


Network Flow

CMSC 451, Summer 2009



Some slides by Kevin Wayne
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

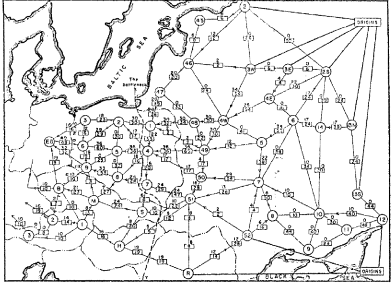
Reminders

- Homework 2 due now
- Group project checkpoint 1 due now

- Midterm exam on Friday (during class time)

- Homework 3 due Monday
 - Part 1: 7.2, 7.5

Soviet Rail Network, 1955



Reference: *On the history of the transportation and maximum flow problems.*
Alexander Schrijver in *Math Programming*, 91: 3, 2002.

Maximum Flow and Minimum Cut

Max flow and min cut.

- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality.

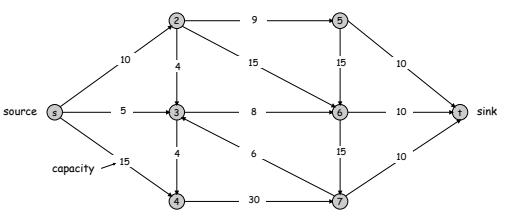
Nontrivial applications / reductions.

- Data mining.
- Open-pit mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Network reliability.
- Distributed computing.
- Egalitarian stable matching.
- Security of statistical data.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Many many more ...

Minimum Cut Problem

Flow network.

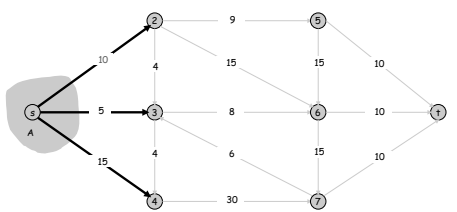
- Abstraction for material **flowing** through the edges.
- $G = (V, E)$ = directed graph, no parallel edges.
- Two distinguished nodes: s = source, t = sink.
- $c(e)$ = capacity of edge e .



Cuts

Def. An s - t cut is a partition (A, B) of V with $s \in A$ and $t \in B$.

Def. The **capacity** of a cut (A, B) is: $cap(A, B) = \sum_{e \text{ out of } A} c(e)$



Capacity = 10 + 5 + 15
= 30

Cuts

Def. An **s-t cut** is a partition (A, B) of V with s ∈ A and t ∈ B.

Def. The **capacity** of a cut (A, B) is: $cap(A, B) = \sum_{e \text{ out of } A} c(e)$

Capacity = 9 + 15 + 8 + 30 = 62

Minimum Cut Problem

Min s-t cut problem. Find an s-t cut of minimum capacity.

Capacity = 10 + 8 + 10 = 28

Flows

Def. An **s-t flow** is a function that satisfies:

- For each e ∈ E: $0 \leq f(e) \leq c(e)$ [capacity]
- For each v ∈ V - {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [conservation]

Def. The **value** of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$.

Flows

Def. An **s-t flow** is a function that satisfies:

- For each e ∈ E: $0 \leq f(e) \leq c(e)$ [capacity]
- For each v ∈ V - {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [conservation]

Def. The **value** of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$.

Maximum Flow Problem

Max flow problem. Find s-t flow of maximum value.

Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving s.

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$

Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s - t cut. Then, the net flow sent across the cut is equal to the amount leaving s .

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$

Value = 6 + 0 + 8 - 1 + 11 = 24

Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s - t cut. Then, the net flow sent across the cut is equal to the amount leaving s .

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$

Value = 10 + 4 + 8 - 0 + 10 = 24

Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s - t cut. Then

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f).$$

Pf.

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

by flow conservation, all terms except $v = s$ are 0

$$\rightarrow \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right)$$

$$= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e).$$

Flows and Cuts

Weak duality. Let f be any flow, and let (A, B) be any s - t cut. Then the value of the flow is at most the capacity of the cut.

Cut capacity = 30 \Rightarrow Flow value \leq 30

Capacity = 30

Flows and Cuts

Weak duality. Let f be any flow. Then, for any s - t cut (A, B) we have $v(f) \leq \text{cap}(A, B)$.

Pf.

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

$$\leq \sum_{e \text{ out of } A} f(e)$$

$$\leq \sum_{e \text{ out of } A} c(e)$$

$$= \text{cap}(A, B)$$

Certificate of Optimality

Corollary. Let f be any flow, and let (A, B) be any cut. If $v(f) = \text{cap}(A, B)$, then f is a max flow and (A, B) is a min cut.

Value of flow = 28
Cut capacity = 28 \Rightarrow Flow value \leq 28

Towards a Max Flow Algorithm

Greedy algorithm.

- Start with $f(e) = 0$ for all edge $e \in E$.
- Find an s-t path P where each edge has $f(e) < c(e)$.
- Augment flow along path P.
- Repeat until you get stuck.

Flow value = 0

Towards a Max Flow Algorithm

Greedy algorithm.

- Start with $f(e) = 0$ for all edge $e \in E$.
- Find an s-t path P where each edge has $f(e) < c(e)$.
- Augment flow along path P.
- Repeat until you get stuck.

Flow value = 20

Towards a Max Flow Algorithm

Greedy algorithm.

- Start with $f(e) = 0$ for all edge $e \in E$.
- Find an s-t path P where each edge has $f(e) < c(e)$.
- Augment flow along path P.
- Repeat until you get **stuck**.

locally optimality \neq global optimality

greedy = 20 opt = 30

Residual Graph

Original edge: $e = (u, v) \in E$.

- Flow $f(e)$, capacity $c(e)$.

Residual edge.

- "Undo" flow sent.
- $e = (u, v)$ and $e^R = (v, u)$.
- Residual capacity:

$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$

Residual graph: $G_f = (V, E_f)$.

- Residual edges with positive residual capacity.
- $E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$.

Augmenting Path Algorithm

```

Augment(f, c, P) {
  b ← bottleneck(P)
  foreach e ∈ P {
    if (e ∈ E) f(e) ← f(e) + b
    else f(e^R) ← f(e^R) - b
  }
  return f
}
    
```

the minimum cost of any edge along the path

forward edge
reverse edge

decrease the flow in the original graph along the edges by b if our path in the residual graph uses a reverse edge

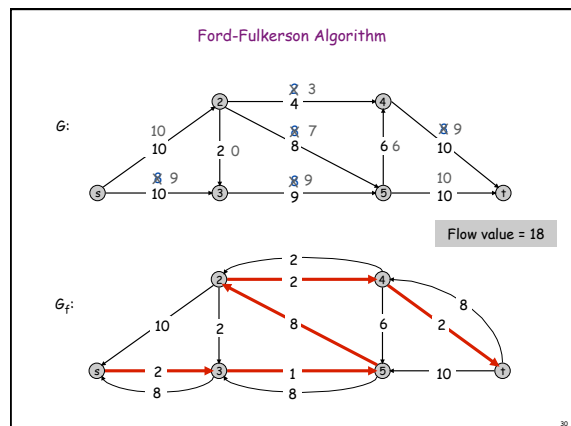
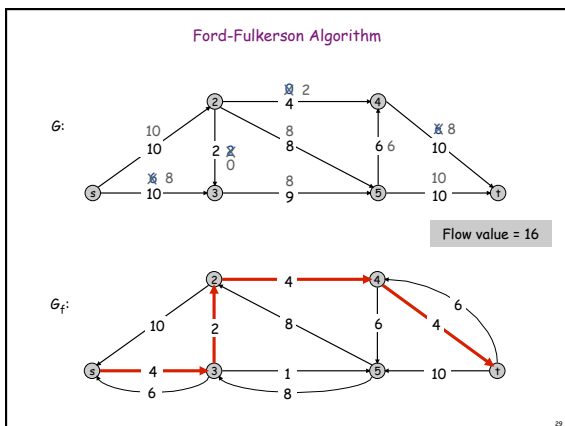
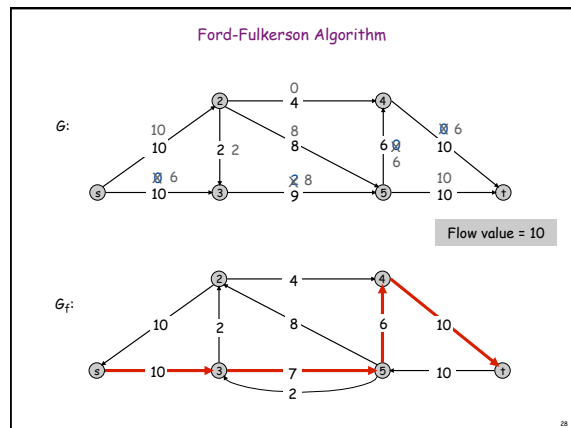
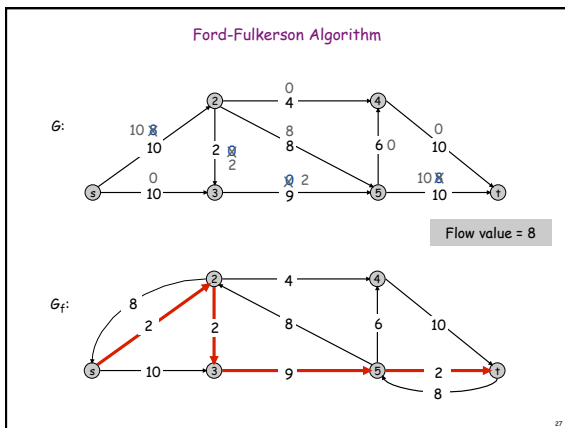
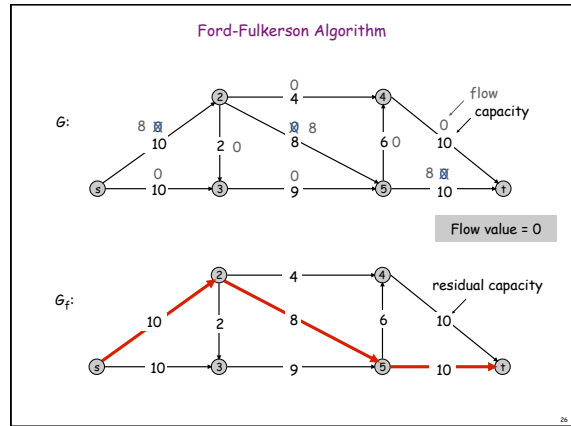
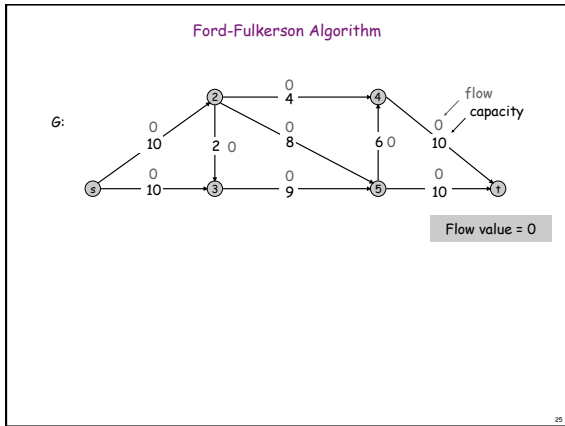
```

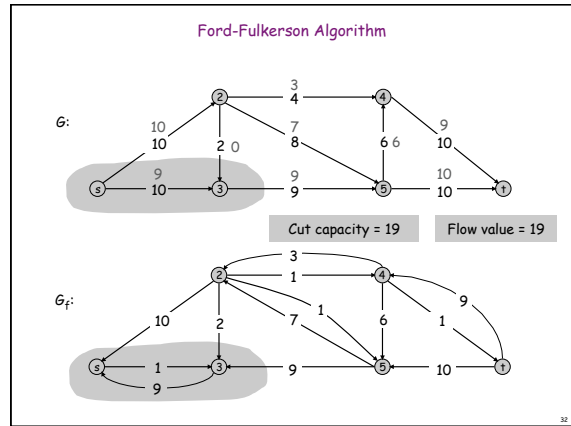
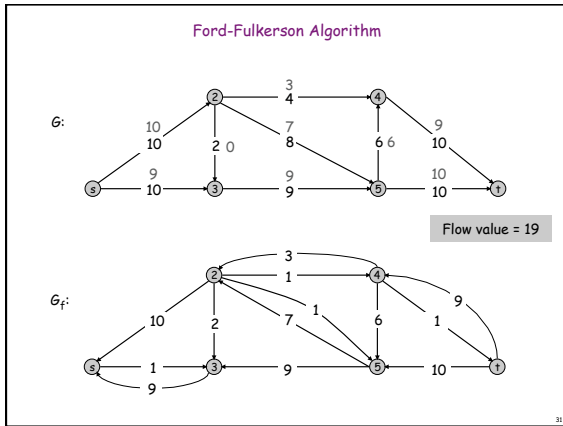
Ford-Fulkerson(G, s, t, c) {
  foreach e ∈ E f(e) ← 0
  G_f ← residual graph

  while (there exists augmenting path P) {
    f ← Augment(f, c, P)
    update G_f
  }
  return f
}
    
```

Ford-Fulkerson Algorithm

G:





Ford-Fulkerson Algorithm: Example

```

Augment(f, c, P) {
  b ← bottleneck(P)
  foreach e ∈ P {
    if (e ∈ E) f(e) ← f(e) + b
    else f(erev) ← f(erev) - b
  }
  return f
}

Ford-Fulkerson(G, s, t, c) {
  f ← 0
  G ← residual graph
  while (there exists augmenting path P) {
    f ← Augment(f, c, P)
    update Gf
  }
  return f
}
    
```

Max-Flow Min-Cut Theorem

Augmenting path theorem. Flow f is a max flow if and only if there are no augmenting paths.

Max-flow min-cut theorem. [Elias-Feinstein-Shannon 1956, Ford-Fulkerson 1956] The value of the max flow is equal to the value of the min cut.

Pf. We prove both simultaneously by showing that the following are equivalent:

- (i) There exists a cut (A, B) such that $v(f) = \text{cap}(A, B)$.
- (ii) Flow f is a max flow.
- (iii) There is no augmenting path relative to f .

(i) \Rightarrow (ii) This was the corollary to weak duality lemma.
 (ii) \Rightarrow (iii) We show the contrapositive. (not (iii) implies not (ii))

- Let f be a flow. If there exists an augmenting path, then we can improve f by sending flow along the path.

Proof of Max-Flow Min-Cut Theorem

(iii) \Rightarrow (i) (There is no augmenting path relative to f . \Rightarrow There exists a cut (A, B) such that $v(f) = \text{cap}(A, B)$.)

- Let f be a flow with no augmenting paths.
- Let A be set of vertices reachable from s in residual graph.
- By definition of A , $s \in A$.
- By definition of f , $t \notin A$.

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

A includes all reachable vertices by definition = $\sum_{e \text{ out of } A} c(e)$ = $\text{cap}(A, B)$

Running Time

Assumption. All capacities are integers between 1 and C .

Invariant. Every flow value $f(e)$ and every residual capacity $c_f(e)$ remains an integer throughout the algorithm.

Theorem. The algorithm takes time at most $O(mC)$.

Pf. Each augmentation increases value by at least 1. BFS or DFS to find an augmenting path takes time $O(n + m) = O(m)$ if we assume each node has at least one adjacent edge. Updating the residual graph along the path also takes time $O(m)$.

Integrality theorem. If all capacities are integers, then there exists a max flow f for which every flow value $f(e)$ is an integer.

Pf. Since algorithm terminates, theorem follows from invariant. ■

Proof of Correctness. Follows from the Max Flow/Min Cut Theorem.