação gráfica:



Solução óptima inteira: $x = 1$ e $y = 4$.

Resolução da relaxação linear

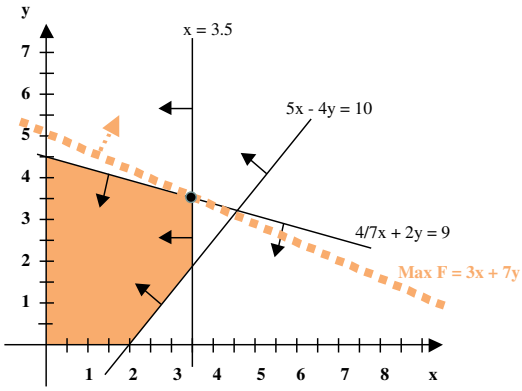
Slide 7

Problema $\mathcal{PL}0$:

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcl} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \end{array}$$



Solução óptima não inteira:
$x = 3.5$ e $y = 3.5$; $F = 35$

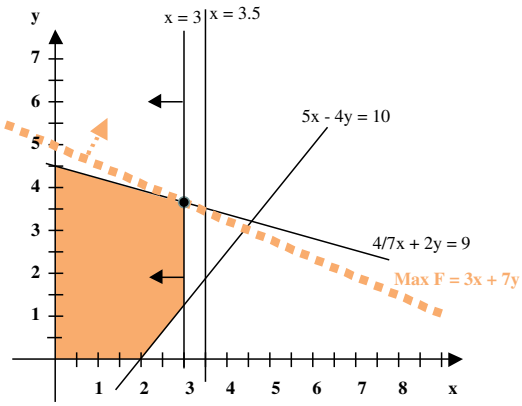
Ramificação em x: $x \leq 3$

Slide 8

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \leq & 3 \end{array}$$



Solução (não inteira):
$x = 3$ e $y = 3.6$; $F = 34.5$

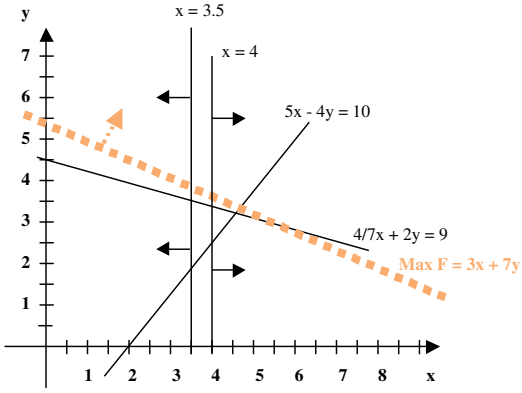
Ramificação em x: $x \geq 4$

Slide 9

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \geq & 4 \end{array}$$



Sem soluções admissíveis.

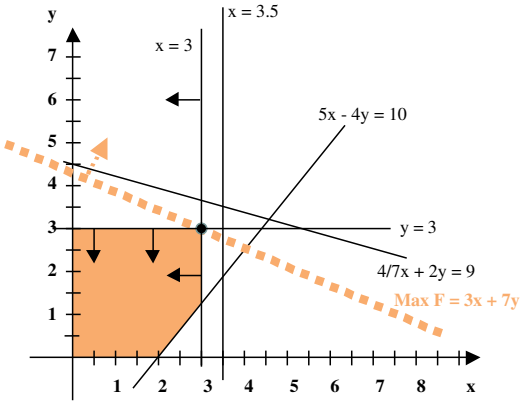
Ramificação em y: $y \leq 3$

Slide 10

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \leq & 3 \\ & & y & \leq & 3 \end{array}$$



Solução (inteira):
$x = 3$ e $y = 3$; $F = 30$
Obtenção de um limite inferior $\Rightarrow$ Soluções não inteiras com valor de F inferior ou igual a 30 não precisam de ser exploradas!

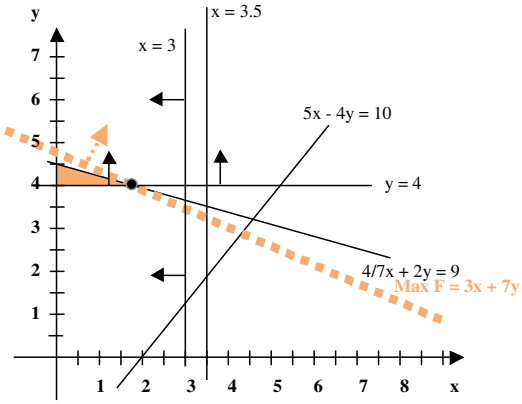
Ramificação em y: $y \geq 4$

Slide 11

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \leq & 3 \\ & & y & \geq & 4 \end{array}$$



Solução (não inteira):
$x = 1.7$ e $y = 4$; $F = 33.2$

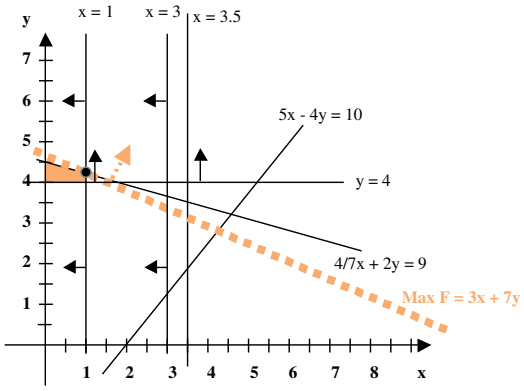
Ramificação em x: $x \leq 1$

Slide 12

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \leq & 3 \\ & & y & \geq & 4 \\ x & & & \leq & 1 \end{array}$$



Solução (não inteira):
$x = 1$ e $y = 4.2$; $F = 32.5$

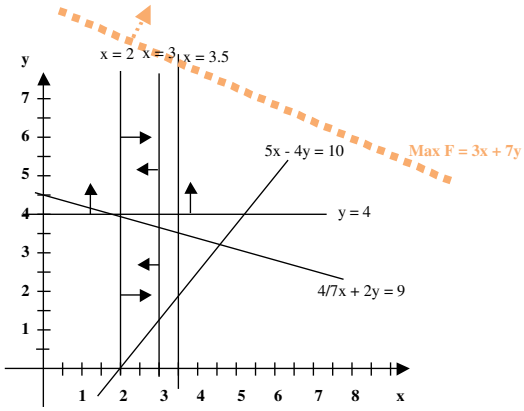
Ramificação em x: $x \geq 2$

Slide 13

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \leq & 3 \\ & & y & \geq & 4 \\ x & & & \geq & 2 \end{array}$$



Sem soluções admissíveis.

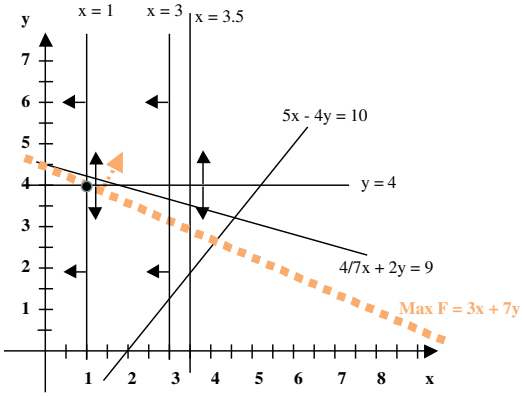
Ramificação em y: $y \leq 4$

Slide 14

$$\max F = 3x + 7y$$

suj. a:

$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \leq & 3 \\ & & y & \geq & 4 \\ x & & & \leq & 1 \\ & & y & \leq & 4 \end{array}$$



Solução (inteira):
$x = 1$ e $y = 4$; $F = 31$
Melhor solução inteira até ao momento!

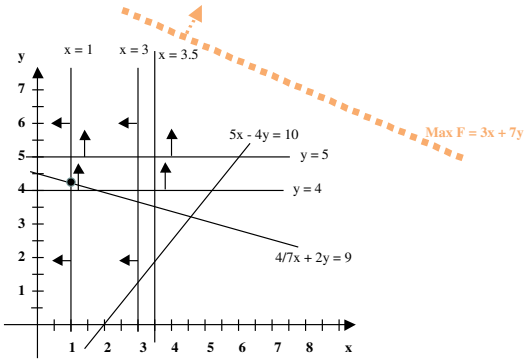
Ramificação em y: $y \geq 5$

Slide 15

$$\max F = 3x + 7y$$

suj. a:

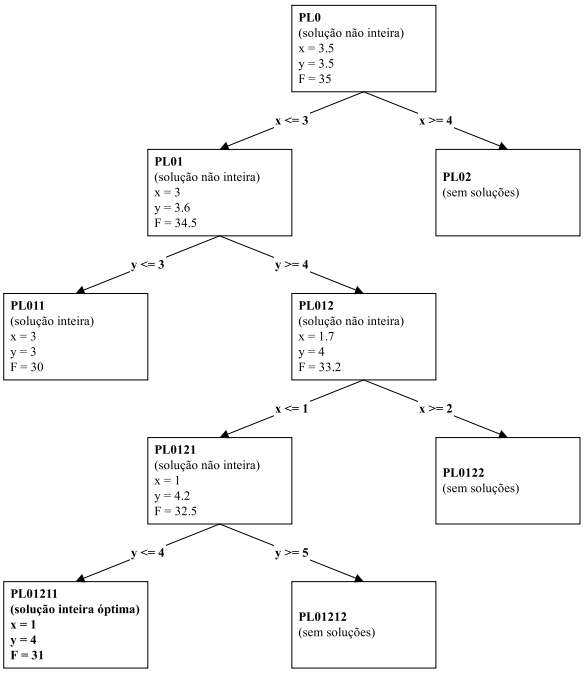
$$\begin{array}{rcrcr} x & & & \leq & 3.5 \\ 5x & - & 4y & \leq & 10 \\ \frac{4}{7}x & + & 2y & \leq & 9 \\ x & , & y & \geq & 0 \\ x & & & \leq & 3 \\ & & y & \geq & 4 \\ x & & & \leq 1 & \\ & & y & \geq & 5 \end{array}$$



Sem soluções admissíveis.

Árvore de pequisa do "Branch-and-Bound"

Slide 16



Limites

Limites (inferiores e superiores):

Slide 17

- tornam o algoritmo de "branch & bound" mais eficiente ao permitir descartar nós da árvore de pesquisa ainda não completamente explorados, pela certeza de que nunca originarão soluções melhores do que as que já temos.
- permitem "medir a distância" (em termos de valor da função objectivo) a que estamos da solução óptima.

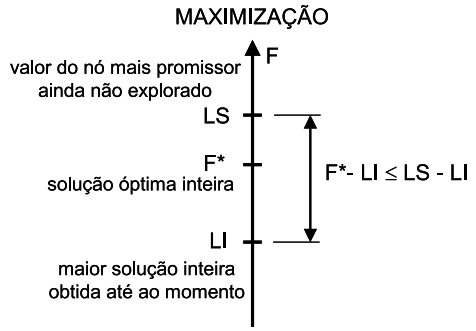
Limites

Num problema de **maximização**:

- um limite **inferior** LI é dado por uma solução inteira que se tenha já obtido – a solução óptima $F^\star$ nunca poderá ser pior (inferior) que a solução inteira que já temos;

Slide 18

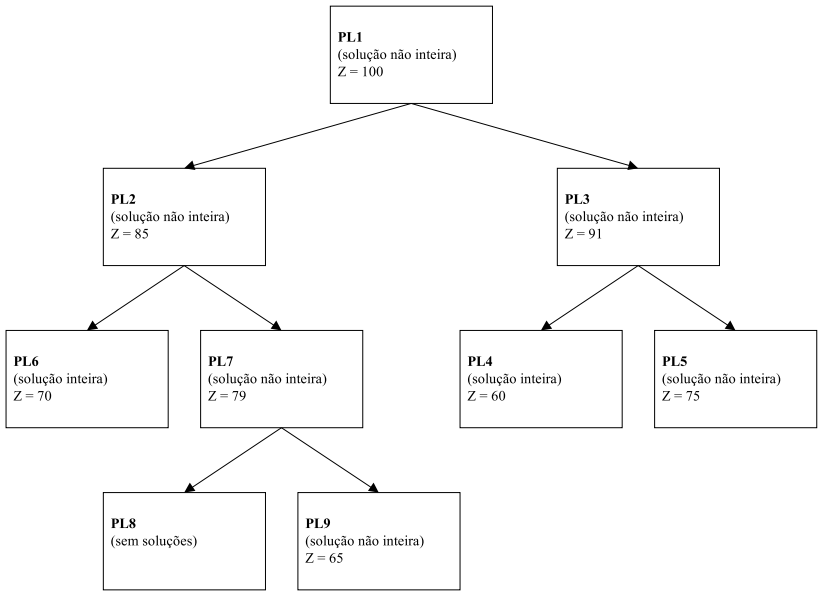
- um limite **superior** LS será dado pelo maior valor da função objectivo de entre todos os nós ainda não completamente explorados (a maior esperança que ainda temos de encontrar uma solução inteira melhor do que aquela que já temos).



Limites – exemplo

Considere um problema de maximização exclusivamente com variáveis inteiras.
Resolvendo o problema através de "Branch-and-Bound", obtém-se, num dado estágio, a seguinte árvore:

Slide 19



Limites – exemplo

Slide 20

- Qual é, nesta altura, o melhor **limite superior** sobre a solução inteira óptima?

  O melhor **limite superior** sobre a solução inteira óptima no momento de resolução retratado na árvore é dado pela solução do problema $PL5$ e é igual a 75. Qualquer solução inteira que surja a partir da exploração desse nó terá um valor da função objectivo ≤ 75

- Qual é, nesta altura, o melhor **limite inferior** sobre a solução inteira óptima?

  Os limites inferiores são dados por valores de soluções admissíveis (variáveis já inteiras) que ainda se desconhece se são ou não óptimas. Neste caso temos já 2 soluções inteiras, para PL6 e para PL4. A que tem o maior valor da função objectivo fornece o melhor **limite inferior**, 70 neste caso.

Limites – exemplo

Slide 21

- Indique que **nós já** foram **explorados** e explique porquê.

  Já foram explorados os nós $PL1$, $PL2$, $PL3$ e $PL7$ porque já têm ramos.

  Os nós $PL4$ e $PL6$ já foram explorados porque deram origem a soluções inteiras.

  O nó $PL8$ já foi explorado porque corresponde a um problema sem solução admissível.

  O nó $PL9$ já foi explorado porque pode ser cortado. Corresponde a um problema com solução óptima não inteira e com um valor para a função objectivo inferior ao valor da solução inteira do problema $PL6$.

- Indique os nós que **ainda não** foram **explorados** e explique porquê.

  O nó $PL5$ ainda não foi explorado, dado que corresponde a um problema com solução óptima não inteira, mas com um valor para a função objectivo superior ao valor da melhor solução inteira obtida até ao momento (problema $PL6$).

Limites – exemplo

- Já foi atingida a solução óptima do problema inteiro? Porquê?

  Não se sabe ainda se já foi obtida a solução óptima do problema inteiro, porque ainda há nós por explorar ($PL5$). Só quando os melhores limites inferiores e superiores coincidirem é que se pode afirmar que a melhor solução inteira obtida é a óptima.

Slide 22

- Qual o valor máximo do erro absoluto sobre a solução óptima inteira, se o algoritmo for terminado neste ponto?

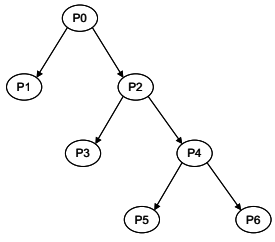
  O valor máximo do erro absoluto sobre a solução óptima inteira, se o algoritmo for terminado neste ponto será 5, isto é, a diferença entre os melhores limites superior e inferior.

Questões em aberto

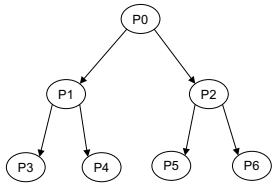
- **Estratégias de ramificação** – Dado um conjunto de nós ainda não explorados, como escolher o nó seguinte a explorar?

  – Pesquisa em profundidade: Selecção do nó que está mais fundo na árvore (último nó a ser gerado).

Slide 23



  – Pesquisa em largura: Selecção do nó que está mais acima na árvore (o nó mais antigo ainda não explorado).



  – O nó mais promissor: Selecção do nó que tem melhor valor de função objectivo (aquele que potencialmente nos pode levar à melhor solução inteira).

Questões em aberto

Slide 24

- **Selecção da variável a ramificar** – Seleccionado o nó a explorar, que variável escolher para ramificação, de entre todas as variáveis que não tomam valores inteiros?

  Algumas estratégias foram apresentadas na literatura, mas o seu desempenho revelou-se extremamente dependente do problema concreto a que são aplicadas.

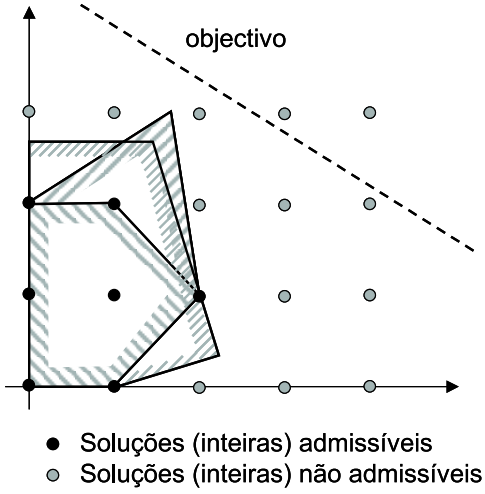
  ⇓

  Estratégias dependentes da aplicação e do significado físico das variáveis.

Formulações em programação inteira

Um mesmo problema de programação inteira pode ter diferentes formulações, isto é, diferentes conjuntos de restrições que definem o mesmo conjunto de soluções inteiras.

Slide 25

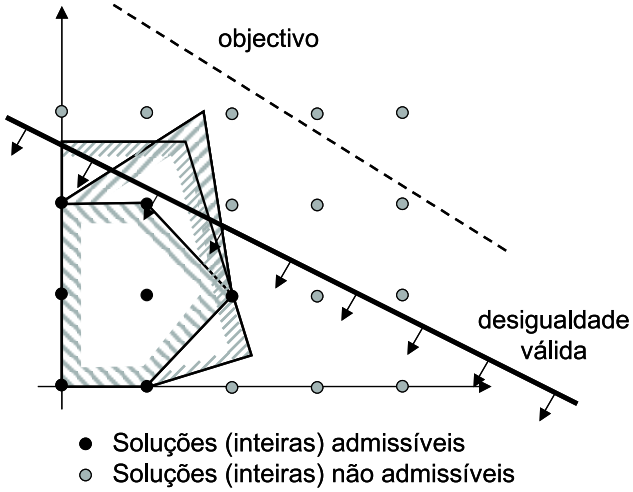


O ideal seria ter o invólucro convexo das soluções inteiras admissíveis.

Desigualdades válidas

Slide 26

Para melhorar a qualidade das formulações podem-se introduzir **desigualdades válidas**: restrições que não fazem parte do problema original mas que, não cortando soluções inteiras admissíveis, cortam à região admissível, melhorando assim o desempenho da pesquisa da solução óptima inteira.



Exemplo de uma desigualdade válida

Slide 27

Considere a seguinte instância de um problema mochila 0-1:

$$\begin{array}{rll} \max & 12x_1 + 14x_2 + 15x_3 + 24x_4 + 24x_5 + 17x_6 & \\ \text{suj. a} & 15x_1 + 14x_2 + 14x_3 + 18x_4 + 17x_5 + 12x_6 \leq 60 & \\ & x_i \in \{0,1\} & \end{array}$$

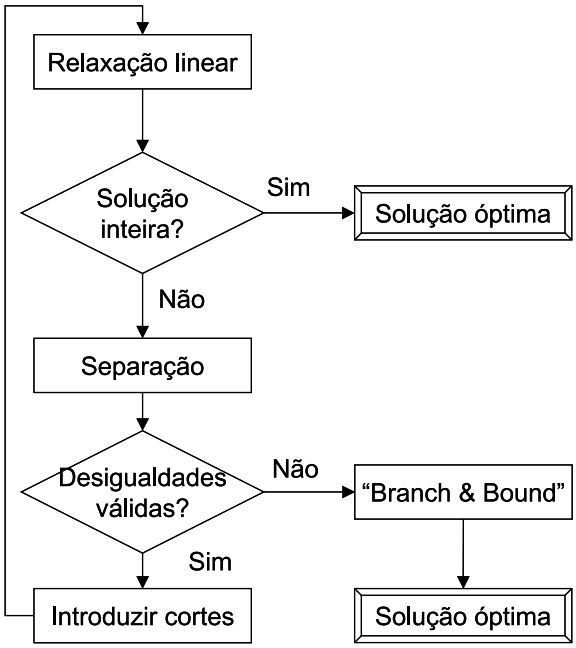
Observando que, por exemplo, os items 1, 2, 4 e 5 nunca poderão fazer parte simultaneamente de nenhuma solução inteira admissível, poder-se-ia introduzir a seguinte desigualdade válida:

$$x_1 + x_2 + x_4 + x_5 \leq 3$$

Algoritmo de planos de corte

Como proceder para aplicar sistematicamente desigualdades válidas na resolução de um problema?

Slide 28



Algoritmo de planos de corte

- **Relaxação linear** – resolução do problema sem as restrições de integralidade.
- **Separação** – fase em que se geram as desigualdades válidas. Utilizando uma (ou mais) propriedade observada (e descoberta) anteriormente aplica-se à instância concreta e/ou à solução actual. Caso a actual solução fraccionária viole esta nova desigualdade, então ela é válida (no sentido de que vai cortar à região admissível do problema relaxado).

Slide 29

- **Introduzir cortes** – introdução das desigualdades válidas, encontradas na fase de separação, no modelo.

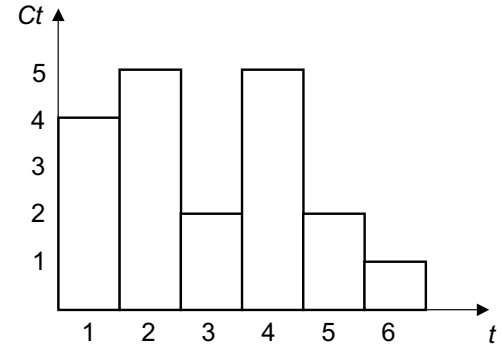
Exemplo – um problema (simples) de planeamento da produção

Dados – 6 períodos e 8 encomendas.
Objectivo – produzir o mais próximo possível da data de entrega.

Slide 30

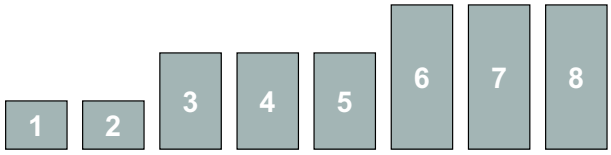
t	1	2	3	4	5	6
C_t	4	5	2	5	2	1

Capacidade disponível C_t em cada período t



e	1	2	3	4	5	6	7	8
q_e	1	1	2	2	2	3	3	3
d_e	2	1	3	1	2	5	1	3

Encomendas e, com quantidades q_e e datas de entrega d_e



Exemplo – continuação

Variáveis de decisão: $x_{et} \in \{0, 1\}$ que valem 1 se a encomenda e é produzida no período t.

Restrições:

Slide 31

- Cada encomenda tem que ser produzida uma e uma só vez: $\sum_t x_{et} = 1$
- As capacidades dos períodos têm que ser respeitadas: $\forall_t \sum_e q_e \times x_{et} \leq C_t$

Observação: há encomendas que, dadas as respectivas quantidades e as capacidades dos períodos, nunca poderão ser produzidas em simultâneo.

Exemplo – continuação

Regras para a geração de desigualdades válidas:

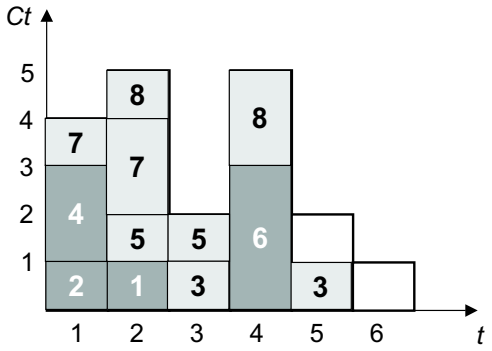
1. No período 1 (capacidade 4) não se podem produzir simultaneamente duas encomendas com quantidades 2 e 3, ou 3 e 3.
2. Nos períodos 2 e 4 (capacidade 5) não se podem produzir simultaneamente duas encomendas com quantidades 3.

Slide 32

3. Nos períodos 3 e 5 (capacidade 2) não se podem produzir simultaneamente duas encomendas com quantidades 2, duas encomendas com quantidades 1 e 2, nem qualquer encomenda com quantidade 3.
4. No período 6 (capacidade 1) apenas se podem produzir encomendas com quantidade 1.

Exemplo – conclusão

Solução da relaxação linear do exemplo:



Slide 33

Esta solução viola uma desigualdade do tipo 1 e uma desigualdade do tipo 2.

São então desigualdades válidas:

$$x_{41} + x_{71} \leq 1$$
$$x_{64} + x_{84} \leq 1$$

Bibliografia

Slide 34

- Alves, José Carlos (1989). *Provas de Aptidão Científica e Capacidade Pedagógica.* FEUP.
- Carravilla, Maria Antónia (1996). *Modelos e Algoritmos para o Planeamento Hierárquico da Produção – Aplicações a um Caso de Estudo*, Tese de Doutoramento, FEUP.
- Goldbarg, Marco Cesar e Luna, Henrique Pacca (2000). *Otimização Combinatória e Programação Linear*, Editora CAMPUS.
- Nemhauser, George L. e Wolsey, Laurence A. (1988). *Integer and Combinatorial Optimization* John Wiley & Sons, Inc..