Advanced Algorithm Design: Tree Embedding (cont.)

Lectured by Prof. Moses Charikar
Transcribed by Ohad Fried*

Apr 24th, 2013

1 Recap

In the previous lectures we discussed tree embeddings. We saw that embedding onto random trees can be useful, providing approximation algorithms to many problems. Random tree embedding was introduced by Bartal (1996), and we mentioned that the bound was improved by Fakcharoenpol, Rao and Talwar (2003), based on a construction by Calinescu, Karloff and Rabani (2001). In this lecture we will analyze this improved construction. Also, we will introduce the concept of a minimum communication spanning tree.

2 A New Partitioning Procedure

Let us pick $R \in [\frac{\delta}{4}, \frac{\delta}{2}]$ uniformly and let σ be a random permutation of $[n]$ (n being the amount of points we wish to embed). Let $B(p, r)$ denote a ball of radius r around p (more precisely: $B(p, r) = \{x \mid d(x, p) \leq r\}$). We can define a ball of radius R around each point (after undergoing the random permutation) as $B_i = B(\sigma(i), R)$. Our partition will be:

$$\hat{B_i} = B_i \setminus (\bigcup_{j<i} B_j) \tag{2.1}$$

which is a fancy way of saying we partition according to the balls B_i, forcing that if a point is selected at some stage to be inside a set, it will not be selected for the succeeding sets.

Given a pair of points u and v, we ask ourselves what is the probability that they are separated by the above procedure. We notice that if a ball misses the pair completely, we do not care since it will not change the outcome. If a ball contains both u and v, it "protects" the points from separation on this

*ohad@cs.princeton.edu

level (keeping in mind that we will perform this procedure multiple times in a hierarchical fashion). If a ball contains only one point out of the pair, the pair will be separated.

Just for the sake of analysis, let us renumber the points according to their distance from $\{u, v\}$. Thus, $d(1, \{u, v\}) \leq d(2, \{u, v\}) \leq \cdots \leq d(n, \{u, v\})$. Let us assume for now that $d(u, v) \leq \frac{\delta}{8}$. We notice that i can cut $\{u, v\}$ only if $\frac{\delta}{8} \leq d(i, \{u, v\}) \leq \frac{\delta}{2}$. Assuming w.l.o.g that $d(i, u) < d(i, v)$, we get that in order to cut $\{u, v\}$ by the ball centered around i, it needs to hold that $R \in [d(i, u), d(i, v)]$. The probability of such event is easy to compute:

$$Pr[R \in [d(i,u), d(i,v)]] = \frac{d(i,v) - d(i,u)}{\frac{\delta}{4}} \leq \frac{d(u,v)}{\frac{\delta}{4}} \tag{2.2}$$

where the last part holds due to triangle inequality. Also, in order for the ball around i to cut $\{u, v\}$, i must be the first amongst $[i]$ in the random permutation σ. We notice that the probability for that to happen is $\frac{1}{i}$. Combining that with Equation 2.2 we get:

$$\begin{aligned} Pr[u, v \text{ separated at diam } \delta] &= \sum_{i: \frac{\delta}{8} \leq d(i,\{u,v\}) \leq \frac{\delta}{2}} Pr[u, v \text{ separated by } i \text{ at diam } \delta] \\ &\leq \sum_i \frac{1}{i} \frac{4d(u,v)}{\delta} \\ &\leq O(log(n)) \frac{d(u,v)}{\delta} \end{aligned} \tag{2.3}$$

We notice that since we sum over all i values we get the same result as in the previous lecture (accumulating all δ values will get us $O(d(u, v) \cdot log^2(n))$ as before). But we can do better! Remembering that

$$\mathbb{E}[d_T(u,v)] \leq \sum_\delta c \cdot \delta \cdot Pr[u, v \text{ separated at diam } \delta] \tag{2.4}$$

(for some constant c) we can refine our counting a bit:

$$\begin{aligned} \mathbb{E}[d_T(u,v)] \leq\ & c \cdot \delta \cdot \frac{d(u,v)}{\delta} \sum_{i: \frac{\delta}{8} \leq d(i,\{u,v\}) \leq \frac{\delta}{2}} \frac{1}{i} \\ & + c \cdot \frac{\delta}{2} \cdot \frac{d(u,v)}{\frac{\delta}{2}} \sum_{i: \frac{\delta/2}{8} \leq d(i,\{u,v\}) \leq \frac{\delta/2}{2}} \\ & + c \cdot \frac{\delta}{4} \cdot \frac{d(u,v)}{\frac{\delta}{4}} \sum_{i: \frac{\delta/4}{8} \leq d(i,\{u,v\}) \leq \frac{\delta/4}{2}} \frac{1}{i} \\ & \vdots \\ & + c \cdot \frac{\delta}{2^i} \cdot \frac{d(u,v)}{\frac{\delta}{2^i}} \sum_{i: \frac{\delta/2^i}{8} \leq d(i,\{u,v\}) \leq \frac{\delta/2^i}{2}} \frac{1}{i} \\ & \leq O(d(u,v) \cdot log(n)) \end{aligned} \tag{2.5}$$

and we managed to shave off the extra $log(n)$, due to the fact that in the above expression we sum each $\frac{1}{i}$ at most twice.

3 Min Communication Spanning Tree

Let us start with some definitions. We would like to define a mapping from the tree leaves to the graph vertices as before, but extend that mapping to include all vertices of the tree. Also, we would like to define a mapping from tree edges to paths in the graph. Given a tree $T(E_t, V_t)$ and a graph $G(V, E)$ we will define the following mappings:

$$\begin{aligned} &m_v : V_t \to V \text{ (node mapping)} \\ &m_e : E_t \to E^* \text{ (edge-path mapping)} \end{aligned} \tag{3.1}$$

such that m_e maps $e_t = (u_t, v_t)$ to the path P_{uv} between $u = m_v(u_t)$ and $v = m_v(v_t)$. Given requirements $r(u, v) \geq 0$ our goal is to pick a single tree T that minimizes $\sum d_T(u, v) \cdot r(u, v)$. Comparing to $\sum d_G(u, v) \cdot r(u, v)$ (the best we can achieve), we can use the tree embedding to get a randomized algorithm with $O(log(n))$ ratio.

We can define the requirement of a tree edge:

$$r(e_t) = \sum_{\substack{u \in V_{e_t} \\ v \in \overline{V_{e_t}}}} r(u, v) \tag{3.2}$$

where V_{e_t} and $\overline{V_{e_t}}$ denote the two disjoint sets of graph vertices defined by removing e_t and looking at all leaves in each of the two resulting disjoint trees. We notice that using these definition we get $load_T(e) = \sum_{\substack{e_t \in E_t \\ e \in m_e(e_t)}} r(e_t)$ and we can rewrite the expression for $cost(T)$:

$$cost(T) = \sum_{e \in E} load_T(e) \cdot l(e) \tag{3.3}$$

We have already shown, using tree embedding, that the following holds:

$$cost(T) \leq O(log(n)) \sum_{r(e)} l(e) \tag{3.4}$$

The next step is to reuse the same infrastructure, defined over capacities, to approximate the minimum congestion multicommodity flow. Moving on to capacities, we can define:

$$c(u_t, v_t) = \sum_{\substack{u \in V_{e_t} \\ v \in \overline{V_{e_t}}}} c(u, v) \tag{3.5}$$

where V_{e_t} and $\overline{V_{e_t}}$ are defined exactly as before. Given a multicommodity flow f in G and congestion C_G, we would like to map it to a tree T with congestion C_T such that $C_T \leq C_G$ and also get a probability distribution $\{\lambda_i\}$ on trees T_i s.t.

$$min\beta = \max_{e \in E} \left\{ \sum_i \lambda_i \frac{load_T(e)}{c(e)} \right\} \tag{3.6}$$

In the next lecture we will show that the above is possible with $\beta = O(log(n))$, using the same mechanism we developed throughout this lecture. This in turn will imply that we can solve any instance of minimum congestion multicommodity flow, obliviously, with approximation $O(log(n))$.