Notes by Samir Khuller.

9 Branching Problem

Consider a directed graph $G^D = (V, E)$ with a specified vertex r as the root.

A *branching* of G is a subgraph that is a spanning tree if the directions of the edges are ignored, and which contains a directed path from r to every other vertex.

Given a cost function $c : E \to R^+$, the problem is to obtain a branching of least total cost. This problem looks similar to the minimum spanning tree problem, but is actually much trickier. The greedy algorithm does not solve it!

Notice the following property about a min-cost branching.

Lemma 9.1 *G^D contains a branching rooted at r if and only if for all $v \in V$, G^D contains a path from r to v.*

The proof is quite easy and left as an exercise. Notice that if T is a branching then every vertex in T (except for r) has in-degree 1.

We are going to assume that the costs on edges are non-negative (this is not restrictive, since we can add a large constant K to the cost of each edge without affecting the structure of the optimal solution).

Consider all the incoming edges to a single vertex v ($\neq r$). A branching will contain exactly one of these edges. Hence we can subtract any constant from the costs of these edges without affecting the structure of the optimal solution. For every vertex, we add a negative constant to the incoming edges to v, so as to make the min-cost incoming edge have cost 0. We can now assume that the cheapest incoming edge to any single vertex has cost 0. If we consider the subgraph of 0 cost edges, we can either find a path from r to v by walking backwards on the 0 cost edges, or we find a 0 cost cycle. If we find a path, we simply delete the path and continue. If we find a cycle, we shrink the cycle to a single node and find a branching in the smaller graph. This gives an easy polynomial time algorithm for the problem.

We now prove that the algorithm actually gives us the min cost branching. The proof uses a notion of "combinatorial dual" that should remind you of linear programming duality. (In fact, if you write the IP for Min Cost Branching and relax it to an LP, and then write the dual of this LP, you get exactly the problem of computing a weight function $w(F)$ for each r-cut. In class we called this a dual variable y_S.)

We define a *rooted r-cut* as follows: partition V into sets $S, V \setminus S$ with $r \in S$. The set of edges from S to $V \setminus S$ is called a rooted r-cut.

Let $\mathcal{F}$ be the set of all rooted r-cuts. We define a weight function $w : \mathcal{F} \to R^+$. This weight function is said to be *valid* if for any edge e

$$c(e) \geq \sum_{F \text{ contains } e} w(F).$$

Theorem 9.2 *If w is a valid weight function then*

$$c(B) \geq \sum_{F \in \mathcal{F}} w(F)$$

where B is any branching.

Proof:

The proof of this theorem is easy. Consider any branching B, and add up the costs of the edges in the branching.

$$c(B) = \sum_{e \in B} c(e)$$

$$c(B) \geq \sum_{e \in B} \sum_{F \text{ contains } e} w(F)$$

Each $w(F)$ term occurs in the summation as many times as the number of edges of F that are in common with B. Since each F is crossed by every branching, for any rooted r-cut F and B, we have $\mid F \cap B \mid \geq 1$. Hence the theorem follows. □

We now prove that there is a valid weight function for which $\sum_{F \in \mathcal{F}} w(F)$ is equal to $c(B)$, where B is the branching produced by the algorithm. Thus B is an optimal branching.

Theorem 9.3 *If w is a valid weight function then*

$$\min_B(c(B)) = \max_w \sum_{F \in \mathcal{F}} w(F).$$

Proof:

The proof is by induction on (number of vertices + number of non-zero cost edges). The proof is easy to verify for a graph with 2 nodes (base case). We will prove the $\leq$ part of the equality (the other follows from the previous theorem).

The first stage of the algorithm reduces the cost on all edges incoming to v (let's call this set of edges F_v). We change the cost of the min-cost branching by K (cost of cheapest edge in F_v). The new graph has the same number of vertices, but fewer non-zero cost edges, so we can apply the induction hypothesis. F_v is a rooted r-cut that has weight 0 in the new graph G'. In G^D we define its weight to be K; it can be checked that this yields a valid weight function. The other rooted r-cuts get the same weight as they received in the shrunken graph.

The other case is when the algorithm contracts a 0 cycle. We now get a graph with fewer vertices, so we can apply the induction hypothesis to the shruken graph G'. We can obtain a min-cost branching for G^D with the same cost as that for G'. The rooted r-cuts for G' correspond to the rooted r-cuts for G^D for which all the vertices of the 0 cycle are in the same set of the partition. We keep their weights unchanged, and define the weights of the other rooted r-cuts to be 0. The weight function clearly achieves the bounds in the lemma. □