# 10 Embedding into Sub-trees

In the last two lectures we dealt with the problem of embedding finite metrics into a distribution over dominating trees and we have seen that it is possible to obtain a distortion of $O(\log n)$ for this problem. This is optimal since any metric that can be expressed as a distribution over tree-metrics can be embedded isometrically into $\ell_1$, and hence an improved distortion for embedding into tree-distributions would imply an improved distortion for embedding into $\ell_1$ which is not possible due to the results from lecture 3.

In this class we consider embeddings into distributions over dominating trees where the dominating trees have to be sub-trees of the underlying graph $G$. This is a much more restricted scenario and we will only be able to show a distortion of $O(\log^2 n \log\log n)$ for this problem.

The algorithm that we will use for embedding shortest-path metrics of a graph into a distribution over sub-trees uses the concept of a star-decomopsition which is defined as follows.

**Definition 10.1 (star-decomposition)** *A $\delta$-star-decomposition of a graph $G$ with a designated root node $r_0$ is a set of disjoint connected components $G_0 = (V_0, E_0), \ldots, G_k = (V_k, E_k)$ together with a collection of root nodes $r_0, \ldots, r_k$ such that*

- *$r_i \in V_i$ for all $0 \le i \le k$,*
- *each $r_i$, $1 \le i \le k$ has a neighbor in $V_0$, and*
- *each component has* $\text{radius}(r_i, G_i) \le \delta$ *.*

Here the radius of a graph $G$ with respect to a root node $r$ is $\text{radius}(r, G) = \max_{v \in V(G)} d(r, v)$. See Figure 10.1 for an example of a star-decomposition. The algorithm for constructing a spanning tree of $G$ is the following.

**Algorithm:**

given a graph $G$ with root node $r_0$ and radius $\Delta$

find a $\frac{7\Delta}{8}$-star-decomposition with clusters $G_0, G_1, G_2, \ldots$

for each root node $r_i$, $i \ge 1$ mark some edge
that connects $r_i$ to the center cluster $G_0$

delete all unmarked edges going between components

recurse on the components

the remaining edges form a spanning tree of $G$

The crucial step in this algorithm is to find a star-decomposition of a given graph $G$. Before presenting this algorithm let us first sketch the analysis that we want to do. We assume for simplicity that all edges have length 1, and that therefore the diameter is bounded by $n$.

Assume, for the time being that if an edge $e = (x, y)$ is cut during a $\delta$-star-decomposition in the above scheme, then it has length $O(\delta)$ in the final tree. If we can bound the probability

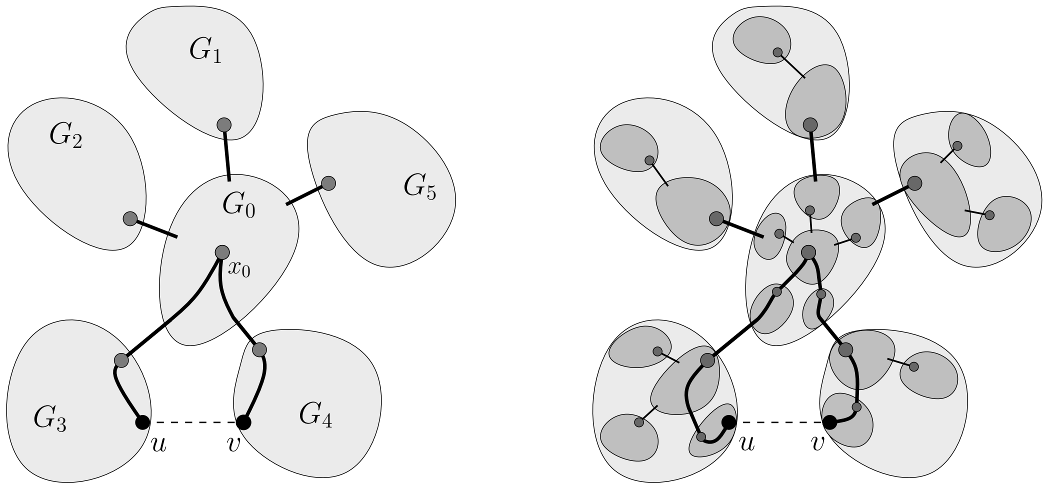


Figure 10.1: Recursive application of the star-decomposition. Note that the initial $u$-$v$ path increases in length due to later levels of the recursion.

that an edge $e$ is cut during a $\delta$-star-decomposition by something like

$$\mathbf{Pr}[\text{edge } e \text{ is cut}] \le \frac{1}{\beta} \cdot \frac{d(e)}{\delta}$$

for some parameter $\beta \le 1$, then we get an expected distance between $x$ and $y$ of

$$\begin{aligned} E[d_T(x,y)] &= \sum_{\text{levels } j} \frac{1}{\beta} \frac{d(e)}{\Delta \cdot (\frac{7}{8})^j} \cdot O(\Delta \cdot (\frac{7}{8})^j) \\ &= O(\log n \cdot \frac{1}{\beta}) \cdot d(e) \ . \end{aligned}$$

However, the crucial part of the analysis is to ensure our assumption that if two nodes $x$ and $y$ are cut during a $\delta$-star-decomposition, that then their final distance $d_T(x,y)$ will be $O(\delta)$.

If we consider the shortest path between $x$ and $y$, *right after*[1] the $\delta$-star-decomposition, then we can claim that $x$ and $y$ are at distance at most $O(\delta)$ (see Figure 10.1). This is due to the fact that each component has radius at most $\delta$ and the path $x \to r_x \to r_0 \to r_y \to y$ has length at most $4\delta + 2 = O(\delta)$.

However, during later levels of the recursion the length of this path may actually increase as further edges in $G_0, G_1, G_2, \ldots$ are deleted. The main idea of the following algorithm is to bound the amount of this increase. For this, observe that after each level of the recursion (after $x$ and $y$ have been cut) we can view the path between $x$ and $y$ as a concatenation of paths

[1] i.e., edges between $G_0, G_1, G_2, \ldots$ have already been deleted, but we did not yet start the recursive edge-deletion inside $G_0, G_1, G_2, \ldots$.

going from a node of a component to the root of the component (these paths are concatenated by edges that have already become tree-edges).

The idea is to show that during one level of the recursion a node-to-root path does not increase its length by more than a factor of $(1 + \frac{1}{\log n})$. Since there are only $O(\log n)$ levels of recursion the total increase over all levels will then be bounded by $(1 + \frac{1}{\log n})^{O(\log n)} = O(1)$. Hence we can claim that if an edge $x$,$y$ is cut during a $\delta$-star-decomposition its final distance in the tree will be $O(\delta)$.

The following algorithm for constructing a star-decomposition is designed with the aim that an edge is cut with small probability and a node-to-root path *does not* increase its length too much. The algorithm first constructs the center component, as follows.

**Construct component $G_0$ (forward cut):**

Choose a radius $\gamma$ uniformly at random from the interval $[\Delta/4, \Delta/2]$. Cut all edges at distance $\gamma$ from the root $r_0$ (more formally: cut edges $(u, v) \in E$ for which $d(r, u) \leq \gamma \leq d(r, v)$). Let $V_0$ denote the connected component that contains $r_0$.

Let $x_1, \ldots, x_s$ denote the vertices in $V \setminus V_0$ that had a neighbor in $V_0$ (which is now cut away). We call these nodes *portal nodes*. In order to select the components $V_1, \ldots, V_k$ we successively cut pieces from the remaining graph $G \setminus V_0$. We will again use random radius cuts to do this; however we first introduce a new distance function on the node set $X = V \setminus V_0$ and the random radius will then be interpreted with respect to this distance function.

For the definition of the new distance function we replace each edge in $G \setminus V_0$ by two directed edges in opposite direction. We define the length $\ell(u, v)$ of such a directed edge $(u, v)$ as

$$\ell(u, v) = \begin{cases} 1 & \text{if } d(r_0, v) = d(r_0, u) - 1 \\ 1 & \text{if } d(r_0, v) = d(r_0, u) \\ 0 & \text{if } d(r_0, v) = d(r_0, u) + 1 \end{cases}$$

By this length-definition we get a shortest path distance on $X$ which we call the *backward-edge distance* in the following (the distance from $x$ to $y$ along a path counts how often the distance to $r_0$ does not increase while going from $x$ to $y$).

Using the new distance function we construct the components $G_1, \ldots, G_k$ in the following way. (Note that we only used directed edges to define the new distance function on $X$. In the following all edges are undirected again.)

**Construct components $G_1, \ldots, G_k$ (backward cuts):**

As long as there exists a portal node $x_i$ that is not yet assigned to any component, construct a new component as follows.

Start a region-growing with probability $p$ (defined later) from node $x_i$ with respect to the backward-edge-distance. This means you flip a coin that comes up heads with probability $p$. If it comes up heads you cut all edges at backward-edge-distance 1. If not, you increase the radius and try again (cf. Bartal Region Growing, Lecture 5).

After you made a cut the portal node $x_i$ becomes the root of the new component.

Note that the definition of backward-edge distance guarantees that whenever there is a node that is not yet assigned to a component then there is a portal node that is not assigned to a component. This property is in fact one major reason behind the definition of the backward-edge distance.

If we choose $p = \frac{32\log^2 n}{\Delta}$ the above region growing process will make a cut after at most $\Delta/(16\log n)$ steps with high probability.

The following claim shows that the above algorithm really constructs a $\frac{7\Delta}{8}$-star-decomposition with high probability.

**Claim 10.2** *With high probability the radius of component $V_i$ is at most $\frac{7}{8}\Delta$.*

**Proof.** Let $s$ denote the length of a shortest path in $G_i$ from root $r_i$ to some other node $v \in V_i$. We show that with high probability $s \leq \frac{7}{8}\Delta$.

With high probability the random radius $\gamma$ for the backward cut that created component $G_i$ is at most $\Delta/16$. This means that all but $\Delta/16$ edges on the path from $r_i$ to $v$ increase the distance from the initial root node $r_0$. The node $r_i$ is at least at distance $\Delta/4$ from $r_0$ because of the parameters in the definition of a forward cut. This means that $v$ is at least at distance $\Delta/4 + s - 2\frac{\Delta}{16}$ from $r_0$ (Note that each step that does not increase the distance to $r_0$ can decrease it at most by one since all edges have length 1). Since this has to be less than $\Delta$ we get $s \leq \frac{7}{8}\Delta$. ■

**Claim 10.3** *One level of the recursion only increases the distance of a node-to-root path by a factor of at most $(1 + \frac{1}{\log n})$.*

**Proof.** Fix a node $v$, and let $d$ denote the distance function on the given (sub-)graph $G$. If the distance $d(r_0, v) < \Delta/4$, then $v$ will be contained in the center cluster and the distance from $v$ to the root doesn't change at all. Therefore, assume that after creating the star-decomposition of the input cluster, $v$ is contained in some component $G_i$, $i \neq 0$.

Fix the shortest path from $r_i$ to $v$ with respect to backward-edge distance, and let $s$ denote the number of edges on this path. Nearly, along every edge the distance to $r_0$ increases (there are at most $\frac{\Delta}{16\log n}$ edges where the distance decreases because of the small backward-edge-radius). Since the final distance (at the end of the path) is $d(r_0, v)$ it must hold that $d(r_0, r_i) + s - 2\frac{\Delta}{16\log n} \leq d(r_0, v)$, which gives

$$d(r_0, r_i) + s \leq d(r_0, v) + 2\frac{\Delta}{16\log n} \leq (1 + \frac{1}{\log n}) \cdot d(r_0, v) \ .$$

Now, observe that the shortest path going from $r$ to $r_i$ still (after deletion of edges) has length $d_(r_0, r_i)^2$. This means there is a path of length at most $(1 + \frac{1}{\log n}) \cdot d(r_0, v)$ from $r_0$ to $v$. ■

[2] For this we need that the edge from $r_i$ to its predecessor on the shortest $r_0$-$r_i$-path survives. We simply choose this edge as the tree edge that connects $r_i$ to the center cluster $G_0$

The probability of cutting an edge in the above scheme is $O(4/\Delta)$ for cutting in a forward-cut, and $O(\log^2 n/\Delta)$ for cutting in the backward-cutting phase. This gives an overall probability of $O(\log^2 n/\Delta)$ which results in a $\beta \geq \Omega(\frac{1}{\log^2 n})$. Altogether this scheme guarantees a distortion of $O(\log n/\beta) = O(\log^3 n)$.

### 10.1 Improving the result to distortion $O(\log^2 n \log\log n)$

The analysis in the previous section can be improved by using the observation that the increase of length of an $x$-$y$ path is much smaller in the expected sense. Choose $p = \Theta(\frac{\log n \log\log n}{\Delta})$. Then, on expectation, the backward-region growing will make a cut after $\Delta/(\log n \log\log n)$ steps. With high probability it will make a cut after $\Delta/\log\log n$ steps.

Suppose that the pair $(x, y)$ is cut on some level $i$. For the following $O(\log\log n)$ levels we use the high probability bound to bound the increase in the path-length between $x$ and $y$. This gives us that after $O(\log\log n)$ levels the distance has only increased by a constant factor, i.e., it is now something like $\Theta(\Delta_i)$ (Note that the path has length larger than $\Delta_i/2$ because the path goes over the root-node corresponding to $x$ and $y$ in the level $\Delta_i$ partition). However the path is a concatenation of node-to-root paths in a partition where each component now has radius $\leq (\frac{7}{8})^j \Delta_i$, where $j = \Omega(\log\log n)$ is the number of levels after level $i$. This means that each of this sub-path has a very small length (say $\leq O(\Delta_i/\log^2 n)$). This, in turn, means that the increase in path-length between $x$ and $y$, which is the sum of the increases of every sub-path, is a sum of independent random variables (independent because each sub-path is contained in a different component, and the recursions on these components are independent) where each random variable has a contribution of at most $O(\Delta_i/\log^2 n)$. The *expected increase* in path length is $\Theta(\Delta_i/\log n)$. We can use Chernoff Bounds to show that with high probability the increase in path length will be very close to its expectation. This means that after $O(\log\log n)$ levels the increase in path-length can be bounded in terms of the expected increase which is only a $1 + \frac{1}{\log n \log\log n}$ factor. Increasing the distance by this factor over the remaining $\Theta(\log n)$ levels, only accumulates to a constant factor increase.

Altogether if $x$ and $y$ are cut on level $i$ they will be at distance $\Delta_i$ in the final tree (w.h.p.). This scheme cuts an edge with probability $O(\log n \log\log n/\Delta)$ and gives therefore a distortion of $O(\log^2 n \log\log n)$.

## References


[DGR06] Kedar Dhamdhere, Anupam Gupta, and Harald Räcke. Improved embedding of graph metrics into random trees. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 61–69, 2006.

[EEST05] Michael Elkin, Yuval Emek, Daniel Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2005.

[Kar98] Richard M. Karp. A $2k$-competitive algorithm for the circle. Manuscript, 1998.