CSL 3010: Design and Analysis of Algorithms July-November 2022

Lecture 8: Max-Min Cut

Lecturer: Dr. Saket Saurabh *Scribe: Mukul, Sawan, Jaimin, Tanay*

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

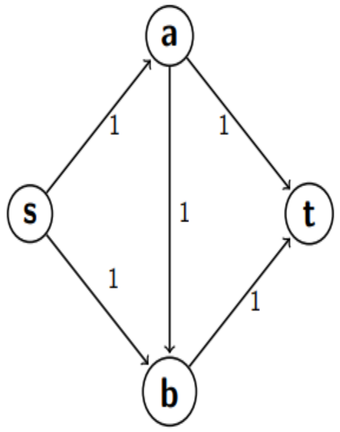
Suppose f is the max flow. If $|f| > 0$,then by flow decomposition theorem, we know that there exists a path flow f_p with $|f_p| > 0$. Therefore, we can find it by finding any path from s to t on edges e with $c_e > 04$ BFS or DFS can be applied to determine reachability. However, we are not guaranteed to find the max flow in this manner.

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 for all e in P, Ce      Fe > 0.
    f':= f + 1p     min[e in P] (Ce - Fe) note: 1P is a vector with 1 in e in P
    f := f'
```

Listing 1: Pseudo code for the algorithm

While algorithm above serves as improvement on algorithm 0, it is still not guaranteed to find the optimal solution.
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 $\rightarrow$ a $\rightarrow$ b $\rightarrow$ 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: Gf = (C, $E^{'}$, C^f) is the residual graph for a flow f, where $\forall e = (u, v) \in E$,

$i.forwardedge(F-edge): if$c$_{uv} > f_{uv}$, then add edge (u, v) to E' and set $c_{uv} = c_{uv} f_{uv}$
ii. backward edge (B-edge): if $f_{uv} > 0$, then add edge (v, u) to E' and set $c_{vu} = f_{uv}$
Finally, we looked at ford-fulkerson algirithm.

```
Result: Max flow from s to t
Initialize: set flow f := vector(0) and Gf = G
while there exists s -> t path P in Gf do
    let delta = min(e in P) cf(e) > 0;
    for every e = (u, v) in P do
        if (u, v) is F-edge then
            fuv := fuv +   ;
        else
            fvu := fvu        ;
        end
    end
    Reconstruct Gf ;
end
```

Listing 2: Pseudo code for subset sum

0.2 Max Flow Min Cut Duality

From Ford Fulkerson algorithm, we have 3 theorems:
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. Proof. We first fix f, P, $\delta = min e \in P$c^f_e For any forward edge e, $\delta <= c^f_e = c - efe$. Thus, for f_e, the flow on e, we have $fe + \delta c_e$. Current flow on e is not larger than the capacity.
For any backward edge e = (u, v), we have $c^f_{uv} = f_{vu} > 0$. Since $\delta <= c^f_{uv} = f_{vu}$, we have $f_e \delta >= f_e f_{vu} = 0$. 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, S' = V S) where S is a set of vertices in the graph such that s S, t S'. We define C(S, S') = C(S) ,

$$\textstyle\sum_{u \in v, v \in S'} c_{uv}$$

A problem related to finding max-flow is finding min s-t cut - find the st cut (S, S') that minimizes C(S). A crucial observation we have is that flow f and cut S, $|f| <= C(S)$. For edge (u, v) s.t. u S, v $\notin S$,f$_{uv} <= c_{uv}$ and $f_{vu} >= 0$. Therefore, $|f| <= \sum u \in v, v \in S'\ c_{uv} = C(S)$

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

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

Now, to prove claim 2, we will prove the following 3 statements are equivalent:
1. f is the max flow in G
2. Gf has no augmenting path from s to t. (An augmenting path is a path P from s to t s.t.
$\forall eP$,c$_{fe} > 0$)
3. $\exists cut(S, S') s.t. C(S) = |f|$

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

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

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 $\rightarrow$ 3]
Proof : $Assuming G\_f has no augmenting path, then there must be a subset of vertices that we cannot reach from s. With this obser$
$S, t \notin S We will then show the s$ - t$cut($S , S'$) is exactly the one we desire in statement 3. A key observation is that for any edge (u, v) cro$
$E, where u \in S, v \notin S, we have$ f$_{uv} = c_{uv}$. Otherwise, if $f_{uv}! = c_{uv}$, then we would have a positive value on c_{uv}^f and be able to reach some vertices in S' from s. Consequently, $\forall (v, u) \in E, where v \notin S, uS, we have$ f$_{vu}$ = 0 Therefore,

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

From the following two equations:

$$\sum_{s,u} f_{su} = \sum_{u,s} f_{us} = |f|$$

$$\forall u \notin s, t, \sum_{u,v} f_{uv} = \sum_{v,u} f_{vu} = 0$$

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 $\in S, v \notin S. The sum will be |f|, which is obvious from above t$

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

We have proved C(S) = $\sum_{u \in S, v \notin S} f_{uv} and \sum_{u \in S, v \notin S} f_{uv} = 0$. Hence, C(S) =—f— as desired. Eventually, we need to prove statement 3 implies statement 1. [3 $\rightarrow$ 1]
$Proof. We know from previous analysis that for any flow f and cut (S, S'), |f| <= C(S). Therefore, if$—f— =
C(S)$(by statement 3), then f must be the max flow. Now that we have proved these three statements are equivalent, we conclude that$
$FF algorithm terminates with the max flow f.$