Mathematical Foundations of Computer Science

CS 499, Shanghai Jiaotong University, Dominik Scheder

10 Network Flow

- Homework assignment published on Monday 2018-05-07
- Submit questions and first solution by Sunday, 2018-05-13, 12:00
- Submit final solution by Sunday, 2018-05-20.

Exercise 10.1. [From the video lecture] Recall the definition of the value of a flow: $\mathrm{val}(f) = \sum_{v \in V} f(s, v)$. Let $S \subseteq V$ be a set of vertices that contains s but not t. Show that

$$\mathrm{val}(f) = \sum_{u \in S, v \in V \setminus S} f(u, v) \ .$$

That is, the total amount of flow leaving s equals the total amount of flow going from S to $V \setminus S$. **Remark.** It sounds obvious. However, find a formal proof that works with the axiomatic definition of flows.

Proof. We know that $\forall u \in S \setminus \{s\}, \sum_{v \in V} f(u,v) = 0$. So

$$\begin{aligned}\sum_{u,v\in S} f(u,v) &= \sum_{v\in S} f(s,v) + \sum_{u\in S\setminus\{s\}} \sum_{v\in S} f(u,v) \\ &= \sum_{v\in V} f(s,v) - \sum_{v\in V\setminus S} f(s,v) + \sum_{u\in S\setminus\{s\}} (0 - \sum_{v\in V\setminus S} f(u,v)) \\ &= \mathrm{val}(f) - \sum_{u\in S, v\in V\setminus S} f(u,v).\end{aligned}$$

It's easy to prove that $\sum_{u,v\in S} f(u,v) = 0$, since

$$\sum_{u,v\in S} f(u,v) = \sum_{u,v\in S} -f(v,u) = -\sum_{v,u\in S} f(v,u) = -\sum_{u,v\in S} f(u,v).$$

So we know that $\mathrm{val}(f) - \sum_{u\in S, v\in V\setminus S} f(u,v)$. □

Exercise 10.2. Let $G = (V, E, c)$ be a flow network. Prove that flow is "transitive" in the following sense: If there is a flow from s to r of value k, and a flow from r to t of value k, then there is a flow from s to t of value k. **Hint.** The solution is extremely short. If you are trying something that needs more than 3 lines to write, you are on the wrong track.

Proof. Let C_1 be the min cut of s, r, C_2 be the min cut of r, t, C be the min cut of s, t, then $|C_1|, |C_2| \geq k$. If you want to disconnect s and t, you must disconnect either s, r or r, t, so $|C| \geq min(|C_1|, |C_2|) \geq k$, and there is a flow from s to t of value k. □

10.1 An Algorithm for Maximum Flow

Recall the algorithm for Maximum Flow presented in the video. It is usually called the Ford-Fulkerson method.

We proved in the lecture that f is a maximum flow and S is a minimum cut, by showing that upon termination of the while-loop, $\mathrm{val}(f) = \mathrm{cap}(S)$. The problem is that the while-loop might not terminate. In fact, there is an example with capacities in $\mathbb{R}$ for which the while loop does not terminate, and the value of f does not even converge to the value of a maximum flow. As indicated in the video, a little twist fixes this:

Algorithm 1 Ford-Fulkerson Method

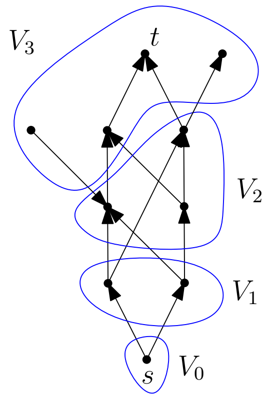
procedure FF$(G = (V, E), s, t, c)$
    Initialize f to be the all-0-flow.
    while there is a path p form s to t in the residual network G_f **do**
        $c_{\min} := \min\{c_f(e) \mid e \in p\}$
        let f_p be the flow in G_f that routes $c_{\min}$ flow along p
        $f := f + f_p$
    end while
    // now f is a maximum flow
    $S := \{v \in V \mid G_f$ contains a path from s to $v\}$
    // S is a minimum cut
    return (f, S)
end procedure

> **Edmonds-Karp Algorithm:** Execute the above Ford-Fulkerson Method, but in every iteration choose p to be a shortest s-t-path in G_f. Here, "shortest" means minimum number of edges.

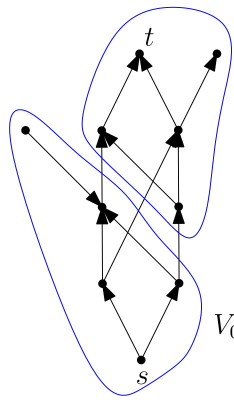
In a series of exercises, you will now show that this algorithm always terminates after at most $n \cdot m$ iterations of the while loop (here $n = |V|$ and $m = |E|$).

Definition 10.3. *Let (G, s, t, c) be a flow network and $k \in \mathbb{N}_0$. A k-*layering *is a partition of $V = V_0 \cup \cdots \cup V_k$ such that (1) $s \in V_0$, (2) $t \in V_k$, (3) for every edge $(u, v) \in E$ the following holds: suppose $u \in V_i$ and $v \in V_j$. Then $j \leq i + 1$. In words, point (3) states that every edge moves at most one level forward.*

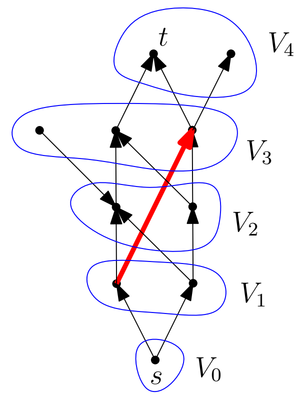
The figure below illustrates this concept: for one network we show two possible layerings and something that looks like a layering but is not:



A 3-layering.



A 1-layering.



Not a layering: the fat red edge "jumps ahead".

Exercise 10.4. Suppose the network (G, s, t, c) has a k-layering. Show that $\text{dist}(s, t) \geq k$. That is, every s-t-path in G has at least k edges.

Proof. Since the network (G, s, t, c) has a k-layering ($s \in V_0$ and $t \in V_k$) and $j \leq i + 1$ for every edge $(u, v) \in E(u \in V_i, v \in V_j)$, for any s-t path $< a_0, a_1, \ldots, a_n >$ $(a_0 = s, a_n = t)$, there must be at least k edges $(a_i, a_{i+1}) \in E$ $(a_i \in V_t, a_{i+1} \in V_{t+1})$. Therefore, $dis(s, t) = dis(a_0, a_n) \geq \sum dis(a_i, a_{i+1}) = k$. □

Exercise 10.5. Conversely, suppose $\text{dist}(s, t) = k$. Show that (G, s, t, c) has a k-layering.

Answer Since every s-t path $< a_0, a_1, \ldots, a_k >$ $(a_0 = s, a_k = t)$ has k edges, we can divide the whole E into $V_0, V_1, \ldots, V_k$ and put each a_i into V_i. Then we do Breadth-first search from s and include each point in the same level with a_i into V_i. When our search reaches t, the rest of the points will all be put into V_k. Assume there exist one edge $(u, v) \in E$ ($u \in V_i$ and $v \in V_j$), where $j > i + 1$, there must be a "level jump" step in the process of BFS which is impossible. Therefore, the division $(V_0, V_1, \ldots, V_k)$ satisfy the definition of layering and (G, s, t, c) has a k-layering.

Let (G, s, t, c) be a flow network and $V_0, \ldots, V_k$ a k-layering. We call this layering *optimal* if $\text{dist}_G(s, t) = k$. Here, $\text{dist}_G(u, v)$ is the shortest-path distance from s to t (measured by number of edges). If there is no path from s to t, we set $\text{dist}_G(s, t) = \infty$. In this case, no layering is optimal. For example, the 3-layering in the above figure is optimal, but the 1-layering in the middle of the above figure is not. Let us explore how layerings and the Ford-Fulkerson Method interact.

Exercise 10.6. Let (G, s, t, c) be a flow network and $V_0, V_1, \ldots, V_k$ be an optimal layering (that is, $k = \mathrm{dist}_G(s, t)$. Let p be a path from s to t of length k. Suppose we route some flow f along p (of some value $c_{\min} > 0$) and let (G_f, s, t, c_f) be the residual network. Show that $V_0, V_1, \ldots, V_k$ is a layering of (G_f, s, t, c_f), too. Obviously, condition (1) and (2) in the definition of k-layerings still hold, so you only have to check condition (3).

Answer In G we can know that G has $V_0, V_1, \ldots, V_k$ as an optimal layering. The difference between G and G_f is their edges. Assume that (u, v) is an edge of G and $u \in V_i, v \in V_j$. The edge has three situations:

(1) $j > i+1$ Because G has an optimal layering, so it's not existed in not only in G but also in G_f.

(2) $j + 1 < i$ If the flow covered this edge, the residual network will not have optimal layering because (v, u) is in G_f. Fortunately it's not really happened because flow f only along the path whose length is k. If flow covered (u, v) the length must be larger than k.

(3) The other situation Because both (u, v) and (v, u) is acceptable in G_f and the third condition of optimal layering is satisfied.

So we can prove that $V_0, V_1, \ldots, V_k$ is a layering of (G_f, s, t, c_f), too.

Exercise 10.7. Show that every network (G, s, t, c) has an optimal layering, provided there is a path from s to t.

Proof. For every network, we can find an optimal layering:

$$V_i = \{v | dist_G(s, v) = i\},\ i = 0, 1, \ldots, k-1$$
$$V_k = \{v | dist_G(s, v) \geq k\},\ k = dist(s, t)$$

It's obvious that condition (1) and (2) holds. If there's an edge between $u \in V_i$ and $v \in V_j$ and $j - i > 1$, we can find path from s to u then to v whose length is shorter than j, so (3) holds. And since $dist(s, t) = k$, this layering is optimal. □

Exercise 10.8. Imagine we are in some iteration of the while-loop of the Edmond-Karp method. Let $V_0, \ldots, V_k$ be an optimal layering of (G, s, t, c). Show that after at most m iterations of the while-loop, $V_0, \ldots, V_k$ ceases to be an optimal layering. **Remark.** Note that it is the *network* that changes from iteration to iteration of the while-loop, not the partition $V_0, \ldots, V_k$. We consider the partition $V_0, \ldots, V_k$ to be fixed in this exercise.

Proof. Since $V_0, \ldots, V_k$ is an optimal layering of (G, s, t, c), we know $dist_G(s,t){=}k$. In every iteration, we find a shortest s-t-path p, saturate at least one edge in path p, and replace this kind of edges with an opposite one. Notice that every edge $(u,v) \in p$ satisfies $u \in V_i,\ v \in V_j,\ i+1{=}j$.
Because there are m edges in G, there will be no such edges after at most m iterations. Then if we want to find an s-t-path in G, we have to contain (u,v) that $i+1 < j$. In this way, $dist_G(s,t) > k$, so $V_0, \ldots, V_k$ ceases to be an optimal layering of (G, s, t, c).
In conclusion, after at most m iterations of the while-loop, $V_0, \ldots, V_k$ ceases to be an optimal layering. □

Exercise 10.9. Show that the Edmonds-Karp algorithm terminates after $n \cdot m$ iterations of the while-loop. **Hint.** Initially, compute an optimal k-layering (which?). Then keep this layering as long as its optimal. Once it ceases to be optimal, compute a new optimal layering. Note that the Edmonds-Karp algorithm does not actually need to compute any layering. It's us who compute it to show that $n \cdot m$ bound on the number of iterations.

Proof. We've shown that after at most m iterations of the while loop, $dist_G(s,t)$ increases by at least 1. The possible $dist_G(s,t)$ ranges from 1 to n. When $dist_G(s,t)$ equals to $n+1$, it means there is no available path from s to t, and the algorithm terminates. So Edmonds-Karp algorithm terminates after $n \cdot m$ iterations of the while-loop. □

Exercise 10.10. Show that every network has a maximum flow f. That is, a flow f such that $\mathrm{val}(f) \geq \mathrm{val}(f')$ for every flow f'. **Remark.** This sounds obvious but it is not. In fact, there might be an infinite sequence of flows $f_1, f_2, f_3, \ldots$ of increasing value that does not reach any maximum. Use the previous exercises!

Proof. We know if there is no available argument path in G_f, f is a maximum flow.
By Exercise 10.9, we know after Edmonds-Karp algorithm terminates, there is no possible path from s to t in G_f. Also, the algorithm will terminates after $n \cdot m$ on any given flow network.

So every network's maximum flow can be calculated by Edmonds-Karp algorithm in finite time. In other words, every network has a maximum flow f. $\square$