CSE 421: Introduction to Algorithms

Network Flow

Paul Beame
Bipartite Matching

- **Given:** A bipartite graph \( G=(V,E) \)
  - \( M \subseteq E \) is a matching in \( G \) iff no two edges in \( M \) share a vertex

- **Goal:** Find a matching \( M \) in \( G \) of maximum possible size
Bipartite Matching
The Network Flow Problem

- How much stuff can flow from s to t?
Bipartite matching as a special case of flow

Can always send at least $|M|$ flow to $N$.
Net Flow: Formal Definition

Given:
A digraph $G = (V, E)$
Two vertices $s, t$ in $V$ (source & sink)
A capacity $c(u,v) \geq 0$ for each $(u,v) \in E$
(and $c(u,v) = 0$ for all non-edges $(u,v)$)

Find:
A flow function $f: E \to \mathbb{R}$ s.t., for all $u,v$:
- $0 \leq f(u,v) \leq c(u,v)$ [Capacity Constraint]
- if $u \neq s,t$, i.e. $f^{out}(u) = f^{in}(u)$ [Flow Conservation]

Maximizing total flow $\nu(f) = f^{out}(s)$

Notation:
\[
\begin{align*}
    f^{in}(v) &= \sum_{e=(u,v) \in E} f(u,v) \\
    f^{out}(v) &= \sum_{e=(v,w) \in E} f(v,w)
\end{align*}
\]
Example: A Flow Function

\[ f^{\text{in}}(u) = f(s, u) = 2 = f(u, t) = f^{\text{out}}(u) \]

flow/capacity, not .66...
Example: A Flow Function

- Not shown: \( f(u,v) \) if \( f = 0 \)
- Note: max flow \( \geq 4 \) since 
  \( f \) is a flow function, with \( \nu(f) = 4 \)
Max Flow via a Greedy Alg?

While there is an $s \rightarrow t$ path in $G$
  
  Pick such a path, $p$  
  
  Find $c$, the min capacity of any edge in $p$  
  
  Count $c$ towards the flow value  
  
  Subtract $c$ from all capacities on $p$  
  
  Delete edges of capacity 0
Max Flow via a Greedy Alg?

While there is an $s \rightarrow t$ path in $G$
  Pick such a path, $p$
  Find $c$, the min capacity of any edge in $p$
  Count $c$ towards the flow value
  Subtract $c$ from all capacities on $p$
  Delete edges of capacity 0

- This does NOT always find a max flow:

  If pick $s \rightarrow b \rightarrow a \rightarrow t$
  first, flow stuck at 2.
  But flow 3 possible.
## A Brief History of Flow

<table>
<thead>
<tr>
<th>#</th>
<th>year</th>
<th>discoverer(s)</th>
<th>bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1951</td>
<td>Dantzig</td>
<td>(O(n^2mU))</td>
</tr>
<tr>
<td>2</td>
<td>1955</td>
<td>Ford &amp; Fulkerson</td>
<td>(O(nmU))</td>
</tr>
<tr>
<td>3</td>
<td>1970</td>
<td>Dinitz</td>
<td>(O(nm^2)) (= O(n^2m))</td>
</tr>
<tr>
<td>4</td>
<td>1970</td>
<td>Edmonds &amp; Karp</td>
<td>(O(n^2m))</td>
</tr>
<tr>
<td>5</td>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>(O(m^2 \log U))</td>
</tr>
<tr>
<td>6</td>
<td>1973</td>
<td>Dinitz</td>
<td>(O(nm \log U))</td>
</tr>
<tr>
<td>7</td>
<td>1973</td>
<td>Gabow</td>
<td>(O(nm \log U))</td>
</tr>
<tr>
<td>8</td>
<td>1974</td>
<td>Karzanov</td>
<td>(O(n^3))</td>
</tr>
<tr>
<td>9</td>
<td>1977</td>
<td>Cherkassky</td>
<td>(O(n^2 \sqrt{m}))</td>
</tr>
<tr>
<td>10</td>
<td>1980</td>
<td>Galil &amp; Naamad</td>
<td>(O(nm \log^2 n))</td>
</tr>
<tr>
<td>11</td>
<td>1983</td>
<td>Sleator &amp; Tarjan</td>
<td>(O(nm \log n))</td>
</tr>
<tr>
<td>12</td>
<td>1986</td>
<td>Goldberg &amp; Tarjan</td>
<td>(O(nm \log(n^2/m)))</td>
</tr>
<tr>
<td>13</td>
<td>1987</td>
<td>Ahuja &amp; Orlin</td>
<td>(O(nm + n^2 \log U))</td>
</tr>
<tr>
<td>14</td>
<td>1987</td>
<td>Ahuja et al.</td>
<td>(O(nm \log(n \sqrt{\log U} / (m+2))))</td>
</tr>
<tr>
<td>15</td>
<td>1989</td>
<td>Cheriyan &amp; Hagerup</td>
<td>(E(nm + n^2 \log^2 n))</td>
</tr>
<tr>
<td>16</td>
<td>1990</td>
<td>Cheriyan et al.</td>
<td>(O(n^2 / \log n))</td>
</tr>
<tr>
<td>17</td>
<td>1990</td>
<td>Alon</td>
<td>(O(nm + n^{8/3} \log n))</td>
</tr>
<tr>
<td>18</td>
<td>1992</td>
<td>King et al.</td>
<td>(O(nm + n^{2+\epsilon}))</td>
</tr>
<tr>
<td>19</td>
<td>1993</td>
<td>Phillips &amp; Westbrook</td>
<td>(O(nm(\log_{m/n} n + \log^{2+\epsilon} n)))</td>
</tr>
<tr>
<td>20</td>
<td>1994</td>
<td>King et al.</td>
<td>(O(nm \log_{m/(n \log n)} n))</td>
</tr>
<tr>
<td></td>
<td>1997</td>
<td>Goldberg &amp; Rao</td>
<td>(O(m^{3/2} \log(n^2/m) \log U)) (= O(n^{2/3} m \log(n^2/m) \log U))</td>
</tr>
<tr>
<td></td>
<td>2012</td>
<td>Orlin + King et al.</td>
<td>(O(nm))</td>
</tr>
</tbody>
</table>

\(n = \# \text{ of vertices}\)
\(m = \# \text{ of edges}\)
\(U = \text{Max capacity}\)

Source: Goldberg & Rao, FOCS '97
Greed Revisited: Residual Graph & Augmenting Path
Greed Revisited: Residual Graph & Augmenting Path

Residual Graph
Greed Revisited: Residual Graph & Augmenting Path

Residual Graph
Greed Revisited: Residual Graph & Augmenting Path

Residual Graph
Greed Revisited: Residual Graph & Augmenting Path

Residual Graph
Greed Revisited: An Augmenting Path

New Residual Graph

no path from $s$ to $t$
Residual Capacity

- The residual capacity (w.r.t. $f$) of $(u,v)$ is $c_f(u,v) = c(u,v) - f(u,v)$ if $f(u,v) \leq c(u,v)$ and $c_f(u,v) = f(v,u)$ if $f(v,u) > 0$

  - e.g. $c_f(s,b) = 7$; $c_f(a,x) = 1$; $c_f(x,a) = 3$
Residual Graph & Augmenting Paths

- The *residual graph* (w.r.t. \( f \)) is the graph \( G_f = (V,E_f) \), where
  \[
  E_f = \{ (u,v) \mid c_f(u,v) > 0 \}
  \]
  - Two kinds of edges
    - Forward edges
      - \( f(u,v) < c(u,v) \) so \( c_f(u,v) = c(u,v) - f(u,v) > 0 \)
    - Backward edges
      - \( f(u,v) > 0 \) so \( c_f(v,u) = f(u,v) > 0 \)
- An *augmenting path* (w.r.t. \( f \)) is a simple \( s \rightarrow t \) path in \( G_f \).
A Residual Network

c(s,a) = 5
f(s,a) = 4
c_f(s,a) = c(s,a) - f(s,a) = 5 - 4 = 1
f_c(a,s) = f(s,a) = 4

\( g, f \)
An Augmenting Path
Augmenting A Flow

```
augment(f,P)
  c_P <-- \min_{(u,v) \in P} c_f(u,v)  "bottleneck(P)"
  for each e \in P
    if e is a forward edge then
      increase f(e) by c_P
    else (e is a backward edge)
      decrease f(e') by c_P where e' = reverse of e
    endif
 .endfor
.return(f)
```
Augmenting A Flow
Claim: Augmented flow is legal

If $G_f$ has an augmenting path $P$, then the function $f' = \text{augment}(f, P)$ is a legal flow.

Proof:

- $f'$ and $f$ differ only on the edges of $P$ so only need to consider such edges $(u,v)$.
Proof: Augmented flow is legal

- If \((u,v)\) is a forward edge then
  \[ f'(u,v) = f(u,v) + c_p \leq f(u,v) + c_f(u,v) \]
  \[ = f(u,v) + c(u,v) - f(u,v) \]
  \[ = c(u,v) \]

- If \((u,v)\) is a backward edge then \(f\) and \(f'\) differ on flow along \((v,u)\) instead of \((u,v)\)
  \[ f'(v,u) = f(v,u) - c_p \geq f(v,u) - c_f(u,v) \]
  \[ = f(v,u) - f(v,u) = 0 \]

- Other conditions like flow conservation still met
Ford-Fulkerson Method

Start with \( f = 0 \) for every edge

While \( G_f \) has an augmenting path, augment

Questions:

- Does it halt?
- Does it find a maximum flow?
- How fast?
Observations about Ford-Fulkerson Algorithm

- At every stage the capacities and flow values are always integers (if they start that way)

- The flow value \( \nu(f') = \nu(f) + c_P > \nu(f) \) for \( f' = \text{augment}(f, P) \)
  - Since edges of residual capacity 0 do not appear in the residual graph

- Let \( C = \sum_{(s,u) \in E} c(s,u) \)
  - \( \nu(f) \leq C \)
  - F-F does at most \( C \) rounds of augmentation since flows are integers and increase by at least 1 per step
Running Time of Ford-Fulkerson

- For $f=0$, $G_f=G$
- Finding an augmenting path in $G_f$ is graph search $O(n+m)=O(m)$ time
- Augmenting and updating $G_f$ is $O(n)$ time
- Total $O(mC)$ time
- Does it find a maximum flow?
  - Need to show that for every flow $f$ that isn’t maximum $G_f$ contains an $s$-$t$-path
Cuts

- A partition \((A, B)\) of \(V\) is an \(s-t\)-cut if
  - \(s \in A, \ t \in B\)
- Capacity of cut \((A, B)\) is \(c(A, B) = \sum_{u \in A \atop v \in B} c(u, v)\)
Convenient Definition

- $f_{out}(A) = \sum_{v \in A, w \not\in A} f(v, w)$
- $f_{in}(A) = \sum_{v \in A, u \not\in A} f(u, v)$
**Two claims**

- For any flow $f$ and any cut $(A, B)$,
  1) the net flow across the cut equals the total flow, i.e., $\nu(f) = f_{out}(A) - f_{in}(A)$, and
  2) the net flow across the cut cannot exceed the capacity of the cut, i.e. $f_{out}(A) - f_{in}(A) \leq c(A, B)$

- **Corollary:**
  Max flow $\leq$ Min cut
Proof of Claim 1

- Consider a set $A$ with $s \in A$, $t \notin A$

- $f_{\text{out}}(A) - f_{\text{in}}(A) = \sum_{v \in A, w \notin A} f(v, w) - \sum_{v \in A, u \notin A} f(u, v)$

- We can add flow values for edges with both endpoints in $A$ to both sums and they would cancel out so

- $f_{\text{out}}(A) - f_{\text{in}}(A) = \sum_{v \in A} (\sum_{w \in V} f(v, w) - \sum_{u \in V} f(u, v))$

- $= \sum_{v \in A} (f_{\text{out}}(v) - f_{\text{in}}(v))$

- $= f_{\text{out}}(s) - f_{\text{in}}(s)$

- since all other vertices have $f_{\text{out}}(v) = f_{\text{in}}(v)$

- $v(f) = f_{\text{out}}(s)$ and $f_{\text{in}}(s) = 0$
Proof of Claim 2

\[ v(f) = f^{\text{out}}(A) - f^{\text{in}}(A) \geq 0 \]

\[ \leq f^{\text{out}}(A) \]

\[ = \sum_{v \in A, w \notin A} f(v, w) \]

\[ \leq \sum_{v \in A, w \notin A} c(v, w) \]

\[ \leq \sum_{v \in A, w \in B} c(v, w) \]

\[ = c(A, B) \]

\[ f^{\text{out}}(A) \leq c(A | B) \]
Max Flow / Min Cut Theorem

Claim 3 For any flow $f$, if $G_f$ has no augmenting path then there is some $s$-$t$-cut $(A,B)$ such that $\nu(f)=c(A,B)$ (proof on next slide)

- We know by Claims 1 & 2 that any flow $f'$ satisfies $\nu(f') \leq c(A,B)$ and we know that F-F runs for finite time until it finds a flow $f$ satisfying conditions of Claim 3
  - Therefore by Claim 3 for any flow $f'$, $\nu(f') \leq \nu(f)$

- Theorem (a) F-F computes a maximum flow in $G$
- (b) For any graph $G$, the value $\nu(f)$ of a maximum flow = minimum capacity $c(A,B)$ of any $s$-$t$-cut in $G$
Claim 3

Let \( A = \{ u \mid \exists \text{ an path in } G_f \text{ from } s \text{ to } u \} \)
\( B = V - A; \ s \in A, t \in B \)

This is true for every edge crossing the cut, i.e.
\[
\begin{align*}
\text{flow out} (A) &= \sum_{u \in A} \sum_{v \in B} f(u,v) = c(u,v) = c(A,B) \\
\text{flow in} (A) &= 0 \text{ so } c(A,B) = c(A,B) - c(A,B) = 0
\end{align*}
\]
Flow Integrality Theorem

If all capacities are integers

- The max flow has an integer value
- Ford-Fulkerson method finds a max flow in which $f(u,v)$ is an integer for all edges $(u,v)$
Corollaries & Facts

- If Ford-Fulkerson terminates, then it’s found a max flow.
- It will terminate if $c(e)$ integer or rational (but may not if they’re irrational).
- However, may take exponential time, even with integer capacities:

```
c = 10^9, say
```
Bipartite matching as a special case of flow

Integer flows implies each flow is just a subset of the edges

Therefore flow corresponds to a matching

$O(mC) = O(nm)$ running time
Capacity-Scaling algorithm

- General idea:
  - Choose augmenting paths \( P \) with ‘large’ capacity \( c_P \)
  - Can augment flows along a path \( P \) by any amount \( \Delta \leq c_P \)
  - Ford-Fulkerson still works
  - Get a flow that is maximum for the high-order bits first and then add more bits later
Capacity Scaling
Capacity Scaling
Capacity Scaling Bit 1

Capacity on each edge is at most 1 (either 0 or 1 times $\Delta=4$)
Capacity Scaling Bit 1

\[ O(nm) \text{ time} \]
Residual capacity across min cut is at most $m$ (either 0 or 1 times $\Delta=2$)

# of steps $O(m^2)$
Residual capacity across min cut is at most $m$

$\Rightarrow \leq m$ augmentations
Residual capacity across min cut is at most $m$ (either 0 or 1 times $\Delta=1$)
After $\leq m$ augmentations
Capacity Scaling Final

```
5/5  2/4  3/3  2/3
s----a----x----t
|    |    |    |
5/7 4/4 7/7
b----y----z
|    |    |
5/6 1/1 4
|    |
c----z
```

Capacity Scaling Min Cut
Total time for capacity scaling

- $\log_2 U$ rounds where $U$ is largest capacity
- At most $m$ augmentations per round
  - Let $c_i$ be the capacities used in the $i^{th}$ round and $f_i$ be the maxflow found in the $i^{th}$ round
    - For any edge $(u,v)$, $c_{i+1}(u,v) \leq 2c_i(u,v) + 1$
    - $i+1^{st}$ round starts with flow $f = 2f_i$
    - Let $(A,B)$ be a min cut from the $i^{th}$ round
      - $\nu(f_i) = c_i(A,B)$ so $\nu(f) = 2c_i(A,B)$
      - $\nu(f_{i+1}) \leq c_{i+1}(A,B) \leq 2c_i(A,B) + m = \nu(f) + m$

- $O(m)$ time per augmentation
- Total time $O(m^2 \log U)$
Edmonds-Karp Algorithm

- Use a shortest augmenting path (via Breadth First Search in residual graph)

- Time: $O(n m^2)$

\[ \text{not care analysis} \]
Distance from $s$ in $G_f$ is never reduced by:

- **Deleting an edge**
  
  **Proof:** no new (hence no shorter) path created

- **Adding an edge** $(u,v)$, provided $v$ is nearer than $u$
  
  **Proof:** BFS is unchanged, since $v$ visited before $(u,v)$ examined
Key Lemma

Let $f$ be a flow, $G_f$ the residual graph, and $P$ a shortest augmenting path. Then no vertex is closer to $s$ after augmentation along $P$.

Proof: Augmentation along $P$ only deletes forward edges, or adds back edges that go to previous vertices along $P$. 
Augmentation vs BFS

G:

Gf

Gf'
Theorem

The Edmonds-Karp Algorithm performs $O(mn)$ flow augmentations

Proof:

Call $(u,v)$ critical for augmenting path $P$ if it’s closest to $s$ having min residual capacity.

It will disappear from $G_f$ after augmenting along $P$.

In order for $(u,v)$ to be critical again the $(u,v)$ edge must re-appear in $G_f$ but that will only happen when the distance to $u$ has increased by $2$ (next slide).

It won’t be critical again until farther from $s$ so each edge critical at most $n/2$ times.
Critical Edges in $G_f$

Shortest s-t path $P$ in $G_f$

For $(u,v)$ to be critical later for some flow $f'$ it must be in $G_f$, so must have augmented along a shortest path containing $(v,u)$

Then we must have $d_{f'}(s,u) = d_{f'}(s,v) + 1 \geq d_f(s,v) + 1 = d_f(s,u) + 2$
Corollary

- Edmonds-Karp runs in $O(nm^2)$ time
Applications of Network Flow

1. Edge-Disjoint Paths in Graphs:
   - Given vertices $s$ and $t$ in $G$,
     find as many simple paths as possible
     from $s$ to $t$ that don't share any edge.

(a) Directed Graphs:

- Ignore
- Ignore
- $\text{cut} C \leq n-1$
- Capacity 1 on all edges
- Compute a max flow using Ford-Fulkerson
  time $O(mn)$

**Suppose max flow value is $K$**
- F-F all flow an integer
- flow $\leq$ sum of edge values
  $\text{flow} = 1$
- $\text{node deg} = \text{out deg}$
- Only draw edges with flow
- $s$

- $n$
Undirected graph $G$.

John is a member of a club of edge weights.

Check if the graph is a cycle using

\[ x = \text{min} \setminus \text{of edge weights}. \]

If true, then it is a cycle.

\[ 0 = \text{the cycle value}. \]

Find the edge $e$ with $\text{min} \setminus \text{of edge weights}$.

Delete edge $e$. When deleted, check if the graph is a cycle.

If not a cycle, then add $e$. Repeat until $x = 0$.

\[ n \text{ is the cut of a cycle in graph.} \]
Solve \( u \rightarrow v \rightarrow \cdots \) run FF
and thus directed single path leading never get both
in a single path.

Menger's for Undirected Graphs = 
min \# of edges whose deletion disconnects \( s \) and \( t \)
= min \# of edges whose deletion disconnects \( s \) and \( t \).
Directed Network with suppliers and consumers. 

Capacities on edges \( u \) vertex: either a supplier or a consumer.

Consumer: demand \( d_v \geq 0 \)

Supplier: demand \( d_v < 0 \)

Supply \(-d_v\) with.

\(-5\) \(3\) \(-4\)

Circulation with demand: \( d_v = f^{in}(v) - f^{out}(v) \)

Can we meet all the demand, i.e., suppliers send out all their supply and consumers get all their needs filled?

Need: \( \sum d_v = 0 \)
Compute maxflow

FF alg with actual compute values
if \( D = D \) yet get circulants
\( d < D \) can't satisfy
Project Selection
a.k.a. The Strip Mining Problem

Given

- a directed acyclic graph $G=(V,E)$ representing precedence constraints on tasks (a task points to its predecessors)
- a profit value $p(v)$ associated with each task $v \in V$ (may be positive or negative)

Find

- a set $A \subseteq V$ of tasks that is closed under predecessors, i.e. if $(u,v) \in E$ and $u \in A$ then $v \in A$, that maximizes $Profit(A) = \sum_{v \in A} p(v)$
Project Selection Graph

Each task points to its predecessor tasks
Extended Graph $G'$

For each vertex $v$

If $p(v) \geq 0$ add $(s,v)$ edge with capacity $p(v)$

If $p(v) < 0$ add $(v,t)$ edge with capacity $-p(v)$
Extended Graph $G'$

- Want to arrange capacities on edges of $G$ so that for minimum $s$-$t$-cut $(S,T)$ in $G'$, the set $A=S\backslash \{s\}$
  - satisfies precedence constraints
  - has maximum possible profit in $G$

- Cut capacity with $S=\{s\}$ is just $C=\sum_{v: p(v) \geq 0} p(v)$
  - $\text{Profit}(A) \leq C$ for any set $A$

- To satisfy precedence constraints don’t want any original edges of $G$ going forward across the minimum cut
  - That would correspond to a task in $A=S\backslash \{s\}$ that had a predecessor not in $A=S\backslash \{s\}$

- Set capacity of each of the edges of $G$ to $C+1$
  - The minimum cut has size at most $C$
Extended Graph $G'$

Cut value

$= 13 + 3 + 2 + 3 + 4$

$= 13 + 3$

$+ C - 4 - 8 - 10 - 11 - 12 - 14$
Project Selection

**Claim** Any $s$-$t$-cut $(S,T)$ in $G'$ such that $A=S\setminus\{s\}$ satisfies precedence constraints has capacity

$$c(S,T)=C - \sum_{v\in A} p(v) = C - \text{Profit}(A)$$

**Corollary** A minimum cut $(S,T)$ in $G'$ yields an optimal solution $A=S\setminus\{s\}$ to the profit selection problem

**Algorithm** Compute maximum flow $f$ in $G'$, find the set $S$ of nodes reachable from $s$ in $G'_f$ and return $S\setminus\{s\}$
Proof of Claim

- \( A=S-\{s\} \) satisfies precedence constraints
  - No edge of \( G \) crosses forward out of \( A \) since those edges have capacity \( C+1 \)
  - Only forward edges cut are of the form \((v,t)\) for \( v \in A \) or \((s,v)\) for \( v \notin A \)
  - The \((v,t)\) edges for \( v \in A \) contribute
    \[
    \sum_{v \in A: p(v)<0} -p(v) = - \sum_{v \in A: p(v)<0} p(v)
    \]
  - The \((s,v)\) edges for \( v \notin A \) contribute
    \[
    \sum_{v \notin A: p(v)\geq0} p(v) = C - \sum_{v \in A: p(v)\geq0} p(v)
    \]
  - Therefore the total capacity of the cut is
    \[
    c(S,T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A)
    \]