601.435/635 Approximation Algorithms **Lecturer:** Michael Dinitz
Topic: GST and Tree Embeddings **Date:** 3/14/19
Scribe: Michael Dinitz

14.1 Introduction

Last class we saw an $O(\log n \log k)$-approximation for Group Steiner Tree as long as the input is a tree. This was a highly nontrivial algorithm and analysis – how can we possibly hope to extend it all the way to general graphs? We're going to do this by using a technique called *metric embeddings*: we're going to *embed* general metric spaces into trees.

14.2 Definitions

Recall the definition of a metric space

Metric: A pair (V, d) such that $\forall\, u, v, w \in V$

1. $d(u, v) = 0 \iff u = v$
2. $d(u, v) = d(v, u)$
3. $d(u, v) \leq d(u, w) + d(w, v)$

Note that it is common to simply refer to the metric as d instead of the pair (V, d). We're going to be concerned with a special type of metric known as a *tree metric*.

Definition 14.2.1 *A* tree metric (V', T) *for a set of nodes V is a tree T on vertices V', where $V \subseteq V'$ are the leaves of T. Every edge of T has a nonnegative length.*

The distance in T between any two vertices $u, v \in V'$ is denoted $d_T(u, v)$, where the distance in T is the length of the unique $u - v$ path in T.

Definition 14.2.2 *Let (V, d) be a metric and (V', T) a tree metric for V. Then (V, d) embeds into T with distortion α if $d(u, v) \leq d_T(u, v) \leq \alpha \cdot d(u, v)$ for all $u, v \in V$.*

Intuitively, if we can embed (V, d) into some tree (V', T) with small distortion, then T is "like" the original metric space so we might hope that we can just solve any problem that we care about it on T instead of on the original metric. Unfortunately, this is not always possible: even simple metric spaces like the cycle C_n might require large distortion to embed into any tree. This is trivial to see if we required T to be a subtree of the input graph, but since we're not requiring that, this is a bit harder to prove. It is possible to show that C_n requires distortion at least $\frac{n-1}{8}$ to embed into any tree.

What can we do? Let's take inspiration from the cycle: there's no tree which allows small distortion, but if we fix some pair $u, v \in V$, then a *random* subtree of C_n is pretty good in expectation! For example, if u and v are adjacent in C_n, then with probability $1/n$ they get distance n, while with

probability $\frac{n-1}{n}$ they're still at distance 1. So the expected distance is at most 2. So for any pair of nodes the expected distortion is small, even though once we instantiate some particular tree, there *will be* some pair which is badly distorted. As it turns out, though this kind of expected distortion is enough for many applications.

The best and provably optimal result for doing this is due to Fakcharoenphol, Rao, and Talwar, who proved the following theorem.

Theorem 14.2.3 *Let (V, d) be a metric. Then there is a randomized, polytime algorithm that produces a tree metric (V', T) for V such that*

1. *$d(u, v) \leq d_T(u, v)$ for all $u, v \in V$, and*
2. *$\mathbf{E}[d_T(u, v)] \leq O(\log n) \cdot d(u, v)$ for all $u, v \in V$.*

In other words, this theorem gives an embedding into a *distribution* of *dominating* trees (a distribution of trees each of which does not contract any pair). This algorithm is tight: there are metrics for which any embedding into a distribution of dominating trees requires distortion $\Omega(\log n)$.

We're going to spend the next couple of classes proving this theorem and analyzing tree embeddings, but before we do that, let's show why they're useful. It's not hard to see that almost any problem which involves distances can be turned into a problem on trees by using this theorem and losing an extra $O(\log n)$ in the approximation ratio, but let's see this for a particular problem: Group Steiner Tree.

14.3 Group Steiner Tree on General Metrics

Recall the GST problem:

- **Input**:, A graph $G = (V, E)$, edge costs $c : E \to \mathbb{R}_{\geq 0}$, a root vertex $r \in V$, and groups $g_1, \ldots, g_k \subseteq V$.
- **Feasible solution**: A tree T such that for all $i \in [k]$, there is some $v \in g_i$ such that T has a path between r and v.
- **Objective**: $\min \sum_{e \in T} c(e)$

We saw last class that Garg, Konjevod, and Ravi (GKR) gave an $O(\log n \log k)$-approximation when the input graph is a tree, and that the problem is $\Omega(\log^{2-\epsilon} n)$-hard to approximate even on trees. How can we design an approximation algorithm for general metrics? Use FRT to change the input into a tree!

Slightly more formally, consider the following algorithm:

1. Extend c to a metric space (V, d) where $d(u, v)$ is the minimum cost of any $u - v$ path.
2. Use FRT (Theorem 14.2.3) to embed (V, d) into a tree (V', T) with distortion $O(\log n)$. Note that since V are the leaves of T, all of the terminals (vertices in the groups) are now leaves.

3. Make a new group which is just $\{r\}$, and then use the GKR algorithm to get a subtree T' of T which is an $O(\log |V'| \log k)$-approximation to the optimal solution on T.

4. Shortcut T' to get a cycle C only on terminals.

5. Use C (with one arbitrary edge removed) as our solution in the metric space (V, d). To get a solution on G, replacing any edge of C which doesn't exist in G with a path of the same length.

This algorithm clearly gives a feasible solution: GKR returns a tree which connects at least one terminal from each group (including r) to the root of T, so C has r and at least one terminal from each group. Thus C gives a feasible solution. The algorithm also clearly takes only polynomial time. So we just need to analyze the approximation ratio.

Theorem 14.3.1 *This algorithm is a $O(\log^2 n \log k)$-approximation, i.e.,*

$$\mathbf{E}[c(C)] \leq O(\log^2 n \log k) \cdot OPT.$$

Proof: Let's set up some notation.

- Let S be the terminals connected by OPT (so $S \cap g_i \neq \emptyset$ for all $i \in [k]$)
- Let C_S be the cycle on S obtained by shortcutting OPT (so $c(C_S) \leq 2 \cdot OPT$).
- T will be the (random) tree built by FRT, with costs c_T or d_T.
- Let $OPT(T)$ denote the optimal solution in T.
- Let T_S be the subtree of T induced by S (i.e., the subtree of T which consists of all the paths from nodes in S up to the LCA of S.

Now we can actually prove the theorem.

$$\begin{aligned}
\mathbf{E}[c(C)] &\leq \mathbf{E}[c_T(C)] && \text{distances in } T \text{ are nondecreasing} \\
&\leq \mathbf{E}\big[2 \cdot c_T(T')\big] && \text{shortcutting costs at most a factor of 2} \\
&= 2 \cdot \mathbf{E}\big[c_T(T')\big] && \text{linearity of expectation} \\
&\leq 2\mathbf{E}[O(\log n \log k) \cdot c_T(OPT(T))] && \text{GKR} \\
&= O(\log n \log k) \cdot \mathbf{E}[c_T(OPT(T))] && \text{linearity of expectations} \\
&\leq O(\log n \log k) \cdot \mathbf{E}[c_T(T_S)] && \text{by definition of } OPT(T) \\
&\leq O(\log n \log k) \cdot \mathbf{E}[c_T(C_S)] && \text{shortcutting: each edge of } T_S \text{ counted at most twice} \\
&= O(\log n \log k) \cdot \mathbf{E}\left[\sum_{(u,v) \in C_S} d_T(u,v)\right] && \text{by definition} \\
&= O(\log n \log k) \cdot \sum_{(u,v) \in C_S} \mathbf{E}[d_T(u,v)] && \text{linearity of expectations}
\end{aligned}$$

$$
\begin{aligned}
&\leq O(\log n \log k) \cdot \sum_{(u,v)\in C_S} (O(\log n) \cdot d(u,v)) && \text{FRT} \\
&= O(\log^2 n \log k) \cdot \sum_{(u,v)\in C_S} d(u,v) && \text{linearity of expectations} \\
&\leq O(\log^2 n \log k) \cdot 2 \cdot OPT && \text{shortcutting} \\
&= O(\log^2 n \cdot \log k) \cdot OPT && \text{asymptotic notation}
\end{aligned}
$$

■

So combining FRT with GKR gives an $O(\log^2 n \log k)$-approximation to GST in general! This is still the state of the art. The question of whether this extra $\log n$ loss can be avoided is still an extremely important open question in approximation algorithms.

14.4 Metric Embeddings in General

We're not going to talk too much about general metric embeddings, but our approach for GST can be generalized to many other problems and other metrics. Let's see this a bit abstractly.

Definition 14.4.1 *(V, d) embeds into (V, d') with distortion α is $d(u,v) \leq d'(u,v) \leq \alpha d(u,v)$ for all $u, v \in V$.*

There are equivalent definitions based on contraction rather than expansion or on both, which are slightly more natural in some contexts, but this definition is more intuitive based on what we've been doing.

Now suppose that we have a β-approximation for some problem in d', but not in d. Then consider the algorithm which first embeds d into d' with distortion α, and then uses the β-approximation for d'. If the problem that we care about has costs which are just sums of distances (like many of the problems we've been thinking about), then we get that

$$
\begin{aligned}
c(ALG) = \sum_{\{u,v\}\in ALG} d(u,v) &\leq \sum_{\{u,v\}\in ALG} d'(u,v) \leq \beta \sum_{\{u,v\}\in OPT(d')} d'(u,v) \leq \beta \sum_{\{u,v\}\in OPT} d'(u,v) \\
&\leq \beta\alpha \sum_{\{u,v\}\in OPT} d(u,v) = \beta\alpha \cdot c(OPT)
\end{aligned}
$$

Handling probabilistic embeddings, like we did with FRT for GST, just involves putting expectations in the right places, but it all works out the same. So as long as our problem is "about" distances, we can use metric embeddings to transform the input metric into a "simpler" metric (like a tree) by paying the distortion in the approximation ratio.

14.5 Hierarchical Decompositions and FRT Algorithm

Now we're going to spend the rest of today, and much of the next class, trying to prove Theorem 14.2.3. The first key idea, which is what FRT will actually construct, is (the tree corresponding to) a *hierarchical cut decomposition.*

14.5.1 Hierarchical Cut Decompositions

Without loss of generality, we may assume (by scaling) that $\min_{u,v \in V: u \neq v} d(u,v) = 1$. For any set $S \subseteq V$, let $\text{diam}(S) = \max_{u,v \in S} d(u,v)$ denote its diameter. Let $\Delta = 2^{\lceil \log \text{diam}(V) \rceil}$ be the smallest power of 2 such that $\Delta \geq \text{diam}(V)$.

Hierarchical Cut Decomposition: A (rooted) tree metric (V', T) for (V, d) so that

1. Every vertex $\ell \in T$ is associated with some subset $S_\ell \subseteq V$ (note that by definition of tree metric the subset associated with a leaf vertex is a node in V).
2. The root r of T has $S_r = V$.
3. If u at level i of T, then $\text{diam}(S_u) < 2^i$ (leaves at level 0, root at level $\log \Delta$).
4. If u has children $w_1, \ldots w_k$, then $\{S_{w_i}\}_{i \in [k]}$ partition S_u (i.e., $S_u = \cup_{i=1}^k S_{w_i}$ and $S_{w_i} \cap S_{w_j} = \emptyset$ for all $i \neq j$).
5. Length of an edge between a level i node and a level $i+1$ node is 2^{i+1}.

The FRT algorithm will construct a hierarchical cut decomposition, but before we give the algorithm, let's start by showing a simple lemma which holds for any hierarchical cut decomposition. Consider a hierarchical cut decomposition (V', T) of some metric (V, d).

Lemma 14.5.1 *If the least common ancestor of two leaf nodes u and v in T is at level i, then $d_T(u,v) \leq 2^{i+2}$. Furthermore, $d_T(u,v) \geq d(u,v)$ for all $u, v \in V$*

Proof: Let u and v be leaf nodes in T, and let w be u and v's least common ancestor (so w is at level i). Then by construction we know that $d_T(u,w) = \sum_{j=1}^{i} 2^j$, so $2^i \leq d_T(u,w) < 2^{i+1}$. Similarly, $2^i \leq d_T(v,w) < 2^{i+1}$. Since $d_T(u,v) = d_T(u,w) + d_T(w,v)$, we get that $2^{i+1} \leq d_T(u,v) < 2^{i+2}$. That proves the first part of the lemma. And because u, v are both contained in S_w, we know that $d(u,v) \leq \text{diam}(S_w) \leq 2^i$, which implies the second part. ■

14.6 Constructing the FRT tree

We can now finally give the FRT algorithm for constructing a tree embedding. FRT constructs a hierarchical decomposition in a certain way, but since it does construct a hierarchical decomposition, we know that no pair is contracted, and the distance between two nodes in the tree depends only on the level of their LCA. This is going to make reasoning about distances in the tree much easier.

(As a side note, you might have noticed that these trees are not just trees, they're special trees where the distance between two nodes grows exponentially with the level of their LCA. So we're actually doing more than just giving a tree embedding: we're giving a tree embedding into a special class of trees known as Hierarchically Well-Separated Trees (HSTs). Occasionally it is useful to utilize this property algorithmically: for GST we didn't care whether we were in a HST or a general tree, but for other problems it is sometimes easier to handle HSTs than general trees, and thanks to FRT we only need to handle HSTs).

We probably won't have time today to analyze this algorithm, so let's defer it until after spring break.

Algorithm 1 FRT embedding

Let π be a permutation of V, chosen uniformly at random
Let r_0 be a value in $\left[\frac{1}{2}, 1\right)$, chosen uniformly at random
Let $r_i = r_0 \cdot 2^i$ for all i such that $1 \leq i \leq \log \Delta$
Let T be a tree with only a root node (at level $\log \Delta$) which represents V
for $i \leftarrow \log \Delta$ to 1 **do**
  Let $\mathcal{C}$ be the set of nodes at level i
  for $C \in \mathcal{C}$ **do**
    $S \leftarrow C$
    for $j \leftarrow 1$ to n **do**
      $P \leftarrow$ the nodes in S enclosed by a circle centered on $\pi(j)$ with radius r_{i-1}
      if $P \neq \emptyset$ **then**
        $S \leftarrow S \setminus P$
        Add P to T as a child of C at level $i - 1$
      end if
    end for
  end for
end for

return T