Modelos de Programação Inteira (PI)

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

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

Universidade do Minho
November 21, 2013

Conteúdo

- Exemplos de motivação:
  - Saco de mochila
  - Selecção de subconjuntos de um conjunto
    - Partição de um conjunto
    - Cobertura de um conjunto
    - Empacotamento
- Expressões lógicas e Restrições com variáveis binárias
- Restrições activas e redundantes
- Aplicações:
  - Custo fixo
  - Localização de serviços
  - Planeamento de rotas de veículos
  - Selecção de projectos de investimento
  - Planeamento de operações numa máquina
- Considerações sobre arredondamento de variáveis
- Considerações sobre qualidade de modelos

Problema do saco de mochila (*knapsack problem*)

Objectivo: seleccionar o conjunto de itens a colocar num saco de mochila para maximizar o lucro (utilidade).

- Cada item tem um lucro p_j e um peso w_j.
- Peso máximo do saco de mochila é W.
- Variáveis de decisão x_j : número de itens do tipo j a incluir no saco de mochila.

$$
\begin{array}{ll}
\max & \displaystyle\sum_{j=1}^{n} p_j x_j \\
\textit{suj.} & \displaystyle\sum_{j=1}^{n} w_j x_j \leq W \\
 & x_j \geq 0 \text{ e inteiro}, \; j = 1, 2, \ldots, n
\end{array}
$$

Exemplo

- Um investidor tem 90 U.M. para aplicar em acções, e pretende maximizar o valor esperado do retorno.
- As acções de cada companhia são vendidas em lotes indivisíveis.

- Valor do investimento e o valor esperado de retorno de cada lote:

companhia	1	2	3	4	5
retorno (U.M.)	25	40	7	9	10
investimento (U.M.)	12	20	4	6	8

solução óptima fraccionária (se se pudessem comprar fracções de lotes):

- Solução óptima seria investir em 7.5 (=90/12) lotes de acções da companhia 1 (porquê?)

Exemplo (cont.)

- O problema de investimento pode ser formulado como um problema de saco de mochila:

$$\begin{array}{ll} \max & 25x_1 + 40x_2 + 7x_3 + 9x_4 + 10x_5 \\ suj. & 12x_1 + 20x_2 + 4x_3 + 6x_4 + \ \ 8x_5 \leq 90 \\ & \qquad\qquad x_j \geq 0 \text{ e inteiro}, \ j = 1, 2, \ldots, n \end{array}$$

solução óptima inteira:

- O portfolio óptimo é 7 lotes da companhia 1 e 1 lote da companhia 4, com um valor esperado de retorno de 184.

problema de saco de mochila multidimensional:

- há restrições relativas a vários recursos, *e.g.*, peso, volume, etc.: $\max\{px : wx \leq W, x \geq 0$ e inteiro$\}$, sendo todos os coeficientes positivos, *i.e.*, $w \in \mathbb{R}_+^{m\times n}$ e $W \in \mathbb{R}_+^{m\times 1}$.

Exemplo: 2-partição

Objectivo: dividir a lista de itens em dois conjuntos com pesos exactamente iguais.

- Pesos dos itens = { 100, 86, 75, 68, 56, 37, 24, 20, 18, 16 }
- Soma dos pesos da lista = 500.

- Desafio: formular como um problema de saco de mochila que forneça uma solução ou que mostre que não existe nenhuma solução.

Selecção de subconjuntos de um conjunto

- S : um conjunto finito com m elementos,
- $S_1, \ldots, S_j, \ldots, S_n$: subconjuntos de S.
- c_j : valor associado ao subconjunto S_j.

Objectivo: seleccionar uma colecção de subconjuntos que sejam:

- uma partição do conjunto S (*set partitioning problem*), ou
- uma cobertura do conjunto S (*set covering problem*), ou
- um empacotamento no conjunto S (*set packing problem*),

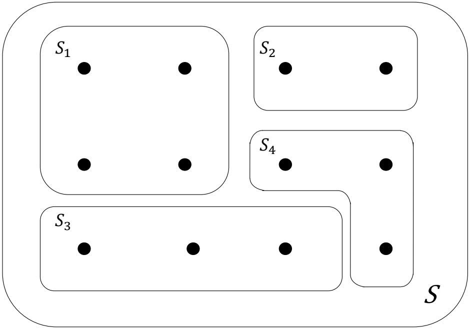
optimizando uma função objectivo relacionada com o valor do subconjuntos.

Partição de um conjunto

- Uma partição do conjunto S é uma colecção de k dos n subconjuntos, $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$, com índices $i_1, i_2, \ldots, i_k$, tal que:

$$\begin{array}{lll} \cup_{j=1}^{k} S_{i_j} = S & : & \text{(união dos subconjuntos é o conjunto } S\text{)} \\ S_{i_i} \cap S_{i_j} = \emptyset, \ \forall i,j & : & \text{(os subconjuntos são disjuntos)} \end{array}$$

- exemplo:



Selecção da partição de maior peso

- Variável de decisão binária x_j : $x_j = 1 \Leftrightarrow$ seleccionar S_j.
- Coluna $A_j = [a_{ij}]$ da variável x_j tem elementos $a_{ij}, i = 1, \ldots, m$:

$$a_{ij} = \begin{cases} 1 & \text{, se } i \in S_j \\ 0 & \text{, caso contrário} \end{cases}$$

- Selecção da partição de maior peso:

$$\begin{array}{ll} \max & \sum_{j=1}^{n} c_j x_j \\ \text{suj.} & \sum_{j=1}^{n} a_{ij} x_j = 1, \;\; i = 1, 2, \ldots, m \\ & x_j = 0 \text{ ou } 1, \;\; j = 1, 2, \ldots, n \end{array}$$

Exemplo

- Uma companhia pretende vender 3 lotes de terreno, A, B e C.
- Os interessados podiam licitar um lote ou um conjunto de lotes.
- A companhia recebeu as seguintes propostas:

Proposta	1	2	3	4	5	6	7	8	9
lote A	1	1	1	1		1		1	
lote B	1	1			1	1	1		
lote C	1		1		1				1
	12	9	7	2	7	8	4	3	4

- A proposta 2, por exemplo, significa uma oferta de 9 U.M. pelo conjunto de lotes A e B.

Exemplo

- Objectivo: determinar a partição de maior peso.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
lote A	1	1	1	1		1		1		= 1
lote B	1	1			1	1	1			= 1
lote C	1		1		1				1	= 1
max	12	9	7	2	7	8	4	3	4	

- Solução óptima: aceitar as propostas 2 e 9, com um valor de 13 U.M..

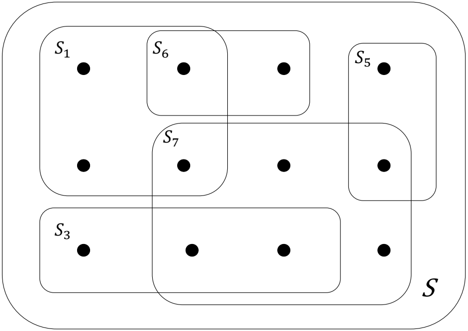
- A versão de minimização é a *partição de menor peso*.

Cobertura de um conjunto

- Uma cobertura do conjunto S é uma colecção de subconjuntos que podem não ser disjuntos, *i.e.*, tal que:

$$\cup_{j=1}^{k} S_{i_j} \quad = \quad S$$

- exemplo:



- Selecção da cobertura de menor custo:

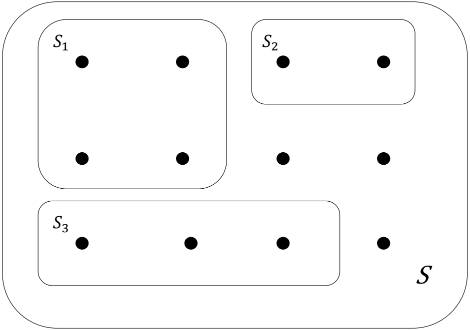
$$
\begin{aligned}
\min \quad & \sum_{j=1}^{n} c_j x_j \\
suj. \quad & \sum_{j=1}^{n} a_{ij} x_j \geq 1, \;\; i = 1, 2, \ldots, m \\
& x_j = 0 \text{ ou } 1, \;\; j = 1, 2, \ldots, n
\end{aligned}
$$

Empacotamento

- Um empacotamento é uma colecção de subconjuntos disjuntos cuja reunião seja um subconjunto de S, isto é, tal que:

$$\begin{aligned} \cup_{j=1}^{k} S_{i_j} &\subseteq S \\ S_{i_j} \cap S_{i_k} &= \emptyset, \forall j, k \end{aligned}$$

- exemplo:

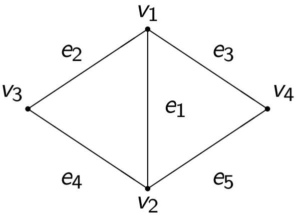


Empacotamento de um conjunto

- Selecção do empacotamento de maior peso:

$$\begin{array}{ll} \max & \sum_{j=1}^{n} c_j x_j \\ suj. & \sum_{j=1}^{n} a_{ij} x_j \leq 1, \; i = 1, 2, \ldots, m \\ & x_j = 0 \text{ ou } 1, \; j = 1, 2, \ldots, n \end{array}$$

- Exemplo: *emparelhamento de peso máximo*



$$A = \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \left(\begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array} \right) \end{array} \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}$$

- Se houver apenas dois 1's por coluna, o empacotamento pode ser representado sobre um grafo: cada linha do modelo corresponde a um vértice e cada coluna (variável de decisão) a um arco.

Expressões lógicas e restrições com variáveis binárias

Expressões lógicas

- Conjunto de literais $\{x_1, x_2, \ldots, x_n\}$, cada um podendo tomar
- o valor lógico verdadeiro (V) ou falso (F).
- $\overline{x}_i$ é o complemento (negação) do literal x_i.
- O símbolo $\wedge$ designa a operação lógica *e*.
- O símbolo $\vee$ designa a operação lógica *ou*.

Restrições com variáveis binárias

- Os valores 1 e 0 das variáveis binárias correspondem aos valores lógicos V e F , respectivamente.
- A expressão lógica é verdadeira se e só se a correspondente restrição binária for obedecida.
- exemplos:

Expressão lógica	Restrição binária
$a \vee b \vee c$	$a+b+c \geq 1$
$\overline{a} \vee b$	$(1-a)+b \geq 1$

Exemplo: implicação lógica

- Implicação lógica: se a actividade a é seleccionada, então a actividade b deve ser seleccionada:

Expressão lógica	Restrição binária
$\overline{a} \vee b$	$(1-a)+b \geq 1$
$\overline{a} \vee b$	$a \leq b$
$a \Rightarrow b$	$a \leq b$

- Tabela lógica da implicação e verificação da restrição binária:

a	b	$a \Rightarrow b$	a	b	$a \leq b$
F	F	V	0	0	OK
F	V	V	0	1	OK
V	F	F	1	0	–
V	V	V	1	1	OK

Outros exemplos

Expressão lógica	Restrição binária
$a \Rightarrow b$	$a \leq b$
$\overline{b} \Rightarrow \overline{a}$	$(1-b) \leq (1-a)$
$\overline{b} \Rightarrow \overline{a}$	$a \leq b$
$a \Rightarrow \overline{b}$	$a+b \leq 1$
$b \Rightarrow \overline{a}$	$a+b \leq 1$
$a \dot{\vee} b$ (ou exclusivo)	$a+b=1$
seleccionar *exactamente* uma das opções	$a+b+\ldots+z=1$
seleccionar, *no máximo.* uma das opções	$a+b+\ldots+z \leq 1$
$a.b \Rightarrow c$	$a+b-1 \leq c$

- Generalização:

- *Forma normal conjuntiva*: qualquer expressão lógica pode ser expressa como a conjunção de um número finito de disjunções, *cláusulas lógicas*, onde cada literal aparece apenas uma vez.
- *A Satisfação de uma expressão lógica* é um *problema de decisão*: existe alguma atribuição de valores lógicos aos literais que tornem a expressão lógica verdadeira?
- Para a expressão lógica ser satisfeita, todas as cláusulas lógicas devem ser satisfeitas.
- exemplo:

$$(x_1 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2) \wedge (x_2 \vee x_3)$$

- Os valores lógicos $x_2 =$V e $x_1 = x_3 =$F satisfazem a expressão lógica.

Satisfação Lógica e Programação Inteira

Satisfação de uma expressão lógica

- Os valores lógicos $x_2 =$V e $x_1 = x_3 =$F satisfazem

$$(x_1 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2) \wedge (x_2 \vee x_3)$$

Programação Inteira

- O ponto $(x_1, x_2, x_3)^t = (0,1,0)^t$ é um ponto válido do domínio definido pelas restrições:

$$\left\{ \begin{array}{l} x_1 + (1 - x_3) \geq 1 \\ (1 - x_1) + x_2 \geq 1 \\ x_2 + x_3 \geq 1 \\ x_1, x_2, x_3 \text{ binárias} \end{array} \right. \qquad \left\{ \begin{array}{l} x_1 - x_3 \geq 0 \\ -x_1 + x_2 \geq 0 \\ x_2 + x_3 \geq 1 \\ x_1, x_2, x_3 \text{ binárias} \end{array} \right.$$

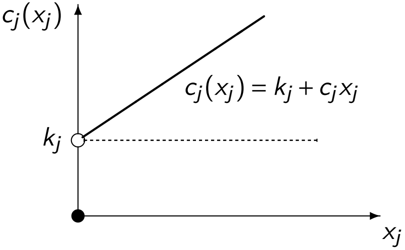
- Pode usar-se qualquer função objectivo (queremos encontrar um ponto válido).

Aplicações

Custo fixo

Objectivo: modelar o custo de produção de um lote de x_j artigos do tipo j, dado por uma função não-linear $c_j(x_j)$:

$$c_j(x_j) = \begin{cases} k_j + c_j x_j & , \text{ se } x_j > 0 \\ 0 & , \text{ se } x_j = 0 \end{cases}$$



- O *custo fixo de preparação*, k_j, só existe se $x_j > 0$, e
- os *custos variáveis*, $c_j x_j$, são uma função linear do número de unidades produzidas, x_j.

Custo fixo: modelação

- Variável de decisão binária y_j associada aos artigos do tipo j :

$$y_j = \begin{cases} 1 & \text{, se é produzido artigo do tipo } j \\ 0 & \text{, caso contrário} \end{cases}$$

É necessário estabelecer a relação entre x_j e y_j :

$$x_j \leq My_j,\ j = 1, 2, \ldots, n$$
$$x_j \geq 0 \text{ e inteiro},\ j = 1, 2, \ldots, n$$
$$y_j \text{ binário},\ j = 1, 2, \ldots, n$$

- M é um limite superior para a produção de artigos do tipo j.

função objectivo passa a incluir a parcela $k_j y_j$

- Se $x_j > 0 \Rightarrow y_j = 1$ (há custo fixo).
- Se $x_j = 0 \Rightarrow y_j$ pode assumir os valores 0 ou 1 (irá ter o valor 0, mais favorável do ponto de vista de minimização).

Exemplo: modelo apenas com custos variáveis (lineares)

Modelo: decidir quantos artigos produzir de cada tipo (o número total deve ser 100), obedecendo a uma restrição de capacidade, de modo a minimizar os custos de produção.

$$
\begin{array}{rl}
\text{min} & 4x_1 + 3x_2 + 4x_3 \\
\text{suj. a} & 1x_1 + 1x_2 + 1x_3 = 100 \\
 & 4x_1 + 6x_2 + 5x_3 \leq 480 \\
 & x_j \geq 0 \text{ e inteiro, } j = 1, 2, 3
\end{array}
$$

Solução óptima

Produzir 60 artigos de tipo 1 e 40 artigos de tipo 2, com um custo óptimo de 360.

Exemplo: modelo com custos fixos e custos variáveis

Há um custo de preparação da máquina, igual a 50, qualquer que seja o tipo de artigo:

$$c_1(x_1) = \begin{cases} 4x_1 + 50 & , \text{se } x_1 > 0 \\ 0 & , \text{se } x_1 = 0 \end{cases}$$

$$c_2(x_2) = \begin{cases} 3x_2 + 50 & , \text{se } x_2 > 0 \\ 0 & , \text{se } x_2 = 0 \end{cases}$$

$$c_3(x_3) = \begin{cases} 4x_3 + 50 & , \text{se } x_3 > 0 \\ 0 & , \text{se } x_3 = 0 \end{cases}$$

Questão: a existência de custos fixos de preparação favorece a produção de um maior ou de um menor número de tipos de artigos?

Exemplo: (cont.)

$$\begin{aligned}
\min \quad & 4x_1 + 3x_2 + 4x_3 + 50y_1 + 50y_2 + 50y_3 \\
\text{suj. a} \quad & 1x_1 + 1x_2 + 1x_3 = 100 \\
& 4x_1 + 6x_2 + 5x_3 \leq 480 \\
& x_1 \leq My_1 \\
& x_2 \leq My_2 \\
& x_3 \leq My_3 \\
& x_j \geq 0 \text{ e inteiro}, \; j = 1, 2, 3 \\
& y_j \text{ binário}, \; j = 1, 2, 3
\end{aligned}$$

M pode ser igual a 100.

Solução óptima

Produzir 100 artigos de tipo 1, com um custo óptimo de 450.

Localização de Serviços (por exemplo, armazéns)

Objectivo: seleccionar um conjunto de locais de serviço (dados os locais candidatos) e associar cada cliente a um local de serviço.

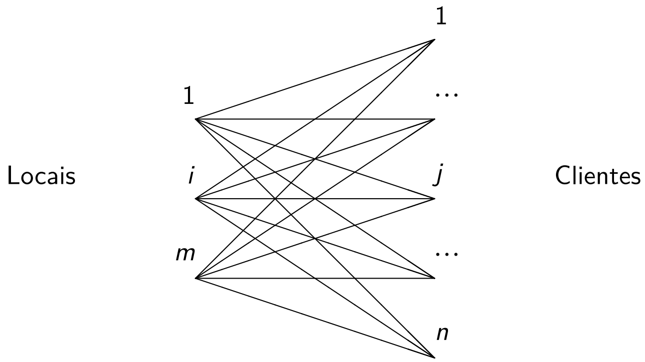
- Função objectivo: minimizar soma dos *custos de renda dos locais de serviço* e dos *custos de transporte entre esses locais de serviço e os clientes*.

Versão a analisar:

- capacidade de cada local é ilimitada, *i.e.*, é possível que haja apenas um local de serviço a servir todos os clientes.
- número de locais a seleccionar não é pré-determinado, e depende apenas dos custos.

- Conjunto de locais candidatos $I = \{1, \ldots, i, \ldots, m\}$.
- Conjunto de clientes a servir $J = \{1, \ldots, j, \ldots, n\}$.

Construção do modelo



c_{ij} : custo unitário de transporte entre o local i e o cliente j.
f_i : renda do local de serviço i (custo fixo).

$$x_{ij} = \begin{cases} 1 & \text{, se o cliente } j \text{ está associado ao local } i \\ 0 & \text{, caso contrário} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{, se o local } i \text{ é seleccionado} \\ 0 & \text{, caso contrário} \end{cases}$$

Modelo

$$
\begin{aligned}
min \quad & \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \\
suj. \quad & \sum_{i\in I} x_{ij} = 1 \ , \ \forall j \\
& y_i - x_{ij} \geq 0 \ , \ \forall i,j \\
& x_{ij} \text{ binário}, \ \forall i,j \\
& y_i \text{ binário}, \ \forall i
\end{aligned}
$$

- cada cliente j é associado apenas a um local i.
- restrições $y_i - x_{ij} \geq 0$ são implicações lógicas: $x_{ij} = 1 \Rightarrow y_i = 1$, *i.e.*, quando o cliente j está associado ao local i, deve haver aí um serviço.
- por outro lado, se $y_i = 0 \Rightarrow x_{ij} = 0, \ \forall j$
- quando um serviço é instalado num local candidato i, o respectivo custo f_i é imputado aos custos totais.

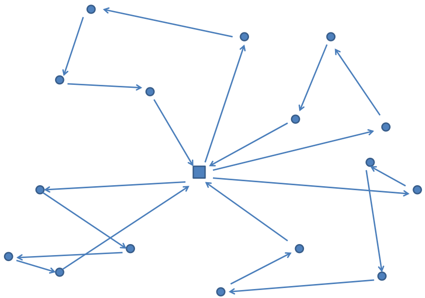
Planeamento de rotas de veículos

Objectivo

- Dados um conjunto de veículos com capacidades e
- um conjunto de clientes com procuras e janelas temporais de visita,

encontrar a solução de custo mínimo, em que haja

- um conjunto de rotas, todas começando e terminando no depósito,
- sendo cada cliente visitado por um único veículo.



Modelo

- V : conjunto de clientes.
- P : conjunto de todos os caminhos possíveis, cada um deles satisfazendo todas as restrições impostas ao problema.
- y_p : variável binária que indica se o caminho $p \in P$ é usado, ou não,
- c_p : custo de utilizar o caminho p, e
- $\delta_{ip} = \begin{cases} 1 & \text{, se o caminho } p \text{ visita o cliente } i \\ 0 & \text{, caso contrário} \end{cases}$
- Modelo de planeamento de rotas:

$$\begin{array}{lll} \min & \sum_{p \in P} c_p y_p & \\ \text{suj. a} & \sum_{p \in P} \delta_{ip} y_p = 1 & , \ \forall i \in V \\ & y_p \ \text{binário} & , \ \forall p \in P \end{array}$$

Exemplo com 8 clientes

- Há apenas 15 rotas possíveis.
- A título de exemplo, a rota correspondente à variável x_{15} visita os clientes 5 e 8, com um custo de 7 U.M..
- Modelo de partição de um conjunto:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	
cliente 1	1	1	1	1	1	1										$=1$
2	1	1					1	1	1	1	1					$=1$
3			1	1			1					1	1	1		$=1$
4	1							1	1			1				$=1$
5					1					1					1	$=1$
6		1	1					1					1			$=1$
7				1			1				1	1		1		$=1$
8						1					1		1	1	1	$=1$
custo	8	7	10	9	8	7	10	11	7	6	10	9	10	12	7	

Considerações

- Para determinar os caminhos possíveis, é necessário usar um grafo auxiliar em que se estabelece quais os clientes j que é possível visitar depois de ter visitado o cliente i.
- Para instâncias de pequena dimensão, é possível fazer a uma enumeração completa de todas as alternativas, e resolver o correspondente problema de programação inteira binária.
- Para instâncias de grande dimensão, recorre-se ao *método da geração diferida de colunas* (permite obter soluções gerando apenas um pequeno subconjunto das rotas possíveis).
- Resolvem-se, na prática, problemas com 100 clientes até à solução óptima (ou dentro de uma aproximação de 1% ou 2%).

Selecção de Projectos de Investimento

Objectivo: seleccionar os projectos de investimento que, respeitando as restrições, maximizam o valor presente (usando uma taxa de juro i).

Cada projecto tem um conjunto de VF_t (valor futuro): *cashflow* no ano t.

Usando as fórmulas de matemática financeira, o valor presente (VP):

$$VP = \sum_t \frac{FV_t}{(1+i)^t}$$

- exemplo: usando uma taxa de juro i=15%:

	Cashflow do Projecto			
ano	A	B	C	D
0	-500	-250	-400	-100
1	-600	-100	50	-40
2	200	150	200	20
3	500	350	200	180
4	700	0	100	100
5	300	0	100	0
VP	7.6	6.6	33.1	55.9

Construção do modelo

- Variável binária associada a cada projecto:

$$x_j = \begin{cases} 1 & \text{, se o projecto } j \text{ é seleccionado} \\ 0 & \text{, caso contrário} \end{cases}$$

Restrições:

- apenas um dos projectos B ou C pode ser seleccionado,
- a selecção do projecto D implica a selecção do projecto A,
- no ano 0, a companhia dispõe de 850 U.M., e
- no ano 1, a companhia dispõe de 750 U.M..

Modelo

- Apenas um dos projectos B ou C pode ser seleccionado, ou então, nenhum deles:

$$x_B + x_C \leq 1.$$

- A selecção do projecto D implica a selecção do projecto A:

$$x_A \geq x_D, \text{ ou seja, } x_D - x_A \leq 0.$$

- Capital disponível para investir no ano 0 e no ano 1:

$$\begin{aligned} 500x_A + 250x_B + 400x_C + 100x_D &\leq 850 \\ 600x_A + 100x_B - 50x_C + 40x_D &\leq 750 \end{aligned}$$

- No ano 1, o projecto C tem *cashflow* positivo, pelo que, a ser seleccionado, aumenta o capital disponível nesse ano.

Função objectivo (maximização do valor presente):

- $\max 7.6\ x_A + 6.6\ x_B + 33.1\ x_C + 55.9\ x_D$

Restrições activas e redundantes

- A variável binária y_1 pode ser usada para tornar uma restrição *activa* ou *redundante* (M deve ter um valor adequado):

$$A^1 x \leq b^1 + M(1 - y_1)$$

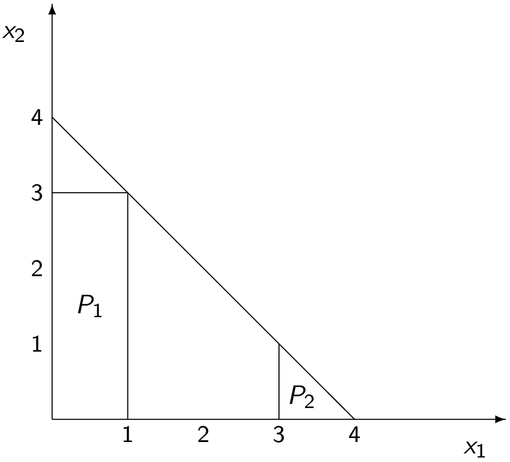
- A mesma ideia pode ser estendida para conjuntos de restrições:
- Há m conjuntos de restrições P_i :

$$P_i = \{x : A^i x \leq b^i, x \geq 0\}, \ \ i = 1, 2, \ldots, m$$

- Apenas k dos m conjuntos de restrições são activos:

$$\begin{aligned} & A^i x \leq b^i + M(1 - y_i), \ \ i = 1, 2, \ldots, m \\ & \sum_{i=1}^{m} y_i = k \\ & y_i \ \ \text{binário} \end{aligned}$$

Exemplo



- *Dicotomia:* domínio não-convexo que resulta da união dos domínios convexos P_1 e P_2, cada um deles definido através de um conjunto de restrições lineares.

Exemplo (cont.)

- Definição de cada um dos domínios convexos:

$$P_1 : \begin{cases} x_1 \leq 1 \\ x_2 \leq 3 \\ x_1, x_2 \geq 0 \end{cases} \qquad P_2 : \begin{cases} x_1 \geq 3 \\ x_1 + x_2 \leq 4 \\ x_1, x_2 \geq 0 \end{cases}$$

- Domínio não-convexo, união dos 2 domínios convexos:

$$\begin{array}{rcl} x_1 & \leq & 1 + M(1 - y_1) \\ x_2 & \leq & 3 + M(1 - y_1) \\ -x_1 & \leq & -3 + M(1 - y_2) \\ x_1 + x_2 & \leq & 4 + M(1 - y_2) \\ y_1 + y_2 & = & 1 \\ x_1, x_2 \geq 0 & , & y_i \ \text{binários} \end{array}$$

- Pode substituir-se y_1 por y, e y_2 por $(1 - y)$.

Planeamento de tarefas numa única máquina

Objectivo: sequenciar a execução das tarefas de modo a optimizar a medida de eficiência escolhida.

- As tarefas são definidas por um tempo de processamento p_j.
- Duas tarefas i e j não ocupam simultaneamente a máquina.

- Variável de decisão: $x_j =$ instante de início da execução da tarefa j.
- Restrições disjuntivas: o instante do fim da execução de uma tarefa é anterior ao instante de início da outra:

$$x_i + p_i \leq x_j \quad \text{ou} \quad x_j + p_j \leq x_i.$$

Planeamento numa única máquina: exemplo

- Dicotomia: duas tarefas não podem ocupar simultaneamente a máquina.
- Variável y_{ij} exprime a dicotomia:

$$y_{ij} = \begin{cases} 1 & \text{, se a tarefa } i \text{ precede a tarefa } j \\ 0 & \text{, se a tarefa } j \text{ precede a tarefa } i \end{cases}$$

- Restrições de não-simultaneidade:

$$x_i + p_i \leq x_j + M(1 - y_{ij})$$
$$x_j + p_j \leq x_i + My_{ij}$$

Outras restrições típicas de problemas de planeamento

- data de disponibilidade (*release date*): se uma tarefa só puder ser executada a partir de uma data r_j, podemos usar $x_j \geq r_j$.
- prazo de entrega (*due date*): se uma tarefa tiver que ser concluída antes da data limite de entrega d_j, podemos usar $x_j + p_j \leq d_j$.
- precedência: se a tarefa i tiver que ser executada obrigatoriamente antes da tarefa j, as restrições disjuntivas de não-simultaneidade são substituídas por $x_i + p_i \leq x_j$.

- (*completion date*): $C_j = x_j + p_j$: instante em que termina a execução da tarefa j.

Algumas medidas de eficiência de problemas de planeamento

- $\min C_{max} = max_j \ C_j$: minimizar o instante em que termina a execução da última tarefa. O instante C_{max} é usualmente designado por *makespan*.
- $\min L_{max} = max_j(C_j - d_j)$: minimizar o maior atraso (*maximum lateness*) existente numa das tarefas.
- $\min T = \sum_{j=1}^{n} T_j$: sendo $T_j = max[0, C_j - d_j]$, o atraso positivo (*tardiness*): minimizar a soma dos atrasos das tarefas que terminam atrasadas.
- $\min F = \sum_{j=1}^{n} C_j$: minimizar a soma dos tempos de permanência das tarefas no sistema.
- $\min U = \sum_{j=1}^{n} U_j$, sendo $U_j = \begin{cases} 1 & \text{, se } C_j > d_j \\ 0 & \text{, caso contrário} \end{cases}$: minimizar o número de tarefas em atraso, qualquer que seja o valor do atraso.

- É comum associar coeficientes de ponderação w_j às tarefas, para traduzir a sua importância relativa.

Considerações sobre arredondamento de variáveis

Relaxação de um modelo de programação inteira:

modelo em que algumas (ou todas as) condições de integralidade são ignoradas.

Relaxação das variáveis inteiras x_j

Após obter a solução óptima da relaxação, se se arredondarem os valores dos x_j^*, obtém-se uma solução com um valor que *geralmente* não é muito diferente do valor do óptimo do modelo inteiro.

Relaxação das variáveis binárias y_j

Após obter a solução óptima da relaxação, se se arredondarem os valores dos y_j^*, *geralmente* obtêm-se soluções não válidas, ou com valores muito diferentes dos do modelo inteiro.

Exemplo: modelo dos custos fixos (ver diapositivos anteriores)

Solução óptima

- $x_1 = 100, x_2 = 0, y_1 = 1, y_2 = 0$, com um custo óptimo de 450.

Relaxação das variáveis inteiras x_j

Solução óptima:

- $x_1 = 100, x_2 = 0, y_1 = 1, y_2 = 0$, com um custo óptimo de 450.

Solução coincide com solução óptima inteira, mas isso nem sempre acontece.

Relaxação das variáveis binárias y_j

Solução óptima:

- $x_1 = 60, x_2 = 40, y_1 = 0,6, y_2 = 0,4$, com um custo óptimo de 410.

Arredondando $y_1 = 1, y_2 = 0$, a solução não é válida, porque $x_2 = 40$, e há a restrição $x_2 \leq 100y_2$.

Considerações sobre qualidade de modelos

Qualidade de um modelo de programação inteira:

na prática, os modelos em que a diferença (*gap*) entre o óptimo do modelo inteiro (z_I^*) e o óptimo da respectiva relaxação linear (z_{PL}^*) é mais pequena são resolvidos pelos *software* de optimização de uma forma mais eficiente.

Solução óptima com y_j relaxados e $M = 100$

- $x_1 = 60, x_2 = 40, y_1 = 0,6, y_2 = 0,4$, com um custo óptimo de 410.
- $gap = z_I^* - z_{PL}^* = 450 - 410 = 40$

Solução óptima com y_j relaxados e $M = 1000$

- $x_1 = 60, x_2 = 40, y_1 = 0,06, y_2 = 0,04$, com um custo óptimo de 365.
- $gap = z_I^* - z_{PL}^* = 450 - 365 = 85$

- Não iremos aprofundar este tópico, que é crucial em programação inteira, teve avanços muito significativos nas últimas décadas, e continua a ser tema de investigação.

Fim