Analysis of Algorithms I: Edmonds-Karp and Maximum Bipartite Matching

Xi Chen

Columbia University

Last week we discussed the Ford-Fulkerson method:

```
1  set $f$ to be the zero flow
2  while there exists a simple path $p$ from $s$ to $t$ in $G_f$ do
3      use $p$ to modify $f$ and increase its value by $c_f(p)$
```

We proved the max-flow min-cut Theorem and it implies that when Ford-Fulkerson stops (meaning there is no path p from s to t or t is not reachable from s in G_f, the residual graph with respect to the current flow f), we have found a max flow f.

However, the Ford-Fulkerson method has very bad worst-case running time. In this class we show that if the path from s to t we pick in each while-loop is not just an arbitrary path from s to t, but one that minimizes the number of edges (or hops), then the number of while-loops is bounded from above by $O(nm)$.

Edmonds-Karp:

```
1  set $f$ to be the zero flow
2  while $t$ is reachable from $s$ in the residual graph $G_f$ do
3      find a shortest path (number of edges!) $p$ from $s$ to $t$
4      use $p$ to modify $f$ and increase its value by $c_f(p)$
```

Clearly we can find a shortest path from s to t using BFS in time $O(n+m) = O(m)$. Here the equation follows from the assumption that every vertex $v \in V$ is reachable from s in G so $m \geq n-1$. If we can show that the number of while-loops is bounded by $O(nm)$, then the total running time is $O(nm^2)$.

Before the proof, we start with some notation. Given a flow f in G, we use G_f to denote its residual graph and $c_f(u, v) > 0$ to denote the residual capacity of an edge (u, v) in G_f. Also recall that (u, v) in G_f implies that either $(u, v) \in E$ or $(v, u) \in E$. Given two vertices u and v, we let $\delta_f(u, v)$ denote the shortest path distance from u to v in G_f: the minimum length (or number of hops) of a path from u to v. (Again, it only depends on edges in G_f and has nothing to do with their residual capacities.)

Basic observation: Let p be an augmenting path from s to t in G_f. Recall that capacity of p is

$$c_f(p) = \min\left\{c_f(u,v) : (u,v) \text{ is on } p\right\} > 0$$

We say that an edge (u,v) on p is critical if $c_f(u,v) = c_f(p)$. It is easy to show that after augmenting f using p, (u,v) disappears from G_f. To see this, we let f' denote the new flow in G after augmenting f using p. Consider the following two cases:

1. If (u, v) is a forward edge, meaning that $(u, v) \in G$ and $c_f(u, v) = c(u, v) - f(u, v) > 0$, then we have

$$f'(u, v) = f(u, v) + c_f(p) = f(u, v) + c_f(u, v) = c(u, v)$$

2. If (u, v) is a reverse edge, meaning that $(v, u) \in G$ and $c_f(u, v) = f(v, u) > 0$, then we have

$$f'(v, u) = f(v, u) - c_f(p) = f(v, u) - c_f(u, v) = 0$$

In both cases, it is clear that (u, v) is no longer an edge in $G_{f'}$. Also prove by yourself the following claim: If (u, v) is not an edge in G_f but reappears in $G_{f'}$, then (v, u) must be on the path p.

We start the proof with the following crucial lemma:

Lemma

If we run Edmonds-Karp on $G = (V, E)$, then for every vertex $v \in V$, the shortest-path distance $\delta_f(s, v)$ in the residual graph G_f increases monotonically with each flow augmentation.

Let f be a flow and p be a shortest path from s to t in the residual graph. Let f' denote the new flow after augmenting f using p. Let v be a vertex in V, and we assume for contradiction that

$$\delta_{f'}(s, v) < \delta_f(s, v) \tag{1}$$

Without loss of generality, we let v be a vertex with the minimum $\delta_{f'}(s, v)$ among all vertices that satisfy (1).

Let $p = s \rightsquigarrow u \to v$ denote a shortest path from s to v in $G_{f'}$. So (u, v) is an edge in $G_{f'}$. We have

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1$$

The way we chose v implies that the distance from s to u did not decrease: $\delta_{f'}(s, u) \geq \delta_f(s, u)$. We claim that (u, v) cannot be an edge in G_f: Otherwise, if (u, v) is in G_f, then

$$\delta_{f'}(s, v) = \delta_{f'}(s, u) + 1 \geq \delta_f(s, u) + 1 \geq \delta_f(s, v)$$

contradicting with our assumption (1).

Now how come $(u, v) \notin G_f$ but $(u, v) \in G_{f'}$? By an earlier lemma, this can only happen if (v, u) is on p, a shortest path from s to t in G_f. Note that u is the successor of v on p. This implies (why?)

$$\delta_f(s, u) = \delta_f(s, v) + 1$$

To summarize, we have

$$\delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 2$$

contradicting with our assumption (1).

We prove that the total number of while-loops in Edmonds-Karp is $O(nm)$. First note that there are at most $2m$ pairs (u, v) of vertices that can appear as an edge in a residual graph during the execution of Edmonds-Karp: either $(u, v) \in E$ or $(v, u) \in E$. Second, note that in each while-loop, at least one edge on the augmenting path p must be critical, by definition. Finally we will show that for each of the $2m$ pairs (u, v) that may appear as an edge in a residual graph, it can serve as a critical edge for at most $n/2$ times, during the execution of Edmonds-Karp. It follows that the number of while-loops is no more than

$$2m \cdot (n/2) = O(nm)$$

To prove the last claim, we note that whenever (u, v) is a critical edge, it disappears from the residual graph after augmenting the flow. So before (u, v) becomes critical again, it has to first reappear in the residual graph. We show that from the time when (u, v) is critical (and disappears) to the time when it reappears in the residual graph, the distance from s to u in the residual graph must increase by at least 2. It then follows that (u, v) can serve as a critical edge for at most $n/2$ times because at any time, $\delta_f(s, u)$ is either $\leq n - 1$ or $+\infty$. Once $\delta_f(s, u)$ becomes $+\infty$, it remains $+\infty$ ever after and (u, v) can never be a critical edge again.

Now assume (u, v) is a critical edge in the ith while-loop and reappears again in the residual graph after the jth while-loop, where $i < j$. Let f denote the flow at the beginning of the ith while-loop and let f' denote the flow at the beginning of the jth while-loop. Since (u, v) is critical in the ith while-loop, we know (u, v) is on the augmenting path, a shortest path from s to t in G_f. This implies that $\delta_f(s, v) = \delta_f(s, u) + 1$. On the other hand, because (u, v) reappears after the jth while-loop, (v, u) must be on the augmenting path used in the jth while-loop, a shortest path from s to t in $G_{f'}$. This implies $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$ and thus,

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 \geq \delta_f(s, v) + 1 = \delta_f(s, u) + 2$$

So far we have been working on a seemingly very basic version of the maximum flow problem: only the edges have capacities and no antiparallel edges are allowed (i.e., $(u, v) \in E$ implies $(v, u) \notin E$). It turns out that many variants and extensions of maximum flow, some of which may seem much more difficult to solve, can all be easily reduced to this basic setting:

1. Antiparallel edges
2. Vertex capacities (Exercise in HW 7)
3. Multiple sources and sinks (Exercise in HW 7)

First, any algorithm for finding a maximum flow in the basic setting can be used to deal with graphs with antiparallel edges. To see this, we modify G to get G' as follows: For every two antiparallel edges (u, v) and $(v, u) \in E$, add a new vertex w and replace (u, v) with (u, w) and (w, v). Also set $c(u, w) = c(w, v)$ to be the capacity $c(u, v)$ of the original edge (u, v). It is clear that the new graph G' has no antiparallel edges and thus, is reduced. It can be shown that G' is essentially equivalent to G: a maximum flow in G' has the same value as a maximum flow in G. Moreover, there is clearly a one-to-one correspondence between flows in G' and flows in G. Given any maximum flow in G', we can use to construct efficiently a maximum flow in G.

Second, to work with multiple sources $s_1, \ldots, s_k$ and multiple sinks $t_1, \ldots, t_\ell$, we only need to add a new supersource vertex s and a new supersink vertex t, an edge from s to each s_i and an edge from each t_i to t, all with capacity $c(s, s_i) = c(t_i, t) = +\infty$. Denote the new graph by G'. One of the exercises asks you to show that a maximum flow in G' has the same value as a maximum flow in G. Also given any maximum flow in G', one can construct a maximum flow in G efficiently. So again, any algorithm for the basic setting can be used to deal with multiple sources and sinks. An exercise in HW 7 asks you to work on the extension with vertex capacities.

Moreover, many combinatorial problems can be cast as (or reduced to) maximum-flow problems, and we can use a maximum flow algorithm to solve them. The correctness of many such reductions crucially uses the following integrality theorem: Given a flow f in $G = (V, E)$, we say f is integer-valued if $f(u, v)$ is a nonnegative integer for all edges (u, v) in G.

Theorem

If the capacity function c is integer-valued, then there exists at least one maximum flow that is integer-valued. Moreover, the Ford-Fulkerson method (or any of its implementations, e.g., Edmonds-Karp) outputs a maximum and integer-valued flow.

It is clear that not every maximum flow is integer-valued, even if all the capacities are integers. The theorem only says that, if the capacities are integers, then there is at least one integer-valued maximum flow and Ford-Fulkerson finds one such flow. The proof uses induction, by showing that at the beginning of every while loop of Ford-Fulkerson, f is integer-valued. It is easy to see that if f is integer-valued at the beginning of a while-loop, then the residual capacities of edges in G_f are integers as well. Thus, the capacity of any augmenting path in G_f is an integer and the augmentation results in a new integer-valued flow.

We present an application of maximum flow: Maximum Bipartite Matching. Some notation: An undirected graph $G = (V, E)$ is bipartite if V can be partitioned into $L \cup R$ such that L and R are disjoint and every edge has one vertex from L and one from R. Given a bipartite undirected graph $G = (V, E)$, a matching is a subset of edges $M \subseteq E$ such that every vertex $v \in V$ is incident to at most one edge in M. A maximum matching is a matching of maximum cardinality. In the Maximum Bipartite Matching problem, we are given a bipartite undirected graph $G = (V, E)$, and are asked to find a maximum matching in G.

We give a reduction from Maximum Bipartite Matching to the maximum flow problem as follows: Given any bipartite graph $G = (V, E)$, we construct a directed graph $G' = (V', E')$ as follows: Add two new vertices source s and sink t so that

$$V' = V \cup \{s, t\}$$

Let $V = L \cup R$ be the vertex partition of G, then replace each undirected edge by a directed edge from L to R; For each $u \in L$ add a directed edge (s, u) and for each $v \in R$ add (v, t).

To summarize, E' is the following union:

$$\{(s,u) : u \in L\} \cup \{(u,v) : (u,v) \in E, u \in L\} \cup \{(v,t) : v \in R\}$$

So $|E'| = |E| + 2|V| = \Theta(|E|)$. The latter follows from the assumption that every vertex in G has deg ≥ 1 so $2|E| \geq |V|$. To complete the construction of a maximum flow instance, we also set the capacity of each edge in E' to be 1.

It is easy to show that

Lemma (Flow-Matching)

If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = |M|$. Conversely, if f is an integer-valued flow in G', then there is a matching M in G with $|M| = |f|$.

The first part is trivial. We prove the second part.

Given an integer-valued flow f in G', we let

$$M = \Big\{(u, v) : (u, v) \in E, u \in L, v \in R \text{ and } f(u, v) > 0\Big\} \qquad (2)$$

First we show that M is a matching. To see this, because f is integer-valued, we have $f(u, v) = 1$ if $(u, v) \in M$. Now if there is a vertex incident to more than one edge in M:

1. If this is a vertex $u \in L$, then the in-flow of u must be ≥ 2 and thus, $f(s, u) \geq 2 > 1 = c(s, u)$, contradiction.
2. If this is a vertex $v \in L$, then the out-flow of v must be ≥ 2 and thus, $f(v, t) \geq 2 > 1 = c(v, t)$, contradiction.

Prove by yourself that $|M| = |f|$.

Theorem

The cardinality of a maximum matching M in a bipartite graph G equals the value of a maximum flow f in G'. Moreover, let f denote the maximum flow found by Ford-Fulkerson, then M constructed from f in (2) is a maximum matching in G.

By the integrality theorem, we know there is a maximum and integer-valued flow f in G'. Let M be a maximum matching in G. If $|M| < |f|$, then by the Flow-Matching lemma, there is a matching M' in G such that $|M'| = |f| > |M|$, contradicting with the assumption that M is maximum. If $|M| > |f|$, then by the Flow-Matching lemma, there is an integer-valued flow f' such that $|f'| = |M| > |f|$, contradicting with the assumption that f is maximum. So $|f| = |M|$ and Ford-Fulkerson outputs such a flow.

This gives us the following algorithm for Max Bipartite Matching:

1. construct $G' = (V', E')$ from $G = (V, E)$
2. use Ford-Fulkerson to get an integer-valued max flow f in G'
3. use (2) to construct from f a max matching M in G

What is the running time of Ford-Fulkerson on G'? An upper bound we used earlier is $O(|E'| \cdot |f|)$, where f is a maximum flow in G'. It is clear that $|f|$ is the cardinality of a maximum matching in G, which is bounded from above by n. So the running time is $O(nm)$, since $O(|E'|) = O(|E|) = O(m)$.