6.4 Maximum Flow

- introduction
- Ford–Fulkerson algorithm
- maxflow–mincut theorem
- analysis of running time
- Java implementation (see video)
- applications
6.4 Maximum Flow

- introduction
- Ford-Fulkerson algorithm
- maxflow-mincut theorem
- analysis of running time
- Java implementation
- applications
Mincut problem

**Input.** An edge-weighted digraph, source vertex $s$, and target vertex $t$. 

Each edge has a positive capacity.

The diagram shows a directed graph with labeled edges indicating capacities.
**Mincut problem**

**Def.** A *st-cut (cut)* is a partition of the vertices into two disjoint sets, with *s* in one set *A* and *t* in the other set *B*.

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

```
capacity = 10 + 5 + 15 = 30
```
**Mincut problem**

**Def.** A *st-cut* (cut) is a partition of the vertices into two disjoint sets, with *s* in one set *A* and *t* in the other set *B*.

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

![Diagram](image)

Capacity = $10 + 8 + 16 = 34$
**Min-cut problem**

**Def.** A \textit{st-cut (cut)} is a partition of the vertices into two disjoint sets, with \( s \) in one set \( A \) and \( t \) in the other set \( B \).

**Def.** Its \textit{capacity} is the sum of the capacities of the edges from \( A \) to \( B \).

**Minimum st-cut (mincut) problem.** Find a cut of minimum capacity.

capacity = 10 + 8 + 10 = 28
What is the capacity of the \( st \)-cut \( \{ A, E, F, G \} \)?

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

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

C. \( 45 \ (20 + 25) \)

D. \( 79 \ (20 + 25 + 8 + 11 + 9 + 6) \)
“Free world” goal. Disrupt rail network (if Cold War turns into real war).

rail network connecting Soviet Union with Eastern European countries
(map declassified by Pentagon in 1999)
Though maximum flow algorithms have a long history, revolutionary progress is still being made.

BY ANDREW V. GOLDBERG AND ROBERT E. TARJAN

Efficient Maximum Flow Algorithms

Efficient Maximum Flow Algorithms by Andrew Goldberg and Bob Tarjan

https://vimeo.com/100774435
Maxflow problem

**Input.** An edge-weighted digraph, source vertex $s$, and target vertex $t$.

Each edge has a positive capacity.
Maxflow problem

**Def.** An *st*-flow (flow) is an assignment of values to the edges such that:
- Capacity constraint: $0 \leq$ edge’s flow $\leq$ edge’s capacity.
- Local equilibrium: inflow = outflow at every vertex (except $s$ and $t$).
**Maxflow problem**

**Def.** An *st-flow (flow)* is an assignment of values to the edges such that:
- Capacity constraint: $0 \leq$ edge's flow $\leq$ edge's capacity.
- Local equilibrium: inflow = outflow at every vertex (except $s$ and $t$).

**Def.** The value of a flow is the inflow at $t$.

---

The value is calculated as $\sum$ of inflow into $t$. In this case, the value is $5 + 10 + 10 = \boxed{25}$.

We assume no edges point to $s$ or from $t$. 

---

![Graph with values on edges and node](attachment:graph.png)
Maxflow problem

**Def.** An *st-flow (flow)* is an assignment of values to the edges such that:
- Capacity constraint: $0 \leq \text{edge’s flow} \leq \text{edge’s capacity}$.
- Local equilibrium: inflow = outflow at every vertex (except $s$ and $t$).

**Def.** The *value* of a flow is the inflow at $t$.

**Maximum st-flow (maxflow) problem.** Find a flow of maximum value.

![Graph](image-url)
Maxflow application (Tolstoĭ 1930s)

**Soviet Union goal.** Maximize flow of supplies to Eastern Europe.

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

Input. An edge-weighted digraph, source vertex $s$, and target vertex $t$.

Min cut problem. Find a cut of minimum capacity.

Max flow problem. Find a flow of maximum value.

Remarkable fact. These two problems are dual! [stay tuned]
6.4 Maximum Flow

- introduction
- Ford–Fulkerson algorithm
- maxflow–mincut theorem
- analysis of running time
- Java implementation
- applications
Ford–Fulkerson algorithm

Initialization. Start with 0 flow.
Idea: increase flow along augmenting paths

**Augmenting path.** Find an undirected path from $s$ to $t$ such that:
- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

1st augmenting path

![Diagram showing the first augmenting path with a bottleneck capacity of 10 and a value of flow of 10.](image)
Idea: increase flow along augmenting paths

Augmenting path. Find an undirected path from $s$ to $t$ such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

2\textsuperscript{nd} augmenting path

value of flow

10 + 10 = 20
Idea: increase flow along augmenting paths

Augmenting path. Find an undirected path from $s$ to $t$ such that:
- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

3rd augmenting path

$20 + 5 = 25$
Idea: increase flow along augmenting paths

**Augmenting path.** Find an undirected path from $s$ to $t$ such that:
- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).

4th augmenting path

\[
\text{value of flow} = 25 + 3 = 28
\]
Idea: increase flow along augmenting paths

**Termination.** All paths from $s$ to $t$ are blocked by either a
- Full forward edge.
- Empty backward edge.

no more augmenting paths

[Diagram showing a network with nodes and edges, indicating flow values and cut capacity with a value of 28.]
Maxflow: quiz 2

Which is an augmenting path?

A.  \( A \to F \to G \to D \to H \)

B.  \( A \to F \to B \to G \to C \to D \to H \)

C.  Both A and B.

D.  Neither A nor B.
Ford–Fulkerson algorithm

Start with 0 flow.

While there exists an augmenting path:
  – find an augmenting path
  – compute bottleneck capacity
  – update flow on that path by bottleneck capacity

Fundamental questions.

• How to find an augmenting path?
• How many augmenting paths?
• Guaranteed to compute a maxflow?
• Given a maxflow, how to compute a mincut?
6.4 **Maximum Flow**

- introduction
- Ford–Fulkerson algorithm
- maxflow–mincut theorem
- analysis of running time
- Java implementation
- applications

https://algs4.cs.princeton.edu
Relationship between flows and cuts

**Def.** The net flow across a cut \((A, B)\) is the sum of the flows on its edges from \(A\) to \(B\) minus the sum of the flows on its edges from \(B\) to \(A\).

\[
\text{net flow across cut} = 5 + 10 + 10 = 25
\]

value of flow = 25
**Relationship between flows and cuts**

**Def.** The net flow across a cut \((A, B)\) is the sum of the flows on its edges from \(A\) to \(B\) minus the sum of the flows on its edges from \(B\) to \(A\).

\[
\text{net flow across cut} = 10 + 5 + 10 = 25
\]
Def. The net flow across a cut \((A, B)\) is the sum of the flows on its edges from \(A\) to \(B\) minus the sum of the flows on its edges from \(B\) to \(A\).

\[
\text{net flow across cut} = (10 + 10 + 5 + 10 + 0 + 0) - (5 + 5 + 0 + 0) = 25
\]
Which is the net flow across the $st$-cut \{ $A, E, F, G$ \}?

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

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

C. 42 \((20 + 22)\)

D. 45 \((20 + 25)\)
Relationship between flows and cuts

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

**Intuition.** Conservation of flow.

**Pf.** By induction on the size of $B$.

- Base case: $B = \{ t \}$.
- Induction step: remains true by local equilibrium when moving any vertex from $A$ to $B$.

**Corollary.** Outflow from $s = \text{inflow to } t = \text{value of flow}$. 
Relationship between flows and cuts

Weak duality. Let $f$ be any flow and let $(A, B)$ be any cut. Then, the value of the flow $f \leq$ the capacity of the cut $(A, B)$.

Pf. Value of flow $f = \text{net flow across cut } (A, B) \leq \text{capacity of cut } (A, B)$.
Augmenting path theorem. A flow $f$ is a maxflow iff no augmenting paths.


**Pf.** For any flow $f$, the following three conditions are equivalent:

i. $f$ is a maxflow.

ii. There is no augmenting path with respect to $f$.

iii. There exists a cut whose capacity equals the value of the flow $f$.

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

- 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 maxflow. ●
Maxflow–mincut theorem

Augmenting path theorem. A flow $f$ is a maxflow iff no augmenting paths.

Pf. For any flow $f$, the following three conditions are equivalent:
  i. $f$ is a maxflow.
  ii. There is no augmenting path with respect to $f$.
  iii. There exists a cut whose capacity equals the value of the flow $f$.

[ iii $\implies$ i ]
  • Suppose that $(A, B)$ is a cut with capacity equal to the value of $f$.
  • Then, the value of any flow $f' \leq$ capacity of $(A, B) = \text{value of } f$.
  • Thus, $f$ is a maxflow. ■

weak duality by assumption
[ ii ⇒ iii ]

- Let $f$ be a flow with no augmenting paths.
- Let $A$ be set of vertices connected to $s$ by an undirected path with no full forward or empty backward edges.
- By definition of cut $(A, B)$, $s$ is in $A$.
- By definition of cut $(A, B)$ and flow $f$, $t$ is in $B$.
- Capacity of $(A, B) = \text{net flow across cut} = \text{value of flow } f$.  

**Maxflow-mincut theorem**

**Diagram:**
- **S** (source) connected to **A**
- **B** (sink) connected from **A**
- **t** (sink) connected from **B**
- Forward edge from **A** to **B** (flow = capacity)
- Backward edge from **B** to **A** (flow = 0)
- By construction of cut
- Flow-value lemma
Computing a mincut from a maxflow

To compute mincut \((A, B)\) from maxflow \(f\):

- By augmenting path theorem, no augmenting paths with respect to \(f\).
- Compute \(A = \) set of vertices connected to \(s\) by an undirected path with no full forward or empty backward edges.
- Capacity of cut \((A, B) = \) value of flow \(f \Rightarrow \) mincut.
Maxflow: quiz 4

Given the following maxflow, which is a mincut?

A. \( S = \{ A \} \).

B. \( S = \{ A, B, C, E, F \} \).

C. Both A and B.

D. Neither A nor B.
6.4 Maximum Flow

- introduction
- Ford–Fulkerson algorithm
- maxflow–mincut theorem
- analysis of running time
- Java implementation
- applications
Ford–Fulkerson algorithm analysis (with integer capacities)

Important special case. Edge capacities are integers between 1 and $U$.

Invariant. The flow is integral throughout Ford–Fulkerson.

Pf. [by induction]
- Bottleneck capacity is an integer.
- Flow on an edge increases/decreases by bottleneck capacity. ■

Proposition. Number of augmentations $\leq$ the value of the maxflow.

Pf. Each augmentation increases the value by at least 1. ■

Integrality theorem. There exists an integral maxflow.

Pf.
- Proposition + Augmenting path theorem $\Rightarrow$ FF terminates with maxflow.
- Proposition + Invariant $\Rightarrow$ FF terminates with an integral flow. ■
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.

![Network diagram showing a bad case for Ford–Fulkerson]

- Initialize with 0 flow.
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.

**2nd augmenting path**
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.

---

*3rd augmenting path*
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.

![Diagram](attachment:image.png)
Bad news. Even when edge capacities are integers, number of augmenting paths could be very large.
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.

![Graph showing the 199th augmenting path](image)
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.

![Diagram](image_url)

*200th augmenting path*
Bad case for Ford–Fulkerson

**Bad news.** Even when edge capacities are integers, number of augmenting paths could be very large.

exponential in input size

\((V, E, \log U)\)
How to choose augmenting paths?

Good news. Clever choices lead to efficient algorithms.

<table>
<thead>
<tr>
<th>augmenting path</th>
<th>number of paths</th>
<th>implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortest path (fewest edges)</td>
<td>( \leq \frac{1}{2} EV )</td>
<td>queue (BFS)</td>
</tr>
<tr>
<td>fattest path (max bottleneck capacity)</td>
<td>( \leq E \ln(EU) )</td>
<td>priority queue</td>
</tr>
</tbody>
</table>

Flow network with V vertices, E edges, and integer capacities between 1 and U

---

Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems

JACK EDMONDS  
University of Waterloo, Waterloo, Ontario, Canada  
AND  
RICHARD M. KARP  
University of California, Berkeley, California

**Abstract.** This paper presents new algorithms for the maximum flow problem, the Hitchcock transportation problem, and the general minimum-cost flow problem. Upper bounds on the numbers of steps in these algorithms are derived, and are shown to compare favorably with upper bounds on the numbers of steps required by earlier algorithms.

---

**Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation**

E. A. Dинич

Different variants of the formulation of the problem of maximal stationary flow in a network and its many applications are given in [1]. There is also given an algorithm solving the problem in the case where the initial data are integers (or, what is equivalent, commensurable). In the general case this algorithm requires preliminary rounding off of the initial data, i.e. only an approximate solution of the problem is possible. In this connection the rapidity of convergence of the algorithm is inversely proportional to the relative precision.

---

Edmonds–Karp 1972 (USA)  
Dinic 1970 (Soviet Union)
6.4 Maximum Flow

- introduction
- Ford–Fulkerson algorithm
- maxflow–mincut theorem
- analysis of running time
- Java implementation
- applications
Maxflow and mincut applications

Maxflow/mincut is a widely applicable problem-solving model.

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

**Problem.** Given \( n \) people and \( n \) tasks, assign the tasks to people so that:

- Every task is assigned to a *qualified* person.
- Every person is assigned to exactly one task.
Problem. Given a bipartite graph, find a perfect matching (if one exists).
Maxflow formulation of bipartite matching

- Create $s$, $t$, one vertex for each task, and one vertex for each person.
- Add edge from $s$ to each task (of capacity 1).
- Add edge from each person to $t$ (of capacity 1).
- Add edge from task to qualified person (of infinite capacity).

**Flow network**

The diagram shows a flow network with vertices representing tasks and people, along with directed edges indicating assignments. The interpretation of flow on edge $4 \rightarrow 5'$ as 1 means assigning task 4 to person 5'.
Maxflow formulation of bipartite matching

1–1 correspondence between perfect matchings in bipartite graph and integral flows of value $n$ in flow network.

Integrality theorem + 1–1 correspondence $\Rightarrow$ Maxflow formulation is correct.
How many augmentations does the Ford–Fulkerson algorithms make to find a perfect matching in a bipartite graph with $n$ vertices per side?

A. $n$
B. $n^2$
C. $n^3$
D. $n^4$
(Yet another) holy grail for theoretical computer scientists.

<table>
<thead>
<tr>
<th>year</th>
<th>method</th>
<th>worst case</th>
<th>discovered by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>simplex</td>
<td>$E^3 U$</td>
<td>Dantzig</td>
</tr>
<tr>
<td>1955</td>
<td>augmenting path</td>
<td>$E^2 U$</td>
<td>Ford–Fulkerson</td>
</tr>
<tr>
<td>1970</td>
<td>shortest augmenting path</td>
<td>$E^3$</td>
<td>Dinitz, Edmonds–Karp</td>
</tr>
<tr>
<td>1970</td>
<td>fattest augmenting path</td>
<td>$E^2 \log E \log(EU)$</td>
<td>Dinitz, Edmonds–Karp</td>
</tr>
<tr>
<td>1977</td>
<td>blocking flow</td>
<td>$E^{5/2}$</td>
<td>Cherkasky</td>
</tr>
<tr>
<td>1978</td>
<td>blocking flow</td>
<td>$E^{7/3}$</td>
<td>Galil</td>
</tr>
<tr>
<td>1983</td>
<td>dynamic trees</td>
<td>$E^2 \log E$</td>
<td>Sleator–Tarjan</td>
</tr>
<tr>
<td>1985</td>
<td>capacity scaling</td>
<td>$E^2 \log U$</td>
<td>Gabow</td>
</tr>
<tr>
<td>1997</td>
<td>length function</td>
<td>$E^{3/2} \log E \log U$</td>
<td>Goldberg–Rao</td>
</tr>
<tr>
<td>2012</td>
<td>compact network</td>
<td>$E^2 / \log E$</td>
<td>Orlin</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>$E$</td>
<td>?</td>
</tr>
</tbody>
</table>

Maxflow algorithms for sparse networks with $E$ edges, integer capacities between 1 and $U$. 
Maximum flow algorithms: practice

**Warning.** Worst-case order-of-growth is generally not useful for predicting or comparing maxflow algorithm performance in practice.

**Best in practice.** Push–relabel method with gap relabeling: $E^{3/2}$.

**Computer vision.** Specialized algorithms for problems with special structure.

---

On Implementing Push-Relabel Method for the Maximum Flow Problem

Boris V. Cherkassky\(^1\) and Andrew V. Goldberg\(^2\)

\(^1\) Central Institute for Economics and Mathematics, Krasikova St. 32, 117418, Moscow, Russia
cer@cemi.msk.su
\(^2\) Computer Science Department, Stanford University
Stanford, CA 94305, USA
goldberg@cs.stanford.edu

**Abstract.** We study efficient implementations of the push-relabel method for the maximum flow problem. The resulting codes are faster than the previous codes, and much faster on some problem families. The speedup is due to the combination of heuristics used in our implementations. We also exhibit a family of problems for which the running time of all known methods seem to have a roughly quadratic growth rate.
Summary

**Min-cut problem.** Find an $st$-cut of minimum capacity.

**Maxflow problem.** Find an $st$-flow of maximum value.

**Duality.** Value of the maxflow = capacity of mincut.

**Proven successful approaches.**
- Ford–Fulkerson (various augmenting-path strategies).
- Preflow–push (various versions).

**Open research challenges.**
- Practice: solve real-world maxflow/mincut problems in linear time.
- Theory: prove it for worst-case inputs.
- Still much to be learned!