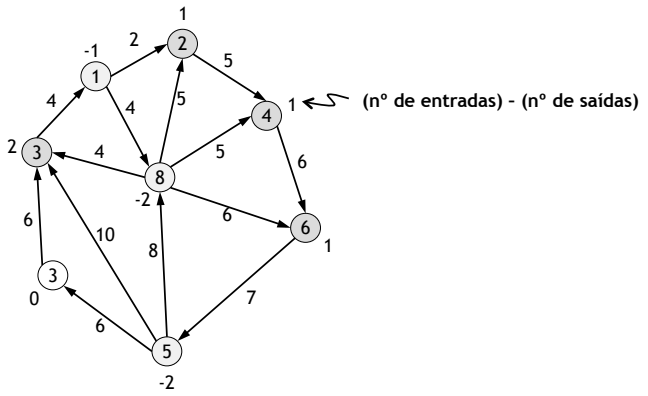
Caso de grafos dirigidos

- Modificações ao algoritmo anterior:
  1) Identificam-se os vértices com nº diferentes de arestas a entrar e a sair
  2) Procuram-se os caminhos mais curtos de vértices que têm défice de saídas para vértices que têm défices de entradas
  3) Constrói-se um grafo bipartido, em que os vértices são distinguidos consoante têm mais arestas a entrar ou a sair e são replicados consoante a diferença entre entradas e saídas
  4) Procura-se um emparelhamento perfeito de peso mínimo num grafo bipartido
     - Em vez de replicar os vértices no ponto 3, pode-se associar a cada vértice uma multiplicidade (nº de emparelhamentos em que pode participar) e converter o problema diretamente para um problema de fluxo máximo de custo mínimo em que algumas arestas têm capacidade superior a 1 -> Ver no exemplo a seguir
- Resolúvel igualmente em tempo polinomial
- Infelizmente, o problema é NP-completo (tempo exponencial) quando se combinam arestas dirigidas com arestas não dirigidas (grafos mistos)
  - Exemplo: determinar o percurso a seguir pelo camião do lixo, quando algumas ruas têm sentidos únicos

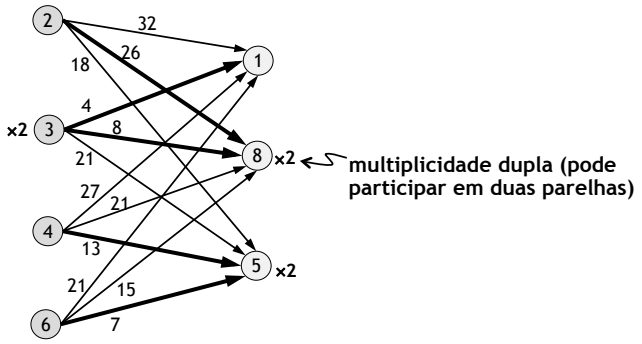
Exemplo (1/4)

- Grafo *G* e vértices com diferente nº de entradas e saídas:



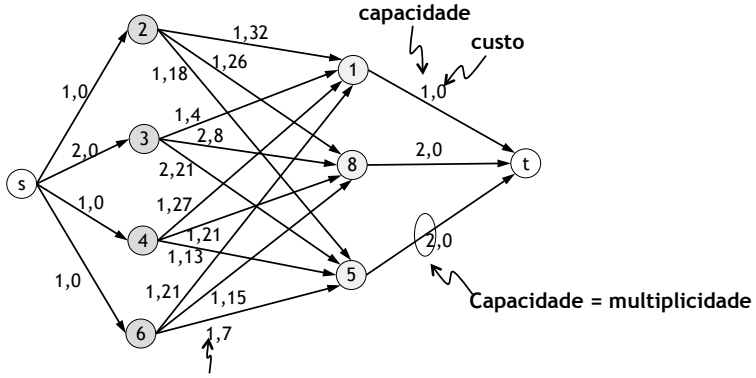
Exemplo (2/4)

- Grafo *G'* com distâncias de vértices com deficit de saídas para vértices com deficit de entradas, e emparelhamento perfeito de peso mínimo (arestas a traço forte, obtidas resolvendo o problema de fluxo máximo de custo mínimo indicado no slide seguinte):



Exemplo (3/4)

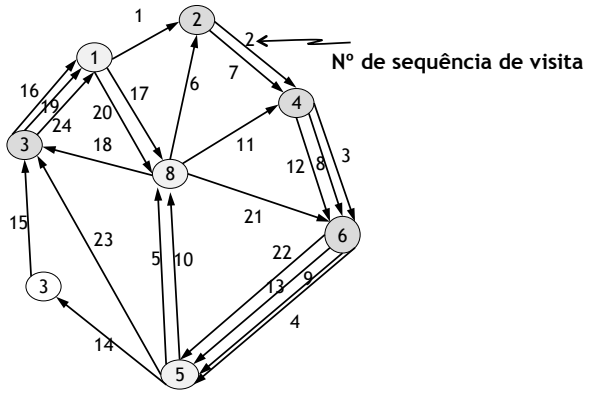
- Formulação do problema de emparelhamento óptimo como problema de fluxo máximo de custo mínimo:



Capacidade = min(multiplicidade(origem),multiplicidade(destino))

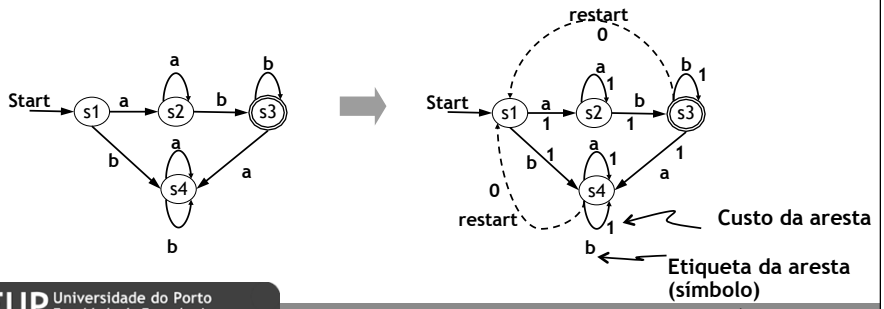
Exemplo (4/4)

- Grafo *G** correspondente, após duplicação de caminhos mais curtos entre os vértices emparelhados, e uma numeração possível das arestas ao longo de um circuito de Euler (distâncias não são mostradas):



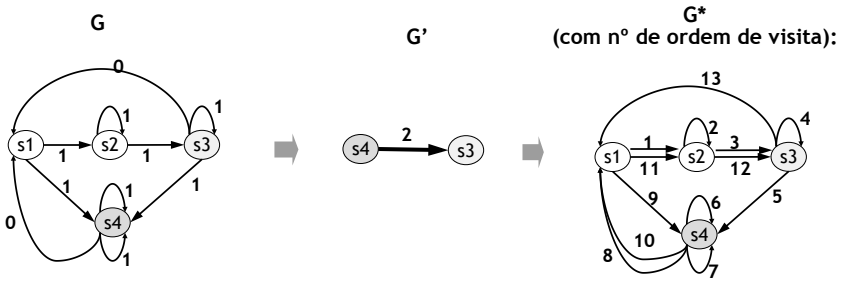
Exemplo de aplicação (1/2)

- Achar um conjunto de sequências de teste completas (do estado inicial a um estado final) de comprimento total mínimo cobrindo todas as transições num autómato finito
  - Ligam-se os estados finais ao estado inicial e procura-se um percurso ótimo do carteiro
  - Nota: conceito de estado final faz mais sentido em máquinas de estados UML; no caso de autómatos finitos, podem-se considerar como tal estados de aceitação e estados absorventes (donde não é possível sair)



Exemplo de aplicação (2/2)

- Resolução do problema do carteiro chinês dirigido:



- Solução final:
  - Caminho de Euler usando etiquetas:
    a-a-b-b-a-a-b-restart-b-restart-a-b-restart
  - Strings de teste: aabbaab, b, ab

Referências e mais informação

- "The Algorithm Design Manual", Steven S. Skiena, Springer-Verlag, 1998