A Study of the Function and Structure of Algorithms and Data Structures

Giannis Tyrovolas

February 8, 2020

1 Lecture 5: Network Flow I

Imagine flow networks as a set of pipes with water going through them. Since there's a direction to the water the graph is obviously a directed graph. Each edge is basically a tube. Each tube has a certain *integer* capacity and we think of start and finishing points. There's a lot of problems you can think of but an obvious question is what's the maximum amount of water you can send from the *source* to the *sink*.
Although not necessary, for our sake we'll consider flows with the following properties:

- no anti-parallel edges
- no edge enters s
- no edge leaves t

Now to write all this in a fancy way:

Definition 1.1 (Flow Networks)**.** A *flow network* is a tuple (G, s, t, c) where $G = (V, E)$ is a directed graph, $s, t \in V$ and $c\colon E \longrightarrow \mathbb{N}$.
The other conditions can be stated as:

- $(u, v) \in E \implies (v, u) \notin E$
- $\nexists u$ such that $(u, s) \in E$
- $\nexists v$ such that $(t, v) \in E$

We also need to define a particular flow:

Definition 1.2 (Flow)**.** A *flow* is an assignment of load to each edge of the form $f\colon E \longrightarrow \mathbb{R}_{\geq 0}$ That respects the capacity of each edge, i.e.

$$\forall e \in E \; 0 \leqslant f(e) \leqslant c(e)$$

Also much load enters a node leaves the node. That is except for s and t So:

$$\forall v \in V \sum_{e \text{ into } v} f(e) = \sum_{e \text{ leaving } v} f(e)$$

Value of f, denoted $|f|$ is the sum of the load of the edges leaving s.

Now, lets think of how we can maximise $|f|$. Before finding a solution lets think of an upper bound.

Say we partition V in A and B with $s \in A$ and $t \in B$. Then the maximum flow from s to t cannot exceed the total amount of flow leaving A. If we call this partition an s-t cut and $cap(A, B) \coloneqq \sum_{e \text{ leaves } A} c(e)$. We derive the following:

Lemma 1.3. For any s-t cut: $|f| \leqslant cap(A, B)$.

Hence, finding the minimum cut gives a potentially good bound on the maximum flow. But this has an easy corollary.

Corollary 1.4. For a partition (A, B) and a flow f:

$$|f| = cap(A, B) \implies f \text{ is a maximum flow and } (A, B) \text{ is a minimum cut.}$$

Proof. For a set S and a set of upper bounds B if $s \in S,\ b \in B$ and $s = b$ then s is maximum and b is minimum. □

Now for an actual solution. The first thing all computer scientists should try is a greedy algorithm. Of course this fails in this case.

Algorithm 1.5 (Greedy max-flow)**.** Pick any path from s to t and assign the maximum load possible. Repeat until no improvement can be made.

This won't work as an arbitrary assignment is inefficient. There might be multiple ways of going from node u to t but an arbitrary choice might "fill" a unique path from v to t.
Now there's the Ford-Flukerson algorithm which is the greedy algorithm that works. It works because if you can "fill up" a node. The idea is that you select a path, add as much as you can and then add reverse edges for each edge in that path.