Excess Balancing Algorithms for the Maximum Flow Problem

COS 528 class notes – October 16, 2006

Scribe: Dávid Papp

1 Basics

Main idea: maintain a **pseudoflow** (an $f : E \to \mathbb{R}_0^+$ function satisfying the capacity constraints) and try to balance out the excesses/deficits of the nodes. Directionality is implied, flow is moved from the nodes with (positive) excesses to nodes with deficits (negative excesses).

Compare to preflow-push (PP) algorithms: they maintain a **preflow** (pseudoflow with nonnegative excesses except for the source); excesses are moved forward (towards the sink). For the PP algorithms we need to maintain a distance function as well, to know which direction "forward" is.

Notation: V is the set of nodes $e(v), v \in V$ is the excess of node v, $c(u, v)$ and $r(u, v)$ are the capacity and residual capacity, resp., of arc (u, v).

Generic step: find an arc (v, w) with $e(v) > e(w)$ and $r(v, w) > 0$; move

$$\min\left\{r(v, w), \frac{e(v) - e(w)}{2}\right\}$$

units of flow from v to w. This either saturates the arc (v, w), or equalizes the excesses of nodes v and w.

Generic algorithm: start with dummy excesses $e(s) = \infty, e(t) = -\infty$, or equivalently, with initial pseudoflow

$$f(v, w) = \begin{cases} c(v, w) & v = s \text{ or } w = t \\ 0 & \text{otherwise} \end{cases};$$

repeat *generic step* until *stopping condition.*

Issues with the generic algorithm:

- stopping condition (the "no more moves available" may never hold);
- precision (integrality of pseudoflow and residual capacities is not maintained, even though we assume integral capacities);
- what order the arcs should be processed?

2 A polynomial time algorithm

A **canonical cut** with boundary a is defined as the the cut (S, T) with $S = \{v \in V \mid e(v) \geq a\}$, $T = \{v \in V \mid e(v) < a\}$. Let us use the notation

$$S(a) = \{v \in V \mid e(v) \geq a\}.$$

Note that the generic algorithm never moves flow through a canonical cut if it is **saturated**, that is, if every forward arc across it is saturated. Hence V can be partitioned into **sections** of the form

$$S(a, b) := \{v \in V \mid a \leq e(v) < b\} = S(a) \setminus S(b),$$

where $a < b$, $S(a)$ and $S(b)$ are saturated canonical cuts, and $S(a, b)$ is minimal with respect to inclusion. The algorithm never moves flow across sections, but may split sections into two by saturating a canonical cut which falls into the section.

The unique section $S(a, b)$ with $a \leq 0 < b$ is called the **active section**.

Let us address now the question about the stopping condition.

Claim. *Suppose that in the active section $S(a, b)$ every node v satisfies $|e(v)| < 1/n$. Then the canonical cuts bordering the active section are minimal. The flow through these cuts have the same value as the maximum flow in the network.*

Remark. Claim proves that our stopping condition in the generic algorithm could be "$|e(v)| < 1/n$ for every node v in the active section". Alternatively, we can multiply initially every capacity by n, and then we only have to deal with excesses greater or equal to one.

Proof of claim. We are going to correct the pseudoflow to a flow without changing the flow through one of the bordering cuts, say, $(S(a), V \setminus S(a))$. Hence we will obtain a flow which saturates a cut, consequently we will have

a maximum flow and a minimum cut. We assume that we have multiplied every capacity by n, and hence in the active section every node v satisfies $|e(v)| < 1$.

First, eliminate all cycles of flow. This does not affect any of the saturated cuts, because they don't have forward residual edges. Then the arcs with positive pseudoflow value form an acyclic subnetwork. In this network consider the nodes in $V \setminus S(a)$, where clearly every node has deficit (negative excess). Process them in forward topological order, and increase the flow in their outgoing forward arcs so that their deficit becomes zero. (Verify that this is possible without violating capacity constraints!) Now consider the nodes in $S(a)$, where excesses of any sign are possible. Process the nodes with positive excess in backward topological order, and decrease the flow in their incoming backward arcs so that their excess becomes zero.

At this point every node has zero excess, except for some nodes in the active section with small, fractional negative excesses, s, and t. Consider now the residual arcs with fractional residual capacity. Since the capacities are assumed to be integral, these arcs come in backward/forward pairs, so we can consider this subnetwork as an undirected graph. Eliminate all cycles of flow again, then the graph being considered becomes a forest. Clearly, every leaf (node with degree one) has to be either the source or a node with negative excess. By increasing the flow along paths of the form $s \to v$ and $v \to w$ satisfying $e(v) \geq e(w)$ with every such path we decrease the number of fractional residual arcs by at least one, and such paths always exist as long as there is at least one node v with negative excess. In at most n iterations we obtain a flow. □

Remark. The proof shows that this method is apparently better suited for minimum cut problems than for maximum flow problems. To extract a minimum cut from the pseudoflow virtually no post-processing is necessary. Obtaining a maximum flow needs extra work.

We have yet to bound the number of steps necessary to satisfy the stopping condition.

Claim. *Suppose that we process the arcs in a round-robin fashion. Then $O(n^2)$ passes of round-robin decrease $\max_{v \in S(a,b)} e(v) - \min_{v \in S(a,b)} e(v)$, that is the maximum difference between excesses within the active section, by a constant factor.*

Proof. We prove that after $O(n^2)$ passes any section is dissected to "roughly equal" parts, which guarantees that the maximum excess difference in the new parts are smaller by a constant factor than the maximum excess differ-

ence in the original section. Let the section be $S = S(c, c+d)$, let the number of nodes in the section be $s \leq n-2$, and let $b_i := \frac{d/2+di}{3s+3}$ for $i = 0, \ldots, 3s+2$. That is, divide the interval $[c, c+d]$ into $3s+3$ equal size parts ("buckets"), and let b_i mark the middle of these buckets. Define the potential function of the section as

$$\Phi = \sum_{i=0}^{3s+2} \Phi(b_i), \quad \text{where } \Phi(b) = \frac{n}{d} \sum_{v \in S} |e(v) - b|.$$

There are at least $2s + 3$ empty buckets, consider now one of them, with midpoint b_i. There must be nonsaturated arcs crossing this bucket, otherwise b_i would be a saturated canonical cut dissecting the section S, a contradiction. So the next pass of the round robin pushes flow along this arc, decreasing $\Phi(b_i)$ (and hence Φ) by at least $\frac{n}{d} \cdot \frac{d}{3s+3} > \frac{1}{3}$.

The potential function is in the range $[0, (3s + 3)ns]$, so at most $O(n^3)$ such potential increase is possible before all the canonical cuts corresponding to the empty buckets get saturated. Moreover, unless at least $s + 2$ of these cuts are already saturated, the increase in the potential is at least $\frac{n}{d} \cdot \frac{d}{3s+3} \cdot (s+2) > \frac{1}{3}n$ per pass, because we can repeat the previous argument for every nonsaturated empty bucket. Consequently in $O(n^2)$ rounds we saturate at least $s+2$ of the cuts corresponding to the empty buckets. Since there are only $3s+3$ buckets in total, at this point there is a saturated empty bucket among the $2s + 2$ middle buckets, that is, in the middle $2/3$ of the section. At this point we can dissect the section into two large enough parts – in both the maximum excess difference is at most $2/3d$. □

Corollary. *The generic algorithm with round-robin arc processing and with the above stopping condition terminates in $O(n^2 m \log(nU))$ time.*

Proof. From the claims, $O(n^2)$ rounds of round-robin decrease the maximum difference between excesses within the active section by a constant factor, and $O(\log(nU))$ such iterations are needed to reduce it from $2U$ to $1/n$. Clearly, each iteration runs in $O(m)$ time. Multiplying these terms yield the statement. □

The above theorem shows a possible alternative implementation. Instead of going through all the arcs in every pass it suffices to consider only the arcs which cross the active section. The price is that we need to keep track of the active section. The overall running time does not improve.

The above running time bound is tight. Example: let the network be simply an s–t path with unit capacities.