COMS 4995-8: Advanced Algorithms (Spring'21) Feb 16, 2021

Lecture 11: Max Flow: Ford-Fulkerson, Max-Flow Min-Cut Duality

Instructor: *Alex Andoni* Scribe: *Haoran Zhu*

1 Review

i. **Graph**: $G(V, E, C)$ where V is the collection of vertices, E is the collection of edges, and C is the collection of capacity of each edge. m is the number of vertices, and n is the number of edges in graph.

ii. **Path Flow**: A particular type of flow which is a path.

Cycle Flow: A particular type of flow that starts from a node and comes back into that node.

iii. **Flow Decomposition Theorem**: for any valid flow f, f can be decomposed into a set of path flows $f_{p_1}, ..., f_{p_k}$ and a set of cycle flows $f_{c_1}, ..., f_{c_l}$ s.t.

$$f = \sum_{i=1}^{k} f_{p_i} + \sum_{j=1}^{l} f_{c_j}$$

and we have $k + l \leq m$

2 Flow Decomposition Theorem

Now we present the proof of flow decomposition theorem stated above.

Proof. In this proof, we will look at several cases and examine them individually.

Case 1: Suppose there are some positive flows coming out of s (start vertex) and at least some of those flows eventually reach t (destination vertex). Now we have a path P_1 from s to t. Let S be the set of edges on P_1

Define $\delta(P_1) = \min_{e \in S} f_e$. Thus, for a path flow vector f_{p_1}, we have $f_{P_1 e} = \delta(P_1) \cdot \begin{cases} 1, & \text{if } e \in S \\ 0, & \text{otherwise} \end{cases}$.

To write it more concisely,

$$f_{p_1} = \min_{e \in S} \ f_e \cdot (0, 0, 1, ..., 0, 1) \ \ [\text{1's in } e \in S]$$

$$|f_p| = min_{e \in S} \ f_e > 0$$

Case 2: Now suppose instead of reaching t, the flow returns to a vertex it has previously visited. In this scenario, we have a cycle C_1. Similarly, we define

$$f_{C_1} = min_{e \in C_1} \ f_e (0, 0, 1, ..., 0, 1) \ \ [\text{1's in } e \in C_1]$$

In either case, we can define a new flow $f' = f - f_{P_1}$ or $f' = f - f_{C_1}$. f' will still be a valid flow, and we continue decomposing f', just like how we decompose f above.

Case 3: Suppose there is no edge coming out of s that has non-trivial flow, but there are edges with positive flow in the graph. According to flow conservation, such flow must get to somewhere. However, it can not reach t as, by definition, the amount of flow coming out of s = the amount of flow reaching t. Thus, it must run into a cycle.

Now it seems like we are making progress. But when do we terminate? Each time we extract f_{P_i} or f_{C_j}, there must be some edge e with $f_e > 0$ becoming $f_e = 0$. In other words, each time we extract some flow path/cycle, we zero out the flow on some edge. Hence, $k + l \leq m$ $\square$

Now, using this theorem, we may examine Ford-Fulkerson algorithm, which is used to solve max-flow problem.

3 Ford-Fulkerson Algorithm

Before diving into Ford-Fulkerson algorithm, we shall examine two algorithms which, while being able to return some non-trivial flow, are not guaranteed to return max flow.

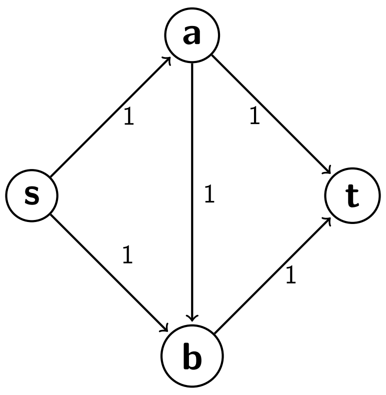
Algorithm 0: find a path from s to t using edges e with $c_e > 0$.

Suppose f^* is the max flow. If $|f^*| > 0$, then by flow decomposition theorem, we know that there exists a path flow f_p with $|f_p| > 0$. Therefore, we can find it by finding any path from s to t on edges e with $c_e > 0$. BFS or DFS can be applied to determine reachability. However, we are not guaranteed to find the max flow in this manner.

Algorithm 0.5: the main idea is to iteratively find a path P from s to t using edges with updated capacity $c_e - f_e$ (**residual capacity**), where f is the current flow.

```
while we can find a path $P$ satisfying below condition do
    find a path $P$ from $s$ to $r$ such that $\forall e \in P, c_e - f_e > 0$.
    $f' := f + \mathbf{1}_P \cdot \min_{e \in P}(c_e - f_e)$        note: $\mathbf{1}_P$ is a vector with 1 in $e \in P$
    $f := f'$
```

While algorithm 0.5 serves as improvement on algorithm 0, it is still not guaranteed to find the max flow. Consider following graph:



Obviously, the max flow in above graph is $|f^*| = 2$. However, if we follow algorithm 0.5 and find the path $s \to a \to b \to t$ in the first iteration, we can no longer another path with all positive residual capacity. In this case, algorithm 0.5 returns flow with value 1, which is not the max flow.

Hence, we would like a modification of the algorithm that reflects the ability to "push back" flow in the direction opposite to the current flow. This idea of routing flow on edges is implemented in Ford-Fulkerson algorithm. Now let us define residual graph.

Residual Graph: $G_f = (C, E', C^f)$ is the residual graph for a flow f, where $\forall e = (u,v) \in E$,

i. *forward edge (F-edge):* if $c_{uv} > f_{uv}$, then add edge (u,v) to E' and set $c^f_{uv} = c_{uv} - f_{uv}$

ii. *backward edge (B-edge):* if $f_{uv} > 0$, then add edge (v,u) to E' and set $c^f_{vu} = f_u v$

Finally, we now dive into Ford-Fulkerson algorithm.

Algorithm 1: Ford-Fulkerson algorithm

```
Result: Max flow from $s$ to $t$
Initialize: set flow $f := \vec{0}$ and $G_f = G$
while $\exists\ s \to t$ path $P \in G_f$ do
    let $\delta = \min_{e \in P} c^f_e > 0$;
    for every $e = (u,v) \in P$ do
        if $(u,v)$ is F-edge then
            $f_{uv} := f_{uv} + \delta$;
        else
            $f_{vu} := f_{vu} - \delta$;
        end
    end
    Reconstruct $G_f$;
end
```

4 Max-Flow Min-Cut Duality

Regarding FF algorithm, we have 3 claims.

1. Flow f, the result of FF algorithm, is a valid flow
2. FF algorithm terminates with the max flow f
3. Claim 3 is about how many iterations FF algorithm requires (runtime). We will discuss it in next lecture.

Now let us start proving claim 1 - FF algorithm returns a valid flow.

Proof. We first fix $f, P, \delta = \min_{e \in P} c_e^f$

For any forward edge e, $\delta \leq c_e^f = c_e - f_e$. Thus, for f_e, the flow on e, we have $f_e + \delta \leq c_e$. Current flow on e is not larger than the capacity.

For any backward edge $e = (u, v)$, we have $c_{uv}^f = f_{vu} > 0$. Since $\delta \leq c_{uv}^f = f_{vu}$, we have $f_e - \delta \geq f_e - f_{vu} = 0$.

Also, if we construct a residual graph, we can easily see that for any vertex in the graph, the in-flow and out-flow are the same in magnitude. Thus, flow conservation is satisfied.

Since f satisfy all three constraints, we conclude that it is a valid flow. □

Proof for claim 1 is rather trivial. But the proof for claim 2 is more complex. Before starting the proof, we first define **s-t cut** and introduce **max-flow min-cut duality**.

s-t cut: A s-t cut is $(S, \bar{S} = V - S)$ where S is a set of vertices in the graph such that $s \in S,\ t \in \bar{S}$. We define

$$C(S, \bar{S}) = C(S) \triangleq \sum_{u \in S,\ v \in \bar{S}} c_{uv}$$

A problem related to finding max-flow is finding min $s-t$ cut – find the $s-t$ cut $(S, \bar{S})$ that minimizes $C(S)$.

A crucial observation we have is that $\forall$ flow f and cut S, $|f| \leq C(S)$. For edge (u, v) s.t. $u \in S, v \notin S$, $f_{uv} \leq c_{uv}$ and $f_{vu} \geq 0$. Therefore, $|f| \leq \sum_{u \in S,\ v \notin S} c_{uv} = C(S)$

Max-Flow Min-Cut Duality: For any graph G, we have:

$$\max{}^{s-t}\text{-flow in } G = \min{}^{s-t}\text{-cut in } G$$

Now, to prove claim 2, we will prove the following 3 statements are equivalent:

1. f is the max flow in G
2. G_f has no **augmenting path** from s to t. (An augmenting path is a path P from s to t s.t. $\forall e \in P,\ c_{fe} > 0$)
3. $\exists$ cut $(S, \bar{S})$ s.t. $C(S) = |f|$

Remark: By showing above three statements are equivalent, we have proved that FF algorithm terminates with max flow because the algorithm only stops when G_f has no augmenting path, which corresponds to statement 2. By equivalence, we have statement 1 that f, returned by FF algorithm, is the max flow. We will first prove statement 1 implies statement 2, then prove statement 2 implies statement

3, eventually showing statement 3 implies statement 1.

Now let us start by proving statement 1 implies statement 2. [**1 → 2**]

Proof. Assume for contradiction that there exists an augmenting path P in G_f.
Let $\delta = \min_{e \in P} c_e^f > 0$. With this augmenting path P, we can send more flow along some path from s to t. Now we have

$$|f'| = |f| + \delta > |f|$$

.
Obviously, f' is a strictly larger flow than f, which contradicts statement 1 that f is the max flow. □

Now we need to prove statement 2 implies statement 3. [**2 → 3**]

Proof. Assuming G_f has no augmenting path, then there must be a subset of vertices that we cannot reach from s. With this observation, we define S as the set of vertices reachable from s. Obviously, $s \in S, t \notin S$. We will then show the $s - t$ cut $(S, \bar{S})$ is exactly the one we desire in statement 3.

A key observation is that for any edge (u, v) crossing the cut that goes from S to $\bar{S}$, (u, v) must be saturated. In other words, $\forall\ (u, v) \in E$, where $u \in S,\ v \notin S$, we have $f_{uv} = c_{uv}$. Otherwise, if $f_{uv} \neq c_{uv}$, then we would have a positive value on c_{uv}^f and be able to reach some vertices in $\bar{S}$ from s. Consequently, $\forall\ (v, u) \in E$, where $v \notin S,\ u \in S$, we have $f_{vu} = 0$. Therefore,

$$C(S) = \sum_{u \in S,\ v \notin S} c_{uv} = \sum_{u \in S,\ v \notin S} f_{uv}$$

Recall that we have following two equations,

$$\sum_{(s,u)} f_{su} - \sum_{(u,s)} f_{us} = |f| \qquad \text{(definition of } |f|\text{)}$$
$$\forall\ u \notin \{s, t\}, \sum_{(u,v)} f_{uv} - \sum_{(v,u)} f_{vu} = 0 \qquad \text{(flow conservation)}$$

Now if we sum $f_{uv} - f_{vu}$ for all nodes $u \in S$, we will notice that if $v \in S$, we will add $f_{uv} - f_{vu}$ to $f_{vu} - f_{uv}$. So they will cancel out. The only remaining terms will be for edges (u, v), where $u \in S,\ v \notin S$. The sum will be $|f|$, which is obvious from above two equations. Mathematically,

$$\sum_{u \in S,\ v \notin S} f_{uv} - \sum_{u \in S,\ v \notin S} f_{vu} = |f|$$

We have proved $C(S) = \sum_{u \in S,\ v \notin S} f_{uv}$, and $\sum_{u \in S,\ v \notin S} f_{vu} = 0$. Hence, $C(S) = |f|$ as desired. □

Eventually, we need to prove statement 3 implies statement 1. [**3 → 1**]

Proof. We know from previous analysis that for any flow f and cut $(S, \bar{S})$, $|f| \leq C(S)$. Therefore, if $|f| = C(S)$ (by statement 3), then f must be the max flow. □

Now that we have proved these three statements are equivalent, we conclude that FF algorithm terminates with the max flow f.

5 Forecast

In the next lecture, we will analyze the run time of FF algorithm (claim 3). Specifically, how many iterations does FF algorithm required? Moreover, how can we choose the most efficient path(s) so that we can minimize the number iterations?