601.435 / 601.635 Approximation Algorithms **Spring 2024**
Homework #4 **Due:** April 4, 2024, 9:00am

Reminder: you may work in groups of up to three people, but must write up solutions entirely on your own. Collaboration is limited to discussing the problems – you may not look at, compare, reuse, etc. any text from anyone else in the class. Please include your list of collaborators on the first page of your submission. Many of these problems have solutions which can be found on the internet – please don't look. You can of course use the internet (including the links provided on the course webpage) as a learning tool, but don't go looking for solutions.

Please include proofs with all of your answers, unless stated otherwise.

Student: Ha Manh Bui (hbui13@jhu.edu)

1 Tree embeddings with total cost (Exercise 8.12) (33 points)

We saw in class that given a metric space (V, d), the FRT tree embedding algorithm is a randomized algorithm that generates a tree metric (V', T) such that 1) $V' \supseteq V$, 2) $d(u, v) \leq d_T(u, v)$ for all $u, v \in V$ and T in the support of the algorithm, and 3) $\mathbf{E}[d_T(u, v)] \leq O(\log n) \cdot d(u, v)$ for all $u, v \in V$ (where the expectation is taken over the randomized choice of T). Note that this is for the distribution over trees generated by FRT – any individual tree might distort some distance quite badly. But sometimes we will want a bound on a single tree, so we might ask for something else: a single tree in which the *total cost* is small.

Suppose that we are given a metric space (V, d) and costs $c : \binom{V}{2} \to R^+$ (where costs have nothing to do with distances, and $\binom{V}{2}$ denotes all unordered pairs of nodes in V). Give a polynomial-time randomized algorithm that returns a tree metric (V', T) for V (so V is the leaves of T) such that:

1) $d(u, v) \leq d_T(u, v)$ for all $u, v \in V$ with probability 1, and
2) $\sum_{u,v \in V} c(u, v) d_T(u, v) \leq O(\log n) \sum_{u,v \in V} c(u, v) d(u, v)$ with very high probability (at least $1 - \frac{1}{2^n}$).

Hint: Try to *use* FRT rather than *modifying* it.

Reusing the result from the class, we know that the FRT gives a polynomial-time randomized algorithm s.t. with probability 1, for all $u, v \in V$,

$$\begin{cases} d(u, v) \leq d_T(u, v) \\ \mathbb{E}\left[d_T(u, v)\right] \leq \mathcal{O}(\log n) d(u, v). \end{cases} \tag{1}$$

Therefore, we get

$$\begin{aligned} \mathbb{E}\left[\sum_{u,v \in V} c(u, v) d_T(u, v)\right] &= \sum_{u,v \in V} \mathbb{E}\left[c(u, v) d_T(u, v)\right] \\ &\leq \mathcal{O}(\log n) \sum_{u,v \in V} c(u, v) d(u, v) \text{ (by Eq. 1).} \end{aligned} \tag{2}$$

Since this inequality holds over the random choices of the trees from the randomized algorithm, there must exist some tree T generated by the algorithm such that the inequality holds. So, let's sample using FRT to select the optimal tree T, we know that $\sum_{u,v\in V} c(u,v)d_T(u,v)$ is a nonnegative random variable and $\mathcal{O}(\log n)\sum_{u,v\in V} c(u,v)d(u,v) > 0$, hence, using Markov's inequality, we obtain

$$p\left(\sum_{u,v\in V} c(u,v)d_T(u,v) \geq \mathcal{O}(\log n)\sum_{u,v\in V} c(u,v)d(u,v)\right) \leq \frac{\mathbb{E}\left[\sum_{u,v\in V} c(u,v)d_T(u,v)\right]}{\mathcal{O}(\log n)\sum_{u,v\in V} c(u,v)d(u,v)}. \quad (3)$$

Consider $B(u,r)$ be the ball around u of radius r, by FRT, on level i, w cut (u,v), $u \in B(w,r_i)$ and $v \notin B(w,r_i)$, we know that r_i is chosen uniformly at random from $[2^{i-1},2^i)$, since r_0 is chosen uniformly at random from $\left[\frac{1}{2},1\right)$, charge the cost c to the cut, by Eq. 2 and $c:\binom{V}{2}\to R^+$, we get

$$\mathbb{E}\left[\sum_{u,v\in V} c(u,v)d_T(u,v)\right] \leq \frac{1}{2^n}\mathcal{O}(\log n)\sum_{u,v\in V} c(u,v)d(u,v), \quad (4)$$

plugging into Eq. 3, we obtain

$$p\left(\sum_{u,v\in V} c(u,v)d_T(u,v) \geq \mathcal{O}(\log n)\sum_{u,v\in V} c(u,v)d(u,v)\right) \leq \frac{\mathbb{E}\left[\sum_{u,v\in V} c(u,v)d_T(u,v)\right]}{\mathcal{O}(\log n)\sum_{u,v\in V} c(u,v)d(u,v)} \leq \frac{1}{2^n}, \quad (5)$$

i.e., $\sum_{u,v\in V} c(u,v)d_T(u,v) \leq \mathcal{O}(\log n)\sum_{u,v\in V} c(u,v)d(u,v)$ with at least $1-\frac{1}{2^n}$.

2 Capacitated Dial-a-Ride (Exercise 8.11) (67 points)

In the Capacitated Dial-a-Ride problem, we are given a metric (V,d), a single vehicle with a given integer capacity $C > 0$ located at a given node $r \in V$, and k source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$. At each source s_i there is an item which must be delivered to the sink t_i by the vehicle. The vehicle can carry at most C items at a time. The goal is to find the shortest tour that starts and ends at the r, delivers each item from its source to its destination without exceeding the vehicle capacity, and returns to r. Note that such a tour may visit a node of V multiple times. We assume that the vehicle is allowed to temporarily leave items at any node in V.

(a) (34 points) Suppose that (V,d) is actually a tree metric (V,T) (so T is a tree with vertex set V, not just with leaves V). Give an $O(1)$-approximation algorithm for this case.

Hint: for each $i \in [k]$ there is a unique path $p(i)$ in T from s_i to t_i. First show that without loss of generality, you can restrict your attention to the subtree induced by r and the source-sink pairs. For each edge e which remains, let $\ell(e)$ denote the number of these paths which use edge e. Prove that OPT needs to traverse e at least $\max(1,\ell(e)/C)$ times. Design an algorithm which traverses each edge at most $O(1)\cdot\max(1,\ell(e)/C)$ times.

(b) (33 points) Give a randomized $O(\log n)$-approximation for the general Capacitated Dial-a-Ride problem, where $n = |V|$. Please do this formally – it's not enough to say "FRT embeds with distortion $O(\log n)$, so we lose an $O(\log n)$ factor". Hint: you might want to use the steiner point removal result discussed in the lecture notes and (briefly) discussed in class.

(a) For each $i \in [k]$, there is a unique path $p(i)$ in T from s_i to t_i. Let the turning point of i be the highest node in the tree on the path $p(i)$. Let v be the turning point of i. The path $p(i)$ first moves upwards from s_i to v, and then downwards from v to t_i.
Consider vertices lie on a path $p(i)$ for some item i. Let T' be the Steiner tree spanning all these vertices and node r. Then, we know that any tour that starts and ends at r, that performs all the pickups and deliveries must visit all these vertices and r. Thus any tour must traverse every edge in T' at least 2 times. Therefore, without loss of generality, we can restrict our attention to the subtree T', and edges not in T' are discarded.
For each edge e that remains, let $\ell(e)$ denote the number of paths that use edge e. Let decompose $\ell(e) = \ell_u(e) + \ell_d(e)$, where $\ell_u(e)$ and $\ell_d(e)$ are the number of paths that pass upwards and downwards through the edge e respectively. Since the vehicle can carry at most C items at a time, any tour that performs all the pickups and deliveries must traverse the edge e at least $2\max(\lceil \ell_u(e)/C \rceil, \lceil \ell_d(e)/C \rceil)$, yielding the OPT needs to traverse e at least $\max(1, \ell(e)/C)$ times. Therefore, we get a polynomial-time algorithm by performing two depth-first traversal steps as follows:

- Up-sweep: Starting from r, each item i is moved up from its source to its turning point. We perform the depth-first traversal by recursively ensuring that after the subtree rooted at e is explored, all $\ell_u(e)$ items that need to move up the edge e are collected at the lower-end point of e. We then traverse the edge e back and forth carrying at most C items at a time and moving all these items up the edge e. Since we can temporarily leave items at any node, we get an edge e is traversed $2\max(1, \lceil \ell_u(e)/C \rceil)$ times.
- Down-sweep: Each item i is moved down from its turning points to its destination, then ending at r. We perform the depth-first traversal by recursively ensuring that before the subtree rooted at e is explored, all $\ell_d(e)$ items that need to move down the edge e are collected at the upper-end point of e. We then traverse the edge e back and forth carrying at most C items at a time and moving all these items down the edge e. Since we can temporarily leave items at any node, we get an edge e is traversed $2\max(1, \lceil \ell_d(e)/C \rceil)$ times.

Using the up-sweep and the down-sweep results, the total number of traversals of edge e is at most $O(1) \cdot \max(1, \ell(e)/C)$ times. Hence, the tour produced by the algorithm is at most $O(1)$ times the length of an optimal tour OPT, i.e., we obtain an $O(1)$-approximation algorithm for this case.

(b) Convert metric (V, d) into an instance (V, T) on the weighted tree T s.t. r, sources and destinations of all items are at the leaves of the tree. We know that if (V', T') is a tree embedding for T which is a hierarchical cut decomposition, then we can find some other T s.t. $d_{T'}(u, v) \leq d_T(u, v) \leq 4d_{T'}(u, v)$ for all $(u, v) \in V$. Let OPT be the cost of the optimal solution to (V, d) and OPT' is the cost of the optimal solution to (V, T). By FRT, since $\mathbb{E}[d_{T'}(u, v)] \leq \mathcal{O}(\log n) d(u, v)$, so we have $\mathbb{E}[d_T(u, v)] \leq \mathcal{O}(\log n) d(u, v)$, hence, we obtain

$$\mathbb{E}[OPT'] \leq \mathcal{O}(\log n) \cdot OPT. \tag{6}$$

On the other hand, by FRT, we also know $d(u, v) \leq d_{T'}(u, v)$ for all $(u, v) \in V$, therefore, for all $(u, v) \in V$, we obtain

$$d(u, v) \leq d_{T'}(u, v) \leq d_T(u, v) \leq 4d_{T'}(u, v) \leq 4d_T(u, v). \tag{7}$$

Hence, a solution S' to (V, T) can be converted to a solution S of (V, d) s.t. $cost(S) \leq cost(S')$. Therefore, since we have a $O(1)$-approximation algorithm for (V, T) by (a), this gives us an $O(\log n)$-approximation for the general instances.