**CPSC 536N: Randomized Algorithms** 2011-12 Term 2

# Lecture 25

*Prof. Nick Harvey* *University of British Columbia*

The algorithm for probabilistically embedding metric spaces into trees has numerous theoretical applications. It is a key tool in the design of many approximation algorithms and online algorithms. Today we will illustrate the usefulness of these trees in designing an algorithm for the online Steiner tree problem.

## 1 Online Steiner Tree

Let $G = (V, E)$ be a graph and let $c : E \to \mathbb{R}_{>0}$ be lengths on the edges. Let $(V, d_G)$ be the shortest path metric on $G$.

For any $U \subseteq V$, a **Steiner tree** is a subtree of $G$ that spans (i.e., contains) all vertices in $U$, but does not necessarily span all of $V$. The vertices in $U$ are called "terminals". Equivalently, we can define a Steiner tree to be an acyclic, connected subgraph of $G$ that spans all of $U$. Computing a minimum-length Steiner tree is NP-hard.

Today we consider the problem of constructing a Steiner tree in an **online** setting. There is a sequence of $k$ time steps. In each time step $i$, we are given a vertex $u_i \in V$. Our algorithm must then choose a connected subgraph $C_i$ of $G$ which spans $\{u_1, \ldots, u_i\}$ (and possibly other vertices). The objective is to minimize the total length $c(\cup_{i=1}^k C_i)$. Since we only care about the cost of the *union* of the $C_i$'s, we may assume without loss of generality that $C_1 \subseteq \cdots \subseteq C_k$. There is no restriction on computation time of the algorithm.

**Remark.** The $C_i$'s are not actually Steiner trees because we did not insist that they are acyclic. If trees are desired, one could remove cycles from each $C_i$ arbitrarily. This is equivalent to the problem that we stated above.

If we knew the terminal set $U = \{u_1, \ldots, u_k\}$ in advance then the problem is trivial. The algorithm could compute in exponential time the minimum-length Steiner tree $C^*$ spanning $U$, then set $C_i = C^*$ in every time step. Unfortunately the algorithm does not know $U$ in advance. Instead, our goal will be for the algorithm to behave almost as well as if it did know $U$. Formally, define **competitive ratio** of the algorithm to be the ratio

$$\frac{c(\cup_{i=1}^k C_i)}{c(C^*)}.$$

We want our algorithm to have small competitive ratio.

**Theorem 1** *There is a randomized, polynomial time algorithm with expected competitive ratio $O(\log n)$.*

I think this is optimal but I could not find a definitive reference. For very similar problems, Alon & Azar prove a $\Omega(\log n / \log\log n)$ lower bound and Imase & Waxman prove a $\Omega(\log n)$ lower bound.

### 1.1 The Algorithm

The main idea is to use algorithm of the last lecture to approximate the metric $(V, d_G)$ by a tree $T$ with edge lengths $w$. Let $d_T$ be the corresponding distance function on $T$. Recall that the leaves of $T$ are identified with the vertices in $V$. The algorithm will then build a sequence of Steiner trees $T_1, \ldots, T_k$ that are subtrees of $T$, where each $T_i$ spans $\{u_1, \ldots, u_i\}$. This is trivial: since $T$ is itself a tree, there is really only one reasonable choice for what $T_i$ should be. We set $T_i$ to be the unique minimal subtree of $T$ that spans $\{u_1, \ldots, u_i\}$.

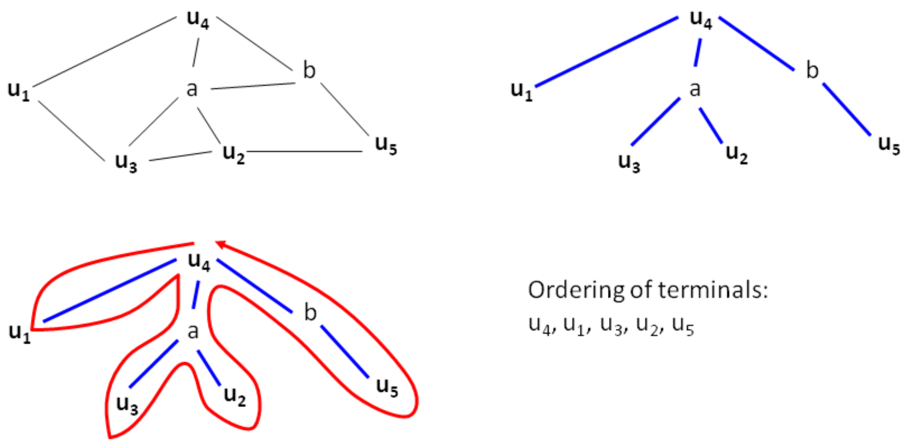
**Remark.** This step of the algorithm illustrates the usefulness of probabilistically approximating the metric by a tree. Many problems can be solved either trivially or by very simple algorithms, when the underlying graph is a tree.

Clearly $T_1 \subseteq \cdots \subseteq T_k$. We would like to understand how the length of our final Steiner tree $T_k$ compares to the optimal Steiner tree $C^*$.

**Claim 2** $\mathrm{E}[w(T_k)] \leq O(\log n) \cdot c(C^*)$

Unfortunately the tree $T_k$ itself isn't a solution to our problem. Recall that $T$ is not a subtree of $G$: the construction of $T$ required adding extra vertices and edges. So $T_k$ is not a subtree of $G$ either. To obtain our actual solution, we will see below how to use the trees $T_i$ to guide our construction of the desired subgraphs $C_i$ of $G$.

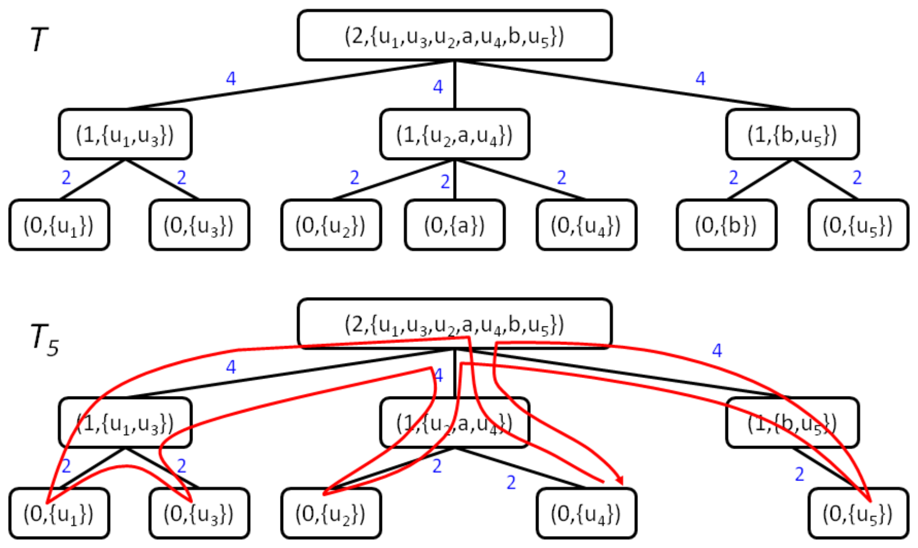
Proof:(of Claim 2). Let $\pi : [k] \to U$ be an ordering of the terminals given by a depth-first traversal of $C^*$. Equivalently, let $2 \cdot C^*$ denote the graph obtained from $C^*$ by replacing every edge with two parallel copies. Perform an Euler tour of $2 \cdot C^*$, and let $\pi$ be the order in which the terminals are visited.



The Euler tour traverses every edge of $2 \cdot C^*$ exactly once, so $c(C^*)$ is exactly half the length of the Euler tour. Thus

$$c(C^*) \;=\; (1/2) \cdot (\text{cost of Euler tour}) \;\geq\; (1/2) \cdot \sum_{i=1}^{k} d_G(\pi_i, \pi_{i+1}). \qquad (1)$$

Now consider performing a walk through $T_k$, visiting the terminals in the order given by $\pi$. Since this walk visits every leaf of $T_k$, it is a traversal of the tree, and hence it crosses every edge at least once. (In fact, it is an Eulerian walk, so it crosses every edge at least twice.)



Thus

$$w(T_k) \leq \sum_{i=1}^{k} d_T(\pi_i, \pi_{i+1}). \tag{2}$$

Combining (1) and (2), we get

$$\mathrm{E}[\,w(T_k)\,] \;\leq\; \mathrm{E}\Big[\,\sum_{i=1}^{k} d_T(\pi_i, \pi_{i+1})\,\Big] \;\leq\; O(\log n)\cdot \sum_{i=1}^{k} d_G(\pi_i, \pi_{i+1}) \;\leq\; O(\log n)\cdot c(C^*), \tag{3}$$

as required. □

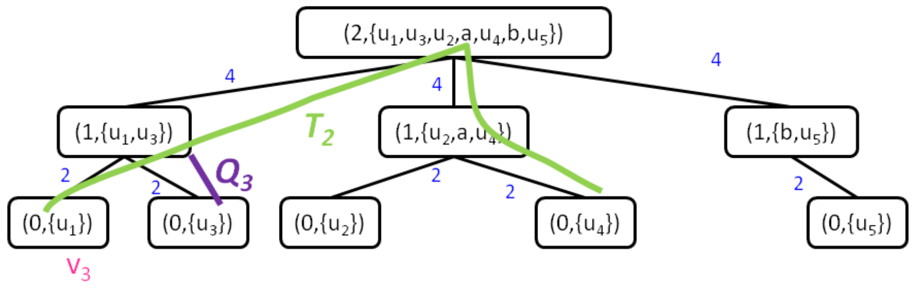
**Remark.** The analysis in (3) illustrates why our random tree $T$ is so useful. The quantity that we're trying to analyze (namely $w(T_k)$) is bounded by a *linear function* of some distances in the tree (namely $\sum_i d_T(\pi_i, \pi_{i+1})$). Because we can bound $\mathrm{E}[d_T(\pi_i, \pi_{i+1})]$ by $O(\log n)\cdot d_G(\pi_i, \pi_{i+1})$, and because of linearity of expectation, we obtain a bound on $\mathrm{E}[w(T_k)]$ involving distances in $G$.

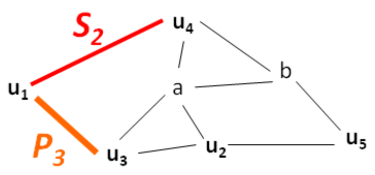
Now let us explain how the algorithm actually solves the online Steiner tree problem. It will maintain a sequence $C_1 \subseteq \cdots \subseteq C_k$ of subgraphs of $G$ such that each $C_i$ spans $\{u_1, \ldots, u_i\}$. Initially $C_1 = \{u_1\}$. Then at every subsequent time step $i$, we do the following.

- Let $v_i$ be the closest vertex to $u_i$ amongst $\{u_1, \ldots, u_{i-1}\}$, using distances in $T$. In other words, $v_i = \mathrm{argmin}_{v \in \{u_1, \ldots, u_{i-1}\}} d_T(u_i, v)$.
- Let $P_i$ be a shortest path in $G$ from $u_i$ to $v_i$.
- Let $C_i = C_{i-1} \cup P_i$.

The trees $T_1, \ldots, T_k$ described above can also be viewed in this iterative fashion. Initially $T_1 = \{u_1\}$. Then at every subsequent time step $i$, we do the following.

- Let $Q_i$ be the unique path in $T$ from $u_i$ to the closest vertex in $T_{i-1}$.
- Let $T_i = T_{i-1} \cup Q_i$.





The following important claim relates $P_i$ and $Q_i$.

**Claim 3** *$c(P_i) \leq 2 \cdot w(Q_i)$ for all $i$.*

Proof: Let $z_i$ be the closest vertex in $T_{i-1}$ to $u_i$, so $Q_i$ is a $u_i$-$z_i$ path. The vertex $z_i$ would only be added to $T_{i-1}$ if some leaf $v$ beneath $z_i$ belongs to $\{u_1, \ldots, u_{i-1}\}$. By the choice of weights in $T_{i-1}$, the weight of the $z_i$-$v$ path is no longer than the weight of the $u_i$-$z_i$ path. Thus $d_T(u_i, v_i) \leq 2 \cdot w(Q_i)$ for all $i$. Consequently

$$c(P_i) \;=\; d_G(u_i, v_i) \;\leq\; d_T(u_i, v_i) \;\leq\; 2 \cdot w(Q_i),$$

as required. $\square$

Consequently,

$$c(C_k) \;\leq\; \sum_{i=1}^{k} c(P_i) \;\leq\; 2\sum_{i=1}^{k} w(Q_i) \;=\; 2 \cdot w(T_k),$$

since $C_k$ is the union of the $P_i$ paths and $T_k$ is the *disjoint* union of the $Q_i$ paths. So

$$\mathrm{E}[c(C_k)] \;\leq\; O(\log n) \cdot c(C^*),$$

by Claim 2. This proves that the expected competitive ratio is $O(\log n)$.