CPSC 665 : An Algorithmist's Toolkit Lecture 1 : 30 Mar 2015

Presentation: Tree Metrics

Lecturer: Sushant Sachdeva *Scribe: Alex Reinking*

1. Introduction

The basic idea we want to talk about is that of **tree metrics**, which give estimations of distance functions on regular graphs. These are desirable since the related metrics are defined on trees, which are often far easier to work with than general graphs, and it turns out that using these approximate metrics only introduces a small ($O(\log n)$) loss in accuracy.

2. Definitions

We start by laying out some basic terminology. Tree metrics are best defined in terms of graph metrics, so we'll state these formally.

Definition 2.1 (Distance Metric)**.** *A metric (V, d) on a set of vertices V gives a distance d_{uv} to every pair of vertices $u, v \in V$ such that*

(1) $d_{uv} = 0$ iff $u = v$.
(2) $d_{uv} = d_{vu}$ $\forall$ $u, v \in V$
(3) $d_{uv} \leq d_{uw} + d_{wv}$ $\forall$ $u, v, w \in V$ (Triangle inequality)

This is a very standard definition for this kind of thing, but it does guarantee us the triangle inequality, which we'll use later. So now let's define tree metrics in terms of this

Definition 2.2 (Tree Metric)**.** *A tree metric (V', T) is a tree T defined on vertices $V' \supseteq V$ together with the non-negative lengths on each edge of T. The distance T_{uv} between vertices $u, v \in V'$ is the length of the unique shortest path between $u, v \in T$. We hold that T_{uv} must obey the usual graph metric laws above.*

It is important to note that T does not at all need to be a spanning tree of V and that V' can be much, much, larger. The important property that we care about is that T_{uv} doesn't go too far away from d_{uv} when $u, v \in V$. We capture this intuition with the following definition:

Definition 2.3 (Distortion)**.** *We call the distortion of the tree metric (V', T) the smallest value of α such that*

$$d_{uv} \leq T_{uv} \leq \alpha T_{uv} \tag{1}$$

for all $u, v \in V$.

Given a low-distortion embedding, we can often reduce problems on general metric spaces to problems on tree metrics with a loss of just α in accuracy. This is cool because, as we will see, it is much easier to solve problems on a tree than it is to solve them on a graph.

A natural question to ask at this point is: are there any graphs for which no good approximate tree exists?

Example 2.4 (High Distortion Graph)**.** *It happens to be that in general, C_n, the cycle on n vertices has a distortion of at least $\frac{n-1}{8}$. Consider, for example, the figure of C_7 below:*

Now $d_{ab} = 1 \leq T_{ab} = 6 \leq 6d_{ab}$, so the distortion is at least $6 = n - 1$. Then in general, if you just drop an edge from the n-cycle to get a tree, the distortion will be at least $n - 1$. You could only do better, then, by adding some vertices.

So the question becomes: how do we sidestep this problem? If we can't always find low-distortion embeddings, what can we do? Well, we can give up and do a different problem instead. Rather than deterministically finding low-distortion embeddings, we'll just find low-distortion embeddings *in expectation*.

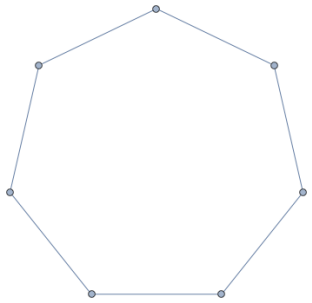

FIGURE 1. C_7 is poorly-approximated by a tree metric

3. Sampling from a Low-Distortion Distribution

The primary theorem we are going to prove comes from Fakcharoenphol, Rao, and Telwar.

Theorem 3.1 (FRT '04)**.** *Given a distance metric (V, d) such that $d_{uv} \geq 1$, $\forall u \neq v \in V$, there is a randomized polynomial-time algorithm that produces a tree metric (V', T), $V' \supseteq V$ such that $\forall u, v \in V$, $d_{uv} \leq T_{uv}$ and $\mathbf{E}[T_{uv}] \leq O(\log n) d_{uv}$.*

It is known that there exist metrics such that any probabilistic tree metric approximation has distortion $\Omega(\log n)$, so this result is actually the best possible up to constant factors.

The assumption that $d_{uv} \geq 1$ is important for the analysis, but not an issue in practice, since for most problems, the graph can simply be scaled.

4. An Application to Balanced Separator

We won't go too deep into this (ie. I won't prove anything), but since tree metrics have so many applications (network design, route planning, etc.), I'd like to showcase one example here.

Recall the SDP relaxation for Balanced Separator

$$SDP \begin{cases} \min & \sum_{(i,j)\in E} \left\|v_i - v_j\right\|^2 \\ \text{subject to} & \sum_{i,j\in V} \left\|v_i - v_j\right\|^2 \geq c(1-c)n^2 \\ \text{and} & \left\|v_i - v_j\right\|^2 \leq \|v_i - v_k\|^2 + \left\|v_k - v_j\right\|^2 \end{cases} \tag{2}$$

We proved that solving this SDP and cutting the resulting vectors with a hyperplane gave good results. But we can do this somewhat faster by solving a linear program to produce a graph metric. The linear program is produced by analogy:

$$LP \begin{cases} \text{vars} & d_{ij} \\ \min & \sum_{(i,j)\in E} d_{ij} \\ \text{subject to} & \sum_{i,j\in V} d_{ij} \geq c(1-c)n^2 \\ \text{and} & d_{ij} \leq d_{ik} + d_{kj} \\ & d_{ij} \geq 0, d_{ij} = d_{ji} \forall i, j \end{cases} \tag{3}$$

Now applying the FRT theorem to get a low-distortion tree, we can solve Balanced Separator on the tree. This is far easier than on the graph, since we can check every cut of V' in $O(E)$ time by just looking at the cut gained by removing that edge. This will induce a cut on V by intersections with only an additional $O(\log n)$ factor added to the approximation.

5. Hierarchical Cut Decomposition

The idea here is that we're going to produce a tree whose vertices correspond to nearby sets of vertices in the original graph. Each node in the tree will be partitioned by its children; the top level (the root) of the tree will correspond to all of V, and the bottom level (the leaves) will correspond to individual vertices. By partitioning it in this way, we can be sure we don't stretch the distances too much. Below is a figure to help visualize the situation.

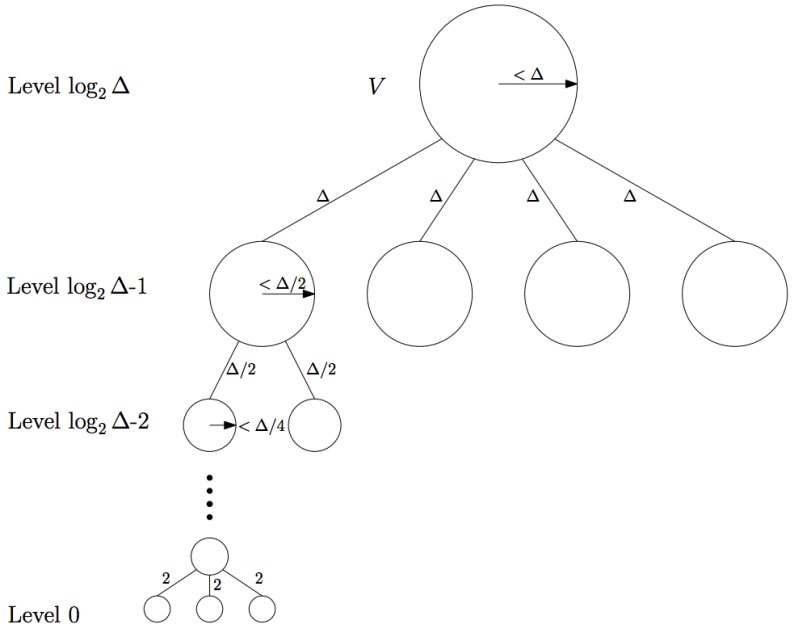

Figure 2. A hierarchical cut decomposition of the metric space V. Picture from Design of Approximation Algorithms, Section 8.5 by David P. Williamson and David B. Shmoys

We will prove that such trees form a family that, when sampled, imply the FRT theorem. Let us briefly state a few guarantees provided by our construction.

The resulting tree will contain $\log_2 \Delta + 1$ levels, where Δ is the smallest power of two above twice the diameter of the graph, ie.

$$\Delta = \min_n \left\{ 2^n \geq 2 \max_{u,v} d_{uv} \right\}$$

Furthermore, if S is the set of vertices in V corresponding to some node at level i of T, we will have that S is contained in a ball of radius $[2^{i-1}, 2^i)$ centered on some vertex in V. For the leaves at level 0, notice that they surely contain only one vertex since they are contained in a ball of size $[2^{-1}, 2^0) = [\frac{1}{2}, 1)$, and we assumed that $d_{uv} \geq 1$ whenever $u \neq v$. Also, notice that by our definition of Δ, the whole graph is contained in the single vertex at level $\log_2 \Delta$. This is because it's in a ball of radius at most Δ, which is no smaller than twice the diameter of the original graph.

But before we actually see *how* to construct these trees, let's try to see that they're actually what we want by proving the first part of FRT, and deriving a bound for the tree metric.

Lemma 5.1. *Any tree T obtained by a Hierarchical Cut Decomposition of a metric (V, d) has $T_{uv} \geq d_{uv}$. Further, if the least common ancestor of the vertices in T corresponding to the vertices $u, v \in V$ is at level i, then $T_{uv} \leq 2^{i+2}$*

Proof. Let $S \in T$ be some node at level i. If $u, v \in S$, then $d_{uv} < 2^{i+1}$ since the radius of S is at most 2^i (so if u, v are a diameter then the distance between them could be up to 2^{i+1}). Thus, these two vertices cannot belong to a node at level $\lfloor \log_2 d_{uv} \rfloor - 1$ since otherwise the distance between them would be bounded by $2^{\lfloor \log_2 d_{uv} \rfloor} < d_{uv}$, which is a clear contradiction. Thus, the lowest level where u and v meet is $\lfloor \log_2 d_{uv} \rfloor$, and so

$$T_{uv} \geq 2 \sum_{j=1}^{\lfloor \log_2 d_{uv} \rfloor} 2^j \geq d_{uv}$$

This can be clearly seen referring to the picture – the factor of 2 comes from moving up and back down to the least common ancestor, and each path is an accumulation of powers of two. Finally, this observation also shows that if u and v have their least common ancestor at level i, then

$$T_{uv} = 2 \sum_{j=1}^{i} 2^j = 2^{i+2} - 4 \leq 2^{i+2}$$

□

5.1. **Algorithm.** So now we can actually proceed to describe the algorithm itself. We begin by drawing on two sources of randomness that will be fix a few values at the start of the algorithm. First, we will shuffle the vertices of V according to some uniformly random permutation π. We will then pick a random radius for the bottom level $r_0 \in [\frac{1}{2}, 1)$, and then multiplying the chosen value by two for each successive level so $r_i = 2^i r_0$.

This might seem a little mysterious at first. We choose r_0 specifically to take advantage of the assumption that $d_{uv} \geq 1$ to ensure the bottom layer consists of only single-vertex sets. This explains the open bound of 1. But we choose the lower bound $1/2$ since it is the lowest we can choose such that level 1 will not share that same guarantee. If the radius r_0 were chosen to be $1/2 - \epsilon/2$, then $r_1 = 1 - \epsilon$ and so both levels 0 and 1 will necessarily be identical. Granted, this might still be the case regardless, but at least we don't have that awkward behavior.

So, in order to produce the tree metric, all that's left is to describe the process of breaking a given node down into the next level of partitions. Given a set S at level i, we assign each vertex $u \in S$ to the **first** vertex $\pi(j)$ so that $u \in B(\pi(j), r_{i-1})$. All of the vertices assigned to the same $\pi(j)$ are grouped together, and the intersections of these groups with S become S's children.

Notice that this implies that S need not be centered around a node it actually contains! This, of course, holds for all its children too.

To make this much clearer, I will borrow the example from the Design of Approximation Algorithms, Section 8.5 by David P. Williamson and David B. Shmoys to help explain this.

The algorithm operates, then, by starting at the top vertex, the one corresponding to V, breaking it down into its child nodes, and recursively applying that same procedure for $\log_2 \Delta$ levels.

5.2. **Proof.** Now it's time to actually prove the FRT theorem. Take a moment to glance back up at it if you've forgotten the statement.

Proof. The first part of FRT was already shown by the lemma, so we just need to place an upper bound on the expected value of the tree metric. By the lemma, we can use our bound for T_{uv} if we assume that the least common ancestor of u, v is on level $i+1$ (we know it won't be at level 0, so we don't lose anything by adopting this convention). Under these circumstances, we have $T_{uv} \leq 2^{i+3}$. For this to actually happen, though, u and v must be in separate balls at level i. By construction, then, there is some w so that $B(w, r_i)$ contains exactly one of u or v.

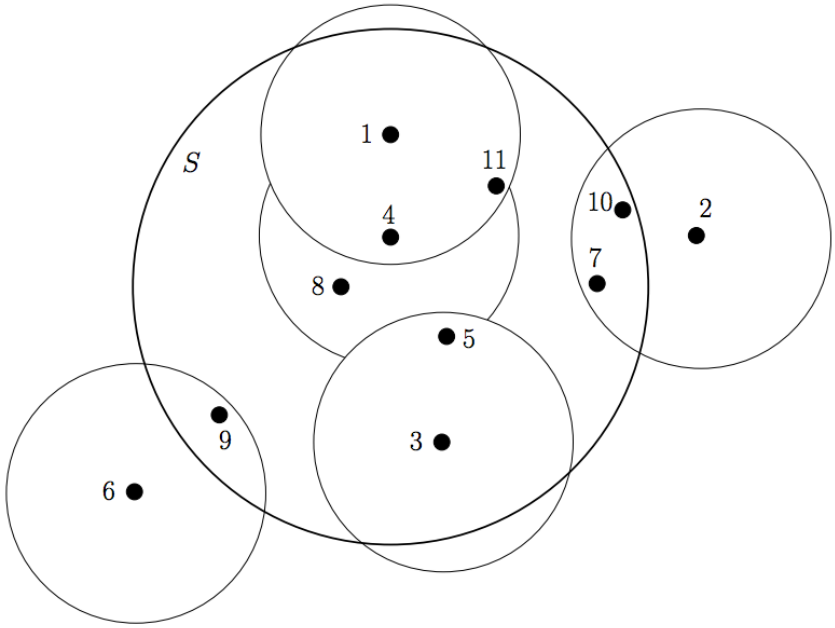

FIGURE 3. This example shows the set $S = \{1, 3, 4, 5, 7, 8, 9, 10, 11\}$ being partitioned at one level of the Hierarchical Cut Decomposition. We take the random partition as the identity (ie. the vertices are already permuted) and start from vertex 1. The ball centered at 1 captures $1, 4, 11$ so $\{1, 4, 11\}$ becomes a child node of S. Similarly, the ball around 2 contains $2, 7, 10$ but since 2 is not contained in S, $\{7, 10\}$ becomes a child node. The ball centered at 3 gives rise to $\{3, 5\}$ The ball centered at 4 contains 8, but since it already affiliated to a subset, only $\{8\}$ forms a child node. The ball around 5 has no effect, and then the ball around 6 contains 9, making the final child node $\{9\}$. So in total, the children of S are $\{1, 4, 11\}$, $\{7, 10\}$, $\{3, 5\}$, $\{8\}$, $\{9\}$.

For the sake of clarity, we will say that w **settles** u and v on level i if the ball $B(w, r_i)$ is the first in π to include either u or v, ie. this particular vertex settles the question of whether they are together after that point.

By contrast, we will say that w **cuts** u and v on level i is exactly one of u or v is in the ball $B(w, r_i)$.

Now, we will define events to analyze this probabilistically. For any pair $u, v \in V$:

$$X_{iw} = w \text{ cuts } (u, v) \text{ at level } i$$
$$S_{iw} = w \text{ settles } (u, v) \text{ at level } i$$

Then we have a kind of trivial bound of T_{uv}:

$$T_{uv} \leq \max_i \mathbb{1}[\exists w \in V : X_{iw} \text{ and } S_{iw}] \cdot 2^{i+3} \tag{4}$$

We can make this a bit more manageable with a sum since these two events can only happen for the first time once.

$$T_{uv} \leq \sum_{w \in V} \sum_{i=0}^{\log_2 \Delta - 1} \mathbb{1}[X_{iw} \text{ and } S_{iw}] \cdot 2^{i+3} \tag{5}$$

So in expectation:

$$\mathbf{E}[T_{uv}] \leq \sum_{w \in V} \sum_{i=0}^{\log_2 \Delta - 1} \mathbf{Pr}[X_{iw} \text{ and } S_{iw}] \cdot 2^{i+3} \tag{6}$$

So let's take a look at where we're going before we jump into some technical details. We will shortly prove the following lemmas, but just state them for now.

Lemma 5.2. $\mathbf{Pr}[S_{iw}|X_{iw}] \leq b_w$ *and* $\sum_{w \in V} b_w = O(\log n)$.

Lemma 5.3. $\sum_{i=1}^{\log_2 \Delta} \mathbf{Pr}[X_{iw}] \cdot 2^{i+3} \leq 16 d_{uv}$.

So with these two lemmas in place, we can finish up:

$$\mathbf{E}[T_{uv}] \leq \sum_{w \in V} \sum_{i=0}^{\log_2 \Delta - 1} \mathbf{Pr}[S_{iw}|X_{iw}] \mathbf{Pr}[X_{iw}] \cdot 2^{i+3}$$

by the second lemma,

$$\leq \sum_{w \in V} b_w \left[\sum_{i=0}^{\log_2 \Delta - 1} \cdot 2^{i+3} \right]$$

by the first lemma,

$$\begin{aligned} &\leq \sum_{w \in V} b_w \left[16 d_{uv}\right] \\ &= 16 d_{uv} \sum_{w \in V} b_w \end{aligned}$$

by the second lemma,

$$\begin{aligned} &\leq 16 \; d_{uv} \; O(\log n) \\ &\leq O(\log n) d_{uv} \end{aligned}$$

Lemma 5.2. Suppose WLOG $d_{uw} \leq d_{vw}$. then the probability that w cuts (u, v) on level i is the probability that $u \in B(w, r_i)$ and $v \notin B(w, r_i)$. Equivalently, this is $d_{uw} \leq r_i \leq d_{vw}$. Since $r_i \in [2^{i-1}, 2^i)$ uniformly, this is just:

$$\mathbf{Pr}[X_{iw}] = \frac{\left|[2^{i-1}, 2^i) \cap [d_{uw}, d_{wv})\right|}{\left|[2^{i-1}, 2^i)\right|} = \frac{1}{2^{i-1}} \left|[2^{i-1}, 2^i) \cap [d_{uw}, d_{wv})\right|$$

Scaling by 2^{i+3}:

$$\begin{aligned} &= \frac{2^{i+3}}{2^{i-1}} \left|[2^{i-1}, 2^i) \cap [d_{uw}, d_{wv})\right| \\ &= 16 \left|[2^{i-1}, 2^i) \cap [d_{uw}, d_{wv})\right| \end{aligned}$$

But since the power-of-two intervals actually partition the whole interval, we actually have

$$\sum_{i=0}^{\log_2 \Delta - 1} 2^{i+3} \cdot \mathbf{Pr}[X_{iw}] \leq 16 \left|[d_{uw}, d_{wv})\right| = 16(d_{vw} - d_{uw}) \leq 16 d_{uv}$$

with the last inequality following from the triangle inequality. □

Lemma 5.3. We need to bound $\mathbf{Pr}[S_{iw}|X_{iw}]$. So we order the vertices $w \in V$ by their distance to either u or v. That is, we order by $\min\{d_{uw}, d_{vw}\}$. Note that if X_{iw} occurs then one of u or v is in $B(w, r_i)$. Thus, any z closer to the pair than w will also have one in its ball. So in order for w to be the first cutting vertex (and therefore also the settling vertex), it must come before any other such z. This is a standard exercise in probability (look up example 10.3.4 in Invitation to Discrete Mathematics).

Anyway, we let $b_w = (\text{ position of } w \text{ in } \pi)^{-1}$. If the position is j, then $b_w = 1/k$. In sum, $\sum_{w \in V} v_w = \sum j = 1^n 1/j = H_n = O(\log n)$.

And that completes the proof! □

□