

COS423
Spring '02

Maximum Flow Problem

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

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

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

Assume network is symmetric:

$$(v, w) \in E \text{ iff } (w, v) \in E$$

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

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

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

$$e(w) = \sum_v f(v, w) = 0 \quad \forall w \notin \{s, t\}$$

Objective: maximize $e(t)$ ($= -e(s)$)

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.

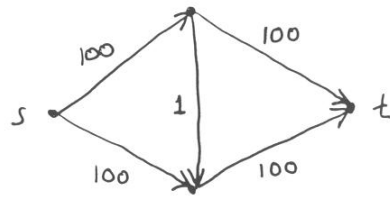
Residual capacities

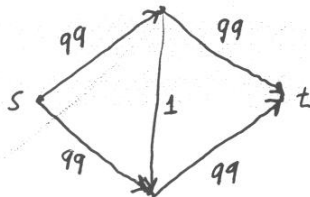
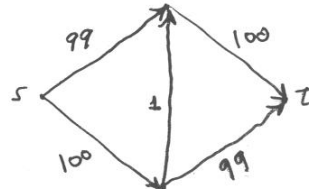
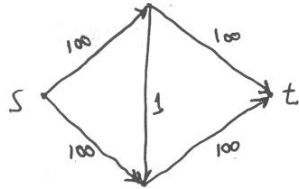
$$u_f(v,w) = u(v,w) - f(v,w)$$

(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





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

Date	Discoverer	Time
1956	Ford & Fulkerson	$O(nmU)$
1969	Edmonds & Karp	$O(nm^2)$
1970	Dinic	$O(n^2m)$
1974	Karzanov (same bound by several others later)	$O(n^3)^*$
1977	Cherkasky	$O(n^2 m^{1/2})^*$
1978	Galil	$O(n^{5/3} m^{2/3})^*$
1978	Galil & Naamad; Shiloach	$O(nm(\log n)^2)$
1980	Sleator & Tarjan	$O(nm \log n)$
1983	Gabow	$O(nm \log U)$

* Forerunners of preflow 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 algorithms!

Preflow (Karzanov): like a flow except that the total flow into a vertex can exceed the total flow out.

A vertex $v \neq t$ 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) $d(v) \leq d(w) + 1$ if $u_f(v, w) > 0$

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

Algorithm

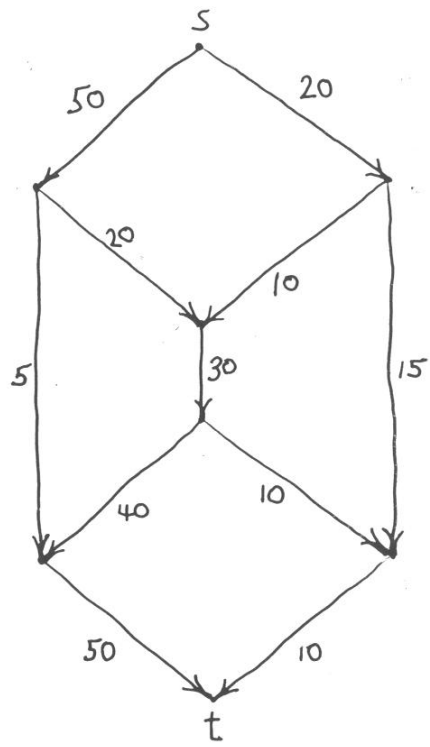
1. Saturate all edges leaving s . Choose initial d .
2. Repeat push and relabel steps in any order until no vertex is active.

push(v, w):

if v is active, $u_f(v, w) > 0$, and $d(v) = d(w) + 1$
then move $\min\{c(v, w), u_f(v, w)\}$ units of
flow from v to w (the push is saturating if
 $u_f(v, w)$ units are moved)

relabel(v):

if v is active and for all (v, w) , $u_f(v, w) = 0$ or $d(v) \leq d(w)$
then let $d(v) = \min\{d(w) + 1 \mid u_f(v, w) > 0\}$



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 heart of the analysis is in bounding
the number of nonsaturating pushes.

Generic Bound: $O(n^2 m)$

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

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

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

$O(n^2 m)$ due to saturating pushes:

$O(n)$ per saturating push.

$\Rightarrow O(n^2 m)$ 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 $\Phi = \max_{v \text{ active}} d(v)$. $0 \leq \Phi \leq 2n$.

A phase reduces Φ by one unless a relabeling occurs.

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

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

$\Rightarrow O(n^2)$ total phases.

$\Rightarrow O(n^3)$ nonsaturating pushes.

Ahuja-Orlin Excess Scaling

Maintain Δ , an upper bound on max excess

Maintain integrality of flow.

After each phase, replace Δ 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

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

Analysis

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

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

$$0 \leq \Phi \leq 2n^2$$

Each nonsaturating push decreases Φ by $\geq 1/2$.

Increases in Φ : $O(n^2)$ associated with relabeling.

$O(n^2)$ per phase from change in Δ .

$O(\log U)$ phases \Rightarrow

$O(n^2 \log U)$ nonsaturating pushes

saturating pushes = $O(nm)$

nonsaturating 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.

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

Cheriyán ~ Mehlhorn

What about relabeling time??

Preflow Push Algorithms

Date	Discoverer	Time	Method
1985	Goldberg	$O(n^3)$	FIFO
1987	Cheriyān & Maheshwari	$O(n^2 m^{1/2})$	Max Distance
1986	Goldberg & Tarjan	$O(nm \log(\frac{n^2}{m}))$	FIFO + Trees
1986	Ahuja & Orlin	$O(nm + n^2 \log U)$	Excess Scaling
1987	Ahuja & Orlin	$O(nm + n^2 (\log U)^{1/2})$	"
1987	Ahuja, Orlin, & Tarjan	$O(nm \log(\frac{n}{m} (\log U)^{1/2} + 2))$	Excess Scaling + Trees
?	?	$O(nm)$?
1989	Cheriyān & Hagerup (improved)	$O(nm + n^2 (\log n)^2)$	Excess Scaling + Trees + Randomiza
1989	Cheriyān & Hagerup + Mehlhorn	$\sim O(n^3 / \log n)$	+ Random Acce
?	?	?	?

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.