CS 344 Maximum Flow Problem

Consider a directed graph $G = (V, E)$ where for each edge $e = (u, v)$ in E there is a capacity $c_e > 0$. There is a distinguished vertex s called source and a distinguished vertex t called destination. We assume there is at least one directed path from s to t.

Definition 1. A flow f is a vector in E-dimensional space, one dimension f_e for each edge e. Two kinds of constraints must be satisfied:

(i) Each $e \in E$ the following *bound constraints* are satisfied:

$$0 \le f_e \le c_e,$$

(ii) For each vertex $u \in V$, other than s or t, the following *flow conservation equations*:

$$\sum_{(w,u)\in E} f_{(w,u)} = \sum_{(u,v)\in E} f_{(u,v)}.$$

This means for each vertex u the sum of outgoing flows is equal to the sum of incoming flows.

Definition 2. (Size of Flow). The size of a flow f is

$$size(f) = \sum_{(s,u)\in E} f_{(s,u)}.$$

(i.e. the sum of outgoing flow at s, which by the conservation equations is the same as incoming flow at t.)

Maximum Flow Problem. Find a flow f with largest size.

Algorithm. Given a flow f, compute the *residual graph* induced by f as follows: Let $G^f = (V, E^f)$ (same vertex set as G). Its edges are defined based on the value of f. If for an edge $e \in E$, f_e is positive (note $f_e \le c_e$), then create an edge in E^f but with reverse direction and put its capacity $c^f_e = f_e$. Also, if $c_e - f_e > 0$, then create an edge into E^f with the same direction is in E with capacity $c^f_e = c_e - f_e$. Additionally, if for an edge $e \in E$, $f_e = 0$, then place the same edge into E^f with the same capacity $c^f_e = c_e$. Thus G^f can end up with at most $2|E|$ edges.

Next do a DFS (or BFS) on G^f to check if there is directed path from s to t. If there is no such a path then we have the optimal flow from s to t. Otherwise, we use the path to construct a new flow f'_e with larger size. This is done by increasing the flow along this path.

We now prove if thee is no directed path from s to t in G^f then the current flow is optimal. First some preliminary results.

Definition. An (s, t)-cut is any partition of V into two sets L and R such that $s \in L$ and $t \in R$. The capacity of the cut is the sum of the capacities of edges in G from L to R. This is denoted by $capacity(L, R)$.

Lemma 1. Given a flow f, for any (s, t)-cut we have

$$size(f) \le capacity(L, R).$$

Proof. This is obvious since any flow f from s to t must use edges from L to R. $\Box$

Theorem 1. (Maximum Flow-Min Cut Theorem) The maximum flow is equal to the minimum of capacity over all (s, t)-cuts.

Proof. Suppose f is an optimal flow. Consider the corresponding G^f. Then by optimality of f no directed path can be found in G^f from s to t. Now let L consist of all vertices that can be reached from s in G^f and let R be $V - L$ (the complement of L). Then (L, R) gives a partition of V. We claim $size(f) = capacity(L, R)$. By Lemma 1 left-hand-side is less than or equal the right-hand-side. To prove equality we note:

(1) For any edge $e = (u, v) \in E$ which is from L to the R we must have $f_e = c_e$. Otherwise, the edge (u, v) is also in G^f and v can be reached from s which contradicts the definition of L, R.

(2) For any edge $e = (v, u) \in E$ which is from R to L, f_e must be zero. Otherwise, $f_e > 0$ means the edge (u, v) is in E^f, contradicting that v cannot be reached from s.

These two facts imply that the size of f equals the capacity of the (L, R) cut. $\Box$

Complexity of the Algorithm. Suppose that each edge capacity c_e is a natural number. Let C be the sum of the capacities. Then in each iteration we can increase the flow by at least one unit. This means the complexity of the algorithm is $O(C \times |E|)$. This is because each iteration does a DFS in $O(|E|)$ time.

Remark. This complexity is not polynomial in the size of the problem. Imagine when C is very large.

If in each iteration we compute the shortest path from s to t in G^f then it can be shown that the complexity is $O(|V| \times |E|^2)$. (each iteration would take $O(|V| \times |E|)$ due to shortest path.