Random Projection: a new approach to VLSI layout

Santosh Vempala
Department of Mathematics and
Laboratory for Computer Science,
M.I.T., Cambridge, MA 02139.
Email: vempala@math.mit.edu

Abstract

We show that random projection*, the technique of projecting a set of points to a randomly chosen low-dimensional subspace, can be used to solve problems in VLSI layout. Specifically, for the problem of laying out a graph on a 2-dimensional grid so as to minimize the maximum edge length, we obtain an* $O(\log^{3.5} n)$ *approximation algorithm (this is the first* $o(n)$ *approximation), and for the bicriteria problem of minimizing the total edge length while keeping the maximum length bounded, we obtain an* $O(\log^3 n, \log^{3.5} n)$ *approximation. Our algorithms also work for* d*-dimensional versions of these problems (for any fixed* d*) with polylog approximation guarantees. Besides random projection, the main components of the algorithms are a linear programming relaxation, and volume-respecting Euclidean embeddings (introduced by Feige).*

1 Introduction

Let the edges of an undirected graph represent connections to be made in the design of a computer chip. Then we would like to place the vertices of the graph on the plane so that the length of connections (wire length) is as small as possible. Of course, the vertices cannot all be very close to each other, so we restrict them to lie on a two-dimensional grid. This is indeed a fundamental framework in VLSI layout. Formally, we have the following two problems:

Given an undirected graph $G = (V, E)$*, place the vertices of* G *on a 2-dimensional grid in such a way as to minimize the* **total** *length of the edges.*

Given an undirected graph $G = (V, E)$*, place the vertices of* G *on a 2-dimensional grid in such a way as to minimize the* **maximum** *length of the edges.*

Above, the length of an edge is the distance between its endpoints on the grid. This could be measured as Euclidean distance or alternatively in the L_1 metric (i.e., wires go along the grid lines).

The complexity of these problems has been widely investigated. It is known that for both problems, finding an exact optimum solution is NP-hard [8, 9]. Starting with the work of Leighton and Rao [8], many researchers have designed *approximation* algorithms for the *Minimum Total Edge Length* problem with poly-logarithmic approximation factors [3]. The *Minimum Maximum Edge Length* problem, however, has thus far eluded any reasonable approximation. All the existing algorithms for the 2-dimensional case (for the total length) are based on the idea of finding a small, balanced separator of the graph (and using this recursively), an idea that goes back to the work of Leighton and Rao. In this paper, we present randomized approximation algorithms for *both* problems based on a fundamentally different approach. Our algorithms are inspired by the recent breakthrough for the 1-dimensional version of the Minimum Maximum Edge Length problem, called the Minimum Bandwidth [1, 4]. Our main tools are a linear programming relaxation, Euclidean embeddings of metric spaces, and *random projection*. The latter has recently enjoyed much success as an algorithmic tool [6, 10, 1, 4].

Our algorithms work for d-dimensional versions of these problems, for fixed d. In d dimensions, we obtain an $O((\log n)^{2.5+\frac{2}{d}})$ approximation for the maximum edge length problem (i.e., embedding a given graph on a d-dimensional grid). For the important 2-dimensional case, this is an $O(\log^{3.5} n)$ approximation, and represents the first $o(n)$ approximation for the problem. For the total length problem we get an approximation factor of $O((\log n)^{2+\frac{2}{d}})$, which is $O(\log^3 n)$ in 2-dimensions (and is not as good as the best-known). However, what is interesting about this latter result is that we derive it using a *nearly identical* algorithm. Also, as a consequence, we can solve the bicriteria problem of minimizing the total edge length while keeping the maximum edge length bounded. We can also approximate (to polylog factors) the edge length under other norms (besides L_1 which is the usual total length and L_∞ which is the maximum length), using the same approach.

We describe our algorithm via the following "combination" problem: Given a graph $G = (V, E)$, and a number b, find a layout of G on a 2-dimensional grid, so that the length of any edge of G is at most b in the layout, and the total length of the edges is as small as possible. We will refer to this problem as simply the VLSI problem. The main result is a bicriteria approximation algorithm which approximates the total length to within $\log^3 n$ while keeping the maximum length at most $b \log^{3.5} n$. From the derivation of this result, it is not hard to see all the results claimed in the previous paragraph.

The algorithm has 3 main steps. The first step is solving a linear programming relaxation to find estimates of the edge lengths in an optimal solution. The second step is an embedding of the vertices of G in a Euclidean space with two properties: (1) the distances are approximately equal to the edge lengths found by the linear program, (2) for subsets of size up to $\log n$, the volume of a subset of vertices S is approximately equal to the product of the edge lengths in a minimum spanning tree for S. The existence of such low-distortion, *volume-respecting*, embeddings was shown by Feige [4], and we directly use his algorithm for finding such an embedding. The last step (which may be viewed as the "rounding step") consists of picking a random d-dimensional plane and projecting the vertices orthogonally to it. The projections of the vertices are rounded off to the grid points of a d-dimensional grid to get the final answer.

The linear programming relaxation relies on *spreading constraints* which are designed to restrict the edge lengths to resemble those along a grid. These constraints were proposed in [1, 3]. Euclidean embeddings with property (1) have proven to be a useful algorithmic tool [7]. The (perhaps) mysterious property (2) allows us to bound the higher moments of random variables in the random projection step. An embedding that satisfies (1) and (2) can also be found by a semi-definite program that explicitly states these properties as constraints. However, such a program might take quasi-polynomial time, and hence we use Feige's algorithm.

The contribution of this paper can be summed up as follows: (1) The observation that *any* metric which satisfies spreading constraints is amenable to volume-respecting embeddings, (2) Random projection (which was previously used for 1-dimensional problems) works for 2-dimensional problems also (and, in fact, for d-dimensional problems, where d is fixed).

The rest of the paper is organized as follows. In the next section we describe the relaxation, show that it can be solved in polytime, and derive a useful property of its solution. Then we recall volume-respecting embeddings and state the main theorem about the existence of low-distortion embeddings. Following that, we describe the algorithm. The subsequent section is devoted to analysis. The final section gives some additional results for VLSI problems and an important open problem.

2 The Relaxation

The idea of the relaxation is the following. Consider a 2-dimensional layout problem. We would like to find an embedding of the vertices of a given graph on a 2-dimensional grid, while minimizing some property of the layout. Suppose we relax the condition that the points should lie on a 2-dimensional grid to just that they should be "spread out" as if on a 2-dimensional grid. That is, in any region of diameter R, there should be no more than $O(R^2)$ points. Similarly we say that a set of points are spread out as if on a 1-dimensional grid (i.e., line) if in any region of diameter R there are no more than $O(R)$ points. This restriction can be modelled by the following set of *spreading constraints*. We state below the spreading constraints for d-dimensional grids (although the interesting consequences we know of are only for $d = 1, 2$). Spreading constraints were used in [3] and also in [1].

$$\sum_{j \in S} x_{ij} \geq |S|^{1+\frac{1}{d}} \qquad \forall S \subset V, \forall i \in V$$

In words, these constraints imply that the average distance of any set of k points from any given point i is $\Omega(k^{\frac{1}{d}})$. Hence, in any region of radius R, there are no more than $O(R^d)$ points. The relaxation for the VLSI problem is then:

$$\begin{array}{llll}
\min & \sum_{ij \in E} x_{ij} & & \\
x_{ij} & \geq \; 0 & \forall i, j \in V & (1) \\
x_{ij} & \leq \; n^{\frac{1}{d}} & \forall i, j \in V & (2) \\
x_{ij} & \leq \; b & \forall ij \in E & (3) \\
\sum_{j \in S} x_{ij} & \geq \; |S|^{1+\frac{1}{d}} & \forall S \subset V, \forall i \in V & (4) \\
& x_{ij} \text{ satisfy the triangle inequality} & & (5)
\end{array}$$

That is, we are trying to find a metric with minimum total edge length and bounded edge length while satisfying the spreading constraints. It can be easily verified that points on a d-dimensional grid satisfy the constraints of the program (to within a constant factor), and hence the optimal value of the program is a lower bound on the true OPT. The feasible region of this relaxation is a convex set in $\boldsymbol{R}^{n^2}$. Although there are exponentially many constraints in this linear program, we can solve it in polynomial time by constructing an efficient (poly-time) *separation oracle* and appealing to the classic GLS theorem [5], based on Ellipsoid algorithm, which says that separation is poly-time equivalent to optimization. A separation oracle for $d = 1$ appeared in [1].

Theorem 1 *The linear programming relaxation can be solved in polynomial time.*

Proof. A separation oracle for this program takes as input a set of lengths $\{x_{ij}\}$ and checks if this set of lengths satisfies all the constraints. If yes, it simply says "yes"; otherwise it reports one constraint that is violated by the input set of edge lengths.

Constraints (1),(2), (3),(5) can all be explicitly checked as they are only polynomially many. To check constraints (4), for each vertex we sort the distances to all the other vertices, and check for each k, $1 \leq k \leq n$, whether the sum of the distances to the subset consisting of the nearest k vertices satisfies the corresponding constraint. This is equivalent to checking all the constraints, since if a set violates the constraints, then the set with smallest edge weights also violates them. Hence the separation oracle takes only polynomial time on any given input. (In all the above if we find a violated constraint, we stop and report that). □

Let $\{x_{ij}\}$ be a solution to the linear program. The following lemma is immediate from the spreading constraints:

Lemma 1 *For any vertex i, and any subset of vertices S,*

$$\sum_{j \in S} \frac{1}{(x_{ij})^d} \leq \log n$$

For each subset S of vertices of G, choose a minimum spanning tree T_S and let the *tree volume* of S, $Tvol(S)$, be the product of the lengths of the edges of T. The next lemma about tree volumes is a consequence of the spreading constraints, and is one of the key observations in this paper. It will be used later in the analysis.

Lemma 2

$$\sum_{S \subset V, |S|=k} \frac{1}{Tvol(S)^d} \leq n(\log n)^{k-1}$$

Proof.

$$\sum_{S \subset V, |S|=k} \frac{1}{Tvol(S)^d} \leq$$

$$\sum_{i_1 \in V} \sum_{i_2 \in V \setminus \{i_1\}} \cdots \sum_{i_k \in V \setminus \{i_1, \ldots, i_{k-1}\}} \frac{1}{(x_{i_1 i_2} x_{i_2 i_3} \ldots x_{i_{k-1} i_k})^d}$$

$$= \sum_{i_1, i_2} \frac{1}{(x_{i_1 i_2})^d} \sum_{i_3} \frac{1}{(x_{i_2, i_3})^d} \cdots \sum_{i_k} \frac{1}{(x_{i_{k-1} i_k})^d}$$

We will now upper bound this sum using the spreading constraints (4). Consider the innermost sum in the above expression, and let $S' = \{i_1, i_2, \ldots, i_{k-1}\}$. Using lemma 1 on the set $V \setminus S'$,

$$\sum_{i_k \in V \setminus S'} \frac{1}{(x_{i_{k-1} i_k})^d} \quad \leq \quad \log n$$

By applying this inequality repeatedly, we have

$$\begin{aligned} \sum_{S \subset V, |S|=k} \frac{1}{Tvol(S)^d} &\leq (\log n)^{k-2} \sum_{i_1, i_2} \frac{1}{(x_{i_1 i_2})^d} \\ &\leq (\log n)^{k-2} \sum_{i_1 \in V} \sum_{i_2 \in V \setminus \{i_1\}} \frac{1}{(x_{i_1 i_2})^d} \\ &\leq n(\log n)^{k-1} \end{aligned}$$

□

3 Euclidean low-distortion embeddings

In this section we recall the definition and properties of low-distortion volume-respecting embeddings from the work of Feige [4]. The goal is to embed a finite metric space (a complete edge-weighted graph) in a Euclidean space so that the Euclidean embedding preserves various properties of the metric. The remarkable observation that there exist embeddings that preserve edge lengths (distances) with only a logarithmic distortion was first made by Bourgain [2]. Feige extended these embeddings to volume-respecting embeddings.

For a set of points S, $|S| = k$, in $\boldsymbol{R}^n$, for any $n \geq k-1$, the Euclidean volume $Evol(S)$ is defined to be the volume of the $(k-1)$-dimensional volume of the simplex formed by the points in S. For a metric space on a set of vertices S, such that $|S| = k$, $Vol(S)$ is defined to be the maximum $Evol(\phi(S))$ over all non-expansive mappings ϕ from S to $\boldsymbol{R}^k$. The *distortion* of an embedding ϕ is the ratio

$$\left(\frac{Vol(S)}{Evol(\phi(S))} \right)^{\frac{1}{k-1}}$$

Given a graph G, an embedding is called (k, c)-volume respecting if for all subsets of at most k vertices of G, the distortion of each subset is no more than c.

Theorem 2 (Feige) *For any connected graph G, and any k, a $(k, \sqrt{k}\log^{\frac{3}{2}} n)$-volume respecting embedding can be found in polynomial time.*

Note that edge distances are only allowed to *decrease* in such an embedding. Feige also proved a bound on the volume of metric space in terms of its edge lengths. Recall that $Tvol(S)$ is the product of the edge lengths in a minimum spanning tree of S.

Lemma 3 *For a metric space S with $|S| = k$,*

$$Tvol(S) \leq 2^k (k-1)! Vol(S)$$

4 The Algorithm

The input to the algorithm is a graph $G = (V, E)$ and a bound b on the maximum edge length allowed. In the description below, by the *width* of a grid we mean the length of a side of a single cell of the grid.

1. Solve the relaxation for G, with b as the upper bound on each edge length, to get a set of lengths x_{ij}, for all $i, j \in V$.
2. Find a $(\log n, \log^2 n)$-volume-respecting Euclidean embedding $\{u_1, u_2 \ldots, u_n\} \subset \boldsymbol{R}^n$ for the vertices of G with edge lengths x_{ij}.
3. Choose d random lines $\ell_1, \ldots, \ell_d$, (independently) passing through the origin (random unit vectors).
4. Project $u_1, \ldots, u_n$ to each of the lines to get an embedding on a d-dimensional plane. So u_i is mapped to $(u_i \cdot \ell_1, u_i \cdot \ell_2, \ldots, u_i \cdot \ell_d)$.
5. Round the embedding to a d-dimensional grid. This is done in two steps, first simply enclose the projected points in a grid of width $\frac{1}{\sqrt{n}}$ (i.e., each cell of the grid is a d-dimensional cube of side $\frac{1}{\sqrt{n}}$). Let the maximum number of points in any single cell of the grid be M. Then subdivide the grid by placing within each cell a subgrid with M cells, i.e., a subgrid of width $\frac{1}{\sqrt{n}M^{\frac{1}{d}}}$. Now assign points that fall in an outer level grid cell arbitrarily to the grid points of the final grid within that cell, and scale up the grid by a factor $\sqrt{n}M^{\frac{1}{d}}$ along each dimension (so that the final width is 1).

5 The Approximation Guarantees

Let us assume without loss of generality that the Euclidean embedding is in $\boldsymbol{R}^n$. The analysis relies on the following two high-level observations. The first is that on projection to a random line in $\boldsymbol{R}^n$, the lengths scale by roughly $\frac{1}{\sqrt{n}}$, and are concentrated around this expectation. The second is that the probability that a set of k points falls in a small interval on the random line is inversely proportional to the $(k-1)$-dimensional volume of the simplex formed by the k points. Below, we state these observations formally. Lemma 4 is folklore by now. A proof of lemma 5 appeared in [4]; we give an alternate proof here.

Lemma 4 *Let $v \in \boldsymbol{R}^n$. For a random unit vector ℓ,*

$$\Pr\left[|v \cdot \ell| \leq \frac{c}{\sqrt{n}}|v|\right] \geq 1 - e^{-c^2/4}.$$
$$\Pr\left[|v \cdot \ell| \leq \frac{1}{c\sqrt{n}}|v|\right] = O\left(\frac{1}{c}\right).$$

Lemma 5 *Let S be a set of vectors $v_1, \ldots, v_k \in \boldsymbol{R}^n$. For a random unit vector ℓ*

$$\Pr\left[max_i\{v_i \cdot \ell\} - min_i\{v_i \cdot \ell\} \leq W\right] = O\left(\frac{W^{k-1} n^{\frac{k-1}{2}}}{(k-1)! Evol(S)}\right)$$

Proof. Fix an order on the set of points (or vectors) in S, say $v_1, v_2, \ldots, v_k$. Assume that the $(k-1)$-dimensional simplex formed by these points has non-zero volume (otherwise, the theorem is trivially true). From this simplex we can define the following orthogonal set of line segments. Start with the line from v_1 to v_2, call this line r_1. Let r_2 be the perpendicular line from r_1 to v_3. In general, let r_i be the perpendicular line from the hyperplane formed by $v_1, \ldots, v_i$ to v_{i+1}. Then

$$Evol(S) = \frac{length(r_1) length(r_2) \ldots length(r_{k-1})}{(k-1)!}$$

In order for the k points to all fall in an interval of width W, it is necessary for each line r_i, $i = 1, \ldots, k-1$, to fall in an interval of width W. We prove the theorem by bounding from above the probability of the latter event.

From Lemma 4, the probability that any single r_i falls in an interval of width less than W can be upper bounded.

$$\Pr[|r_i \cdot \ell| \leq W] = O\left(\frac{W\sqrt{n}}{length(r_i)}\right)$$

If the lengths of the projections of the r_i's were all independent, then by multiplying these probabilities, we could complete the proof. They are, of course, not independent. But, they are orthogonal, and this is good enough to prove the theorem.

We outline the rest of the proof: consider the unit sphere in R^n. The unit vector ℓ is chosen uniformly at random from it. The probability that a single r_i falls in an interval of width W is exactly the $(n-1)$-dimensional surface area of a "band" of width $\frac{W}{length(r_i)}$ on the unit sphere, which is orthogonal to r_i (any unit vector outside this band would cause r_i to have a projection larger than W). The area of this band can be upper bounded as an $O(\frac{W\sqrt{n}}{length(r_i)})$ fraction of the area of the sphere, and hence the claimed probability. Now for each r_i, we have one such band. The probability we wish to estimate is exactly the area of the intersection of these bands. Since the r_i's are orthogonal, so are the bands, and hence it is not hard to upper bound area of the intersection (using the formula for the area of a sphere). This gives us the bound in the theorem. □

Consider the embedding obtained after step 2 of the algorithm. We can bound the maximum number of points that fall in a single cell of the outer grid in step 5 of the algorithm.

Lemma 6 *The number of vertices that fall in any outer grid cell is $O(\log^{2d+2} n)$.*

Proof. Consider a single grid cell C. Along each dimension it has size $W = \frac{1}{\sqrt{n}}$. Throughout the proof we use $k = \log n$ for convenience.

For a subset S of vectors in the embedding after step 2, let X_S^i be a random variable that is 1 if all the vectors in S fall in the cell C along ℓ_i. In order for S to completely fall in C, we need each of $X_S^1, X_S^2, \ldots, X_S^d$ to be 1. Let Y_S be the random variable that indicates this. Then

$$Y_S = X_S^1 X_S^2 \ldots X_S^d$$

Let us now bound the total number of sets S of size k that fall in cell C. This is

$$\sum_{S \subset V, |S|=k} Y_j$$

By linearity of expectation, and the independence of the random variables $X_S^1, X_S^2, \ldots, X_S^d$ (since the lines $\ell_1, \ldots, \ell_d$ were chosen independently), we have

$$\begin{aligned}
\mathbf{E}\left[\sum_{|S|=k} Y_S\right] &= \sum_{|S|=k} \mathbf{E}\left[X_S^1 X_S^2 \ldots X_S^d\right] \\
&= \sum_{|S|=k} \mathbf{E}[X_S^1]\mathbf{E}[X_S^2] \ldots \mathbf{E}[X_S^d] \\
&= \sum_{|S|=k} \mathbf{E}[X_S^1]^d
\end{aligned}$$

The last step above is from the observation that the the X_S^i's are identically distributed. We now rewrite this in terms of probability.

$$\begin{aligned}
\sum_{|S|=k} \mathbf{E}[X_S^1]^d &\leq \sum_{|S|=k} \Pr(X_S^1 = 1)^d \\
&\leq \sum_{|S|=k} \left(\frac{W^k n^{\frac{k}{2}}}{(k-1)! Evol(S)}\right)^d && (6) \\
&\leq \sum_{|S|=k} \frac{1}{[(k-1)! Evol(S)]^d} \\
&\leq \sum_{|S|=k} \left(\frac{(\log^2 n)^k}{(k-1)! Vol(S)}\right)^d && (7) \\
&\leq \sum_{|S|=k} \left(\frac{(\log^2 n)^k 2^k (k-1)!}{(k-1)! Tvol(S)}\right)^d && (8) \\
&\leq (2\log^2 n)^{kd} \sum_{|S|=k} \frac{1}{Tvol(S)^d} \\
&\leq (2\log^2 n)^{kd} n (\log n)^k && (9) \\
&\leq (4\log^{2d+1} n)^k
\end{aligned}$$

Above, (6) is from Lemma 5, (7) follows from Theorem 2, (8) is by applying Lemma 3, and (9) is from Lemma 2.

With high probability (using Lemma 4 on the vectors of the embedding) the total number of cells in the outer grid is no more than n^2. Hence using Markov's inequality, the maximum number of $\log n$ sized subsets that fall in any one cell of the outer grid is at most $(4\log^{2d+1} n)^k$, with high probability. From this it follows that the maximum number of points in any one outer cell is at most

$$k((4\log^{2d+1} n)^k)^{\frac{1}{k}} = 4\log^{2d+2} n$$

(If there are m elements, then there are at most $\binom{m}{k}$ subsets of size k; inversely, if there are N subsets of size k the number of elements is at most $kN^{\frac{1}{k}}$). □

So in two dimensions, after the first rounding step, each cell has $O(\log^6 n)$ points. We are now ready to bound the maximum edge length and the total edge length.

5.1 Maximum length

At the end of the algorithm let $z_1, z_2, \ldots, z_n$, be the final positions of the vertices $1, \ldots, n$. Note that z_i's are grid points of a grid of unit width.

Theorem 3 *For a d-dimensional problem, with high prob-*

ability, for every edge ij,

$$|z_i - z_j| = O\left(x_{ij}(\log n)^{2.5+\frac{2}{d}}\right)$$

The maximum edge length is $O(b(\log n)^{2.5+\frac{2}{d}})$.

In two dimensions this implies an approximation factor of $O(\log^{3.5} n)$ for the maximum edge length.

Proof. After step 2 of the algorithm, for each edge ij,

$$|u_i - u_j| \leq x_{ij}$$

In step 3, by Lemma 4 we can ensure that with high probability for each random line ℓ,

$$|(u_i - u_j) \cdot \ell| = O\left(\frac{|u_i - u_j|\sqrt{\log n}}{\sqrt{n}}\right)$$

Thus each edge ij spans $O(x_{ij}\sqrt{\log n})$ outer grid cells, along each of the d dimensions of the grid. From Lemma 6, each outer grid cell is divided into a total of $O(\log^{2d+2} n)$ cells, or $O((\log n)^{2+\frac{2}{d}})$ along each dimension of the outer grid cell. Hence the total number of cells of the final grid that are spanned by an edge along each dimension is

$O\left(x_{ij}\sqrt{\log n}(\log n)^{2+\frac{2}{d}}\right)$ which is $O\left(x_{ij}(\log n)^{2.5+\frac{2}{d}}\right)$.

The theorem follows for any fixed d. □

5.2 Total length

Theorem 4

$$\mathbf{E}\left[\sum_{ij \in E} |z_i - z_j|\right] = O\left((\log n)^{2+\frac{2}{d}} \sum_{ij \in E} x_{ij}\right)$$

In 2-dimensions, this implies an approximation factor of $O(\log^3 n)$ for the total length while simultaneously bounding the maximum length.

Proof. Below A is some fixed constant.

$$\begin{aligned}
\mathbf{E}\left[\sum_{ij \in E} |z_i - z_j|\right] &= \sum_{ij \in E} \mathbf{E}\left[|z_i - z_j|\right] \\
&\leq \sum_{ij \in E} A\sqrt{n}(\log n)^{2+\frac{2}{d}} \mathbf{E}\left[|(u_i - u_j) \cdot \ell|\right] \\
&\leq \sum_{ij \in E} A\sqrt{n}(\log n)^{2+\frac{2}{d}} \frac{|u_i - u_j|}{\sqrt{n}} \\
&\leq A(\log n)^{2+\frac{2}{d}} \sum_{ij \in E} x_{ij} \\
&= O\left((\log n)^{2+\frac{2}{d}} \sum_{ij \in E} x_{ij}\right)
\end{aligned}$$

□

6 Further work and open questions

An important contention in VLSI layout is the number of wires that pass through a given grid point. In the above problems, we ignored this issue, and so solutions could have many edges routed through the same grid point. Taking wire-widths into account, one can formulate the following problem: Given a graph $G = (V, E)$, place the vertices of G on a 2-dimensional grid so that there is at most one edge of G passing through each grid point and the maximum edge length is as small as possible. While this problem is open, the following decision version can be solved using the techniques in this paper: Given a graph $G = (V, E)$ and a set of edge lengths, determine if it is possible to (approximately) realize this set of lengths on a 2-dimensional grid. That is, can we place the graph on a grid so that all the edges are disjoint, and their lengths are close to the specification?

To solve this problem, subdivide each edge ij by placing along it $\lfloor x_{ij} \rfloor$ new vertices. Let the new graph be G'. Formulate the spreading constraints for G' and check if they are feasible. If the answer is "yes", then by simply embedding G' using our algorithm, we can find the required embedding for G; otherwise, the given set of edge lengths are not realizable. It would be nice if we could use this observation (and possibly some variant of the algorithm presented here) to solve the problem without any prior knowledge of the target edge lengths.

References

[1] A. Blum, G. Konjevod, R. Ravi, and S. Vempala, "Semi-Definite Relaxations for Minimum Bandwidth and other Vertex-Ordering Problems," 1998. to appear in Proc. 30th ACM Symposium on the Theory of Computing.

[2] J. Bourgain, "On Lipshitz embedding of finite metric spaces in Hilbert space," Israel J. Math. 52 (1985) 46-52.

[3] G. Even, J. Naor, S. Rao, and B. Schieber, "Divide-and-Conquer Approximation Algorithms via Spreading Metrics," Proc. 36th Annual Symposium on Foundations of Computer Science, 1995.

[4] U. Feige, "Approximating the bandwidth via volume respecting embeddings," to appear in Proc. 30th ACM Symposium on the Theory of Computing, 1998.

[5] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, 1988.

[6] J. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," Proc. 29th ACM Symposium on Theory of Computing, 1997.

[7] N. Linial, E. London and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," Proc. of 35th Foundations of Computer Science, 1994.

[8] F. T. Leighton and S. Rao, "An approximate max-flow min-cut theorem for uniform multi-commodity flow problems with applications to approximation algorithms," Proc. of the 29th Foundations of Computer Science, 1988.

[9] C. H. Papadimitriou, "The NP-completeness of the bandwidth minimization problem," *Computing*, 16: 263-270, 1976.

[10] S. Vempala, "A Random Sampling based Algorithm for learning the Intersection of Half-spaces," Proc. of the 38th Foundations of Computer Science, 1997.