ORIE 633 Network Flows	September 4, 2007
Lecture 4	
Lecturer: Jon M. Kleinberg	*Scribe: Liaoruo Wang*

1 Polynomial-time algorithms for the max-flow problem

1.1 The push-relabel algorithm

So far we have considering augmenting path algorithms. These algorithms are primal feasible, because capacity constraints are obeyed and flow conservation constraints are obeyed. We maintain a feasible flow and work towards finding a maximum flow. But the next algorithm we will consider, "Push-Relabel", also known as "Preflow-Push", is primal infeasible, because it does not obey flow conservation constraints. Here we will maintain a flow that has value at least that of the maximum, and work towards finding a feasible flow. The algorithm will maintain a *preflow.*
Basic idea: Use preflows instead of flows.

Definition 1 *A preflow is a function $f : A \to \Re$ that obeys*

1) Capacity constraints: $f_{ij} \leqslant u_{ij}$

2) Antisymmetry constraints: $f_{ij} = -f_{ji}$

3) For $\forall i \in V$, $i \neq \{s,t\}$, $f^{in}(i) \geqslant f^{out}(i) \Rightarrow \sum_{j:(j,i)\in A} f_{ji} \geqslant \sum_{k:(i,k)\in A} f_{ik}$

That is, in a preflow, instead of flow in equalling flow out for every vertex other than the source and the sink, we have that total flow in is at least total flow out. We define the *excess* to be the difference between the flow in and flow out.

Definition 2 *We define the excess at node i to be $e_i \equiv f^{in}(i) - f^{out}(i) \geqslant 0$.*

If $e_i = 0$, for $\forall i \neq s, t$, then the preflow is a flow. Given a preflow, we try to reach a feasible flow by pushing excess e_i to sink t and the remainder to source s along shortest paths. Maintaining shortest path lengths is expensive, so instead we maintain a "gradient": a distance labeling d which gives us estimates on the shortest path to the sink.

Definition 3 *A distance labeling d is a set of d_i for $\forall i \in V$. d is valid with respect to f if*

- *d_i is a non-negative integer associated with each node i*
- *$d_t = 0$, $d_s = n$*
- *$d_i \leqslant d_j + 1 \quad \forall\, (i,j) \in A_f$ (residual edge)*

The intuition is that $d_i < n$ gives a lower bound on distance to t, and $d_i \geqslant n$ gives a lower bound on distance to s.

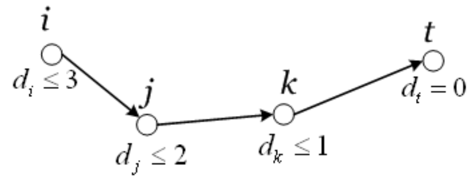

Figure 1: $d_i \leqslant d(i,t)$

Claim 1 *If there is an i-t path in G_f, then d_i is a lower bound on the distance from i to t: $d_i \leqslant d(i,t)$.*

To see this, consider the shortest path P from i to t. Any arc (i,j) on this path has the relation $d_i \leqslant d_j + 1$, as shown in Figure 1. Thus, $d_i \leqslant |P|$, and is the lower bound on distance of i to t.

Claim 2 *An s-t path in G_f never exists as long as d is valid. $\Rightarrow$ If f ever becomes a true flow, it must be optimum.*

Definition 4 *If $e_i > 0$ for $\forall i \in V$, $i \neq s, t$, call node i active.*

Push-Relabel Algorithm

Initialize $f \leftarrow 0$, $e \leftarrow 0$

$f_{sj} \leftarrow u_{sj}$; $f_{js} \leftarrow -f_{sj}$; $e_j = u_{sj}$; $f_{ij} \leftarrow 0$ for all other edges

$d_s \leftarrow n$; $d_t \leftarrow 0$; $d_i \leftarrow 0, \forall i \in V, i \neq s, t$

While $\exists$ an active node i

-If $\exists j$, s.t. $u^f_{ij} > 0$ and $d_i = d_j + 1$, then

Push $\delta \leftarrow \min(e_i, u^f_{ij})$

$f_{ij} \leftarrow f_{ij} + \delta$; $f_{ji} \leftarrow f_{ji} - \delta$

$e_i \leftarrow e_i - \delta$; $e_j \leftarrow e_j + \delta$

-Else

Relabel $d_i \leftarrow \min_{(i,j)\in A_f} (d_j + 1)$.

Question: If this algorithm terminates, will we have a max-flow f?

- Only terminates when f is a flow
- Already argued that if d remains valid, f will be optimal
- By induction over steps of the algorithm, all our operations preserve validity of d

Claim 3 *If i is active ($e_i > 0$) at some point, then there is an i-s path in G_f.*

Proof: Let $S = \{j : \text{there is an } i\text{-}j \text{ path in } G_f\}$ for i active. Note that for all $k \in S$, $j \notin S$, $f_{kj} \leqslant 0$ (otherwise i could reach k). Suppose that $s \notin S$,

$$\begin{aligned}
\sum_{j \in S} e_j &= \sum_{j \in S} \sum_{(k,j) \in A} f_{kj} \quad \text{(using } f_{ij} = -f_{ji}\text{, this is flow in minus flow out)} \\
&= \sum_{k \in S, j \in S} f_{kj} + \sum_{k \in S, j \notin S} f_{kj} \\
&= 0 + \sum_{k \in S, j \notin S} f_{kj} \quad \text{(Each } f_{kj} \text{ cancels each } f_{jk}\text{)} \\
&\leqslant 0
\end{aligned}$$

This is a sum of terms that is all "$\geqslant 0$"$\Longrightarrow e_j = 0$ for all $j \in S \Longrightarrow$ This contradicts the fact that $i \in S$ and i is active. □

Claim 4 *For all i at all points in the algorithm, $d_i \leqslant 2n - 1$.*

Proof: $d_s = n$, $d_t = 0$ never change. d_i increases only when i is active. i is active implies there exists a path P in G_f from i to s. The path in G_f has the length of at most $n - 1$. So $d_i \leqslant d_s + n - 1 = 2n - 1$. □

In the algorithm, there are two types of pushes:

- push is saturating if $\delta = u_{ij}^f$ (reach residual capacity and stop)
- push is nonsaturating if $\delta < u_{ij}^f$, i.e $\delta = e_i$ (reach excess and stop)

Claim 5 *There are at most $2mn$ saturating pushes.*

Proof: In fact, there will only ever be $\leqslant n$ saturating pushes on each $(i, j) \in A_f$. At a saturating push from i to j, $d_i = d_j + 1$. After this, (i, j) leaves A_f, a saturating push cannot be done until it returns. What must happen before it returns? A push from j to i, and $d_j = d_i + 1$. So the distance label of j must increase by at least 2. But $d_j \leq 2n - 1$ and thus can be increased by 2 at most n times. Thus there are at most n saturating pushes on (i, j). □

Claim 6 *There are at most $4mn^2$ nonsaturating pushes.*

Proof: Let $\Phi \equiv \sum_{\text{active } i} d_i$ be the "potential function" (Global Progress Measure). At the start of algorithm $\Phi = 0$, and $\Phi \geqslant 0$ throughout algorithm. $\Longrightarrow$ total of all decreases to Φ during algorithm $\leqslant$ total of all increases to Φ.

Which push/relabel operations make Φ increase? Relabel will increase it by at most $2n^2$. One saturating push may create a new active vertex and increase it by at most $2n - 1$. So Φ can increase by at most $2n^2 + 4mn^2 - 2mn$, as shown in Figure 2.

Therefore, total increases of Φ

- due to relabels $\leqslant$ (($2n$-1) per node)(n nodes) $\leqslant 2n^2$

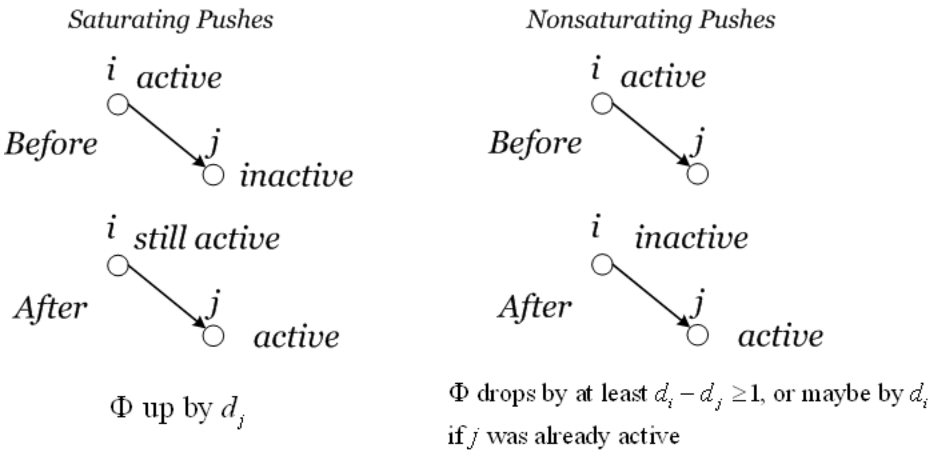

Figure 2: Saturating pushes and nonsaturating pushes

- due to saturating pushes $\leqslant$ ($2mn$ saturating pushes)(($2n-1$) changes per push) $\leqslant 4mn^2 - 2mn$

So Φ can increase by at most $4mn^2 - 2mn + 2n^2$.

Which push/relabel operations make Φ decrease? Only nonsaturating pushes make Φ decrease, because it make i inactive $\Longrightarrow$ no more than $4mn^2 - 2mn + 2n^2$ nonsaturating pushes. $\square$