Improved Bandwidth Approximation for Trees

Anupam Gupta*
Computer Science Division
University of California
Berkeley CA 94720.
Email: angup@cs.berkeley.edu

Abstract

A linear arrangement of an n-vertex graph $G = (V, E)$ is a one-one mapping f of the vertex set V onto the set $[n] = \{0, 1, \ldots, n-1\}$. The bandwidth of this linear arrangement is the maximum distance between the images of the endpoints of any edge in $E(G)$. When the input graph G is a tree, the best known approximation algorithm for the minimum bandwidth linear arrangement (which is based on the principle of volume respecting embeddings) outputs a linear arrangement which has bandwidth within $O(\log^3 n)$ of the optimal bandwidth. In this paper, we present a simple randomized $O(\log^2 n\sqrt{\log n})$-approximation algorithm for bandwidth minimization on trees.

1 Introduction

The *Bandwidth Minimization* problem is the following: given an undirected graph $G = (V, E)$, find a one-one mapping of the vertices $f : V \to [n]$ whose *bandwidth*, which is defined to be

$$\max_{(i,j)\in E} |f(i) - f(j)|,$$

is minimum. This problem is equivalent to the matrix bandwidth minimization problem, which, given a square symmetric matrix M, seeks to find a permutation matrix P such that PMP^T has all its non-zero entries in a band of minimum width about the diagonal. This not only reduces the space needed to store the matrices, but can help speed up matrix operations such as Gaussian elimination, making this of much importance in many engineering applications. Other applications of the Bandwidth Minimization problem are given in [8, 3].

In 1976, this problem was shown to be NP-hard for general graphs by Papadimitriou [14]. Subsequent work strengthened the hardness result to trees with maximum degree 3, and to caterpillars of hair-length at most 3 [5, 13], making this one of the few problems known to be hard even when the input graphs are trees of a very simple form. Furthermore, it has also been shown that the Bandwidth Minimization problem is hard to even approximate on trees. In fact, it is NP-hard to approximate it to within *any* constant even when the input graph is a caterpillar of maximum degree 3 [16].

On the positive side, approximation algorithms were known only for special classes of trees and for asteroidal-triple free graphs [8, 7, 9] until 1998, when Feige developed a $O(\log^{4.5} n)$-approximation algorithm for general graphs [3], and independently, Blum et al. [2] obtained $O(\sqrt{n/b}\log n)$-approximation algorithms, where b is the bandwidth of the input graph. Both these algorithms are based on the idea of obtaining a "nice" embedding of the input graph into Euclidean space and then projecting down onto a random line.

The embeddings used in [3] were called *volume respecting embeddings.* Subsequent improvements on the bandwidth problem have been achieved by displaying better volume respecting embeddings: Feige [4] gave a better analysis of his volume respecting embeddings to improve the approximation guarantee to $O(\log^{3.5} n\sqrt{\log\log n})$ for general graphs, and independently, Rao [15] obtained improved volume respecting embeddings for planar graphs and Euclidean graphs, which gave even better guarantees of $O(\log^3 n)$ and $O(\log^3 n \log k)$ respectively for the Bandwidth Minimization problem on those graphs. This $O(\log^3 n)$ guarantee is also the best known result for trees, which are *a fortiori* planar.

*Supported by NSF grants CCR-9505448 and CCR-9820951.

The algorithm presented in this paper tries to circumvent the step of getting a good volume-respecting embedding[1], and is much simpler by comparison: it assigns random lengths to the edges of the tree, and places the vertices in order of their resulting distance from some arbitrarily chosen vertex. We show that this algorithm approximates the minimum bandwidth of the input tree T to within a factor of $O(\log^2 n\sqrt{\kappa(T)})$, where $\kappa(T)$ is the *caterpillar dimension* of the tree T [2]. Since it can be shown that the caterpillar dimension of any n-vertex tree is at most $O(\log n)$, the performance of our algorithm is always within $O(\log^{2.5} n)$ of optimal.

We note in passing that better approximation algorithms are known for some special classes of trees: deterministic $O(\log n)$-approximations are known for caterpillars [8] and some generalizations of height-balanced trees [7]. Though our general analysis only gives an $O(\log^2 n)$-approximation guarantee for caterpillars, we can, by a more specialized argument, show that our algorithm is in fact an $O(\log n)$-approximation algorithms when the input is a caterpillar.

The rest of the paper is organized as follows: in section 2, we fix some notation and definitions. In section 3, we describe the approximation algorithm, which we analyze in section 4. The analysis for the $O(\log n)$-approximation guarantee for caterpillars is presented in section 5. Finally, more information about the proof of a crucial concentration bound used in the analysis is given in Appendix A.

2 Some definitions

We consider a tree $T = (V, E)$ with n vertices and l leaves, which we root at an arbitrary vertex r. This imposes an ancestor-descendent relationship on the vertex set of T. We shall also assume that a vertex is its own ancestor. Finally, let $d(u, v)$ be the number of edges in the unique path between u and v in T.

The *caterpillar dimension* [12, 11] of a rooted tree T, henceforth denoted by $\kappa(T)$, is defined thus: For a tree with a single vertex, $\kappa(T) = 0$. Else, $\kappa(T) \leq k + 1$ if there exist paths $P_1, P_2, \ldots, P_k$ beginning at the root and pairwise edge-disjoint such that each component T_j of $T - E(P_1) - E(P_2) - \ldots - E(P_k)$ has $\kappa(T_j) \leq k$, where $T - E(P_1) - E(P_2) - \ldots - E(P_k)$ denotes the tree T with the edges of the P_i's removed, and the components T_j are rooted at the unique vertex lying on some P_i. The collection of edge-disjoint paths in the above recursive definition form a partition of E, and are called the *caterpillar decomposition* of T. It is simple to see that the unique path between any two vertices of T intersects at most $2\kappa(T)$ of these paths. It can also be shown that $\kappa(T)$ is at most $\log l$, and that a decomposition with the minimum value of $\kappa(T)$ can be computed in polynomial time by dynamic programming (see, e.g., [12]). Furthermore, if time is at a premium, it is possible to compute a (possibly suboptimal) decomposition of value $O(\log n)$ in linear time.

We assign the vertices of the tree to paths in the caterpillar decomposition in the following manner: a vertex v belongs to the path P if the edge connecting v to its parent belongs to P. The root vertex r is arbitrarily assigned to one of the paths of its children. This also allows us to impose an ordering on the children of a vertex v: the child w which lies on the same path as v is defined to be its leftmost child, and its other children are arbitrarily ordered after it.

The *tree volume* $\mathrm{Tvol}(S)$ of a k-point metric S is a product of the lengths of the edges of the minimum spanning tree of S (considered as a graph with the weight of edge (i, j) being $\mu(i, j)$). Hence, if T is any spanning tree of S, the product of its edge lengths of T is at least $\mathrm{Tvol}(S)$.

The *local density* D of a graph G is defined to be $\max_{v\in V} \max_d \lceil |N(v, d)|/2d\rceil$, where $N(v, d)$ is the set of vertices in G at distance at most d from the vertex v. It is easy to see that this is a lower bound on the bandwidth of G.

3 Minimum Bandwidth Approximation for Trees

Let us consider the following simple algorithm for producing a linear arrangement of a rooted tree: let $\phi(r) = 0$, and $\phi(v) = d(r, v)$. Though this is not a one-one map, we can make it so by arranging the set of vertices falling on a particular position in some arbitrary (or random) fashion. Unfortunately, this is a poor algorithm for bandwidth, since there are

[1] It does not quite escape this paradigm: see section 4.1

[2] This quantity, formally defined in section 2, has been previously used in [11, 12, 6] to capture the "complexity" of trees, being, for example, 2 for caterpillars and $\log n$ for the complete binary tree.

simple examples where the bandwidth is about $\sqrt{n}$, while this algorithm gives us $O(n)$. One such example is given in figure 1, where the degrees of vertices a and all the b_i is about $\sqrt{n}$, a is connected to each b_i by a path of length about $\sqrt{n}$, and the children of the b_i are the leaves.

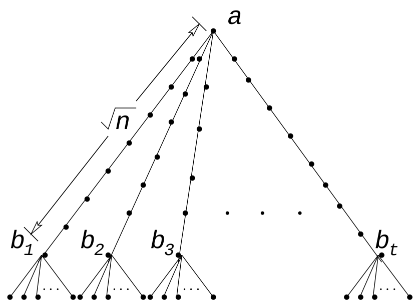

Figure 1: A bad example for the simplest algorithms

A simple twist to the algorithm is to give independent random lengths (say, from the interval $[1, 2]$) to the edges to get a weighted tree (with distance function d'), and embed as above. Unfortunately, this performs very poorly on the same example given above.

Continuing with this idea, we again choose random lengths for the edges, but instead of choosing the length of each edge independently, we fix a caterpillar decomposition of the tree and for any path P in the decomposition, we choose a length in $[1, 2]$ and assign this length to all edges lying on P. The main result of this paper is that this extremely simple algorithm (which clearly runs in linear time, given the caterpillar decomposition) outputs an $O(\log^{2.5} n)$-approximation for the minimum bandwidth problem.

3.1 The Algorithm

Let $T = (V, E)$ be an unweighted undirected tree rooted at r. The algorithm RANDOM-LENGTHS outputs a linear arrangement of T, i.e., a mapping $f : V \to [n]$:

Algorithm RANDOM-LENGTHS

- For each path P_i in caterpillar decomposition, choose a *rate* R_i independently and uniformly in $[1, 2]$. For each edge in P_i, let its length be R_i. Let the distance function using these edge lengths be denoted by d', and let $\phi(v) = d'(r, v)$.
- The map $\phi : V \to \mathbb{R}$ induces a linear order on the vertices in V. The map f is the natural conversion of ϕ into a map from V to $[n]$ thus: $f(i) = j$ if $|\{v \in V \mid \phi(v) < \phi(i)\}| = j$. (Note that this will be a one-one map with probability 1.)

4 The Analysis for Arbitrary Trees

THEOREM 4.1. RANDOM-LENGTHS *is an* $O(\log^2 n\sqrt{\kappa(T)})$*-approximation algorithm for the bandwidth problem on trees.*

Proof of Theorem 4.1: The proof closely follows that given by Feige [3]. The basic structure of his proof is sketched below.

1. Let a *integer interval* be an interval $(a, a+1)$, where a is an integer. For any $S \subseteq V$ with $|S| = k$, show that the chance that $\phi(S)$ falls in some integer interval of unit length (and is called *bad*) is at most $\Gamma^{k-1}/\mathrm{Tvol}(S)$ for some Γ.

   The expected number of bad sets of size k is thus at most $\Gamma^{k-1}\sum_S(1/\mathrm{Tvol}(S))$. By Markov, the total number of bad sets is not more than twice this with probability at least a half.

2. By Theorem 7 of [3], $\sum_S 1/\mathrm{Tvol}(S) \le n(D\log n)^{k-1}$. Substituting this into the previous expression, we get that the number of bad sets is at most $n(D\Gamma\log n)^{k-1}$ with probability half.

3. Now if the bandwidth of f is B, and the length of each edge is at most M, then one of the (at most) $M+1$ integer intervals an edge spans must have had at least $B/(M+1)$ points in it. This implies that we have at least $\binom{(B/M+1)}{k}$ bad sets.

   Choosing $k = \log n$ and simplifying, $B \le D \times M\Gamma\log^2 n$. Since D is a lower bound for the optimal bandwidth, this gives an $O(M\Gamma\log^2 n)$-approximation.

In the previous results, both M and Γ were $O(\sqrt{\log n})$. In lemma 4.1, we will show that the assertion in step 1 holds for our algorithm with $\Gamma = O(\sqrt{\kappa(T)})$. Since the length of any edge is at most $M = 2$, we will get the claimed approximation guarantee of $O(\log^2 n\sqrt{\kappa(T)})$. ∎

Lemma 4.1. *For any set S of k points, the probability that all the points of S fall in some integer interval is bounded above by $O(\sqrt{\kappa(T)})^{k-1}/\mathrm{Tvol}(S)$.*

Before we prove this lemma, let us record a useful result about the distribution of sums of "well spread-out" variables. This is obtained by unraveling a theorem of Leader and Radcliffe [10], the details of which we defer to the appendix.

Lemma 4.2. *Let X_i be independent random variables, where X_i takes a value uniformly from the set $[d_i, 2d_i]$, where $d_i \in \mathbb{Z}_{>0}$. Then there exists a constant $c > 0$ (independent of the d_i's) such that for any unit open interval $I \subseteq \mathbb{R}$,*

$$\mathbf{Pr}\left[\sum_{i=1}^{t} X_i \in I\right] \leq \frac{c}{(\sum_i d_i^2)^{1/2}} \leq \frac{c\sqrt{t}}{(\sum_{i=1}^{t} d_i)}.$$

Proof of Lemma 4.1: Look at the set S of k vertices of the tree T. Note that no two vertices of T which are related to each other can fall into I, and hence we can assume (w.l.o.g.) that S has no ancestor-descendent pairs in it.

An important observation is the following: any two vertices in S must be at *almost* the same distance from their least common ancestor, else S can never be bad. More formally, for any pair $u, v \in S$ with least common ancestor r', it must be the case that $d(v, r')/d(u, r')$ lies between $\frac{1}{2}$ and 2. This follows simply from the fact that edges are being stretched by at most a factor of 2. Thus, if the set S does not satisfy this property, then the vertices of S cannot all lie in the same unit interval. We can thus (again, w.l.o.g.) assume that S satisfies this property.

First, let us give an ordering on the vertices of S. Consider the inorder traversal of the tree T (where the ordering on the children of any node is induced by the caterpillar decomposition, and is defined in section 2), and let $S = \{v_1, v_2, \ldots, v_k\}$, where v_i is visited in this traversal before v_{i+1}. Now let us choose the random rates for the paths in a delayed fashion based on this inorder traversal. The rate for a path is chosen when a vertex belonging to it is visited for the first time. Finally, we choose rates for paths in this way until the position of v_1 is fixed: we now fix I to be the interval $(a, a+1)$ into which v_1 falls, where $a \in \mathbb{Z}$.

We claim that conditional on the rates chosen till v_i is visited, the probability that v_{i+1} also falls into I is $3c\sqrt{\kappa(T)}/d(v_i, v_{i+1})$, where c is the constant in lemma 4.2. This would imply that the chance of S being bad would be bounded by $\Pi_{i=1}^{k-1} 3c\sqrt{\kappa(T)}/d(v_i, v_{i+1})$, which is at most $O(\sqrt{\kappa(T)})^{k-1}/\mathrm{Tvol}(S)$.

To prove the claim, consider the time at which v_i is visited. Let x be the least common ancestor of v_i and v_{i+1}. Clearly, x is distinct from both these vertices, since they both lie in S, and thus are unrelated. Hence it has children y and z which are ancestors of v_i and v_{i+1} respectively, and y lies to the left of z. This implies that x and z lie on different paths, and the path containing z and all the paths that hang off it can not have been seen yet, and hence the paths between x and v_{i+1} must have still not been seen. Further, the total contribution of the paths is $d(x, v_{i+1})$, which by the above observation, is at least $d(x, v_i)/2$, and thus at least $d(v_i, v_{i+1})/3$. Now the position of v_{i+1}, conditional on the events up to when v_i is visited, is a sum of at most $\kappa(T)$ independent random variables X_j, being the contribution of the paths between x and v_{i+1}, and thus lying in some range $[d_j, 2d_j]$, where $\sum d_j = d(x, v_{i+1})$. By lemma 4.2, the chance that v_{i+1} lies in I is at most $c\sqrt{\kappa(T)}/d(x, v_{i+1}) = 3c\sqrt{\kappa(T)}/d(v_i, v_{i+1})$. ■

4.1 Random Projections

Though the above algorithm was described without any reference to the idea of random projections and volume respecting embeddings, it turns out to have a strong connection to volume-respecting embeddings. The distance-preserving embedding given in [11, 12] can be easily seen to also be a *partial* $\kappa(T)$-volume respecting embedding of the tree; being partial in the sense that it only preserves volumes of sets $S \subseteq V$ in which all points lie in different paths of the caterpillar decomposition. Our algorithm can be viewed as projecting such an embedding onto a random vector chosen from the positive orthant so that it is not "too close" to any of the coordinate axes. Loosely speaking, this ensures that sets with more than one point from a single caterpillar path do not create any problems, which gives us the result.

5 Improved Analysis for Caterpillars

In this section, we will show that the algorithm given above has a better performance guarantee when the input tree is a caterpillar.

THEOREM 5.1. *The bandwidth of the output produced by the algorithm* RANDOM-LENGTHS *on caterpillars is an* $\log n$*-approximation to the optimal bandwidth.*

Proof: A Caterpillar is a path P (called the *spine*) with paths $\{L_{ij}\}_j$ (called *hairs*) attached to the vertex $i \in P$, and has caterpillar dimension of at most 2. For the sake of analysis, let us imagine adding paths of length $2n$ to either end of the spine to get a new caterpillar which we shall embed. Note that the local density D' of this modified caterpillar is almost the same as the local density D of the input graph. (In fact, $D' \leq D + 2$.) Hence if we can show that $B \leq O(D' \log n)$, we will be done. W.l.o.g., we can assume that the (extended) spine P is given a unit rate, and thus is isometrically embedded along the real line. Again, it suffices to bound the number of vertices that fall in some unit interval.

Let us focus on the interval $I = [i, i+1]$. Now let us look at the hair L attached to vertex j. If $|L| < \frac{1}{2}|i-j|$, then no vertex from that hair can ever fall into i, else it will have at most one vertex in I. Thus there is a constant c such that the (worst-case) number of vertices from L falling into I is bounded by $n_L = c\sum_{v \in L} 1/d(v_i, v)$, where the v_i is the vertex of P lying at position i. Thus the total number of vertices in I can be bounded by $\sum_L n_L = c\sum_{v \in V} 1/d(v_i, v)$.

It is easy to see that the latter sum is bounded above by $O(D' \log n)$, since there are at most $2D'd$ vertices at distance d, and thus the sum can be bounded above by $2D' \sum_i 1/i$. ■

Acknowledgments

Many thanks to David Aldous, Jaikumar Radhakrishnan and Alex Russell for discussions on the concentration bounds. Thanks also to Ashwin Nayak and Alistair Sinclair for several helpful discussions.

References

[1] Ian Anderson. A variance method in combinatorial number theory. *Glasgow Math. J.*, 10:126–129, 1969.

[2] Avrim Blum, Goran Konjevod, R. Ravi, and Santosh Vempala. Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 100–105, 1998.

[3] Uriel Feige. Approximating the bandwidth via volume respecting embeddings. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 90–99, 1998.

[4] Uriel Feige. Improved analysis of the volume distortion of the random subsets embedding. Manuscript, October 28, 1998.

[5] Michael R. Garey, Ronald L. Graham, David S. Johnson, and Donald E. Knuth. Complexity results for bandwidth minimization. *SIAM J. Appl. Math.*, 34(3):477–495, 1978.

[6] Anupam Gupta. Embedding trees into low dimensional Minkowski spaces. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 694–700, 1999.

[7] J. Haralambides and F. Makedon. Approximation algorithms for the bandwidth minimization problem for a large class of trees. *Theory Comput. Syst.*, 30(1):67–90, 1997.

[8] J. Haralambides, F. Makedon, and B. Monien. Bandwidth minimization: an approximation algorithm for caterpillars. *Math. Systems Theory*, 24(3):169–177, 1991.

[9] Ton Kloks, Deiter Kratsch, and Haiko Müller. Approximating the bandwidth for asteroidal triple-free graphs. In *Proceedings of the ESA '95, LNCS 979*, pages 434–447. Springer, 1995.

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

[11] Nathan Linial, Avner Magen, and Michael Saks. Trees and Euclidean metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 169–177, 1998.

[12] Jiří Matoušek. On embedding trees into uniformly convex Banach spaces. *Israel Journal of Math.*, To appear, 1999. (Czech version in : *Lipschitz distance of metric spaces*, C.Sc. degree thesis, Charles University, 1990).

[13] Burkhard Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM J. Algebraic Discrete Methods*, 7(4):505–512, 1986.

[14] Christos H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, 1976.

[15] Satish B. Rao. Small distortion and volume preserv-

ing embeddings for planar and Euclidean metrics. In *15th Annual ACM Symposium on Computational Geometry*, pages 300–306, 1999.

[16] Walter Unger. 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, 1998.

A Proof of Lemma 4.2

Proof: Let the *concentration* $c(X)$ of a real-valued random variable X be the following:

$$c(X) = \sup_{x \in \mathbb{R}} \mathbf{Pr}[X \in (x, x+1)].$$

The following result due to Leader and Radcliffe tells us how the concentration of a sum of independent random variables behaves in terms of the concentrations of the individual variables.

THEOREM A.1. (LEADER AND RADCLIFFE [10]) *Let X_i be a set of t independent random variables such that the concentration of X_i is at most $1/d_i$, where $d_i \in \mathbb{Z}$. Then the concentration of $\sum_{i=1}^{t} X_i$ is at most the fraction of elements of the poset $M(d_1, \dots, d_t)$ which lie in its largest antichain.*

Here, the poset $M(d_1, d_2, \dots, d_t)$ is the partially ordered set with the base elements from $\Pi_{i=1}^{t}[\mathbf{d_i}]$, and the partial order given by $e \preceq e'$ iff $e_i \leq e'_i$ for all $1 \leq i \leq t$. To estimate the quantity mentioned, we can use the following result:

THEOREM A.2. (ANDERSON [1]) *The number of elements in the largest antichain in $M(d_1, \dots, d_t)$ is*

$$W = \Theta\left(\frac{\prod_{i=1}^{t} d_i}{(\sum_{i=1}^{t}(d_i^2 - 1))^{1/2}}\right). \tag{1.1}$$

As a check, note that if $d_i = 2$ for all i, then this says that the width of the boolean lattice is $\Theta(2^n/\sqrt{n})$, which is indeed true. Combining these results with the fact that the concentration of the X_i random variables is indeed bounded by $1/d_i$ in the statement of lemma 4.2 completes the proof. ∎

In passing, we may mention that the performance of the algorithm is unchanged if we choose discrete rates u.a.r. from the set $\{1, 1+\frac{1}{n}, 1+\frac{2}{n}, \cdots, 2\}$ instead of the continuous interval $[1, 2]$. This can be seen by the fact that the concentration of the random variables is bounded by the same value in the discrete case, and that is the only thing that matters. The mapping may not be one-one any more, but any arbitrary breaking of ties will work just as well.