Universidade do Minho
Escola de Engenharia
Departamento de Produção e Sistemas

# Optimização de Sistemas em Rede

Filipe Pereira e Alvelos
*falvelos@dps.uminho.pt*
*www.dps.uminho.pt/pessoais/falvelos*

Versão 03 - Novembro de 2012

## Índice

# Introdução (1)

- Redes estão generalizadas na nossa sociedade: redes eléctricas, de comunicações, de transportes, de produção e distribuição, etc...
- As redes permitem o envio de entidade(s) (electricidade, água, gás, pessoas, mensagens, produtos,...) de um local para outro.
- A abstracção do conceito de rede permite tratar problemas em que não existe um transporte físico de entidades. Por exemplo, em gestão de projectos é frequente utilizar modelos de rede para representar relações temporais entre as diversas actividades que constituem o projecto. Em sistemas de produção é frequente utilizar modelos de rede para representar o processo produtivo.



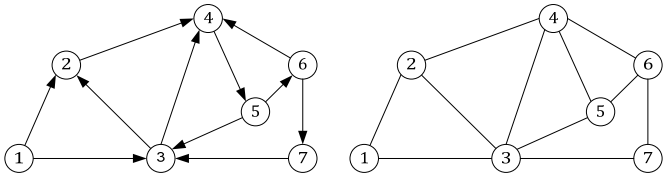


# Introdução (2)

- Uma rede é um grafo ao qual estão associados valores numéricos
  - distâncias, custos, capacidades, fluxos, quantidades disponíveis ou requeridas, ...
- Modelos de fluxos em rede permitem determinar como devem ser encaminhadas as entidades que se servem da rede para optimizar uma medida global de eficiência do sistema.
- Modelos de desenho de redes permitem determinar o conjunto de arcos que devem ser seleccionados para optimizar uma medida global de eficiência do sistema.

- Existem modelos em rede que combinam características de modelos de fluxos e de desenho de redes, assim como outras características.

# Conceitos básicos de redes (1)




Grafo orientado Grafo não-orientado

- Um grafo é um conjunto de nodos (nós ou vértices) e um conjunto de arcos (arestas ou ligações) que une dois nodos.
- Formalmente, $G=(N,A)$ em que $N$ é o conjunto de nodos e $A$ é o conjunto de arcos.
  - No exemplo de grafo orientado, $N=\{1,2,3,4,5,6,7\}$ e $A=\{(1,2),(1,3),(2,4),(3,2),(3,4),(4,5),(5,2),(5,6),(6,4),(6,7),(7,3)\}$
  - No exemplo de grafo não-orientado, $N=\{1,2,3,4,5,6,7\}$ e $A=\{\{1,2\},\{1,3\},\{2,4\},\{3,2\},\{3,4\},\{4,5\},\{5,2\},\{5,6\},\{6,4\},\{6,7\},\{7,3\}\}$
- Num grafo não-orientado, $(i,j)=(j,i)$.





# Conceitos básicos de redes (2)

- Um caminho é uma sequência de nodos distintos e dos arcos que os unem (cada origem de um arco é o destino do arco anterior).
  - No exemplo*, (1,2), (2,4), (4,5) é um caminho entre 1 e 5.
- Um percurso é uma estrutura similar ao caminho mas em que alguns arcos podem ser percorridos em sentido inverso.
  - No exemplo*, (6,7), (6,4), (2,4) é um percurso entre 6 e 2.
- Um circuito é uma sequência de nodos distintos (com excepção do primeiro e do último) e dos arcos que os unem em que o último arco está ligado ao primeiro.
  - No exemplo*, 3453 é um circuito.
- Um ciclo é uma estrutura similar ao circuito mas em que alguns arcos podem ser percorridos em sentido inverso.
  - No exemplo*, 34653 é um ciclo.

* É usual omitirem-se os nodos ou os arcos na representação de caminhos, percursos, circuitos e ciclos.

# Conceitos básicos de redes (3)

- Pressupostos
  - as redes consideradas são ligadas, isto é, existe um percurso entre qualquer par de nodos;
  - não existem arcos com origem e destino no mesmo nodo;
  - entre dois nodos existe no máximo um arco se o grafo é não orientado e no máximo dois arcos (um em cada orientação) se o grafo é não orientado;
  - todos os parâmetros associados à rede têm valores inteiros.





# Introdução (1)

- Modelos de fluxos em rede

  - Existe um determinado objectivo traduzido por uma função (*função objectivo*) que aos fluxos nos arcos faz corresponder um valor (por exemplo, um custo total)
  - Pretende-se determinar o valor do fluxo em cada arco de forma a que a função objectivo tome o menor (ou maior) valor possível

  - Existem nodos nos quais é injectado fluxo na rede
  - Existem nodos nos quais é extraído fluxo da rede
  - Existe conservação de fluxo em todos os nodos da rede

# Introdução (2)

- Modelos
  - Transportes
  - Afectação
  - Caminho mais curto
  - Fluxo de custo mínimo
  - Fluxo máximo
  - Extensões
- Representações
  - Gráfica (rede)
  - Algébrica (Programação Linear)
  - Folhas de cálculo
- Métodos de resolução
  - Algoritmos para Programação Linear (*solver* do *Excel*...)
  - Algoritmos específicos (exemplo: algoritmo para transportes, algoritmo húngaro)





# Transportes – Índice

- Exemplo
- Modelo de Programação Linear
- Conceitos básicos
- Um algoritmo
- Modelo no *Excel*
- Optimização com o *Solver*
- Exercício

# Transportes – Exemplo (1)

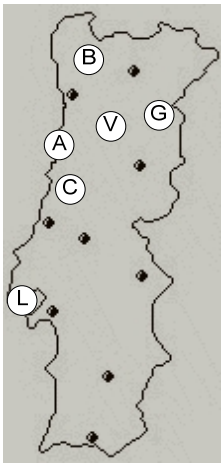
- Uma empresa tem fábricas em Aveiro, Braga e Coimbra que produzem 75, 125 e 150 unidades de um determinado produto por semana. A empresa entrega, todas as semanas, 50, 100 e 200 unidades em Viseu, Guarda e Lisboa, respectivamente. Assumindo que o custo de transporte de cada unidade é proporcional à distância entre as cidades (dada na tabela), qual o melhor plano de transporte semanal?

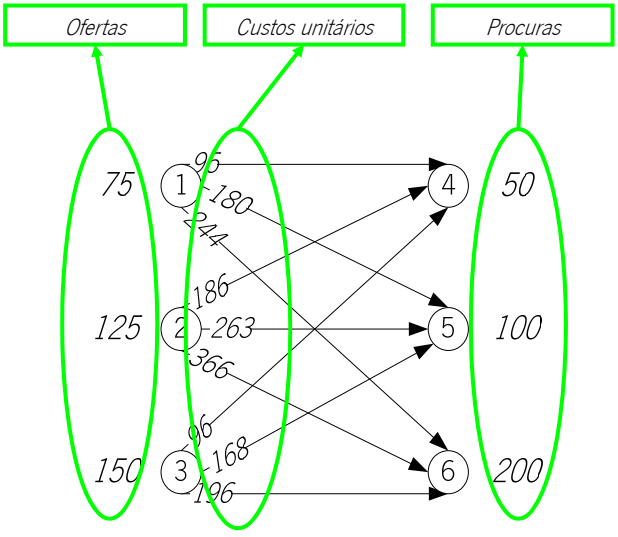
| Distância em *Km* | Viseu | Guarda | Lisboa |
|---|---|---|---|
| Aveiro | 95 | 180 | 244 |
| Braga | 186 | 263 | 366 |
| Coimbra | 96 | 168 | 196 |





# Transportes – Exemplo (2)



# Transportes – Exemplo (3)

- Pretende-se determinar as quantidades a transportar (fluxos) em cada arco de forma a que o custo seja o menor possível, tendo em conta que de cada origem não podem sair mais unidades do que as disponíveis e a cada destino têm de chegar as unidades nele requeridas.

| Quantidade a transportar? | | Destinos | | |
|---|---|---|---|---|
| | | Viseu | Guarda | Lisboa |
| Origens | Aveiro | ? | ? | ? |
| | Braga | ? | ? | ? |
| | Coimbra | ? | ? | ? |





# Transportes – Modelo de Programação Linear (1)

- Modelo de Programação Linear

$$Min\ z = 95x_{14} + 180x_{15} + 244x_{16} + 186x_{24} + 263x_{25} + 366x_{26} + 96x_{34} + 168x_{35} + 196x_{36}$$

Função objectivo

*sujeito a*:

$$x_{14} + x_{15} + x_{16} = 75$$
$$x_{24} + x_{25} + x_{26} = 125$$
$$x_{34} + x_{35} + x_{36} = 150$$
$$x_{14} + x_{24} + x_{34} = 50$$
$$x_{15} + x_{25} + x_{35} = 100$$
$$x_{16} + x_{26} + x_{36} = 200$$
$$x_{14}, x_{15}, \ldots, x_{36} \geq 0$$

Restrições

## Transportes – Modelo de Programação Linear (2)

- Parâmetros de um modelo de transportes
  - $n$ origens
  - $m$ destinos
  - $A$ conjunto de arcos (cada um entre uma origem $i$ e um destino $j$)
  - $a_i$ oferta da origem $i$, $i=1,..,n$
  - $b_j$ procura do destino $j$, $j=n+1,...,m+n$
  - $c_{ij}$ custo unitário de transporte entre $i$ e $j$, $\forall ij \in A$
  - $u_{ij}$ quantidade máxima que é possível transportar entre $i$ e $j$ (capacidade do arco $ij$), $\forall ij \in A$
- Variáveis de decisão
  - $x_{ij}$ quantidade transportada entre $i$ e $j$ (fluxo no arco $ij$), $\forall ij \in A$





## Transportes – Modelo de Programação Linear (3)

$$Min\ z = \sum_{ij \in A} c_{ij} x_{ij}$$

$$s.a:$$

$$\sum_{j:ij \in A} x_{ij} = a_i, i = 1,..., n$$

$$\sum_{i:ij \in A} x_{ij} = b_j, j = n+1,..., n+m$$

$$0 \le x_{ij} \le u_{ij}, \forall ij \in A$$

# Transportes – Conceitos básicos

- Uma *solução* corresponde a um conjunto de valores atribuídos aos fluxos nos arcos.
- Numa *solução admissível* a conservação de fluxo e as capacidades nos arcos (se presentes) são respeitadas.
- Uma *solução óptima* é uma solução admissível para a qual a função objectivo tem um valor não superior a qualquer outra solução admissível.
- Estes conceitos são comuns a todos os modelos de optimização (de que fazem parte os modelos de fluxos em rede).
- Os algoritmos existentes para transportes permitem obter soluções óptimas para problemas de enormes dimensões.





# Transportes – Um algoritmo (1)

- Um algoritmo para o problema de transportes que garante a obtenção de uma solução óptima
  - Um algoritmo é uma sequência não ambígua de instruções que é executada até que determinada condição se verifique
- Passo 1: obter uma solução admissível
- Passo 2: a solução actual pode ser melhorada?
  - Se não, parar porque a solução actual é óptima
  - Se sim, continuar
- Passo 3: actualizar solução e ir para o passo 1

# Transportes – Um algoritmo (2)

- **Passo 1:** obter uma solução admissível (por exemplo, enviando a maior quantidade possível pelos arcos com menor custo unitário)

| | | |
|---|---|---|
| **95** | 180 | 244 |
| 186 | 263 | 366 |
| 96 | 168 | 196 |

Menor custo unitário

| | Viseu | Guarda | Lisboa | | | |
|---|---|---|---|---|---|---|
| Aveiro | **50** | | | 50 | = | 75 |
| Braga | 0 | | | | = | 125 |
| Coimbra | 0 | | | | = | 150 |
| | 50 | | | | | |
| | = | = | = | | | |
| | 50 | 100 | 200 | | | |





# Transportes – Um algoritmo (3)

| | | |
|---|---|---|
| 95 | 180 | 244 |
| 186 | 263 | 366 |
| 96 | **168** | 196 |

| | Viseu | Guarda | Lisboa | | | |
|---|---|---|---|---|---|---|
| Aveiro | **50** | 0 | | 50 | = | 75 |
| Braga | 0 | 0 | | | = | 125 |
| Coimbra | 0 | **100** | | 100 | = | 150 |
| | 50 | 100 | | | | |
| | = | = | = | | | |
| | 50 | 100 | 200 | | | |

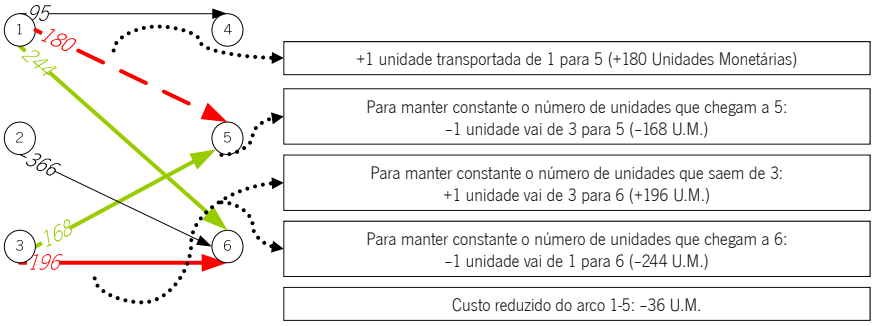
| | | |
|---|---|---|
| 95 | 180 | **244** |
| 186 | 263 | **366** |
| 96 | 168 | **196** |

| | Viseu | Guarda | Lisboa | | | |
|---|---|---|---|---|---|---|
| Aveiro | **50** | 0 | **25** | 75 | = | 75 |
| Braga | 0 | 0 | **125** | 125 | = | 125 |
| Coimbra | 0 | **100** | **50** | 100 | = | 150 |
| | 50 | 100 | 200 | | | |
| | = | = | = | | | |
| | 50 | 100 | 200 | | | |

Custo total desta solução: 83200

# Transportes – Um algoritmo (4)

- Passo 2: Para cada arco sem fluxo, determinar a variação do custo total se o arco passar a transportar uma unidade (esta variação designa-se por custo reduzido), mantendo a admissibilidade da solução
  - Exemplo para Aveiro – Guarda








# Transportes – Um algoritmo (5)

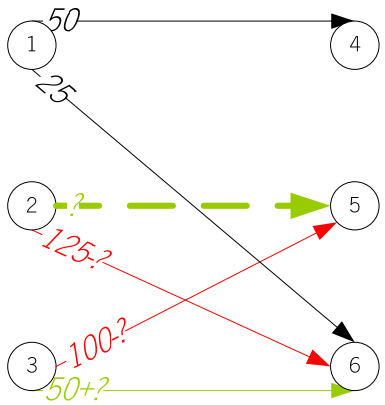
- Custos reduzidos para todos os arcos sem fluxo

| | Viseu | Guarda | Lisboa |
|---|---|---|---|
| Aveiro | | –36 | |
| Braga | –31 | –75 | |
| Coimbra | +49 | | |

- Note-se que, para cada arco sem fluxo, existe um e só um ciclo (no exemplo anterior o ciclo 1-5-3-6-1) que permite o cálculo do custo reduzido. Note-se ainda que, para a construção do ciclo, à excepção do arco para o qual se está a calcular o custo reduzido, apenas se consideram arcos com fluxo positivo.
- Existem arcos com custos reduzidos negativos, logo é possível melhorar (diminuir o custo) da solução actual. Se todos os custos reduzidos fossem maiores ou iguais a zero, a solução actual era óptima, logo parava-se.

# Transportes – Um algoritmo (6)

- **Passo 3:** Para o arco com menor custo reduzido determinar a maior quantidade que pode transportar, actualizar a solução e ir para o passo 2.
  - Arco com menor custo reduzido é o 2-5
  - Quanto maior a quantidade transportada menor o custo, mas não pode haver fluxos negativos
  - Na próxima solução a quantidade transportada no arco 2-5 é 100 (mínimo entre 100 e 125)








# Transportes – Um algoritmo (7)

- Nova iteração (uma iteração é constituída pelos passos 2 e 3) com solução actual (custo 83200 – 75.100 = 75700)

| | Viseu | Guarda | Lisboa |
|---|---|---|---|
| Aveiro | 50 | | 25 |
| Braga | | 100 | 25 |
| Coimbra | | | 150 |

- Custos reduzidos

| | Viseu | Guarda | Lisboa |
|---|---|---|---|
| Aveiro | | 39 | |
| Braga | –31 | | |
| Coimbra | 49 | 75 | |

- Existe um arco com custo reduzido negativo
- Na próxima solução a quantidade transportada no arco 2-4 é 25

# Transportes – Um algoritmo (8)

- Nova iteração com solução actual (custo 75500 – 31.25 =74925)

| | Viseu | Guarda | Lisboa |
|---|---|---|---|
| Aveiro | 25 | | 50 |
| Braga | 25 | 100 | |
| Coimbra | | | 150 |

- Custos reduzidos

| | Viseu | Guarda | Lisboa |
|---|---|---|---|
| Aveiro | | 8 | |
| Braga | | | 31 |
| Coimbra | 49 | 44 | |

- Não existem arcos com custos reduzidos negativos, logo a solução actual é uma solução óptima.





# Transportes – Modelo no *Excel*

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | | Viseu | Guarda | Lisboa | | | | |
| 2 | Aveiro | | | | | 0 | <= | 75 |
| 3 | Braga | | | | | 0 | <= | 125 |
| 4 | Coimbra | | | | | 0 | <= | 150 |
| 5 | | | | | | | | |
| 6 | | 0 | 0 | 0 | | | | |
| 7 | | = | = | = | | | | |
| 8 | | 50 | 100 | 200 | | | | |
| 9 | | | | | | | | |
| 10 | | Viseu | Guarda | Lisboa | | | | |
| 11 | Aveiro | 95 | 180 | 244 | | | | |
| 12 | Braga | 186 | 263 | 366 | | | | |
| 13 | Coimbra | 96 | 168 | 196 | | | | |
| 14 | | | | | | | | |
| 15 | **Custo total** | 0 | | | | | | |
| 16 | | | | | | | | |

*=SUM(B2:D2)*
*=SUM(B3:D3)*
*=SUM(B4:D4)*

*=SUM(B2:B4)*
*=SUM(C2:C4)*
*=SUM(D2:D4)*

*=SUMPRODUCT(B2:D4,B11:D13)*

# Transportes – *Solver* do *Excel* (1)

- *Solver* do *Excel*
  - *Excel* 2003
    - *Tools -> Solver*
    - [ Se não está disponível fazer Tools -> Add-Ins e seleccionar *Solver Add-In*. ]
  - *Excel* 2007
    - *Data -> Analysis -> Solver*
    - [ Se não está disponível fazer *Office button -> Excel Options -> Manage Add-Ins* e seleccionar *Solver Add-In*. ]
  - Implementa o algoritmo simplex para modelos de programação linear (mais gerais do que o modelo de transportes)





# Transportes – *Solver* do *Excel* (2)

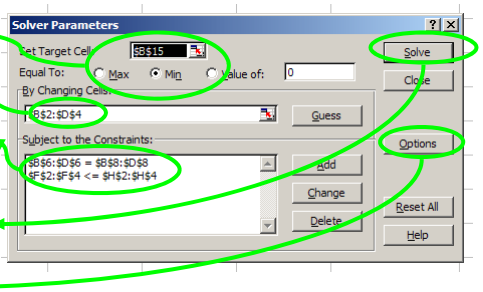
Minimizar o custo total

Quantidades a transportar

O que sai de cada origem é igual à oferta
O que chega a cada destino é igual à procura

Resolver modelo
(depois de consultar/alterar as opções)

É essencial especificar as opções!



# Transportes – *Solver* do *Excel* (3)

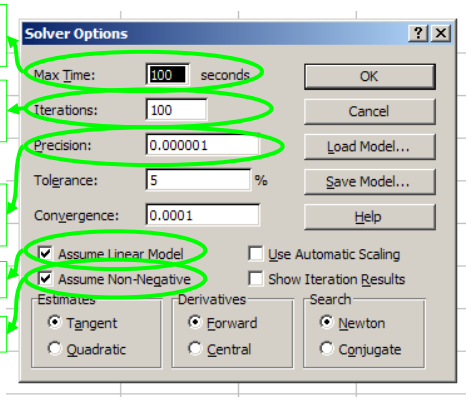
Tempo máximo, se atingido é apresentada a melhor solução obtida até ao momento.

Número de iterações máximo, se atingido é apresentada a melhor solução obtida até ao momento.

Tolerâncias para satisfação de restrições e integralidade de variáveis.

Todas as relações são lineares (essencial!!!)

Não é possível transportar quantidades negativas.



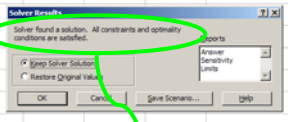





# Transportes – *Solver* do *Excel* (4)

Solução óptima! (Qualquer solução admissível – que respeita as restrições – tem um custo não inferior)

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | Viseu | Guarda | Lisboa | | | | | |
| 2 | Aveiro | 25 | 0 | 50 | | 75 | <= | 75 | |
| 3 | Braga | 25 | 100 | 0 | | 125 | <= | 125 | |
| 4 | Coimbra | 0 | 0 | 150 | | 150 | <= | 150 | |
| 5 | | | | | | | | | |
| 6 | | 50 | 100 | 200 | | | | | |
| 7 | | = | = | = | | | | | |
| 8 | | 50 | 100 | 200 | | | | | |
| 9 | | | | | | | | | |
| 10 | | Viseu | Guarda | Lisboa | | | | | |
| 11 | Aveiro | 95 | 180 | 244 | | | | | |
| 12 | Braga | 186 | 263 | 366 | | | | | |
| 13 | Coimbra | 96 | 168 | 196 | | | | | |
| 14 | | | | | | | | | |
| 15 | **Custo total** | 74925 | | | | | | | |
| 16 | | | | | | | | | |

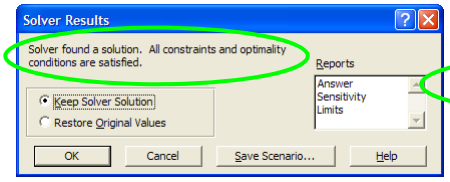



Se não aparecer esta mensagem, a solução não tem significado!

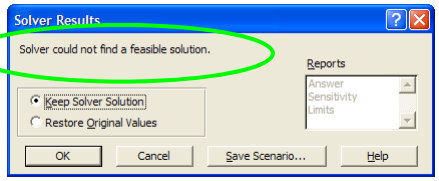
Menor custo possível

# Transportes – *Solver* do *Excel* (5)

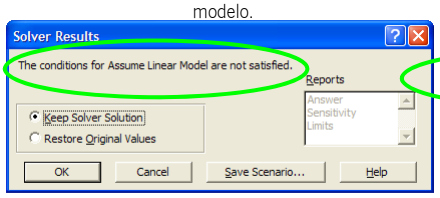
Solução óptima encontrada
(dentro das tolerâncias definidas!)



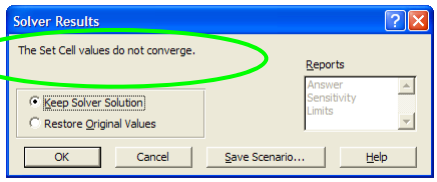

Problema impossível. Verificar modelo e/ou problema. Faz sentido que não haja nenhuma solução / alternativa admissível?




Modelo não é linear. Pode incluir expressões algébricas não lineares, ou funções como IF, ABS, MAX, MIN. Também pode ser por uma introdução incorrecta de dados ou funções lineares. Verificar modelo.




Problema ilimitado. Verificar modelo e/ou problema. Faz sentido que haja uma solução tão boa quanto desejarmos?








# Transportes – Exercício (1)

- Pretende-se efectuar o planeamento da produção de um determinado artigo para as próximas cinco semanas. As quantidades encomendadas (procura) do artigo em cada uma das semanas são conhecidas e são entregues aos clientes no final de cada semana. Consideram-se custos unitários de produção em cada semana e custos unitários de armazenamento também em cada semana (em que se incorre quando os artigos que existem no final de uma semana não são entregues aos clientes mas ficam armazenados até ao final da semana seguinte).
  Utilize um modelo de transportes e o *Solver* do *Excel* para determinar as quantidades que devem ser produzidas em cada semana de forma a minimizar o custo total. Todos os dados são apresentados na Tabela seguinte.

# Transportes – Exercício (2)

| Semana | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Procura | 10 | 12 | 17 | 13 | 15 |
| Custo unitário de produção | 4 | 5 | 3 | 6 | 2 |
| Custo unitário de armazenamento | - | 2 | 1 | 2 | 3 |

- Exemplo de uma solução admissível
  - Produzir em 1 para satisfazer as procuras de 1, 2 e 3
    - Custo de produção = 4 . (10 + 12 + 17)
    - Custo de armazenamento = 2 . (12 + 17) + 1 . 17
  - Produzir em 4 para satisfazer as procuras de 4 e 5
    - Custo de produção = 6 . (13 + 15)
    - Custo de armazenamento = 3 . 15
  - Custo total = 444





# Transportes – Exercício (3)

- Solução óptima
  - Produzir 10 unidades na semana 1 para a semana 1
  - Produzir 12 unidades na semana 2 para a semana 2
  - Produzir 30 unidades na semana 3
    - 17 para a semana 3
    - 13 para a semana 4
  - Produzir 15 unidades na semana 5 para a semana 5
  - Custo: 246

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 10 | 0 | 0 | 0 | 0 |
| 2 | 0 | 12 | 0 | 0 | 0 |
| 3 | 0 | 0 | 17 | 13 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 15 |

# Transportes – Exercício (4)

- Obtenha uma solução óptima com o *solver* do *Excel* para o caso em que é possível entregar encomendas com atraso, assumindo que, em cada semana, o custo unitário de atraso é de 1 unidade monetária.
  - [Solução óptima tem custo total 208.]
- Obtenha uma solução óptima com o *solver* do *Excel* para o caso em que, em cada período, não podem ser produzidas mais de 14 unidades.
  - [Solução óptima tem custo total 276.]
- Apresente os modelos de programação linear para cada uma das três variantes abordadas.





# Afectação - Índice

- Exemplo
- *Solver* do *Excel*
- Modelo de Programação Linear
- Representação em rede
- Algoritmo húngaro
- Exercício

# Afectação – Exemplo

- Num determinado serviço de um hospital pretende-se fazer a escala de 10 enfermeiros para 10 turnos críticos. Cada enfermeiro deve trabalhar exactamente em um turno e em cada turno deve estar presente exactamente um enfermeiro.
  Na Tabela são apresentadas as preferências de cada enfermeiro em relação a cada turno numa escala de 1 a 5 em que 1 corresponde à preferência máxima.
  Determine qual o turno que cada enfermeiro deve efectuar.

Turno

| Enfermeiro | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 5 | 3 | 5 | 1 | 1 | 2 |
| 3 | 2 | 2 | 7 | 9 | 9 | 6 | 2 | 3 | 1 |
| 2 | 10 | 3 | 1 | 3 | 2 | 7 | 4 | 2 | 3 |
| 8 | 4 | 4 | 4 | 6 | 7 | 1 | 5 | 4 | 5 |
| 9 | 5 | 5 | 2 | 4 | 6 | 2 | 9 | 5 | 4 |
| 4 | 3 | 6 | 5 | 7 | 5 | 3 | 10 | 6 | 8 |
| 5 | 6 | 7 | 8 | 10 | 4 | 8 | 6 | 10 | 7 |
| 7 | 7 | 8 | 9 | 1 | 10 | 9 | 8 | 9 | 6 |
| 6 | 8 | 9 | 10 | 2 | 8 | 10 | 7 | 7 | 10 |
| 10 | 9 | 10 | 6 | 8 | 1 | 4 | 3 | 8 | 9 |

Enfermeiro





# Afectação – *Solver* do *Excel*

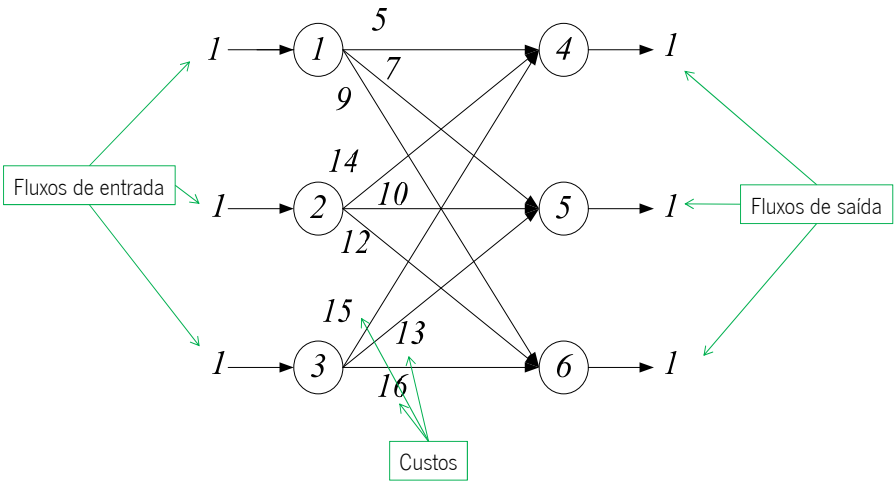
- Número de afectações possíveis: 10! = 3628800
- Se fossem 30 enfermeiros e 30 turnos haveria, aproximadamente, $2.65 \times 10^{32}$ alternativas. Testando um bilião de alternativas por segundo (um computador muito bom!), o tempo total seria mais de oito milénios!
- Solução representada à direita foi obtida através de Programação Linear (*Solver* do *Excel*)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 = 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 = 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 = 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 = 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 = 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 = 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 = 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 = 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 = 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 = 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| = | = | = | = | = | = | = | = | = | = | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

Soma das preferências dos pares escolhidos 24

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 5 | 3 | 5 | 1 | 1 | 2 |
| 3 | 2 | 2 | 7 | 9 | 9 | 6 | 2 | 3 | 1 |
| 2 | 10 | 3 | 1 | 3 | 2 | 7 | 4 | 2 | 3 |
| 8 | 4 | 4 | 4 | 6 | 7 | 1 | 5 | 4 | 5 |
| 9 | 5 | 5 | 2 | 4 | 6 | 2 | 9 | 5 | 4 |
| 4 | 3 | 6 | 5 | 7 | 5 | 3 | 10 | 6 | 8 |
| 5 | 6 | 7 | 8 | 10 | 4 | 8 | 6 | 10 | 7 |
| 7 | 7 | 8 | 9 | 1 | 10 | 9 | 8 | 9 | 6 |
| 6 | 8 | 9 | 10 | 2 | 8 | 10 | 7 | 7 | 10 |
| 10 | 9 | 10 | 6 | 8 | 1 | 4 | 3 | 8 | 9 |

# Afectação – Representação em Rede








# Afectação – Modelo de Programação Linear (1)

- Parâmetros de um modelo de afectação
  - $n$ agentes
  - $n$ tarefas
  - $A$ conjunto de arcos (cada um entre um agente $i$ e uma tarefa $j$)
  - $c_{ij}$ custo unitário de associar o agente $i$ à tarefa $j$, $\forall ij \in A$
- Variáveis de decisão
  - $x_{ij}=1$ se agente $i$ realiza a tarefa $j$; $x_{ij}=0$, caso contrário, $\forall ij \in A$
  - As condições $x_{ij} \in \{0,1\}$ podem ser substituídas por $0 \leq x_{ij} \leq 1$, já que mesmo efectuando essa substituição existe sempre uma solução óptima em que, para todos os arcos $ij$, $x_{ij}=0$ ou $x_{ij}=1$

## Afectação – Modelo de Programação Linear (2)

$$Min\ \mathrm{z} = \sum_{ij \in A} c_{ij} x_{ij}$$

$$s.a:$$

$$\sum_{j:ij \in A} x_{ij} = 1, i = 1,\ldots,n$$

$$\sum_{i:ij \in A} x_{ij} = 1, j = n+1,\ldots,2n$$

$$x_{ij} \geq 0, \forall ij \in A$$





## Afectação – Algoritmo húngaro (1)

- Dada a sua estrutura, um modelo de afectação pode ser representado numa tabela em que cada célula corresponde a uma variável de decisão e onde é representado o custo da afectação a ela associada.

| | | |
|---|---|---|
| *5* | *7* | *9* |
| *14* | *10* | *12* |
| *15* | *13* | *16* |

# Afectação – Algoritmo húngaro (2)

- A solução óptima de um problema de afectação não é alterada se for adicionada uma constante a uma qualquer linha ou a uma qualquer coluna da matriz de custos do problema.
- Para obter o quadro ao lado, subtraiu-se 5 à primeira linha, 10 à segunda, 13 à terceira e, para a matriz resultante, subtraiu-se 2 à terceira coluna.

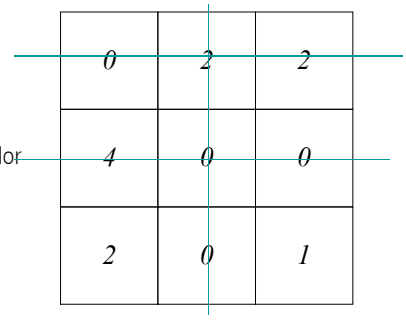
| | | |
|---|---|---|
| *0* | *2* | *2* |
| *4* | *0* | *0* |
| *2* | *0* | *1* |





# Afectação – Algoritmo húngaro (3)

- Conseguimos obter uma solução com valor zero? Responder a esta pergunta corresponde a:
  - Com o menor número de traços passar por cima de todos os zeros.
  - Se o número de traços é igual a $n$, conseguimos obter uma solução com valor zero, logo uma afectação óptima.



# Afectação – Algoritmo húngaro (4)

- Para determinar a solução basta seleccionar um conjunto de zeros que forme uma afectação admissível.
- No exemplo,

| | | |
|---|---|---|
| 0 | 2 | 2 |
| 4 | 0 | 0 |
| 2 | 0 | 1 |

- Custo total = 5+12+13=30.





# Afectação – Algoritmo húngaro (5)

- Algoritmo Húngaro
  - Inicialização:
    - Subtrair o menor elemento de cada linha a todos os elementos dessa linha;
    - Subtrair o menor elemento de cada coluna a todos os elementos dessa coluna;
  - Iteração:
    - Com o menor número de traços passar cima de todos os zeros.
    - Se o número de traços é igual à dimensão do problema (número de linhas ou de colunas), parar: existe uma afectação óptima que se pode obter através da selecção de elementos com valor zero. Se não, continuar.
    - Dos elementos não traçados, escolher o menor.
    - Subtrair o seu valor a todos os não traçados.
    - Somar o seu valor àqueles que estão traçados duas vezes.

# Afectação – Algoritmo húngaro (6)

- Exemplo

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





# Afectação – Algoritmo húngaro (7)

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

| | | | |
|---|---|---|---|
| *0* | *8* | *7* | *5* |
| *11* | *0* | *10* | *4* |
| *2* | *3* | *5* | *0* |
| *0* | *11* | *9* | *5* |

- Subtraindo o menor elemento de cada coluna a todos os elementos dessa coluna:

| | | | |
|---|---|---|---|
| *0* | *8* | *2* | *5* |
| *11* | *0* | *5* | *4* |
| *2* | *3* | *0* | *0* |
| *0* | *11* | *4* | *5* |

# Afectação – Algoritmo húngaro (8)

- Com o menor número de traços passar por cima de todos os zeros. Dimensão do problema (4) é maior do que o número de traços (3), logo o quadro não contém uma afectação óptima.

| | | | |
|---|---|---|---|
| 0 | 8 | 2 | 5 |
| 11 | 0 | 5 | 4 |
| 2 | 3 | 0 | 0 |
| 0 | 11 | 4 | 5 |

- Menor valor dos não traçados é 2. Subtraindo 2 a todos os não traçados e somando 2 aos traçados duas vezes:

| | | | |
|---|---|---|---|
| 0 | 6 | 0 | 3 |
| 13 | 0 | 5 | 4 |
| 4 | 3 | 0 | 0 |
| 0 | 9 | 2 | 3 |





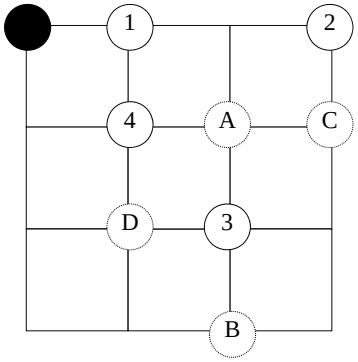
# Afectação – Algoritmo húngaro (9)

- Número de traços é igual a 4, logo existe uma afectação óptima.
- Afectação óptima: A-III, B-II, C-IV, D-I. Custo total = 9+4+11+4=28.

| | | | |
|---|---|---|---|
| 0 | 6 | 0 | 3 |
| 13 | 0 | 5 | 4 |
| 4 | 3 | 0 | 0 |
| 0 | 9 | 2 | 3 |

# Afectação – Exercício (1)

- Num determinado armazém, pretende-se recolher quatro objectos e guardar outros quatro objectos. No esquema são indicadas as posições dos quatros objectos que se pretende recolher (1, 2, 3 e 4) e as posições onde se pretende guardar os outros quatro objectos (A, B, C e D).
  O círculo a negro representa a posição onde se encontram inicialmente o veículo responsável pelo transporte e os objectos A, B, C e D. Os objectos 1, 2, 3 e 4 têm de ser recolhidos para esse mesmo local.








# Afectação – Exercício (2)

- A grelha do esquema representa percursos que o veículo pode seguir. Considere que a unidade de distância é o lado do quadrado da grelha. O veículo apenas pode carregar um objecto de cada vez. Por exemplo, uma viagem poderá consistir em sair do local inicial com o objecto A, guardá-lo, recolher o objecto 4 e voltar à posição inicial. A distância percorrida é 6.
  Como devem ser guardados e recolhidos os objectos de forma a que o veículo percorra a menor distância possível? Mostre que a solução que obteve é óptima através do algoritmo húngaro.

# Caminho mais curto (CMC) – Índice

- Exemplo
- *Solver* do *Excel*
- Modelo de Programação Linear

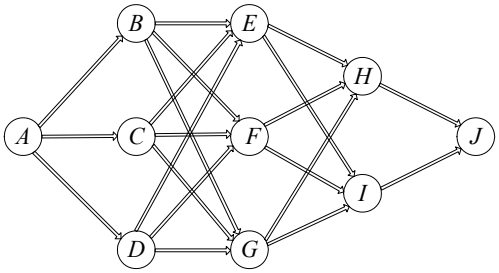




# CMC – Exemplo (1)

- No século XIX, um caçador de fortunas do Missouri decidiu ir para oeste, juntando-se à corrida do ouro na Califórnia. Para tal teve de executar quatro viagens diárias pelos territórios intermédios, correndo sérios riscos de segurança. De forma a minimizar esses riscos, o caçador de fortunas pensou na seguinte forma de os estimar: a companhia oferecia seguros de vida aos passageiros calculados com base no risco da viagem. Assim, quanto mais baixo o valor da apólice, mais segura a viagem. Os valores das apólices para as possíveis viagens são dados em seguida. Indique qual o caminho que o caçador de fortunas seguiu.

# CMC – Exemplo (2)




| | B | C | D |
|---|---|---|---|
| A | 2 | 4 | 3 |

| | E | F | G |
|---|---|---|---|
| B | 7 | 4 | 6 |
| C | 3 | 2 | 4 |
| D | 4 | 1 | 5 |

| | H | I |
|---|---|---|
| E | 1 | 4 |
| F | 6 | 3 |
| G | 3 | 3 |

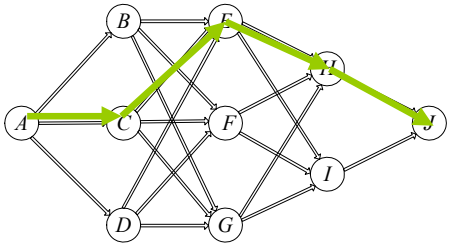
| | J |
|---|---|
| H | 3 |
| I | 4 |





# CMC – *Solver* do *Excel*

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | AB | AC | AD | BE | BF | BG | CE | CF | CG | DE | DF | DG | EH | EI | FH | FI | GH | GI | HJ | IJ | |
| A | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | 1 = 1 |
| B | -1 | | | 1 | 1 | 1 | | | | | | | | | | | | | | | 0 = 0 |
| C | | -1 | | | | | 1 | 1 | 1 | | | | | | | | | | | | 0 = 0 |
| D | | | -1 | | | | | | | 1 | 1 | 1 | | | | | | | | | 0 = 0 |
| E | | | | -1 | | | -1 | | | -1 | | | 1 | 1 | | | | | | | 0 = 0 |
| F | | | | | -1 | | | -1 | | | -1 | | | | 1 | 1 | | | | | 0 = 0 |
| G | | | | | | -1 | | | -1 | | | -1 | | | | | 1 | 1 | | | 0 = 0 |
| H | | | | | | | | | | | | | -1 | | -1 | | -1 | | 1 | | 0 = 0 |
| I | | | | | | | | | | | | | | -1 | | -1 | | -1 | | 1 | 0 = 0 |
| J | | | | | | | | | | | | | | | | | | | -1 | -1 | -1 = -1 |
| | 2 | 4 | 3 | 7 | 4 | 6 | 3 | 2 | 4 | 4 | 1 | 5 | 1 | 4 | 6 | 3 | 3 | 3 | 3 | 4 | 11 |



# CMC – Modelo de Programação Linear (1)

- Parâmetros de um modelo de caminho mais curto
  - $A$ conjunto de arcos
  - $c_{ij}$ custo unitário do arco de $i$ para $j$, $\forall ij \in A$
  - $o$ nodo origem
  - $d$ nodo destino
- Variáveis de decisão
  - $x_{ij}=1$ se o arco $ij$ faz parte do caminho mais curto; $x_{ij}=0$, caso contrário, $\forall ij \in A$
  - As condições $x_{ij} \in \{0,1\}$ podem ser substituídas por $0 \leq x_{ij} \leq 1$, já que mesmo efectuando essa substituição existe sempre uma solução óptima em que, para todos os arcos $ij$, $x_{ij}=0$ ou $x_{ij}=1$





# CMC – Modelo de Programação Linear (2)

$$Min\ z = \sum_{ij \in A} c_{ij} x_{ij}$$

$$s.a:$$

$$\sum_{j:ij \in A} x_{ij} - \sum_{i:ji \in A} x_{ji} = \begin{cases} 1\ para\ i = o \\ 0\ para\ i \neq 0, i \neq d \\ -1\ para\ i = d \end{cases}$$

$$x_{ij} \geq 0, \forall ij \in A$$

# Fluxo de custo mínimo (FCM) – Índice

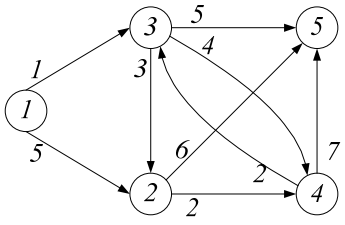
- Exemplo 1
- Modelo de Programação Linear
- Modelo no *Excel*
- Optimização com o *Solver*
- Exercício 1
- Exemplo 2
- Exercício 2





# Fluxo de Custo Mínimo (FCM) – Exemplo 1

- Uma determinada empresa de distribuição tem um produto armazenado em duas localizações distintas. Na localização 1 tem 50 unidades e na localização 4 tem 10 unidades. Dois clientes, localizados em 2 e 5, pretendem adquirir 40 e 20 unidades do produto em causa.
  Na figura seguinte representam-se, numa rede, todas as localizações relevantes deste problema, bem como o custo associado a transportar uma unidade do produto entre aquelas em que é possível fazê-lo.
  Qual a forma mais económica de transportar as 60 unidades com o menor custo possível, sabendo que não é possível transportar mais de 40 unidades entre cada par de localizações?



# FCM – Modelo de Programação Linear (1)

$$Min\ z = 5x_{12} + x_{13} + 2x_{24} + 6x_{25} + 3x_{32} + 4x_{34} + 5x_{35} + 2x_{43} + 7x_{45}$$

*sujeito a :*

$$x_{12} + x_{13} = 50$$
$$-x_{12} - x_{32} + x_{24} + x_{25} = -40$$
$$-x_{13} - x_{43} + x_{32} + x_{34} + x_{35} = 0$$
$$-x_{24} - x_{34} + x_{43} + x_{45} = 10$$
$$-x_{25} - x_{35} - x_{45} = -20$$
$$0 \le x_{ij} \le 40, \forall ij \in A$$

Função objectivo

Restrições

- Notar que
  - Cada restrição está associada (à conservação de fluxo de) um nodo
  - Cada variável aparece sempre em duas restrições (com coeficiente +1 numa e –1 na outra)





# FCM – Modelo de Programação Linear (2)

- Parâmetros de um modelo de fluxo de custo mínimo
  - $A$ e $N$ conjunto de arcos e conjunto de nodos, respectivamente
  - $u_{ij}$ capacidade do arco $ij, \forall ij \in A$
  - $c_{ij}$ custo unitário do arco arco $ij, \forall ij \in A$
  - $b_i$ oferta/procura do nodo $i$, $\forall i \in N$
    - Se $b_i > 0$ então o nodo $i$ é uma origem
    - Se $b_i < 0$ então o nodo $i$ é um destino
    - Se $b_i = 0$ então o nodo $i$ não é origem nem destino, designando-se por nodo de transbordo
- Variáveis de decisão
  - $x_{ij}$ fluxo no arco $ij, \forall ij \in A$

# FCM – Modelo de Programação Linear (3)

$$Min\ z = \sum_{ij \in A} c_{ij} x_{ij}$$

$$s.a:$$

$$\sum_{j:ij \in A} x_{ij} - \sum_{j:ji \in A} x_{ji} = b_i, \forall i \in N$$

$$0 \le x_{ij} \le u_{ij}, \forall ij \in A$$



# FCM – Modelo no *Excel* (1)

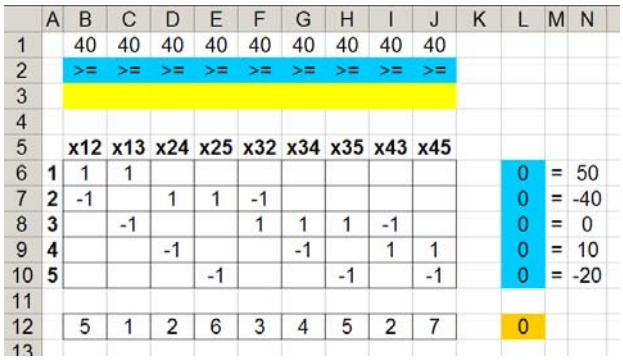
| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | | | | |
| 2 | | >= | >= | >= | >= | >= | >= | >= | >= | >= | | | | |
| 3 | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | |
| 5 | | x12 | x13 | x24 | x25 | x32 | x34 | x35 | x43 | x45 | | | | |
| 6 | 1 | 1 | 1 | | | | | | | | | 0 | = | 50 |
| 7 | 2 | -1 | | 1 | 1 | -1 | | | | | | 0 | = | -40 |
| 8 | 3 | | -1 | | | 1 | 1 | 1 | -1 | | | 0 | = | 0 |
| 9 | 4 | | | -1 | | | -1 | | 1 | 1 | | 0 | = | 10 |
| 10 | 5 | | | | -1 | | | -1 | | -1 | | 0 | = | -20 |
| 11 | | | | | | | | | | | | | | |
| 12 | | 5 | 1 | 2 | 6 | 3 | 4 | 5 | 2 | 7 | | 0 | | |
| 13 | | | | | | | | | | | | | | |

Variáveis de decisão

Restrições

Valor da função objectivo ($z$)

# FCM – Modelo no *Excel* (2)



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | | | | |
| 2 | | >= | >= | >= | >= | >= | >= | >= | >= | >= | | | | |
| 3 | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | |
| 5 | | x12 | x13 | x24 | x25 | x32 | x34 | x35 | x43 | x45 | | | | |
| 6 | 1 | 1 | 1 | | | | | | | | | 0 | = | 50 |
| 7 | 2 | -1 | | 1 | 1 | -1 | | | | | | 0 | = | -40 |
| 8 | 3 | | -1 | | | 1 | 1 | 1 | -1 | | | 0 | = | 0 |
| 9 | 4 | | | -1 | | | -1 | | 1 | 1 | | 0 | = | 10 |
| 10 | 5 | | | | -1 | | | -1 | | -1 | | 0 | = | -20 |
| 11 | | | | | | | | | | | | | | |
| 12 | | 5 | 1 | 2 | 6 | 3 | 4 | 5 | 2 | 7 | | 0 | | |
| 13 | | | | | | | | | | | | | | |

- Exemplos
  - Na célula L8: "=SUMPRODUCT($B$3:$J$3,B8:J8)"
  - Na célula L12: "=SUMPRODUCT($B$3:$J$3,B12:J12)"

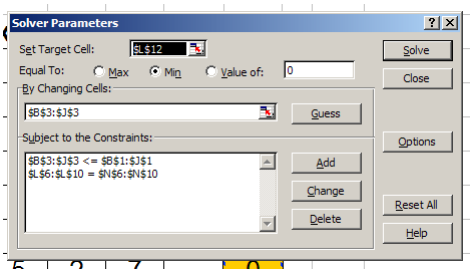


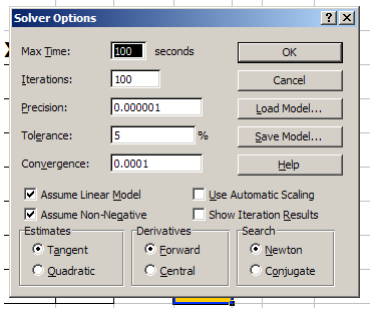


# FCM – Modelo de Programação Linear (3)

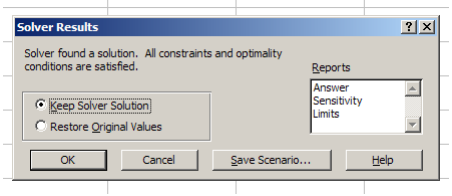
- Estrutura do modelo de Programação Linear
  - A cada arco está associada uma variável (coluna)
    - Fluxo no arco
  - A cada nodo está associado uma restrição (linha)
    - Conservação de fluxo: fluxo que entra é igual ao fluxo que sai
  - Cada variável tem coeficiente diferente de zero em duas restrições
    - Na restrição do nodo de origem do arco tem coeficiente 1
    - Na restrição do nodo de destino do arco tem coeficiente –1
  - Termos independentes das restrições correspondem a fluxos que entram na rede (sinal +) ou saem da rede (sinal –)

# FCM – Optimização com o *Solver* (1)




- Uma solução óptima foi obtida








# FCM – Optimização com o *Solver* (2)

- Solução óptima
  - $x_{12} = 10, x_{13} = 40, x_{32} = 30, x_{35} = 10, x_{45} = 10, x_{24} = x_{25} = x_{34} = x_{43} = 0$

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | | | | |
| 2 | | >= | >= | >= | >= | >= | >= | >= | >= | >= | | | | |
| 3 | | 10 | 40 | 0 | 0 | 30 | 0 | 10 | 0 | 10 | | | | |
| 4 | | | | | | | | | | | | | | |
| 5 | | x12 | x13 | x24 | x25 | x32 | x34 | x35 | x43 | x45 | | | | |
| 6 | 1 | 1 | 1 | | | | | | | | | 50 | = | 50 |
| 7 | 2 | -1 | | 1 | 1 | -1 | | | | | | -40 | = | -40 |
| 8 | 3 | | -1 | | | 1 | 1 | 1 | -1 | | | 0 | = | 0 |
| 9 | 4 | | | -1 | | | -1 | | 1 | 1 | | 10 | = | 10 |
| 10 | 5 | | | | -1 | | | -1 | | -1 | | -20 | = | -20 |
| 11 | | | | | | | | | | | | | | |
| 12 | | 5 | 1 | 2 | 6 | 3 | 4 | 5 | 2 | 7 | | 300 | | |

# FCM – Exercício 1

- Apresente um modelo de fluxo de custo mínimo para o problema descrito nos slides 32 e 33.

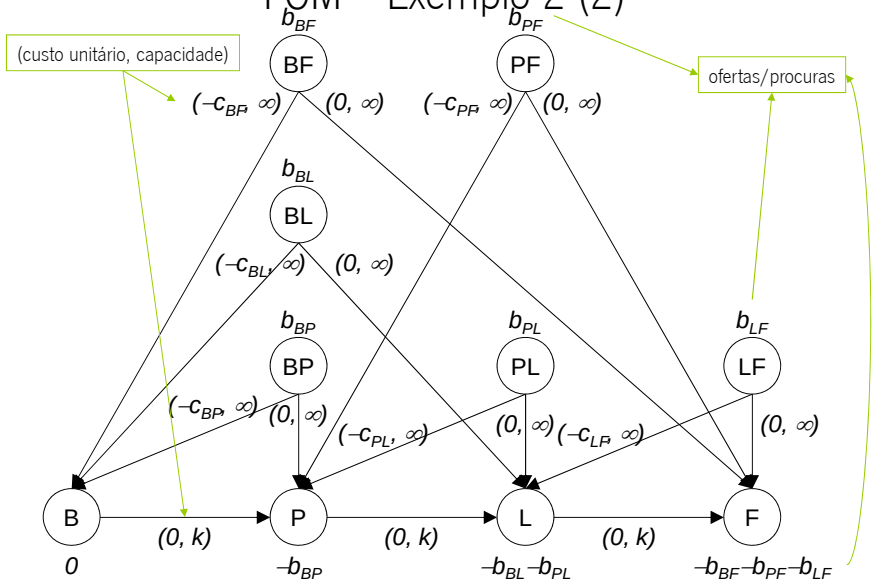




# FCM – Exemplo 2 (1)

- Uma determinada companhia aérea pondera a criação de um voo diário que tenha início em Bragança, pare no Porto, depois em Lisboa e termine em Faro. Em cada cidade podem entrar ou sair passageiros do avião. Este tem uma capacidade de $k$ lugares.
  Represente o número de passageiros que pretendem ir da cidade $i$ para a cidade $j$ por $b_{ij}$ e o preço do bilhete correspondente por $c_{ij}$.
  A companhia aérea pretende determinar o número de lugares que deve reservar para cada viagem entre cidades de forma a maximizar o lucro.
  Apresente um modelo de fluxo de custo mínimo para este problema.

# FCM – Exemplo 2 (2)








# FCM – Exercício 2

- Considere o enunciado anterior para o caso em que a capacidade do avião é de 100 lugares e o número de pessoas que pretende viajar entre cada par de cidades e o preço de cada viagem são os dados na tabela seguinte.

| | BF | BL | BP | PF | PL | LF |
|---|---|---|---|---|---|---|
| Preço | 200 | 150 | 100 | 175 | 75 | 125 |
| Procura | 20 | 45 | 10 | 40 | 90 | 55 |

a) Apresente um modelo de Programação Linear para este problema.
b) Obtenha uma solução óptima através da representação do modelo no *Excel* e da sua optimização.
c) Interprete a solução obtida. Qual o valor do lucro total?

# Fluxo máximo (FM) – Índice

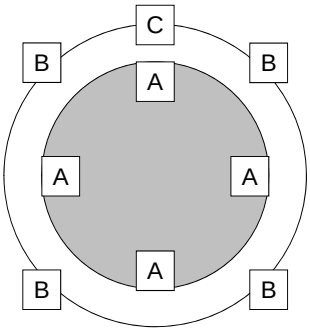
- Exemplo
- Modelo de Programação Linear
- Exercício



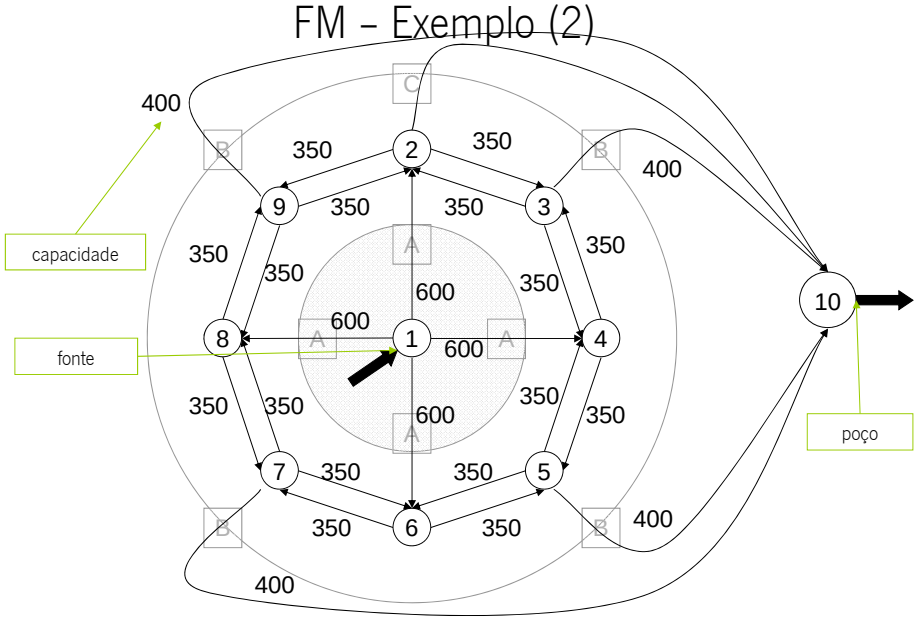


# FM – Exemplo (1)

- Considere o problema de determinar a taxa de evacuação de uma sala de espectáculos com a configuração representada na figura, onde a área a evacuar está assinalada a cinzento. As diferentes portas estão representadas por letras associadas a diferentes capacidades (A – 600 pessoas por minuto; B – 400 pessoas por minuto; C – 800 pessoas por minuto). O corredor de passagem entre as portas A e B tem capacidade para 350 pessoas por minuto em cada sentido. Apresente um modelo de fluxo máximo para este problema.



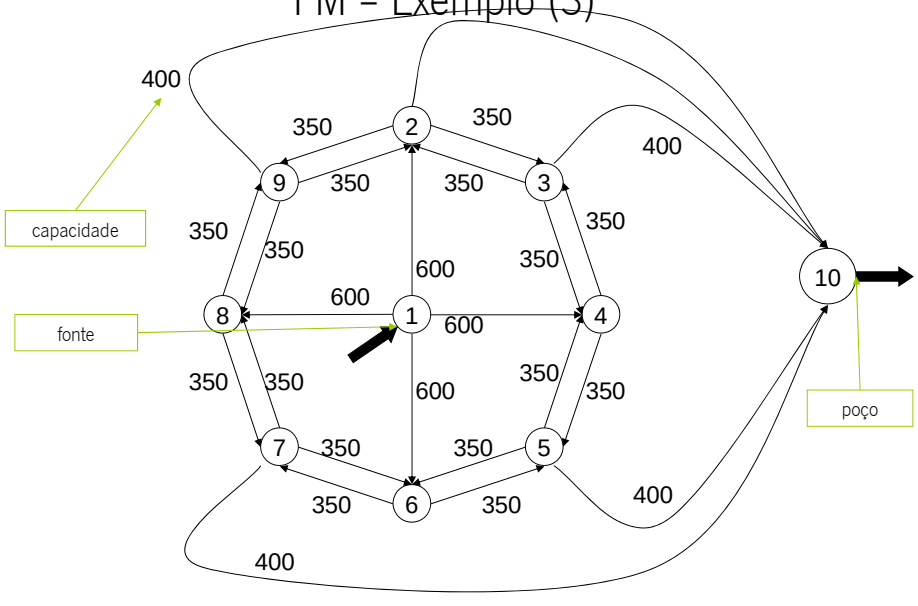
# FM – Exemplo (2)








# FM – Exemplo (3)



# FM – Modelo de Programação Linear

- Parâmetros de um modelo de fluxo máximo
  - $A$ e $N$ conjunto de arcos e conjunto de nodos, respectivamente
  - $s$ e $t$ nodos fonte e poço, respectivamente
  - $u_{ij}$ capacidade do arco $ij$, $\forall ij \in A$
- Variáveis de decisão
  - $v$ fluxo máximo entre $s$ e $t$
  - $x_{ij}$ fluxo no arco $ij$, $\forall ij \in A$

$$Max\ v$$
$$s.a:$$
$$\sum_{j:ij\in A} x_{ij} - \sum_{j:ji\in A} x_{ji} = \begin{cases} v\ para\ i = s \\ 0\ para\ i \neq s, i \neq t \\ -v\ para\ i = t \end{cases}$$
$$0 \leq x_{ij} \leq u_{ij}, \forall ij \in A$$





# FM – Exercício

- Para o exemplo:
  - Apresente um modelo de Programação Linear.
  - Obtenha uma solução óptima através da representação do modelo no *Excel* e da sua optimização.
  - Interprete a solução obtida. Qual a taxa de evacuação?

# Transformações (1)

- Arcos não orientados para arcos orientados
  - Se custo unitário associado ao arco não orientado é $c_{\{i,j\}}$ então $c_{ij}=c_{ji}= c_{\{i,j\}}$
  - Se a capacidade do arco não orientado é $u_{ij}$, então $x_{ij}+x_{ji}\leq u_{ij}$

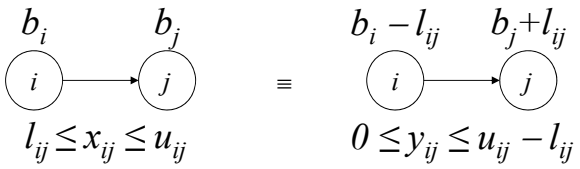

$$x_{\{i,j\}} = x_{ij} + x_{ji}$$





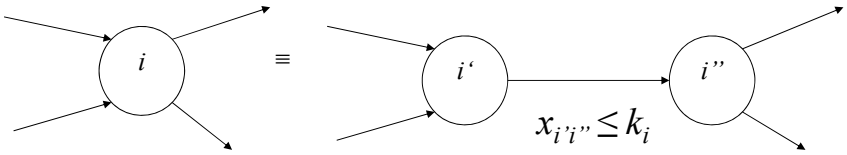
# Transformações (2)

- Remoção de limites inferiores
  - $x_{ij} \geq l_{ij}$ em que $l_{ij}$ é o limite inferior no arco $ij$
  - $u_{ij}$ capacidade (limite superior) do arco $ij$
  - $b_i$ oferta/procura do nodo $i$
  - Substituir variáveis $x_{ij}$ por $y_{ij}$ em que $y_{ij}=x_{ij}-l_{ij}$



# Transformações (3)

- Remoção de capacidades nos nodos
  - $k_i$ capacidade no nodo $i$ (não podem passar mais de $k_i$ unidades no nodo $i$)
  - Dividir nodo em dois (um de entrada e um de saída) com um limite superior no fluxo do arco que os une



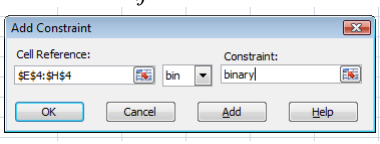


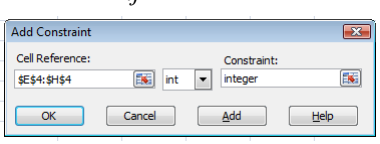



# Extensões – Introdução

- Os modelos de fluxos em rede podem servir de base à formulação de modelos mais elaborados
  - Exemplo 1: Transportes com restrições adicionais
  - Exemplo 2: Caminho mais curto com restrição temporal
  - Exemplo 3: Caminhos mais curtos disjuntos
- Tipicamente, ao adicionarem-se novas restrições/variáveis, se se pretender que fluxos sejam binários ou inteiros, tem de se passar a considerá-lo explicitamente.

$x_{ij} \in \{0,1\}$ $x_{ij}$ *inteiro*



# Extensões – Exemplo 1 (1)

- Um avião tem três compartimentos (Frente – F, Meio – M, Trás – T) dedicados ao transporte de carga. O peso e o volume a serem transportados em cada compartimento não podem exceder os valores dados na tabela seguinte.

| Compartimento | Peso máximo (toneladas) | Volume máximo ($m^3$) |
|---|---|---|
| F | 10 | 6800 |
| M | 16 | 8700 |
| T | 8 | 5300 |

De forma a ser mantido o equilíbrio do avião, a proporção entre o peso da carga efectivamente transportada em cada compartimento e o peso máximo do compartimento deverá ser igual para os três compartimentos.
Existem quatro cargas (A, B, C e D) a transportar no próximo voo. As suas características são dadas na tabela seguinte.





# Extensões – Exemplo 1 (2)

| Carga | Peso (toneladas) | Volume (m3/tonelada) | Lucro (€/tonelada) |
|---|---|---|---|
| A | 18 | 480 | 310 |
| B | 15 | 650 | 380 |
| C | 23 | 580 | 350 |
| D | 12 | 390 | 285 |

- Qualquer fracção de qualquer uma das cargas pode ser colocada em qualquer compartimento.
  Apresente um modelo de programação linear que lhe permita determinar qual a quantidade de cada carga que deve ser transportada em cada um dos compartimentos de forma a maximizar o lucro total.

# Extensões – Exemplo 2

- Determine o caminho mais curto entre os nodos 1 e 6 na rede e com as distâncias da seguinte Tabela.

| Arco | 12 | 13 | 24 | 25 | 32 | 34 | 35 | 45 | 46 | 56 |
|---|---|---|---|---|---|---|---|---|---|---|
| Distância | 1 | 10 | 1 | 2 | 1 | 4 | 12 | 10 | 1 | 2 |

- Considere que, para além da distância, também existem tempos associados aos arcos (dados na Tabela seguinte). Obtenha o caminho mais curto com a restrição adicional de este não demorar mais de 14 unidades de tempo a ser percorrido.

| Arco | 12 | 13 | 24 | 25 | 32 | 34 | 35 | 45 | 46 | 56 |
|---|---|---|---|---|---|---|---|---|---|---|
| Tempo | 10 | 3 | 1 | 3 | 2 | 7 | 3 | 1 | 7 | 2 |





# Extensões – Exemplo 3 (1)

- Para o problema de caminho mais curto descrito no exemplo anterior (sem a restrição temporal), obtenha os dois caminhos mais curtos disjuntos nos arcos (cada arco não pode fazer parte de mais de um caminho).

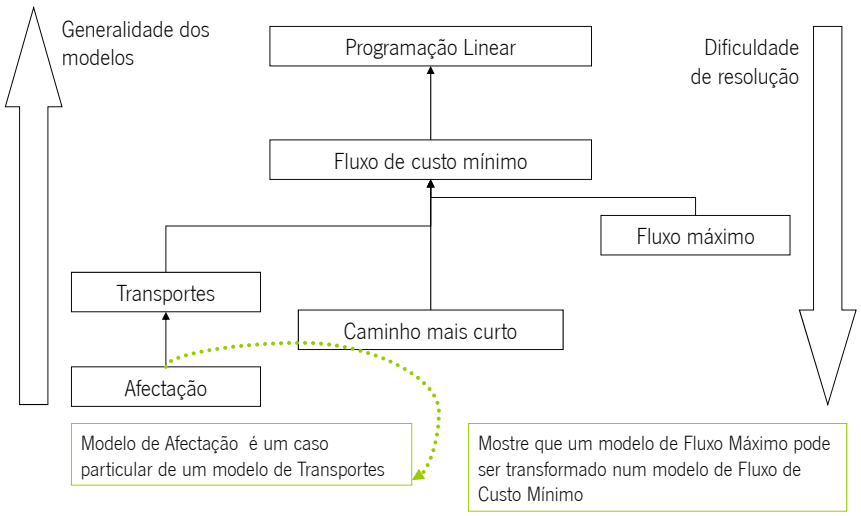
# Extensões – Exemplo 3 (2)

| | | | | | Caminho | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1 | | | | | | | | | | | 2 | | | | | | | | | | | | | | | | | | |
| | | | | | Arco | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | 12 | 13 | 24 | 25 | 32 | 34 | 35 | 45 | 46 | 56 | 12 | 13 | 24 | 25 | 32 | 34 | 35 | 45 | 46 | 56 | | | | | | | | | | |
| | | | | | Custo | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | 1 | 10 | 1 | 2 | 1 | 4 | 12 | 10 | 1 | 2 | 1 | 10 | 1 | 2 | 1 | 4 | 12 | 10 | 1 | 2 | 18 | | | | | | | | | |
| | | | | | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | | |
| Caminho | | Nodo | | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | 1 | = | 1 | | | | | | | |
| | | | | 2 | -1 | | 1 | 1 | -1 | | | | | | | | | | | | | | | | -0 | = | 0 | | | | | | | |
| | | | | 3 | | -1 | | | 1 | 1 | 1 | | | | | | | | | | | | | | 0 | = | 0 | | | | | | | |
| | 1 | | | 4 | | | -1 | | | -1 | | 1 | 1 | | | | | | | | | | | | 0 | = | 0 | | | | | | | |
| | | | | 5 | | | | -1 | | | -1 | -1 | | 1 | | | | | | | | | | | 0 | = | 0 | | | | | | | |
| | | | | 6 | | | | | | | | | -1 | -1 | | | | | | | | | | | -1 | = | -1 | | | | | | | |
| | | | | 1 | | | | | | | | | | | 1 | 1 | | | | | | | | | 1 | = | 1 | | | | | | | |
| | | | | 2 | | | | | | | | | | | -1 | | 1 | 1 | -1 | | | | | | 0 | = | 0 | | | | | | | |
| | | | | 3 | | | | | | | | | | | | -1 | | | 1 | 1 | 1 | | | | 0 | = | 0 | | | | | | | |
| | 2 | | | 4 | | | | | | | | | | | | | -1 | | | -1 | | 1 | 1 | | 0 | = | 0 | | | | | | | |
| | | | | 5 | | | | | | | | | | | | | | -1 | | | -1 | -1 | | 1 | -0 | = | 0 | | | | | | | |
| | | | | 6 | | | | | | | | | | | | | | | | | | | -1 | -1 | -1 | = | -1 | | | | | | | |
| | | | | 12 | 1 | | | | | | | | | | 1 | | | | | | | | | | 1 | <= | 1 | | | | | | | |
| | | | | 13 | | 1 | | | | | | | | | | 1 | | | | | | | | | 1 | <= | 1 | | | | | | | |
| | | | | 24 | | | 1 | | | | | | | | | | 1 | | | | | | | | 1 | <= | 1 | | | | | | | |
| | | | | 25 | | | | 1 | | | | | | | | | | 1 | | | | | | | 1 | <= | 1 | | | | | | | |
| | | Arco | | 32 | | | | | 1 | | | | | | | | | | 1 | | | | | | 1 | <= | 1 | | | | | | | |
| | | | | 34 | | | | | | 1 | | | | | | | | | | 1 | | | | | 0 | <= | 1 | | | | | | | |
| | | | | 35 | | | | | | | 1 | | | | | | | | | | 1 | | | | 0 | <= | 1 | | | | | | | |
| | | | | 45 | | | | | | | | 1 | | | | | | | | | | 1 | | | 0 | <= | 1 | | | | | | | |
| | | | | 46 | | | | | | | | | 1 | | | | | | | | | | 1 | | 1 | <= | 1 | | | | | | | |
| | | | | 56 | | | | | | | | | | 1 | | | | | | | | | | 1 | 1 | <= | 1 | | | | | | | |





# Considerações finais (1)



# Considerações finais (2)

- Se os dados do problema (transportes, afectação, caminho mais curto, fluxo de custo mínimo, fluxo máximo) são inteiros e existe uma solução óptima então existe sempre uma solução óptima inteira (isto é, em que os fluxos são todos inteiros)
- Existem algoritmos muito eficientes, que permitem obter soluções óptimas para problemas de muito grande dimensão em muito pouco tempo, para os problemas aqui abordados, por exemplo,
  - Para Transportes, o algoritmo Simplex para transportes
  - Para Afectação, o algoritmo húngaro
  - Para Caminho Mais Curto, o algoritmo de Dijkstra
  - Para o Fluxo de Custo Mínimo, o algorimo Simplex de rede
  - Para Fluxo Máximo, o algoritmo de Ford-.Fulkerson





# Introdução (1)

- Modelos de desenho (concepção) de redes (*network design*)
  - Existe um determinado objectivo traduzido por uma função (*função objectivo*) que aos arcos seleccionados faz corresponder um valor (por exemplo, um custo total)
  - Pretende-se determinar um conjunto de arcos de forma a que a função objectivo tome o menor (ou maior) valor possível

# Introdução (2)

- Problemas
  - Árvore de suporte de custo mínimo ("fácil de resolver")
  - Caixeiro viajante ("difícil de resolver")
- Representações
  - Gráfica (rede)
- Métodos de resolução
  - ASCM
    - Algoritmos específicos (exemplo: algoritmo de Kruskal)
  - PCV
    - Algoritmos específicos + Programação Inteira (mais potentes do que o *solver* do *Excel*...) para obter soluções óptimas
    - Heurísticas para obter soluções com alguma qualidade em tempos reduzidos (exemplo: heurística do vizinho mais próximo )



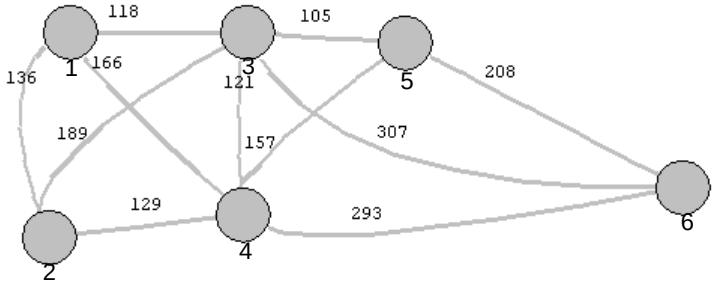


# Árvore de suporte de custo mínimo (ASCM) - Índice

- Exemplo
- Conceitos básicos
- Algoritmo de Kruskal

# ASCM – Exemplo

- Uma empresa de TV por cabo pretende planear a sua rede de cabo para abranger seis novas zonas residenciais. No esquema seguinte são representadas essas seis zonas, bem como as distâncias entre elas. Quais as ligações que devem ser estabelecidas de forma a que o comprimento de cabo utilizado seja o menor possível?



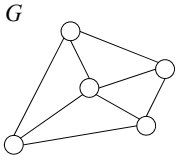




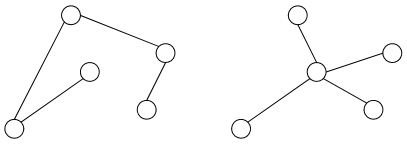

# ASCM – Conceitos básicos (1)

- Um grafo não orientado $G=(N,A)$ é uma árvore se
  - tem $n-1$ arcos (em que $n$ é o número de nodos de $G$) e é conexo (existe um caminho entre todos os pares de nodos) **OU**
  - tem $n-1$ arcos e não tem circuitos
- Um subgrafo $G'=(N',A')$ é uma árvore de suporte de um grafo não orientado $G=(N,A)$ se é uma árvore e $N'=N$ e $A' \subseteq A$
- Exemplos

$G$



Duas árvores de suporte de $G$



# ASCM – Conceitos básicos (2)

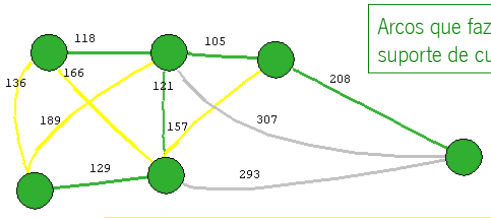
- Modelo da Árvore de Suporte de Custo Mínimo
  - Árvore de suporte garante a conectividade da rede com o menor número de arcos possível
  - Quando existem custos/distâncias associados à construção/utilização dos arcos, árvore de suporte de custo mínimo é o conjunto de arcos que garante a conectividade da rede com o menor custo/distância
  - Ponto de partida de muitos modelos de concepção de redes
  - Formulações de programação linear não são triviais (*solver* não é indicado) e existem algoritmos muito eficientes para obter uma solução óptima, como o algoritmo de Kruskal
  - Para resolver problemas pequenos / estudar o algoritmo, utilizar, por exemplo, o software *GIDEN – A graphical environment for network optimization – http://users.iems.northwestern.edu/~giden/*





# ASCM – Algoritmo de Kruskal (1)

- Enquanto não estiverem seleccionados $n-1$ arcos
  - Considerar o arco de menor custo dos que ainda não foram examinados
  - Se a inserção desse arco não formar um ciclo então incluí-lo
  - Se a inserção desse arco formar ciclo então excluí-lo




Arcos que fazem parte da árvore de suporte de custo mínimo

Arcos que não chegaram a ser examinados

Arcos que foram examinados mas formavam ciclos (não fazem parte da ASCM)

# ASCM – Algoritmo de Kruskal (2)

- Algoritmo do tipo guloso ("greedy")
  - Em cada passo, acrescenta um componente (um arco) à solução tendo em conta as consequências imediatas (menor custo) mas não tendo em conta as consequências globais
  - Tipicamente, os algoritmos gulosos não conduzem a uma solução óptima (por exemplo, o algoritmo para obter uma solução inicial do modelo de Transportes ou o algoritmo de vizinho mais próximo para o caixeiro viajante são gulosos e não conduzem a soluções óptimas)
  - O algoritmo de Kruskal para o problema da ASCM (embora seja guloso) conduz sempre a uma solução óptima
- Muito eficiente

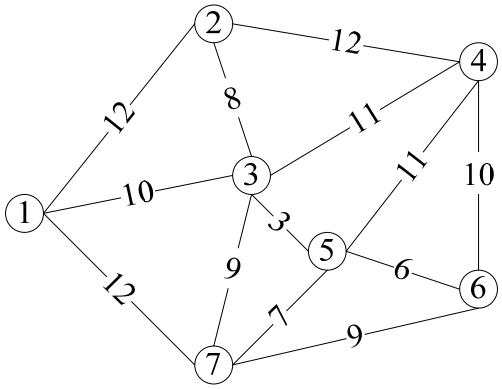




# Caixeiro viajante (TSP) - Índice

# TSP - Exemplo

- Visitar um conjunto 7 cidades regressando àquela de que se partiu, percorrendo a menor distância possível. As distâncias são dadas na figura. Cada cidade tem de ser visitada uma e uma só vez.





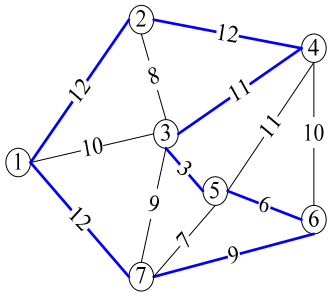



# TSP – Algoritmo do vizinho mais próximo (1)

- Considerar um nodo arbitrariamente e um caminho formado apenas por esse nodo
- Enquanto há nodos que não fazem parte do caminho
  - Determinar o nodo mais próximo em relação ao último considerado (de entre os que não fazem parte do caminho)
  - Incluí-lo no caminho
- Formar o circuito, incluindo no caminho o arco que liga o último nodo considerado ao primeiro

# TSP – Algoritmo do vizinho mais próximo (2)

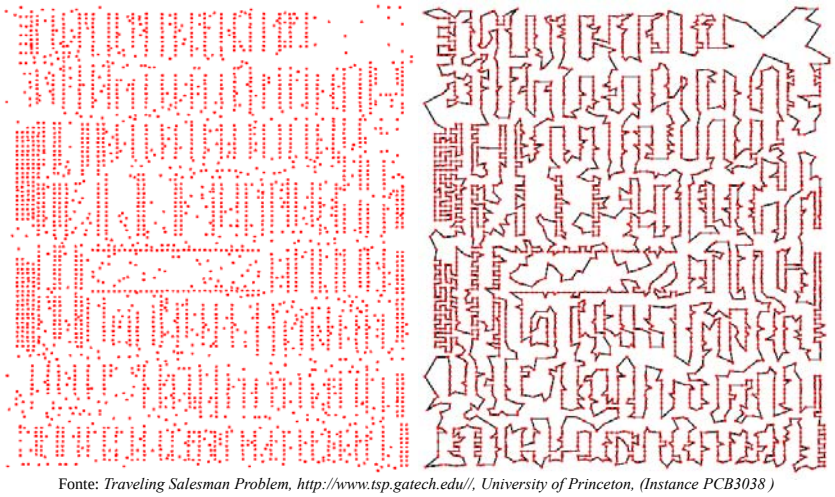
- Experimentar em http://wase.urz.uni-magdeburg.de/mertens/TSP/
- Solução obtida pela heurística do vizinho mais próximo (partindo de 3): 35 – 56 – 67 – 71 – 12 – 24 - 43 com valor 65








# TSP – Notas finais (1)



Fonte: *Traveling Salesman Problem, http://www.tsp.gatech.edu//, University of Princeton, (Instance PCB3038 )*

# TSP – Notas finais (2)

- O algoritmo do vizinho mais próximo é heurístico (por oposição a algoritmos exactos): não garante a obtenção de uma solução óptima
- Existem muitos outros métodos (heurísticos e exactos) para o caixeiro viajante
- Note-se que a abordagem de "força bruta" para obter a melhor solução não é viável. Num problema com 30 cidades há, aproximadamente, $2.65 \times 10^{32}$ alternativas. Testando 1 bilião de alternativas por segundo (um computador muito bom!), o tempo total de computação seria mais de oito milénios! Pode experimentar-se em *http://www.tutor.ms.unimelb.edu.au/tsp/tsp.html*
- Aplicações do modelo do caixeiro viajante: http://www.tsp.gatech.edu/apps/index.html
- Prémio para quem conseguir arranjar um algoritmo exacto eficiente para o TSP: 1 milhão de dólares (ver http://www.claymath.org/millennium/P_vs_NP/, o TSP é NP).





# TSP – Notas finais (3)

- "In May 2004, the traveling salesman problem of visiting all 24,978 cities in Sweden was solved: a tour of length 855,597 TSPLIB units (approximately 72,500 kilometers) was found and it was proven that no shorter tour exists. This is currently the largest solved TSP instance, surpassing the previous record of 15,112 cities through Germany set in April 2001."
  David Applegate, AT&T Labs – Research; Robert Bixby, ILOG and Rice University; Vašek Chvátal, Rutgers University; William Cook, Georgia Tech; Keld Helsgaun, Roskilde University.

# Resultados de aprendizagem

- Modelar problemas com base em modelos de fluxos em rede.
- Representar modelos de fluxos em rede através de representações em rede, de programação linear e em folhas de cálculo.
- Discutir as diferenças entre os diferentes modelos de fluxos em rede.
- Aplicar os algoritmos de transportes e húngaro a, respectivamente, problemas de transportes e de afectação de pequenas dimensões.
- Utilizar o *solver* do *Excel* (ou *software* similar para folhas de cálculo) para obter soluções óptimas de modelos de fluxos em rede.
- Modelar problemas com base em modelos de desenho de redes.
- Distinguir as diferenças entre os diferentes modelos de desenhos de redes e entre estes e os modelos de fluxos em rede.
- Aplicar os algoritmos de Kruskal e vizinho mais próximo a, respectivamente, problemas de árvore de suporte de custo mínimo e caixeiro viajante de pequenas dimensões.
- Distinguir algoritmos exactos de algoritmos heurísticos.





# Bibliografia e *links*


- R. Ahuja, T. Magnanti, J. Orlin, "Network Flows", 1993, Prentice-Hall.
- H. A. Eiselt, C.-L. Sandblom, "Integer Programming and Network Flows", Springer-Verlag, 2000.
- GIDEN – A graphical environment for network optimization, http://giden.northwestern.edu/
- F. S. Hillier and G. J. Lieberman, "Introduction to Operations Research", Mc-GrawHill, 8th edition, 2005.
- P. Jensen, University of Texas, http://www.me.utexas.edu/~jensen/ (inclui add-ins para *Excel*).
- Jim Orlin, MIT, Network Optimization, http://web.mit.edu/jorlin/www/15.082/15082_syllabus_2003.html.
- John Vande Vate, MIT OpenCourseWare, Systems Optimization, http://ocw.mit.edu/OcwWeb/Sloan-School-of-Management/15-057Systems-OptimizationSpring2003/CourseHome/index.htm.
- Ronald L. Rardin, "Optimization in Operations Research", Prentice-Hall, 1998.
- TutORial, http://www.ifors.ms.unimelb.edu.au/tutorial/
- J. M. Valério de Carvalho, "Optimização combinatória", Universidade do Minho, 2001.