Lecture 3: Totally Unimodularity and Network Flows

(3 units)

Outline

- Properties of Easy Problems
- Totally Unimodular Matrix
- Minimum Cost Network Flows
- Dijkstra Algorithm for Shortest Path Problem
- Ford-Fulkerson Algorithm for Max-flow Problem
- Minimum Spanning Tree and Minimum Steiner Tree

Properties of Easy Problems

- Consider a graph $G = (V, E)$ with n nodes and m edges ($m \geq n$). An algorithm for the graph is efficient if the algorithm requires $O(m^p)$ elementary calculations in the worst case.
- Separation Problem. For COP problem $\max\{c^T x \mid x \in X \subseteq R^n\}$. Given $x^* \in R^n$, is $x^* \in conv(X)$? If not, find a cutting plane: $\pi^T x \leq \pi_0$, separating X and x^*.
- Four properties for an easy problem:

$$(P) \qquad \max\{c^T x \mid x \in X\}.$$

- Efficient optimization property: there exists an efficient algorithm for (P).
- Strong duality property: there exists a strong dual problem

$$(D) \quad \min\{\omega(u) \mid u \in U\},$$

  with which we can check the optimality condition quickly: $x^* \in X$ is optimal to (P) iff there exists $u^* \in U$ with $c^T x^* = \omega(u^*)$.
- Efficient separation property: there exists an efficient algorithm for the associated separation problem.
- Explicit convex hull property: A compact description of $conv(X)$ is known. So (P) is equivalent to linear program:

$$\max\{c^T x \mid x \in conv(X)\}.$$

Totally Unimodular Matrix

- Consider the linear integer programming:

$$(IP) \qquad \begin{aligned} \min \ & c^T x \\ \text{s.t.} \ & Ax \leq b \\ & x \geq 0, x \text{ integer}. \end{aligned}$$

- Let $X = \{x \mid Ax \leq b, x \in \mathbb{Z}_+^n\}$. Consider the following linear program:

$$(LP) \qquad \begin{aligned} \min \ & c^T x \\ \text{s.t.} \ & Ax \leq b, x \in \mathbb{R}_+^n. \end{aligned}$$

- Question: When are the extreme points of the feasible region (polyhedron) in (LP) all integral?

- Sufficient condition: If the optimal basis B of (A, I) satisfies $det(B) = \pm 1$, then (LP) is equivalent to (IP).
  $(A, I) = (B, N)$, the optimal solution of (LP): $x^* = (B^{-1}b, 0)$.
  Cramer's rule: $B^{-1} = B^*/det(B)$, where B^* is the adjoint matrix with entries being products of terms of B.
- Totally unimodular matrix: A matrix A is TU if every square submatrix of A has determinant $+1$, -1 or 0.
- If A is TU, then $a_{ij} \in \{+1, -1, 0\}$.
- A is TU if and only if
  - A^T is TU; or
  - (A, I) is TU.

- A useful sufficient condition for TU: A is TU if
  (i) $a_{ij} \in \{+1, -1, 0\}$;
  (ii) $\sum_{i=1}^{m} |a_{ij}| \leq 2$;
  (iii) $\exists$ partition (M_1, M_2) of rows $\{1, 2, \ldots, m\}$ such that each column j containing two nonzero coefficients satisfies $\sum_{i \in M_1} a_{ij} = \sum_{i \in M_2} a_{ij}$.
  Proof: Assume that A is not TU and let B be the smallest square submatrix of A for which $deta(B) \neq 0, 1, -1$. B cannot contain a column with a single nonzero entry (otherwise B would not be minimal). So B contains two nonzero entries in each column. Adding the rows in M_1 and subtracting the rows in M_2 gives the zero vector and so $det(B) = 0$, a contradiction.
- The linear program $\max\{c^T x \mid Ax \leq b, x \in R_+^n\}$ has an integral optimal solution for all integral b for which it has a finite optimal value iff A is TU.
- Strong duality, explicit convex hull and efficient separation properties all holds when A is TU.

Minimum cost network flows

- A digraph $G = (V, A)$ with arc capacity h_{ij} (integer), for $(i, j) \in A$, demand b_i at node $i \in V$, unit flow cost c_{ij} for $(i, j) \in A$.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = b_i, \ \ i \in V \\ & 0 \le x_{ij} \le h_{ij}, x_{ij} \text{ integer.} \end{aligned}$$

- An example with 6 nodes and 11 arcs.

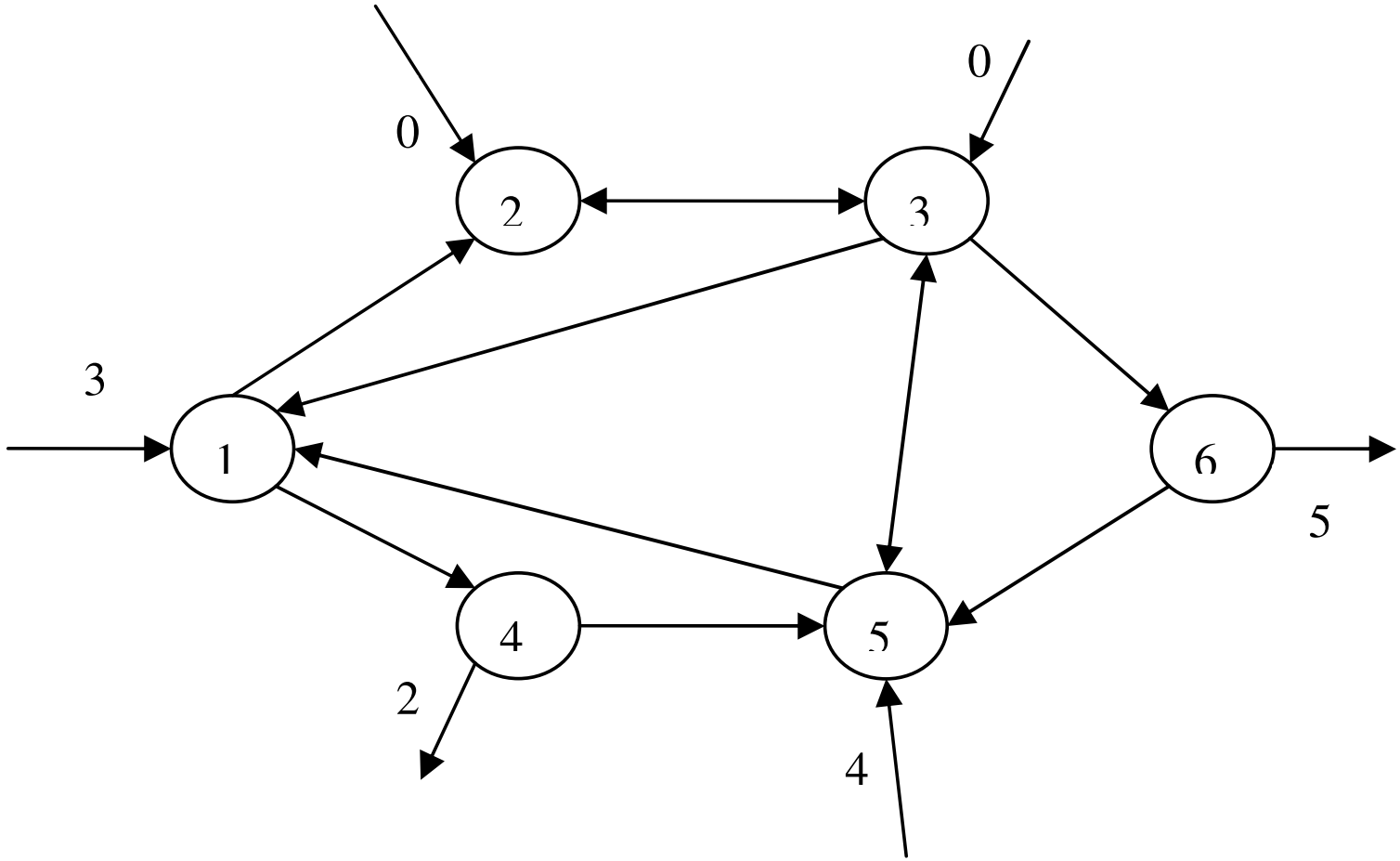


Figure: A digraph G with 6 nodes and 11 arcs

- The node-arc adjacent matrix of G:

	x_{12}	x_{14}	x_{23}	x_{31}	x_{32}	x_{35}	x_{36}	x_{45}	x_{51}	x_{53}	x_{65}	b
1	1	1	0	-1	0	0	0	0	-1	0	0	$=3$
2	-1	0	1	0	-1	0	0	0	0	0	0	$=0$
3	0	0	-1	1	1	1	1	0	0	-1	0	$=0$
4	0	-1	0	0	0	0	0	1	0	0	0	$=-2$
5	0	0	0	0	0	-1	0	-1	1	1	-1	$=4$
6	0	0	0	0	0	0	-1	0	0	0	1	$=-5$

- The constraint in Minimum Cost Flow Problem is TU.
- If the demands b_i and the capacities h_{ij} are integral, then each extreme point is integral. The constraints describe the convex hull of the integer feasible flow.

- Shortest path. Given two nodes $s, t \in V$ and nonnegative arc costs c_{ij} for $(i,j) \in A$, find a minimum cost flow from s to t.

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = 1, \ i = s \\
& \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = 0, \ i \in V \setminus \{s,t\} \\
& \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = -1, \ i = t \\
& x_{ij} \in \mathbb{Z}_+, \ (i,j) \in A.
\end{aligned}$$

- Theorem: z is the length of a shortest $s-t$ path iff there exist π_i for $i \in V$ such that $\pi_s = 0$, $\pi_t = z$ and $\pi_j - \pi_i \leq c_{ij}$ for $(i,j) \in A$.

- Proof. The dual of the shortest path problem is:

$$\begin{aligned} &\max\ \pi_t - \pi_s \\ &\text{s.t. } \pi_j - \pi_i \le c_{ij},\ (i,j) \in A. \end{aligned}$$

  Replacing π_j by $\pi_j + \alpha$ for all $j \in V$ does not change the dual value, we can fix $\pi_s = 0$. The theorem is true by the strong duality (because of TU).
- Maximum flow problem. Given two nodes $s, t \in V$ and nonnegative arc capacities h_{ij} for $(i,j) \in A$, find a maximum flow from s to t.
- Adding a backward arc from t to s.

$$\begin{aligned} &\max\ x_{ts} \\ &\text{s.t.} \quad \sum_{k \in V_i^+} x_{ik} - \sum_{k \in V_i^-} x_{ki} = 0,\ i \in V, \\ &\qquad 0 \le x_{ij} \le h_{ij}, (i,j) \in A, \end{aligned}$$

  where $h_{ts} = +\infty$.

- The dual is:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} h_{ij} w_{ij} \\ \text{s.t.} \quad & u_i - u_j + w_{ij} \geq 0, \ (i,j) \in A, \\ & u_t - u_s \geq 1, \ w_{ij} \geq 0, \quad (i,j) \in A. \end{aligned}$$

- The dual optimal solution is also integral because the constraint matrix is TU. Since replacing u_j by $u_j + \alpha$ does not change the dual, we can assume $u_s = 0$. Given such a solution (u, w), let

$$X = \{j \in V \mid u_j \leq 0\}, \quad \tilde{X} = V \setminus X = \{j \in V \mid u_j \geq 1\}.$$

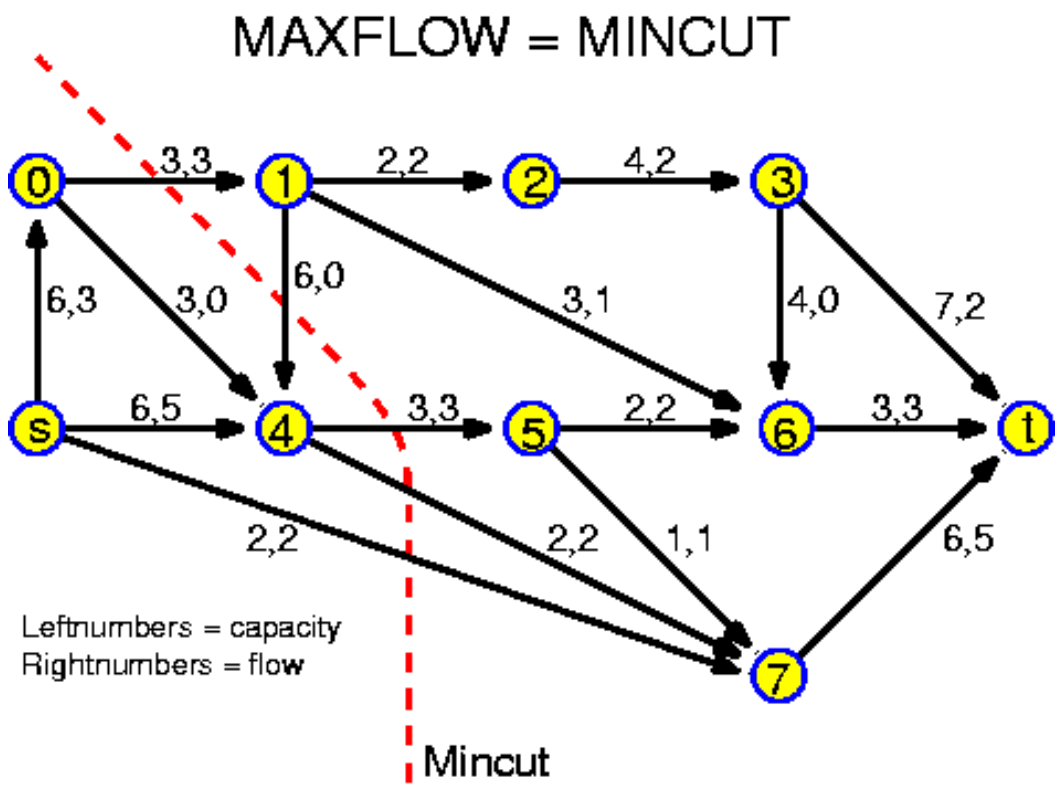
- Obviously, $s \in X$ and $t \in \tilde{X}$. When $i \in X, j \in \bar{X}$, we have $w_{ij} \geq u_j - u_i \geq 1$. So

$$\sum_{(i,j)\in A} h_{ij} w_{ij} \geq \sum_{(i,j)\in A, i\in X, j\in \bar{X}} h_{ij} w_{ij} \geq \sum_{(i,j)\in A, i\in X, j\in \bar{X}} h_{ij}.$$

- The lower bound is attained at the solution $u_j = 0$ for $j \in X$ and $u_j = 1$ for $j \in \tilde{X}$, $w_{ij} = 1$ for $(i,j) \in A$ and $i \in X, j \in \tilde{X}$.

- Theorem: The dual to the maximum $s-t$ flow problem is the minimum $s-t$ cut problem:

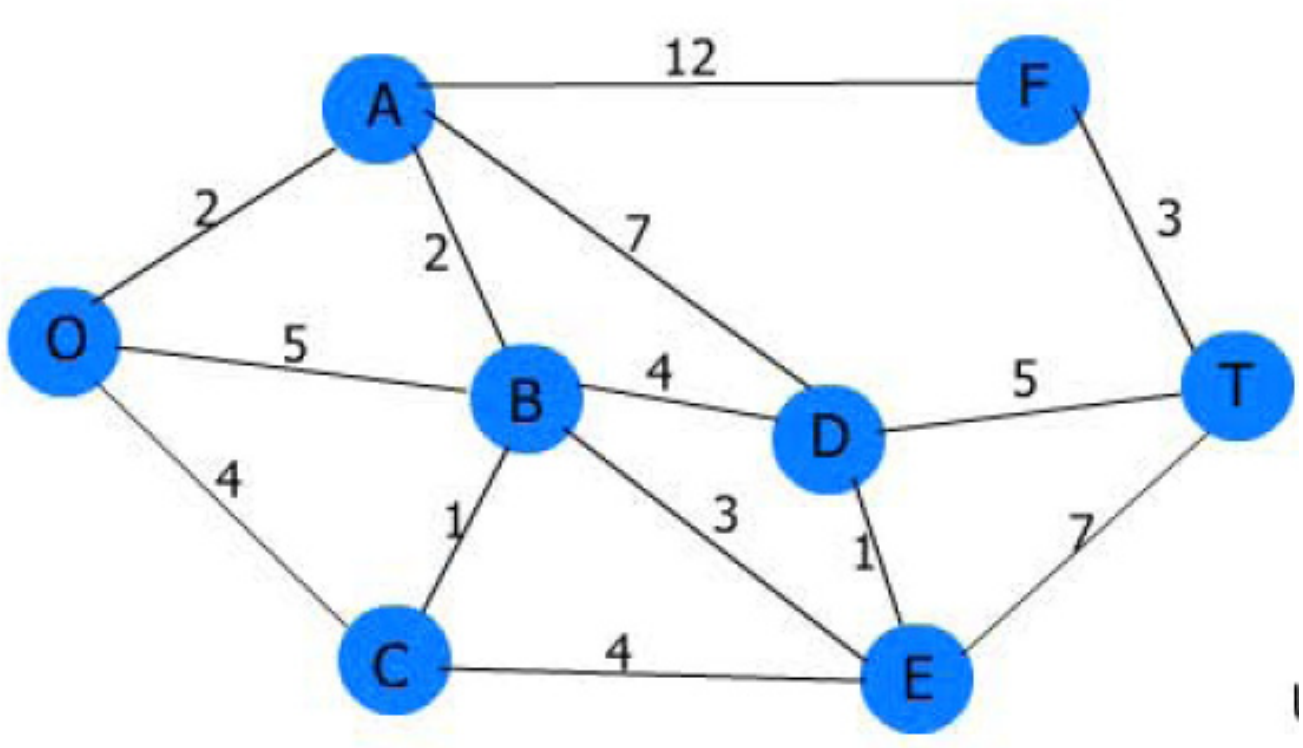
$$\min_{X} \sum_{(i,j)\in A, i\in X, j\in \tilde{X}} \{h_{ij} \mid s \in X \subset V \setminus \{t\}\}.$$



Dijkstra Algorithm for shortest path problem

- Given digraph $G = (V, A)$. For edge $e = (i, j)$, $w_e \geq 0$ is the cost of the direct route between nodes i and j.
- Problem: find shortest paths from a starting point to all other nodes.
- Suppose k is an intermediate node on a shortest $1 - i$ path p_i. Then the $1 - k$ subpath p_k of p_i is a shortest $1 - k$ path.
- Dijkstra's shortest path algorithm
  - Step 1(Initialization): $g(1) = 0$, $U = \{1\}$, $h(j) = w_{1j}$ if $(1, j) \in A$, $h(j) = \infty$ otherwise.
  - Step 2: Let $i = \arg\min_{j \in V \setminus U} h(j)$. Set $U \leftarrow U \cup \{i\}$ and $g(i) = h(i)$. If $U = V$. stop.
  - Step 3: For all $j \in U$ with $(i, j) \in A$, $h(j) \leftarrow \min(g(i) + w_{ij}, h(j))$. Return to Step 2.
- Complexity: $O(|V|^2)$.

- An example with 8 nodes:



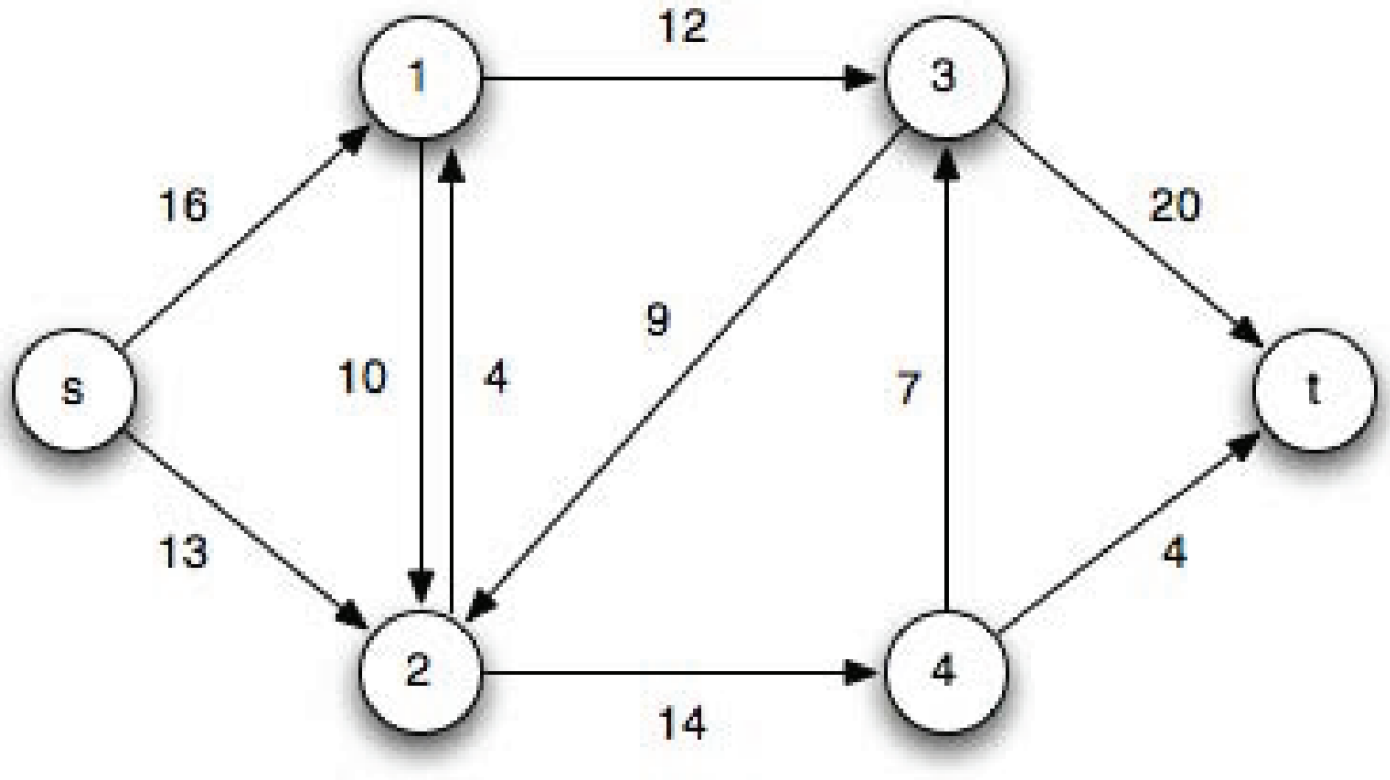
- The iteration process of Dijkstra algorithm is illustrated as follows.

Ford-Fulkerson Algorithm for Max-flow Problem

- Ford-Fulkerson algorithm is also called Augmentation Path algorithm. This is an iterative method. At each iteration, the algorithm is searching for a path from the source node to the sink node along which it can send a positive flow. Such a path is referred to as augmenting path.
- After a flow is sent along an augmenting path the capacities of the links on that path are adjusted. These adjusted capacities are referred to as residual capacities. Correspondingly, the resulting network is referred to as residual network.
- The algorithm terminates when an augmenting path cannot be found

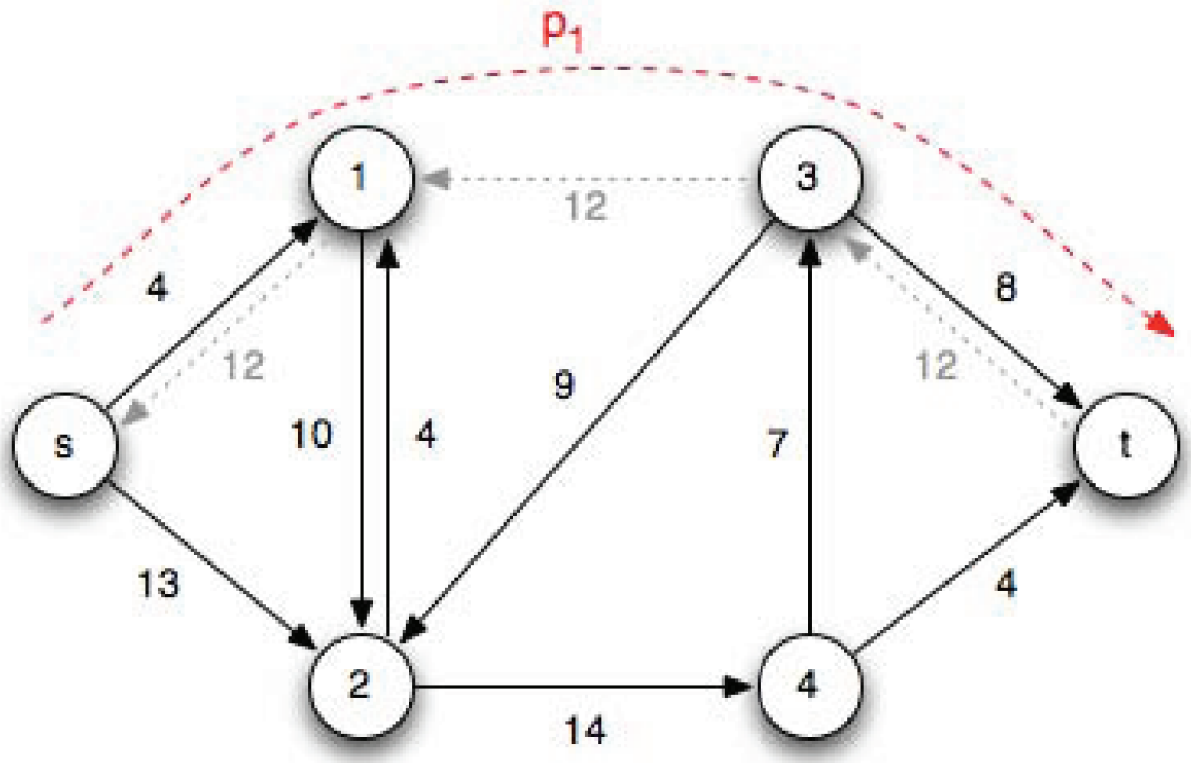
A max-flow example

- Consider the following network:

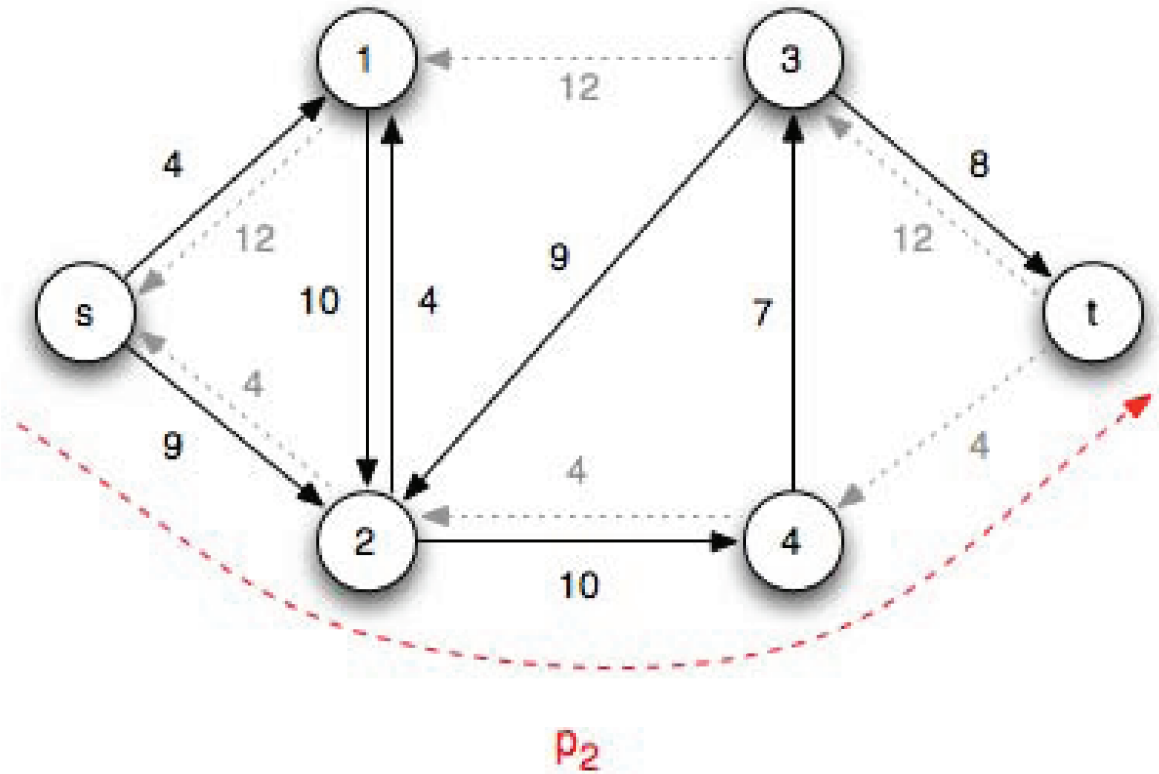


- Note that clearly $|f^*| \leq 24$ (the smaller of the capacities leaving the source or entering the sink).

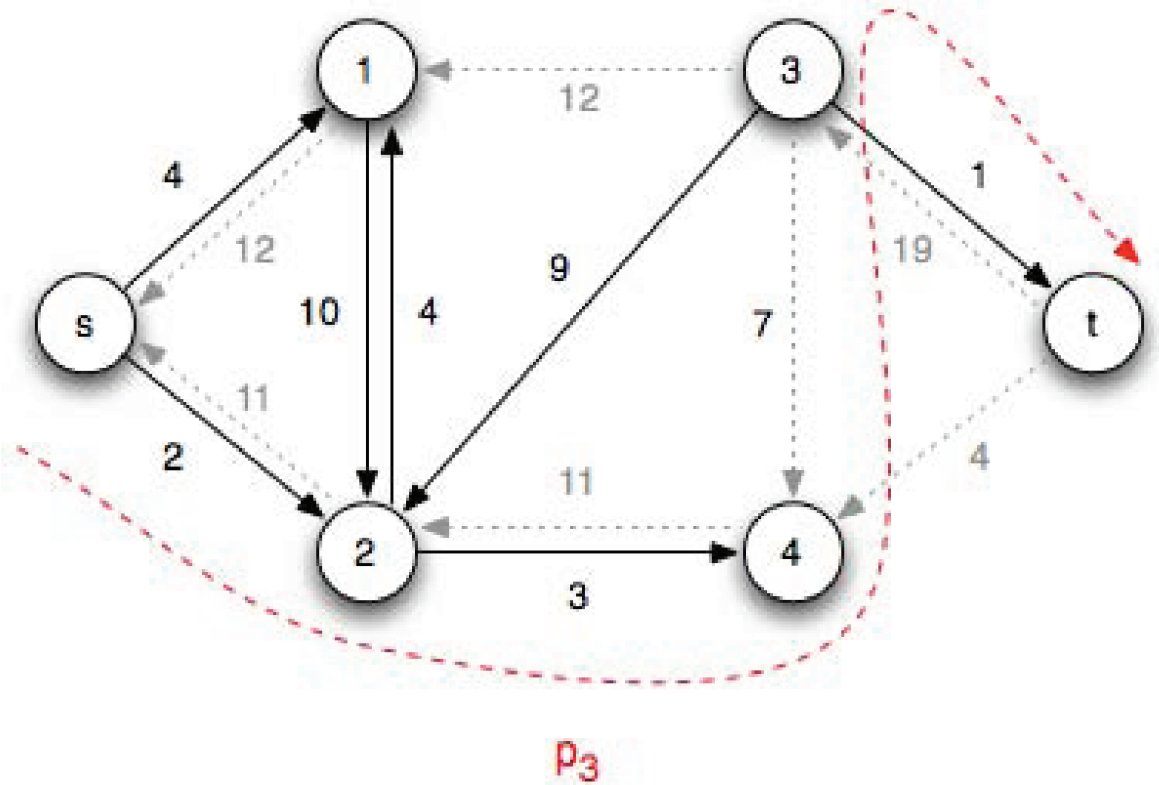
- Iteration 1: Choose the augmenting path $p_1 =< s, 1, 3, t >$ which has $c_f(p_1) = 12$ (due to $c(1, 3)$) giving the residual network:



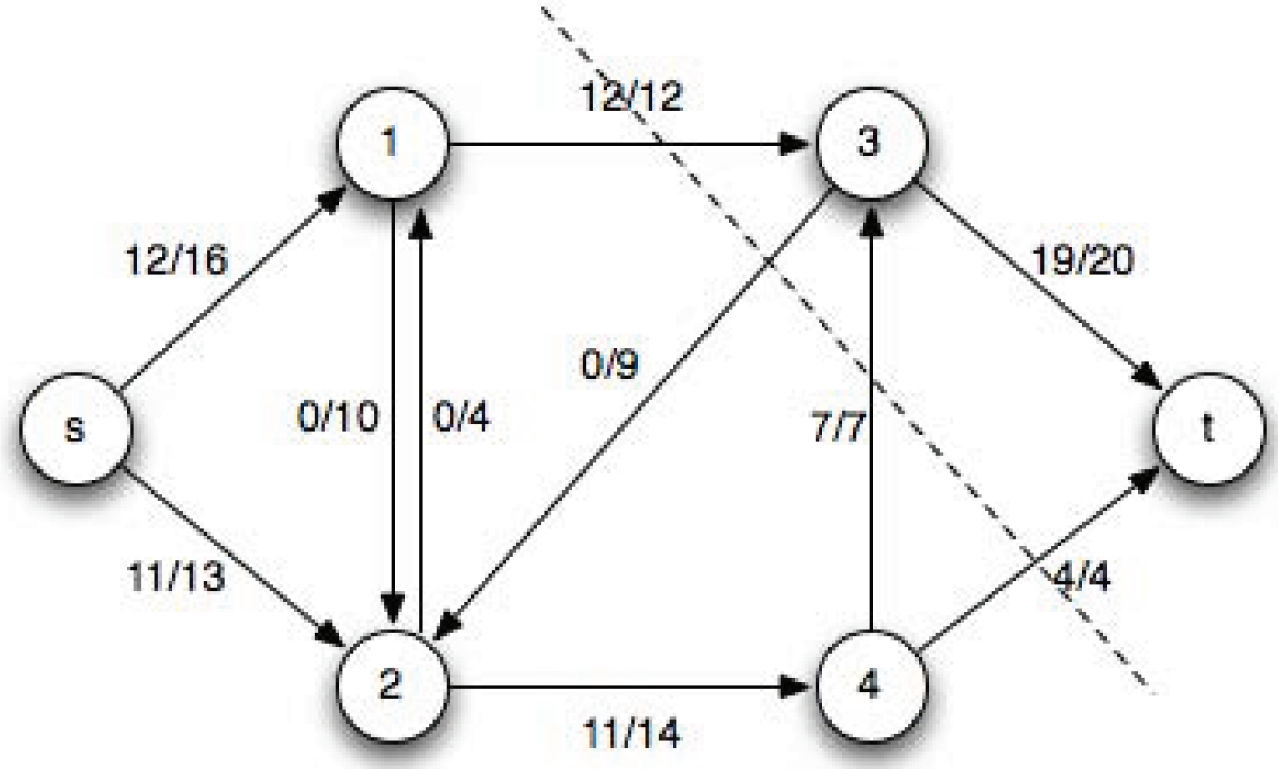
- Iteration 2: Choose the augmenting path $p_2 =< s, 2, 4, t >$ which has $c_f(p_2) = 4$ (due to $c(4, t)$) giving the residual network:



- Iteration 3: Choose the augmenting path $p_3 =< s, 2, 4, 3, t >$ which has $c_f(p_3) = 7$ (due to $c(4, 3)$) giving the residual network:



- At this point there are no other augmenting paths (since vertex 3 is the only vertex with additional capacity to the sink but has no flow in from other vertices). Hence the max-flow network (=min-cut) is



The maximal flow is $|f^*| = 19 + 4 = 23$.

Minimum spanning tree

- A tree is a connected graph without cycles.
- Properties of trees:
  - A graph is a tree if and only if there is one and only one path joining any two of its vertices.
  - A connected graph is a tree if and only if every one of its edges is a bridge.
  - A connected graph is a tree if and only if it has N vertices and N-1 edges.
- A subgraph that spans (reaches out to) all vertices of a graph is called a spanning subgraph.
- A subgraph that is a tree and that spans (reaches out to) all vertices of the original graph is called a spanning tree.
- Among all the spanning trees of a weighted and connected graph, the one (possibly more) with the least total weight is called a minimum spanning tree (MST).
- The minimum spanning tree of a graph defines the cheapest subset of edges that keeps the graph in one connected component.

Kruskal's Algorithm

Step 1 Find the cheapest edge in the graph (if there is more than one, pick one at random). Mark it with any given colour, say red.

Step 2 Find the cheapest unmarked (uncoloured) edge in the graph that doesn't close a red circuit. Mark this edge red.

Step 3 Repeat Step 2 until you reach out to every vertex of the graph (or you have $N - 1$ coloured edges, where N is the number of vertices.) The red edges form the desired minimum spanning tree.

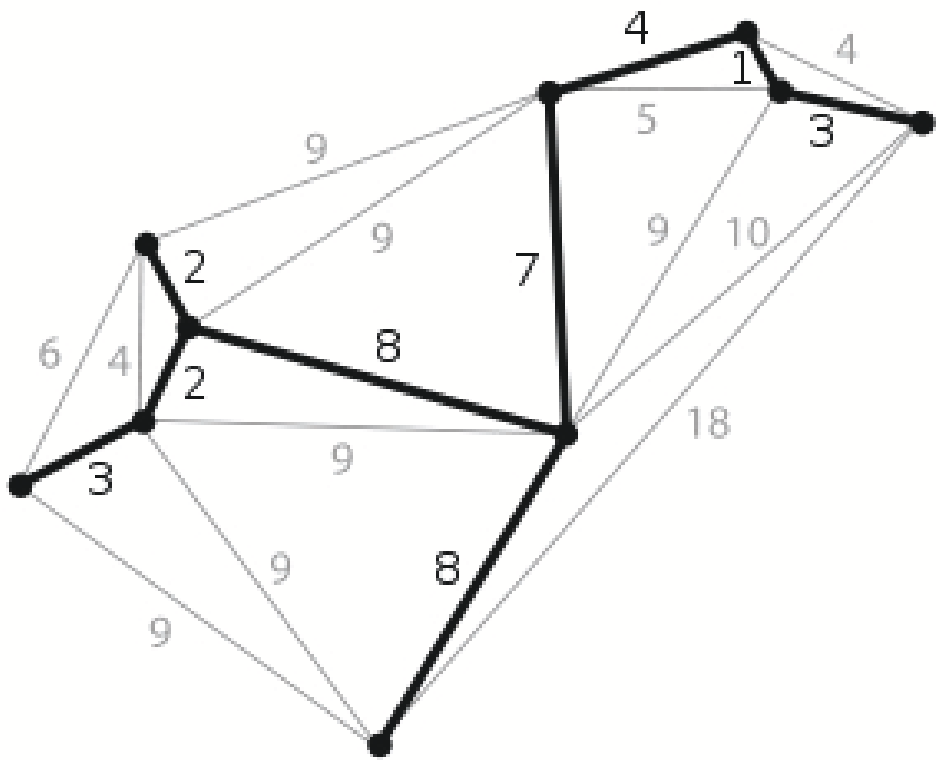


Figure: Minimum spanning tree

Minimum Steiner tree

- minimum Steiner Tree: Given a graph $G = (V, E)$, and a subset of vertices $T \subseteq V$ as terminals. Find the smallest tree connecting all the vertices of T.
- Suppose we are given a set of sites that must be connected by wires as cheaply as possible. The minimum Steiner tree is to connect them using the smallest amount of wire. Other examples: networks of water pipes, heating ducts in buildings, VLSI circuit layout.
- The difference between the Steiner tree problem and the minimum spanning tree problem is that, in the Steiner tree problem, extra intermediate vertices and edges may be added to the graph in order to reduce the length of the spanning tree. These new vertices introduced to decrease the total length of connection are known as Steiner points or Steiner vertices.

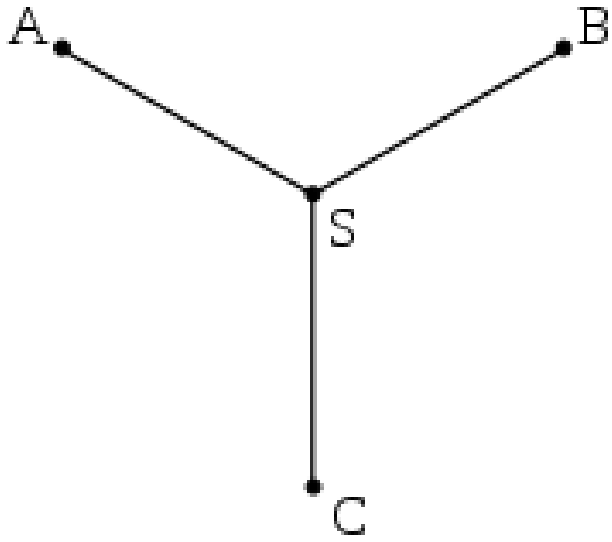
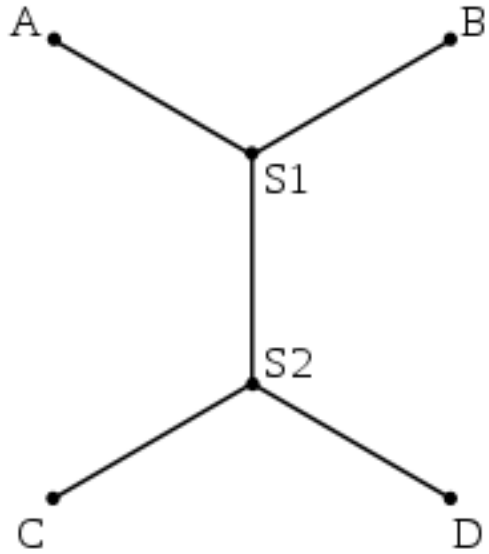


Figure: Solution to Steiner tree tree with 3 points



- Steiner tree often arises in network design and wiring layout problems.
- $|T| = 2 \Leftrightarrow$ shortest path, $|T| = |E| \Leftrightarrow$ optimal tree.
- In general minimum weight steiner tree problem is NP-hard.
- Steiner ratio conjecture: Let P be a set of n points on the Euclidean plane. Let $L_s(P)$ and $L_m(P)$ denote the lengths of the Steiner minimum tree and the minimum spanning tree on P, respectively. In 1968, Gilbert and Pollak conjectured that:

$$\frac{L_m(p)}{L_s(P)} \leq \frac{2}{\sqrt{3}}.$$

- Du and Hwang's proof (Proc. Nat. Acad. Sci., 1990; Algorithmica, 1992).
- N. Innami, B. H. Kim, Y. Mashiko and K. Shiohama, The Steiner Ratio Conjecture of Gilbert-Pollak May Still Be Open. Algorithmica, 2008.