

# Algoritmos em Grafos: Fluxo de Custo Mínimo em Redes de Transporte

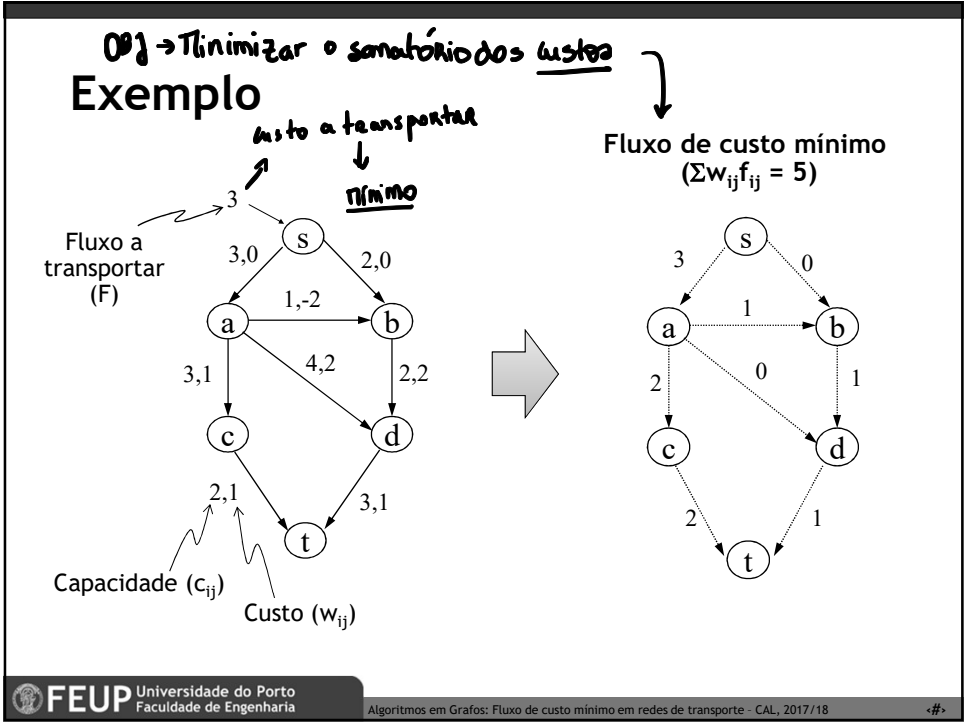
R. Rossetti, L. Ferreira, H. L. Cardoso, F. Andrade

FEUP, MIEIC

## Problema

- O objetivo é transportar uma certa quantidade  $F$  de fluxo ( $\leq$  máximo permitido pela rede) da fonte ( $s$ ) para o poço ( $t$ ), com um custo total mínimo
  - Para além da capacidade, arestas têm associado um custo ( $w_{ij}$ , custo de transportar uma unidade de fluxo)
  - Podem existir arestas de custo negativo (útil em problemas de maximização do valor, introduzindo sinal negativo)

↳ Algum incentivo por exemplo



### Formalização

Dados de entrada :

- $c_{ij}$  - capacidade da aresta que vai do nó  $i$  a  $j$  (0 se não existir)
- $w_{ij}$  - custo de passar uma unidade de fluxo pela aresta  $(i, j)$
- $F$  - quantidade de fluxo a passar pela rede

Dados de saída (variáveis a calcular):

- $f_{ij}$  - fluxo que atravessa a aresta que vai do nó  $i$  para o nó  $j$  (0 se não existir)

Restrições:

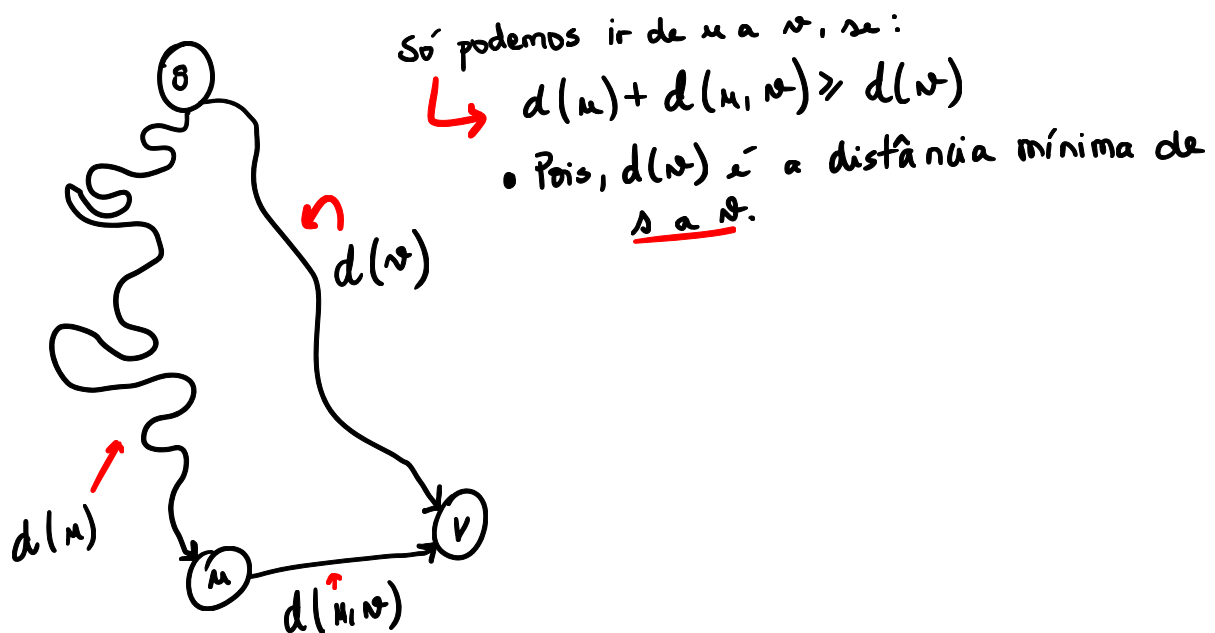
- $0 \leq f_{ij} \leq c_{ij}, \forall_{ij} \rightarrow$  Fluxo não pode ultrapassar capacidade
- $\sum_j f_{ij} = \sum_j f_{ji}, \forall_{i \neq s, t} \rightarrow$  Fluxos conservativos, tudo que entra, sai
- $\sum_j f_{sj} = F \rightarrow$  Fluxo total =  $F$

Objectivo:

$$\min \sum_{ij} f_{ij} \times w_{ij}$$

# Algoritmos

- Há muitos algoritmos propostos na literatura, aplicados à resolução deste problema, incluindo:
  - Cycle cancelling algorithms (*negative cycle optimality*)
  - Successive Shortest Path algorithms (*reduced cost optimality*)
  - Out-of-Kilter algorithms (complimentary slackness)
  - Network Simplex
  - Push/Relabel Algorithms
  - Dual Cancel and Tighten
  - Primal-Dual
  - ... *entre outros!*



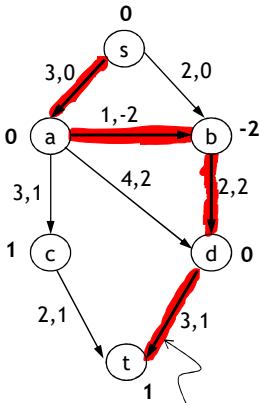
### Método dos caminhos de aumento mais curtos sucessivos

- Algoritmo ganancioso: no algoritmo de Ford-Fulkerson, escolhe-se em cada momento um caminho de aumento mais curto (no sentido de ter custo mínimo)
  - Pára-se quando se atinge o fluxo pretendido ou quando não há mais caminhos de aumento (neste caso dá um fluxo máximo de custo mínimo)
- **Restrição: aplicável só a redes sem ciclos de custo negativo**
  - Senão usa-se método mais genérico (cancelamento de ciclos negativos)
- Prova-se que dá a solução óptima (ver referências)

→ Ocasionalmente um loop infinito

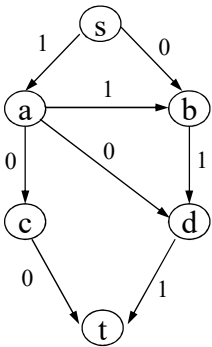
### Exemplo (1/2)

Grafo inicial de resíduos e custos =  
Grafo base de capacidades e custos



Caminho de custo mínimo de s a t  
(fluxo = 1) (custo unitário = 1)

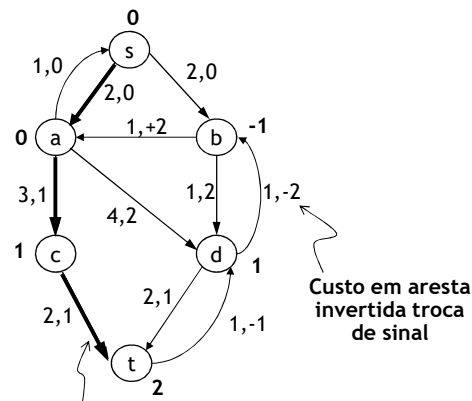
Grafo de fluxos  
resultante



custo:  $s \rightarrow a \rightarrow b$   
custo:  $b \rightarrow d \rightarrow t$

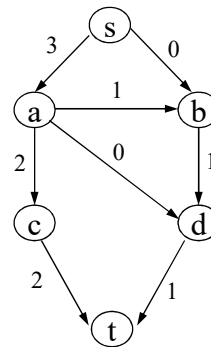
## Exemplo (2/2)

Novo grafo de resíduos e custos



Caminho de custo mínimo de s a t  
(fluxo = 2)(custo unit. = 2)

Grafo de fluxos resultante



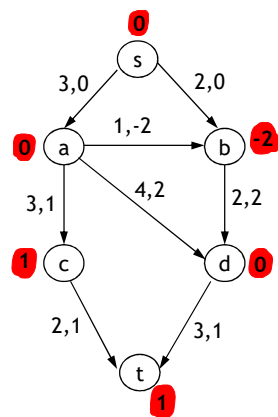
Atingiu-se o fluxo pretendido (3) =>  
Termina (custo total = 5)

## Melhoramento

- Dificuldade na abordagem anterior: arestas de custo negativo no grafo de resíduos
  - Devido a custos iniciais negativos ou a inversão de arestas no grafo de resíduos
  - Obriga a usar algoritmo menos eficiente na procura do caminho de custo mínimo (Bellman-Ford  $O(|V| \cdot |E|)$ )
- Solução: converte-se o grafo de resíduos num equivalente (para efeito de encontrar caminho de custo mínimo) sem custos negativos *→ se houverem custos negativos*
  - Na 1ª iteração usa-se algoritmo de **Bellman-Ford**  $O(|V| \cdot |E|)$
  - Em todas as seguintes, usa-se algoritmo de **Dijkstra**  $O(|E| \log |V|)$

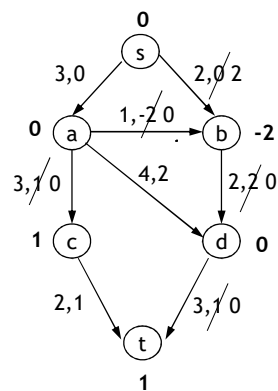
## Conversão do grafo de resíduos (1/2)

1. No grafo de resíduos inicial, determinar a “distância” mínima de  $s$  a todos os vértices ( $d(v)$ )
- Se existirem arestas (mas não ciclos) de peso negativo no grafo de resíduos inicial, usa-se o algoritmo de Bellman-Ford, de tempo  $O(|E| |V|)$
  - $d(v)$  também é chamado neste contexto o “potencial do nó  $v$ ”



## Conversão do grafo de resíduos (2/2)

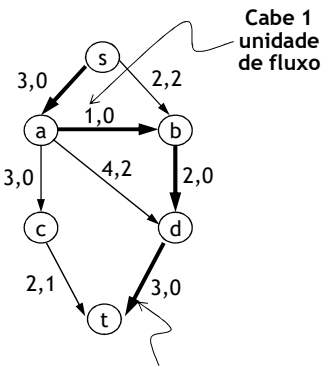
2. Substituir os custos iniciais  $w(u,v)$  por custos “reduzidos”  
 $w'(u,v) = w(u,v) + d(u) - d(v)$
- ↓
- $w'(u,v) \geq 0$  pois  $d(v) \leq d(u) + w(u,v)$
  - O custo  $w'$  de um caminho de  $s$  a  $t$ , usando os custos reduzidos, é igual ao custo usando os custos antes da redução subtraído de  $d(t)$  (demonstrar !)



### Determinação do próximo caminho de aumento

3. Seleccionar um caminho de custo mínimo de  $s$  para  $t$  no grafo de resíduos

- Os caminhos de custo mínimo de  $s$  para  $t$  têm custo reduzido 0 e custo “real” (antes da redução)  $d(t)$
- Como os caminhos de custo mínimo percorrem apenas arestas de custo 0, podem ser encontrados como uma pesquisa simples (DFS) em tempo linear
  - conceitualmente, eliminam-se arestas de custo  $> 0$

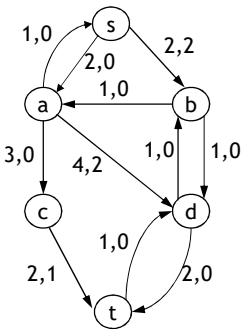


Caminho de custo mínimo (custo unitário reduzido 0) (custo unitário “real” 1)

### Aplicação do caminho de aumento

4. Aplicar o caminho de aumento

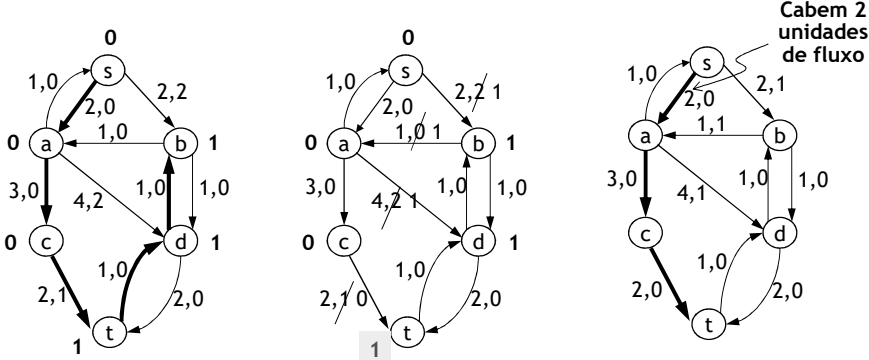
- Custo das arestas invertidas no grafo de resíduos é multiplicado por  $(-1)$
- Só que  $-1 \times 0 = 0 \dots$ 
  - ↓
- Evita-se assim a introdução de arestas de custo negativo!



(quantidade atual do fluxo = 1) (custo atual/real do fluxo = 1)

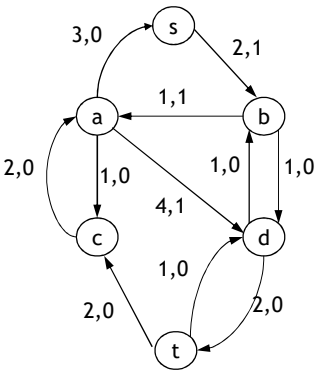
## Nova conversão do grafo de resíduos

5. Como não há mais caminhos de aumento de custo 0, volta-se a efectuar uma “redução” dos custos no grafo de resíduos
- Isto é, recalculam-se as distâncias mínimas (e.g. pelo algoritmo de Dijkstra!), e reduzem-se os custos (pode ser feito numa única passagem ...)



## Repetição do processo

6. Aplicar o caminho de aumento de custo 0
- Actualiza-se o grafo de resíduos



Quantidade atual de fluxo = 3  
= pretendido => FIM

Custo atual do fluxo = 5

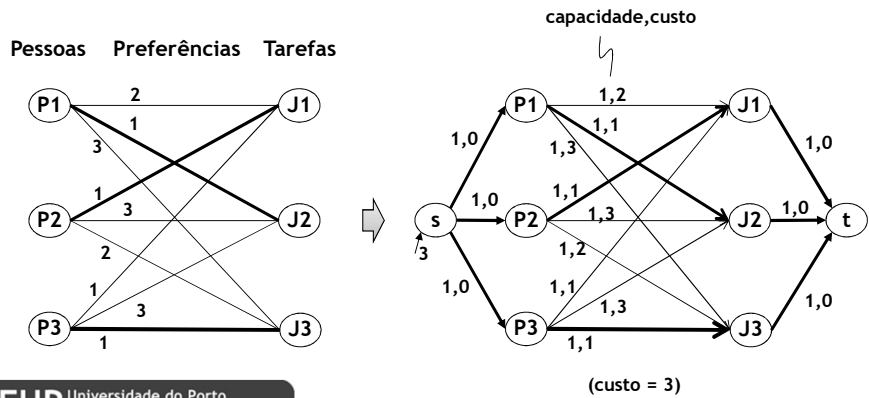


## Eficiência temporal

- Primeira redução do grafo de resíduos:  $O(|V| \cdot |E|)$  pelo algoritmo de Bellman-Ford
- Subsequentes reduções do grafo de resíduos e determinação do caminho de aumento de custo mínimo:  $O(|E| \log |V|)$  pelo algoritmo de Dijkstra
- Se todas as grandezas forem inteiras, o nº máximo de iterações é  $F$ , pois em cada iteração o valor do fluxo é incrementado de uma unidade
- Tempo total fica  $O(F \cdot |E| \log |V|)$

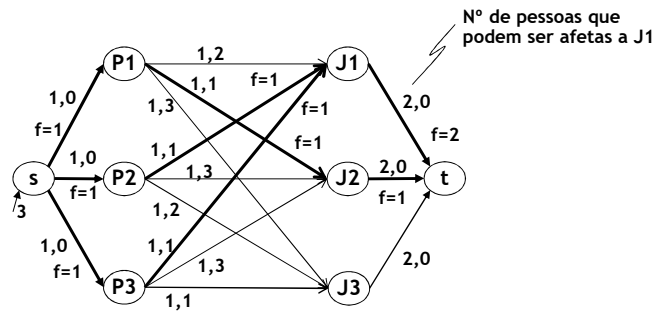
## Aplicação a problemas de emparelhamento (1/2)

- Problema de encontrar um emparelhamento de custo/peso mínimo num grafo bipartido (*minimum cost/weight bipartite matching*) (problema de afetação) pode ser reduzido ao problema de encontrar um fluxo de custo mínimo numa rede de transporte



## Aplicação a problemas de emparelhamento (2/2)

- E no caso de se poderem afetar várias pessoas à mesma tarefa?
  - Por exemplo, no caso anterior, admitindo 2 pessoas por tarefa



## Referências e informação adicional

- “Network Flows: Theory, Algorithms and Applications”, R. Ahuja, T. Magnanti & J. Orlin, Prentice-Hall, 1993
- “Efficient algorithms for shortest paths in sparse networks”. D. Johnson, J. ACM 24, 1 (Jan. 1977), 1-13