Modelos de Optimização

Maria Antónia Carravilla

Julho 1997

1. PROGRAMAÇÃO LINEAR 1

2. PROBLEMAS DE TRANSPORTES 8

2.1 Tipos de problemas de transportes 8

2.2 Exemplo de Problema de Transportes: 9
2.2.1 Formulação do problema: 9

2.3 Regras para a obtenção de uma solução (básica) inicial 11
2.3.1 Regra do canto NW 11
2.3.2 Regra dos custos mínimos 11
2.3.3 Exemplo 12

2.4 Desequilíbrio entre a oferta e a procura 14
2.4.1 Caso 1 - Oferta excessiva 14
2.4.2 Caso 2 - Procura excessiva 14
2.4.3 Problemas de Maximização 15

3. PROBLEMAS DE AFECTAÇÃO 16

3.1 Algoritmo Húngaro 19
3.1.1 Aplicação do Algoritmo Húngaro na resolução de um problema de afectação: 20

3.2 "The Bottleneck assignment problem" 22

4. ALGUNS PROBLEMAS EM REDES 24

4.1 Determinação do caminho mais curto 24
4.1.1 Algoritmo de Ford 24

4.2 Árvore de Suporte de Comprimento Mínimo 26
4.2.1 Algoritmo guloso ("greedy procedure") 26

4.3 Determinação do fluxo máximo numa rede de transportes (the maximal flow problem) .. 27

4.4 Determinação de CIRCUITOS EULERIANOS 30
4.4.1 O problema das pontes de Königsberg 30
4.4.2 O problema do carteiro chinês (chinese postman problem CPP) 31

4.5 Determinação de CIRCUITOS HAMILTONIANOS 31
4.5.1 Definição de Circuitos Hamiltonianos 31
4.5.2 O problema do Caixeiro Viajante (travelling salesman problem TSP) 32

5. PROGRAMAÇÃO DINÂMICA 33

1. PROGRAMAÇÃO LINEAR

Consideremos o caso, extremamente simplificado de uma empresa que pretende fabricar dois produtos A e B, usando os recursos que tem disponíveis, que são trabalho e materiais.

Para produzir uma unidade de A ou de B, são necessários as quantidades de cada um dos recursos apresentadas na tabela seguinte:

	Trabalho (horas/homem)	**Materiais (kg)**
A	2	4
B	1	5

Numa semana estão disponíveis as seguintes capacidades:

Trabalho	125 hom. · 40 horas
Materiais	15 000 kg

Admite-se que tudo o que é produzido será vendido, e que os lucros unitários para A e para B são respectivamente de 10 e de 7 conto (isto é, uma unidade vendida de A corresponde a um lucro de 10 contos).

Pretende-se saber quanto deve ser fabricado de cada um dos produtos, se quisermos, por exemplo, maximizar o lucro semanal.

Para isso vamos definir duas variáveis (correspondentes às decisões a tomar):

x_1 - quantidade a fabricar de A por semana;

x_2 - quantidade a fabricar de B por semana.

A quantidade de trabalho necessário para produzir x_1 unidades de A e x_2 unidades de B, será:

$$2x_1 + x_2$$

quantidade esta que não poderá ultrapassar a disponibilidade semanal do recurso trabalho.

A quantidade de materiais necessários para produzir x_1 unidades de A e x_2 unidades de B, será:

$4\ x_1 + 5\ x_2$

quantidade esta que não poderá ultrapassar a disponibilidade semanal do recurso materiais.

lucro obtido será:

$f = 10x_1 + 7x_2$

Obtém-se assim um modelo de optimização, da classe designada por **programação linear**, consistindo em:

maximizar:

$f = 10\ x_1 + 7\ x_2$

sujeito a:

$$\begin{array}{rcrcr} 2\ x_1 & + & x_2 & \leq & 5000 \\ 4\ x_1 & + & 5\ x_2 & \leq & 15000 \\ x_1 & , & x_2 & \geq & 0 \end{array}$$

A região das soluções admissíveis para este problema (isto é, possíveis de serem implementadas na prática) está representada graficamente a escuro na Figura 1.1 (onde os eixos representam as quantidades a produzir):

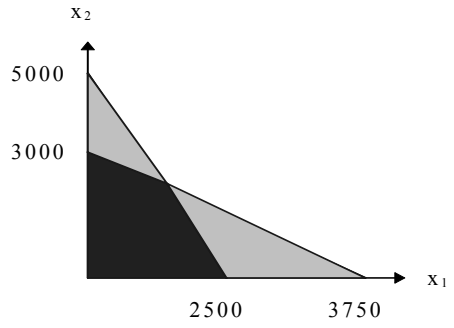


Figura 1.1- Região das soluções admissíveis

No quadro seguinte apresentam-se algumas soluções possíveis para este problema, e que coincidem com os vértices do polígono representado:

x_1	2500	**1666**	0	0
x_2	0	**1666**	3000	0
lucro	25000	**28322**	21000	0

A solução óptima encontra-se entre as soluções apresentadas no quadro, e é a que está a carregado (a essa solução corresponde obviamente o lucro máximo).

Neste caso os valores óptimos para as quantidades de A e de B são inteiros; no entanto podia não ter acontecido isso, isto é, as soluções podiam ser fraccionárias. Nesse caso, se não fossem permitidas soluções fraccionárias, seria necessário usar Programação Inteira.

Para resolver problemas de programação linear, podemos utilizar o algoritmo do SIMPLEX que se encontra disponível no mercado implementado em inúmeros "packages" de software (como é o caso do Sistema de Apoio ao Planeamento STORM).

Na Figura 1.2 apresenta-se a recta (a tracejado) da solução do problema com **lucro=21 000**.

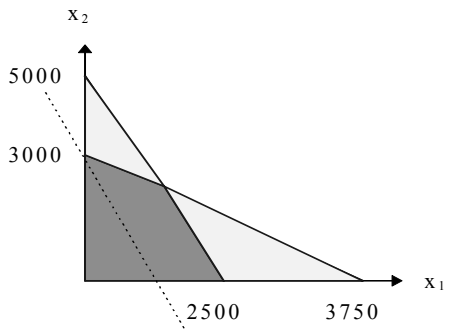


Figura 1.2 - Solução do problema com lucro = 21 000

Para resolver um problema pelo método Simplex, é necessário começar por passar o problema para a forma canónica. Na forma canónica as restrições terão que ser igualdades, sendo necessário introduzir variáveis de folga (slack).

Para passar as desigualdades de $\leq$ para igualdades, é necessário introduzir uma variável de folga s_i, para passar as desigualdades de $\geq$ para igualdades, é necessário introduzir uma variável de folga $-s_i$. Neste caso teremos que introduzir uma variável de folga s_1, correspondente à quantidade de material não usado e uma variável de folga s_2, correspondente à quantidade de trabalho não usado.

Forma canónica do problema de P.L.:

maximizar:

$f = 10\ x_1 + 7\ x_2$

sujeito a:

$$\begin{array}{rcrcrcr} 2\ x_1 & + & & x_2 & + & s_1 & = & 5000 \\ 4\ x_1 & + & 5\ x_2 & + & & s_2 & = & 15000 \\ x_1, & & x_2, & & & s_1 & \geq & 0 \\ & & & & & , & & \end{array}$$

Forma tabular:

BASE	x_1	x_2	s_1	s_2	
s_1	2	1	1	0	5000
s_2	4	5	0	1	15000
f	10	7	0	0	0

Neste momento s_1, s_2 estão na base e por isso são diferentes de 0, e x_1, x_2 estão fora da base e por isso são =0.

Uma solução básica admissível para este problema seria: $\begin{cases} s_1=5 \\ s_2=15000 \\ x_1=0 \\ x_2=0 \end{cases}$, que corresponde a um vértice (a origem) do polígono das soluções admissíveis.

Inicia-se agora um processo iterativo, em que por cada iteração uma variável entra na base (vértice adjacente do poliedro), com aumento do valor de f, até se atingir o óptimo.

Exercício:

Resolver este problema , utilizando o algoritmo SIMPLEX

Regras:

- Entra na base a variável com coeficiente mais positivo em f, x_1 neste caso.
- Sai da base a variável que primeiro se anula, quando a que entra vai aumentando de valor, $s_1$neste caso.

Ir resolvendo o sistema de equações para as variáveis que estão na base

BASE	x_1	x_2	s_1	s_2	
<- s_1	2	1	1	0	5000

s_2	4	5	0	1	15000
f	10	7	0	0	0
	↑				

A variável que entra na base é x_1, e a variável que sai da base é s_1, dado que:

$s_1=5000 - 2x_1 - x_2$

$s_2=15000 - 4x_1 - 5x_2$

BASE	x_1	x_2	s_1	s_2	
x_1	1	0.5	0.5	0	2500
<- s_2	0	3	-2	1	5000
f	10	2	-5	0	-25000
		↑			

A variável que entra na base é x_2, e a variável que sai da base é s_2.

BASE	x_1	x_2	s_1	s_2	
x_1	1	0	$\frac{5}{6}$	$\frac{1}{6}$	1666
x_2	0	1	$-\frac{2}{3}$	$\frac{1}{3}$	1666
f	10	2	$-\frac{11}{3}$	$\frac{2}{3}$	$\frac{85000}{3}$

Solução óptima: $\begin{cases} x_1=1666 \\ x_2=1666 \\ s_1=0 \\ s_2=0 \end{cases}$

No caso de se tratar de um problema de minimização, pode-se fazer min f = max (-f) ou usar um critério inverso de entrada das variáveis na base, (entra a que tem um coeficiente mais negativo)

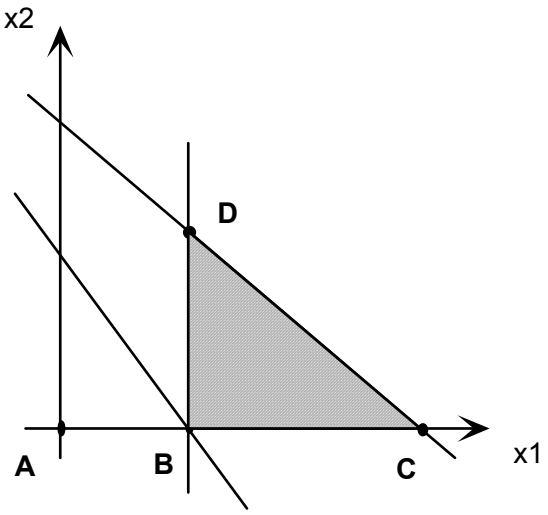
Situações especiais que podem surgir na resolução de um problema de programação linear:

1. Inexistência de soluções admissíveis;
2. Solução óptima não limitada;
3. Mais do que uma solução óptima;
4. Empate no critério de entrada na base;

5. Degenerescência.

No caso de haver desigualdades de $\geq$ ou igualdades, devem-se introduzir variáveis artificiais, para obtenção de uma solução inicial.

Exemplo:



maximizar: $2x_1 + x_2$

sujeito a:

$$\begin{array}{lcl} x_1 & \geq & 1 \\ x_1 + x_2 & \leq & 3 \\ x_1 \; ; \; x_2 & \geq & 0 \end{array}$$

Forma canónica:

maximizar: $2x_1 + x_2$

sujeito a:

$$\begin{array}{llllll} x_1 & - & s_1 & + & \mathbf{a_1} & = 1 \\ x_1 + x_2 & & + s_2 & & & = 3 \end{array}$$

onde $\mathbf{a_1}$ é uma variável artificial, que se introduz apenas para obter uma solução básica admissível. Assim o problema será dividido em duas fases: Numa primeira fase vamos minimizar uma função objectivo igual ao somatório das variáveis artificiais:

fase 1 min $F = \_a_i = a_1$

O mínimo corresponderá a $\forall a_i = 0$ e $F=0$.

Numa segunda fase vamos então maximizar a função objectivo f:

fase 2 max f

Segue-se o primeiro quadro simplex para o problema do exemplo:

	x_1	x_2	s_1	s_2	a_1	
a_1	1	0	-1	0	1	1
s_2	1	1	0	1	0	3
min F	0	0	0	0	1	0
	-1	0	1	0	0	-1
max f	2	1	0	0	0	0

Para resolver a fase 1, vamos começar por minimizar F. Nesse caso, x_1 entra na base e a_1 sai da base. Como todas as variáveis artificiais já saíram da base (F=0), pode-se eliminar a linha de F e a coluna de a_1 do quadro, e passamos a ter um problema semelhante ao do primeiro exemplo, podendo prosseguir agora na tentativa de maximizar f.

2. PROBLEMAS DE TRANSPORTES

2.1 Tipos de problemas de transportes

Existem diferentes tipos de problemas de transportes, dos quais se apresentam exemplos a seguir. Na Figura 2.1 está representado um problema de transportes "básico", com um conjunto de círculos que correspondem a origens e um conjunto de círculos que correspondem a destinos. Existe uma ligação entre uma origem e um destino, sempre exista "capacidade transporte" entre eles.

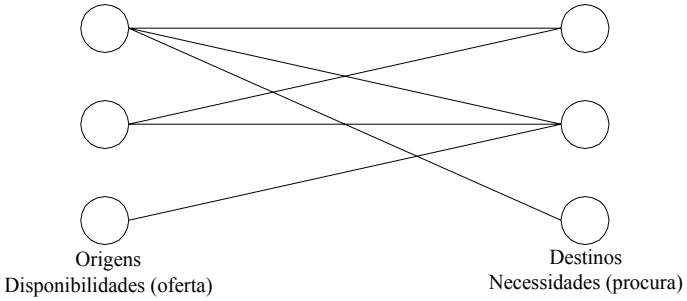


Figura 2.1 - Transportes ("transportation")

Na Figura 2.2 está representado o caso em que existem depósitos intermédios entre as origens e os destinos.

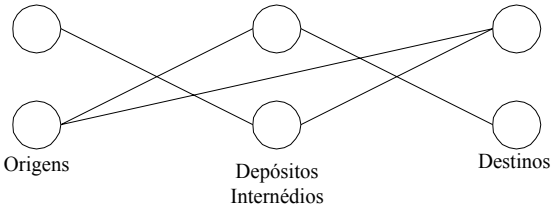


Figura 2.2 - Transferências

Pode-se também considerar uma situação em que existam transferências limitadas, que correspondem a que existam limitem de capacidade de transporte.

2.2 Exemplo de Problema de Transportes:

Um produto é fabricado em 3 fábricas F_1, F_2 e F_3 e deve ser transportado para três clientes C_1, C_2 e C_3.

Disponibilidades

F_1	2
F_2	3
F_3	5

Necessidades

C_1	4
C_2	1
C_3	5

Custos Unitários de Transporte ($)

	C_1	C_2	C_3
F_1	7	3	4
F_2	2	1	3
F_3	3	4	6

Que quantidade transportar de cada origem para cada destino, por forma a minimizar o custo total de transporte?

2.2.1 Formulação do problema:

Seja x_{ij} a quantidade transportada da fábrica F_i para o cliente C_j

Custos unitários	C_1	C_2	C_3	dispon. (oferta)
F_1	7	3	4	2
F_2	2	1	3	3
F_3	3	4	6	5
necess. (procura)	4	1	5	10

Para que o problema de transportes se enquadre no formato adequado à aplicação de algoritmos (forma standard), é necessário que:

$$\sum \text{oferta} = \sum \text{procura}$$

Se for c_{ij} o custo unitário de transporte de i para j, temos:

minimizar: $\sum_i \sum_j c_{ij} x_{ij} = 7x_{11} + 3x_{12} + 4x_{13} + 2x_{21} + x_{22} + 3x_{23} + 3x_{31} + 4x_{32} + 6x_{33}$

sujeito a:

1.	x_{11}	+	x_{12}	+	x_{13}	=	2
2.	x_{21}	+	x_{22}	+	x_{23}	=	3
3.	x_{31}	+	x_{32}	+	x_{33}	=	5

restrições da oferta (i=1,2,3)

4.	x_{11}	+	x_{21}	+	x_{31}	=	4
5.	x_{12}	+	x_{22}	+	x_{32}	=	1
6.	x_{13}	+	x_{23}	+	x_{33}	=	5

restrições da procura (j=1,2,3)

$\forall_{ij} x_{ij} \geq 0$

Somando as equações 1., 2. e 3., obtemos: $\sum_i \sum_j x_{ij} = 10$

e somando as equações 4., 5. e 6. obtemos igualmente: $\sum_i \sum_j x_{ij} = 10$.

Assim, podemos afirmar que pelo menos uma das 6 equações é linearmente dependente das outras, pelo que basta considerar 5 equações "efectivas", isto é:

Número de equações = número de origens + número de destinos - 1

O problema de transportes é um problema de programação linear, com uma estrutura particular (todos os coeficientes das variáveis nas restrições são ou 0 ou 1).

Na resolução destes problemas, ter-se-á em cada iteração, um quadro como o seguinte:

	C_1	C_2	C_3	
F_1	7	3	4	2
F_2	2	1	3	3
F_3	3	4	6	5
	4	1	5	

onde cada célula será preenchida com o valor actual da variável correspondente, ou seja:

	C_J
F_I	x_{IJ}

7

Note-se que só 5 (3+3-1) dos x_{IJ} serão diferentes de zero (constituindo as variáveis básicas da solução actual). Desta observação resulta que uma solução óptima para este problema não poderá nunca conter mais do que 5 variáveis com valor positivo (ou seja, no máximo, apenas serão utilizadas 5 de entre as 9 "estradas" existentes).

2.3 Regras para a obtenção de uma solução (básica) inicial

2.3.1 Regra do canto NW

Começa-se a preencher as variáveis pelo canto NW do quadro (variável F_1-C_1), carregando-as o mais possível e passando sucessivamente para as variáveis adjacentes.

	C_1	C_2	C_3	
F_1	**2** 7	- 3	- 4	2
F_2	**2** 2	**1** 1	- 3	3
F_3	- 3	**0** 4	**5** 6	5
	4	1	5	

Considerou-se uma variável na base com o valor 0 (devidamente assinalada no quadro), para que a base ficasse com 5 elementos (trata-se de uma questão puramente técnica para efeitos do algoritmo optimizante que seria aplicado na sequência).

2.3.2 Regra dos custos mínimos

Começa-se a preencher o quadrado de c_{ij} mínimo, passando-se em seguida para o de custo unitário imediatamente superior, e assim sucessivamente (note-se que neste caso, o número de variáveis na base é 5).

	C_1	C_2	C_3	
F_1	- 7	- 3	**2** 4	2
F_2	**2** 2	**1** 1	- 3	3

F_3	**2** 3	**-** 4	**3** 6	5
	4	1	5	

2.3.3 Exemplo

Neste exemplo, mostra-se como passar de uma solução possível para outra, pela introdução na base de uma variável não básica, ou seja passando a utilizar uma nova estrada. Para tal é necessário fazer um conjunto de compensações por forma a que não sejam violadas as restrições da oferta e da procura (como, por exemplo, de uma fábrica só pode ser enviado aquilo que aí é produzido).

Concretamente, vamos supor que, partindo da situação representada no quadro seguinte, e cujo custo é de 49 unidades de dinheiro), se considera ser vantajoso passar a utilizar a estrada F_1-C_3. Seja θ a quantidade a transportar por essa estrada, e cujo valor se pretende determinar.

	C_1	C_2	C_3	
F_1	2 7	- 3	θ 4	2
F_2	2 2	1 1	- 3	3
F_3	- 3	0 4	5 6	5
	4	1	5	

(Custo desta solução: $2x_7 + 2x_2 + 1x_1 + 0x_4 + 5x_6 = 49$)

Para que, com a introdução de θ, não se crie nenhum desequilíbrio na oferta ou na procura, é necessário proceder a um conjunto de compensações (por linha e por coluna) como as indicadas a seguir.

	C_1	C_2	C_3	
F_1	**2- θ** 7	**-** 3	**θ** 4	2
F_2	**2+ θ** 2	**1- θ** 1	**-** 3	3
F_3	**-** 3	**0+ θ** 4	**5- θ** 6	5
	4	1	5	

É agora fácil de concluir que o maior valor possível para θ é dado por $\theta = \min\{2,1,5\} = 1$, o que, após substituição, conduzirá à seguinte solução de custo igual a 45:

$u_i \backslash v_j$	C_1	C_2	C_3	
F_1	**1** 7	- 3	**1** 4	2
F_2	**3** 2	- 1	- 3	3
F_3	- 3	**1** 4	**4** 6	5
	4	1	5	

Custo da nova solução: $1x_7 + 1x_4 + 3x_2 + 1x_4 + 4x_6 = 45$.

2.4 Desequilíbrio entre a oferta e a procura

2.4.1 Caso 1 - Oferta excessiva

É necessário considerar um consumidor fictício que absorva o excesso de oferta. Para isso, introduz-se uma coluna adicional com custos de transporte adequados.

No exemplo anterior, considere-se agora que F_1 produz 5 unidades em vez de 2:

	C_1	C_2	C_3	C^*	
F_1	7	3	4		5
F_2	3	1	3		3
F_3	3	4	6		5
	4	1	5	**(3)**	

2.4.2 Caso 2 - Procura excessiva

É necessário considerar uma origem (fábrica) fictícia que produza a quantidade em falta. Para tal, introduz-se no quadro uma linha adicional com custos de transporte adequados, e função da situação particular em estudo.

Ainda no exemplo anterior, considere-se agora que C_2 consome 3 unidades em vez de 1:

	C_1	C_2	C_3	
F_1	7	3	4	2
F_2	3	1	3	3
F_3	3	4	6	5
F*				(2)
	4	3	5	

2.4.3 Problemas de Maximização

Existem problemas de planeamento com a estrutura de problemas de transporte, em que, em vez de se procurar minimizar custos, se procura maximizar uma dada medida de eficiência (como, por exemplo, o lucro).

Nesse caso, para utilizar os procedimentos standard de resolução, deve-se:

- fazer C_{ij}=lucro unitário associado à utilização emparelhada de F_i e de C_j
- usar um critério de entrada na base inverso (isto é, passar a utilizar estradas que induzam um aumento no valor da função objectivo).

3. PROBLEMAS DE AFECTAÇÃO

Considere-se, a título de exemplo, o seguinte problema de optimização, representativo da classe designada por Problemas de Afectação ("assignment problems"):

Uma companhia tem m tarefas para serem realizadas, e m empregados que as podem executar (possivelmente gastando tempos diferentes, em função da sua diferente especialização).

Que empregado deverá ser afectado a cada tarefa, de modo a minimizar o tempo global para completar todas as m tarefas (e considerando que cada empregado só pode ser afectado a uma tarefa)?

Numa matriz dos custos de afectação, representar-se-ão os custo das diferentes afectações (emparelhamentos) dos operários às tarefas. Em termos gerais, poderá considerar-se a afectação de quaisquer recursos a diferentes actividades.

Os custos de afectação podem ser:

- **positivos**, correspondendo a custos efectivos de operação, tempo gasto, etc., e nesse caso trata-se de um problema de minimização;
- **negativos**, correspondendo a lucros, produtividade, ou quaisquer outras medidas de eficiência, tratando-se nesse caso de um problema de maximização.

Continuando a apresentar o exemplo inicial, consideremos que a **seguinte matriz dos custos de afectação**, representa correctamente os tempos (em semanas), gastos por cada operário a realizar cada tarefa:

		Tarefas			
		I	**II**	**III**	**IV**
	A	2	10	9	7
Operários	**B**	15	4	14	8
	C	13	14	16	11
	D	4	15	13	9

Este problema pode ser formulado como um problema de programação linear:

Seja :

$x_{ij} = \begin{cases} 0, \text{ se o recurso i não for afectado à actividade j} \\ 1, \text{ se o recurso i for afectado à actividade j} \end{cases}$

c_{ij} = custo associado à afectação de i a j

sendo neste caso i = A, B, C, D e j = I, II, III, IV.

O que se pretende é:

minimizar $Z = \sum_i \sum_j c_{ij} x_{ij}$

sujeito a : $\begin{cases} \sum_j x_{ij} = 1 \\ \sum_i x_{ij} = 1 \\ x_{ij} = 1 \vee 0 \end{cases}$

Este problema é obviamente um caso particular de **Problema de Transportes**, se se considerarem as restrições mais fracas $0 \le x_{ij} \le 1$, em vez de impor que $x_{ij} = 1$ ou 0. Permitir que tal aconteça não altera, neste caso, a solução óptima (ou seja, é possível "relaxar" a restrição de integridade das variáveis sem que isso "piore" o resultado obtido).

EXEMPLO:

$$\min Z = 2x_{AI} + 10x_{AII} + 9x_{AIII} + 7x_{AIV} + 15x_{BI} + 4x_{BII} \cdots + 9x_{DIV}$$

sujeito a : $\begin{cases} x_{AI} + x_{AII} + x_{AIII} + x_{AIV} = 1 \\ x_{B1} + x_{BII} + x_{BIII} + x_{BIV} = 1 \\ \vdots \\ x_{AI} + x_{BI} + x_{CI} + x_{DI} = 1 \\ \vdots \\ x_{AIV} + x_{BIV} + x_{CIV} + x_{DIV} = 1 \\ \forall_{ij} x_{ij} = 0 \vee 1 \end{cases}$

3.1 Algoritmo Húngaro

Trata-se de um algoritmo muito eficiente em termos computacionais, que tira partido das características estruturais particulares do problema de afectação.

Propriedades da matriz dos custos de afectação:

A. Pode-se somar ou subtrair uma constante a cada linha ou coluna, sem alterar a conjunto das afectações óptimas;

B. Se, por transformações sucessivas, obtivermos pelo menos um "0" em cada linha ou coluna, a afectação que corresponde a esses "0", resulta no custo global mínimo.

Definição: Um conjunto de elementos da matriz das eficiências diz-se independente se não existirem 2 elementos na mesma linha ou coluna.

Algoritmo (processo iterativo):

1. Com o mínimo de "traços", tentar cobrir todos os zeros da matriz;

2. O número de traços feitos, corresponde ao número de zeros independentes. Se o número de traços feitos for igual a m, está encontrada nesses zeros a afectação óptima (nesses zeros);

3. Dos elementos não "traçados", escolhe-se o menor;

4. Subtrai-se esse elemento a todos os "não traçados";

5. Soma-se esse elemento aos que estão "traçados duas vezes";

Voltar a **1.**

Note-se que os passos **4.** e **5.**, são uma aplicação directa da propriedade **A.** e equivalem a :

1. Subtrair o elemento escolhido a todas as linhas;

2. Somá-lo às linhas e colunas "traçadas".

3.1.1 Aplicação do Algoritmo Húngaro na resolução de um problema de afectação:

	I	*II*	*III*	*IV*
A	2	10	9	7
B	15	4	14	8
C	13	14	16	11
D	4	15	13	9

PASSO 0:

"Baixar" o mais possível os valores da matriz eficiência

Subtrair o menor elemento de cada linha a todos os elementos dessa linha:

	I	*II*	*III*	*IV*
(−2)	0	8	7	5
(−4)	11	0	10	4
(−11)	2	3	5	0
(−4)	0	11	9	5

Subtrair o menor elemento de cada coluna, a todos os elementos dessa coluna:

			(−5)	
A	0	8	2	5
B	11	0	5	4
C	2	3	0	0
D	0	11	4	5

PASSO 1:

"Traçar" todos os zeros com o menor número possível de traços:

~~0~~	~~8~~	(2)	5
~~11~~	~~0~~	5	4
~~2~~	~~3~~	~~0~~	~~0~~
~~0~~	~~11~~	4	5

PASSO 2:

Quantos "traços" (zeros independentes)?

3 < m = 4

PASSO 3:

Qual o menor elemento dos não traçados?

a = 2

PASSO 4:

Subtrair **a** a todos os não traçados *****;

PASSO 5:

Somar **a** aos traçados 2 vezes **°**:

0	8	*0	*3
11	0	*3	*2
°4	°5	0	0
0	11	*2	*3

PASSO 1:

"Traçar" os zeros com o menor número possível de traços:

~~0~~	~~8~~	~~0~~	~~3~~
~~11~~	~~0~~	3	2
~~4~~	~~5~~	~~0~~	~~0~~
~~0~~	~~11~~	2	3

PASSO 2:

Quantos "traços" (zeros independentes)?

4 = m (existem 4 zeros independentes)

Logo: escolhendo quatro zeros independentes, isto é, que não pertençam a linhas ou colunas comuns, fica determinada a afectação óptima, que terá alternativas no caso de existirem outros conjuntos de zeros independentes.

Neste caso:

	I	II	III	IV
A	0	8	(0)	3
B	11	(0)	3	2
C	4	5	0	(0)
D	(0)	11	2	5

Ao empregado A é atribuída a tarefa III, ao empregado B é atribuída a tarefa II, ao empregado C é atribuída a tarefa IV, e ao empregado D é atribuída a tarefa V;

sendo a "custo" mínimo obtido (ou seja, o tempo gasto globalmente na realização das tarefas).de

$$9 + 4 + 11 + 4 = 28.$$

3.2 "The Bottleneck assignment problem"

Pretende-se, nesta variante do problema de afectação, que a tarefa mais demorada demore o menos tempo possível. Este problema corresponde, por exemplo, à situação em que se pretende realizar um projecto constituído por um conjunto de tarefas, para cuja execução se dispõe de uma equipa permanentemente disponível.

Considere-se, para os dados do problema anterior, uma afectação qualquer (), assinalada sobre a matriz dos custos de afectação:

	I	II	III	IV
A	2	9	2	(7)
B	6	(8)	7	6
C	4	6	(5)	3
D	(4)	2	7	3

O tempo de execução mais demorado é 8. Vejamos se é possível fazer uma afectação só com valores inferiores a 8 (vamos assinalar esses valores com ϕ):

	I	II	III	IV
A	ϕ	9	ϕ	(ϕ)7
B	(ϕ)6	8	ϕ	ϕ
C	ϕ	ϕ	(ϕ)5	ϕ
D	ϕ	(ϕ)2	ϕ	ϕ

É óbvio que se podem definir imensas afectações com os pontos assinalados com ϕ (uma é por exemplo a indicada, cuja tarefa mais demorada tem uma duração de 7).

E com valores inferiores a 7, será possível fazer uma afectação? Vamos novamente assinalar esses valores com ϕ:

	I	II	III	IV
A	ϕ	9	(ϕ)2	7
B	(ϕ)6	8	7	ϕ
C	ϕ	ϕ	ϕ	(ϕ)3
D	ϕ	(ϕ)2	7	ϕ

A afectação assinalada é uma das possíveis, sendo a duração da actividade mais demorada igual a 6.

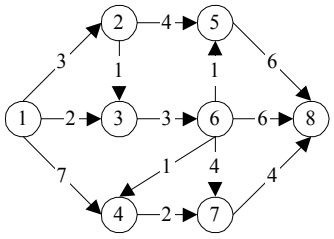
Com valores inferiores a 6, não é possível definir qualquer afectação, pelo que a **solução óptima** segundo o critério definido, será a dada pelo quadro acima indicado, ou por qualquer das alternativas que dele se podem extrair.

Este critério usa-se quando se pretende minimizar a ineficiência individual, e não o conjunto da afectação.

4. ALGUNS PROBLEMAS EM REDES

4.1 Determinação do caminho mais curto

4.1.1 Algoritmo de Ford

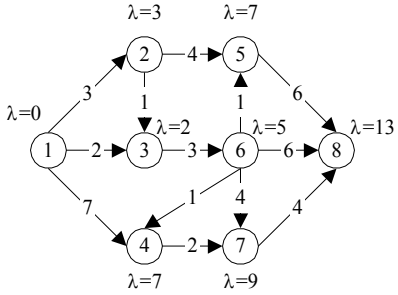


l_{ij} - distância do nó (i) ao nó (j) (l_{15}=1)

1. enumerar os nós, impondo que a entrada da rede seja o nó 1 e a saída seja o nó n;

2. afectar todos os nós i (i diferente de 1) de um parâmetro $\lambda_i = +\infty$, fazendo $\lambda_1 = 0$.

i=1	$\lambda_1 = 0$	$l_{12} = 3$	$\lambda_1+l_{12} = 3 < +\infty$	$\rightarrow$	$\lambda_2 = 3$
		$l_{13} = 2$	$\lambda_1+l_{13} = 2 < +\infty$	$\rightarrow$	$\lambda_3 = 2$
		$l_{14} = 7$	$\lambda_1+l_{14} = 7 < +\infty$	$\rightarrow$	$\lambda_4 = 7$
i=2	$\lambda_2 = 3$	$l_{25} = 4$	$\lambda_2+l_{25} = 7 < +\infty$	$\rightarrow$	$\lambda_5 = 7$
		$l_{23} = 1$	$\lambda_2+l_{23} = \underline{4 > 2}$	$\rightarrow$	λ_3 mantém
i=3	$\lambda_3 = 2$	$l_{36} = 3$	$\lambda_3+l_{36} = 5 < +\bullet$	$\rightarrow$	$\lambda_6 = 5$
i=4	$\lambda_4 = 7$	$l_{47} = 2$	$\lambda_4+l_{47} = 9 < +\infty$	$\rightarrow$	$\lambda_7 = 9$
i=5	$\lambda_5 = 7$	$l_{58} = 6$	$\lambda_5+l_{58} = 13 < +\infty$	$\rightarrow$	$\lambda_8 = 13$



i=6	$l_6 = 0$	$l_{64} = 1$	$\lambda_6+l_{64} = 6 < 7$	→	$\lambda_4 = 6$
				(atenção:	j<l)
		$l_{65} = 3$	$\lambda_6+l_{65} = 8 > 7$	→	λ_5 mantém
		$l_{67} = 4$	$\lambda_6+l_{67} = 9 = 9$	→	λ_7 mantém
		$l_{68} = 6$	$\lambda_6+l_{68} = 11 < 13$	→	$\lambda_8 = 11$
i=4	$l_4 = 6$	$l_{47} = 2$	$\lambda_4+l_{47} = 8 < 9$	→	$\lambda_7 = 8$
i=7	$l_7 = 8$	$l_{78} = 4$	$\lambda_7+l_{78} = 12 > 11$	→	λ_8 mantém
				$l_8 = 11$	

A distância mínima (ou o "custo associado") entre o nó 1 e o nó 8 é de 11 unidades.

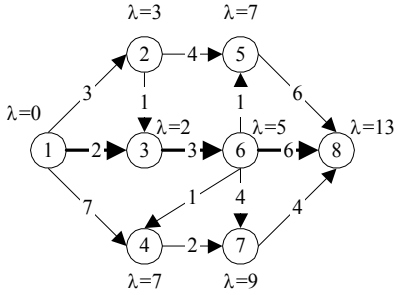
Como determinar o caminho associado a esta distância ?

1. $i\tau^{-1}(8) = \{5,6,7\}$ $\lambda_8-\lambda_6 = 11-5 = 6$ $l_{68} = 6$

reparar que: $\lambda_8-\lambda_5 = 11-7 \neq l_{58} = 6$

2. $\tau^{-1}(6) = \{3\}$ $\lambda_6-\lambda_3 = 5-2 = 3$ $l_{36} = 3$

3. $\tau^{-1}(3) = \{1,2\}$ $\lambda_3-\lambda_1 = 2-0 = 2$ $l_{13} = 2$



Logo, o caminho de custo mínimo é: nó 1, nó 3, nó 6, nó 8.

4.2 Árvore de Suporte de Comprimento Mínimo

Definições (para grafos não orientados):

- uma **árvore** é um grafo conexo que não contém ciclos;
- um grafo diz-se **conexo** se existir uma cadeia (sequência de ramos) ligando qualquer par de nós entre si.

Problema:

Determinar a árvore de comprimento total mínimo que suporte todos os nós da rede (i.e. que ligue todos os nós da rede) ("minimal spanning tree").

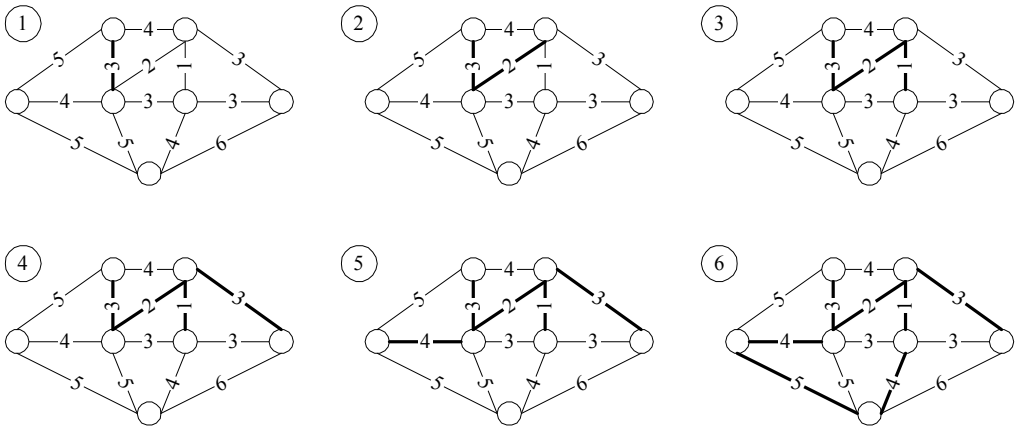
Aplicações:

Redes de comunicações / redes de distribuição de energia.

4.2.1 Algoritmo guloso ("greedy procedure")

1. Seleccionar um nó arbitrariamente, e ligá-lo ao nó mais próximo;
2. Identificar o nó ainda isolado que esteja mais próximo de um nó já ligado, e ligar estes dois nós;

Repetir **2.** até que todos os nós estejam ligados entre si.



PARA UMA REDE COM N NÓS A SHORTEST SPANNING TREE TEM N-1 RAMOS

4.3 Determinação do fluxo máximo numa rede de transportes (the maximal flow problem)

Problema:

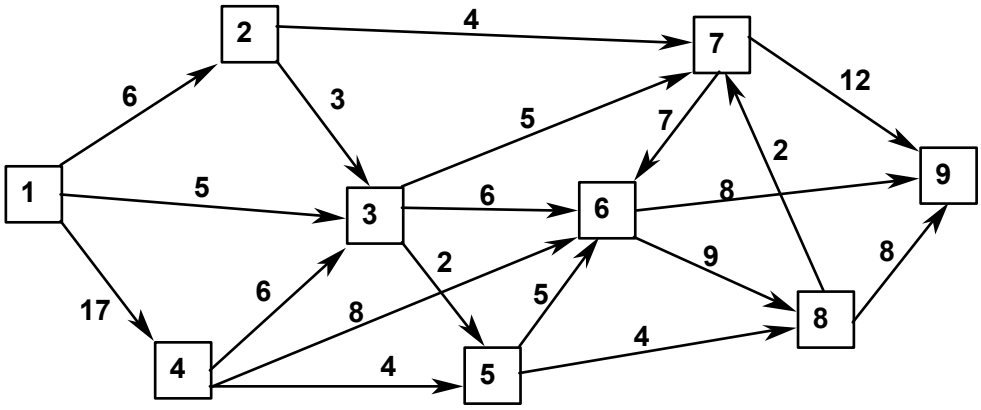
Determinar o máximo fluxo que é possível fazer passar entre dois nodos de uma rede (que se designa por rede de transportes), em que os números que quantificam os arcos indicam capacidades de transporte.

Algoritmo de **Ford Fulkerson**

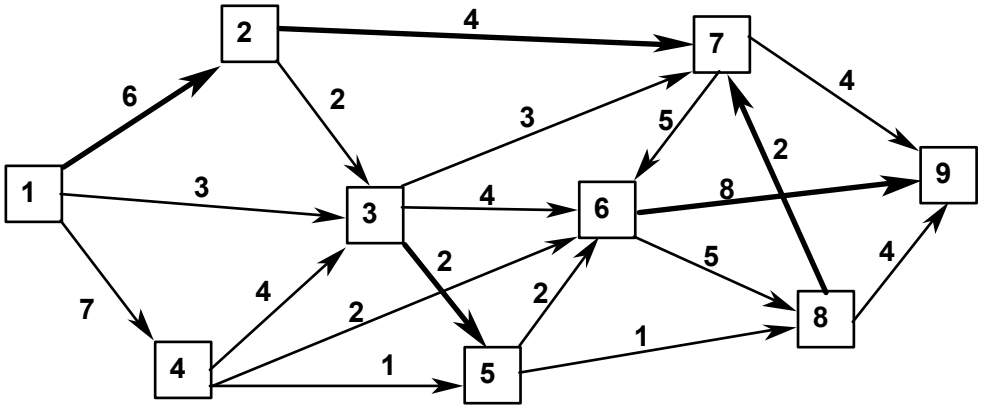
1. Fazer passar um fluxo arbitrário;
2. Encontrar um fluxo completo(i.e. em que todos os caminhos têm, pelo menos, um arco saturado); (um arco diz-se saturado, quando o fluxo é igual à capacidade)
3. Marcar os nós do seguinte modo:
   - marcar um nó de entrada (+);
   - dado um qualquer nó x_i que esteja marcado, marcar de (+i) todo o nó x_j se existir o arco $\overline{x_i,x_j}$ não saturado;
   - marcar com (-i) todos os nós x_j ainda não marcados, desde que exista o arco $\overline{x_j,x_i}$ de fluxo não nulo, e x_i esteja marcado;
   - se, por este processo, se conseguir marcar a saída da rede, o fluxo não é máximo; senão FIM
4. No caso do fluxo completo não ser máximo, alterar o fluxo, (ao longo dos arcos que ligam a entrada à saída, só por nós marcados), por forma a aumentar o fluxo que atinge a saída -> o fluxo, em pelo menos um dos arcos considerados, tornar-se-á máximo ou nulo.
5. Voltar a (**3.**)

Exemplo:

Os valores c_{ij} representam a capacidade de cada arco, por exemplo: $c_{37}=5$



i] Fluxo arbitrário (alguns dos arcos encontram-se já saturados)

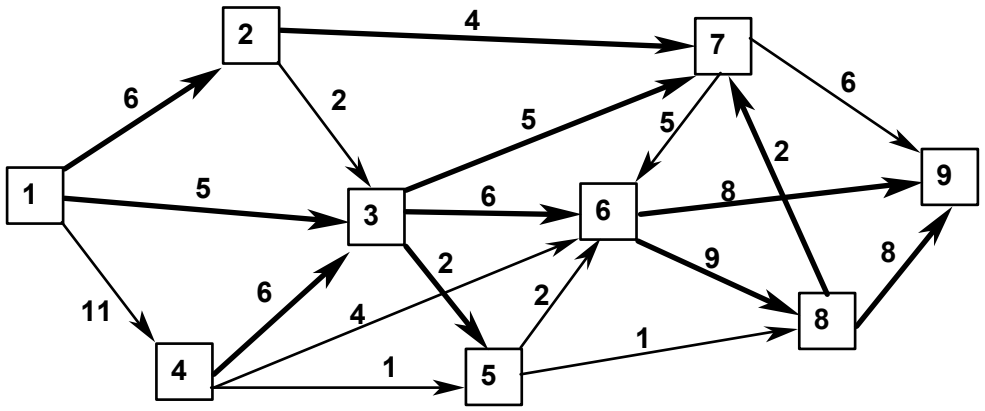


ii] Obtenção de um fluxo completo:

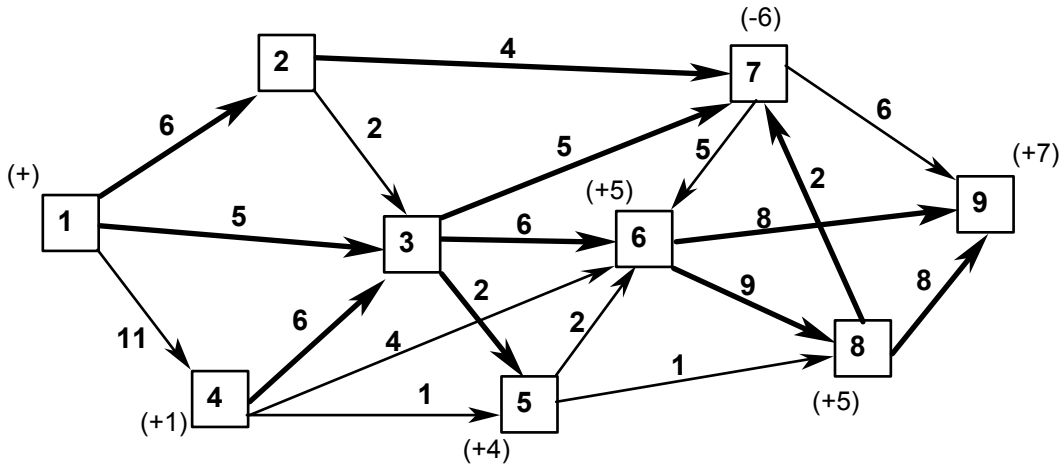
-aumentar duas unidades em (1,3,7,9);

-aumentar 2 unidades em (1,4,3,6,8,9);

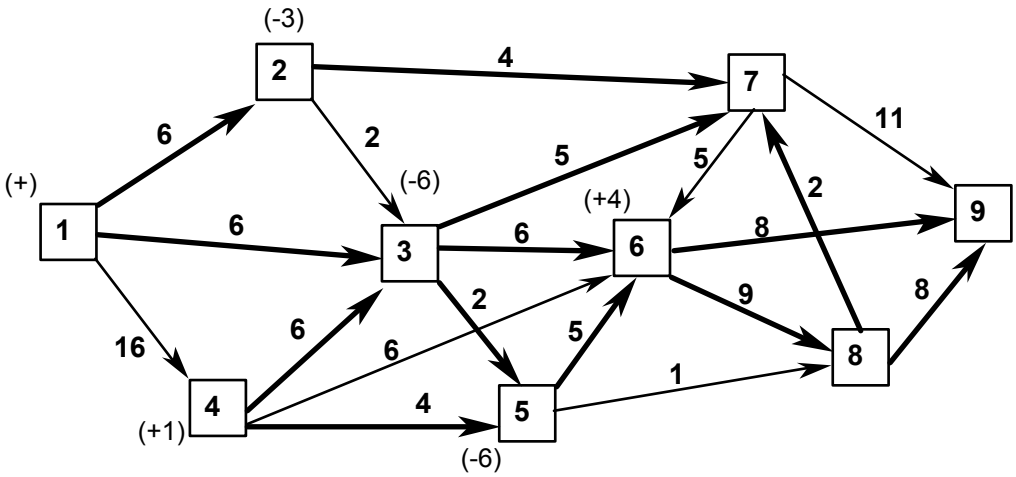
-aumentar 2 unidades em (1,4,6,8,9).



iii] Marcação dos nós, conseguindo-se marcar a saída da rede, 9 .



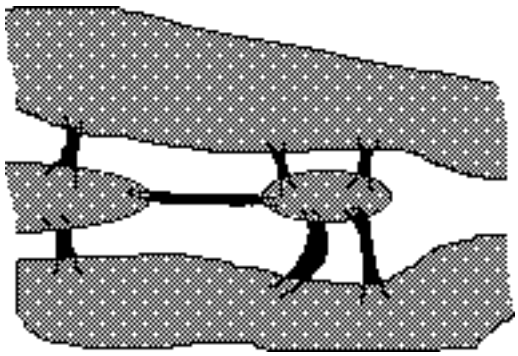
iv] Fluxo máximo [obtido por aumento do fluxo nos caminhos (1,4,6,7,9) e (1,4,5,6,7,9)].



4.4 Determinação de CIRCUITOS EULERIANOS

4.4.1 O problema das pontes de Königsberg

O rio Pregel banha a cidade de Königsberg, na Prússia Oriental, e rodeia a ilha de Kneiphof. A ligar os vários pontos das margens, havia 7 pontes dispostas como na figura:

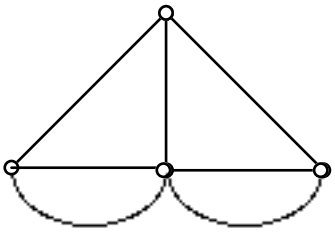


No seu passeio dominical, os habitantes da cidade procuravam voltar ao ponto de partida, passando uma só vez por todas as pontes.

Leonard Euler estudou o problema e demonstrou a sua impossibilidade (1736).

Definição de Circuito Euleriano:

Um Circuito Euleriano, é um caminho finito em que o nó inicial coincide com o nó final e que passa uma única vez por todos os arcos da rede.



Teorema de Euler

Um grafo admite um circuito euleriano se e só se for conexo e o número de vértices de grau impar for zero. (o grau de um vértice, corresponde ao número de arcos incidente no vértice)

4.4.2 O problema do carteiro chinês (chinese postman problem CPP)

Considerem-se redes de distribuição linear, como por exemplo:

- a recolha de lixo numa cidade;
- a distribuição domiciliária do correio;
- a inspecção periódica de redes de energia, de telefones, etc.

O problema consiste em definir circuitos que se aproximem do Circuito Euleriano ideal (repetindo arcos em número mínimo ou de modo a tornar mínimo o aumento de percurso).

4.5 Determinação de CIRCUITOS HAMILTONIANOS

4.5.1 Definição de Circuitos Hamiltonianos

Um circuito diz-se Hamiltoniano, se passar uma e uma só vez por todos os vértices de uma rede.

A designação provém de um passatempo imaginado pelo islandês Hamilton (1859), no qual se procurava encontrar um caminho que percorresse 20 cidades de todo o mundo, representadas pelos vértices de um dodecaedro de madeira, voltando ao ponto inicial. (grafo plano na figura)

4.5.2 O problema do Caixeiro Viajante (travelling salesman problem TSP)

Um caixeiro viajante sai de uma cidade, visita n outras cidades e volta àquela de onde partiu, sem repetir nenhuma das cidades visitadas.

De todos os circuitos possíveis de estabelecer, qual é o mais curto?

Trata-se da pesquisa do circuito hamiltoniano mais curto num grafo de (n+1) vértices -> número de soluções possíveis = n!

Algoritmos conducentes à solução óptima

Pouco eficientes;

Baseados num método de branch and bound como o de Little, 1963.

Algoritmos que levam a soluções quase óptimas

Muito eficientes;

Baseiam-se em regras heurísticas como a de Cicero-Ruggiero, 1972

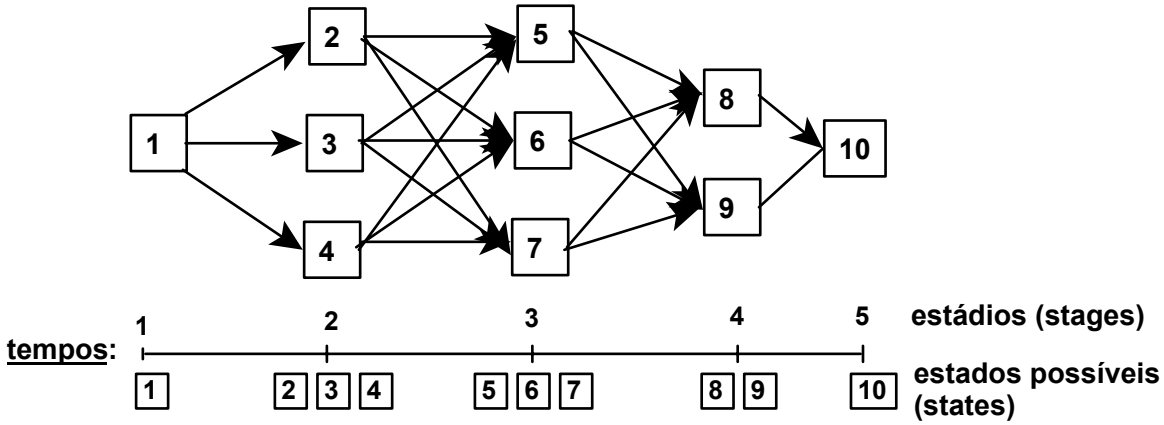
5. PROGRAMAÇÃO DINÂMICA

A utilização da programação dinâmica em sequências de decisões inter-relacionadas, permite determinar a combinação daquelas que maximiza a eficiência global.

Vamos agora descrever o **problema da diligência**:

Exemplo 1:

A diligência, para chegar ao seu destino, tinha que atravessar territórios de perigosos índios. A viagem (entre as cidades [1] e [10]), a ser feita em 4 etapas, podia ser feita por diferentes percursos (diferentes cidades de passagem). A segurança de cada estrada intermédia era razoavelmente bem medida pelo custo da apólice de seguro respectiva.



O custo da apólice de seguro para ir de [i] para [j] é c_{ij}:

	2	3	4
1	2	4	3

	5	6	7
2	7	4	6
3	3	2	4
4	4	1	5

	8	9
5	1	4
6	6	3
7	3	3

	10
8	3
9	4

Que percurso fazer, por forma a minimizar o custo total do seguro?

Variáveis de decisão:

x_n- destino imediato decidido no estádio (tempo) n (n=1,2,3,4)

Vamos supor que a diligência, no tempo **n** está no estado **s** e selecciona como destino imediato x_n.

Seja $f_n(s,x_n)$ o custo total da melhor política para os estádios restantes.

Seja x^*_n o valor que minimiza $f_n(s,x_n)$ e $f^*_n(s)$ o correspondente valor mínimo de $f_n(s,x_n)$, $[f^*_n(s) = f_n(s,x^*_n)]$.

O objectivo é encontrar $f^*_1(1)$ e a correspondente política.

A programação dinâmica consegue-o calculando sucessivamente $f^*_4(s)$, $f^*_3(s)$, $f^*_2(s)$ e $f^*_1(s)$.

Resolução do problema

n=4 (último estádio)

s	$f^*_4(s)$	x^*_4
8	3	10
9	4	10

n=3

	$f_3(s,x_3) = c_s x_3 + f^*_4(x_3)$			
$s \backslash x_3$	8	9	$f^*_3(s)$	x^*_3
5	4	8	4	8
6	9	7	7	9
7	6	7	6	8

n=2

	$f_2(s,x_2) = c_s x_2 + f^*_3(x_2)$				
$s \backslash x_2$	5	6	7	$f^*_2(s)$	x^*_2
2	11	1	12	11	5 ou 6
3	7	9	10	7	5
4	8	8	11	8	5 ou 6

n=1 (primeiro estádio)

	$f_1(s,x_1) = c_s x_1 + f^*_2(x_1)$				
$s \backslash x_1$	2	3	4	$f^*_1(s)$	x^*_1
1	13	11	11	11	3 ou 4

Solução óptima:

1ª decisão (n=1): ir para as cidades 3 ou 4 (estados 3 e 4)

Alternativa 1 :

$x^*_1=3$

para s=3 -> $x^*_2=5$ [$x^*_2(3)=5$]

para s=5 -> $x^*_3=8$

para s=8 -> $x^*_4=10$

Um percurso óptimo será então:

1 -> 3 ->5 ->8 ->10

Alternativa 2 : 1 -> 4 ->5 ->8 ->10

Alternativa 3 : 1 -> 4 ->6 ->9 ->10

Características dos problemas de programação dinâmica:

1. 1] O problema pode ser dividido em ESTÁDIOS (stages), sendo tomada uma DECISÃO em cada estádio;
2. 2] A cada estádio estão associados determinados ESTADOS (states);
3. 3] O efeito da decisão tomada em cada estádio é transformar o estado actual num estado associado com o estádio seguinte, possivelmente de acordo com uma distribuição de probabilidades;
4. 4] Dado o estado actual, a política óptima para os restantes estádios é independente da política adoptada nos estádios prévios; (PRINCÍPIO DE OPTIMALIDADE DE BELLMAN);
5. 5] O processo de resolução começa pela determinação da decisão óptima para cada estado do ÚLTIMO ESTÁDIO;
6. É possível definir uma RELAÇÃO RECURSIVA que identifica a política óptima para cada estado no estádio n, dada a política óptima para cada estado no estádio (n+1). Exemplo:

$$f^*_n(s) = \min_{x_n}\{c_s x_n + f^*_{n+1}(x_n)\}$$

7. Usando esta RELAÇÃO DE RECURSIVIDADE, a resolução do problema processa-se para trás (backward) estádio a estádio - em cada momento determina-se a decisão óptima para cada estado daquele estádio - até se encontrar a decisão óptima para o estádio inicial.

Exemplo 2:

(distribuir 5 equipas médicas por 5 países)

$$\max \sum_{i=1}^{3} p_i(x_i)$$

$$\text{sujeito a: } \sum_{i=1}^{3} x_i = 5$$

$x_i \_ 0$ e inteiros

x_i - número de equipas afectadas ao país i

País

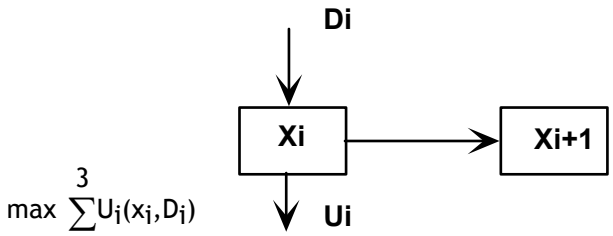
nº equipas médicas	1	2	3
0	0	0	0
1	45	20	50
2	75	45	70
3	90	75	80
4	105	110	100
5	120	150	130

$p_i(x_i)$: 1000 de anos de vida/homem adicionais

estados do sistema - nº de equipas disponíveis em cada estádio

solução óptima: $x_1=3$ $x_2=3$ $x_3=3$

Exemplo 3



$$\max \sum_{i=1}^{3} U_i(x_i, D_i)$$

A] Transformações de estado

$x_2=x_2(x_1,D_1)$

$x_1 \backslash D_1$	1	2	3	4
1	3	2	1	4
2	4	3	3	4
3	3	1	2	4
4	2	4	2	1

$x_3=x_3(x_2,D_2)$

$x_2 \backslash D_2$	1	2	3	4
1	-	2	5	1
2	3	4	3	-
3	4	5	4	-
4	3	4	2	3

B] Utilidade

$U_1=U_1(x_1,D_1)$

$x_1 \backslash D_1$	1	2	3	4
1	3	4	1	4
2	2	4	3	3
3	3	4	5	4
4	4	2	3	2

$U_2=U_2(x_2,D_2)$

$x_2 \backslash D_2$	1	2	3	4
1	-	1	5	4
2	5	4	2	-
3	2	3	3	-
4	3	5	4	2

$U_3=U_3(x_3,D_3)$

$x_2 \backslash D_2$	1	2	3	4

1	2	1	3	-
2	4	3	2	-
3	3	5	4	3
4	-	4	3	5
5	4	2	4	3