CMSC 651 Advanced Algorithms	Lecture 09
Lecturer: Samir Khuller	Feb 26, 2002

Notes by Samir Khuller.

16 The Max Flow Problem

Today we will study the Edmonds-Karp algorithm that works when the capacities are integral, and has a much better running time than the Ford-Fulkerson method. (Edmonds and Karp gave a second heurisitc that we will study later.)

Assumption: Capacities are integers.

1st Edmonds-Karp Algorithm:

while (there is an augmenting path $s-t$ in G_f) **do**
 pick up an augmenting path (in G_f) with the highest residual capacity;
 use this path to augment the flow;

Analysis: We first prove that if there is a flow in G of value $|f|$, the highest capacity of an augmenting path in G_f is $\geq \frac{|f|}{m}$. In class we covered two different proofs of this lemma. The notion of flow decompositions is very useful so I am describing this proof in the notes.

Lemma 16.1 *Any flow in G can be expressed as a sum of at most m path flows in G and a flow in G of value 0, where m is the number of edges in G.*

Proof:
Let f be a flow in G. If $|f| = 0$, we are done. (We can assume that the flow on each edge is the same as the capacity of the edge, since the capacities can be artifically reduced without affecting the flow. As a result, edges that carry no flow have their capacities reduced to zero, and such edges can be discarded. The importance of this will become clear shortly.) Otherwise, let p be a path from s to t in the graph. Let $c(p) > 0$ be the bottleneck capacity of this path (edge with minimum flow/capacity). We can reduce the flow on this path by $c(p)$ and we output this flow path. The bottleneck edge now has zero capacity and can be deleted from the network, the capacities of all other edges on the path is lowered to reflect the new flow on the edge. We continue doing this until we are left with a zero flow ($|f| = 0$). Clearly, at most m paths are output during this procedure. $\Box$

Since the entire flow has been decomposed into at most m flow paths, there is at least one augmenting path with a capacity at least $\frac{|f|}{m}$.

Let the max flow value be f^*. In the first iteration we push at least $f_1 \geq \frac{f^*}{m}$ amount of flow. The value of the max-flow in the residual network (after one iteration) is at most

$$f^*(1 - 1/m).$$

The amount of flow pushed on the second iteration is at least

$$f_2 \geq (f^* - f_1)\frac{1}{m}.$$

The value of the max-flow in the residual network (after two iteations) is at most

$$f^* - f_1 - f_2 \leq f^* - f_1 - (f^* - f_1)\frac{1}{m} = f^*(1 - \frac{1}{m}) - f_1(1 - \frac{1}{m}) \leq f^*(1 - \frac{1}{m}) - \frac{f^*}{m}(1 - \frac{1}{m}).$$

$$= f^*(1 - \frac{1}{m})^2.$$

Finally, the max flow in the residual graph after the k^{th} iteration is

$$\leq f^*(\frac{m-1}{m})^k.$$

What is the smallest value of k that will reduce the max flow in the residual graph to 1?

$$f^*(\frac{m-1}{m})^k \leq 1 \ ,$$

Using the approximation $\log m - \log(m-1) = \Theta(\frac{1}{m})$ we can obtain a bound on k.

$$k = \Theta(m \log f^*).$$

This gives a bound on the number of iterations of the algorithm. Taking into a consideration that a path with the highest residual capacity can be picked up in time $O(m + n \log n)$, the overall time complexity of the algorithm is $O((m + n \log n) m \log f^*)$.

Tarjan's book gives a slightly different proof and obtains the same bound on the number of augmenting paths that are found by the algorithm.

History

Ford-Fulkerson (1956)
Edmonds-Karp (1969) $O(nm^2)$
Dinic (1970) $O(n^2 m)$
Karzanov (1974) $O(n^3)$
Malhotra-Kumar-Maheshwari (1977) $O(n^3)$
Sleator-Tarjan (1980) $O(nm \log n)$
Goldberg-Tarjan (1986) $O(nm \log n^2/m)$