Design and Analysis of Algorithms (VIII)

Max-Flow Min-Cut Theorem

Guoqiang Li
School of Software

SHANGHAI JIAO TONG UNIVERSITY

Max-Flow and Min-Cut Problem

A Flow Network

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

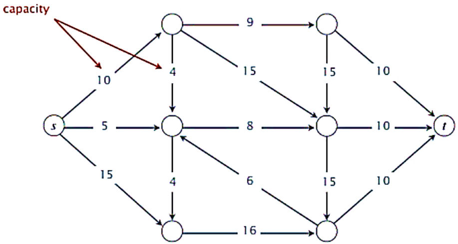
- Diagraph (V, E) with source $s \in V$ and sink $t \in V$.
- Capacity $c(e) > 0$ for each $e \in E$.

A Flow Network

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

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

Intuition. Material flowing through a transportation network, which originates at source and is sent to sink.



Minimum-Cut Problem

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

Minimum-Cut Problem

SHANGHAI JIAO TONG UNIVERSITY

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

Its capacity is the sum of the capacities of the edges from A to B.

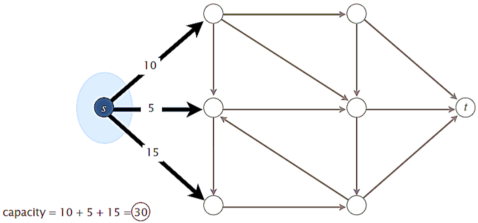
$$\text{cap}(A, B) = \sum_{e \text{ out of } A} c(e)$$

Minimum-Cut Problem

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

Its capacity is the sum of the capacities of the edges from A to B.

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



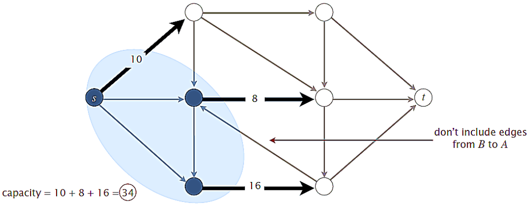
Minimum-Cut Problem

SHANGHAI JIAO TONG UNIVERSITY

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

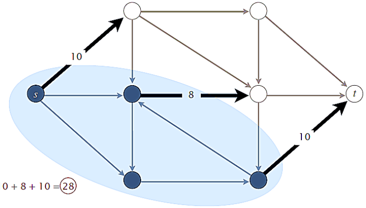
Its capacity is the sum of the capacities of the edges from A to B.

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



Minimum-Cut Problem

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



capacity = 10 + 8 + 10 = 28

Quiz 1

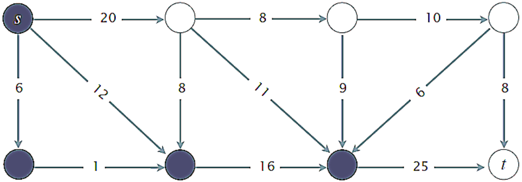
Which is the capacity of the given st-cut?

A. $11\ (20+25-8-11-9-6)$

B. $34\ (8+11+9+6)$

C. $45\ (20+25)$

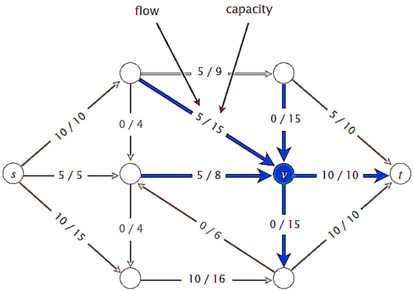
D. $79\ (20+25+8+11+9+6)$



Maximum-Flow Problem

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

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



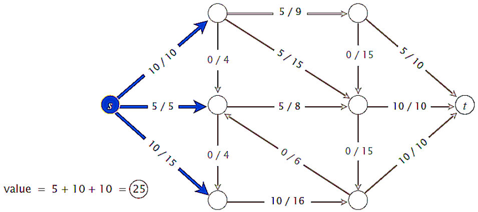
Maximum-Flow Problem

SHANGHAI JIAO TONG UNIVERSITY

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

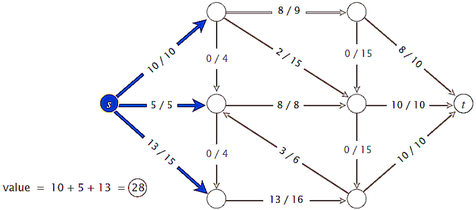
- For each $e \in E$: $0 \leq f(e) \leq c(e)$
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$

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



Maximum-Flow Problem

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

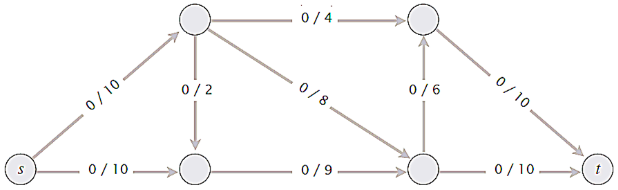


Ford-Fulkerson Algorithm

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.

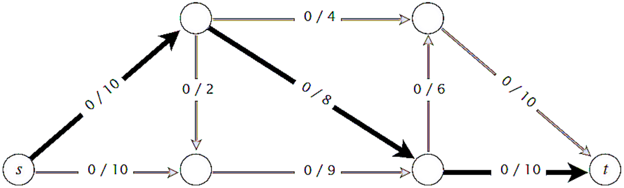


Toward a Max-Flow Algorithm

SHANGHAI JIAO TONG UNIVERSITY

Greedy algorithm.

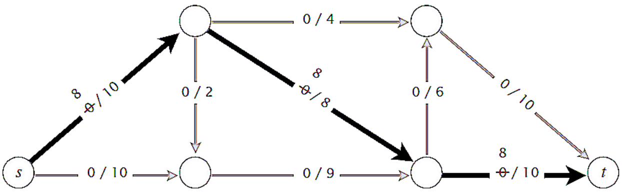
- Start with $f(e) = 0$ for each edge $e \in E$.
- Find an $s \rightsquigarrow t$ path P where each edge has $f(e) < c(e)$.
- Augment flow along path P.
- Repeat until you get stuck.



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.

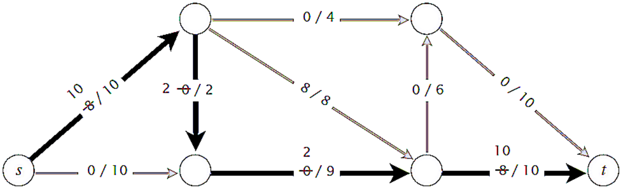


Toward a Max-Flow Algorithm

SHANGHAI JIAO TONG UNIVERSITY

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.

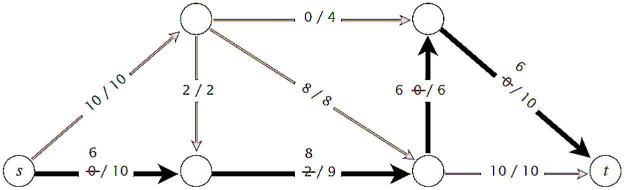


Toward a Max-Flow Algorithm

SHANGHAI JIAO TONG UNIVERSITY

Greedy algorithm.

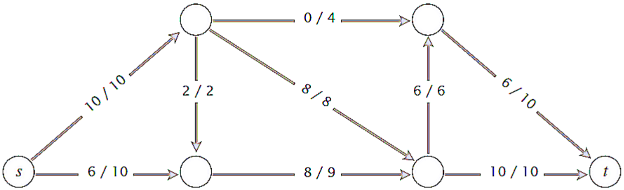
- Start with $f(e) = 0$ for each edge $e \in E$.
- Find an $s \rightsquigarrow t$ path P where each edge has $f(e) < c(e)$.
- Augment flow along path P.
- Repeat until you get stuck.



Toward a Max-Flow Algorithm

Greedy algorithm.

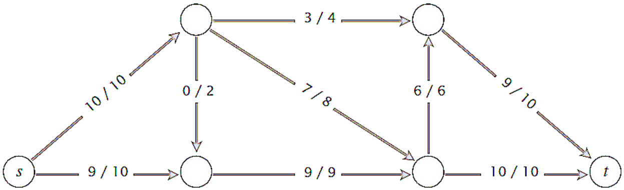
- Start with $f(e) = 0$ for each edge $e \in E$.
- Find an $s \rightsquigarrow t$ path P where each edge has $f(e) < c(e)$.
- Augment flow along path P.
- Repeat until you get stuck.



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?

Why the Greedy Algorithm Fails

Q. Why does the greedy algorithm fail?

A. Once greedy algorithm increases flow on an edge, it never decreases it.

Why the Greedy Algorithm Fails

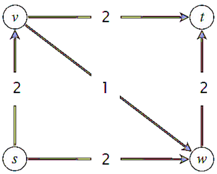
SHANGHAI JIAO TONG UNIVERSITY

Q. Why does the greedy algorithm fail?

A. Once greedy algorithm increases flow on an edge, it never decreases it.

Ex. Consider flow network G.

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



Why the Greedy Algorithm Fails

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

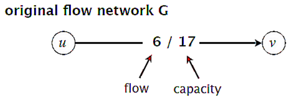
Residual Network

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

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

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

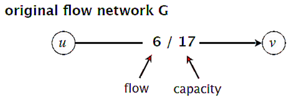
- Undo flow sent.



Residual Network

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

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

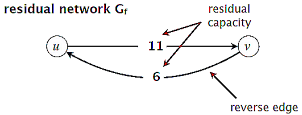


Reverse edge $e^{\text{reverse}} = (v, u)$

- Undo flow sent.

Residual capacity

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

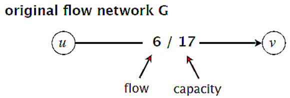


Residual Network

SHANGHAI JIAO TONG UNIVERSITY

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

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

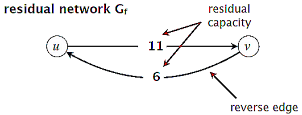


Reverse edge $e^{\text{reverse}} = (v, u)$

- Undo flow sent.

Residual capacity

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



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

- $E_f = \{e : f(e) < c(e)\} \cup \{e^{\text{reverse}} : f(e) > 0\}$.
- Key property: f' is a flow in G_f iff $f + f'$ is a flow in G

Augmenting Path

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

Augmenting Path

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

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

Augmenting Path

Key Property

Let f be a flow and let P be an augmenting path in G_f. After calling $f' \leftarrow \textsc{Augment}(f, P)$, the resulting f' is a flow and

$$val(f') = val(f) + bottleneck(G_f, P)$$

Augmenting Path

Key Property

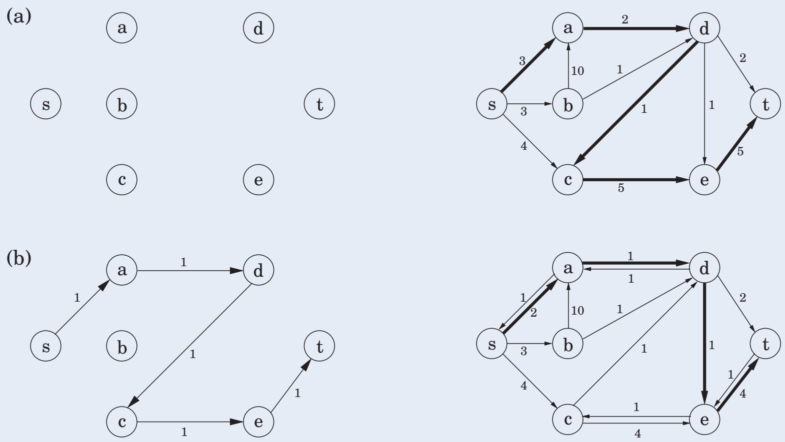
Let f be a flow and let P be an augmenting path in G_f. After calling $f' \leftarrow \text{AUGMENT}(f, P)$, the resulting f' is a flow and

$$val(f') = val(f) + bottleneck(G_f, P)$$

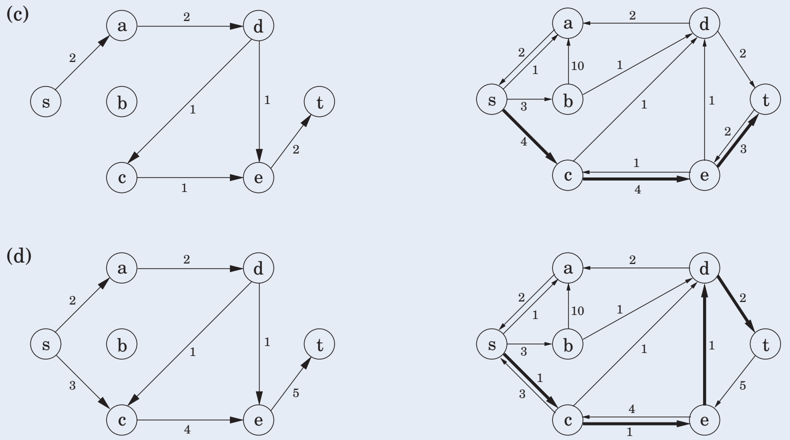
AUGMENT(f,P)

$\delta \leftarrow$ bottleneck capacity of augmenting path P;
for *each edge* $e \in P$ **do**
  if $(e \in E)$ **then** $f(e) \leftarrow f(e) + \delta$;
  else
    $f(e^{\text{reverse}}) \leftarrow f(e^{\text{reverse}}) - \delta$
  end
end
RETURN f;

A Flow Example



A Flow Example



Ford–Fulkerson Algorithm

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 Algorithm

FORD–FULKERSON(G)

for *each edge* $e \in E$ **do**
    $f(e) \leftarrow 0$
end
$G_f \leftarrow$ residual network of G with respect to flow f;
while *there exists an* $s \rightsquigarrow t$ *path* P *in* G_f **do**
    $f \leftarrow$ AUGMENT(f,P) ;
    UPDATE(G_f) ;
end
RETURN f;

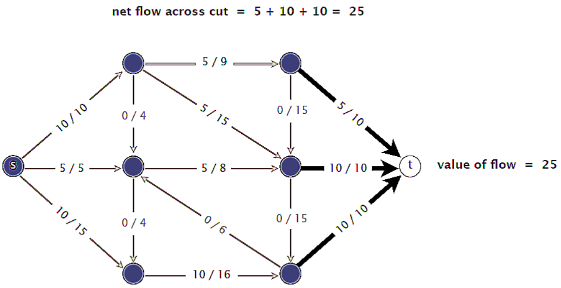
Max-Flow Min-Cut Theorem

Flows and Cuts

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).

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



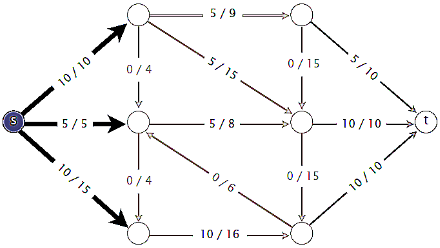
Flows and Cuts

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).

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

net flow across cut = 10 + 5 + 10 = 25



value of flow = 25

Flows and Cuts

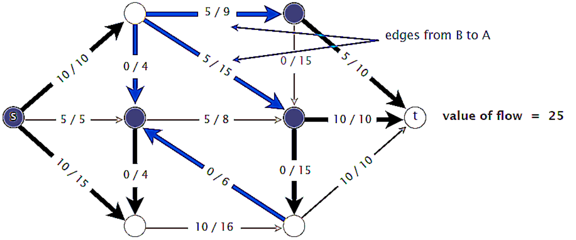
SHANGHAI JIAO TONG UNIVERSITY

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).

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

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



Quiz 3

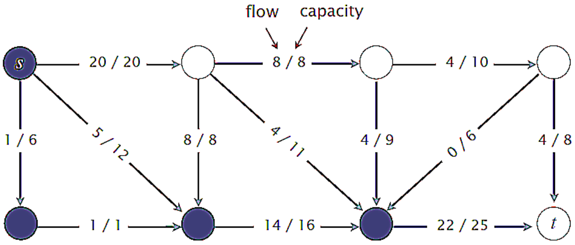
Which is the net flow across the given cut?

A. 11 $(20 + 25 - 8 - 11 - 9 - 6)$

B. 26 $(20 + 22 - 8 - 4 - 4)$

C. 42 $(20 + 22)$

D. 45 $(20 + 25)$



Flows and Cuts

SHANGHAI JIAO TONG UNIVERSITY

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).

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

Flows and Cuts

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).

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

Proof.

$$\begin{aligned} val(f) &= \sum_{e \text{ out of } s} f(e) - \sum_{\text{e in to } s} f(e) \\ &= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{\text{e in to } v} f(e) \right) \\ &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e). \end{aligned}$$

Flows and Cuts

Theorem (Weak Duality)

Let f be any flow and (A, B) be any cut. Then, $val(f) \leq \mathrm{cap}(A, B)$.

Flows and Cuts

Theorem (Weak Duality)

Let f be any flow and (A, B) be any cut. Then, $val(f) \leq \text{cap}(A, B)$.

Proof.

Flows and Cuts

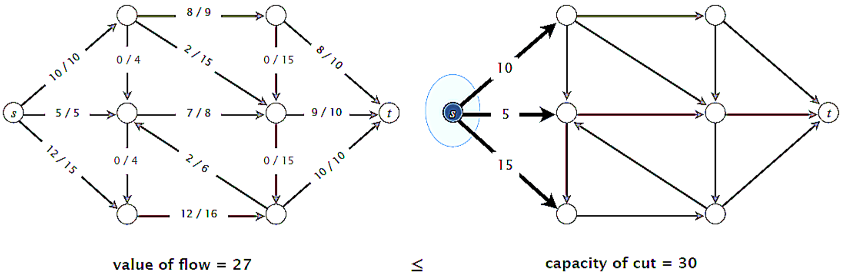
SHANGHAI JIAO TONG UNIVERSITY

Theorem (Weak Duality)

Let f be any flow and (A, B) be any cut. Then, $val(f) \leq \text{cap}(A, B)$.

Proof.

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \leq \sum_{e \text{ out of } A} f(e) \leq \sum_{e \text{ out of } A} c(e)$$
$$= \text{cap}(A, B)$$



Certificate of Optimality

Corollary

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

Certificate of Optimality

SHANGHAI JIAO TONG UNIVERSITY

Corollary

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

Proof.

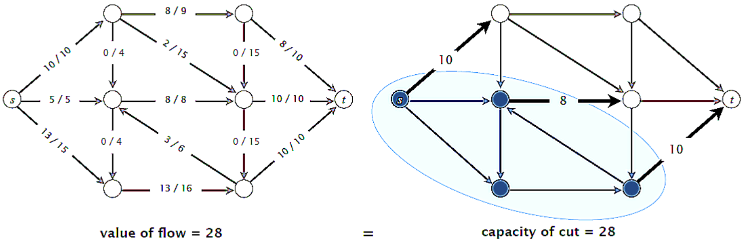
Certificate of Optimality

Corollary

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

Proof.

- For any flow f' : $\quad val\,(f') \le \text{cap}(A, B) = val(f)$.
- For any cut (A', B') : $\text{cap}\,(A', B') \ge val(f) = \text{cap}(A, B)$



Max-Flow Min-Cut Theorem

Max-Flow Min-Cut Theorem

Value of a max flow = Capacity of a min cut.

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.

SHANGHAI JIAO TONG UNIVERSITY

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 no augmenting paths.

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

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

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 no augmenting paths.

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

i. There exists a cut (A, B) such that $\mathrm{cap}(A, B) = val(f)$.
ii. f is a max flow.
iii. There is no augmenting path with respect to f.

[i$\Rightarrow$ ii]

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 no augmenting paths.

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

i. There exists a cut (A, B) such that $\mathrm{cap}(A, B) = val(f)$.
ii. f is a max flow.
iii. There is no augmenting path with respect to f.

[i$\Rightarrow$ ii] This is the weak duality corollary.

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 no augmenting paths.

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

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

[ii$\Rightarrow$ iii]

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 no augmenting paths.

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

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

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

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 no augmenting paths.

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

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

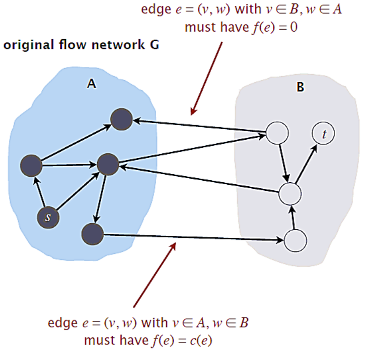
[ii$\Rightarrow$ iii] We prove contrapositive: $\neg$ iii $\Rightarrow$ $\neg$ii.

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

[iii $\Rightarrow$ i]

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

$$
\begin{aligned}
val(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
&= \sum_{e \text{ out of } A} c(e) - 0 \\
&= \mathrm{cap}(A, B)
\end{aligned}
$$



Analysis of the Algorithm

Analysis of the Algorithm

Assumption. Every edge capacity $c(e)$ is an integer between 1 and C.

Analysis of the Algorithm

Assumption. Every edge capacity $c(e)$ is an integer between 1 and C.

Integrality invariant. Throughout Ford-Fulkerson, every edge flow $f(e)$ and residual capacity $c_f(e)$ is an integer.

Analysis of the Algorithm

Assumption. Every edge capacity $c(e)$ is an integer between 1 and C.

Integrality invariant. Throughout Ford-Fulkerson, every edge flow $f(e)$ and residual capacity $c_f(e)$ is an integer.

Proof. By induction on the number of augmenting paths.

Analysis of the Algorithm

Assumption. Every edge capacity $c(e)$ is an integer between 1 and C.

Integrality invariant. Throughout Ford-Fulkerson, every edge flow $f(e)$ and residual capacity $c_f(e)$ is an integer.

Proof. By induction on the number of augmenting paths.

Theorem

Ford–Fulkerson terminates after at most $\mathrm{val}(f^*) \leq |V| \cdot C$ *augmenting paths, where* f^* *is a max flow.*

Analysis of the Algorithm

Assumption. Every edge capacity $c(e)$ is an integer between 1 and C.

Integrality invariant. Throughout Ford-Fulkerson, every edge flow $f(e)$ and residual capacity $c_f(e)$ is an integer.

Proof. By induction on the number of augmenting paths.

Theorem

Ford–Fulkerson terminates after at most $\mathrm{val}(f^*) \leq |V| \cdot C$ augmenting paths, where f^* is a max flow.

Proof. Each augmentation increases the value of the flow by at least 1.

Analysis of the Algorithm

SHANGHAI JIAO TONG UNIVERSITY

Corollary

The running time of Ford–Fulkerson is $O(|V| \cdot |E| \cdot C)$.

Analysis of the Algorithm

SHANGHAI JIAO TONG UNIVERSITY

Corollary

The running time of Ford–Fulkerson is $O(|V| \cdot |E| \cdot C)$.

Proof. Can use either BFS or DFS to find an augmenting path in $O(|E|)$ time.

Analysis of the Algorithm

Corollary

The running time of Ford–Fulkerson is $O(|V| \cdot |E| \cdot C)$.

Proof. Can use either BFS or DFS to find an augmenting path in $O(|E|)$ time.

Integrality Theorem

There exists an integral max flow f^*

Analysis of the Algorithm

Corollary

The running time of Ford–Fulkerson is $O(|V| \cdot |E| \cdot C)$.

Proof. Can use either BFS or DFS to find an augmenting path in $O(|E|)$ time.

Integrality Theorem

There exists an integral max flow f^*

Proof. Since Ford–Fulkerson terminates, theorem follows from integrality invariant.

Exponential Example

Q. Is generic Ford–Fulkerson algorithm poly-time in input size?

Exponential Example

Q. Is generic Ford–Fulkerson algorithm poly-time in input size?

A. No. If max capacity is C, then algorithm can take $\geq C$ iterations.

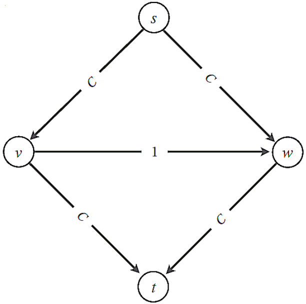
Exponential Example

SHANGHAI JIAO TONG UNIVERSITY

Q. Is generic Ford–Fulkerson algorithm poly-time in input size?

A. No. If max capacity is C, then algorithm can take $\geq C$ iterations.

- $s \to v \to w \to t$
- $s \to w \to v \to t$
- $s \to v \to w \to t$
- $s \to w \to v \to t$
- $\ldots$
- $s \to v \to w \to t$
- $s \to w \to v \to t$



Quiz 4

The Ford–Fulkerson algorithm is guaranteed to terminate if the edge capacities are . . .

A. Rational numbers.

B. Real numbers.

C. Both A and B.

D. Neither A nor B.

Choosing Good Augmenting Paths

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.

Choosing Good Augmenting Paths

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.

Pathology. When edge capacities can be irrational, no guarantee that Ford–Fulkerson terminates (or converges to a maximum flow)!

Choosing Good Augmenting Paths

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.

Pathology. When edge capacities can be irrational, no guarantee that Ford–Fulkerson terminates (or converges to a maximum flow)!

Goal. Choose augmenting paths so that:

- Can find augmenting paths efficiently.
- Few iterations.

Choosing Good Augmenting Paths

Choose augmenting paths with:

Choosing Good Augmenting Paths

Choose augmenting paths with:

- Max bottleneck capacity ("fattest").

Choosing Good Augmenting Paths

Choose augmenting paths with:

- Max bottleneck capacity (“fattest”).
- Sufficiently large bottleneck capacity.

Choosing Good Augmenting Paths

Choose augmenting paths with:

- Max bottleneck capacity ("fattest").
- Sufficiently large bottleneck capacity.
- Fewest edges.

Referred Materials

Referred Materials

- Content of this lecture comes from Section 7.1-7.2 in [KT05].