15-750: Graduate Algorithms March 1, 2017

Lecture 19: Max Flow II: Edmonds-Karp

Lecturer: David Witmer *Scribes: Ellango Jothimurugesan, Ziqiang Feng*

1 Failure of Ford-Fulkerson

Recall from last time that in the setting of integral capacities, Ford-Fulkerson finds the max flow in $O(F(m+n))$ time, where F is the value of the max flow. That is, Ford-Fulkerson is only a psuedo-polynomial time algorithm. But things get even worse if the graph has real-valued capacities.

Theorem 1.1. *There exists a flow network with real capacities such that Ford-Fulkerson does not terminate. Furthermore, the values of the flows found may converge to some value arbitrarily far from the max flow.*

Proof. Consider the following graph, where each edge is labeled with its capacity. The number $\phi = (\sqrt{5}-1)/2$ is a solution to $\phi = 1 - \phi^2$.

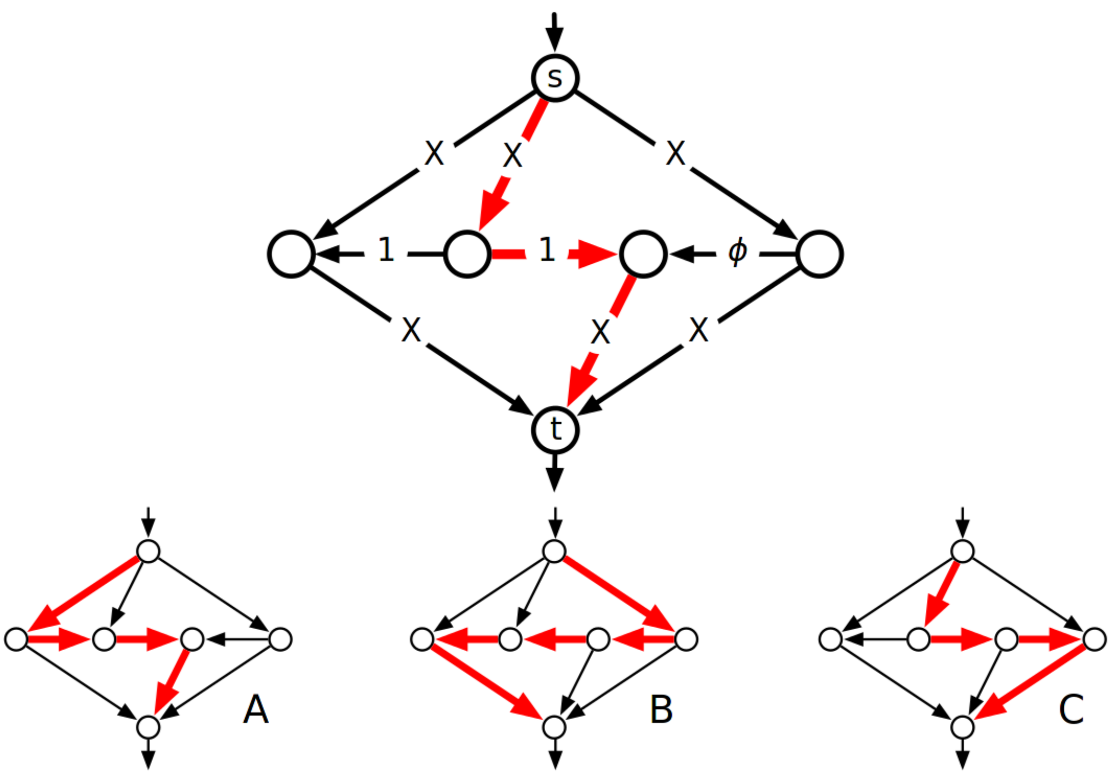


Figure 1: A simple graph where Ford-Fulkerson can fail, and three marked augmenting paths. Credit to Jeff Erickson [1] for the figure.

The max flow is $2X+1$, but we will show that Ford-Fulkerson can find a sequence of augmenting paths with flow values $1, \phi, \phi, \phi^2, \phi^2, \ldots$, so that the combined flow value converges to

$$1 + 2\sum_{i=1}^{\infty} \phi^i = 1 + \frac{2}{1-\phi} = 4 + \sqrt{5} < 7 \ll 2X + 1$$

for our choice of arbitrarily large X.

Suppose we first find the path through the center with flow 1. Observe that the resulting residual capacities along the three horizontal edges from left to right are: 1, 0, ϕ.

Now for the inductive step, assume that the residual capacities of the three edges are $\phi^{k-1}, 0, \phi^k$. We will show that we can find 4 augmenting paths with flows $\phi^k, \phi^k, \phi^{k+1}, \phi^{k+1}$, such that the new residual capacities for the three edges are $\phi^{k+1}, 0, \phi^{k+2}$ afterwards. The 4 paths are the following:

Augmenting path	Residual capacities for the three edges
1. Add flow of ϕ^k along path B.	$\phi^{k-1} - \phi^k = \phi^{k-1}(1-\phi) = \phi^{k-1}\phi^2 = \phi^{k+1}$
	$0 - (-\phi^k) = \phi^k$
	$\phi^k - \phi^k = 0$
2. Add flow of ϕ^k along path C.	ϕ^{k+1} (unchanged)
	$\phi^k - \phi^k = 0$
	$0 - (-\phi^k) = \phi^k$
3. Add flow of ϕ^{k+1} along path B.	$\phi^{k+1} - \phi^{k+1} = 0$
	$0 - (-\phi^{k+1}) = \phi^{k+1}$
	$\phi^k - \phi^{k+1} = \phi^k(1-\phi) = \phi^k\phi^2 = \phi^{k+2}$
4. Add flow of ϕ^{k+1} along path A.	$0 - (-\phi^{k+1}) = \phi^{k+1}$
	$\phi^{k+1} - \phi^{k+1} = 0$
	ϕ^{k+2} (unchanged)

□

2 Max Flow in Polynomial Time: Edmonds-Karp Algorithm

2.1 Edmonds-Karp 1: Pick largest-capacity augmenting path

As usual, suppose we have graph $G = (V, E)$, $|V| = n, |E| = m$.

Claim 2.1. *If the max flow of G is F, then there exists an $s \to t$ path with capacity of at least $\frac{F}{m}$.*

Proof. Imagine we delete all edges with capacity $< \frac{F}{m}$. We argue it *cannot* disconnect s from t, because otherwise it means there exists an s-t cut with capacity $< m \cdot \frac{F}{m} = F$ (we can have deleted at most m edges). But we know all cuts must be $\geq F$. So there must be an $s \to t$ path with capacity $\geq \frac{F}{m}$. □

Claim 2.2. *Edmonds-Karp 1 makes at most $O(m \ln F)$ iterations.*

Proof. Let F' be the max flow in the (changing) residual graph. In Edmonds-Karp 1, we iteratively reduce F' until it become < 1 (assume integrity of capacity).

Note Claim 2.1 holds for every residual graph too. So in each iteration, we pick the largest-capacity augmenting path with capacity $\geq \frac{F'}{m}$, reducing F' by a factor of $(1 - \frac{1}{m})$.

Start with $F' = F$, how many iterations x do we need to reduce F' to under 1?

$$F(1-\frac{1}{m})^x < 1$$
$$\implies F(1-\frac{1}{m})^x \approx Fe^{-\frac{x}{m}} < 1$$
$$\implies x = m \ln F$$

□

Finding the augmenting path with largest capacity. We use Algorithm 1 similar to Dijkstra's algorithm.

Algorithm 1 Finding the largest-capacity path

Let $c(v)$ be the capacity of the highest-capacity path $s \to v$, $v \in V$
Maintain a tree T of vertices for which we have computed $c(v)$
while $V \setminus T \neq \emptyset$ **do**
  for each $v \in V$ adjacent to T **do**
    $c(v) \leftarrow \max_{u \in T,(u,v)\in E}\{\min\{c_u, c_{(u,v)}\}\}$
  end for
  Add $v \in V \setminus T$ with largest $c(v)$ to T
end while

Runtime of Algorithm 1 By using a heap (recall previous lectures in this semester on heaps), the algorithm runs in $O(m \log n)$ time.

Total runtime By Claim 2.2, we run at most $O(m \ln F)$ iterations. In each iteration, we run Algorithm 1 that takes $O(m \ln n)$ time. Total runtime is thus $O(m^2 \ln F \ln n)$.

But the above runtime still depends on F, which can be huge and independent of the shape of the graph. *Can we get rid of F?* We do this in Edmonds-Karp 2 below.

2.2 Edmonds-Karp 2: Pick shortest augmenting path

Claim 2.3. *For all $v \in V \setminus \{s,t\}$, the shortest path distance $d_f(s,v)$ in the residual graph G_f is non-decreasing.*

Proof. by contradiction.

Let say after adding an augmenting path, there exist some vertices $W \subset V$ whose shortest path distances actually decrease. Let the residual graph before adding the path be G_f and after be $G_{f'}$. Let $v \in W$ be the vertex with the smallest shortest distance in $G_{f'}$: $v = \arg\min_{w\in W} d_{f'}(s,w)$

Note $d_{f'}(s,v) < d_f(s,v)$.

Let P be a shortest path from s to v in $G_{f'}$. There must exists a predecessor of v in P, let it be u.

Observation

1. $d_{f'}(s,u) = d_{f'}(s,v) - 1$
2. $d_{f'}(s,u) \geq d_f(s,u)$. Otherwise $u \in W$ and $d_{f'}(s,u) < d_{f'}(s,v)$ violates minimality of v

Claim 2.4.

$$(u, v) \notin E_f$$

Proof. by contradiction

If $(u, v) \in E$, then

$$\begin{aligned} d_f(s,v) &\le d_f(s,u) + 1 && \text{property of shortest path} \\ &\le d_{f'}(s,u) + 1 && \text{by observation 2} \\ &= d_{f'}(s,v) && \text{by definition of } u \end{aligned}$$

This contradicts $d_{f'}(s,v) < d_f(s,v)$. □

So, we have

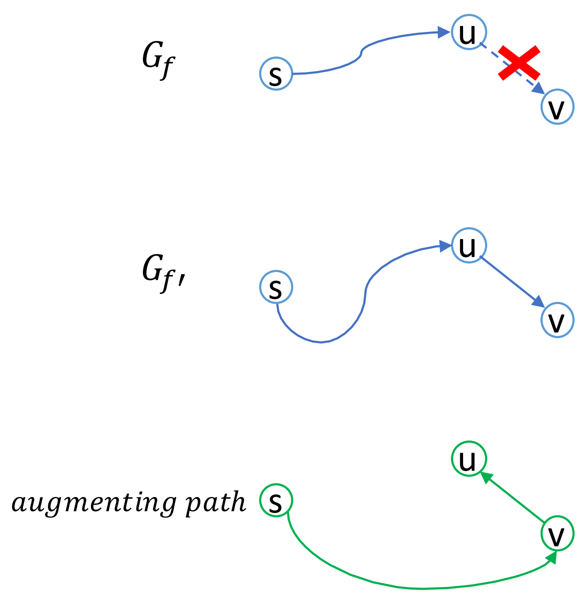
$$(u, v) \notin E_f, (u, v) \in E_{f'}$$

For this to happen, the augmenting path must have added a flow from v to u. Because the algorithm says we should pick the shortest path in G_f, it means the edge (v, u) is on the shortest path $s \to u$ in G_f.

$$\begin{aligned} d_f(s,v) &= d_f(s,u) - 1 && (v,u) \text{ is on the shortest path in } G_f \\ &\le d_{f'}(s,u) - 1 && \text{by observation 2} \\ &= d_{f'}(s,v) - 2 && \text{by observation 1} \end{aligned}$$

This contradicts $d_{f'}(s,v) < d_f(s,v)$.

Now, we have proved the shortest path distance $d_f(s,v)$ in the residual graph G_f is non-decreasing.



□

Claim 2.5. *Edmonds-Karp 2 has $O(mn)$ iterations.*

Proof.

Definition 2.6. An edge $e \in G_f$ is **critical** for an augmenting path P if P puts flow $c_f(e)$ in e. In other words, e is critical if it is "saturated" by P.

Observation

1. After augmenting path P, e will be removed from G_f.
2. On every augmenting path, at least one edge is critical. Otherwise we can increase the flow of the path.

Claim 2.7. *Each edge $e \in E$ can be critical for $\leq \frac{n}{2}$ times.*

Suppose $\hat{e} = (u, v) \in E_f$ is a *critical* edge in G_f. Since Edmonds-Karp says we should pick the shortest path,

$$d_f(s, v) = d_f(s, u) + 1$$

By observation 1, (u, v) will be removed from the residual graph G_f after augmenting the path. It can't re-appear until we put a flow on (v, u). Let f' be the flow when it happens. Again, since we are picking the shortest path:

$$\begin{aligned} d_{f'}(s, u) &= d_{f'}(s, v) + 1 \\ \therefore d_{f'}(s, u) &\geq d_f(s, v) + 1 && \text{by Claim 2.3} \\ &= d_f(s, u) + 2 \end{aligned}$$

So, every time an edge (u, v) becomes critical (again), its shortest path distance $d_f(s, u)$ increases by ≥ 2. Any shortest path on G_f must be shorter than n. Hence an edge can only become critical for $\leq \frac{n}{2} = O(n)$ times.

$\therefore$ The total number of critical edges is $\leq m \cdot O(n) = O(mn)$. □

Total runtime Claim 2.5 says Edmonds-Karp 2 runs in $O(mn)$ iterations. In each iteration, we run a BFS to find the shortest augmenting path, taking $O(m + n)$ time. So total runtime is $O(mn) \cdot O(m + n) = O(m^2 n)$.

References

[1] Jeff Erickson. Lecture 23: Maximum flows and minimum cuts. In *Algorithms*. http://jeffe.cs.illinois.edu/teaching/algorithms/notes/23-maxflow.pdf, January 2015. 1