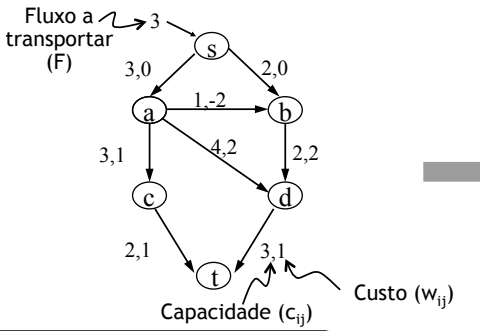
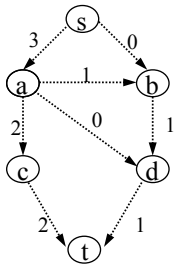
Fluxo de custo mínimo *

Problema

- O objectivo é transportar uma certa quantidade *F* de fluxo (≤ máximo permitido pela rede) da fonte para o poço, com um custo total mínimo
  - Para além da capacidade, arestas têm associado um custo (custo de transportar uma unidade de fluxo)
  - Podem existir arestas de custo negativo (útil em problemas de maximização do valor, introduzindo sinal negativo)



Fluxo de custo mínimo (= $\Sigma w_{ij} f_{ij}$ = 5)



FEUP Universidade do Porto Faculdade de Engenharia
Algoritmos em Grafos: Fluxo máximo em redes de transporte - CAL, 2013/14 ‹#›

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:

$$0 \leq f_{ij} \leq c_{ij}, \forall_{ij}$$

$$\sum_j f_{ij} = \sum_j f_{ji}, \forall_{i \neq s,t}$$

$$\sum_j f_{sj} = F$$

Objectivo:

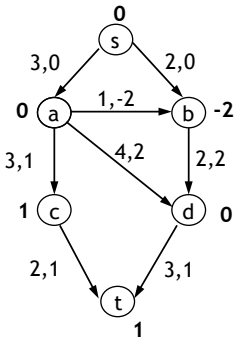
$$\min \sum_{ij} f_{ij} \times w_{ij}$$

Método dos caminhos de aumento mais curtos

- Algoritmo ganancioso: no algoritmo de Ford-Fulkerson, escolhe-se em cada momento um caminho de aumento de 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 quando a rede não tem ciclos de custo negativo
  - Senão aplica-se método mais genérico (cancelamento de ciclos negativos)
- Prova-se que dá a solução óptima (ver referências)
- Dificuldade: 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
- Solução: conversão do grafo de resíduos num equivalente (para efeito de encontrar caminho de custo mínimo) sem custos negativos

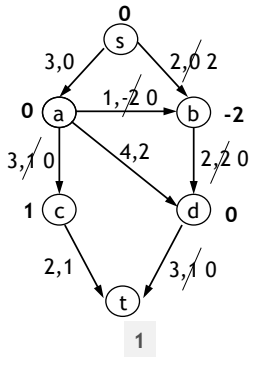
Conversão do grafo de resíduos (1/2)

1. Determinar a distância mínima de *s* a todos os vértices no grafo de resíduos (*d(v)*)
   - Grafo de resíduos inicial (com fluxos nulos) é idêntico ao grafo de capacidades
   - 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|)



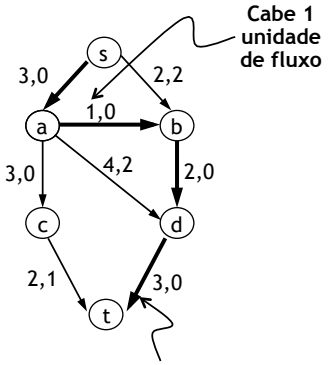
Conversão do grafo de resíduos (2/2)

2. Substituir os custos iniciais w(u,v) por custos "reduzidos" w'(u,v) = w(u,v) + d(u) - d(v)
   - w'(u,v) >= 0 pois d(v) <= 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

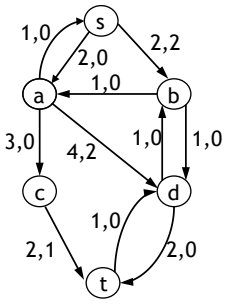
3. Seleccionar um caminho de custo mínimo de *s* para *t* no grafo de resíduos
   - Os caminhos de custo mínimo de *s* para *t* têm custo reduzido 0 e custo "real" (antes da redução) d(t)
   - Como os caminhos de custo mínimo percorrem apenas arestas de custo 0, podem ser encontrados como uma pesquisa simples em tempo linear



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 x 0 = 0 …
   - Evita-se assim a introdução de arestas de custo negativo!

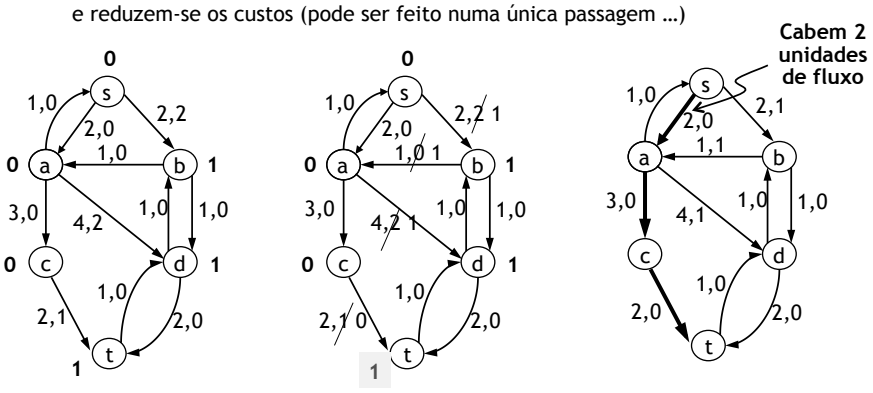


(quantidade actual do fluxo = 1)

(custo actual do fluxo = 1)

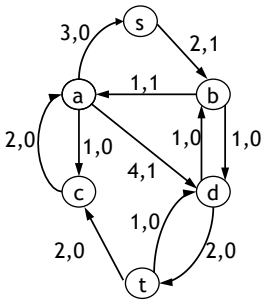
Nova conversão do grafo de resíduos

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 (agora 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 actual de fluxo = 3 = pretendido => FIM

Custo actual do fluxo = 5

Eficiência *

- Primeira redução do grafo de resíduos: O(|V| |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 (≤ |E|), pois em cada iteração o valor do fluxo é incrementado de uma unidade
- Tempo total fica $O(F\ |E|\ \log|V|)$ usando heap binário (2-heap)
- Pode ser melhorado para $O(F\ |E|\ \log_{\lceil |E|/|V|+1 \rceil} |V|)$ usando d-heap em que $d = \lceil |E|/|V|+1 \rceil$ (nº de filhos de cada nó do heap), de acordo com o seguinte teorema:
  - Teorema (Johnson 1977): Seja $d = \lceil m/n+1 \rceil$. Um d-heap suporta *m* operações do tipo 1 (insert/decreaseKey) e *n* operações de tipo 2 (delete) em tempo $O(m \log_d n)$ (neste caso fazemos n=|V| e m=|E|)

Referências e informação adicional

- "Introduction to Algorithms", Second Edition, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press, 2001
- "The Algorithm Design Manual", Steven S. Skiena, Springer-Verlag, 1998
- "Efficient algorithms for shortest paths in sparse networks". D. Johnson, J. ACM 24, 1 (Jan. 1977), 1-13