Preflow-Push Method for Maximum Flows

Jonathan Turner

March 14, 2013

The algorithms we have studied for the max flow problem all work by adding flow along augmenting paths. At every step, we take a valid flow function and increase the flow, while respecting the capacity constraints and flow conservation conditions. At all times, the flow function we're working with is a valid, if sub-optimal solution to the problem. As we add flow along new augmenting paths, we approach the max flow value "from below."

In this section we're going to study a class of algorithms that uses a very different approach. Instead of maintaining a valid flow function at all times, it uses a more general kind of flow function, called a *preflow* that is allowed to violate the flow conservation constraints; specifically, a preflow may specify more flow entering a vertex than there is leaving the vertex. We'll start with a preflow that has a value that is at least as large as the maximum flow value. As the algorithm runs, it seeks to convert the preflow into an ordinary flow by reducing the number of vertices at which there is excess incoming flow. In the process, it will reduce the value of the preflow, approaching the valid max flow amount from above rather than from below. For a preflow f and a vertex u, we define $\Delta f(u)$ to be the *excess flow* at u; that is, it is the difference between the total incoming flow at u and the total outgoing flow. We say that a vertex u is *balanced* if $\Delta f(u) = 0$. When all vertices are balanced, then f is a valid flow function.

Like Dinic's algorithm, the preflow-push method constrains the search to a subset of edges. To do this, it defines a set of distance labels that play a similar role as the *level* values played in Dinic's algorithm. We say that a set of distance labels $d(u)$ is *valid* if $d(t) = 0$ and for all edges (u, v) with $res(u, v) > 0$ and $d(u) \leq d(v) + 1$. Note that this implies that, $d(u)$ can be no longer than the shortest path distance from u to t in the residual graph. We say that a set of distance labels are *exact* if $d(u)$ is the shortest path distance to t for all u. We say that an edge (u, v) is *admissible* if $res(u, v) > 0$

and $d(u) = d(v) + 1$ and we say that a path is admissible if all its edges are. Note that a minimum length augmenting path is admissible with respect to the exact distance labels. While the preflow-push method does not add flow along augmenting paths, it does constrain flow changes to the edges that are admissible at any given point in time.

We can now describe the general preflow-push method. First, we create a preflow f by letting $f(s, u) = cap(s, u)$ for all edges leaving the source. Note that $|f|$ is at least as large as the value of a maximum flow. Next, we compute a set of valid vertex labels by letting $d(s) = n$, and for all other vertices u, letting $d(u)$ be the length of the shortest path from u to the sink. We then repeat the following step so long as there are unbalanced vertices.

> Select an unbalanced vertex u. If there are admissible edges leaving u, select one such edge (u, v) and add flow to it until either the edge becomes saturated or u becomes balanced. If there are no admissible edges at u, increase $d(u)$ until at least one edges becomes admissible.

The step that increases the distance labels makes $d(u) = d(v) + 1$ where v has the smallest distance label among those neighbors of u with $res(u, v) > 0$. Note that the initial distance labels are valid (since all edges leaving s are saturated), and that each step maintains the validity of the labels. Also, note that the method never violates any edge capacity constraints, so when it terminates, f is a valid flow function. Moreover, at all times there is a saturated cut separating the source from the sink, so that when the algorithm terminates, f is not just any valid flow, it is a max flow.

Observe that the general method does not specify how to select unbalanced vertices or admissible edges. Different selection methods produce different algorithms with different performance characteristics.

Let's look at example of the preflow-push method in action. Figure 1 shows the initial state, where the sources have been saturated. Note the

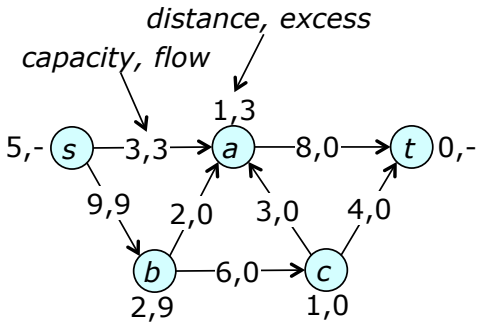

Figure 1: Initial state of general preflow-push method

excess flow at vertices a and b. We can select either vertex a or b for the first step. Let's assume that vertex b is selected twice in a row. This leads to the state shown in Figure 2. Notice that c is now unbalanced also.

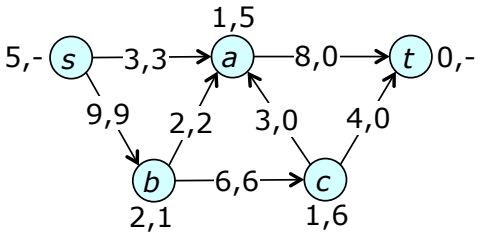

Figure 2: After selecting b twice

If we select vertex c at this point, we can add flow on the admissible edge (c, t), saturating it and reducing the excess at c to 2. If we select c a second time, none of the incident edges is admissible, leading us to relabel c, making the edge (c, a) admissible. Selecting c for a third time leads to the state shown in Figure 3.

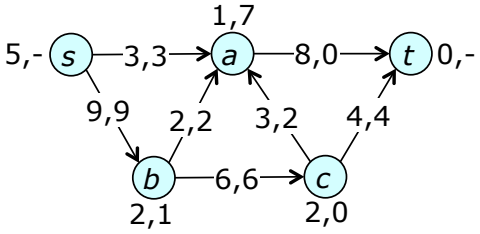

Figure 3: After selecting c, relabeling and selecting c again

At this point, if we select a, we get the state shown in Figure 4. Notice

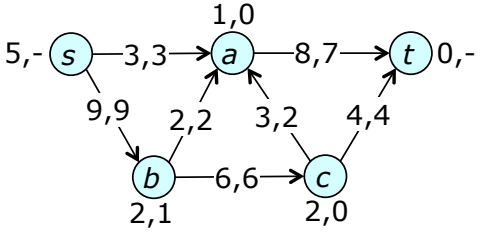

Figure 4: After selecting a

that now the only unbalanced vertex is b, but b had no admissible edges leaving it. This means we must relabel b. Since the only edge at b with positive residual capacity leaving b is (b, s), the new label at b is 6. This makes the edge (b, s) admissible, allowing us to push one unit of flow back to the source. This makes b balanced, yielding a valid maximum flow.