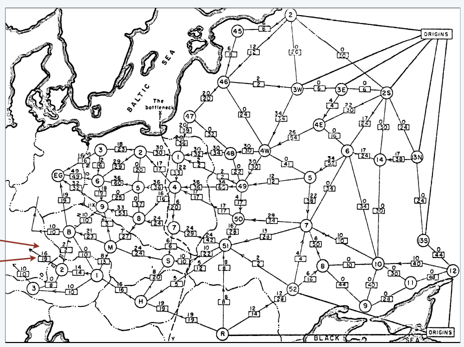
CSCI 355: Algorithm Design and Analysis
9. Network Flow

- max-flow and min-cut problems
- Ford–Fulkerson algorithm
- max-flow min-cut theorem

Maximum flows (Tolstoĭ, 1930s)

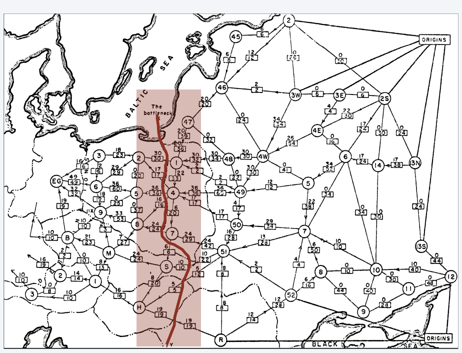
Soviet Union's goal. Maximize flow of supplies to Eastern Europe.



rail network connecting Soviet Union with Eastern European countries
(map declassified by Pentagon in 1999)

Minimum cuts (RAND, 1950s)

United States' goal. Cut supplies (if Cold War turns into real war).

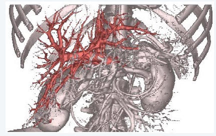


rail network connecting Soviet Union with Eastern European countries
(map declassified by Pentagon in 1999)

Max-flow and min-cut

A widely applicable model.

- Data mining.
- Open-pit mining.
- Bipartite matching.
- Network reliability.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Markov random fields.
- Distributed computing.
- Security of statistical data.
- Egalitarian stable matching.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Sensor placement for homeland security.
- Many, many, more.


liver and hepatic vascularization segmentation

CSCI 355: Algorithm Design and Analysis
9. Network Flow

- *max-flow and min-cut problems*
- *Ford–Fulkerson algorithm*
- *max-flow min-cut theorem*

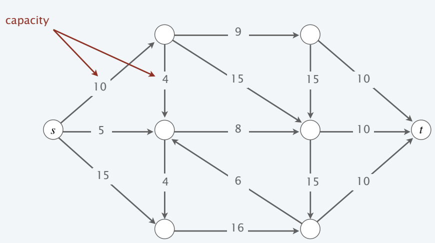
Flow network

A flow network is a tuple $G = (V, E, s, t, c)$.

- Digraph (V, E) with source $s \in V$ and sink $t \in V$.
- Capacity $c(e) \geq 0$ for each $e \in E$.

assume all vertices are reachable from s

Intuition. Material flowing through a transportation network; material originates at the source and is sent to the sink.



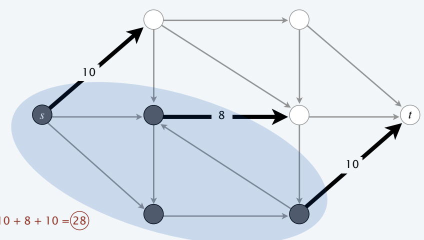
Minimum-cut problem

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

Def. The capacity of a cut is the sum of the capacities of edges from A to B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$

Min-cut problem. Find a cut of minimum capacity.



capacity = 10 + 8 + 10 = 28

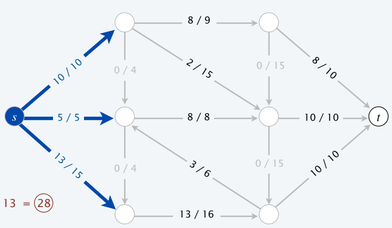
Maximum-flow problem

Def. An *st*-flow (flow) f is a function that satisfies:

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

Def. The value of a flow f is: $val(f) = \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ in to } s} f(e)$

Max-flow problem. Find a flow of maximum value.



value = 10 + 5 + 13 = 28

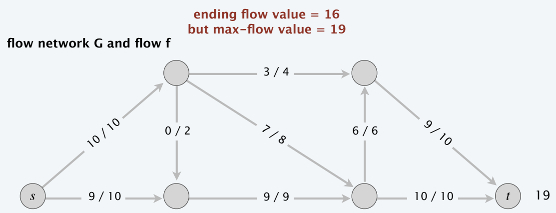
CSCI 355: Algorithm Design and Analysis
9. Network Flow

- *max-flow and min-cut problems*
- *Ford–Fulkerson algorithm*
- *max-flow min-cut theorem*

Toward a max-flow algorithm

Greedy algorithm.

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

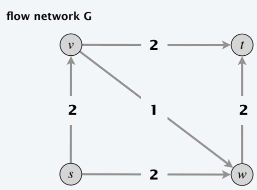


Why the greedy algorithm fails

Q. Why does the greedy algorithm fail?
A. Once the greedy algorithm increases the flow on an edge, it never decreases it.

Ex. Consider the flow network G.

- The unique max flow f^* has $f^*(v, w) = 0$.
- Greedy algorithm could choose $s \to v \to w \to t$ as first path.

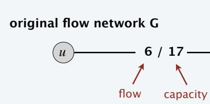


Bottom line. Need some mechanism to "undo" a bad decision.

Residual networks

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

- Flow $f(e)$.
- Capacity $c(e)$.

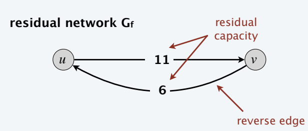


Reverse edge. $e^{\text{reverse}} = (v, u)$.

- "Undo" flow sent.

Residual capacity.

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



edges with positive residual capacity

Residual network. $G_f = (V, E_f, s, t, c_f)$.

- $E_f = \{e : f(e) < c(e)\} \cup \{e^{\text{reverse}} : f(e) > 0\}$.

where the flow on a reverse edge negates the flow on the corresponding forward edge

Key property. f' is a flow in G_f iff $f + f'$ is a flow in G.

Augmenting paths

Def. An augmenting path is a simple $s \rightsquigarrow t$ path in the residual network G_f.

Def. The bottleneck capacity of an augmenting path P is the minimum residual capacity of any edge in P.

Key property. Let f be a flow and let P be an augmenting path in G_f. Then, after calling $f' \leftarrow$ AUGMENT(f, P), the resulting f' is a flow and $val(f') = val(f) + bottleneck(G_f, P)$.

```
AUGMENT(f, P)
δ ← bottleneck capacity of augmenting path P.
FOREACH edge e ∈ P :
   IF (e ∈ E) f(e) ← f(e) + δ.
   ELSE       f(e^reverse) ← f(e^reverse) − δ.
RETURN f.
```


Ford–Fulkerson algorithm

see 9A

Ford–Fulkerson augmenting path algorithm.

- Start with $f(e) = 0$ for each edge $e \in E$.
- Find an $s \rightsquigarrow t$ path P in the residual network G_f.
- Augment flow along path P.
- Repeat until you get stuck.


Ford


Fulkerson

```
FORD–FULKERSON(G)
FOREACH edge e ∈ E : f(e) ← 0.
G_f ← residual network of G with respect to flow f.
WHILE (there exists an s⇝t path P in G_f)    ← augmenting path
   f ← AUGMENT(f, P).
   Update G_f.
RETURN f.
```


CSCI 355: ALGORITHM DESIGN AND ANALYSIS
9. NETWORK FLOW

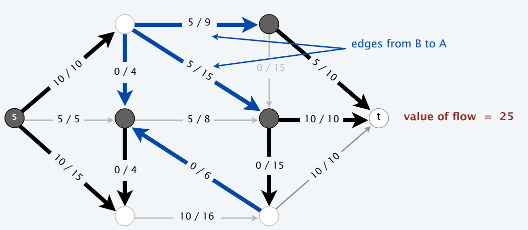
- max-flow and min-cut problems
- Ford–Fulkerson algorithm
- max-flow min-cut theorem

Relationship between flows and cuts

Flow value lemma. Let f be any flow and let (A, B) be any cut. Then the value of the flow f equals the net flow across the cut (A, B).

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

net flow across cut = (10 + 10 + 5 + 10 + 0 + 0) – (5 + 5 + 0 + 0) = 25



Relationship between flows and cuts

Flow value lemma. Let f be any flow and let (A, B) be any cut. Then the value of the flow f equals the net flow across the cut (A, B).

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

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

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

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

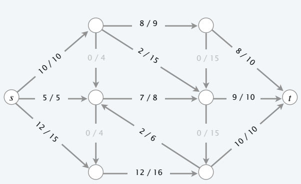
$$= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \quad ▪$$

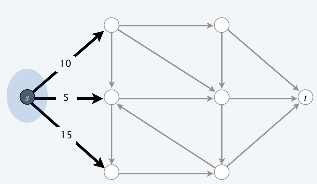
Relationship between flows and cuts

Weak duality. Let f be any flow and (A, B) be any cut. Then $val(f) \le cap(A, B)$.

Pf.
$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$
$$\le \sum_{e \text{ out of } A} f(e)$$
$$\le \sum_{e \text{ out of } A} c(e)$$
$$= cap(A, B) \quad ▪$$

flow value lemma





value of flow = 27 ≤ capacity of cut = 30

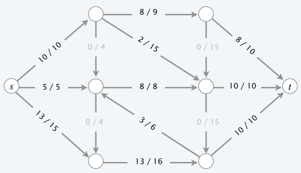
Certificate of optimality

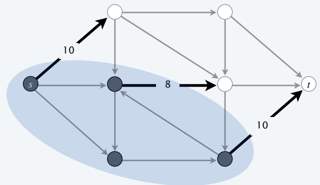
Corollary. Let f be a flow and let (A, B) be any cut.
If $val(f) = cap(A, B)$, then f is a max flow and (A, B) is a min cut.

Pf.
weak duality

- For any flow f': $val(f') \leq cap(A, B) = val(f)$.
- For any cut (A', B'): $cap(A', B') \geq val(f) = cap(A, B)$. ▪

weak duality



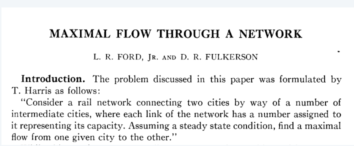


value of flow = 28 = capacity of cut = 28

Max-flow min-cut theorem

Max-flow min-cut theorem. Value of a max flow = capacity of a min cut.

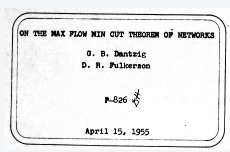
strong duality



MAXIMAL FLOW THROUGH A NETWORK

L. R. FORD, JR. AND D. R. FULKERSON

Introduction. The problem discussed in this paper was formulated by T. Harris as follows:
"Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other."

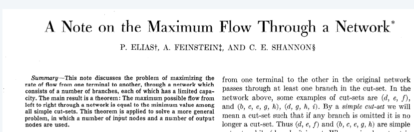


ON THE MAX FLOW MIN CUT THEOREM OF NETWORKS

G. B. Dantzig
D. R. Fulkerson

P-826

April 15, 1955



A Note on the Maximum Flow Through a Network*

P. ELIAS†, A. FEINSTEIN‡, AND C. E. SHANNON§

Summary—This note discusses the problem of maximizing the rate of flow from one terminal to another, through a network which consists of a number of branches, each of which has a limited capacity. The main result is a theorem: The maximum possible flow from left to right through a network is equal to the minimum value among all simple cut-sets. This theorem is applied to solve a more general problem, in which a number of input nodes and a number of output nodes are used.

from one terminal to the other in the original network passes through at least one branch in the cut-set. In the network above, some examples of cut-sets are (d, e, f), and (b, c, e, g, h), (d, g, h, i). By a *simple cut-set* we will mean a cut-set such that if any branch is omitted it is no longer a cut-set. Thus (d, e, f) and (b, c, e, g, h) are simple

Max-flow min-cut theorem

Max-flow min-cut theorem. Value of a max flow = capacity of a min cut.

Augmenting path theorem. A flow f is a max flow iff there are no augmenting paths.

Pf. The following three conditions are equivalent for any flow f:

i. There exists a cut (A, B) such that $cap(A, B) = val(f)$.
ii. f is a max flow.
iii. There is no augmenting path with respect to f. ← if Ford–Fulkerson terminates, then f is a max flow

[i ⇒ ii]

- This is the weak duality corollary. ▪

Max-flow min-cut theorem

Max-flow min-cut theorem. Value of a max flow = capacity of a min cut.

Augmenting path theorem. A flow f is a max flow iff there are no augmenting paths.

Pf. The following three conditions are equivalent for any flow f:

i. There exists a cut (A, B) such that $cap(A, B) = val(f)$.

ii. f is a max flow.

iii. There is no augmenting path with respect to f.

[ii ⇒ iii] We prove the contrapositive: ¬ iii ⇒ ¬ ii.

- Suppose that there is an augmenting path with respect to f.
- We can improve the flow f by sending the flow along this path.
- Thus, f is not a max flow. ▪

Max-flow min-cut theorem

Pf.

[iii ⇒ i]

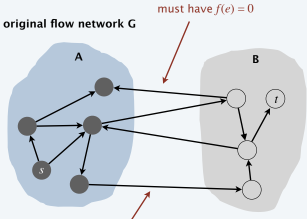
- Let f be a flow with no augmenting paths.
- Let A = set of vertices reachable from s in the residual network G_f.
- By the definition of A: $s \in A$.
- By the definition of flow f: $t \notin A$.

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

flow value lemma

$$= \sum_{e \text{ out of } A} c(e) - 0$$

$$= cap(A, B) \quad ▪$$



Computing a minimum cut from a maximum flow

Theorem. Given any max flow f, we can compute a min cut (A, B) in $O(m)$ time.

Pf. Let A = set of vertices reachable from s in the residual network G_f. ▪

argument from previous slide implies that capacity of (A, B) = value of flow f