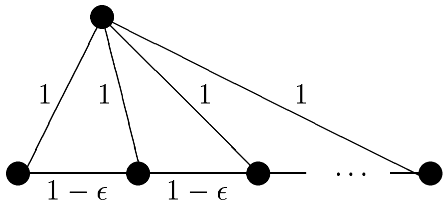
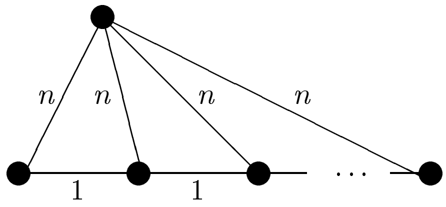
15-859(J): Metric Embeddings and Methods **Lecturer:** R. Ravi
Topic: Spanners **Date:** 23 Sept 2003
Scribe: Charlie Garrod

7.1 Introduction to Spanners

Given a weighted graph $G = (V, E)$ with distance function $l : E \to \mathbb{R}^+$, a spanner of G is a lightweight, sparse distance-preserving subgraph. Spanners have many applications in areas ranging from distributed computing to computational biology.

Minimizing the weight of a spanning tree and preserving distances from a given root are often conflicting goals. Consider the following examples:





In the first example the lightest spanning tree preserves distances extremely poorly, distorting distances by a factor of n. However, a spanner exists that preserves distances exactly and is only slightly heavier than the lightest spanning tree. In the second example the lightest spanner to exactly preserve distances is extremely heavy, while the lightest spanning tree distorts distances by only a constant factor. In both cases, achieving one goal exactly results in extremely poor performance with respect to the other criterion.

In this lecture we examine two algorithms that produce spanners that are both lightweight and approximately distance-preserving. We first describe Khuller, Raghavachari and Young's Light Approximate Shortest-path Trees (LASTs), which preserve approximate distances between a single root node and each other node in the tree [3]. We then describe a solution by Chandra et al. which preserves approximate distances between all node pairs [2].

7.2 LASTs

Theorem 7.2.1 *Given a graph $G = (V, E)$ with distance function $l : E \to \mathbb{R}^+$, a root node r, and some α there exists a spanning tree T such that:*

(i) $\forall x \in V, d_T(r, x) \leq \alpha \cdot d_G(r, x)$

(ii) $\sum_{e \in T} l(e) \leq \left(1 + \frac{2}{\alpha - 1}\right) l(M)$

where M is the minimum spanning tree of G. In other words, distances in T are an α-approximation of those in G and T is at most $\left(1 + \frac{2}{\alpha-1}\right)$ times as heavy as G's minimum spanning tree.

Proof: We prove Theorem 7.2.1 by construction. The algorithm we use is basically the greedy algorithm. We start with the minimum spanning tree of G, and augment our solution with the shortest possible path if a given path is too long.

Let M be the minimum spanning tree of G rooted at r and S be the shortest path tree of G also rooted at r. Let $v_0, v_1, \ldots, v_n$ be an in-order traversal of M (with $r = v_0$).

Consider the following algorithm:

Init: $H_0 \leftarrow M$
$m_0 = r$
$j = 0$

Do: For i in 1 to n-1 do
  If $d_{H_j}(r, v_i) > \alpha \cdot d_G(r, v_i)$
    j++
    $H_j = H_{j-1} \cup \mathrm{Path}_S(r, v_i)$
    $m_j = v_i$

Output: T = shortest path tree of H_j rooted at r

Observe that this algorithm trivially satisfies $d_T(r, x) \leq \alpha \cdot d_G(r, x)$ for all x. The algorithm considers each vertex and adds the shortest possible path in G from r if that requirement has not yet been met, and finally outputs a shortest path tree from r. We merely need to show that T satisfies the second requirement, ie. that T is not too heavy.

Let P be the Euler path associated with the in-order traversal of M. Since P traverses each edge of M twice we have that $l(P) = 2 \cdot l(M)$.

Every time the algorithm augments the graph to form H_j it records a milestone vertex, m_j. Since $d_{H_{j-1}}(r, m_j) > \alpha \cdot d_S(r, m_j)$ we have that *any* path in H_{j-1} was too long, and thus that the particular path via the previous milestone vertex is also too long. Therefore we have $d_{H_{j-1}}(r, m_{j-1}) + d_P(m_{j-1}, m_j) > \alpha \cdot d_S(r, m_j)$ and thus that $d_S(r, m_{j-1}) + d_P(m_{j-1}, m_j) > \alpha \cdot d_S(r, m_j)$. We can then sum over all j = 1 to k to yield

$$\sum_{j=1}^{k} [d_S(r, m_{j-1}) + d_P(m_{j-1}, m_j)] \quad > \quad \alpha \cdot \sum_{j=1}^{k} d_S(r, m_j)$$

which telescopes and can be simplified to

$$\sum_{j=1}^{k} d_P(m_{j-1}, m_j) \quad > \quad \left[(\alpha - 1) \sum_{j=1}^{k-1} d_S(r, m_j) \right] + \alpha \cdot d_S(r, m_k)$$

But the lefthand side of the inequality is at most the length of the entire Euler path P, which yields

$$2 \cdot l(M) \quad > \quad (\alpha - 1) \sum_{j=1}^{k} d_S(r, m_j)$$

But the total size of T is then just

$$\begin{aligned} l(T) &= l(M) + \sum_{j=1}^{k} d_S(r, m_j) \\ &< \left(1 + \frac{2}{\alpha - 1}\right) l(M) \end{aligned}$$

as desired. ■

As an exercise consider the following computational problems. How hard is it to find the following?

1. An MST T that minimizes $\max_v d_T(r, v)$ for a given root node r?
2. An MST T that minimizes $\max_v \frac{d_T(r,v)}{d_G(r,v)}$ for a given root node r?
3. A shortest-path tree T of minimum total length?

For those problems that are intractable, to what extent are they approximable?

7.3 Preserving distances between all node pairs

Here we explore the variation of the problem in which we want to approximately preserve distances between all node pairs.

Theorem 7.3.1 *Given a graph $G = (V, E)$ with distance function $l : E \to \mathbb{R}^+$, and some α,δ with $\alpha - 1 > \delta > 0$, there exists a subgraph H of G such that:*

(i) $d_H(x, y) \leq \alpha \cdot d_G(x, y)$

(ii) $l(H) = \left(3 + \frac{16\alpha}{\delta^2}\right) n^{\frac{2\alpha\delta}{\alpha-1-\delta}} \cdot l(M)$

where M is the minimum spanning tree of G.

For example, if $\alpha = \Theta(\log n)$ and $\delta = \Theta(1)$ then we get an $\mathcal{O}(\log n)$-stretch spanner with weight $\mathcal{O}(\log n) \cdot l(M)$. If $\alpha = \Theta(\log^2 n)$ and $\delta = \Theta(\log n)$ then this yields $\mathcal{O}(\log^2 n)$-stretch spanners of weight $\mathcal{O}(1) \cdot l(M)$.

We prove only the special case of $\delta = 2$, yielding $l(H) = \mathcal{O}\left(\alpha n^{\frac{4}{\alpha-3}}\right) \cdot l(M)$.

Our proof utilizes a lemma about graph girth which we do not prove.

Definition 7.3.2 *The* girth *of an undirected graph is the length of its shortest induced chordless cycle.*

Lemma 7.3.3 *Given an undirected graph $G = (V, E)$ with girth g, we have $|E| \leq n \cdot \lceil n^{\frac{2}{g-2}} \rceil$.*

We prove the theorem by construction, again using a greedy algorithm. We inspect the edges in non-decreasing order by length, adding an edge to the graph if the shortest path between the edge's endpoints is currently too long. More formally, suppose that $l(e_1) \leq l(e_2) \leq \cdots \leq l(e_m)$ and consider the following algorithm.

Init: $H \leftarrow \emptyset$
Do: For i in 1 to m do
  let $(x, y) = e_i$
  If $d_H(x, y) > \alpha \cdot d_G(x, y)$
    $H = H \cup e_i$
Output: H

Lemma 7.3.4 $d_H(x, y) \leq \alpha \cdot d_G(x, y)\ \forall\ x, y \in V$.

Proof: The algorithm directly ensures that $d_H(x, y) \leq \alpha \cdot d_G(x, y)$ for each $(x, y) \in E$. Consider the case when $(x, y) \notin E$ and let $v_0, v_1, \ldots, v_k$ be the shortest path from x to y in G (so that $x = v_0$ and $y = v_k$). Our algorithm directly ensures that $d_H(v_i, v_{i+1}) \leq \alpha \cdot d_G(v_i, v_{i+1})$ for each adjacent pair of vertices in the path. Thus the total length of the associated path in H is no more than α times the total length of the path in G. We then have that $d_H(x, y) \leq \alpha \cdot d_G(x, y)\ \forall\ x, y \in V$ as desired. ■

Claim 7.3.5 *H contains a minimum spanning tree of G.*

This can be seen by observing that H contains every edge that Kruskal's algorithm would add if it considered edges in the same order.

We can therefore partition the edges of H into $E_T \cup E_0 \cup E_1 \cup \ldots \cup E_{\log n}$ in the following manner. E_T is a minimum spanning tree of G, E_0 contains all other edges of length $\left[0, \frac{L}{2n}\right]$, and in general E_j contains all edges not in E_T of length $\left(\frac{2^{j-1}L}{n}, \frac{2^j L}{n}\right]$ where $L = 2 \cdot l(E_T)$.

We can now bound the size of each of these components in terms of the weight of the minimum spanning tree, eventually allowing us to bound the size of H.

Claim 7.3.6 $l(E_0) \leq 2n^{\frac{2}{\alpha-1}} \cdot l(M)$.

Proof: Consider any cycle in E_0 with edges $e_1, e_2, \ldots, e_g$ with lengths $l_1 \leq l_2 \leq \ldots \leq l_g$ (e_i is not necessarily adjacent to e_{i+1}). From the algorithm we have that

$$\sum_{i=1}^{g-1} l_i > \alpha \cdot l_g$$

because otherwise the final edge l_g would not have been added. This then gives us

$$\begin{aligned} l_g(g-1) &= \sum_{i=1}^{g-1} l_g \\ &\geq \sum_{i=1}^{g-1} l_i \\ &> \alpha \cdot l_g \end{aligned}$$

and thus that $g > \alpha + 1$. We thus have that the girth of E_0 is greater than $\alpha + 1$ and by Lemma 7.3.3 that $|E_0| \leq n^{1+\frac{2}{\alpha-1}}$. Since each edge in E_0 is of length at most $\frac{L}{2n}$, this yields a total weight of $L \cdot n^{\frac{2}{\alpha-1}} = 2n^{\frac{2}{\alpha-1}} \cdot l(M)$. ■

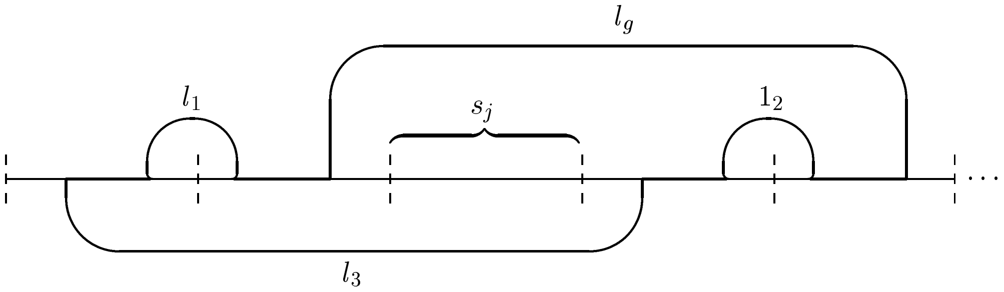


Figure 7.3.1: The horizontal line represents the Euler path P, broken into cluster segments of width s_j. A cycle in graph H_j of length $l_1 + l_2 + \ldots + l_g$ corresponds to a cycle in G of length at most $l_1 + l_2 + \ldots + l_g + g \cdot s_j$.

We generalize this proof to a similar claim about the total weight of each class E_j for $j \geq 1$. For each class E_j we cluster vertices into supernodes and form a multi-graph H_j which contains every edge in E_j as an edge. We then use a similar argument to bound the girth of H_j, again allowing us to bound the size of E_j and thus its total weight.

Let M be a minimum spanning tree of G, and consider an Euler tour P of M with length $L = 2 \cdot l(M)$. Separate the nodes on the tour into clusters of width $s_j = \frac{2^{j-1}L}{n}$, creating a total of $n_j = \frac{L}{\left(\frac{2^{j-1}L}{n}\right)} = \frac{n}{2^{j-1}}$ clusters.

For each $j \geq 1$ construct multi-graph H_j by creating a supernode v_i for each cluster above, contracting the vertices of G in each cluster. Let edge (v_i, v_k) be in H_j if there exists edge (x, y) in G for any x in supernode v_i and y in supernode v_k.

Claim 7.3.7 *Every edge in E_j is an inter-cluster edge and thus creates an edge in H_j.*

Proof: Consider $e = (u, v) \in E_j$. We have $l(uv) > \frac{2^{j-1}L}{n}$ and thus, by the triangle inequality, any path along P must be at least that long, giving us $d_P(u, v) > \frac{\alpha 2^{j-1}L}{n} = s_j$. Note that this is true for any appearance of u and v along P (each vertex appears in the Euler path P in multiplicity equal to its degree in G). Thus, the endpoints of u, v must be in different segments of P and (u, v) must be an inter-cluster edge. ■

This allows us to bound the size of E_j by bounding the number of edges in each multi-graph H_j. We do this as we did in the case of E_0, by bounding the girth of each H_j in terms of α.

Claim 7.3.8 $|E_j| \leq n_j^{1+\frac{4}{\alpha-3}}$

Proof: Let $C = e_1, e_2, \ldots, e_g$ be a cycle in H_j with edge weights $l_1 \leq l_2 \leq \ldots \leq l_g$. Notice that this cycle in H_j of length $\sum_j l_j$ implies an associated cycle of length $\sum_j l_j + g \cdot s_j$ in H, formed by taking the edges of C between clusters and taking a path of length at most s_j between nodes that were contracted into each supernode in the cycle as shown in Figure 7.3.1. Thus, by the same reasoning as in the proof of Claim 7.3.6 we have that the total weight of the cycle must be much larger than the heaviest edge, namely that

$$\sum_{j=1}^{g-1} l_j + g \cdot s_j \quad > \quad \alpha \cdot l_g$$

Substituting the value of s_j and simplifying yields that $g > \frac{\alpha+1}{2}$ which, using Lemma 7.3.3, gives us $|E_j| \leq n_j^{1+\frac{4}{\alpha-3}}$. ∎

Lemma 7.3.9 *For $j \geq 1$, $l(E_j) \leq 2Ln^{\frac{4}{\alpha-3}} \cdot \left(\frac{\alpha-3}{\alpha}\right)^{j-1}$*

Proof: Lemma 7.3.8 bounds the size of each partition E_j, and each edge in E_j is at most size $\frac{2^j L}{n}$ and $n_j = \frac{n}{2^{j-1}}$. This gives us the total weight of partition E_j:

$$\begin{aligned} l(E_j) &\leq \frac{2^j L}{n} \cdot n_j^{1+\frac{4}{\alpha-3}} \\ &= 2L \cdot \left(\frac{n}{2^{j-1}}\right)^{\frac{4}{\alpha-3}} \\ &= 2Ln^{\frac{4}{\alpha-3}} \cdot \left(\frac{1}{16^{\frac{1}{\alpha-3}}}\right)^{j-1} \\ &\leq 2Ln^{\frac{4}{\alpha-3}} \cdot \left(\frac{\alpha-3}{\alpha}\right)^{j-1} \end{aligned}$$

since $16^{\frac{1}{\alpha-3}} \geq 1 + \frac{1}{\alpha-3}$. ∎

Lemma 7.3.10 *$l(H) = (3+4\alpha)\, n^{\frac{4}{\alpha-3}} \cdot l(M)$ where M is a minimum spanning tree of G.*

Proof: We finally calculate $l(H)$ by summing the weights of each of its components examined above, yielding

$$\begin{aligned} l(H) &= l(E_T) + l(E_0) + \sum_{j=1}^{\log n} E_j \\ &\leq l(M) + 2n^{\frac{2}{\alpha-1}} \cdot l(M) + 2Ln^{\frac{4}{\alpha-3}} \cdot \sum_{j=1}^{\log n} \left(\frac{\alpha-3}{\alpha}\right)^{j-1} \\ &\leq l(M) + 2n^{\frac{2}{\alpha-1}} \cdot l(M) + 4 \cdot l(M) \cdot n^{\frac{4}{\alpha-3}} \cdot \alpha \\ &\leq 3n^{\frac{2}{\alpha-1}} \cdot l(M) + 4 \cdot l(M) \cdot n^{\frac{4}{\alpha-3}} \cdot \alpha \\ &\leq 3n^{\frac{4}{\alpha-3}} \cdot l(M) + 4 \cdot l(M) \cdot n^{\frac{4}{\alpha-3}} \cdot \alpha \\ &\leq (3+4\alpha)n^{\frac{4}{\alpha-3}} \cdot l(M) \end{aligned}$$

∎

References

[1] B. Awerbuch, A. Baratz, and D. Peleg. Efficient Broadcast and Light-weight Spanners. Manuscript, 1991.

[2] B. Chandra, G. Das, G. Narasimhan, and J. Soares. New Sparseness Results on Graph Spanners. *International Journal of Computational Geometry and Applications*, 1995.

[3] S. Khuller, B. Raghavachari, and N. Young. Balancing Minimum Spanning Trees and Shortest-Path Trees. *Proceedings of the ACM Symposium on Discrete Algorithms*, 1993.