Maximum Flow Problem

Network $G = (V, E)$, source $s$, sink $t$

edge capacities $u(v, w)$ for $(v, w) \in E$

$|V| = n$ $|E| = m$ $U = \max |u(v, w)|$

Assume network is symmetric:

$(v, w) \in E$ iff $(w, v) \in E$

Flow $f : E \rightarrow \mathbb{R}$

$f(v, w) \leq u(v, w)$

$f(v, w) = -f(w, v)$

c(w) = \sum_{v} f(v, w) = 0 \forall W \in \{s, t\}$

Objective: maximize $c(t) (\geq -c(s))$
Techniques

Iterative Improvement:
locally modify the current solution to improve it.

Successive Approximation:
solve successively closer approximations of the original problem, using each solution as a starting point for the next problem.

Data Structures:
represent relevant information about the current flow in an appropriate way.
Ford-Fulkerson Approach:

Find an augmenting path, push flow along it, repeat.

Residual capacities

\[ u_f(v,w) = u(v,w) - f(v,w) \]

\((v,w)\) is residual if \(u_f(v,w) > 0\)

An augmenting path is a path of residual edges from \(s\) to \(t\).
(Bad) Example

\[ \text{Diagram with nodes labeled } S, 1, \text{ and } T, \text{ and edges labeled } 100, 100, \text{ and } 100. \]
etcetera
Edmonds & Karp: always augment along a shortest (fewest edges) path:

\[ O(m) \text{ time per path} \times O(m) \text{ paths per length} \times O(n) \text{ path lengths} = O(nm^2) \text{ time} \]

Dinic: find all augmenting paths of a given length at once, in a phase:

\[ O(n) \text{ time per path} \times O(nm) \text{ paths} + O(m) \text{ time per phase} \times O(n) \text{ phases} = O(n^2m) \text{ time} \]
Classical Algorithms

<table>
<thead>
<tr>
<th>Date</th>
<th>Discoverer</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>Ford &amp; Fulkerson</td>
<td>$O(nmU)$</td>
</tr>
<tr>
<td>1969</td>
<td>Edmonds &amp; Karp</td>
<td>$O(n^2m)$</td>
</tr>
<tr>
<td>1970</td>
<td>Dinic</td>
<td>$O(n^k_m)$</td>
</tr>
<tr>
<td>1974</td>
<td>Karzanov</td>
<td>$O(n^*_i)$ (same bound by several others later)</td>
</tr>
<tr>
<td>1977</td>
<td>Cherkasny</td>
<td>$O(n^2m^{1/2})^*$</td>
</tr>
<tr>
<td>1978</td>
<td>Galil</td>
<td>$O(n^{5/3}m^{2/3})^*$</td>
</tr>
<tr>
<td>1978</td>
<td>Galil &amp; Naamad; Shiloach</td>
<td>$O(nm(\log n)^2)$</td>
</tr>
<tr>
<td>1980</td>
<td>Sleator &amp; Tarjan</td>
<td>$O(nm \log n)$</td>
</tr>
<tr>
<td>1983</td>
<td>Gabow</td>
<td>$O(nm \log U)$</td>
</tr>
</tbody>
</table>

*Forrunners of prior push method*
Preflow Push Approach (Goldberg)

Two ideas:

Make the basic steps in the computation smaller
(relax the flow conservation requirement)

Use a less global, more distributed approach to
do the preprocessing associated with each phase

Main effect: simpler algorithm?
Presflow (Karzanov): like a flow except that the total flow into a vertex can exceed the total flow out.

A vertex $v$ with extra incoming flow is active. The net incoming flow $e(v)$ is the excess of vertex $v$.

Idea: Move flow excess toward sink along estimated shortest paths. Move excess that cannot reach the sink back to the source, also along estimated shortest paths.

To estimate path lengths: a valid labeling is an integer function $d$ on vertices such that:

(i) $d(t) = 0$
(ii) $d(s) = n$
(iii) $d(v) = d(w) + 1$ if $U_f(v,w) > 0$

$d(v)$ is a lower bound on the minimum of distance to $t$, $n$ + distance to $s$. 
Algorithm

1. Saturate all edges leaving s. Choose initial s.
2. Repeat push and relabel steps in any order until no vertex is active.

\text{push}\,(v,w):
\begin{align*}
&\text{if } v \text{ is active, } u_f(v,w) > 0, \text{ and } d(v) = d(w) + 1 \\
&\text{then move } \min\{e(v), u_f(v,w)\} \text{ units of flow from } v \text{ to } w \text{ (the push is saturating if } u_f(v,w) \text{ units are moved)}
\end{align*}

\text{relabel}\,(v):
\begin{align*}
&\text{if } v \text{ is active and for all } (v,w), u_f(v,w) = 0 \text{ or } d(v) \leq 0 \\
&\text{then let } d(v) = \min\{d(w)+1, u_f(v,w) > 0\}
\end{align*}
Bounds

Every active vertex has a label of at most $2n-1$:

there is always a residual path to $s$.

$\Rightarrow O(n^2)$ relabelings, taking $O(nm)$ time.

Between saturating pushes through the same edge, ends of edge must be relabeled

$\Rightarrow O(nm)$ saturating pushes.

The rest of the analysis is in bounding the number of non-saturating pushes.
Generic Bound: $O(n^2m)$

Pf. Define $\Phi = \sum_{v \text{ active}} d(v)$.

$0 \leq \Phi \leq 2n^2$. A nonsaturating push decreases $\Phi$ by one.

Increases to $\Phi$: $O(n^2)$ in total due to relabelings.

$O(n^3m)$ due to saturating pushes:

$O(n)$ per saturating push.

$\Rightarrow O(n^2m)$ nonsaturating pushes.
FIFO Method

Maintain a queue of active vertices.
Always push from the vertex on the front of the queue.
Add newly active vertices to the rear of the queue.

Analysis

Phases: phase 1 = processing of vertices originally on queue.
        phase i+1 = processing of vertices added to queue
                 during phase i.

Only one non-saturating push per vertex per phase:
        such a push reduces the excess to zero and
        removes the vertex from the queue.
$O(n^2)$ bound on # phases

Define $\Phi = \max_{v \in \text{active}} d(v)$, $0 \leq \Phi \leq 2n$.

A phase reduces $\Phi$ by one unless a relabeling occurs.

All increase in $\Phi$ is due to relabelings, totals $O(n^2)$.

The number of phases in which $\Phi$ doesn't change is also $O(n^2)$

$\Rightarrow O(n^3)$ total phases.

$\Rightarrow O(n^3)$ nonsaturating pushes.
Ahuja–Orlin Excess Scaling

Maintain $\Delta$, an upper bound on max excess

Maintain integrality of flow.

After each phase, replace $\Delta$ by $\Delta/2$.

Stop when $\Delta < 1$.

Push from a vertex $v$ of smallest $d(v)$ with $e(v) > \Delta/2$.

When pushing from $v$ to $w \neq t$, move

$$\min \{ e(v), u_r(v,w), \Delta - e(w) \}$$
Analysis

Each nonsaturating push moves at least \( \Delta/2 \) units of flow.

Let \( \Phi = \sum_{v \text{ active}} e(v) d(v) / \Delta \)

\( 0 \leq \Phi \leq 2n^2 \)

Each nonsaturating push decreases \( \Phi \) by \( \geq 1/2 \).

Increases in \( \Phi \): \( O(n^2) \) associated with relabeling.

\( O(n^3) \) per phase from change in \( \Delta \).

\( O(\log U) \) phases \( \Rightarrow \)

\( O(n^2 \log U) \) nonsaturating pushes
saturating pushes = $O(nm)$
non-saturating pushes = $O(n^2 \log U)$

Can these estimates be balanced?

Yes: change algorithm: make all pushes large enough by retaining enough excess to immediately saturate very small-capacity edges.

# pushes = $O(n^{3/2} \sqrt{m} (\log U)^{1/2})$

Cheriyan–Mehlhorn

What about relabeling time??
<table>
<thead>
<tr>
<th>Date</th>
<th>Discoverer</th>
<th>Time</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>Goldberg</td>
<td>$O(n^3)$</td>
<td>FIFO</td>
</tr>
<tr>
<td>1987</td>
<td>Cherian &amp; Maheshwari</td>
<td>$O(n^2 m^{1/2})$</td>
<td>Max Distance</td>
</tr>
<tr>
<td>1986</td>
<td>Goldberg &amp; Tarjan</td>
<td>$O(nm \log(n^2/m))$</td>
<td>FIFO + Trees</td>
</tr>
<tr>
<td>1986</td>
<td>Ahuja &amp; Orlin</td>
<td>$O(nm + n^2 \log U)$</td>
<td>Excess Scaling</td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja &amp; Orlin</td>
<td>$O(nm + n^2 (\log U)^2)$</td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja, Orlin, &amp; Tarjan</td>
<td>$O(nm \log (n^{1/2} / (\log U)^{1/2}))$</td>
<td>Excess Scaling + Trees</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td>$O(nm)$</td>
<td>?</td>
</tr>
<tr>
<td>1989</td>
<td>Cherian &amp; Hagerup</td>
<td>$O(nm + n^2 (\log n)^2)$</td>
<td>Excess Scaling + Trees + Randomize</td>
</tr>
<tr>
<td></td>
<td>(improved)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Cherian &amp; Hagerup</td>
<td>$O(n^3 / \log n)$</td>
<td>+ Random Acci</td>
</tr>
<tr>
<td></td>
<td>+ MedHorn</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Practice

Appropriate versions of the preflow push method are easy to implement and very fast in practice: 4-14 times faster than Dinic on reasonable classes of graphs.

Important heuristic: periodically compute tight distance labels using breadth-first search. (Otherwise the relabeling time is too high.)

The FIFO algorithm can be parallelized: push from all active vertices at once. It seems to give drastic speedups in practice.

Whether dynamic trees help on very large graphs has not yet been studied.