Mathematical Foundations of Computer Science

CS 499, Shanghai Jiaotong University, Dominik Scheder

Spring 2019

10 Network Flow

- Homework assignment published on Thursday 2019-05-16
- Submit questions and first solution by Wednesday, 2019-05-22, 12:00
- Submit final solution by Wednesday, 2019-05-29.

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.

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.

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

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.

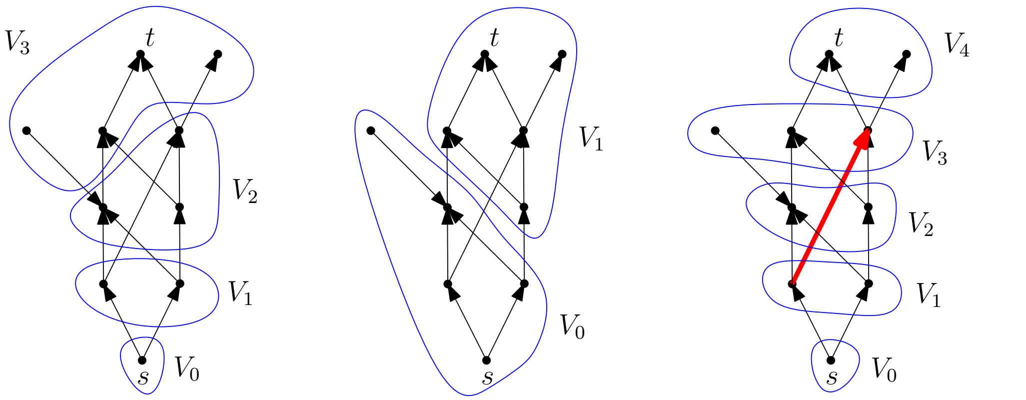
We proved in the lecture that f is a maximum flow and S is a minimum cut, by showing that upon termination of the while-loop, $\text{val}(f) = \text{cap}(S)$. The problem is that the while-loop might not terminate. In fact, there is an example with capacities in $\mathbb{R}$ for which the while loop 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:

> **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.

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

Let (G, s, t, c) be a flow network and $V_0, \dots, V_k$ 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 $V_0, V_1, \dots, V_k$ be an optimal layering (that is, $k = \text{dist}_G(s, t)$. 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 $V_0, V_1, \dots, V_k$ 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).

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

Exercise 10.8. Imagine we are in some iteration of the while-loop of the Edmonds-Karp algorithm. 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.

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.

Exercise 10.10. Show that every network has a maximum flow f. That is, a flow f such that $\text{val}(f) \geq \text{val}(f')$ 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!