MATH 409 LECTURE 15
EDMONDS-KARP ALGORITHM FOR MAX FLOW

REKHA THOMAS

Last time we saw the Ford-Fulkerson algorithm for max flow in a network and saw an example in which the algorithm could take exponentially many augmentations in the size of the input. Therefore, the Ford-Fulkerson algorithm is not a polynomial time algorithm. However, it turns out that a minor modification results in a polynomial time algorithm as was pointed out by Edmonds and Karp in 1972. See [1, Section 8.3] for more details.

Definition 1. We say that an f-augmenting path in the residual graph G_f is **shortest** if it has the least number of edges among all f-augmenting paths in G_f.

In pratice, the residual graph is modeled as follows. If $e \in E(G)$ with $u_f^+(e) = u_e - f(e) > 0$ then $e \in E(G_f)$. If $e \in E(G)$ has $u_f^-(e) = f(e) > 0$ then put in an arc e' into $E(G_f)$ where e' is e with its direction reversed. Under this rule, any directed path from s to t in G_f will be an f-augmenting path. We no longer have to worry about making sure that all the forward edges in the path have $u_f^+(e) > 0$ and all backward edges have $u_f^-(e) > 0$.

A shortest path from s to any vertex v in G_f can be found by **breadth-first search** in $O(m)$ time where $m = |E(G)|$. Check that any such shortest path from s to v has at most $n-1$ edges where $n = |V(G)|$.

Edmonds and Karp algorithm for Max Flows
Input: A network (G, u, s, t).
Output: A max (s,t)-flow in the network.

(1) Set $f(e) = 0$ for all $e \in E(G)$.
(2) Find a **shortest** f-augmenting path P. If none exists then stop.
(3) Compute γ as in the Ford Fulkerson algorithm and augment f along P by γ. Go to (2).

Date: May 3, 2010.

Theorem 2. *The Edmonds-Karp algorithm requires at most $\frac{mn}{2}$ augmentations. This count is independent of the edge capacities.*

We will prove this theorem shortly. Before that we derive a corollary which proves that the Edmonds-Karp algorithm runs in polynomial time in the size of the input.

Corollary 3. *The max flow problem in a network (G, u, s, t) can be solved in $O(m^2 n)$ time.*

Proof. Each run of step (2) takes $O(m)$ time since a shortest f-augmenting path can be found in $O(m)$ time by breadth first search. Each run of step (3) also takes $O(m)$ time. By the above theorem, there are at most $\frac{mn}{2}$ augmentations and so in total, the algorithm takes $O(m^2 n)$ time. $\square$

We now prove the Edmonds-Karp theorem. The proof relies on the following lemma which is quoted below without proof. Please see Lemma 8.13 in [1] if you would like to see a proof.

Lemma 4. *Let $f_1, f_2, \ldots$ be a sequence of flows such that f_{i+1} is gotten from f_i by augmenting along path P_i where P_i is a shortest f_i-augmenting path. Then*

(1) $|E(P_k)| \leq |E(P_{k+1})|$.

(2) $|E(P_k)| + 2 \leq |E(P_l)|$ *for all $l > k$ such that there is some edge $e \in E(G)$ with the property that both e and its reverse edge are used in the union of P_k and P_l.*

Proof of theorem. Let $P_1, P_2, \ldots$ be the augmenting paths chosen during the Edmonds Karp algorithm. By the choice of γ in step (3), each P_i contains at least one bottleneck edge in the residual graph.

For a fixed e, let $P_{i_1}, P_{i_2}, \ldots$ be the sequence of augmenting paths containing e as a bottleneck edge. Note that e could be an edge in $E(G)$ or the reverse of an edge in $E(G)$. Between P_{i_j} and $P_{i_{j+1}}$ there must exist an augmenting path P_k ($i_j < k < i_{j+1}$) containing e' the edge reverse to e. Now apply part (2) of the above lemma to the pairs of paths: P_{i_j}, P_k and $P_k, P_{i_{j+1}}$ to get for all j, the inequalities:

$$|E(P_{i_j})| + 4 \leq |E(P_k)| + 2 \leq |E(P_{i_{j+1}})|.$$

But now recall that the length of any P_i is at most $n - 1$. Combining this with the above inequalities which say that the number of edges in the paths $P_{i_1}, P_{i_2}, \ldots$ increase by at least four in each step, we get that there are at most $n/4$ paths in the sequence $P_{i_1}, P_{i_2}, \ldots$ that have a fixed e as bottleneck edge.

Since each edge or its reverse edge can play the role of e, we have that there are at most $(2m)(n/4) = mn/2$ augmenting paths in the algorithm.

References

[1] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer, Berlin, 2000.