第二章 网络中的若干优化问题

第四节 其他网络优化问题

目录

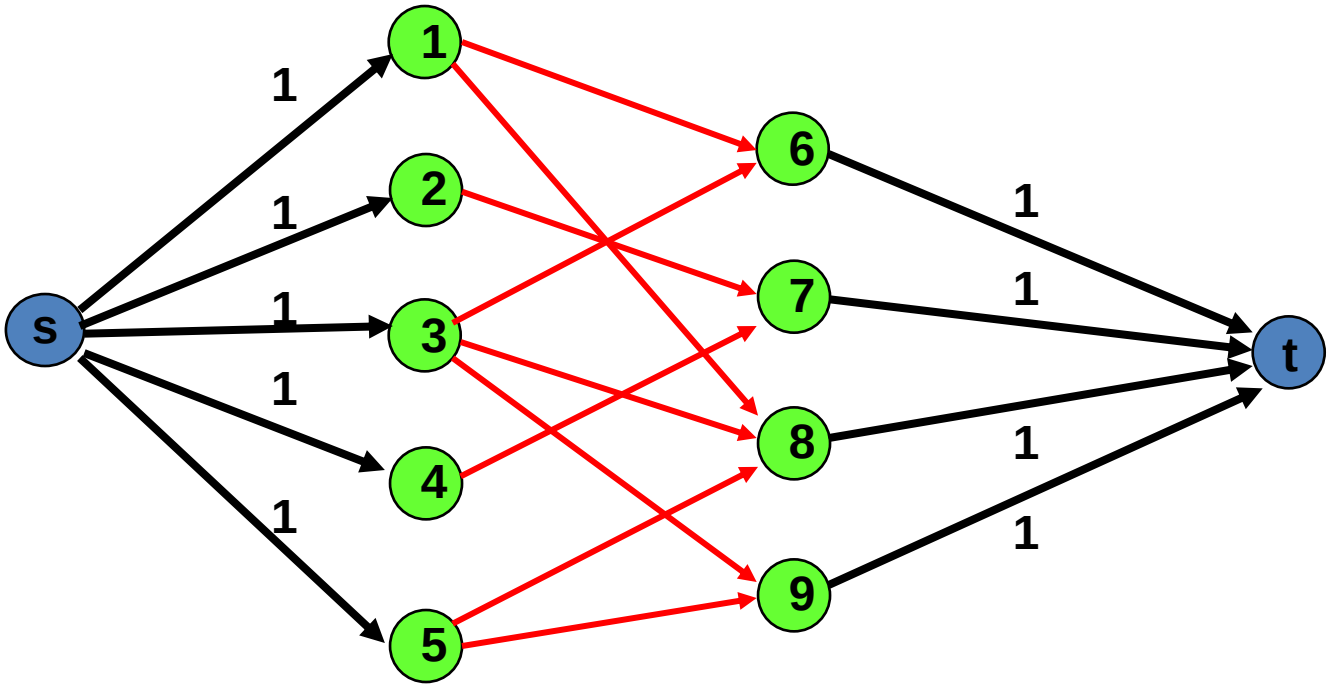
- 匹配问题(Matching problem)
- 最小生成树(Minimum spanning tree)

Matching

- A **matching** in a graph $G=(V, E)$ is a subset M of the edges E such that no two edges in M share a common end node.
  - A **perfect matching** M in G is a matching such that each node of G is incident to an edge in M.
  - If the edges of the graph have an associated weight, then a **maximum(minimum) weighted perfect matching** is a perfect matching such that the sum of the weights of the edges in the matching is maximum (minimum). There is a complicated but polynomial-time algorithm (e.g., $O(n^2m)$ ~ blossom algorithm by Edmonds 1965; $O(n^3)$ ~ Gabow, 1975; Lawler, 1976) for finding a minimum weight perfect matching.
  - A simpler case **(Bipartite weighted matching)**: A graph is **bipartite** if it has two mutually exclusive and complete subsets of nodes and the edges are only allowed between nodes of different subsets. A matching in a bipartite graph therefore assigns nodes of one subset to nodes of the other subset. Matchings in bipartite graphs can be computed efficiently as an assignment problem.

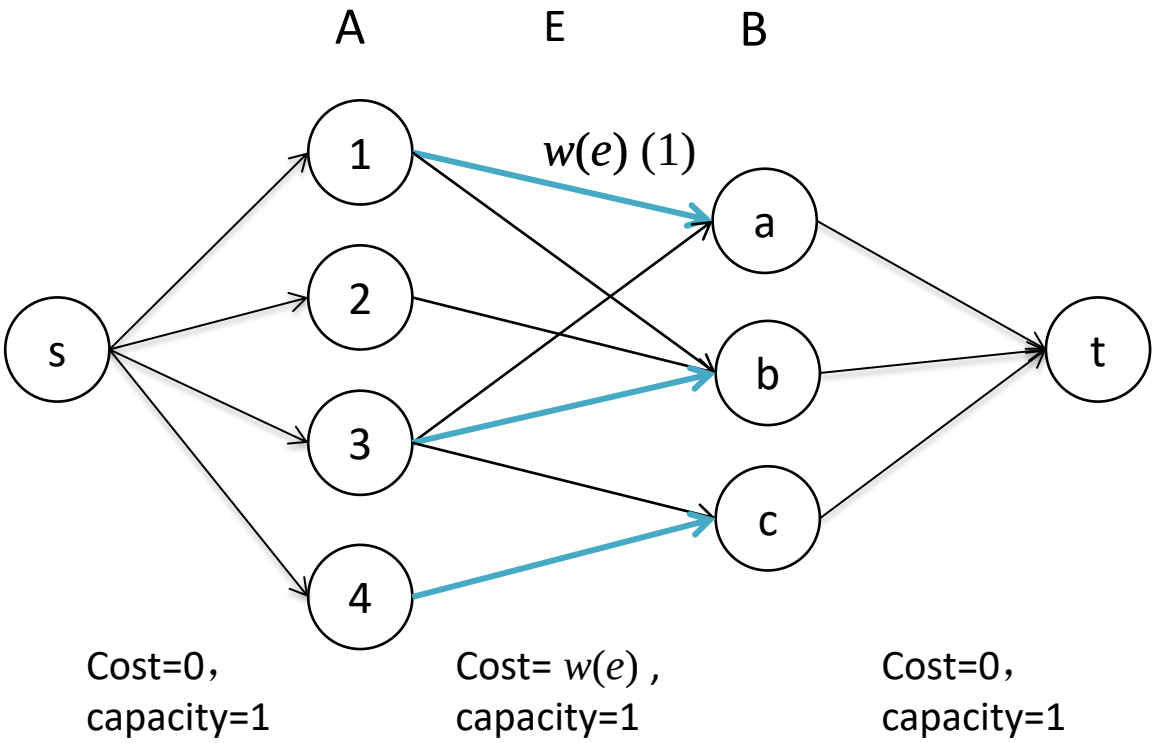
Bipartite Cardinality Matching Problem

- In a bipartite cardinality matching problem (or simply bipartite matching problem), we wish to identify a matching of maximum cardinality in a bipartite undirected network.
- This problem can be solved by transforming it into a maximum flow problem:



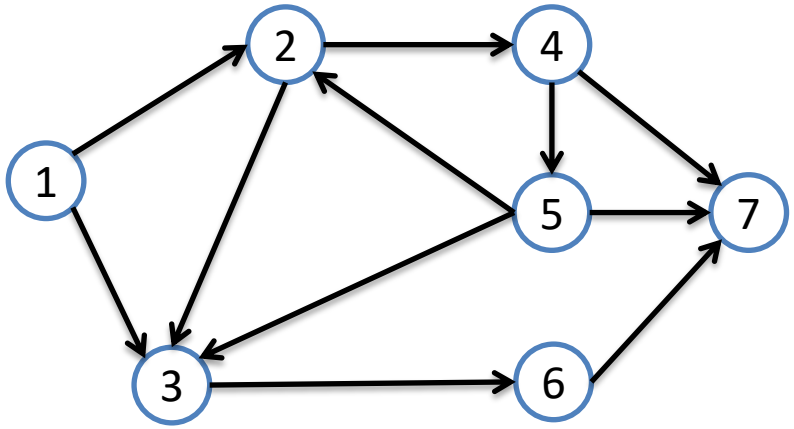
Bipartite weighted matching problem

- In a bipartite weighted matching, we wish to identify a matching of maximal total weight in a bipartite undirected network.

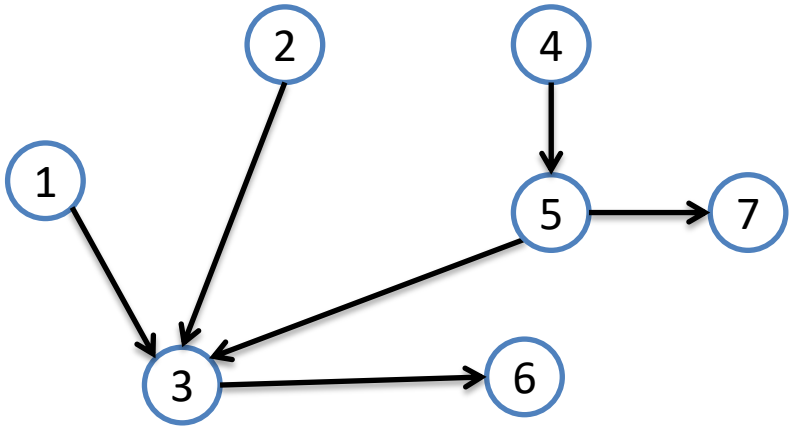


MINIMUM SPANNING TREE (MST)

- Among all the spanning trees of a weighted (cost) and connected graph $G = (N, E)$, the one (or possibly more) with the least total weight is called a minimum spanning tree (MST).



Graph G



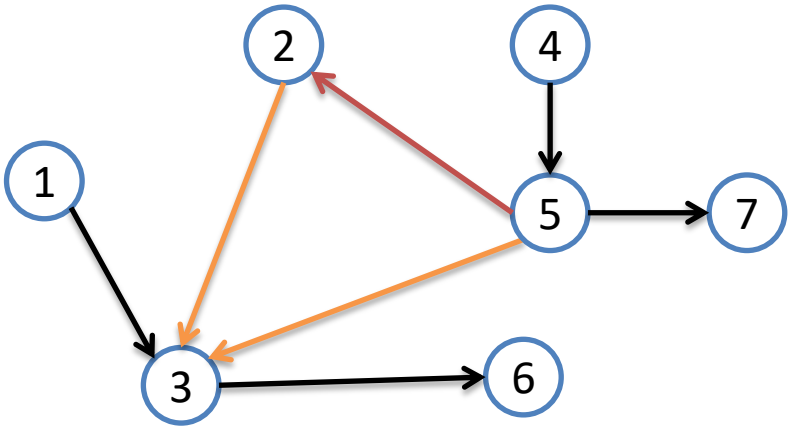
A spanning tree of G

Applications

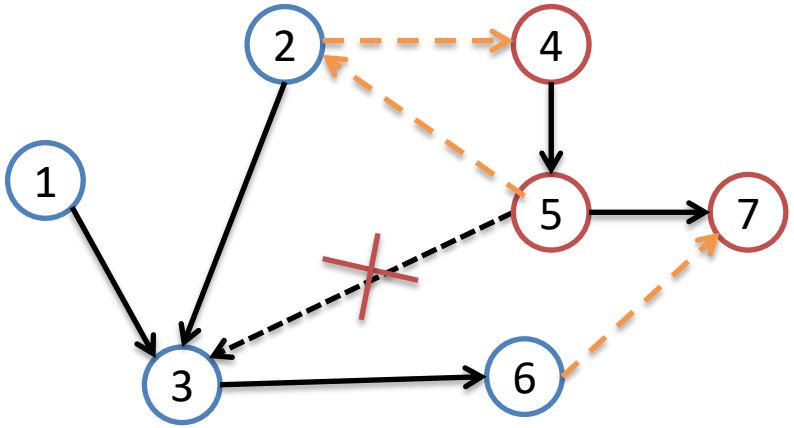
- Minimum spanning tree problem arises in the following design problem:
  - Connect terminals in cabling the panels of electrical equipment. How should we wire the terminals to use the least possible length of the wire?
  - Constructing a pipeline network to connect a number of towns using the smallest possible total length of pipeline
  - Linking isolated villages in a remote region, which are connected by roads but not yet by telephone service. Determine along which stretches of roads we should place telephone lines, using the minimum possible total miles of the lines, to link every pair of villages.

Properties of spanning tree

- For every nontree arc (k,l), the spanning tree T contains a unique path from node k to node l. The arc (k,l) together with this unique path defines a cycle.
- If we delete any tree arc (i,j) from a spanning tree, the resulting graph partitions the node set N into two subsets. The arcs from the underlying graph G whose endpoints belong to the different subsets constitute a cut.



A fundamental cycle



A fundamental cut

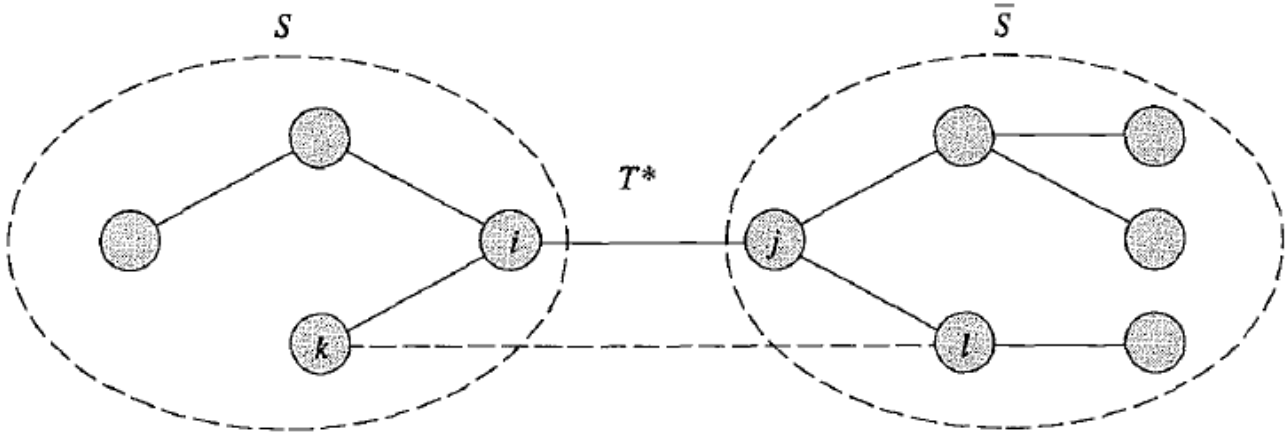
Optimality Conditions 1

- **(Cut Optimality Conditions)** A spanning tree T* is a minimum spanning tree if and only if it satisfies the following cut optimality conditions: For every tree arc $(i,j) \in T^*$, $c_{ij} \leq c_{kl}$ for every arc (k,l) contained in the cut formed by deleting arc (i,j) from T^*.

- The cut optimality conditions imply that every arc in a minimum spanning tree is a minimum cost arc across the cut that is defined by removing it from the tree.

Optimality Conditions 2

- **(Path Optimality Conditions)** A spanning tree T* is a minimum spanning tree if and only if it satisfies the following path optimality conditions: For every nontree arc (k,l) of G, $c_{ij} \leq c_{kl}$ for every arc (i,j) contained in the path in T^* connecting nodes *k* and *l*.



Kruskal's Algorithm

1. Let edge set $E0 = \Phi$. Find the cheapest edge (i,j) in the graph (if there is more than one, pick one at random) and include it into $E0$
2. Find the cheapest edge in $E \setminus E0$ that doesn't close a cycle with those already chosen in $E0$. Include it into $E0$.
3. Repeat Step 2 until you reach out to every vertex of the graph (or $|E0| = n - 1$).

Computational complexity $O(n^2 \log(n))$

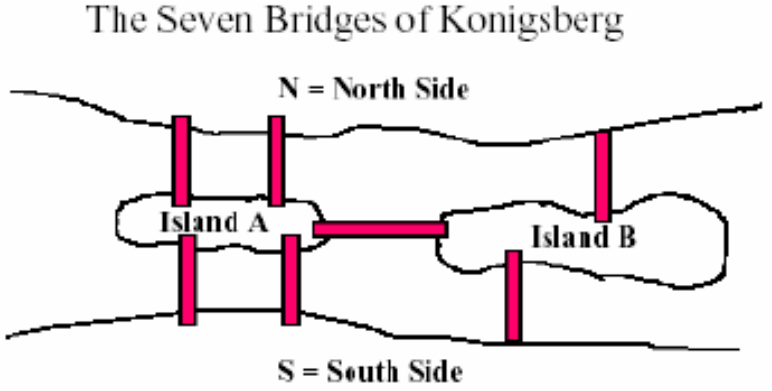
Prim's Algorithm

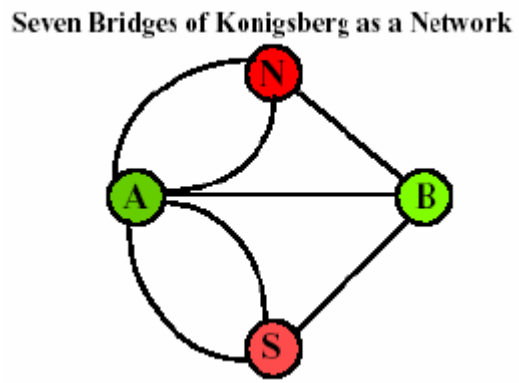
1. Pick any vertex as a starting vertex set. (Call it S). Let $E0 = \Phi$.
2. Find the nearest neighboring vertices $k \in S$ and $p \in N \setminus S$, such that $c(p,k) = \min\{c(i,j), i \in S, j \in N \setminus S\}$. Let $S \leftarrow S \cup \{p\}$, $E0 \leftarrow E0 \cup \{(p,k)\}$.
3. Repeat Step 2 until all vertices are in S, e.g. $|S| = n$.

Computational complexity $O(n^2)$

EULER TOUR

- An Euler tour in an *undirected* graph *G* is a cycle using every edge exactly once. An Euler tour in a *directed* graph *G* is a directed cycle which includes every arc exactly once.

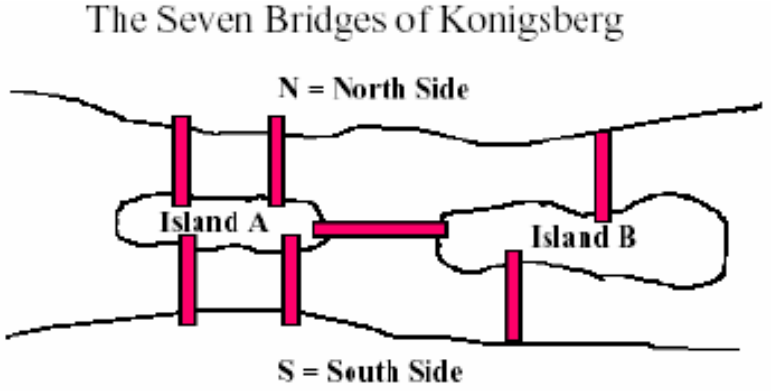


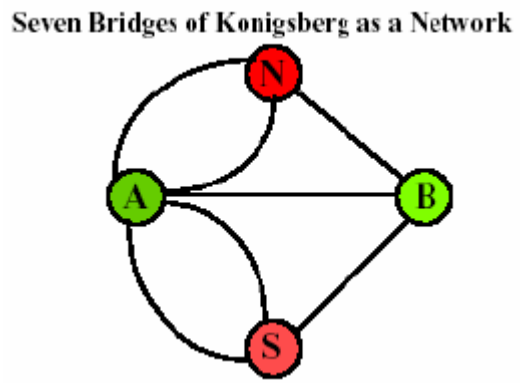


Euler's famous "test problem:" can one find out whether or not it is possible to cross each bridge exactly once?"

EULER TOUR

- A connected undirected graph has an Euler tour if and only if the degree of every vertex is even.
- A weakly connected directed graph has an Euler tour if and only if for any vertex *v*, *indeg(v) = outdeg(v).*





Euler's famous "test problem:" can one find out whether or not it is possible to cross each bridge exactly once?"