Optimization II **Winter 2009/10**

Lecture 3 — October 26

Lecturer: Julián Mestre

3.1 The push-relabel algorithm

Definition 3.1. *Given a directed graph (V, E) and edge capacities $c : E \rightarrow \mathcal{Z}^+$, and two distinguished vertices s (source) and t (sink), we say $f : E \rightarrow \mathcal{Z}^+$ is an s-t flow if*

i) for every edge $e \in E$ the flow does not violate the edge capacity,

$$0 \leq f(e) \leq c(e), \text{ and}$$

ii) for every vertex $u \in V - s - t$, we have flow conservation at u,

$$f^{in}(u) \equiv \sum_{(v,u)\in E} f(v,u) = \sum_{(u,v)\in E} f(u,v) \equiv f^{out}(u).$$

The value of the flow is $value(f) \equiv f^{out}(s) = f^{in}(t)$. (For simplicity, we assume that there are no edges into s or out of t.)

MAXIMUM FLOW

Input: graph (V, E), capacities $c : E \rightarrow \mathcal{Z}^+$, source s, and sink t

Output: flow $f : E \rightarrow \mathcal{Z}^+$

Objective: maximize $value(f)$

A concept central to most maximum flow algorithms is that of the residual graph G_f of a graph G with respect to a flow f. The vertex set of G_f is the same as G. There are two kinds of residual edges in G_f: Forward and backward edges. A *forward edge* $(u, v) \in G_f$ signals the possibility of increasing the flow along the edge $(u, v) \in G$, while a *backward edge* $(u, v) \in G_f$ signals the possibility of decreasing the flow along the edge $(v, u) \in G$. Below is a more formal definition.

Definition 3.2. *Given a flow f on G, we define a residual graph G_f as follows:*

i) The node set of G_f is the same as G.

ii) For each edge $(u, v) \in E$ such that $f(u, v) < c(u, v)$, we add in G_f the forward edge *(u, v) with residual capacity $c(u, v) - f(u, v)$.*

iii) For each edge $(u, v) \in E$ such that $f(u, v) > 0$, we add in G_f the backward edge *(v, u) with residual capacity $f(u, v)$.*

Lemma 3.3. *A flow f has maximum value if and only if there is no s-t path in G_f.*

Although our ultimate goal is to compute a flow, the basic object in our algorithm will be an almost-feasible-flow, which we call preflow.

Definition 3.4. *Given (V, E), c, s, and t, we say $f : E \to \mathcal{Z}^+$ is a* preflow *if*

i) for all $e \in E$ the flow does not violate the capacities,

$$f(e) \leq c(e), \text{and}$$

ii) for all $u \in V - s$ the excess at u is non-negative,

$$e_f(u) \equiv f^{in}(u) - f^{out}(u) \geq \ 0.$$

Observation 3.5. *A preflow f is a flow if $e_f(u) = 0$ for all $u \in V - s - t$.*

Lemma 3.6. *Let f be a preflow in G. If a node u has positive excess then there is a path in G_f from u to s.*

The last ingredient of the algorithm is labeling function on the vertices.

Definition 3.7. *A labeling $h : V \to \mathcal{Z}^+$ is* compatible *with a preflow f if*

i) (boundary conditions) $h(t) = 0$ and $h(s) = n$, and

ii) (steepness conditions) For every residual edge (u, v) in G_f, we have $h(u) \leq h(v) + 1$.

Lemma 3.8. *Let f be preflow compatible with a labeling h. Let p be a path in G_f from some vertex u to some vertex v then $h(u) \leq h(v) + |p|$, where $|p| = \#$ of edges in p.*

Algorithm 1 PUSH-RELABEL(V, E, c, s, t)

$h(s) \leftarrow n$ and $h(u) \leftarrow 0$ for all $u \neq s$. // initial labeling
$f(e) \leftarrow c(e)$ for $e = (s, v) \in E$ and $f(e) \leftarrow 0$. // initial preflow
while $\exists\, u \neq t$ with $e_f(u) > 0$ **do**
  if $\exists\, (u, v)$ residual edge in G_f and $h(v) < h(u)$ **then**
    if (u, v) is forward **then**
      increase $f(u, v)$ by min $\{e_f(u),\ c(u, v) - f(u, v)\}$ // push forward
    if (u, v) is backward **then**
      decrease $f(v, u)$ by min $\{e_f(u),\ f(v, u)\}$ // push backward
  else
    $h(u) \leftarrow h(u) + 1$ // relabel node
return f

3.1.1 Analysis of the generic algorithm

We need to derive some properties about the preflow f and the labeling h the algorithm maintains throughout the execution of PUSH-RELABEL.

Lemma 3.9. *Throughout the execution, f is a preflow and h is compatible with f.*

Lemma 3.10. *Throughout the execution, all nodes have $h(u) < 2n$.*

Everything is in place to bound the number of relabel and push operations.

Lemma 3.11. *The total number of relabeling operations is less than $2n^2$.*

To bound the number of push operations it is convenient consider separately those that are *saturating* and those are not. We say a forward push saturates $(u, v) \in G_f$ if we increase the flow along $(u, v) \in G$ by $c(u, v) - f(u, v)$. Likewise, we say a backward push saturates $(u, v) \in G_f$ if we decrease the flow along $(v, u) \in G$ by $f(u, v)$. The important thing to notice is that after a saturating push along an edge, the edge disappears from the residual graph.

Lemma 3.12. *The total number of nonsaturating push operations is at most $2nm$.*

Lemma 3.13. *The total number of saturating push operations is at most $2n^2m$.*

Theorem 3.14. *The generic* PUSH-RELABEL *algorithm terminates after $O(n^2m)$ iterations and returns a maximum flow.*

Proof: In each iteration we do a push or a relabel operation. It follows from Lemmas 2.11, 2.12, and 2.13 that there are at most $O(n^2m)$ many iterations.

The preflow f the algorithm returns is in fact a flow, since $e_f(u) = 0$ for all $u \in V - s - t$. To see that is in fact maximum, suppose that there is a s-t simple path p in G_f. Since the labeling h is compatible with f, it follows from Lemma 2.8 that $h(s) \leq h(t) + |p|$. However, since the path is simple $|p| < n$, and since the labeling is compatible $h(s) = n$ and $h(t) = 0$. A contradiction and thus f is maximum. □

In the next lecture we will see how to implement PUSH-RELABEL efficiently and how to improve the running time by being more careful when choosing which edge to push flow along.