Mathematical Foundations
of
Computer Science

CS 499, Shanghai Jiaotong University, Dominik Scheder

Group: NOIDEA

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. By flow conservation, $\sum_w f(v, w) = 0$, for all $v \in V \setminus \{s, t\}$.
Also by definition of the value of a flow, we have:

$$\begin{aligned}
\mathrm{val}(f) &= \sum_{w \in V} f(s, w) \\
&= \sum_{u \in S, w \in V} f(u, w) \\
&= \sum_{u \in S} \Big(\sum_{w_1 \in S} f(u, w_1) + \sum_{w_2 \in V \setminus S} f(u, w_2) \Big) \\
&= \sum_{u \in S, w_1 \in S} f(u, w_1) + \sum_{u \in S, w_2 \in V \setminus S} f(u, w_2) \\
&= \sum_{u \in S, v \in V \setminus S} f(u, v) + \sum_{u \in S, w_1 \in S} f(u, w_1)
\end{aligned}$$

Recall the skew-symmetry that $f(u, v) = -f(v, u)$, we have:

$$\sum_{u \in S, w_1 \in S} f(u, w_1) = \sum_{w_1 \in S, u \in S} -f(w_1, u) = -\sum_{u \in S, w_1 \in S} f(u, w_1) = 0 \qquad (1)$$

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

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. Suppose there is no flow from s to t of value k. Then there is a s-t cut $s \in S, t \in V \setminus S$ such that $c(S, V \setminus S) < k$. If $r \in S$, then there is a r-t cut $(S, V \setminus S)$ such that the capacity is less than k, so there's no flow of value k from r to t. Else if $r \in V \setminus S$, then there is a s-r cut $(S, V \setminus S)$, whose capacity is less than k and no flow value of k exists. In either of these cases, we reach a contradiction, so flow must be transitive. $\square$

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, $\mathrm{val}(f) = \mathrm{cap}(S)$.

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

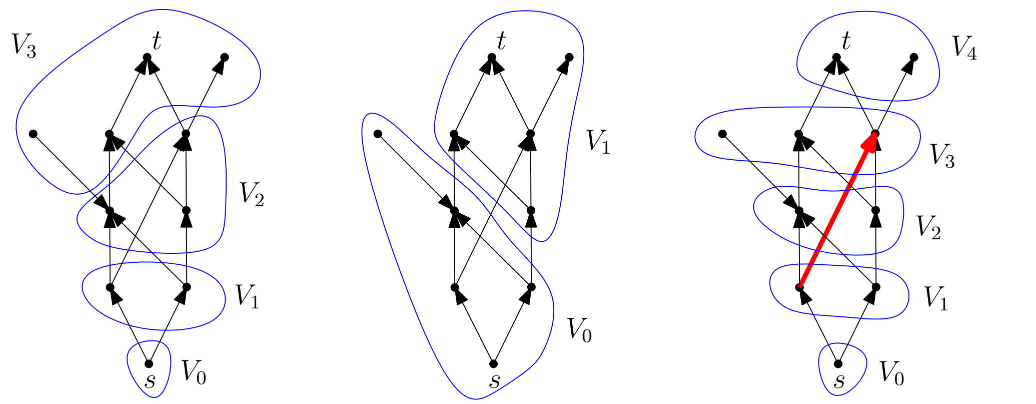
The problem is that the while-loop might not terminate. In fact, there is an example with capacities in $\mathbb{R}$ for which the while loop does not terminate, and the value of f does not even converge to the value of a maximum flow. As indicated in the video, a little twist fixes this:

> **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.

Proof. Prove by contradiction:
Suppose that there is a s-t-path, say s-v_1-$\ldots$-v_n-t, in G that has less than k edges $(n+1 < k)$. By definition, $s \in V_0$ and $t \in V_k$ and every edge moves at most one level forward. Hence, for the s-t-path, s-v_1-$\ldots$-v_n-t, we have:

$$\begin{aligned}
s &\in V_0 \\
v_1 &\in V_{a_1}, a_1 \leq 1 \\
v_2 &\in V_{a_2}, a_2 \leq a_1 + 1 \leq 2 \\
\ldots v_n &\in V_{a_n}, a_n \leq a_{n-1} + 1 \leq n \\
t &\in V_{a_{n+1}}, a_{n+1} \leq a_n + 1 \leq n + 1 < k
\end{aligned}$$

However, $t \in V_k$, a contradiction. □

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

Proof. Prove by construction:
$\text{dist}(s, t) = k$, suppose the shortest path between s and t are s-v_1-$\ldots$-v_{k-1}-t. Construction steps are as follows:

- First, add s to V_0, v_1 to V_1, ..., v_{k-1} to V_{k-1}, t to V_k.
- Consider the vertices that are directly connected with vertices that are already in the partitions, iteratively add them to any partitions as long as the adding satisfies *POINT 3* above.

- Recursively do step 2 above until no vertices left (since a flow network is a connected component as a whole).

□

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).

Proof. Consider the route operation, changes made on original edges in G are only limited on the capacity value and the added new edges each has a corresponding reversely directed edge in G, which is to say, that if any edge in G satisfied *POINT 3*, then any edge in G_f will not violate *POINT 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.

Proof. Since there is a path from s to t, there is a shortest path, say the $dist(s, t) = k$. According to **Exercise 10.5**, G has a $k - layering$, which is the optimal layering by definition. □

Exercise 10.8. Imagine we are in some iteration of the while-loop of the Ford-Fulkerson method. Let $V_0, \dots, V_k$ be an optimal layering of (G, s, t, c). Show that after at most m iterations of the while-loop, $V_0, \dots, V_k$ 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 $V_0, \dots, V_k$. We consider the partition $V_0, \dots, V_k$ to be fixed in this exercise.

Proof. Derived from the difinition of the layering. any flow f can not pass the partition forwarding, which means that flow go from V_i to $V_j (j > i + 1)$

directly is not permitted. Because there is at most m edges from V_i to V_{i+1}, after at most m iterations, if more flows want to pass the partition, they need to go back from V_{i+1} to V_i, making $dist(s, t) > k$ and $V_0, \ldots, V_k$ no longer the optimal layering. $\square$

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. For the Edmonds-Karp algorithm is a special cases of the Ford-Fulkerson algorithm, when using Edmonds-Karp algorithm, after at most m iterations of the while-loop, $V_0, \ldots, V_k$ will cease to be an optimal layering. That is, $dist(s, t)$ is no longer k.

Because the Edmonds-Karp always choose the short path from s to t as p, then $dist(s, t) = k'$ is now at least $k + 1$. And we have $1 \leq k \leq n$, we can find a new optimal layering at most n times.

Hence the Edmonds-Karp algorithm terminates after $n \cdot m$ iterations of the while-loop. $\square$

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!

Proof. Using the Edmonds-Karp algorithm, the iteration with be terminated in at most $n \cdot m$ iterations, thus will never producing a infinite sequence. That is, the Edmonds-Karp algorithm will find a maximum flow f. $\square$