15-859(J): Metric Embeddings and Methods **Lecturer:** Anupam Gupta
Topic: Min-Bandwidth Layouts and Volume Respecting Embeddings **Date:** 10/28/03
Scribe: Nikhil Bansal

Our goal in the next three lectures will be to study the notion of *Volume Respecting Embeddings.* In this lecture we develop the basic framework and see how this idea is used to obtain a non-trivial approximation algorithm for problem of finding the minimum bandwidth layout of a graph.

1 Minimum Bandwidth Problem

Definition 1.1 *Given an unweighted graph $G = (V, E)$ on n vertices, consider a 1-1 map f of the vertices V into $[1, n]$. The* bandwidth *of the map f is defined as the maximum stretch of any edge, i.e.,*

$$\mathsf{bw}(f) = \max_{(i,j)\in E} |f(i) - f(j)|.$$

The bandwidth *of a graph is defined as the minimum possible bandwidth achievable by any 1-1 map from $V \to [1, n]$. That is,*

$$\mathsf{bw}(G) = \min_{f:V\to[1,n]} bw(f).$$

The minimum bandwidth problem is quite hard even for very simple graphs. It is known to be NP-hard for trees, and in fact, even for trees that are *caterpillars* with hair length 3. Moreover, Unger (1998) showed that it is NP-Hard to approximate the bandwidth to within any constant factor for caterpillars as well.

As an aside, the name "bandwidth" comes from matrix algorithms; one can think of a min-bandwidth algorithm as trying to find a permutation of the vertices, such that looking at the adjacency matrix under this permutation causes all non-zero entries to lie in a thin "band" along the diagonal. This is quite a useful operation, since matrix operations on matrices with small bandwidth requires less space and time; e.g., consider taking a matrix-vector product—if the $n \times n$ matrix has bandwidth B, then we need time $O(nB)$, instead of $O(n^2)$. The popular software package `MatLab` has an inbuilt beuristic to find a permutation of vertices with good bandwidth, which is a variation of one due to Cuthill and McKee.

In this lecture we will be interested in *approximation algorithms* for this problem. A by-now classic result due to Uri Feige is that

Theorem 1.2 (Feige (1998)) *Given an unweighted graph $G = (V, E)$, the minimum bandwidth $\mathsf{bw}(G)$ can be approximated to within a factor of $O(\log^{3.5} n)$, where $|V| = n$.*

The proof of this result is extremely elegant, but it requires a few conceptual steps and quite a bit of machinery. The goal of this lecture, as well as the next two lectures, will be to develop this machinery. We will first begin with the following simpler result for trees.

Theorem 1.3 (Gupta (2000)) *For trees, the minimum bandwidth can be approximated to within a factor of $O(\log^{2.5} n)$.*

1.1 A Lower Bound on $\mathsf{bw}(G)$

We first give a simple lower bound on the bandwidth, on which all known approximation guarantees are based. To start off, note that if the maximum degree of a vertex in G is Δ, then $\lceil \Delta/2 \rceil$ is a lower bound on the bandwidth. This is because at least *half* its neighbours must be placed to one side of it, and the edge to the furthest neighbor on this "heavier" side must be stretched by at least $\lceil \Delta/2 \rceil$.

This idea can be naturally extended to look at "local density" around a vertex. For a vertex v and an integer r, let $B(v, r)$ denote the set of vertices within distance r from v. Then,

Lemma 1.4 *For all $v \in V$ and integers r, $|B(v, r) - 1|/2r \leq OPT$*

Proof: Consider the layout of the neighbors of v in $[1, n]$. At least two of these, say u and w, have a distance of $\geq B(v, r) - 1$ on the line. Since the distance of u and w in G is at most $2r$, it follows that some edge on the u-w path is stretched by at least $B(v, r) - 1/2r$. ■

Definition 1.5 *The* local density *of a graph G is*

$$D = D(G) = \min_v \min_r \left\lceil \frac{|B(v,r)-1|}{2r} \right\rceil .$$

By Lemma 1.4, $D \leq \mathsf{bw}(G)$.

2 Min-Bandwidth Algorithms for Trees

To get a feel for the problem we first consider a couple of algorithms that do not work. These will give us important intuition for why the correct algorithm is defined the way it is.

2.1 Algorithm 1

Let us assume that the tree $T = (V, E)$ is rooted at some vertex r. The first algorithm places vertics in order of their distance from the root, breaking ties arbitrarily.

Algorithm 1: Perform a breadth first search (BFS) on the tree starting at the root r. Place the vertices on the line in the order you see them in this BFS traversal (i.e. place them in any order of distance from the root r).

Lemma 2.1 *There are trees for which the above algorithm performs $\Omega(n^{1/2})$ worse than the local density D.*

Proof: Consider the n-vertex tree in Figure 2.1 below, which consists of $n^{1/2}/2$ "brooms" hanging off the root. Each broom is a path consisting of $n^{1/2}$ vertices with a star consisting of $n^{1/2}$ vertices at the end of this path. All the brooms in the graph have the same length.

It is easy to verify that the local density of this graph is $D = \Theta(n^{1/2})$. We claim that Algorithm 1 produces a layout with bandwidth $\Omega(n)$. Indeed, if we consider a BFS traversal, then all the $\Omega(\sqrt{n})$ vertices b_i will be laid out together, and followed by the $\Omega(n)$ leaves. Thus there is at least one b_i-leaf edge that is stretched by $\Omega(n)$. ■

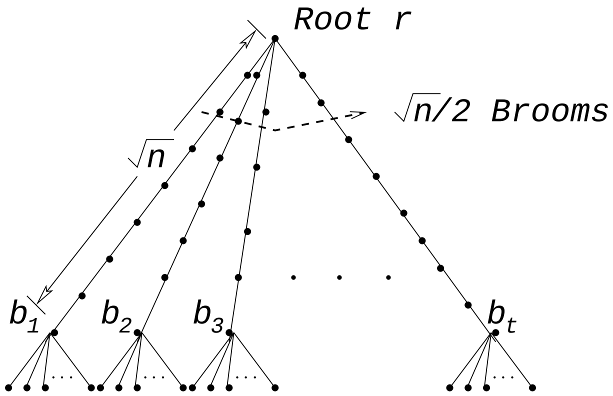

Figure 2.1: The bad example

2.2 Another Attempt: Algorithm 2

The problem with the algorithm above is that if we consider the distances from the root, we could see too many vertices together; this leads to the high bandwidth. A way to beat this might be to randomize the distances of the vertices from the root (without changing them by too much). This motivates us to consider our next natural algorithm.

Algorithm 2: Assign a "length" uniformly at random from the interval $[1, 2]$ to each edge. Place the vertices on the line in the order of the distance from the root.

However, it turns out that this algorithm is not too good either. We will now show that,

Lemma 2.2 *There are graphs on which the output of Algorithm 2 is greater than D by a factor of $\tilde{\Omega}(n^{1/4})$ (ignoring polylog(n) factors).*

Proof: The bad example is the same tree in Figure 2.1. Consider the $n/2$ leaves of the tree; their distance from the root is a sum of $k = \sqrt{n}$ i.i.d. random variables in $[1, 2]$. Now, by a Chernoff bound, the distance of a leaf from r lies in the range $\frac{3}{2}\, k \pm O(\sqrt{k \log k})$ with probability $1 - 1/n^2$. Hence, by a union bound, all the $n/2$ leaves lie in an interval of width $O(\sqrt{k \log k}) = \tilde{O}(n^{1/4})$ with constant probability.

Let us condition on that event happening. Now consider the leaf u which is farthest from r in the random experiment, and let w be an ancestor of u such that there are $O(n^{1/4} \log n)$ edges between them. Since w will be closer to the root than all the leaves, it will be encountered before the leaves. But then a path of $O(n^{1/4} \log n)$ edges will span $\Omega(n)$ leaves, and hence some edge will be stretched by at least $\tilde{\Omega}(n^{3/4})$, which implies the lemma. ■

If we see closely, the problem with this algorithm was that there was "too much" independence in the randomness used by this algorithm, causing the variance to be reduced and too many vertices were assigned a distance close to their mean. We will now see how to fix this.

2.3 An Algorithm that Works

We will use the following lemma:

Lemma 2.3 *It is possible to color the edges of a rooted tree such that*

1. *Each color class lies within some root to leaf path.*
2. *Each root to leaf path has most* $\log n$ *colors.*

Proof: Let us sketch a proof: Let us start at the root r; if it has $\chi(r)$ children, then imagine $\chi(r)$ different painters, each with a different color walking down each edge (away from the root) painting the edge as they go. When a painter reaches the other end of an edge, which is a vertex v with $\chi(v)$ children, it creates $\chi(v) - 1$ new painters (with new distinct colors). It then goes down the edge to the subtree with the most leaves, and sends the new painters down the other $\chi(v) - 1$ edges. It is clear that each color class it contained within a root-leaf path.

How many colors do we see on a root-leaf path; or equivalently, how many new colors were used? Let us consider a vertex v with child u, and a new painter was sent down the edge (v, u). But then we claim that the number of leaves in the subtree T_u rooted at u is at most half the number in the subtree T_v rooted at v. Indeed, if it were not, then T_u would be the subtree with the most leaves and the same color would be used. Hence, a new color implies that the number of leaves halves, and hence the number of new colors is at most $\log(\#\text{ of leaves}) \leq \log n$. ■

It is not hard to see (by considering a complete binary tree) that this lemma is the best one can hope for.

2.3.1 The Algorithm

We now present the final (and correct) algorithm:

Algorithm 3: Consider the coloring of the tree satisfying the conditions in Lemma 2.3. For each color class C, assign all the edges in C the same random number independently and uniformly from the range $[1, 2]$. Again, output the vertices in the order of their distance from the root.

Before we begin, let $\psi(v)$ be the random distance from r to the vertex v. Note that since ψ is 1-1 with probability 1, ψ uniquely defines the layout f that is the output of Algorithm 3:

$$f(u) < f(v) \iff \psi(u) < \psi(v).$$

2.3.2 The "tree-volume" Tvol

Let d_G be the shortest path distance on the vertices V, and hence (V, d_G) is a metric. Given a set of vertices $S \subseteq V$, let d_S be the induced metric in S. Then (S, d_S) can be viewed as the complete graph with the edge (s_1, s_2) having weight $d_S(s_1, s_2)$.

Definition 2.4 *Let $T(S)$ denote the minimum spanning tree on (S, d_S). Define* $\mathsf{Tvol}(S)$*, the* tree-volume *of S, to be*

$$\mathsf{Tvol}(S) = \prod_{e \in T(S)} d_S(e).$$

2.3.3 The Roadmap

In this lecture, we only describe the road map for the proof that the algorithm has a competitive ratio of $O(\log^{2.5} n)$. There will be four main conceptual steps:

1. We will show that the "chance of a set falling into a unit interval is small". Formally, for any set $S \subseteq V$, $|S| = k$,

$$\mathbf{Pr}[\psi(S) \text{ lies in some interval } [i, i+1)] \leq \frac{\eta^{k-1}}{\mathsf{Tvol}(S)} \tag{2.1}$$

   The values of the parameter η we can prove will be specified later.

2. Call a set *bad* if it does not satisfy (2.1) above. By linearity of expectation, (2.1) implies that the expected number of bad sets can be bounded by

$$\mathbf{E}[\text{ number of bad sets }] \leq \sum_{|S|=k} \frac{\eta^{k-1}}{\mathsf{Tvol}(S)} \tag{2.2}$$

3. Let D denote the local density lower bound. We will show that

$$\sum_{|S|=k} \frac{1}{\mathsf{Tvol}(S)} \leq n(D \log n)^{k-1} \tag{2.3}$$

   This allows us to relate the expected number of bad sets to the lower bound D.

$$\mathbf{E}[\text{ number of bad sets }] \leq n(\eta \log n\, D)^{k-1} \tag{2.4}$$

   By Markov's inequality, we get that with probability $\frac{1}{2}$,

$$\text{number of bad sets } \leq 2\, n(\eta \log n\, D)^{k-1} \tag{2.5}$$

4. Now suppose the bandwidth of the layout produced by Algorithm 3 is B. This implies that there is some edge (u, v) such that $B - 1$ vertices $x_1, x_2, \ldots, x_{B-1}$ have

$$\psi(u) < \psi(x_1) \leq \psi(x_2) \leq \ldots \leq \psi(x_{B-1}) \leq \psi(v). \tag{2.6}$$

   Since the length of the edge (u, v) is at most 2 (i.e., $\psi(v) - \psi(u) \leq 2$), at least half of the values $\psi(x_i)$ must lie within some unit interval $[i, i+1)$.

   However, this implies that there are at least $\binom{B/2}{k}$ bad sets of size k. However, we know an upper bound (2.5) on the number of bad sets, and hence

$$2n(\eta \log n D)^{k-1} \geq \left(\frac{B/2}{k}\right)^k \tag{2.7}$$

   Setting $k = \log n$ and simplifying, we get that

$$B/D \leq n^{1/k} \eta k \log n \leq \eta \log^2 n \tag{2.8}$$

This same road-map will be used in subsequent classes to prove Theorem 1.2.

Note that, to complete the proof, we need to prove the facts in (2.2) (for some value of η) and (2.3). We sketch the main idea behind proving (2.2) with $\eta = \sqrt{\log n}$ here; the fact (2.3) is a fact about metrics that is independent of trees, and will be proved in a subsequent class.

2.3.4 Lower Bounds on Concentration of Variables

We now describe the main ingredient which gives an intuition as to why the randomized Algorithm 3 does not suffer from the drawbacks present in Algorithm 2. Loosely, this is because the sum of $\log n$ "well-spread-out" random variables (which define $\psi(v)$) is not very closely concentrated around the mean.

We will be interested in bounding the probability that if $I = [i, i+1) \subseteq R_{\geq 0}$ is a unit interval, then for any vertex v what is $\mathbf{Pr}[\psi(v) \in I]$?

Theorem 2.5 (Leader and Radcliffe (1994)) *Suppose* $d_1, \ldots, d_k \geq 1$, *and let* X_i *be chosen uniformly at random in* $[1, 2]$. *Then*

$$\mathbf{Pr}\Big[\sum d_i X_i \in I\Big] \leq \frac{1}{\sqrt{\sum_{i=1}^{k} d_i^2}} \leq \frac{\sqrt{k}}{\sum d_i}.$$

To see that this implies that

$$\mathbf{Pr}[\psi(v) \in I] \leq \frac{\sqrt{\log n}}{d_T(r, v)} = \frac{\eta}{\mathsf{Tvol}(\{r, v\})},$$

let the d_i's be the lengths of the color classes on the v-root path. Since each length is chosen independently in $[1, 2]$, $\sum_i d_i X_i$ models the behavior of $\psi(v)$, and shows that it is not too concentrated around the mean. This idea can be extended to show the fact (2.2). For details, read the paper of Gupta (2000).

2.3.5 Littlewood-Offord type problems

We will not prove Theorem 2.5 here as well, but let us prove a simpler bound.

Theorem 2.6 *Suppose* $d_1, \ldots, d_k \geq 1$, *and let* Y_i *be chosen uniformly at random in* $[1, 2]$. *Then*

$$\mathbf{Pr}\Big[\sum d_i Y_i \in I\Big] \leq \binom{k}{k/2} 2^{-k} \approx \frac{1}{k}.$$

Proof: Consider the set of all $\{-1, +1\}$ assignments to Y_i such that $\sum Y_i d_i \in I$. We can view this as a lattice, where an assignment $\mathcal{A}_1$ is below another assignment $\mathcal{A}_2$ if all the positions at which $\mathcal{A}_1$ is $+1$ is a subset of those at which $\mathcal{A}_2$ is $+1$. Note that since $d_i \geq 1$ for all i, the assignments for which $\sum Y_i d_i \in I$ will form an antichain in this lattice and hence their number cannot be more than $\binom{k}{k/2}$. Since each assignment has probability 2^{-k} the claim follows. ■

References

Feige, U. (1998). Approximating the bandwidth via volume respecting embeddings. *J. Comput. System Sci.*, 60(3):510–539. 30th Annual ACM Symposium on Theory of Computing (Dallas, TX, 1998).

Gupta, A. (2000). Improved bandwidth approximation for trees and chordal graphs. *J. Algorithms*, 40(1):24–36. (Preliminary version in *11th SODA*, 2000).

Leader, I. and Radcliffe, A. J. (1994). Littlewood-Offord inequalities for random variables. *SIAM J. Discrete Math.*, 7(1):90–101.

Unger, W. (1998). The complexity of the approximation of the bandwidth problem. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 82–91.