A Divide and Conquer Algorithm for d-Dimensional Arrangement

Moses Charikar* Konstantin Makarychev*† Yury Makarychev*‡

Princeton University

Abstract

We give an $O(\sqrt{\log n})$-approximation algorithm for d-dimensional arrangement – the problem of mapping a graph to a d-dimensional grid (for constant $d \geq 2$) to minimize the sum of edge lengths. This improves the previous best $O(\log n \log\log n)$ approximation of Even, Naor, Rao and Schieber. The $d = 1$ case is the well studied Minimum Linear Arrangement problem. The problem is equivalent to the question of mapping a graph to integer points on a line so as to minimize the sum of edge costs, where edge costs are measured by edge lengths raised to the exponent $\alpha = 1/d$. We give a simple recursive partitioning algorithm for this variant of linear arrangement for any exponent $\alpha \in (0, 1)$. Our analysis also applies to a directed version of the problem: given a directed graph, the goal is to map vertices to the line so as to minimize the sum of costs of forward edges. As before, edge costs are edge lengths raised to the exponent α. The $\alpha = 0$ case is the well known Minimum Feedback Arc Set problem, and the $\alpha = 1$ case is essentially the Minimum Storage-Time Product problem. We analyze an extremely simple divide and conquer algorithm that uses a balanced cut subroutine with approximation ratio β to recursively partition the graph. Our analysis shows that this approach gives an approximation ratio of $O(\beta)$ for the minimum linear arrangement problem with exponent α for any fixed $\alpha \in (0, 1)$.

1 Introduction

The MINIMUM LINEAR ARRANGEMENT problem is one of the most basic problems amongst geometric ordering and embedding problems. In this paper, we study a d-dimensional generalization of MINIMUM LINEAR ARRANGEMENT where the goal it to map the vertices of a given graph to a d-dimensional grid so as to minimize the sum of edge lengths.

This problem of GRAPH EMBEDDING IN d-DIMENSIONS was originally studied by Hansen [10] in 1989. He obtained an approximation guarantee of $O((\log n)^2)$ for this problem. Later Even, Naor, Rao, and Schieber [5] applied their spreading metric framework to improve the approximation guarantee for the problem to $O(\log n \log\log n)$.

Since then there has been substantial progress in the one dimensional version (i.e. for $d = 1$). Rao and Richa [11] lowered the approximation ratio to $O(\log n)$. Recently Charikar, Hajiaghayi, Karloff, and Rao [4], and independently, Feige and Lee [9] presented a $O(\sqrt{\log n} \log\log n)$-approximation algorithm. However, no progress was made for $d \geq 2$.

Recently Devanur, Khot, Saket, and Vishnoi [6] proved that GRAPH EMBEDDING IN d-DIMENSIONS is hard to approximate to within any constant factor (for every $d \geq 1$) assuming a strengthening of the Unique Games Conjecture.

Even, Naor, Rao and Schieber [5] showed that GRAPH EMBEDDING IN d-DIMENSIONS can be reduced to the following variant of MINIMUM LINEAR ARRANGEMENT.

DEFINITION 1.1. (MINIMUM LINEAR ARRANGEMENT WITH d-DIMENSIONAL COST FUNCTION) *Given a graph $G = (V, E)$ find a linear arrangement $\phi : V \to \{1, \ldots, n\}$ $(n = |V|)$ that minimizes the cost:*

$$\textit{cost} = \sum_{(u,v)\in E} |\phi(u) - \phi(v)|^{\alpha}.$$

where $\alpha = 1/d$.

We will refer to this problem as MINIMUM LINEAR ARRANGEMENT with exponent α and we study it in this paper for arbitrary $\alpha \in (0, 1)$.

The problem has also been studied in the regime $\alpha > 1$. Here it is more natural to consider minimizing the ℓ_p norm of edge lengths. Blum, Konjevod, Ravi, and Vempala [3] developed an $O(\log^2 n)$–approximation algorithm for minimizing the ℓ_2 norm. The problem

*http://www.cs.princeton.edu/~moses/ Supported by NSF ITR grant CCR-0205594, NSF CAREER award CCR-0237113, MSPA-MCS award 0528414, and an Alfred P. Sloan Fellowship.

†http://www.cs.princeton.edu/~kmakaryc/ Supported by a Gordon Wu fellowship.

‡http://www.cs.princeton.edu/~ymakaryc/ Supported by a Gordon Wu fellowship.

of minimizing the ℓ_∞ norm is the well-known MINIMUM BANDWIDTH problem. For this, Dunagan and Vempala [7] gave the current best $O(\log^3 n\sqrt{\log\log n})$-approximation based on Feige's volume respecting embeddings [8].

Our results.

We give an $O(\sqrt{\log n})$-approximation algorithm for MINIMUM LINEAR ARRANGEMENT with exponent α. The key component of the algorithm is the iterated use of an approximation algorithm for BALANCED SEPARATOR. It is this algorithm that determines our approximation guarantee: any improvement to the approximation ratio for BALANCED SEPARATOR would translate to a better approximation guarantee for MINIMUM LINEAR ARRANGEMENT.

Our algorithm is surprisingly simple:

Given a graph G:

1. Find a balanced cut (S, T);
2. Recursively find linear arrangements of S and T (if one of them contains only one vertex then the ordering is trivial);
3. Concatenate the orderings;

A naïve analysis of this algorithm yields an approximation guarantee that is $O(\log n)$ times that of the BALANCED SEPARATOR algorithm used as a subroutine. We give a more sophisticated analysis that avoids this loss of $\log n$. We get the $O(\sqrt{\log n})$ approximation factor from the BALANCED SEPARATOR algorithm of Arora, Rao and Vazirani [2]. Our results show that even using the best known $O(\log n)$ approximation algorithm for BALANCED SEPARATOR prior to [2], it was possible to get an $O(\log n)$ approximation algorithm for these arrangement problems via a very simple approach. By comparison, the $O(\log n \log\log n)$ approximation algorithm of [5] uses a partitioning based on a spreading metric and is much more complex.

One interesting aspect of our algorithm is that it completely ignores the cost function. Consequently, we show that a single ordering is simultaneously a good approximation for all cost functions with exponent $\alpha \in (0, 1)$.

The approximation guarantee depends on α as $O\left(\frac{\sqrt{\log n}}{\alpha(1-\alpha)}\right)$. In the limit, for $\alpha = 1$ our algorithm gives a suboptimal approximation guarantee of $O((\log n)^{3/2})$ (this can be seen by letting $\alpha = 1 - 1/\log n$, so that the cost function is equal up to a constant factor to the cost function for $\alpha = 1$).

The analysis of our algorithm though not hard is not straightforward. As an intermediate step, we introduce a new problem, MINIMUM COST BALANCED HIERARCHICAL DECOMPOSITION. We then show that MINIMUM LINEAR ARRANGEMENT with exponent α can be reduced to this problem. And finally we present an $O(\sqrt{\log n})$-approximation algorithm for MINIMUM COST BALANCED HIERARCHICAL DECOMPOSITION. The combination of this algorithm and the reduction is, in fact, the algorithm for MINIMUM LINEAR ARRANGEMENT we just described.

Directed Case

Our algorithm gives the same $O(\sqrt{\log n})$ approximation guarantee for an analog of MINIMUM LINEAR ARRANGEMENT for directed graphs. We count only forward edges in this version of the problem:

$$cost = \sum_{\substack{(u,v)\in E \\ \phi(u)<\phi(v)}} (\phi(v) - \phi(u))^\alpha.$$

To the best of our knowledge, this problem was not considered before for general α. Note however that the $\alpha = 0$ and $\alpha = 1$ are well known problems. $\alpha = 0$ is the MINIMUM FEEDBACK ARC SET problem and $\alpha = 1$ is essentially the same as MINIMUM STORAGE–TIME PRODUCT (the latter can be reduced to the former).

Our paper shows that somewhat surprisingly the approximation of MINIMUM LINEAR ARRANGEMENT for α strictly between 0 and 1 is very different from the limit cases $\alpha = 0$ and $\alpha = 1$: Given a β approximation for BALANCED SEPARATOR, the simple recursive partitioning algorithm for the standard MINIMUM LINEAR ARRANGEMENT ($\alpha = 1$) and for MINIMUM FEEDBACK ARC SET ($\alpha = 0$) has approximation guarantee $O(\beta \log n)$, whereas our analysis gives a guarantee $O(\beta)$ for $\alpha \in (0, 1)$.

In Section 2 we introduce the MINIMUM COST BALANCED HIERARCHICAL DECOMPOSITION problem, and reduce MINIMUM LINEAR ARRANGEMENT to it. In Section 3, we present and analyze an approximation algorithm for MINIMUM COST BALANCED HIERARCHICAL DECOMPOSITION. In Section 4, based on the results of the previous sections we get the approximation guarantee for MINIMUM LINEAR ARRANGEMENT. Finally, in Section 5, we discuss the directed case.

2 Hierarchical Decomposition

In this section, we introduce MINIMUM COST BALANCED HIERARCHICAL DECOMPOSITION. Then we reduce MINIMUM LINEAR ARRANGEMENT with exponent α to MINIMUM COST BALANCED HIERARCHICAL DECOMPOSITION.

Let us first recall some definitions. A *hierarchical decomposition* P of a graph $G = (V, E)$ is a sequence of partitions $P_0, P_1, \ldots, P_m$ of V s.t.

1) P_0 consists of one cluster: $P_0 = \{V\}$;

2) P_{i+1} is a refinement of P_i;

3) P_m separates every pair of points, i.e. $P_m = \{\{v\} : v \in V\}$.

There is a natural tree structure on the clusters of a hierarchical decomposition: a cluster $A \in P_i$ is the parent of a cluster $B \in P_{i+1}$ if $A \supset B$. We say that the edge (u, v) is cut at level i if the points u and v belong to the same cluster of P_{i-1}, but they belong to distinct clusters of P_i. We will denote this level i by $l(u, v)$.

A hierarchical decomposition P is b-balanced if the size of every cluster in P_i is at most $b^i|V|$.

Given a cost function $c : \mathbb{N} \to \mathbb{R}^+$, we define the cost of the hierarchical decomposition P as follows: the cost incurred by an edge (u, v) equals $c(l(u, v))$; the cost of the decomposition is the total cost incurred by all edges.

Now we are ready to state the problem.

Definition 2.1. (Minimum Cost Balanced Hierarchical Decomposition Problem) *Given a graph G, a cost function c, and a parameter $b < 1$, find a minimum cost b-balanced hierarchical decomposition of G.*

Remark 2.1. *Perhaps a more natural definition of a b-balanced hierarchical decomposition would be to require that every child cluster has at most a b fraction of vertices of the parent cluster. This definition of* Minimum Cost Balanced Hierarchical Decomposition *is essentially equivalent to the definition we gave. However, we use the other definition for the simplicity of the presentation.*

We will now show that Minimum Linear Arrangement can be reduced to Minimum Cost Balanced Hierarchical Decomposition.

Theorem 2.1. *I. There is a randomized polynomial time algorithm that given a linear arrangement of a graph $G = (V, E)$ of cost M w.r.t. the cost function with exponent α and a parameter $b \in (1/2, 1)$, finds a b-balanced hierarchal decomposition of G with the cost function $c(l) = (b^{l-1}|V|)^\alpha$ of cost at most*

$$\frac{2}{b(2b-1)(1-b^{1-\alpha})}M.$$

II. Conversely, there is a polynomial time algorithm that given a b-balanced hierarchal decomposition of cost M produces a linear arrangement of cost at most M.

Remark 2.2. *The condition that $b > 1/2$ is not essential: the same result (with a slightly different dependence on b) holds for $b \in (0, 1)$.*

Proof. I. Denote $n = |V|$. Let $\phi : V \to \{1, \ldots, n\}$ be the linear arrangement of cost M. The algorithm starts with the trivial partition. Then it consecutively subdivides each cluster into two subclusters. Every cluster consists of consecutive elements w.r.t. the ordering ϕ. The algorithm uses the following procedure to subdivide each partition:

Input: A cluster $C = \phi^{-1}(\{x, \ldots, y\})$ at the level l;
Output: Partition of C;

1. If $|C| < b^{l+1}n$ then do not subdivide C (trivial partition), otherwise:
2. Pick a random real number r in the range $[bx + (1-b)y, (1-b)x + by]$.
3. Cut C into two subclusters

$$S = \{v \in C : \phi(v) \le r\}$$

and

$$T = \{v \in C : \phi(v) > r\}.$$

The algorithm stops when every cluster contains one vertex.

First, clearly the algorithm finds a valid b-balanced hierarchical decomposition. Let us estimate its expected cost. Consider an edge (u, v). Denote $d = |\phi(u) - \phi(v)|$. Suppose that u and v belong to the same cluster C at level $l - 1$. If $|C| < b^l n$ then the edge (u, v) is not cut at level l. Otherwise, it is cut with probability

$$\Pr\Big(\phi(u) < r < \phi(v), \text{ or } \phi(v) < r < \phi(u)\Big)$$
$$\le \frac{d}{((1-b)x + by) - (bx + (1-b)y)}$$
$$= \frac{d}{(2b-1)(|C|-1)} \le \frac{2d}{(2b-1)b^l n}.$$

In either case, the edge (u, v) is cut at level l with probability at most

$$\frac{2d}{(2b-1)b^l n}.$$

On the other hand, u and v are separated at level $\lceil \log_b \frac{d}{n} \rceil$ since every cluster contains at most d vertices at this level.

An edge cut at level l contributes $\left(b^{l-1}n\right)^\alpha$ to the cost of the decomposition. So the expected contribution of the edge is at most:

$$\sum_{l=1}^{\lceil \log_b \frac{d}{n} \rceil} \frac{2d}{(2b-1)b^l n} \cdot \left(b^{l-1}n\right)^{\alpha}$$
$$= \frac{2d}{b(2b-1)n^{1-\alpha}} \sum_{l=1}^{\lceil \log_b \frac{d}{n} \rceil} b^{-(1-\alpha)(l-1)}$$
$$\leq \frac{2d}{b(2b-1)n^{1-\alpha}} \cdot \frac{1}{1-b^{1-\alpha}} \cdot \left(\frac{n}{d}\right)^{1-\alpha}$$
$$= \frac{2}{b(2b-1)(1-b^{1-\alpha})} \cdot d^{\alpha}.$$

Finally, summing over all edges we get that the cost of the hierarchical decomposition is at most

$$\frac{2}{b(2b-1)(1-b^{1-\alpha})} M.$$

II. The second part is simple. Given a b-balanced hierarchical decomposition of the graph, we recursively arrange all its clusters (i.e. we perform a depth first search of the tree corresponding to the hierarchical decomposition). Consider an edge (u, v) cut at level l. We will show that its contribution to the cost of the linear arrangement does not exceed its contribution to the cost of the hierarchical decomposition. The vertices u and v belong to a cluster of size at most $b^{l-1}n$. Therefore, the distance between them in the linear order also does not exceed $b^{l-1}n$. So its contribution to the cost is at most $(b^{l-1}n)^{\alpha} = c(l)$. This concludes the proof.

3 Approximation Algorithm for Hierarchical Decomposition

In this section, we present an $O(\sqrt{\log n})$ approximation algorithm for the Minimum Cost Balanced Hierarchical Decomposition problem.

3.1 Balanced Separator Our algorithm relies on a pseudo-approximation algorithm for Balanced Separator of Arora, Rao, and Vazirani [2].

A cut $(S, T = U \setminus S)$ of $H = (U, E_U)$ is a c-balanced separator if both S and T have at least $c|U|$ vertices. The value of the cut is $\frac{E(S,T)}{\min(|S|,|T|)}$ (where $E(S,T)$ is the number of cut edges). In Balanced Separator, we wish to find a minimum value c-balanced cut. Note that since $c|U| \leq \min(|S|, |T|) \leq \frac{|U|}{2}$, the value of a cut is $\Theta(\frac{1}{n})E(S, \bar{S})$. So we state the result we use in terms of the cut size:

Theorem 3.1. (Arora, Rao, and Vazirani [2]) *There is a randomized polynomial time algorithm that finds a c'-balanced cut that cuts $O(\sqrt{\log |U|}\ OPT_c)$ edges, where OPT_c is the size of the minimum c-balanced cut. The constant c' depends only on c.*

We refer to this as a pseudo-approximation since we compare with the optimum c-balanced cut, but allow the algorithm to produce a c'-balanced cut.

Definition 3.1. *We say that a partition of U is b-fine if every cluster has at most $b|U|$ vertices.*

Note that given a b-fine partition we can join some of its clusters and get a $\frac{1-b}{2}$-balanced cut that cuts at most as many edges as the partition. On the other hand, every c'-balanced cut is a $(1 - c')$-fine partition. Therefore, we can use the Balanced Separator Algorithm to get a pseudo-approximation for the minimum b-fine partition.

Corollary 3.1. *There is a randomized polynomial time algorithm that finds a b'-fine cut that cuts $O(\sqrt{\log |U|}\ OPT_b)$ edges, where OPT_b is the minimum of the number of edges cut over all b-fine partitions. The constant b' depends only on b.*

3.2 Algorithm Our algorithm for Minimum Cost Balanced Hierarchical Decomposition will require that the cost function decreases "as fast as a geometric progression". Here is exactly what we need:

Definition 3.2. *A function $c : \mathbb{N} \to \mathbb{R}$ is Λ-rapidly decreasing if for every integer $k \geq 1$*

$$\sum_{i=k-1}^{\infty} c(i) \leq \Lambda c(k).$$

A geometric progression with ratio $r < 1$ is a $\frac{1}{r(1-r)}$-rapidly decreasing function. We will assume below that the cost function is Λ-rapidly decreasing.

Theorem 3.2. *There is a randomized polynomial time algorithm that finds a b-balanced hierarchical decomposition of cost $O(\Lambda\sqrt{\log n}\ OPT)$, where OPT is the cost of the optimum solution.*

We first present the algorithm. The algorithm starts with one cluster. Then it recursively subdivides each cluster into smaller clusters. Each time it uses the pseudo-approximation algorithm for the b-fine partition from Corollary 3.1 as a subroutine to cut each cluster into two clusters. If the algorithm for the fine partition were an approximation algorithm (rather than *pseudo*-approximation), each of these two clusters would be small enough for the next level of the hierarchy (i.e. contain less than $b^{\text{next level}}n$ vertices). However, since the algorithm is a *pseudo*-approximation algorithm we

Figure 1: Approximation Algorithm for Minimum Cost Balanced Hierarchical Decomposition
Input: Graph $G = (V, E)$, parameter b;
Output: Hierarchical decomposition P;

1. **Set** $\mathit{Level} := 0$;
2. **Set** $P_0 := \{V\}$;
3. **Do**
   a. **For each** cluster C in P_{Level} containing more than one vertex **do** $\mathit{SubdivideCluster}(C, \mathit{Level})$;
   b. Proceed to the next level: $\mathit{Level} := \mathit{Level} + 1$;
4. **Until** all clusters at level Level contain one vertex;

Subroutine $\mathit{SubdivideCluster}$
Parameters: Cluster C, current level Level

1. **If** $|C| \le b^{\mathit{Level}+1} n$ **then**
   Add C to $P_{\mathit{Level}+1}$
2. **else**
   a. **Apply** algorithm from Corollary 3.1 to find a b'-fine cut (S, T) of C;
   b. $\mathit{SubdivideCluster}(S, \mathit{Level})$;
   c. $\mathit{SubdivideCluster}(T, \mathit{Level})$;

repeatedly cut each of these clusters (if necessary) till each of them contains at most $b^{\text{next level}} n$ vertices.

The formal description of the algorithm is presented in Figure 1.

It is straightforward that the algorithm returns a valid b-balanced hierarchical decomposition. We will now compute its cost. Denote the optimal decomposition by Q. Let us count the number of edges cut at level $l+1$. Consider a cluster $C \in P_l$. If it contains less then $b^{l+1} n$ vertices we do not cut it. Otherwise, we cut it into several pieces. Note that the partition of C induced by Q_{l+2} is b-fine. Therefore, the cut (S, T) cuts at most $O(\sqrt{\log n})$ times more edges than the partition of C induced by Q_{l+2} does. Consider now all the edges we cut in the first iteration of the recursion of the subroutine $\mathit{SubdivideCluster}$. Their number is at most $O(\sqrt{\log n}\ \mathit{cut}(Q_{l+2}))$, where $\mathit{cut}(Q_{l+2})$ is the number of edges cut in Q_{l+2}. Their cost is $O(\sqrt{\log n}\ c(l+1)\ \mathit{cut}(Q_{l+2}))$. The depth of recursion of $\mathit{SubdivideCluster}$ is at most $\lceil \log_b b' \rceil$ (this quantity depends only on b).

Similarly, at each level of recursion the cost of cut edges is at most $O(\sqrt{\log n}\ c(l+1)\ \mathit{cut}(Q_{l+2}))$. Finally, the cost of all cut edges at level $l+1$ is

$$O\left(\sqrt{\log n}\ \lceil \log_b b' \rceil c(l+1)\ \mathit{cut}(Q_{l+2})\right).$$

The total cost of hierarchical decomposition P is

$$O\left(\sqrt{\log n}\ \sum_{i=0}^{m-1} c(i+1)\ \mathit{cut}(Q_{i+2})\right).$$

Every edge cut in the hierarchical decomposition Q at level l contributes the following to the sum above:

$$O\left(\sqrt{\log n}\ \sum_{i=l-1}^{m} c(i)\right) = O\left(\sqrt{\log n}\ \Lambda c(l)\right).$$

That is, it contributes $O(\sqrt{\log n}\ \Lambda)$ times more than it contributes to the cost of Q. Therefore, the cost of P is at most $O(\sqrt{\log n}\ \Lambda\ \mathit{OPT})$.

4 Approximation Algorithm for Minimum Linear Arrangement

Combining the results of previous two sections we get an approximation algorithm for Minimum Linear Arrangement. One can easily check that this algorithm is the simple algorithm described in the introduction.

Theorem 4.1. *There is a randomized polynomial time approximation algorithm for* Minimum Linear Arrangement *with exponent* α *with the approximation guarantee*

$$O\left(\frac{\sqrt{\log n}}{\alpha(1-\alpha)}\ OPT\right),$$

where OPT *is the value of the optimal solution.*

Proof. We apply algorithm from Theorem 3.2 to get a 2/3-balanced hierarchical decomposition of the graph with the cost function $c(l) = ((2/3)^{l-1}n)^{\alpha}$. The cost of the decomposition is at most

$$O\left(\frac{\sqrt{\log n}}{(2/3)^{\alpha}(1-(2/3)^{\alpha})}\right) = O\left(\frac{\sqrt{\log n}}{\alpha}\right)$$

times more than the cost of the optimal decomposition. On the other hand, by Theorem 2.1 (part I), the cost of the optimal decomposition is at most $O\left(\frac{1}{1-(2/3)^{1-\alpha}}\ OPT\right) = O\left(\frac{1}{1-\alpha}\ OPT\right)$. Finally, we apply the algorithm from Theorem 2.1 (part II) to get a linear arrangement. Its cost is at most

$$O\left(\frac{\sqrt{\log n}}{\alpha}\cdot\frac{1}{1-\alpha}\ OPT\right) = O\left(\frac{\sqrt{\log n}}{\alpha(1-\alpha)}\ OPT\right).$$

5 Directed Version

In this section, we sketch the algorithm for the directed case. The algorithm proceeds as in the undirected case except that it finds a *directed* balanced separator, instead of an (undirected) balanced separator. Recall that the cost of a directed balanced separator (S, T) equals the number of edges going from S to T. To find the balanced separator we use an $O(\sqrt{\log n})$–pseudo-approximation algorithm of Agarwal, Charikar, Makarychev and Makarychev [1].

Our analysis of the undirected case easily generalizes for the directed case. We will now sketch the differences.

We consider an *ordered* hierarchical decomposition. In *ordered* hierarchical decomposition, all clusters at one level are linearly ordered. The ordering of the next level is a refinement of the ordering of the previous level. Only edges going forward contribute to the cost function. The analog of Theorem 2.1 holds for Directed Minimum Linear Arrangement and Directed Minimum Cost Balanced Hierarchical Decomposition. (The same proof as in the undirected case works.)

We say that a b-fine partition is *ordered*, if clusters in the partition are linearly ordered. The cost of the ordered b-fine partition is the number of *forward* edges cut. The algorithm of [1] gives the analog of Corollary 3.1 for ordered b-balanced partitions.

The rest of the analysis follows exactly as in the undirected case.

References

[1] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. *$O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems.* In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 573–581, 2005.

[2] S. Arora, S. Rao, and U. Vazirani. *Expander flows, geometric embeddings and graph partitioning.* In Proceedings of the 36th Annual ACM Symposium on Theory of Computing, pp. 222–231, 2004.

[3] A. Blum, G. Konjevod, R. Ravi, and S. Vempala. *Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems.* Theor. Comput. Sci., 235(1), pp. 25–42, 2000.

[4] M. Charikar, M. Hajiaghayi, H. Karloff, and S. Rao. *ℓ_2^2 spreading metrics for vertex ordering problems.* In Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms, pp. 1018–1027, 2006.

[5] G. Even, J. Naor, S. Rao, and B. Schieber. *Divide-and-conquer approximation algorithms via spreading metrics.* Journal of the ACM, 47(4), pp. 585-616, 2000.

[6] N. Devanur, S. Khot, R. Saket, N. Vishnoi. *On the Hardness of Minimum Linear Arrangement.* Manuscript, 2006.

[7] J. Dunagan and S. Vempala, *On Euclidean Embeddings and Bandwidth Minimization.* In Proceedings of the 5th Workshop on Randomization and Approximation, pp. 229–240, 2001.

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

[9] U. Feige and J. Lee. *An improved approximation ratio for the minimum linear arrangement problem.* Manuscript, 2005.

[10] M. D. Hansen. *Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems.* In Proceedings of the 30th Annual Symposium on Foundations of Computer Science, pp. 604-610, 1989.

[11] S. Rao and A. Richa. *New approximation techniques for some linear ordering problems.* SIAM Journal of Computing, 34(2), pp. 388-404, 2004.