Lecture23_NetworkFlows2

Friday, October 23, 2020 1:44 PM

Recap:

- **Flow value**: | f | = f (s, V).
- **Cut**: Any partition (S, T) of V such that s ∈ S and t ∈ T.
- **Lemma**. | f | = f (S, T) for any cut (S, T).
- **Corollary**. | f | <= c(S, T) for any cut (S, T).
- **Residual graph**: The graph G_f = (V, E_f) with strictly positive residual capacities c_f (u, v) = c(u, v) – f (u, v) > 0.
- **Augmenting path**: Any path from s to t in G_f.
- ***Residual capacity* of an augmenting path**

$$c_f(p) = \min_{(u,v)\in p} \{c_f(u,v)\}.$$

Max-flow, min-cut theorem

Theorem. The following are equivalent:

1. | f | = c(S, T) for some cut (S, T).
2. f is a maximum flow.
3. f admits no augmenting paths.

- These three statements are equivalent.

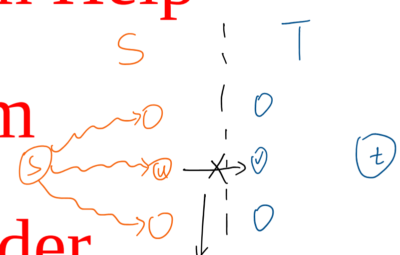
Proof.

(1) -> (2): Since | f | <= c(S, T) for any cut (S, T) (by the corollary from Lecture 22), the assumption that | f | = c(S, T) implies that f is a maximum flow.

- The flow value could never exceed the capacity of any cut. It is the upper bound!

(2) -> (3): If there were an augmenting path, the flow value could be increased, contradicting the maximality of f.

- The contrapositive statement: if there wasn't an augmenting path, it wouldn't be a maximum flow - > definition of augmenting path
- If the augmenting path is 1, we can improve the flow.



(3) -> (1): Suppose that f admits no augmenting paths. Define S = {v ∈ V : there exists a path in G_f from s to v}, and let T = V – S. Observe that s ∈ S and t ∈ T, and thus (S, T) is a cut. Consider any vertices u ∈ S and v ∈ T.

fully utilized.

We must have c_f (u, v) = 0, since if c_f (u, v) > 0, then v ∈ S, not v ∈ T as assumed. Thus, f (u, v) = c(u, v), since c_f (u, v) = c(u, v) – f (u, v). Summing over all u ∈ S and v ∈ T yields f (S, T) = c(S, T), and since | f | = f (S, T), the theorem follows.

- The path is the bottom neck

the flow value of the edge matches the max capacity.

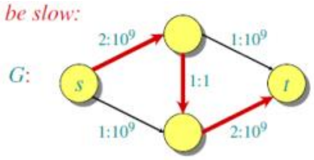
Ford-Fulkerson max-flow algorithm

Algorithm:

$f[u, v] \leftarrow 0$ for all $u, v \in V$
while an augmenting path p in G wrt f exists
  do augment f by $c_f(p)$

- ➢ Non-existing of augmenting path implies that we had the best solution
- How many augmentation should we expect?
  Each time we doing an augmentation, and the augmentation depends on the critical assignment, the critical edge of the augmenting path.
  -> it could be very slow!!
    - ○ Based on edge values not the number of vertices
    - ○ Based on capacity values
    - ➢ We do not prefer the running time depends on those values

Can be slow:



2 billion iterations on a graph with 4 vertices!

Edmonds-Karp algorithm

breadth-first augmenting path: a shortest path in G_f from s to t where each edge has weight 1. These implementations would always run relatively fast.

Since a breadth-first augmenting path can be found in O(E) time, their analysis, which provided the first polynomial-time bound on maximum flow, focuses on bounding the number of flow augmentations.

- How do we pick the augmenting path

Monotonicity lemma

Lemma. Let $\delta(v) = \delta_f(s, v)$ be the breadth-first distance from s to v in G_f. During the Edmonds-Karp algorithm, $\delta(v)$ increases monotonically.

Proof. Suppose that f is a flow on G, and augmentation produces a new flow f'. Let $\delta'(v) = \delta_{f'}(s, v)$. We'll show that $\delta'(v) \geq \delta(v)$ by induction on $\delta(v)$. For the base case, $\delta'(s) = \delta(s) = 0$.

For the inductive case, consider a breadth-first path $s \to \cdots \to u \to v$ in $G_{f'}$. We must have $\delta'(v) = \delta'(u) + 1$, since subpaths of shortest paths are shortest paths. Certainly, $(u, v) \in E_{f'}$, and now consider two cases depending on whether $(u, v) \in E_f$.

- Notice: residual network is constantly changing. Every time we have new flow assignment, we have a modification of residual network.
- Consider a flow before an augmentation and after an augmentation.

Case1

Case: $(u, v) \in E_f$.
We have

$$\begin{aligned} \delta(v) &\leq \delta(u) + 1 && \text{(triangle inequality)} \\ &\leq \delta'(u) + 1 && \text{(induction)} \\ &= \delta'(v) && \text{(breadth-first path),} \end{aligned}$$

and thus monotonicity of $\delta(v)$ is established.

Case2:

Case: $(u, v) \notin E_f$.
Since $(u, v) \notin E_f$, the augmenting path p that produced f' from f must have included (v, u). Moreover, p is a breadth-first path in G_f:

$$p = s \to \cdots \to v \to u \to \cdots \to t\,.$$

Thus, we have

$$\begin{aligned} \delta(v) &= \delta(u) - 1 && \text{(breadth-first path)} \\ &\leq \delta'(u) - 1 && \text{(induction)} \\ &\leq \delta'(v) - 2 && \text{(breadth-first path)} \\ &< \delta'(v)\,, \end{aligned}$$

thereby establishing monotonicity for this case, too. □

- What we are trying to show here is that we are making progress by the monotonicity we established, we are improve by 2.
- Imaging that we have a graph with shortest paths defined, veery time we perform an augmentation, many of the vertices' value are going up, they can go up at most n - 1. (upper bound)
  If they reaches the upper bound, we can say we can't improve more

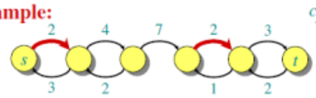
Counting flow augmentations

Theorem. The number of flow augmentations in the Edmonds-Karp algorithm (Ford-Fulkerson with breadth-first augmenting paths) is $O(VE)$.

Proof. Let p be an augmenting path, and suppose that we have $c_f(u, v) = c_f(p)$ for edge $(u, v) \in p$. Then, we say that (u, v) is ***critical***, and it disappears from the residual graph after flow augmentation.

Example: $c_f(p) = 2$

G_f:



The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

- Every time we do augmentation there must be at least one critical edge, and that edge would cause at least one vertex go up by 2. Hence, this particular edge can repeatedly appear and disappear at most n - 1 times.
- Shortest value start 0 potentially, and it went up all the way to n - 1 until that vertex become unreachable and not a part of the augmentation.

- Every edge can be a critical edge for at most O(n) augmentations. Since we have O(E) number of edges, O(VE) = all possible augmentations.

Running time of Edmonds-Karp

Distances start out nonnegative, never decrease, and are at most $|V| - 1$ until the vertex becomes unreachable. Thus, (u, v) occurs as a critical edge $O(V)$ times, because $\delta(v)$ increases by at least 2 between occurrences. Since the residual graph contains $O(E)$ edges, the number of flow augmentations is $O(V E)$.

Corollary. The Edmonds-Karp maximum-flow algorithm runs in $O(V E^2)$ time.

Proof. Breadth-first search runs in $O(E)$ time, and all other bookkeeping is $O(V)$ per augmentation. □

Best to Date

- The asymptotically fastest algorithm to date for maximum flow, due to King, Rao, and Tarjan, runs in $O(V E \log_{E/(V \lg V)} V)$ time.
- If we allow running times as a function of edge weights, the fastest algorithm for maximum flow, due to Goldberg and Rao, runs in time $O(\min\{V^{2/3}, E^{1/2}\} \cdot E \lg (V^2/E + 2) \cdot \lg C)$, where C is the maximum capacity of any edge in the graph.