Network Design Foundation
Homework 2 - Solution

Instructor: Mohammad T. Hajiaghayi

1. Let $T = (V, E)$ be a tree and let $D = \langle d_1, \ldots, d_k \rangle$ denote the set of demands. We reduce the prize-collecting steiner forest problem on trees to the minimum weighted vertex cover problem. We make a bipartite graph $H = (V', E')$ where for each edge $e \in E$ we put a vertex $p_e$ in $V'$ and also for every $d_i \in D$ we put a vertex $q_i$ in $V'$. We put an edge between $p_e$ and $q_i$ iff the edge $e$ is on the unique path between the endpoints of demand $d_i$ in $T$. The weight of $p_e$ would be the length of $e$ and the weight of $q_i$ would be the prize of $d_i$. Recall that a vertex cover is a subset $S$ of vertices which every edge has at least an endpoint in $S$. One can easily show that the cost of the minimum vertex cover in $H$ is equal to the cost of the minimum steiner forest in $T$. We note that for any vertex $q_i$, we have to include $q_i$ in our vertex cover unless we have chosen all the $p_e$ s adjacent to $q_i$ which essentially means that we have chosen all the edges in the path between the endpoints of $d_i$. Vertex cover can be modeled as a set cover problem where the frequency of each member is two, and thus it has a 2-approximation algorithm (consider the edges as elements and the vertices as sets). However since $H$ here is a bipartite graph there is a polynomial time exact algorithm based on primal-dual ideas.

For general graphs, we simply embed the graph into a collection of trees, paying at most a factor of $O(\log n)$ on distances and thus on the approximation ratio. Then we solve the problem on trees with the 2-approximation algorithm (or the exact algorithm) described above.
2. The **MULTICUT PROBLEM** is defined as follows.

Given:

- $G = (V, E)$
- $c : E \rightarrow \mathbb{R}^+$ Weight on edges
- $P = \{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$

where $s_i, t_i \in V$ and $P$ is a set of $k$ source-sink pairs.

A multicut is a set of edges $E' \subseteq E$ such that $\forall i$ there is no path from $s_i$ to $t_i$ in the graph $G' = (V, E - E')$.

The objective is to find a multicut of lowest cost where $\text{cost}(E') = \sum_{e \in E'} c_e$

**On Trees**: We first consider the **MULTICUT PROBLEM** when the underlying graph is a tree. In a tree, there exists a unique path $P_i$ between any pair $s_i$ and $t_i$. The solution is thus to remove at least one edge from each $P_i$ so that the total cost of edges removed is minimized.

The problem can be formulated as the following Integer program.

\[
\begin{align*}
\min & \quad \sum_{e \in E} c_e x_e \\
\text{s.t.} & \quad \sum_{e \in P_i} x_e \geq 1; \forall i \\
& \quad x_e \in \{0, 1\}
\end{align*}
\]

Taking dual of the above and applying LP relaxation we get,

\[
\begin{align*}
\max & \quad \sum_i y_i \\
\text{s.t.} & \quad \sum_{i: e \in P_i} y_i \leq c_e; \forall e \in E \\
& \quad y_i \geq 0
\end{align*}
\]
Rough Primal dual Algorithm:

(a) Start from dual feasible solution \( y_i = 0, \forall i \) which corresponds to the primal infeasible solution \( E' = \emptyset \).
(b) Find a path \( P_i \) whose constraint is not satisfied in the primal and increase corresponding \( y_i \) in the dual till the dual constraint becomes tight for some \( e' \in E \).
(c) Add \( e' \) to \( E' \) and repeat from step 2 till the primal becomes feasible.

Now, in order to get a good bound, we need to modify the above basic algorithm a little.

Good Primal dual Algorithm:

(a) Set \( y_i = 0, \forall i \)
(b) Set \( E' = \emptyset \)
(c) Let \((s'_i, t'_i)\) be the pair such that the lowest common ancestor of \( s'_i \) and \( t'_i \) is the deepest among all unsatisfied pairs.
   Increase \( y'_i \) corresponding to such a pair till dual constraint becomes tight for some \( e' \in E \). i.e. \( \sum_{i: e' \in P_i} y_i = c'_e \)
(d) \( E' = E' \cup e' \)
(e) Repeat steps 3 and 4 till \( E' \) becomes feasible
(f) Prune:
   Loop through edges in \( E' \) in the reverse order in which they were added and remove an edge \( e' \) from \( E' \) if its removal still gives a feasible multicut.
   Let the final set be called \( M \)
(g) Return \( M \)

Analysis:

Applying standard primal-dual analysis techniques, we get

\[
\text{cost(Alg)} = \sum_{e \in M} c_e \tag{7}
\]
\[
= \sum_{e \in M} \sum_{i: e \in P_i} y_i \tag{8}
\]
\[
= \sum_{i} |M \cap P_i| y_i \tag{9}
\]
In lemma 1, we prove that $|M \cap P_i| \leq 2$.

$$\therefore \text{cost}(\text{Alg}) \leq \sum_i 2y_i$$ (10)

$$\therefore \text{cost}(\text{Alg}) \leq 2 \sum_i y_i$$ (11)

$$\therefore \text{cost}(\text{Alg}) \leq 2(\text{cost(dualfeasible)})$$ (12)

$$\therefore \text{cost}(\text{Alg}) \leq 2\text{OPT}_{\text{dual}}$$ (13)

$$\therefore \text{cost}(\text{Alg}) \leq 2\text{OPT}$$ (14)

Hence we get a 2-approximation algorithm for multicut problem on trees.

**Lemma 1:** $|M \cap P_i| \leq 2, \forall i$

**Proof:** Let $v \in V$ be the lowest common ancestor of the pair $s_i, t_i$. To prove $|M \cap P_i| \leq 2$, it is sufficient to show that at most one edge from the path joining $s_i$ to $v$ (and hence by symmetry path joining $t_i$ to $v$) belongs to $M$.

Assume that two edges $e$ and $e'$ from the path joining $s_i$ to $v$ are present in $M$ and let wlog $e'$ be the deeper of the two.

Now since both $e$ and $e'$ are in $M$, none of them were deleted in the pruning step of the algorithm. Consider the instant when $e'$ is being checked while pruning. Since $e'$ is not removed, it means there must be present a pair $(s_i', t_i')$ such that $e'$ lies on the unique path joining $s_i'$ and $t_i'$ but $e$ does not lie on this path. Clearly the lowest common ancestor of $s_i'$ and $t_i'$ lies between $e$ and $e'$, let it be $u$.

Now consider the steps taken while constructing the set $E'$. Since $s_i'$ and $t_i'$ and has a deeper lowest common ancestor ($u$) than $s_i$ and $t_i$ ($v$), it will be tested first. If at this step, the edge $e'$ becomes tight and is added to $E'$, then there would be no path from $s_i$ and $t_i$ as well and $e$ would never be added to $E'$ and we get a contradiction.

On the other hand, consider that some edge $e''$ becomes tight instead of $e'$, then clearly $e'$ can only be added to $E'$ after adding the edge $e''$. Hence, in the pruning step, $e'$ would be considered before $e''$ and would be pruned out because there is already an edge ($e''$) in the multicut which disconnects $s_i'$ and $t_i'$. Hence, we again get a contradiction.

Thus, we have proved that at most one edge from the path connecting a terminal to the lowest common ancestor can lie in the final solution $M$. Clearly, this implies the claim that $|M \cap P_i| \leq 2, \forall i$. \qed

**0.1 On general graphs**

We can apply the Racke (and Feige) result to embed the general graph $G$ into a probability distribution over spanning subtrees. We know that the
expected congestion is at most $\tilde{O}(\log n)$.

**Algorithm:**

(a) Calculate the load on every edge

(b) Obtain the $O(n\log n)$ trees in support of the distribution so that the expected congestion is at most $\tilde{O}(\log n)$ by considering the load on the edge instead of capacity.

(c) Solve the multicut problem on each tree using the above 2-approximation algorithm

(d) Map the solution from each tree to a solution on the original graph

(e) Return the best solution w.r.t the cost of the solution on the graph

**Analysis:**

Let $OPT$ be the cost of the minimum multicut in the graph. Let $OPT_t$ be the mapping of this optimal multicut on the subtree $t$ i.e. it contains those edges in $OPT$ which are present in $t$ only the capacity of each edge is now taken to be its load.

Now to due to the probabilistic decomposition, we have

$$\tilde{O}(\log n)OPT \geq \sum_t P_t OPT_t$$

Now, let $C_t$ be the cost of the minimum multicut in tree $t$. Therefore, we get

$$\tilde{O}(\log n)OPT \geq \sum_t P_t C_t$$

Now, let $Sol_t$ be the cost of the solution returned by above 2-approximation algorithm on tree $t$,

$$\therefore Sol_t \leq 2C_t$$

$$\therefore C_t \geq \frac{1}{2}Sol_t$$

Substituting above, we get

$$\tilde{O}(\log n)OPT \geq \sum_t \frac{1}{2}P_t Sol_t$$

$$\therefore 2\tilde{O}(\log n)OPT \geq \sum_t P_t Sol_t$$

$$\therefore \tilde{O}(\log n)OPT \geq \sum_t P_t Sol_t$$
Let $C^*$ be the cost of the best solution among all trees. Therefore, we get

$$\tilde{O}(\log n) \OPT \geq \sum_t P_t C^*$$

$$\geq C^* \sum_t P_t$$

$$\geq C^*$$

$$\therefore C^* \leq \tilde{O}(\log n) \OPT$$

Hence we have a $\tilde{O}(\log n)$ approximation for the minimum multicut problem on general graphs.
3. Similar as in the lectures, consider a zero sum game in which the player MAP chooses an admissible mapping to a tree, and the player EDGE chooses an edge. The value of the game for EDGE is the relative load on $e$ in the mapping, and hence EDGE wishes to maximize the relative load whereas MAP wishes to minimize it. A probabilistic mapping $\lambda$ is a randomized strategy for MAP. Choosing a distribution $\beta$ over the edges is a randomized strategy for EDGE. By definition, the payoff of choosing tree $T$ and edge $e$ is $\frac{\text{load}_T(e)}{c_e}$. Thus by minimax theorem we have:

$$\min_\lambda \max_\beta \sum_e \sum_T \lambda_T \beta_e \frac{\text{load}_T(e)}{c_e} = \max_\beta \min_\lambda \sum_e \sum_T \lambda_T \beta_e \frac{\text{load}_T(e)}{c_e}.$$ 

Recall that the congestion of a distribution $\lambda$ is defined as $\max_e \sum_T \lambda_T \frac{\text{load}_T(e)}{c_e}$. First we prove that if for all $\beta$ there exist a tree $T$ such that $\sum_e \beta_e \frac{\text{load}_T(e)}{c_e} \leq \rho$ then there exist a distribution $\lambda$ on the collection of trees with congestion $\rho$. Let $\beta^*$ denote the strategy of EDGE which maximizes the right hand side of the minimax theorem. We have $\exists T \sum_e \beta^*_e \frac{\text{load}_T(e)}{c_e} \leq \rho$. Thus the pure strategy of choosing $T$ would decrease the EDGE’s profit to at least $\rho$. Therefore we have:

$$\rho \geq \min_\lambda \sum_T \sum_e \lambda_T \beta^*_e \frac{\text{load}_T(e)}{c_e} = \max_\beta \min_\lambda \sum_T \sum_e \lambda_T \beta_e \frac{\text{load}_T(e)}{c_e} = \min_\lambda \max_\beta \sum_T \sum_e \lambda_T \beta_e \frac{\text{load}_T(e)}{c_e}.$$ 

Observe that a pure strategy $e$ for EDGE would have the profit $\sum_T \lambda_T \frac{\text{load}_T(e)}{c_e}$. The last inequality above shows that for some $\lambda$, EDGE’s profit would be at most $\rho$ for all possible strategies, including all the pure strategies. Therefore $\max_e \sum_T \lambda_T \frac{\text{load}_T(e)}{c_e} \leq \rho$ and $\lambda$ is a distribution with congestion $\rho$.

To prove the other way of the theorem, one can simply use the reverse of the above argument. Note that for a set of numbers $S$ and any distribution $\alpha$ over $S$, we have $\min_{x \in S} x \leq \sum_{x \in S} \alpha_x x \leq \max_{x \in S} x$. 


4. We solve the problem using a dynamic programming. Let $T = (V,E)$ be a tree rooted at $r$. WLOG, we may assume that $T$ is binary, otherwise we could insert dummy nodes and do not count them in sections. Define $OPT(v,x)$ for $v \in V$ and $x \in [\lvert V \rvert]$ as the minimum cost partitioning of the subtree rooted at $v$ such that the section which contains $v$ has exactly $x$ number of vertices. Since we assume that $T$ is binary, we call the two children of $v$, left $l$ and right $r$. We can compute $OPT(v,x)$ recursively by this formula:

$$OPT(v,x) = \min \begin{cases} 
\text{cost}_{e=(v,l)} + \text{cost}_{e=(v,r)} & \text{if } x == 1 \\
\text{cost}_{e=(v,l)} + OPT(r, x-1) & \text{if we cut the edge between } v \text{ and } l \\
\text{cost}_{e=(v,r)} + OPT(l, x-1) & \text{if we cut the edge between } v \text{ and } r \\
\forall y:1 \leq y \leq x-2OPT(l,y) + OPT(r, x-y-1) & \text{if we do not cut the edges to the children}
\end{cases}$$

we omit the details about handling the initial cases and the dummy nodes. Clearly, the final solution would be $OPT(r,\lvert V \rvert/2)$. 

5. We need to give an approximation-preserving reduction from GST to DST. Given an instance of GST $\Gamma = < G = (V,E), r, S >$ we make an instance $\Psi = < G' = (V',E'), r, T >$ such that the cost of optimum GST in $\Gamma$ is the same as the cost of optimum DST in $\Psi$. Here $S = < S_1, \ldots, S_k >$ is the collection of groups in GST and $T$ is the set of terminals in DST. We define $T = \{ t_1, \ldots, t_k \}$ and $V' = V \cup T$. For any edge $e = (u,v) \in E$ we put two directed edges $e'_1 = (u,v)$ and $e'_2 = (v,u)$ in $E'$ with the same cost as $e$. Also for any set $S_i \in S$ we put an edge of cost zero from $v$ to $t_i$ for all $v \in S_i$. Let $\psi$ be a solution to the DST in $\Psi$. The only way to reach a terminal $t_i$ is through the vertices in $S_i$. Thus $\psi$ is touching at least a vertex in each $S_i$ and therefore considering $\psi$ in $\Gamma$ would give us a solution for GST with the same cost as $\psi$. Also for any set $S_i \in S$ we put an edge of cost zero from $v$ to $t_i$ for all $v \in S_i$. Let $\gamma$ be a solution to the GST in $\Gamma$. Consider $\gamma$ rooted at $r$ and make all edges directed from top to bottom, getting farther from $r$. Since $\gamma$ touches at least one vertex in each $S_i$ we can connect $r$ to $t_i$ through that vertex. Thus we also have a solution to the DST in $\Psi$ with the same cost as $\gamma$. So the optimum solution for both problems have the same cost. Note that the size of $\Psi$ is at most twice the size of $\Gamma$ and therefore the $\Omega(\log^{2-\epsilon} n)$ inapproximability result for GST carries on to DST.