Choosing augmenting paths in the Ford-Fulkerson algorithm

Vassos Hadzilacos

In what follows we assume that all edge capacities are non-negative integers. Let n be the number of nodes and m be the number of edges in the flow network. It is reasonable to assume that every node is reachable from s (unreachable nodes are not used in any $s \to t$ flow and can be deleted from the graph, along with their incoming and outgoing edges). Note that under this assumption $m \geq n - 1$ and so $n = O(m)$. Let C be the sum of the capacities of the edges out of the source s.

We have seen that in the Ford-Fulkerson algorithm it is possible to choose augmenting paths so that C augmenting steps are performed before the algorithm terminates. Since each iteration takes $O(m)$ time, the overall running time of this basic version of the Ford-Fulkerson algorithm (where augmenting paths are chosen arbitrarily) is $O(mC)$. This is a so-called ***pseudopolynomial*** algorithm: its running time is polynomial in the ***numeric value*** of the input, rather than the ***size*** of the input's binary representation.

We will now see that by choosing augmenting paths more wisely, we can improve the performance of the algorithm dramatically.

Choosing a "widest" path

Let (G, s, t, c) be a flow network with integer capacities, and f be a flow in this flow network. Consider the residual graph G_f. We define the ***width*** of an $s \to t$ path p in G_f to be the ***minimum*** residual capacity of any edge on p. Note that this is the amount by which the flow f is improved if we augment it along p. Instead of picking an arbitrary augmenting path, we will choose an augmenting path of ***maximum width*** — i.e., a path that improves the value of the flow as much as is possible in a single augmenting step.

Exercise. Describe an algorithm that, given the residual graph in adjacency list form, finds a widest $s \to t$ path in $O(m \log n)$ time. (Hint: Modify an algorithm we discussed earlier in the course.)

It turns out that if we always choose the augmenting path in this way, the Ford-Fulkerson algorithm will finish in at most $O(m \log C)$ iterations. Each iteration now takes $O(m \log n)$ time, rather than $O(m)$ time as in the basic Ford-Fulkerson algorithm; this is because finding a widest $s \to t$ path takes $O(m \log n)$ time, while finding an arbitrary $s \to t$ path can be done in just $O(m)$ time using depth- or breadth-first search. Thus, the overall running time of the algorithm with this way of choosing augmenting paths is $O(m^2 \log n \log C)$. This is ***polynomial in the input size***.

Choosing a path with fewest edges

Another way to choose augmenting paths to ensure good performance of the Ford-Fulkerson algorithm is to always use as augmenting path one with the ***fewest*** edges. We can find such a path in $O(m)$ time using breadth-first search (starting at s).

It turns out that with this choice of augmenting paths, the Ford-Fulkerson algorithm is guaranteed to terminate after at most mn augmentations, independent of the capacities, assuming that each arithmetic operation can be performed in $O(1)$ time. (The algorithm only adds or subtracts capacities.) Thus, the overall running time of the algorithm with this way of choosing augmenting paths is $O(m^2 n)$. This is polynomial in the input size, and independent of the capacities. Algorithms whose performance does not depend at all on the numeric value of the input (assuming $O(1)$-time arithmetic operations on numbers

regardless of their size) are called ***strongly polynomial***. Thus, the Ford-Fulkerson algorithm with this method of choosing augmenting paths is strongly polynomial.

In the remainder of this document we prove that each of these methods of choosing augmenting paths results in the stated bounds for the number of iterations. These proofs are optional material for the course: You should know the two ways of choosing augmenting paths and their effect on the running time of the Ford-Fulkerson algorithm, but you are not required to read the proofs. I hope that some of you will find them interesting. The "widest path" and "fewest-edges path" methods of choosing augmenting paths and the resulting running time of the Ford-Fulkerson algorithm were described by Jack Edmonds and Richard Karp in their paper "Theoretical improvements in algorithmic efficiency for network flow problems", *Journal of the ACM*, 12:2(248-264), April 1972.

Number of iterations using widest augmenting paths

Step 1. Let f be any flow and f^* be a maximum flow of (G, s, t, c). Let δ be an upper bound on the widths of all augmenting paths of G_f; that is, the width of ***every*** augmenting path of G_f is at at most δ. We will show that

$$\mathcal{V}(f^*) - \mathcal{V}(f) \leq m \cdot \delta. \tag{1}$$

The proof of this is similar to the proof that the value of the maximum flow is equal to the value of a minimum (s, t)-cut.

We define the following (s, t)-cut: Let S be the set of nodes reachable from s in G_f via a path of width ***greater than*** δ; and let T be the remaining nodes of G_f. By the definition of δ, $t \in T$ (every $s \to t$ path has width at most δ, so there is no $s \to t$ path of width greater than δ). So (S, T) is an (s, t)-cut.

Claim 1 *For every $e \in out(S) \cap in(T)$, $f(e) \geq c(e) - \delta$; and for every $e \in out(T) \cap in(S)$, $f(e) \leq \delta$.*

Proof. If $(u, v) \in out(S) \cap in(T)$ and $f(u, v) < c(e) - \delta$, then G_f has a forward edge (u, v) with residual capacity more than δ. Since $u \in S$, v is also reachable in G_f from s using edges with residual capacity more than δ, and therefore $v \in S$ — contradicting that $(u, v) \in out(S) \cap in(T)$. Similarly, if $(u, v) \in out(T) \cap in(S)$ and $f(u, v) > \delta$, then G_f has a backward edge (v, u) with residual capacity more than δ. Since $v \in S$, u is also reachable in G_f from s using edges with residual capacity more than δ, and therefore $u \in S$ — contradicting that $(u, v) \in out(T) \cap in(S)$. □

We now have:

$$\begin{aligned}
\mathcal{V}(f) &= \sum_{e \in out(S) \cap in(T)} f(e) - \sum_{e \in out(T) \cap in(S)} f(e) && \text{[Flow value lemma]} \\
&\geq \sum_{e \in out(S) \cap in(T)} \big(c(e) - \delta\big) - \sum_{e \in out(T) \cap in(S)} \delta && \text{[Claim 1]} \\
&= \sum_{e \in out(S) \cap in(T)} c(e) - \sum_{e \in \big(out(S) \cap in(T)\big) \cup \big(out(T) \cap in(S)\big)} \delta \\
&\geq \sum_{e \in out(S) \cap in(T)} c(e) - m \cdot \delta && \text{[}\leq m \text{ edges cross cut } (S, T) \text{ in either direction]} \\
&\geq \mathcal{V}(f^*) - m \cdot \delta
\end{aligned}$$

where, in the last inequality, we use the fact that the value of any flow (in particular f^*) is at most the capacity of any cut (in particular (S, T)). Rearranging, we get (1).

Step 2. Let f be any flow and f^* be a maximum flow of (G, s, t, c). Let P be a maximum-width path in the residual graph G_f, and f' be the result of augmenting f on path P; that is, $f' = \textsc{augment}(f, P)$. We will show that

$$\mathcal{V}(f^*) - \mathcal{V}(f') \leq \Big(\mathcal{V}(f^*) - \mathcal{V}(f)\Big) \cdot (1 - \frac{1}{m}). \tag{2}$$

To prove this, we use the result of Step 1.

Let b be the width of P, (the maximum-width augmenting path in G_f). Therefore, every augmenting path in G_f has width at most b. By (1),

$$\mathcal{V}(f^*) - \mathcal{V}(f) \leq m \cdot b \tag{3}$$

Let $f' = \textsc{Augment}(f, P)$. Thus,

$$\mathcal{V}(f') = \mathcal{V}(f) + b \tag{4}$$

Therefore,

$$\begin{aligned}
\frac{\mathcal{V}(f^*) - \mathcal{V}(f')}{\mathcal{V}(f^*) - \mathcal{V}(f)} &= \frac{\mathcal{V}(f^*) - \big(\mathcal{V}(f) + b\big)}{\mathcal{V}(f^*) - \mathcal{V}(f)} && \text{[by (4)]} \\
&= 1 - \frac{b}{\mathcal{V}(f^*) - \mathcal{V}(f)} \\
&\leq 1 - \frac{b}{mb} && \text{[by (3)]} \\
&= 1 - \frac{1}{m}
\end{aligned}$$

Rearranging, we get (2).

Step 3. Now we will use the result of Step 2 to prove that, if we use the maximum-width augmentation path in each iteration, the Ford-Fulkerson algorithm will finish after $O(m \log C)$ iterations.

Let f_0 be the trivial flow (that carries no traffic in any edge), and f_i be the flow after i augmentation steps, where in each step we choose the maximum width augmenting path. By (2) and straightforward induction, $\mathcal{V}(f^*) - \mathcal{V}(f_i) \leq \big(\mathcal{V}(f^*) - \mathcal{V}(f_0)\big) \cdot (1 - 1/m)^i$. Note that $\mathcal{V}(f_0) = 0$ and $(1 - 1/m)^i < e^{-i/m}$ (since $1/m \neq 0$, and $1 - x < e^{-x}$ for all $x \neq 0$). Thus, $\mathcal{V}(f^*) - \mathcal{V}(f_i) < \mathcal{V}(f^*) \cdot e^{-i/m}$. Letting $i = m \ln C$, we get $\mathcal{V}(f^*) - \mathcal{V}(f_i) < \mathcal{V}(f^*)/C \leq 1$ (for the last inequality we use the fact that, by definition of C, $\mathcal{V}(f^*) \leq C$). Since each f_i is an integral flow and f^* is a maximum flow, it follows that $\mathcal{V}(f^*) - \mathcal{V}(f_i) = 0$, i.e., $\mathcal{V}(f_i) = \mathcal{V}(f^*)$. So, after at most $i = m \ln C = O(m \log C)$ augmentations, the algorithm computes a maximum flow.

Number of iterations using shortest augmenting paths

In what follows, when we speak of "shortest path" we mean a path with the minimum number of edges (as opposed to the minimum sum of capacities, as in the weighted shortest path problem).

Our aim is to show that by always choosing a shortest augmenting path, the Ford-Fulkerson algorithm will terminate after at most mn iterations. The basic intuition behind this fact is as follows. Each augmenting step causes at least one edge to be deleted from the residual graph. It may also cause some edges to be inserted. The deleted edges make shortest paths longer. And, it turns out, that re-inserted edges do not create paths that are shorter than those that already exist. Thus, shortest paths to nodes get longer and longer. Since a shortest $s \to t$ path cannot have more than $n - 1$ edges, after some number of augmenting steps s and t will become disconnected, meaning that the FF algorithm terminates. A careful counting argument shows that this must happen after no more than mn augmenting steps. We now give the detailed proof that makes this intuition precise.

Let G_k be the residual graph after k augmenting steps, using the shortest path method. Define the ***depth*** of a node u in G_k to be the minimum number of edges on any path from s to u in G_k (i.e., the depth of u on a BFS tree of G_k rooted at s); we denote this as $\mathbf{depth}_k(u)$. Consider how augmenting step $k+1$ changes the residual graph G_k to G_{k+1}. Let $p = u_0, u_1, \ldots, u_m$ be the augmenting path of G_k used in this step, where $u_0 = s$ and $u_k = t$, and each node u_i has depth i in G_k; this is because, by the choice of augmenting paths, p is a shortest path from s to t. For each edge (u_i, u_{i+1}) on p:

- If (u_i, u_{i+1}) is a forward edge, this edge is deleted from the residual graph if and only if the augmenting step saturated (u_i, u_{i+1}); and the reverse edge (u_{i+1}, u_i) is added to the residual graph as a backward edge if and only if the edge (u_i, u_{i+1}) had zero flow before the augmenting step.
- If (u_i, u_{i+1}) is a backward edge, this edge is deleted from the residual graph if and only if the augmenting step removed all flow from (u_{i+1}, u_i); and the reverse edge (u_{i+1}, u_i) is added to the residual graph as a forward edge if and only if the edge (u_{i+1}, u_i) was saturated before the augmenting step.

No other changes to the set of edges of the residual graph are made by the augmenting step. So, every edge added to the residual graph during augmenting step $k+1$ goes from a node with depth $i+1$ to a node with depth i in G_k. Thus, we have:

Claim 2 *If an edge (u, v) is added to the residual graph G_{k+1} during augmenting step $k+1$ (i.e., (u, v) is not in G_k but is in G_{k+1}), then $\boldsymbol{depth}_k(u) = \boldsymbol{depth}_k(v)+1$.*

Next we will show that the depth of a node does not increase as a result of an augmenting step.

Claim 3 *For every node u, $\boldsymbol{depth}_{k+1}(u) \geq \boldsymbol{depth}_k(u)$.*

Proof. Let u be any node, and let $u_1, u_2, \ldots, u_m$, where $u_1 = s$ and $u_m = u$ be a shortest (fewest edges) path from s to u in G_{k+1}. By definition of depth, $\mathbf{depth}_{k+1}(u_i) = i$ for each i, $1 \leq i \leq m$.

By induction we show that, for each i, $1 \leq i \leq m$, $\mathbf{depth}_k(u_i) \leq \mathbf{depth}_{k+1}(u_i)$. This is clearly true for the basis $i = 1$, since $u_1 = s$ and s is has depth 0 in every residual graph. Suppose, it is true for u_i, for some i, $1 \leq i < m$; we will prove that it is true for u_{i+1}. There are two cases: either edge (u_i, u_{i+1}) is in G_k, or it was added to the residual graph in augmenting step $k+1$.

Case 1. (u_i, u_{i+1}) is in G_k. In this case, we have:

$$\begin{aligned}
\mathbf{depth}_k(u_{i+1}) &\leq \mathbf{depth}_k(u_i) + 1 && \text{[since } (u_i, u_{i+1}) \text{ is in } G_k\text{]} \\
&\leq \mathbf{depth}_{k+1}(u_i) + 1 && \text{[by the IH]} \\
&= i + 1 && \text{[since } \mathbf{depth}_{k+1}(u_i) = i\text{]} \\
&= \mathbf{depth}_{k+1}(u_{i+1}) && \text{[since } \mathbf{depth}_{k+1}(u_{i+1}) = i+1\text{]}
\end{aligned}$$

Case 2. (u_i, u_{i+1}) is not in G_k but is added to G_{k+1}. In this case, we have:

$$\begin{aligned}
\mathbf{depth}_k(u_{i+1}) &\leq \mathbf{depth}_k(u_{i+1}) + 1 \\
&= \mathbf{depth}_k(u_i) && \text{[by Claim 2]} \\
&\leq \mathbf{depth}_{k+1}(u_i) && \text{[by the IH]} \\
&\leq i + 1 && \text{[since } \mathbf{depth}_{k+1}(u_i) = i\text{]} \\
&= \mathbf{depth}_{k+1}(u_{i+1}) && \text{[since } \mathbf{depth}_{k+1}(u_{i+1}) = i+1\text{]}
\end{aligned}$$

□

In each augmenting step, at least one edge is deleted from the residual graph, namely every edge on the augmenting path that has minimum residual capacity b. Let (u, v) be an edge that is deleted in augmenting

step ℓ, and let $i = \mathbf{depth}_\ell(v)$. Since (u, v) is on an augmenting path, which is a shortest $s \to t$ path, $\mathbf{depth}_\ell(u) = i - 1$. If (u, v) is added to the residual graph as a result of a subsequent augmenting step $k + 1 > \ell$, by Claim 2, $\mathbf{depth}_k(u) = \mathbf{depth}_k(v) + 1$. And since by Claim 3 the depth of a node does not decrease in successive augmentations, $\mathbf{depth}_k(v) \geq \mathbf{depth}_\ell(v)$. So, $\mathbf{depth}_{k+1}(u) \geq \mathbf{depth}_k(u) = \mathbf{depth}_k(v) + 1 \geq \mathbf{depth}_\ell(v) + 1 = i + 1$. Thus, the depth of node u must have increased by at least 2 (from $i - 1$ to $i + 1$) between an augmenting step ℓ that deleted (u, v) from the residual graph and the subsequent augmenting step $k+1$ that reinserted it. Since the minimum depth of a node is 0 and the maximum is $n-1$, an edge can be added at most $n/2$ times to the residual graph. Since there are at most $2m$ edges in the residual graph, and each augmenting step deletes at least one edge, after $2m(n/2) = mn$ augmenting steps all edges will have been deleted and cannot be inserted (because each of them will have been re-inserted the maximum number of times that it can). Thus, the Ford-Fulkerson algorithm terminates after at most mn augmenting steps if at each step we chose to augment the flow along a shortest $s \to t$ path in the residual graph.