Lecture: Network Flow Problems and Combinatorial Optimization Problems

http://bicmr.pku.edu.cn/~wenzw/bigdata2016.html

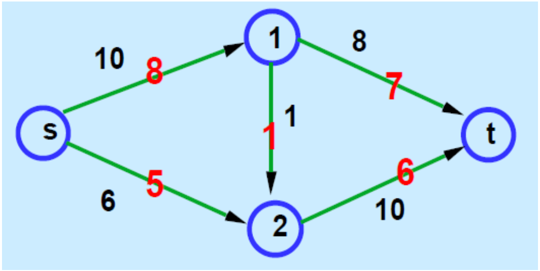
Acknowledgement: this slides is based on Prof. James B. Orlin's lecture notes of "15.082/6.855J, Introduction to Network Optimization" at MIT, as well as Prof. Shaddin Dughmi and Prof. Sewoong Oh's lecture notes

Textbook: Network Flows: Theory, Algorithms, and Applications by Ahuja, Magnanti, and Orlin referred to as AMO

Outline

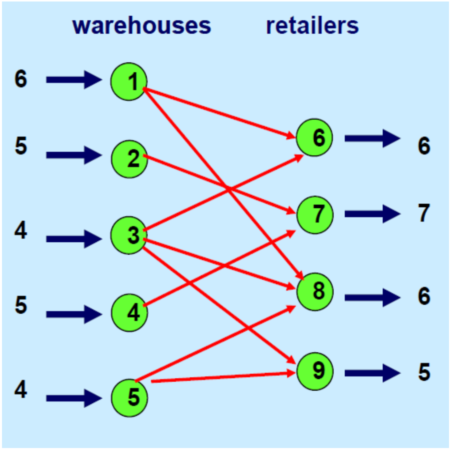
Maximum Flows

We refer to a flow x as maximum if it is feasible and maximizes v. Our objective in the max flow problem is to find a maximum flow.



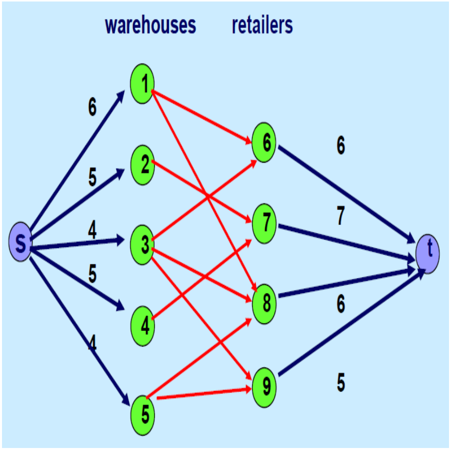
A max flow problem. Capacities and a non- optimum flow.

The feasibility problem: find a feasible flow



Is there a way of shipping from the warehouses to the retailers to satisfy demand?

The feasibility problem: find a feasible flow



There is a 1-1 correspondence with flows from s to t with 24 units (why 24?) and feasible flows for the transportation problem.

The Max Flow Problem

- G = (N,A)
- x_{ij} = flow on arc (i,j)
- u_{ij} = capacity of flow in arc (i,j)
- s = source node
- t = sink node

$$\begin{aligned}
\max \quad & v \\
\text{s.t.} \quad & \sum_j x_{sj} = v \\
& \sum_j x_{ij} - \sum_j x_{ji} = 0, \ \text{for each i } \neq s \text{ or } t \\
& -\sum_i x_{it} = -v \\
& 0 \leq x_{ij} \leq u_{ij} \text{ for all } (i,j) \in A
\end{aligned}$$

Dual of the Max Flow Problem

reformulation:

- $A_{i,(i,j)} = 1, A_{j,(i,j)} = -1$, for $(i,j) \in A$ and all other elements are 0
- $A^\top y = y_i - y_j$

The primal-dual pair is

$$\begin{aligned} \min \quad & (\mathbf{0}, -1)(x, v)^\top \\ \text{s.t. } & Ax + (-1, \mathbf{0}, 1)^\top v = 0 \\ & Ix + \mathbf{0}^\top v \le u \\ & x \ge 0, v \text{ is free} \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} \max \quad & -u^\top \pi \\ \text{s.t. } & A^\top y + I^\top \pi \ge 0 \\ & -1 + (-1, \mathbf{0}, 1)y = 0 \\ & \pi \ge 0 \end{aligned}$$

Hence, we have the dual problem:

$$\begin{aligned} \min \quad & u^\top \pi \\ \text{s.t. } & y_j - y_i \le \pi_{ij}, \quad \forall (i,j) \in A \\ & y_t - y_s = 1 \\ & \pi \ge 0 \end{aligned}$$

Duality of the Max Flow Problem

The primal-dual of the max flow problem is

$$\begin{aligned}
\max \quad & v \\
\text{s.t.} \quad & \sum_j x_{sj} = v \\
& \sum_j x_{ij} - \sum_j x_{ji} = 0, \forall i \notin \{s,t\} \\
& -\sum_i x_{it} = -v \\
& 0 \le x_{ij} \le u_{ij} \quad \forall (i,j) \in A
\end{aligned}$$

$$\begin{aligned}
\min \quad & u^\top \pi \\
\text{s.t.} \quad & y_j - y_i \le \pi_{ij}, \quad \forall (i,j) \in A \\
& y_t - y_s = 1 \\
& \pi \ge 0
\end{aligned}$$

Duality of the Max Flow Problem

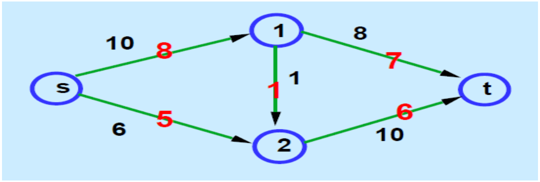
- Dual solution describes fraction π_{ij} of each edge to fractionally cut
- Dual constraints require that at least 1 edge is cut on every path P from s to t.

$$\sum_{(i,j)\in P} \pi_{ij} \geq \sum_{(i,j)\in P} y_j - y_i = y_t - y_s = 1$$

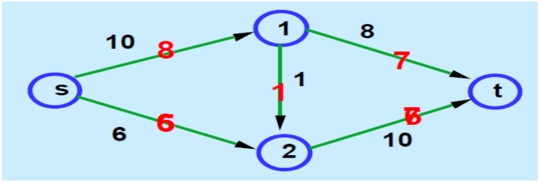
- Every integral s-t cut (A,B) is feasible:
  $\pi_{ij} = 1, \forall i \in A, j \in B$, otherwise, $\pi_{ij} = 0$.
  $y_i = 0$ if $i \in A$ and $y_j = 1$ if $i \in B$
- weak duality: $v \leq u^\top \pi$ for any feasible solution
  max flow $\leq$ minimum flow
- strong duality: $v^* = u^\top \pi^*$ at the optimal solution

Outline

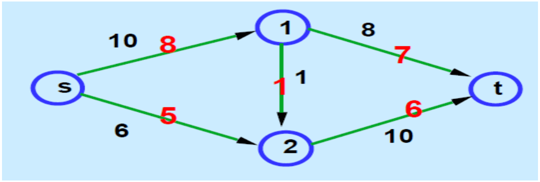
sending flows along s-t paths



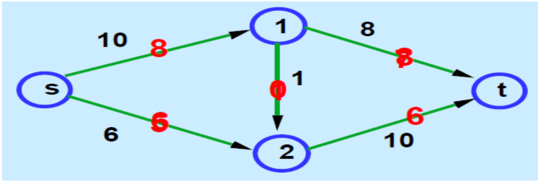
One can find a larger flow from s to t by sending 1 unit of flow along the path s-2-t



A different kind of path

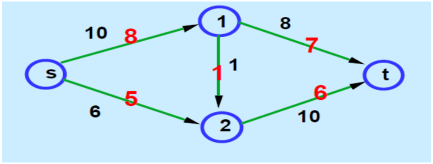


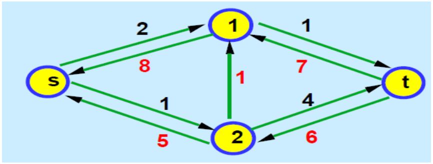
One could also find a larger flow from s to t by sending 1 unit of flow along the path s-2-1-t. (Backward arcs have their flow decreased.)



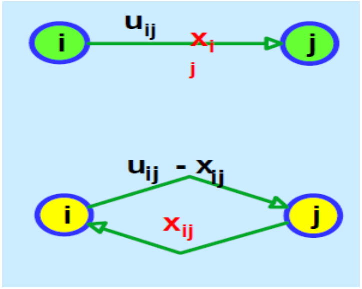
Decreasing flow in (1, 2) is mathematically equivalent to sending flow in (2, 1) w.r.t. node balance constraints.

The Residual Network





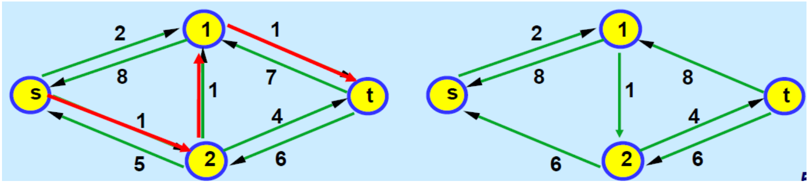
The Residual Network G(x)



We let r_{ij} denote the residual capacity of arc (i,j)

A Useful Idea: Augmenting Paths

- An augmenting path is a path from s to t in the residual network.
- The residual capacity of the augmenting path P is $\delta(P) = \min\{r_{ij} : (i,j) \in P\}$.
- To augment along P is to send $\delta(P)$ units of flow along each arc of the path. We modify x and the residual capacities appropriately.
- $r_{ij} := r_{ij} - \delta(P)$ and $r_{ji} := r_{ji} + \delta(P)$ for (i,j) $\in$P.



The Ford Fulkerson Maximum Flow Algorithm

- x := 0;
  create the residual network G(x);
- while there is some directed path from s to t in G(x) do
  - let P be a path from s to t in G(x);
  - $\delta := \delta(P) = \min\{r_{ij} : (i,j) \in P\}$;
  - send δ-units of flow along P;
  - update the r's:
    $r_{ij} := r_{ij} - \delta(P)$ and $r_{ji} := r_{ji} + \delta(P)$ for (i,j) $\in$P.

To prove correctness of the algorithm

- Invariant: at each iteration, there is a feasible flow from s to t.
- Finiteness (assuming capacities are integral and finite):
  - The residual capacities are always integer valued
  - The residual capacities out of node s decrease by at least one after each update.
- Correctness
  - If there is no augmenting path, then the flow must be maximum.
  - max-flow min-cut theorem.

Integrality

Assume that all data are integral.

Lemma: At each iteration all residual capacities are integral.

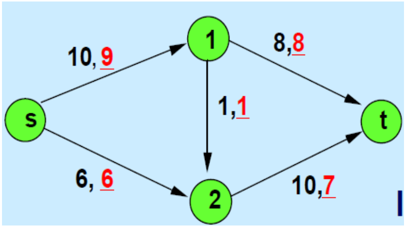
- Proof. It is true at the beginning. Assume it is true after the first k-1 augmentations, and consider augmentation k along path P.
- The residual capacity δ of P is the smallest residual capacity on P, which is integral.
- After updating, we modify residual capacities by 0, or δ, and thus residual capacities stay integral.

Theorem. The Ford-Fulkerson Algorithm is finite

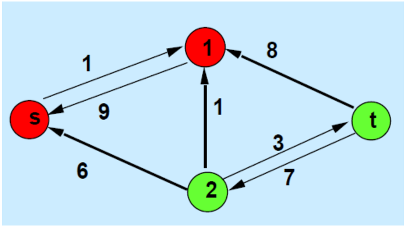
- Proof. The capacity of each augmenting path is at least 1.
- r_{sj} decreases for some j.
- So, the sum of the residual capacities of arcs out of s decreases at least 1 per iteration.
- Number of augmentations is O(nU), where U is the largest capacity in the network.

To be proved: If there is no augmenting path, then the flow is maximum

- G(x) = residual network for flow x.
- x* = final flow
- If there is a directed path from i to j in G, we write $i \to j$.

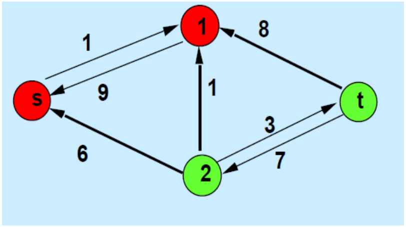


- S* = { j : $s \to j$ in G(x*)}
- T*=N$\setminus$S*

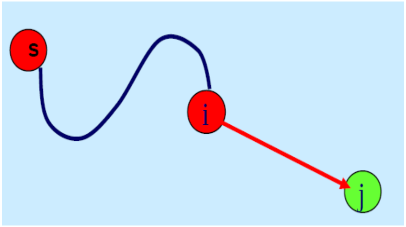


Lemma: there is no arc in G(x*) from S* to T*

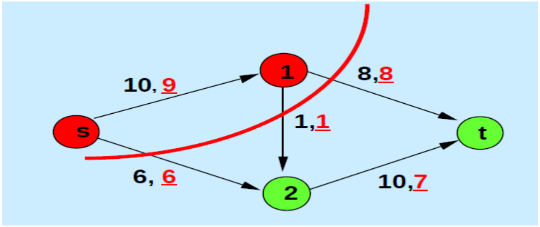
- S* = { j : $s \rightarrow j$ in G(x*)}
- T*=N$\setminus$S*



Proof: If there were such an arc (i,j) and $i \in S^*$, then j would be in S*.



Cut Duality Theory



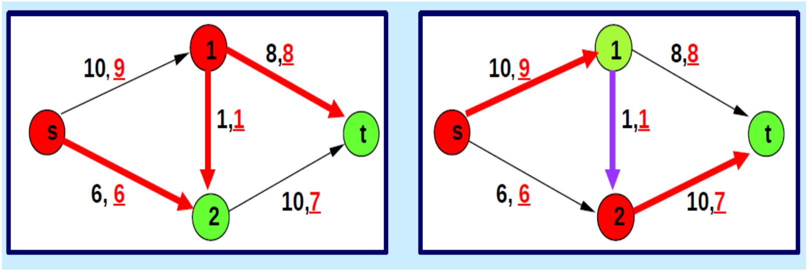
- An (s,t)-cut in a network G = (N,A) is a partition of N into two disjoint subsets S and T such that s $\in$ S and t $\in$T, e.g., S = {s, 1} and T = {2, t}.
- The capacity of a cut (S,T) is

$$\text{cut(S,T)} = \sum_{i \in S} \sum_{j \in T} u_{ij}$$

The flow across a cut

We define the flow across the cut (S,T) to be

$$F_x(S,T) = \sum_{i\in S}\sum_{j\in T} x_{ij} - \sum_{i\in S}\sum_{j\in T} x_{ji}$$



- If S = {s, 1}, then $F_x(S,T)$ = 6 + 1 + 8 = 15
- If S = {s, 2}, then $F_x(S,T)$ = 9 - 1 + 7 = 15

Max Flow Min Cut

Theorem. (Max-flow Min-Cut). The maximum flow value is the minimum value of a cut.

- Proof. The proof will rely on the following three lemmas:
- Lemma 1. For any flow x, and for any s-t cut (S, T), the flow out of s equals $F_x(S,T)$.
- Lemma 2. For any flow x, and for any s-t cut (S, T), $F_x(S,T) \leq \text{cut}(S,T)$.
- Lemma 3. Suppose that x* is a feasible s-t flow with no augmenting path. Let S* = {j : s $\rightarrow$j in G(x*)} and let T* = N$\backslash$S. Then $F_{x^*}(S^*,T^*) = \text{cut}(S^*,T^*)$.

Proof of Theorem (using the 3 lemmas)

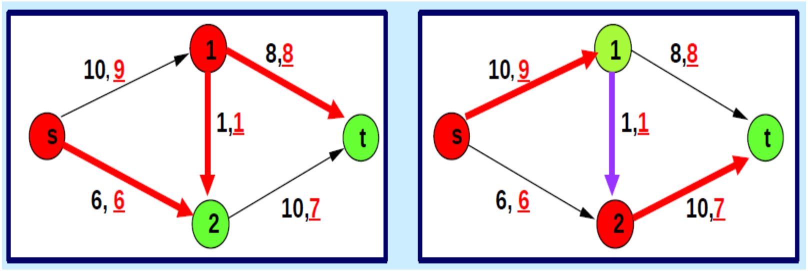
- Let x' be a maximum flow
- Let v' be the maximum flow value
- Let x* be the final flow.
- Let v* be the flow out of node s (for x*)
- Let S* be nodes reachable in G(x*) from s.
- Let T* = N$\setminus$S*.

1. v* $\leq$ v', by definition of v'
2. v' = $F_{x'}$ (S*, T*), by Lemma 1.
3. $F_{x'}$ (S*, T*) $\leq$ cut(S*, T*) by Lemma 2.
4. v* = F_{x^*} (S*, T*) = cut(S*, T*) by Lemmas 1,3.

Thus all inequalities are equalities and v* = v'.

Proof of Lemma 1

Proof. Add the conservation of flow constraints for each node i∈S - {s} to the constraint that the flow leaving s is v. The resulting equality is F_x (S,T) = v.

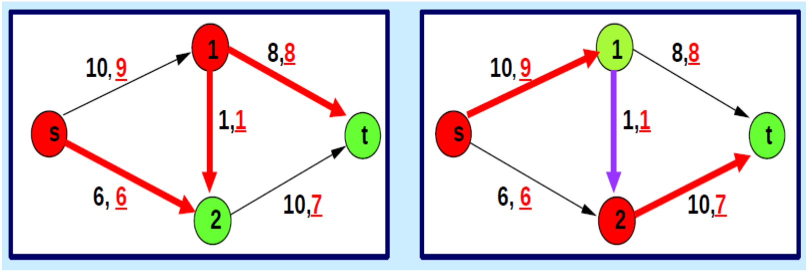


- $x_{s1} + x_{s2} = v, x_{12} + x_{1t} - x_{s1} = 0 \Longrightarrow x_{s2} + x_{12} + x_{1t} = v$
- $x_{s1} + x_{s2} = v, x_{2t} - x_{s2} - x_{12} = 0 \Longrightarrow x_{s1} - x_{12} + x_{2t} = v$

Proof of Lemma 2

Proof. If i $\in$S, and j $\in$T, then $x_{ij} \leq u_{ij}$. If i $\in$T, and j $\in$S, then $x_{ij} \geq 0$.

$$\begin{aligned} F_x(S,T) &= \sum_{i\in S}\sum_{j\in T} x_{ij} - \sum_{i\in S}\sum_{j\in T} x_{ji} \\ \text{cut}(S,T) &= \sum_{i\in S}\sum_{j\in T} u_{ij} - \sum_{i\in S}\sum_{j\in T} 0 \end{aligned}$$

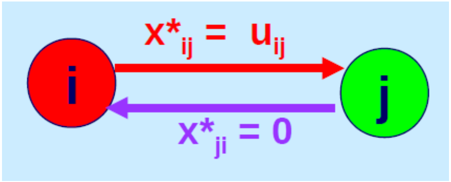


- left: F_x(S,T)=15, cut(S, T) = 15
- right: F_x(S,T)=15, cut(S, T) = 20

Proof of Lemma 3.

We have already seen that there is no arc from S* to T* in G(x*).

$$i \in S^* \text{ and } j \in T^* \implies x^*_{ij} = u_{ij} \text{ and } x^*_{ji} = 0$$



- Otherwise, there is an arc (i, j) in G(x*)
- Therefore F_{x^*} (S*, T*) = cut(S*, T*)

Review

- Corollary. If the capacities are finite integers, then the Ford-Fulkerson Augmenting Path Algorithm terminates in finite time with a maximum flow from s to t.
- Corollary. If the capacities are finite rational numbers, then the Ford-Fulkerson Augmenting Path Algorithm terminates in finite time with a maximum flow from s to t. (why?)
- Corollary. To obtain a minimum cut from a maximum flow x*, let S* denote all nodes reachable from s in G(x*), and T* = N$\setminus$S*
- Remark. This does not establish finiteness if $u_{ij} = \infty$ or if capacities may be irrational.

Speedups of the augmenting path algorithm

- Shortest augmenting path algorithm: always augment along the path in G(x) with the fewest number of arcs.
- Largest augmenting path algorithm: always augment along a path in G(x) with the greatest capacity.

The shortest augmenting path algorithm

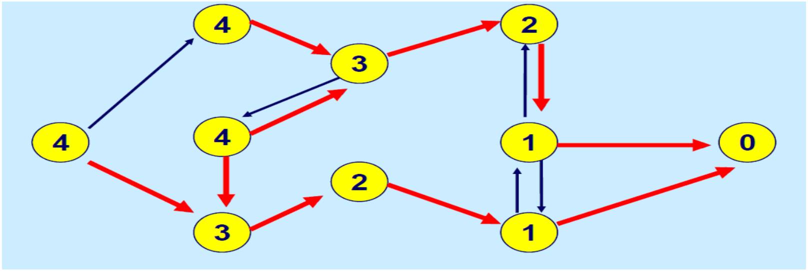
- x := 0;
  create the residual network G(x);
- while there is some directed path from s to t in G(x) do
  - let P be a path from s to t in G(x) with the fewest number of arcs;
  - $\delta := \delta(P) = \min\{r_{ij} : (i,j) \in P\}$;
  - send δ-units of flow along P;
  - update the r's:
    $r_{ij} := r_{ij} - \delta(P)$ and $r_{ji} := r_{ji} + \delta(P)$ for (i,j) $\in$P.

Theorem. The shortest augmenting path algorithm determines a maximum flow in O(nm) augmentations.

This algorithm can be implemented to run in $O(n^2m)$ time.

Distance Labels

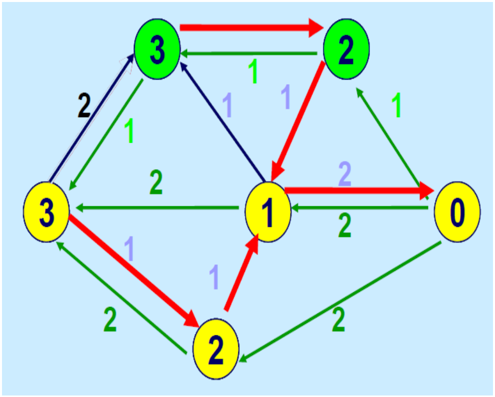
Let d(i) be the length of the shortest path from i to t in G(x)



- FACT 1: If (i, j)$\in$G(x), then d(i) $\leq$ d(j) + 1.
- FACT 2: Arc (i, j) is on a shortest path from i to t if and only if d(i) = d(j) + 1.
- FACT 3: d(t) = 0; d(i) < n for all i s.t. i $\rightarrow$ t in G(x).

Valid Arcs and Saturating Pushes

- An arc (i, j) $\in$ G(x) is valid if d(i) = d(j) + 1.
- FACT: If P is an augmenting path, then every arc of P is valid.



Suppose δ units of flow are sent along P. The augmentation saturates arc (i, j) $\in$ P if $r_{ij} = \delta$.

The number of augmenting paths

Theorem. The number of augmenting paths for the shortest augmenting path algorithm is O(nm).

Fact. In every augmenting path, at least one arc (i, j) is saturated.

Lemma 1. Arc (i, j) and its reversal (j, i) can be saturated at most n/2 times each. (To be proved later.)

Proof of theorem. Let a_{ij} be the number of times that arc (i, j) is saturated. Let k be the number of augmentations.

Then $k \leq \sum_{(i,j) \in A} 2a_{ij} \leq \sum_{(i,j) \in A} n \leq nm$.

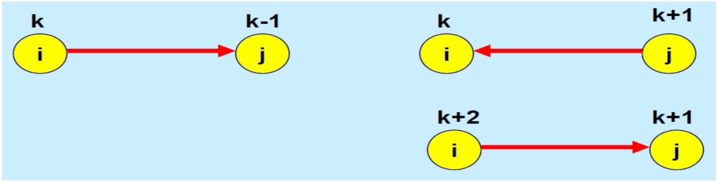
Proof of Lemma 1.

Each arc (i, j) is saturated fewer than n/2 times.

- Lemma 2. Let d be the distance labels at an iteration where arc (i, j) is saturated. Suppose that d' is the vector of distance labels at a subsequent iteration where (i, j) is saturated. Then $n > d'(i) \geq d(i) + 2$. (to be proved on next slide).
- Proof of Lemma 1 from Lemma 2. Before (i, j) can be saturated again, its distance label will increase by at least 2. Since $0 < d(i) < n$, the distance label can increase at most n/2 times.

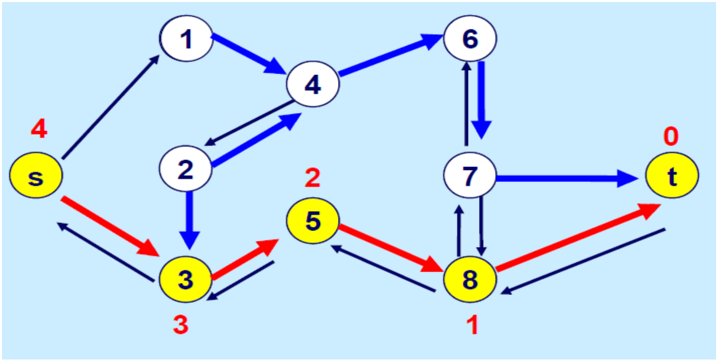
Proof of Lemma 2.

- Lemma 3. Let d be the distance labels at some iteration, and let d' be the distance labels at a subsequent iteration. Then d'(i) $\geq$ d(i) for all i$\in$ N. (to be proved on next slide)
- Proof of Lemma 2 from Lemma 3. Suppose that (i, j) is saturated when d(i) = k. There is no more flow in (i, j) until flow is sent in (j, i) at which point the distance label of j is k+1. But flow cannot be returned in (i, j) until it is valid again, and the distance label is at least k+2.



Proof of Lemma 3

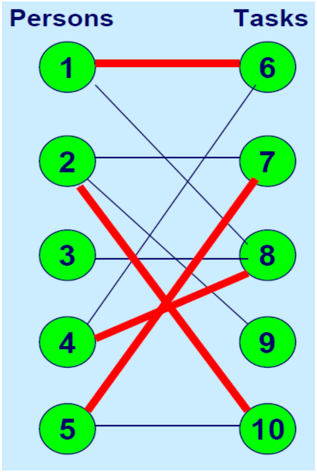
Assume that Lemma 3 is false. Let d be the distance labels at some iteration. Let P be the augmenting path. After the augmentation, the reversals of arcs in P are in the residual network. But adding these arcs to G(x) does not decrease any distance. And deleting arcs of P cannot decrease a distance label.



Outline

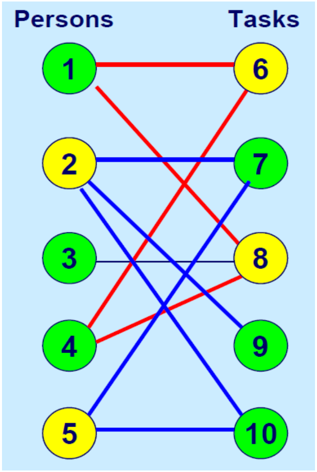
Matchings

- An undirected network G = (N, A) is bipartite if N can be partitioned into N1 and N2 so that for every arc (i,j), i $\in$ N1 and j $\in$ N2.
- A matching in N is a set of arcs no two of which are incident to a common node.
- Matching Problem: Find a matching of maximum cardinality



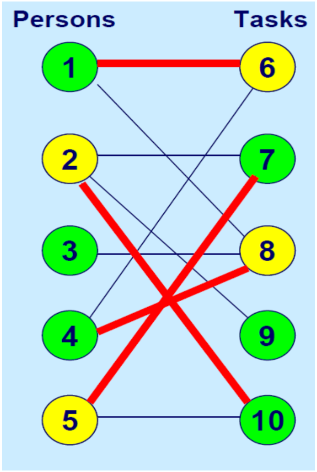
Node Covers

- A node cover is a subset S of nodes such that each arc of G is incident to a node of S.
- Node Cover Problem: Find a node cover of minimum cardinality.

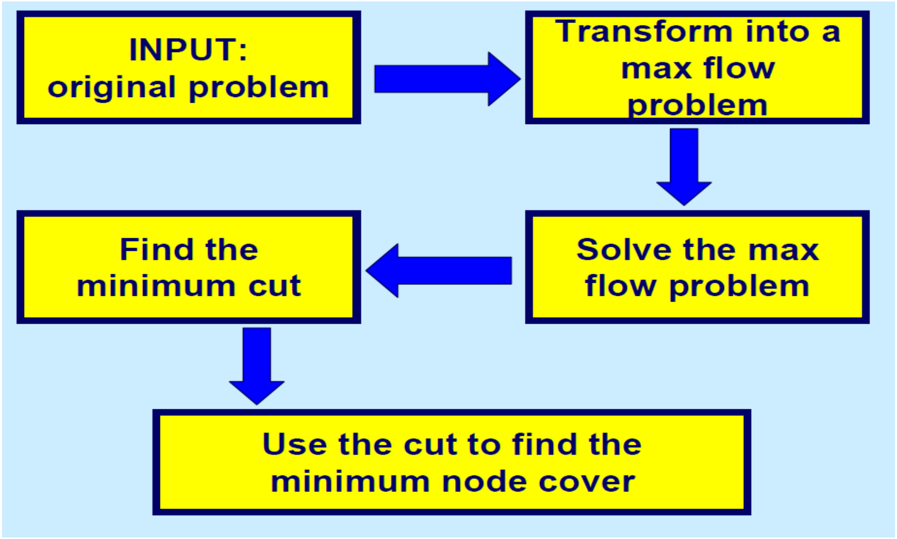


Matching Duality Theorem

- Theorem. König- Egerváry. The maximum cardinality of a matching is equal to the minimum cardinality of a node cover.
- Note. Every node cover has at least as many nodes as any matching because each matched edge is incident to a different node of the node cover.

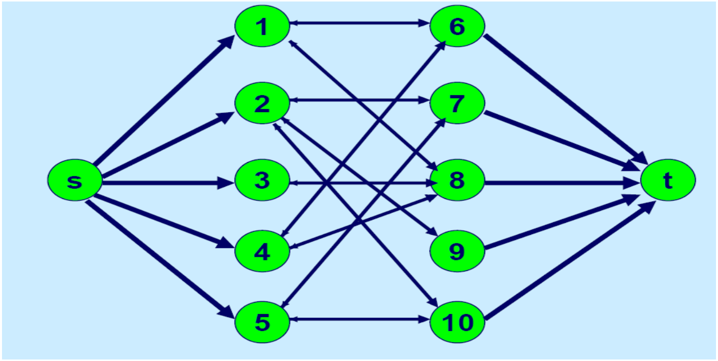


How to find a minimum node cover


INPUT: original problem
Transform into a max flow problem
Solve the max flow problem
Find the minimum cut
Use the cut to find the minimum node cover

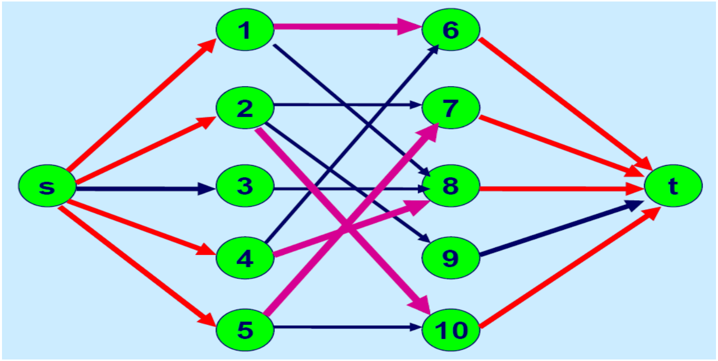
Matching-Max Flow

Solving the Matching Problem as a Max Flow Problem



- Replace original arcs by directed arcs with infinite capacity.
- Each arc (s, i) has a capacity of 1.
- Each arc (j, t) has a capacity of 1.

Find a Max Flow

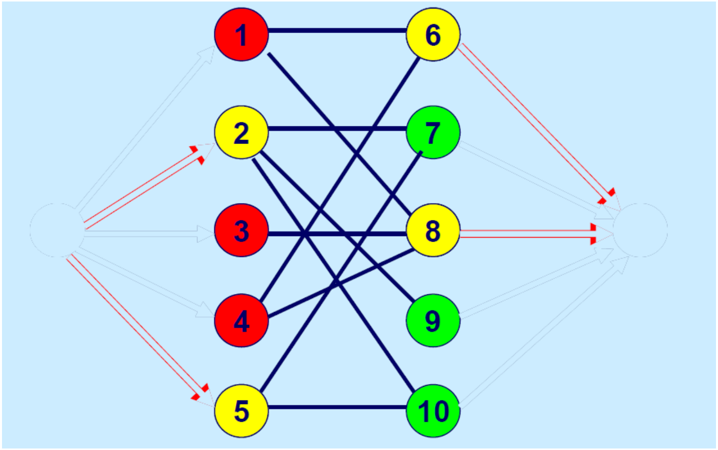


- The maximum s-t flow is 4.
- The max matching has cardinality 4.

Determine the minimum cut

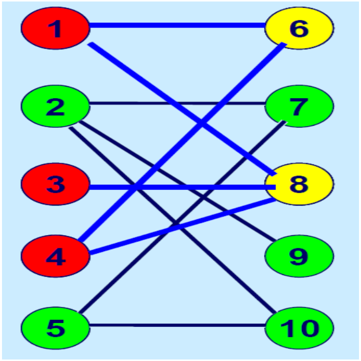
- plot the residual network $G(x)$
- Let S = {j : s $\rightarrow$j in G(x)} and let T = N$\setminus$S.
- S = {s, 1, 3, 4, 6, 8}. T = {2, 5, 7, 9, 10, t}.
- There is no arc from {1, 3, 4} to {7, 9, 10} or from {6, 8} to {2, 5}. Any such arc would have an infinite capacity.

Find the min node cover



- The minimum node cover is the set of nodes incident to the arcs across the cut. Max-Flow Min-Cut implies the duality theorem for matching.
- minimum node cover: {2,5,6,8}

Philip Hall's Theorem

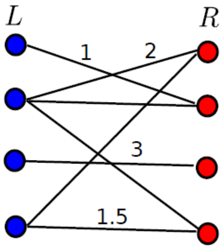


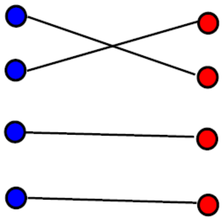
- A perfect matching is a matching which matches all nodes of the graph. That is, every node of the graph is incident to exactly one edge of the matching.
- Philip Hall's Theorem. If there is no perfect matching, then there is a set S of nodes of N1 such that |S| > |T| where T are the nodes of N2 adjacent to S.

The Max-Weight Bipartite Matching Problem

Given a bipartite graph G = (N, A), with N = L ∪ R, and weights w_{ij} on edges (i,j), find a maximum weight matching.

- Matching: a set of edges covering each node at most once
- Let n=|N| and m = |A|.
- Equivalent to maximum weight / minimum cost perfect matching.





The Max-Weight Bipartite Matching

Integer Programming (IP) formulation

$$\begin{aligned}
\max \quad & \sum_{ij} w_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j} x_{ij} \le 1, \forall i \in L \\
& \sum_{i} x_{ij} \le 1, \forall j \in R \\
& x_{ij} \in \{0, 1\}, \forall (i, j) \in A
\end{aligned}$$

- $x_{ij} = 1$ indicate that we include edge (i, j) in the matching
- IP: non-convex feasible set

The Max-Weight Bipartite Matching

Integer program (IP)

$$\begin{aligned} \max \quad & \sum_{ij} w_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j} x_{ij} \leq 1, \forall i \in L \\ & \sum_{i} x_{ij} \leq 1, \forall j \in R \\ & x_{ij} \in \{0, 1\}, \forall (i,j) \in A \end{aligned}$$

LP relaxation

$$\begin{aligned} \max \quad & \sum_{ij} w_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j} x_{ij} \leq 1, \forall i \in L \\ & \sum_{i} x_{ij} \leq 1, \forall j \in R \\ & x_{ij} \geq 0, \forall (i,j) \in A \end{aligned}$$

- Theorem. The feasible region of the matching LP is the convex hull of indicator vectors of matchings.
- This is the strongest guarantee you could hope for an LP relaxation of a combinatorial problem
- Solving LP is equivalent to solving the combinatorial problem

Primal-Dual Interpretation

Primal LP relaxation

$$\begin{aligned} \max \quad & \sum_{ij} w_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j} x_{ij} \le 1, \forall i \in L \\ & \sum_{i} x_{ij} \le 1, \forall j \in R \\ & x_{ij} \ge 0, \forall (i,j) \in A \end{aligned}$$

Dual

$$\begin{aligned} \min \quad & \sum_{i} y_i \\ \text{s.t.} \quad & y_i + y_j \ge w_{ij}, \forall (i,j) \in A \\ & y \ge 0 \end{aligned}$$

- Dual problem is solving minimum vertex cover: find smallest set of nodes S such that at least one end of each edge is in S
- From strong duality theorem, we know $P^*_{LP} = D^*_{LP}$

Primal-Dual Interpretation

Suppose edge weights $w_{ij} = 1$, then binary solutions to the dual are node covers.

Dual

$$\begin{aligned} \min \quad & \sum_i y_i \\ \text{s.t. } & y_i + y_j \geq 1, \forall (i,j) \in A \\ & y \geq 0 \end{aligned}$$

Dual Integer Program

$$\begin{aligned} \min \quad & \sum_i y_i \\ \text{s.t. } & y_i + y_j \geq 1, \forall (i,j) \in A \\ & y \in \{0, 1\} \end{aligned}$$

- Dual problem is solving minimum vertex cover: find smallest set of nodes S such that at least one end of each edge is in S
- From strong duality theorem, we know $P^*_{LP} = D^*_{LP}$
- Consider IP formulation of the dual, then

$$P^*_{IP} \leq P^*_{LP} = D^*_{LP} \leq D^*_{IP}$$

Total Unimodularity

Defintion: A matrix A is Totally Unimodular if every square submatrix has determinant 0, +1 or -1.

Theorem: If $A \in \mathbb{R}^{m \times n}$ is totally unimodular, and b is an integer vector, then $\{x : Ax \leq b; x \geq 0\}$ has integer vertices.

- Non-zero entries of vertex x are solution of $A'x' = b'$ for some nonsingular square submatrix A' and corresponding sub-vector b'
- Cramer's rule:
$$x_i = \frac{\det(A'_i \mid b')}{\det A'}$$

Claim: The constraint matrix of the bipartite matching LP is totally unimodular.

The Minimum weight vertex cover

- undirected graph G = (N, A) with node weights $w_i \geq 0$
- A vertex cover is a set of nodes S such that each edge has at least one end in S
- The weight of a vertex cover is sum of all weights of nodes in the cover
- Find the vertex cover with minimum weight

Integer Program

$$\begin{aligned} \min \quad & \sum_i w_i y_i \\ \text{s.t.}\ & y_i + y_j \geq 1, \forall (i,j) \in A \\ & y \in \{0,1\} \end{aligned}$$

LP Relaxation

$$\begin{aligned} \min \quad & \sum_i w_i y_i \\ \text{s.t.}\ & y_i + y_j \geq 1, \forall (i,j) \in A \\ & y \geq 0 \end{aligned}$$

LP Relaxation for the Minimum weight vertex cover

- In the LP relaxation, we do not need $y \leq 1$, since the optimal solution y^* of the LP does not change if $y \leq 1$ is added.
  Proof: suppose that there exists an index i such that the optimal solution of the LP y_i^* is strictly larger than one. Then, let y' be a vector which is same as y^* except for $y_i' = 1 < y_i^*$. This y' satisfies all the constraints, and the objective function is smaller.
- The solution of the relaxed LP may not be integer, i.e., $0 < y_i^* < 1$
- rounding technique:

$$y_i' = \begin{cases} 0, & \text{if } y_i^* < 0.5 \\ 1, & \text{if } y_i^* \geq 0.5 \end{cases}$$

- The rounded solution y' is feasible to the original problem

LP Relaxation for the Minimum weight vertex cover

The weight of the vertex cover we get from rounding is at most twice as large as the minimum weight vertex cover.

- Note that $y'_i = \min(\lfloor 2y^*_i \rfloor, 1)$
- Let P^*_{IP} be the optimal solution for IP, and P^*_{LP} be the optimal solution for the LP relaxation
- Since any feasible solution for IP is also feasible in LP, $P^*_{LP} \le P^*_{IP}$
- The rounded solution y' satisfy

$$\sum_i y'_i w_i \quad = \quad \sum_i \min(\lfloor 2y^*_i \rfloor, 1) w_i \le \sum_i 2y^*_i w_i = 2P^*_{LP} \le 2P^*_{IP}$$

Outline

The Minimum Cost Spanning Tree (MST) Problem

Given a connected undirected graph G = (N, A), and costs w_{ij} on edges (i,j), find a minimum cost spanning tree of G.

- Spanning Tree: an acyclic set of edges connecting every pair of nodes
- When graph is disconnected, can search for min-cost spanning forest instead
- We use n and m to denote |N| and |A|, respectively.

Kruskal's Algorithm

The minimum spanning tree problem can be solved efficiently by a simple greedy algorithm

Kruskal's Algorithm

```
• $T \leftarrow \emptyset$
• Sort edges in increasing order of cost
• For each edge (i,j) in order:
    • if $T \cup \{(i,j)\}$ is acyclic, add (i,j) to T
```

- Proof of correctness is via a simple exchange argument.

MST Linear Program

Integer program (IP)

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} w_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{(i,j)\in A_S} x_{ij} \le |S|-1, \forall S \subseteq A \\ & x_{ij} \in \{0,1\}, \forall (i,j) \in A \end{aligned}$$

LP relaxation

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} w_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{(i,j)\in A_S} x_{ij} \le |S|-1, \forall S \subseteq A \\ & x_{ij} \ge 0, \forall (i,j) \in A \end{aligned}$$

Theorem

The feasible region of the above LP is the convex hull of spanning trees.

- Proof by finding a dual solution with cost matching the output of Kruskal's algorithm.

Optimization over Sets

- Most combinatorial optimization problems can be thought of as choosing the best set from a family of allowable sets
  - Shortest paths
  - Max-weight matching
  - travelling sales problems (TSP)
- Set system: Pair $(\mathcal{X}, \mathcal{I})$ where $\mathcal{X}$ is a finite ground set and $\mathcal{I} \subseteq 2^{\mathcal{X}}$ are the feasible sets
- Objective: often "linear", referred to as modular
- Analogues of concave convex: submodular and supermodular (in no particular order!)

Matroids

Matroids

A set system $M = (\mathcal{X}, \mathcal{I})$ is a matroid if

- $\emptyset \in \mathcal{I}$
- If $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$ (Downward Closure)
- If $A, B \in I$ and $|B| > |A|$, then $\exists x \in B \backslash A$ such that $A \cup \{x\} \in \mathcal{I}$ (Exchange Property)

- Three conditions above are called the matroid axioms
- $A \in \mathcal{I}$ is called an independent set of the matroid.
- The union of all sets in $\mathcal{I}$ is called the ground set.
- An independent set is called a basis if it is not a proper subset of another independent set.
- The matroid whose independent sets are acyclic subgraphs is called a graphic matroid

Linear Matroids

- $\mathcal{X}$ is a finite set of vectors $\{v_1, \ldots, v_m\} \subseteq \mathbb{R}^n$
- $S \in \mathcal{I}$ iff the vectors in S are linearly independent

- Downward closure: If a set of vectors is linearly independent, then every subset of it is also
- Exchange property: Can always extend a low-dimension independent set S by adding vectors from a higher dimension independent set T

Uniform Matroids

- $\mathcal{X}$ is an arbitrary finite set $\{1, 2, \ldots, n\}$
- $S \in \mathcal{I}$ iff $|S| \leq k$

- Downward closure: If a set S has $|S| \leq k$ then the same holds for $T \subseteq S$.
- Exchange property: If $|S| < |T| \leq k$, then there is an element in $T \backslash S$, and we can add it to S while preserving independence.

Partition Matroids

- $\mathcal{X}$ is the disjoint union of classes $X_1, \ldots, X_m$
- Each class X_j has an upper bound k_j
- $S \in \mathcal{I}$ iff $|S \cap X_j| \leq k_j$ for all j

- This is the “disjoint union” of a number of uniform matroids

Matroid optimization problem

Definition

- suppose each element of the ground set of a matroid $\mathcal{I}$ is given an arbitrary non-negative weight.
- The matroid optimization problem is to compute a basis with maximum total weight.

The Greedy Algorithm on Matroids

The Greedy Algorithm

- $B \leftarrow \emptyset$
- Sort nonnegative elements of $\mathcal{X}$ in decreasing order of weight
    - $\{1, \ldots, n\}$ with $w_1 \geq w_2 \geq \ldots w_n \geq 0$
- For $i = 1$ to n:
    - if $B \cup \{i\} \in \mathcal{I}$, add i to B

Theorem

The greedy algorithm returns the maximum weight set for every choice of weights if and only if the set system $(\mathcal{X}, \mathcal{I})$ is a matroid.