GOOD QUALITY VIRTUAL REALIZATION OF UNIT DISK GRAPHS

Sriram V. Pemmaraju † Imran A. Pirwani ‡

Abstract. The quality of an embedding $\Phi : V \mapsto \mathbb{R}^2$ of a graph $G = (V, E)$ into the Euclidean plane is the ratio of $\max_{\{u,v\} \in E} ||\Phi(u) - \Phi(v)||_2$ to $\min_{\{u,v\} \notin E} ||\Phi(u) - \Phi(v)||_2$. Given a graph $G = (V, E)$, that is known to be a unit disk graph (UDG), we seek algorithms to compute an embedding $\Phi : V \mapsto \mathbb{R}^2$ of best (smallest) quality. Note that $G$ comes with no associated geometric information and in this setting, related problems such as recognizing if $G$ is a UDG, are NP-hard. While any UDG has a 2-dimensional embedding with quality between $1/4$ and $1$, the adaptation of Vempala’s random projection approach [38] by Kuhn et al. [22] provides the best quality bound of $O(\log^{3.5} n \cdot \sqrt{\log \log n})$.

This paper presents a simple, combinatorial algorithm for computing an $O(\log^3 n)$-quality 2-dimensional embedding of a given graph, that is known to be a UDG. If the embedding is allowed to reside in higher dimensional space, we obtain improved results: a quality-2 embedding in $\mathbb{R}^{O(1)}$. Our key technical contribution is the construction of a “growth-restricted approximation” of the given UDG. Construction of a growth-restricted approximation permits us to bypass the standard and costly technique of solving a linear program with exponentially many “spreading constraints.” As a side effect of our construction, we get the first constant-factor approximation to the minimum clique partition problem for UDGs, given without a geometric representation.

Our problem is a version of the well known localization problem in wireless sensor networks, in which network nodes are required to compute virtual 2-dimensional Euclidean coordinates given little or (as in our case) no geometric information.

1 Introduction

A graph $G = (V, E)$ is a unit disk graph (UDG) if there is an embedding $\Phi : V \mapsto \mathbb{R}^2$ such that $\{u, v\} \in E$ iff $||\Phi(u) - \Phi(v)||_2 \leq 1$. Such an embedding $\Phi$ of $G$ is called a realization of $G$. In this paper, we are interested in the problem of finding a realization $\Phi$ of a given UDG. It is unlikely that this problem has a polynomial-time algorithm because the problem of recognizing if a given graph is a UDG is NP-hard [9]. Aspnes et al. [2] have shown that the problem of computing a realization of a given UDG is NP-hard even if all edge lengths with respect to some (unknown) realization between pairs of neighboring vertices are given.
The problem remains NP-hard when all angles with respect to some (unknown) realization between adjacent edges are known [10] and also when all angles plus slightly noisy, pairwise distances are known [3]. Given these negative results, we consider the problem of computing an “approximate” realization of the given UDG. Let $G = (V, E)$ be a UDG. Let $\Phi : V \mapsto \mathbb{R}^d$ be an embedding of $G$ into $\mathbb{R}^d$. If $G$ is not a clique, then the quality of the embedding $\Phi$ is defined as:

$$\frac{\max_{\{u, v\} \in E} \|\Phi(u) - \Phi(v)\|_2}{\min_{\{u', v'\} \notin E} \|\Phi(u') - \Phi(v')\|_2}$$

In case $G$ is a clique, then the quality of $\Phi$ is simply $\max_{\{u, v\} \in E} \|\Phi(u) - \Phi(v)\|_2$. The specific optimization problem we consider is the following.

Given $d$, and a UDG $G = (V, E)$, find an embedding $\Phi : V \mapsto \mathbb{R}^d$ with best (smallest) quality.

We call this the best quality embedding problem. Of course, we are most interested in instances of the problem when $d \in O(1)$ – particularly when $d = 2$. It is easy to see that every realization into $\mathbb{R}^2$ of every connected, non-clique UDG has quality between $1/4$ and $1$.

**Lemma 1.** Every realization into $\mathbb{R}^2$ of every connected, non-clique UDG has quality between $1/4$ and $1$.

**Proof.** Fix an arbitrary realization $\Phi$ into $\mathbb{R}^2$ of the given UDG. For the upper bound, note that every edge has length at most $1$ in this realization, and every non-edge must have length more than $1$. This yields an upper bound of less than $1$ on the quality of the embedding.

For the lower bound, note that since $G$ is connected and is not a clique, there is at least one triple $u, v, w \in V$ such that $\{u, v\} \notin E$ and $\{v, w\} \in E$ and $\{u, w\} \in E$. Since $u$ and $v$ are non-adjacent, either $\|\Phi(u) - \Phi(w)\|_2 > 1/2$ or $\|\Phi(w) - \Phi(v)\|_2 > 1/2$. Thus, a longest edge has length at least $1/2$ in the realization $\Phi$. Also, since $\{u, w\} \in E$ and $\{w, v\} \in E$, $\|\Phi(u) - \Phi(v)\|_2 \leq 2$, implying that a shortest “non-edge” has length at most $2$. This yields the lower bound. □

This paper focuses on devising an approximation algorithm for the best quality embedding problem.

There exist other reasonable measures of the quality of UDG realization. For example, a measure of the quality of embedding considered in the context of VLSI layout problems [38, 13] is to obtain a realization that minimizes the sum of edge lengths. In the VLSI layout problem, the input is a graph $G$ and the goal is to embed the vertices at distinct grid points in a $d$-dimensional grid so as to optimize some objective function over the set of edges under the embedding; one objective is to minimize the sum of (Euclidean) edge lengths obtained over all feasible embeddings. Analogous to the best quality embedding problem, we could seek to compute $\Phi : V \mapsto \mathbb{R}^d$, for a given UDG $G = (V, E)$ and
dimension $d$, so as to minimize the following objective:

$$\sum_{\{u,v\} \in E} \|\Phi(u) - \Phi(v)\|_2$$

$$\min_{\{u',v'\} \in E} \|\Phi(u') - \Phi(v')\|_2$$

This problem seems to be open at least for $d \in O(1)$ and in this paper, we do not consider optimizing this objective.

Motivated by the localization problem in wireless sensor networks, Moscibroda et al. define quality as a measure of the “goodness” of a realization [28]. This paper [28] claims an $O(\log^{2.5} n \cdot \sqrt{\log \log n})$-quality embedding for UDGs into $\mathbb{R}^2$, but subsequently [22] this bound was corrected to $O(\log^{3.5} n \cdot \sqrt{\log \log n})$. However, Kuhn et al. [22] use a weaker volume respecting embedding bound of $O(\log n \sqrt{\log \log n})$ due to Feige [14]. Using Krauthgamer et al.’s [19] optimal bound of $O(\log n)$ on the volume distortion of general metrics, Kuhn et al.’s [22] bound on the quality of the embedding of a UDG improves to $O(\log^{3.5} n)$. Thus prior to the current paper, $O(\log^{3.5} n)$ was the best quality of a 2-dimensional embedding of an $n$-vertex UDG computable in polynomial time. The current paper improves this bound by a $\sqrt{\log n}$ factor. The techniques employed by [28, 22] are an adaptation of the random projection method devised by Vempala [38] who gave a first polylogarithmic approximation to the VLSI layout problem. The only hardness of approximation result for the best quality embedding problem, that we are aware of, is this [23]: there is no polynomial time algorithm (unless $P = NP$) that can compute a $(\sqrt{3/2} - o(1))$-quality, 2-dimensional embedding of a given UDG in which non-adjacent vertices are required to be more than one unit away from each other.

As mentioned earlier, the best quality embedding problem for UDGs is motivated by the localization problem in wireless sensor networks [11, 30]. Two-dimensional UDGs are typically used to model wireless sensor networks. The localization problem requires each node in a wireless sensor network to compute its own coordinates with respect to some global coordinate system. For most sensing applications, it is critical that each node know its location, at least approximately. Knowledge of location information can also dramatically improve the performance of routing algorithms because it allows the use of geometric routing techniques [8, 16, 17, 25]. One technological solution to the localization problem is to equip each node with a GPS receiver. However, this solution seems too costly, currently and in any case such a solution will have to deal with GPS errors and the fact that GPS service may be unavailable indoors. Another solution is to equip a few “anchor” nodes with GPS receivers [5] and have the rest of the nodes compute their coordinates by using the known coordinates of the anchor nodes. The main drawback of this approach is that for a good quality solution, the number of anchor nodes needed may be fairly high and furthermore they may have to be placed manually by using the known coordinates of the anchor nodes. The main drawback of this approach is that for a good quality solution, the number of anchor nodes needed may be fairly high and furthermore they may have to be placed manually. Recent work has suggested that for geometric routing schemes, having “real” coordinates is not necessary; having virtual coordinates suffices to ensure prompt and guaranteed routing [31, 35]. In fact, virtual coordinates that are derived only from connectivity information are preferable as this information is robust to errors in radio signals. Motivated by the
localization problem, we assume that our input graph is a UDG. While the best quality embedding problem allows the host space to have arbitrary dimensions (specified as $d$ in the input), we are particularly interested in the case when $d = 2$. This is partly because two is the “intrinsic” dimension of a UDG and partly because such results are most relevant to the wireless sensor networks application when $d = 2$. Well-known geometric routing algorithms \cite{8, 16, 25} only provide guaranteed delivery for networks in the plane or ones that lie in a thin slab in $\mathbb{R}^3$ \cite{12}.

2 Overview of Results and Techniques

In this paper, we present a combinatorial algorithm for computing an $O(\log^3 n)$-quality embedding of any UDG into $\mathbb{R}^2$. Our result can be seen as improving Kuhn et al.’s bound \cite{22}. However, the most important aspect of our algorithm is that it is combinatorial. Our algorithm avoids the costly first step of Kuhn et al.’s algorithm in which an exponentially large linear program (LP), that imposes spreading constraints on the vertices, is solved via the ellipsoid method. Starting with an LP or a semi-definite program that imposes spreading constraints is a fairly common approach to solving vertex-ordering problems (such as the minimum bandwidth problem) \cite{7, 13}, and the VLSI layout problem \cite{38}. We avoid the spreading constraints approach via a combinatorial algorithm for computing a “growth-restricted approximation” of the given UDG. This step may be of independent interest. Partitioning a network into clusters so that each cluster has edges to at most a constant number of other clusters, is a standard first step in many topology control and routing algorithms on wireless networks (see \cite{25, 24} for representative examples). Thus far such clustering has been accomplished by the use of Euclidean coordinates or pairwise distances that are assumed to be available at nodes. Our algorithm can perform such a clustering without access to geometric information and can also be implemented efficiently in a distributed setting (in $O(\log^* n)$ rounds under the LOCAL model of computation \cite{32}) as well. Our algorithm has three main steps which we outline in the next three subsections.

2.1 Constructing a Growth-Restricted Approximation

In the first step, we partition the given UDG into cliques such that when the cliques are contracted into vertices, we get a “growth-restricted approximation” of the given UDG. To describe this step more precisely, we need some definitions. For any graph $G = (V, E)$ and for any pair of vertices $u, v \in V$, let $d_G(u, v)$ denote the shortest path distance between $u$ and $v$. Let $B_G(v, r) = \{u \in V \mid d_G(u, v) \leq r\}$ denote the closed ball of radius $r$ centered at $v$ in $G$. Where $G$ is clear from the context, we write $B(v, r)$ instead of $B_G(v, r)$. Define the growth rate of $G$ to be

$$
\rho_G = \inf \{\rho : |B_G(v, r)| \leq r^\rho \text{ for all } v \in V \text{ and all integers } r > 1\}.
$$

A class $\mathcal{G}$ of graphs is growth-restricted if there is some constant $c$ such that for every graph $G$ in $\mathcal{G}$, $\rho_G \leq c$. For any partition $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ of the vertex set $V$ of $G$, the cluster graph of $G$ induced by $\mathcal{C}$, denoted $G[\mathcal{C}]$, is obtained from $G$ by contracting each $C_i$ into a vertex. In Section 3, we show how to partition a given UDG $G = (V, E)$ into cliques
$\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ such that the cluster graph $G[\mathcal{C}]$ induced by the clique partition has constant growth rate. Note that if we can construct an $\alpha$-quality embedding $\Phi$ of $G[\mathcal{C}]$ into $\mathbb{R}^L$, we can immediately get an $\alpha$-quality embedding $\Phi'$ of $G$ into $\mathbb{R}^L$: for each vertex $v$ of $G$ set $\Phi'(v) := \Phi(C_i)$, where $C_i$ is the clique that contains $v$. This allows us to focus on the problem of obtaining a good quality embedding of growth-restricted graphs.

It is quite easy to obtain a clique-partition $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ with the desired properties if a realization of $G$ is given. For example, given a UDG with 2-dimensional coordinates of vertices known, one can place an infinite grid of $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ square cells on the plane and obtain a vertex partition $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ in which each part $C_i$ is all vertices in a cell. Ties caused by points falling on cell boundaries can be broken arbitrarily. Due to the size of the cells, each $C_i$ is a clique, and a simple geometric argument shows that there are $O(r^2)$ cells in the radius-$r$ disk centered at the center of any cell. This suffices to show that in $H := G[\mathcal{C}]$, $|B_H(v, r)| = O(r^2)$, implying that $\rho_H$ is bounded by a constant. See Figure 1 for an illustration.

In the absence of geometric information, it is not immediately clear how to obtain the desired clique partition. One possible approach is to start with a maximal independent set (MIS) $I$ of $G$ and attach each $v \in V \setminus I$ to an arbitrary neighbor in $I$. For each $v \in I$, let $S_v$ denote the set consisting of $v$ along with neighbors which have attached to $v$. Let $\mathcal{S} = \{S_v \mid v \in I\}$ be the induced vertex partition of $V$. Since $I$ is an independent set, in any realization $\Phi$ of $G$ into $\mathbb{R}^2$, $\|\Phi(u) - \Phi(v)\|_2 > 1$ for all $u, v \in I, u \neq v$. From this observation, one can deduce the fact that $H := G[\mathcal{S}]$ has bounded growth rate. However, the sets $S_v$ are not cliques, even though the subgraphs $H_v := G[S_v]$ they induce have diameter at most 2. Furthermore, since we know that each $H_v$ is a graph induced by a neighborhood of a UDG, there exists a partition of $H_v$ into at most 5 cliques. But, in the absence of geometry, how can we find a constant-sized partition of $H_v$ into cliques? In this context, the work of Raghavan and Spinrad [34] is relevant. They present an algorithm that computes a maximum cardinality clique of an input UDG, given without any geometric information; one can use their algorithm as a subroutine in the following greedy approach: repeatedly find and remove a maximum size clique from $H_v$, until it becomes empty. Since we know that $H_v$ can be partitioned into a constant number of cliques, $H_v$ contains a clique of size at least $|S_v|/c$ and therefore each step removes a $1/c$ fraction of vertices (or more) from $H_v$. This immediately implies that the greedy approach produces $O(\log n)$ cliques, where $n = |S_v|$. Unfortunately, this bound is tight and there is a simple example (see Figure 2) showing that the greedy approach can lead to a clique-partition of $S_v$ of size $\Omega(\log n)$.

To further motivate the problem, we note that the problem of partitioning $H_v$ into a constant number of cliques is equivalent to the problem of coloring $\overline{H_v}$ (the complement of $H_v$) with a constant number of colors. In general, computing a constant-sized coloring of a graph, even when it is known to have a coloring of constant size, seems quite hard. For example, the best approximation algorithm for coloring 3-colorable graphs uses $\tilde{O}(n^{3/14})$ colors [6].

In Section 3 we present an algorithm for partitioning each $H_v$ into a constant number of cliques; this involves extending ideas developed by Raghavan and Spinrad [34] in the

---

1 Here the $\tilde{O}$ notation hides factors that are logarithmic in $n$. 

---

JoCG 2(1), 69–91, 2011 73
Figure 1: A realization of a UDG $G$, partitioned by a grid of $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ square cells. The cluster graph $G[C]$ induced by the partition $C$ is also shown.

Figure 2: A bad example for the greedy approach to obtaining a clique partition. Each $A_i$ and each $B_i$ is a set of $2^i$ points. A point in $A_i$ is adjacent to all points in $A_j$, $1 \leq j \leq k$ and exactly the points in $B_i$. The adjacencies for points in $B_i$ are symmetric. Thus the largest clique is $A_k \cup B_k$ and has size $2 \cdot 2^k$. Removing this clique leaves sets $A_j$ and $B_j$, $1 \leq j \leq k-1$ intact. Thus the algorithm uses $k$ cliques to cover about $2^{k+2}$ points. On the other hand two cliques, $\bigcup_{i=1}^{k} A_i$ and $\bigcup_{i=1}^{k} B_i$ suffice to cover all of the points. This yields a lower bound of $\Omega(\log n)$ on the size of the clique partition produced by the greedy algorithm.
context of finding a maximum clique in a UDG with no given realization. Since our problem arises in sensor network localization, it is worth pointing out that the overall construction of the growth-restricted cluster graph induced by cliques has a very simple distributed implementation. The first step of the construction is to compute an MIS of $G$, and a recent paper by Schneider and Wattenhofer [37] is relevant here. They [37] show how to compute an MIS on a class of graphs, bounded growth graphs, that contains UDGs in $O(\log^* n)$ rounds using only connectivity information. Using their distributed algorithm [37], one can compute an MIS of $G$ without any geometry. Following this step, the remaining vertices join a neighboring independent vertex in $O(1)$ rounds, forming neighborhood partition of the vertex set. Next, the partition of neighborhoods, $H_v$, into constant number of cliques can be done again in $O(1)$ rounds since the diameter of each neighborhood is at most 2.

2.2 Constructing Volume Respecting Embeddings

The remaining two steps of our algorithm follow the approach introduced by Vempala [38] with some important differences due to the fact that our input graph is growth-restricted. Let $H := G[C]$ be the cluster graph of $G$ constructed in the previous step.

In the second step of our algorithm, we construct a volume respecting embedding of the shortest path metric of $H$. The notion of volume respecting embeddings was introduced by Feige [14] in the context of the minimum bandwidth problem. Let $(X, d)$ be a metric space. An embedding $\Phi : X \mapsto \mathbb{R}^L$ is a contraction if $\|\Phi(u) - \Phi(v)\|_2 \leq d(u, v)$ for all $u, v \in X$. For a set $T$ of $k$ points in $\mathbb{R}^L$, define $\text{Evol}(T)$ to be the $(k - 1)$-dimensional volume of the $(k - 1)$-dimensional simplex spanned by $T$, computed using the $l_2$ norm. Note that $\text{Evol}(T) = 0$ if $T$ is affinely dependent. For any finite metric space $(X, d)$, define $\text{Vol}(X)$ as $\sup_{\Phi} \text{Evol}(\Phi(X))$, where the supremum is over all contractions $\Phi : X \mapsto \mathbb{R}^{|X| - 1}$.

Given an arbitrary metric space $(X, d)$, a contraction $\Phi : X \mapsto \mathbb{R}^L$ is called $(k, D)$-volume respecting embedding if for every size-$k$ subset $S \subseteq X$, we have:

$$\text{Evol}(\Phi(S)) \geq \left(\frac{\text{Vol}(S)}{D^{k-1}}\right)$$

Note that when $k = 2$, this condition reduces to $\|\Phi(u) - \Phi(v)\|_2 \geq d(u, v)/D$ for all $u, v \in X$. Thus, a volume respecting embedding is a generalization of the more commonly used notion of small distortion embeddings [27], in which only pairs of points are considered. A volume respecting embedding will be very useful for the next step our algorithm, in which a random projection of the vertices of $H$ into $\mathbb{R}^2$, will be performed. For the random projection step to spread points fairly well in $\mathbb{R}^2$, we require sets of points to have large volumes, because intuitively, point sets with large volume will be spread out in their projection into a lower dimensional sub-space. For arbitrary metric spaces, Feige [14] presents a polynomial time algorithm to compute a $(k, O(\sqrt{\log n \cdot \log k + \log n}))$-volume respecting embedding. We are interested in $k = \log n$ and therefore, Feige’s algorithm yields a $(\log n, O(\log n \cdot \sqrt{\log \log n}))$-volume respecting embedding. Krauthgamer et al. [19] give a novel embedding technique which they call measured descent embedding. This embedding improves on Feige’s bound to yield a $(k, O(\log n))$-volume respecting embeddings for $1 \leq k \leq n$ for general metrics [19].
Now we mention a useful lemma due to Feige [14] that establishes a lower bound on \( \text{Vol}(S) \). To show that an embedding \( \Phi : X \to \mathbb{R}^L \) is a \((k, D)\)-volume respecting embedding of \( X \), we need to show that \( \text{Evol}(\Phi(S)) \geq \text{Vol}(S)/D^{k-1} \). \( \text{Vol}(S) \) is difficult to compare against and so Feige defined the notion of a tree volume and showed a lower bound on \( \text{Vol}(S) \) in terms of the tree volume of \( S \). For any \( S \subseteq X \), the tree volume of \( S \), denoted \( \text{Tvol}(S) \), is the product of the edge lengths in a minimum spanning tree of \( S \).

**Lemma 2** (Feige [14]). Let \((X, d)\) be a metric space. For any size-\( k \) subset \( S \subseteq X \), \( \text{Tvol}(S) \leq 2^k(k-1)! \text{Vol}(S) \).

### 2.3 Random Projections and Rounding

The third and last step of our algorithm uses a slightly modified version of Vempala’s random projection and rounding technique. A similar technique is used by Moscibroda et al. [28]. After computing a volume respecting embedding of \( H \), we project the vertices onto a plane defined by two unit vectors chosen uniformly and independently at random. We state two lemmas due to Vempala [38] that show how the random projection step affects individual points and subsets of points, respectively. These lemmas are used in estimating the number of vertices of \( H \) that fall in a region of the plane.

**Lemma 3** (Vempala [38]). Let \( v \in \mathbb{R}^d \). For a unit \( d \)-dimensional vector \( l \) chosen uniformly at random, \( c > 1 \),

\[
\Pr \left[ |v \cdot l| \geq \frac{c}{\sqrt{d}} |v| \right] < \frac{1}{e^{c^2/4}}.
\]

**Lemma 4** (Vempala [38]). Let \( S \) be a set of vectors \( v_1, v_2, \ldots, v_k \in \mathbb{R}^d \). For a unit \( d \)-dimensional vector \( l \) chosen uniformly at random,

\[
\Pr \left[ \max_i \{v_i \cdot l\} - \min_i \{v_i \cdot l\} \leq W \right] = O \left( \frac{W^{k-1} d^{k-1}}{(k-1)! \text{Evol}(S)} \right).
\]

Assuming that \( H \) has a \((k, D)\)-volume respecting embedding for sufficiently small \( D \), we can use Lemma 4 to obtain an upper bound on the number of vertices of \( H \) that get projected onto a small region in the plane. To get sufficient separation among pairs of non-neighboring vertices, we partition the plane using a fine enough grid so that every vertex of \( H \) can be mapped to a distinct grid point. Vempala [38] calls this the “rounding” step. Scaling by a factor proportional to the maximum number of vertices in the small region ensures that the minimum distance between any pair of vertices is at least 1. The fact that not too many vertices were projected into the region, in the first place, ensures that the scaling factor is not too large and allows us to show that edges are not stretched too much. Finally, all vertices in the clique \( C_v \) in \( G \) that correspond to a vertex \( v \) in \( H \) are assigned the point to which \( v \) is mapped. Note that mapping all vertices in a clique in \( G \) to the same point does not affect the quality of embedding.
3 Constructing a Growth-Restricted Approximation

In this section, we show how to construct a clique partition \( C = \{C_1, C_2, \ldots, C_t\} \) of a given UDG \( G \) so that \( G[C] \) has constant growth rate. Recall that \( G[C] \) is the graph obtained from \( G \) by contracting each \( C_i \) into a vertex. As mentioned in the introduction, our starting point is the following algorithm.

\begin{algorithm}
\caption{CLIQUE-PARTITION(\( G \))}
1: Compute a maximal independent set (MIS) \( I \) of \( G \).
2: Associate each vertex \( u \in V \setminus I \) to a neighbor in \( I \). For each \( v \in I \), let \( S_v \) consist of \( v \) and its associated vertices.
3: Partition each vertex subset \( S_v \) into a constant number of cliques. Let \( C_v \) be the set of cliques into which \( S_v \) is partitioned. Return \( \bigcup_v C_v \).
\end{algorithm}

The first two steps are simple and for the third step we make use of ideas due to Raghavan and Spinrad [34]. Raghavan and Spinrad [34] present a “robust” algorithm for the problem of finding a maximum cardinality clique (henceforth, \emph{maximum clique}) in a given UDG. Their algorithm is robust in the sense that it takes as input an arbitrary graph \( G \) and in polynomial time, (i) either returns a maximum clique in \( G \) or (ii) produces a certificate indicating that \( G \) is not a UDG. The existence of such an algorithm is surprising because both problems (a) recognizing whether a given graph is a UDG and (b) finding a maximum cardinality clique in a given graph, are NP-hard. The key idea underlying the Raghavan-Spinrad algorithm is the existence of a superclass \( \mathcal{G} \) of the class of UDGs such that in polynomial time one can determine if a given graph \( G \) is in \( \mathcal{G} \) or not. Furthermore, for any \( G \) in \( \mathcal{G} \) a maximum clique can be computed in polynomial time.

The superclass \( \mathcal{G} \) is the set of all graphs that admit a \emph{cobipartite neighborhood edge elimination ordering} (CNEEO). Given a graph \( G = (V, E) \) and an edge ordering \( L = (e_1, e_2, \ldots, e_m) \) of \( E \), let \( G_L[i] \) denote the spanning subgraph of \( G \) with edge set \( \{e_1, e_1+1, \ldots, e_m\} \). For each edge \( e_i = \{x, y\} \) define \( N_L[i] \) to be the set of common neighbors of \( x \) and \( y \) in \( G_L[i] \). An edge ordering \( L = (e_1, e_2, \ldots, e_m) \) of \( G = (V, E) \) is a CNEEO if for every edge \( e_i \), \( N_L[i] \) induces a cobipartite graph (in \( G_L[i] \)). Recall that a cobipartite graph is the complement of a bipartite graph; in other words, the vertices of the cobipartite graph can be partitioned into two cliques. Raghavan and Spinrad prove three results: (1) If a graph \( G \) admits a CNEEO, then there is a simple greedy algorithm for finding a CNEEO of \( G \). (2) Given a graph \( G \) and a CNEEO of \( G \), a maximum clique in \( G \) can be found in polynomial time. (3) Every UDG admits a CNEEO. To see the second result, consider a maximum clique \( C \) in \( G \). Let \( L = (e_1, e_2, \ldots, e_m) \) be a CNEEO of \( G \) and let \( e_i \) be the edge in \( G[C] \) that occurs first in \( L \). Then \( C \) is contained in the cobipartite graph \( N_L[i] \). In fact, \( C \) is a maximum clique in \( N_L[i] \) as well. Using the fact that \( N_L[i] \) is cobipartite and the fact that a maximum independent set in a bipartite graph can be computed in polynomial time, we can compute a clique of cardinality \( |C| \) in \( N_L[i] \) in polynomial time. Raghavan and Spinrad obtain the third result by showing that if we take a geometric representation of the given UDG \( G \) and order edges in non-increasing length order, we get a CNEEO of \( G \). This follows fairly easily from the following geometric observation. Let \( \{x, y\} \) be a
line segment in the plane and let \( r = \|x - y\|_2 \). Then \( \{x, y\} \) partitions the intersection of \( \text{Disk}(x, r) \cap \text{Disk}(y, r) \) into two regions of diameter at most \( r \). See Figure 3(a) for an illustration.

Figure 3: (a) Suppose that edge \( \{x, y\} \) has rank \( i \) in an ordering of the edges of \( G \) in non-increasing length order. The points in the common neighborhood of \( x \) and \( y \) in \( G_L[i] \) are exactly those in the lune shown above. A point outside the lune may be a common neighbor of \( x \) and \( y \) in \( G \), but not in \( G_L[i] \). The diameter of the upper and lower halves of the lune are \( r \) and therefore the vertices in each half induce a clique in \( G_L[i] \). Hence, \( N_L[i] \) induces a cobipartite graph. (b) A sample run of \textsc{Nbd-Clique-Partition} on the “bad example” for the greedy algorithm. The 5 thick lines correspond (in some order) to the 5-edge sequence guessed by \textsc{Nbd-Clique-Partition} in Step (1). Among the guessed edges, the edge between \( A_1 \) and \( A_k \), say \( e_A \), and the edge between \( B_1 \) and \( B_k \), say \( e_B \), are critical because the common neighborhood of the endpoints of \( e_A \) is the clique \( \bigcup_i A_i \) and similarly, the common neighborhood of the endpoints of \( e_B \) is the clique \( \bigcup_i B_i \). Note that in the guess as highlighted by the thick lines, the algorithm will produce a clique partition with at most 10 cliques independent of the order in which the 5 edges are processed.

The three results mentioned above lead to a polynomial-time algorithm that will successfully report a maximum clique for every input UDG and for some graphs that are not UDGs (but admit a CNEEO). As pointed out in the introduction, being able to compute a maximum clique in a UDG without geometry does not imply a way to implement “Step 3” of the \textsc{Clique-Partition} algorithm. However, we now show how to use the CNEEO idea to implement this step in polynomial time. Let \( G_v := G[S_v] \) be the subgraph of \( G \) induced by \( S_v \); we call this \( H_v \) in the introduction. Since \( G_v \) is also a UDG, it admits a CNEEO. We start with a key lemma whose proof is straight forward.

**Lemma 5.** Let \( C \) be a clique in \( G_v \) and let \( L = (e_1, e_2, \ldots, e_m) \) be a CNEEO of \( G_v \). There is an \( i, 1 \leq i \leq m \), such that \( N_L[i] \) contains \( C \).

**Proof.** Let \( e_i = \{x, y\} \) be the edge in \( C \) that occurs first in \( L \). Recall the notation \( N_L[i] \); this denotes the common neighborhood of the endpoints of edge \( e_i \) in the spanning subgraph
of \( G_v \) containing only edges \( e_i, e_{i+1}, \ldots, e_m \). Since \( e_i \) is the first edge in \( C \), \( N_L[i] \) contains \( C \).

Recall that the closed neighborhood of a vertex in a UDG can be partitioned into at most 5 cliques. Let \( C_1, C_2, \ldots, C_5 \) be a clique partition of \( S_v \). The implication of the above lemma is that even though we do not know the clique partition \( C_1, C_2, \ldots, C_5 \), we do know that for every CNEEO \( L = (e_1, e_2, \ldots, e_m) \) of \( G_v \), there is an edge \( e_i \) such that \( N_L[i] \) can be partitioned into two cliques whose union covers \( C_1 \). This follows simply from the fact that \( L \) is a CNEEO and therefore the graph induced by \( N_L[i] \) is cobipartite. This suggests an algorithm that starts by guessing an edge sequence \((f_1, f_2, \ldots, f_5)\) of \( G_v \). Then the algorithm computes \( L \), a CNEEO of \( G_v \). The algorithm’s first guess is “good” if \( f_1 \) is the edge in \( C_1 \) that occurs first in \( L \). Suppose this is the case and further suppose that \( f_1 \) has rank \( i \) in \( L \). Then \( C_1 \) is contained in \( N_L[i] \). Therefore, when \( N_L[i] \) is deleted from \( G_v \) we have a graph, say \( G'_v \), that can be partitioned into 4 cliques, namely \( C_j \setminus N_L[i] \) for \( j = 2, 3, 4, 5 \). For each \( j \), let \( C'_j \) denote \( C_j \setminus N_L[i] \) and let \( L' \) be a CNEEO of \( G'_v \). The algorithm’s second guess, \( f_2 \), is “good” if \( f_2 \) is the edge in \( C'_2 \) that occurs first in \( L' \). Letting \( i' \) denote the rank of \( f_2 \) in \( L' \), we see that \( N_L[i'] \) contains \( C'_2 \). We then delete \( N_L[i'] \) from \( G'_v \) to get a graph that can be partitioned into 3 cliques. Continuing in this manner we get a partition of \( G_v \) into 10 cliques. The algorithm is described more formally in Algorithm 2 (which we call \textsc{NBD-CLIQUE-PARTITION}).

**Algorithm 2 NBD-CLIQUE-PARTITION**\((G_v)\)

1. for all 5-edge sequences \((f_1, f_2, \ldots, f_5)\) do
2. \( G^{(0)} \leftarrow G_v \)
3. for \( j = 1 \) to \( 5 \) do
4. Compute a CNEEO \( L \) of \( G^{(j-1)} \)
5. \( i \leftarrow \) rank of \( f_j \) in \( L \)
6. Partition \( N_L[i] \) into two cliques \( C'_j \) and \( C''_j \)
7. \( G^{(j)} \leftarrow G^{(j-1)} \setminus N_L[i] \)
8. if (\( G^{(5)} = \emptyset \)) then
9. return \( \{C'_j, C''_j \mid j = 1, 2, \ldots, 5\} \)

Figure 3(b) shows a sample run of the algorithm. The correctness of \textsc{NBD-CLIQUE-PARTITION} follows from the fact that there is some edge sequence \((f_1, f_2, \ldots, f_5)\) of \( E(G_v) \) for which \( G_5 \) is empty. The following lemma proves this claim.

**Lemma 6.** Algorithm \textsc{NBD-CLIQUE-PARTITION} partitions \( G_v \) into at most 10 cliques.

**Proof.** As in the algorithm, use \( G^{(0)} \) to denote \( G_v \). Assume, without loss of generality, that \( G^{(0)} \) has a clique partition of size 5. Let \( C''_v^{(0)} = \{C''_1^{(0)}, C''_2^{(0)}, \ldots, C''_5^{(0)}\} \), denote such a clique partition of \( G^{(0)} \). According to Lemma 5, there is a CNEEO \( L^{(0)} = (e_1, e_2, \ldots, e_m) \) of \( G^{(0)} \) and a rank \( i, 1 \leq i \leq m \) such that \( N_{L^{(0)}}[i] \) induces a graph containing the clique \( C''_1^{(0)} \); rename the edge \( e_i \) as \( f_1 \). By definition, the graph induced by \( N_{L^{(0)}}[i] \) is co-bipartite and therefore it can be partitioned into at most two cliques. Therefore \( G^{(1)} := G^{(0)} \setminus N_{L^{(0)}}[i] \) is a UDG having at most 4 cliques; call these \( C''_v^{(1)} = \{C''_1^{(1)}, C''_2^{(1)}, \ldots, C''_4^{(1)}\} \). The process
can be recursively repeated on $G^{(1)}$, yielding a CNEEO $L^{(1)}$ of $G^{(1)}$, an edge $f_2$ of rank $j$ in $L^{(1)}$ such that $N_{L^{(1)}}[j]$ induces a co-bipartite graph containing $C^{(1)}_1$. Continuing in this manner for at most three more recursive calls, we get a clique partition of $G_v$ into at most 10 cliques. Algorithm $\text{NBD-CLIQUE-PARTITION}$ is simply an iterative version of this recursive algorithm. □

The above algorithm can be slightly optimized to yield an 8-clique partition; when three of the cliques of $G_v$ have been deleted (at which time we have output at most 6 cliques), we are left with a cebipartite graph that can be easily partitioned into at most 2 cliques. This also means that it suffices to guess 3-edge sequences, $(f_1, f_2, f_3)$, rather than 5-edge sequences, thereby speeding up the algorithm. The fact that polynomially many edge sequences are considered along with the fact that the CNEEO of a graph, if it exists, can be computed in polynomial time (see \cite{34}) and the fact that a co-bipartite graph can be partitioned into two cliques in linear time (by 2-coloring the complement) imply that the algorithm $\text{NBD-CLIQUE-PARTITION}$ runs in polynomial time. Thus Step (3) of the $\text{CLIQUE-PARTITION}$ algorithm can be implemented in polynomial time by calling the

$\text{NBD-CLIQUE-PARTITION}$ algorithm for each vertex $v \in I$.

Let $C = \{C_1, C_2, \ldots, C_t\}$ be the clique partition of $G$ produced by the algorithm $\text{CLIQUE-PARTITION}$. Let $H := G[C]$. We now prove that $H$ is growth-restricted. For each vertex $c \in V(H)$, there is a corresponding vertex $v$ in the MIS $I$ of $G$. Specifically, $c$ corresponds to a clique in $G$ that was obtained by partitioning $S_v$ for some vertex $v \in I$. Recall that $S_v$ consists of $v$ along with some neighbors of $v$. Denote by $i(c)$ the vertex in $I$ corresponding to $c \in V(H)$. Consider an arbitrary 2-dimensional realization $\Phi$ of $G$ and for any pair of vertices $x, y \in V$, let $|xy|$ denote $||\Phi(x) - \Phi(y)||_2$. Let $B(v, r) = \{u \in V(H) \mid d_H(v, u) \leq r\}$.

**Lemma 7.** For any $u, v \in V(H)$ and $r \geq 0$, if $u \in B(v, r)$ then $|i(u)i(v)| \leq 3r$.

**Proof.** Consider two neighbors in $H$, $x$ and $y$. Let $C_x$ and $C_y$ denote the cliques in $G$ that were contracted into $x$ and $y$ respectively. Since $x$ and $y$ are neighbors in $H$, there are vertices $x' \in C_x$ and $y' \in C_y$ that are neighbors in $G$. Also, because of the way the cliques are constructed, $x'$ is $i(x)$ or a neighbor of $i(x)$ in $G$. Similarly $y'$ is $i(y)$ or a neighbor of $i(y)$ in $G$. Since $G$ is a UDG, by triangle inequality $|i(x)i(y)| \leq |i(x)x'| + |x'y'| + |y'i(y)| \leq 3$.

If $u \in B(v, r)$, then there is a $uv$-path $P$ in $H$ of length at most $r$. Corresponding to $P$ there is a sequence of vertices in $I$ starting with $i(u)$ and ending with $i(v)$ such that consecutive vertices in this sequence are at most 3 units apart in any realization. Therefore, by triangle inequality $|i(u)i(v)| \leq 3r$. □

**Lemma 8.** There is a constant $\alpha$ such that for any $v \in V(H)$, $|B(v, r)| \leq \alpha \cdot r^2$.

**Proof.** Let $X$ be the number of vertices in $B(v, r)$. By the previous lemma, for each $u \in B(v, r)$, there is a vertex $i(u) \in I$ such that $i(u) \in \text{Disk}(i(v), 3r)$. Here $\text{Disk}(i(v), 3r)$ denotes the disk of radius $3r$ centered at vertex $i(v)$ in some realization of $G$. Also, by Lemma 6, there are at most 10 vertices in $H$ that have the same corresponding vertex in $I$. Therefore, the number of vertices in $\text{Disk}(i(v), 3r)$ needs to be at least $X/10$. Any pair of
vertices in $I$ are more than one unit apart (in Euclidean distance) from each other. By the standard packing argument, this implies that the ball $\text{Disk}(i(v), 3r)$ can contain at most $4 \cdot (3r + 1/2)^2$ points in $I$. Therefore, $X/10 \leq 4(3r + 1/2)^2$ and hence for some constant $\alpha$, we have $X \leq \alpha \cdot r^2$.

Lemma 8 leads to the following theorem.

**Theorem 1.** There is a polynomial time algorithm that takes as input a UDG $G = (V, E)$ without geometric representation and constructs a clique partition $C = \{C_1, C_2, \ldots, C_t\}$ of $G$ such that $G'[C]$ has constant growth rate.

A side effect of our construction is that the constructed clique partition $C = \{C_1, C_2, \ldots, C_t\}$ is a constant-factor approximation to the minimum clique partition problem for a UDG, given without geometric representation. The minimum clique partition problem is formulated thus: Given a graph $G = (V, E)$, partition $V$ into $C = \{C_1, C_2, \ldots, C_t\}$ such that each $C_i$ is a clique and $t$ is minimized. Our claim follows from the fact that the size of any independent set is a lower bound on the size of a minimum clique partition and our solution produces a clique partition whose size is at most 8 times the size of an MIS. This is the first constant factor approximation to the minimum clique partition problem for UDGs given without geometric representation. One can use the $O(\log^* n)$ round algorithm for MIS on growth-bounded graphs of Schneider and Wattenhofer [37] to implement “Step 1” of the CLIQUE-PARTITION algorithm. These observations lead to the following theorem.

**Theorem 2.** There is an 8-approximation algorithm for the minimum clique partition problem on UDGs given without geometric representation. Furthermore, this algorithm can be implemented in $O(\log^* n)$ rounds of distributed computation in the LOCAL model.

Using techniques presented in this article, Pirwani and Salavatipour [33] present a $(2 + \varepsilon)$-approximation, for any $\varepsilon > 0$, for the vertex-weighted version of the minimum clique partition problem on UDGs given without geometric representation. While they [33] give a centralized algorithm for the weighted case, their algorithm has an easy $O\left(\log^* n \over \varepsilon \log(1)\right)$ round distributed implementation in the LOCAL model [32] for the unweighted case.

## 4 The Embedding Algorithm

In this section, we describe two embedding algorithms: Our first algorithm obtains an embedding of the input UDG, $G = (V, E)$, that is of quality-2, if the planarity constraint is relaxed to allow the final embedding to reside in a host Euclidean space of $O(1)$-dimensions. Our second algorithm obtains an embedding of the input UDG into the Euclidean plane that is of quality $O(\log^3 n)$.

### 4.1 Quality-2 Embedding in $O(1)$-Dimensions

Levin, together with Linial, London, and Rabinovich [27], made a conjecture (Conjecture 8.2 in [27]) that is quite relevant to the best quality embedding problem. Let $\mathbb{Z}^d_\infty$ be the
infinite graph with vertex set \( \mathbb{Z}^d \) (i.e., the \( d \)-dimensional integral lattice) and an edge \( \{u, v\} \) whenever \( \| u - v \|_\infty = 1 \). For any graph \( G \), define \( \dim(G) \) to be the smallest \( d \) such that \( G \) occurs as a (not necessarily induced) subgraph of \( \mathbb{Z}^d \).

**Conjecture 1.** [Levin, Linial, London, Rabinovich] For any graph \( G = (V, E) \) with growth rate \( \rho_G \), \( G \) occurs as a (not necessarily induced) subgraph of \( \mathbb{Z}^O(\rho_G) \). In other words, \( \dim(G) = O(\rho_G) \).

Linial [26] introduced the following Euclidean analogue to this notion of dimensionality. For any graph \( G = (V, E) \), define \( \dim_2(G) \) to be the smallest \( d \) such that there is a mapping \( \Phi : V \mapsto \mathbb{R}^d \) with the properties: (i) \( \| \Phi(u) - \Phi(v) \|_2 \geq 1 \) for all \( u \neq v \in V \) and (ii) \( \| \Phi(u) - \Phi(v) \|_2 \leq 2 \) for all \( \{u, v\} \in E \).

Krauthgamer and Lee [18] show that the specific bound on \( \dim(G) \), mentioned in the above conjecture does not hold, by exhibiting a graph \( G \) for which \( \dim(G) = \Omega(\rho_G \log \rho_G) \). They also prove a weaker form of the conjecture by showing that \( \dim(G) = O(\rho_G \log \rho_G) \) for any graph \( G \) [18]. Finally, they also prove that \( \dim_2(G) = O(\rho_G \log \rho_G) \). This proof of this upper bound is shown via the probabilistic method; specifically, it is shown using the Lovász Local Lemma (LLL) that an appropriate embedding into \( O(\rho_G \log \rho_G) \)-dimensional space exists with positive probability. One can take this existence proof and turn it into an efficient algorithm for constructing such an embedding using standard algorithmic versions of the LLL [4, 29]. This result, along with Theorem 1 leads to the following theorem.

**Theorem 3.** There is a polynomial time algorithm, that takes as input a UDG without geometric representation and constructs an embedding of quality-2 in \( O(1) \)-dimensional Euclidean space.

**Proof.** Using UDG, \( G = (V, E) \) as input, we construct a growth-bounded cluster graph, \( H = (V', E') \), using Theorem 1. Next, we construct an embedding \( \Phi : V' \mapsto \mathbb{R}^{O(1)} \) using the algorithm of Krauthgamer and Lee [18, Theorem 6.1]. Under the embedding, \( \Phi \), the length of every non-edge is at least 1-unit, while the length of every edge is at most 2-units. Finally, since every \( v \in V' \) corresponded to a clique, \( C_v \), of a clique partition, \( C \), in \( G \), all vertices of \( C_v \) are assigned the same coordinate as that of \( \Phi(v) \), completing the final quality-2 embedding of \( G \). \( \square \)

### 4.2 \( O(\log^3 n) \) Quality Embedding in the Plane

In Section 4.1 we obtained a quality-2 embedding of a UDG into \( O(1) \)-dimensional space. In this section, we shrink the number of dimensions down to just 2 (from a possibly much larger constant), but give up on the quality of the embedding to some extent. Algorithm 3 (Embed\( (G) \)) is a complete description of an algorithm that takes as input a UDG \( G \) (without geometric representation) and constructs an embedding of \( G \) into \( \mathbb{R}^2 \) with quality \( O(\log^3 n) \). Step 1 of this algorithm is to construct a growth-restricted cluster graph \( H \) from the input UDG, \( G \) (see Section 3). Step 2 of this algorithm constructs a \( (\log n, O(\log n)) \)-volume respecting embedding of \( H \). This step depends on the following theorem due to Krauthgamer et al. [19] applied to the hop-metric induced by \( H \).
**Theorem 4** (Krauthgamer et al. [19]). There is a polynomial time algorithm that constructs, with high probability and for any \( k, 1 \leq k \leq n \), a \((k, O(\log n))\)-volume respecting embedding, of any metric space with \( n \) points.

Steps (3)-(5) describe (i) a random projection into \( \mathbb{R}^2 \) of the embedding constructed in Step (2), (ii) a “rounding” step that maps the points in \( \mathbb{R}^2 \) to grid points, and (iii) constructing an embedding of the original input graph \( G \) from the embedding of the cluster graph \( H \). The random projection and the subsequent “rounding” step are essentially the same as in Vempala’s algorithm [38] and are also used by [28, 22]. We summarize all the steps of the algorithm in the following.

**Algorithm 3 Embed(G)**

1: Construct a clique partition \( \mathcal{C} = \{C_1, C_2, \ldots, C_t\} \) of the given UDG \( G = (V, E) \) so that the induced cluster graph \( H := G[\mathcal{C}] \) is growth-restricted. This is described in Section 3.

2: Let \( V(H) = \{v_1, v_2, \ldots, v_n\} \). Construct a \((\log n, O(\log n))\)-volume respecting embedding \( \Phi \) of the shortest path metric of \( H \), by applying Theorem 4 of Krauthgamer et al. [19]. Let \( u_i := \Phi(v_i) \) for \( i = 1, 2, \ldots, n \).

3: Choose 2-dimensional unit vectors, \( \ell_1 \) and \( \ell_2 \), independently and uniformly at random. Project the point set \( \{u_1, u_2, \ldots, u_n\} \) onto each of the two vectors, mapping each \( u_i \) to \( (u_i \cdot \ell_1, u_i \cdot \ell_2) \). Denote each \( (u_i \cdot \ell_1, u_i \cdot \ell_2) \) by \( w_i \).

4: Discretize the plane into grid, with each cell having dimensions \( 1/\sqrt{n} \times 1/\sqrt{n} \). Call each such grid cell an outer grid cell. Let \( M \) be the maximum number of points \( w_i \) that fall in any outer cell after the random projection step. Subdivide each outer grid cell by an inner grid, with each inner grid cell having dimensions \( 1/\sqrt{n \cdot M} \times 1/\sqrt{n \cdot M} \). For each outer cell \( C \), map each point that falls into \( C \) to grid points of the inner grid such that each \( w_i \) gets mapped to a distinct grid point. Finally, scale up all points by a factor of \( \sqrt{n \cdot M} \) along both dimensions so that each inner grid cell has unit width.

5: Since every vertex \( v_i \) in \( H \) is associated with a clique \( C_i \) in \( G \), all vertices in \( C_i \) are assigned the coordinates assigned to \( v_i \) in Step (4), to get the final embedding of \( G \).

### 4.2.1 Analysis of Approximation Guarantee

Here we show that the above algorithm, with high probability, yields a 2-dimensional embedding of quality \( O(\log^3 n) \). The analysis is similar to Vempala’s analysis [38] and that of Kuhn et al. [22] and Moscibroda et al. [28]. The proof of Lemma 11 is included mostly for the sake of completeness. An interesting aspect of our analysis is that a key technical lemma (Lemma 9), that was proved using “spreading constraints” by Vempala [38], follows quite easily, simply from the fact that we are working with a growth-restricted graph. Here, \( d(u, v) \) refers to the fewest number of edges between vertices \( u \) and \( v \) in the graph \( H \).

**Lemma 9.** There is a constant \( \beta \) such that for any \( v \in V(H) \), and any \( S \subseteq V(H) \),

\[
\sum_{u \in S} \left( \frac{1}{d(v, u)} \right)^2 \leq \beta \cdot \log |S|.
\]
Proof. Lemma 8 tells us that for some constant \( \alpha \), \(|B(v, r)| \leq \alpha \cdot r^2\). Recall that \( B(v, r) = \{ u \in V(H)|d(u, v) \leq r \}\). The sum \( \sum_{u \in S} \left( \frac{1}{d(v, u)} \right)^2 \) is maximized when, the largest possible subset \( S_1 \subseteq S \) of vertices is at distance 1 from \( v \), the largest possible subset \( S_2 \subseteq S \setminus S_1 \) of vertices is at distance 2 from \( v \) and so on. Thus, \(|S_1| = \alpha \cdot (i^2 - (i - 1)^2) = \alpha \cdot (2i - 1)\) for \( i = 1, 2, \ldots \). Therefore,

\[
\sum_{u \in S} \left( \frac{1}{d(v, u)} \right)^2 = \sum_{i \geq 1} \alpha \cdot \frac{(2i - 1)}{i^2} \leq \sum_{i \geq 1} \frac{2\alpha}{i} \leq 2\alpha \cdot (\ln |S| + 1).
\]

The last inequality follows from the fact that \( i \) can take on at most \(|S|\) distinct values. Hence, for some constant \( \beta \), \( \sum_{u \in S} \left( \frac{1}{d(v, u)} \right)^2 \leq \beta \cdot \log |S| \).

The above upper bound on the sum on inverse square distances from \( v \) is critically used to derive the following upper bound on sum of inverse squares of tree volumes of all size-\( k \) vertex subsets. For completeness we reproduce the proof from [22].

**Lemma 10** (Kuhn et al. [22]).

\[
\sum_{S \subseteq V(H), |S| = k} \left( \frac{1}{Tvol(S)} \right)^2 \leq 2k! \cdot \left( \frac{\beta}{4} \right)^{k-1} \cdot n \cdot \log^{k-1} n.
\]

Here \( \beta \) is the constant that appears in the previous lemma.

Proof. To obtain an enumeration of all subsets \( S \subseteq V \) of size \( k \), we first choose a \( v_1 \in V(H) \). Then, we choose \( v_2 \in V(H) \setminus \{ v_1 \} \). Then, we choose \( v_3 \in V(H) \setminus \{ v_1, v_2 \} \), and so on, finally choosing \( v_k \in V(H) \setminus \{ v_1, v_2, \ldots, v_{k-1} \} \). For notational convenience, let \( I_j \) denote \( \{ v_1, v_2, \ldots, v_j \} \), \( 1 \leq j < k \). Thus,

\[
\sum_{S \subseteq V(H), |S| = k} \left( \frac{1}{Tvol(S)} \right)^2 = \frac{1}{k!} \sum_{v_1 \in V(H)} \sum_{v_2 \in V(H) \setminus I_1} \cdots \sum_{v_k \in V(H) \setminus I_{k-1}} \left( \frac{1}{Tvol(I_k)} \right)^2
\]

Feige [14] showed that for any finite metric space \((V, d)\) and \( S = \{ v_1, v_2, \ldots, v_k \} \subset V \),

\[
\frac{2^{k-1}}{Tvol(S)} \leq \sum_{\pi \text{ over } [k]} \frac{1}{d(v_{\pi(1)}, v_{\pi(2)}) \cdot d(v_{\pi(2)}, v_{\pi(3)}) \cdots d(v_{\pi(k-1)}, v_{\pi(k)})}
\]

where, the summation is over all permutations of \( S \). Thus,

\[
\left( \frac{1}{Tvol(S)} \right)^2 \leq \frac{1}{2^{2k-2}} \left( \sum_{\pi \text{ over } [k]} \frac{1}{d(v_{\pi(1)}, v_{\pi(2)}) \cdot d(v_{\pi(2)}, v_{\pi(3)}) \cdots d(v_{\pi(k-1)}, v_{\pi(k)})} \right)^2
\]
For notational convenience, denote \( \sum_{\pi} d(v_{\pi(1)}, v_{\pi(2)}) \cdot d(v_{\pi(2)}, v_{\pi(3)}) \cdots d(v_{\pi(k-1)}, v_{\pi(k)}) \) by \( \frac{1}{d(\pi)} \). So,

\[
\left( \frac{1}{T\text{vol}(S)} \right)^2 \leq \frac{1}{2^{2k-2}} \sum_{\pi \pi'} \frac{1}{d(\pi) \cdot d(\pi')} = \frac{1}{2^{2k-2}} \left( \sum_{\pi} \frac{1}{d(\pi)^2} + \sum_{\pi' \neq \pi} \frac{1}{d(\pi) \cdot d(\pi')} \right)
\]

From inequality (1) and (2), we get,

\[
\sum_{S \subseteq V(H), |S| = k} \left( \frac{1}{T\text{vol}(S)} \right)^2 \leq \frac{k!}{2^{2k-2} \cdot \beta} \sum_{v_1} \sum_{v_2} \cdots \sum_{v_k} \left( \frac{1}{d(v_1, v_2)} \right)^2 \sum_{v_3} \left( \frac{1}{d(v_2, v_3)} \right)^2 \cdots \sum_{v_k} \left( \frac{1}{d(v_{k-1}, v_k)} \right)^2
\]

An application of Lemma 9 to inequality (3) yields the following:

\[
\sum_{S \subseteq V(H), |S| = k} \left( \frac{1}{T\text{vol}(S)} \right)^2 \leq \frac{k!}{2^{2k-2} \cdot \beta \log |S|^{k-1}} \leq \frac{k!}{2^{2k-2} \cdot \beta^{k-1} \cdot n (\log n)^{k-1}} = 2k! \left( \frac{\beta}{\pi} \right)^{k-1} \cdot n (\log n)^{k-1}
\]

**Lemma 11.** After Step (4), with high probability, the maximum number of vertices that fall in an outer grid cell is \( O(\log^5 n) \).

**Proof.** Consider an outer grid cell \( C \). Note that \( C \) is defined by two length \( 1/\sqrt{n} \) intervals – one on line \( l_1 \) and the other on line \( l_2 \). For any subset \( S \) of vertices, let \( X_S^i \) be the indicator random variable that is 1 if all vectors associated with the vertices of \( S \) fall in the interval corresponding to cell \( C \) on line \( l_i \). Let \( N_C \) denote the number of size-\( k \) subsets \( S \subseteq V(H) \) that have fallen into \( C \) (as a result of Step (3)). In Step (3), the value of \( k \) that was used
is log \( n \); we will replace \( k \) by log \( n \) later in the proof.

\[
E[N_C] = \sum_{S \subseteq V(H), |S| = k} E[X^1_S \cdot X^2_S]
= \sum_{S \subseteq V(H), |S| = k} E[X^1_S]E[X^2_S]
= \sum_{S \subseteq V(H), |S| = k} \Pr[X^1_S = 1]^2
\leq \gamma \cdot \sum_{S \subseteq V(H), |S| = k} \left( \frac{W^{k-1} n^{k-1}}{(k-1)!\Evol(S)} \right)^2
\leq \gamma \cdot \sum_{S \subseteq V(H), |S| = k} \left( \frac{1}{(k-1)!\Evol(S)} \right)^2
\leq \gamma \cdot \sum_{S \subseteq V(H), |S| = k} \left( \frac{(\delta \log n)^{k-1}}{(k-1)!\Vol(S)} \right)^2
\leq \gamma \cdot \sum_{S \subseteq V(H), |S| = k} \left( \frac{(\delta \log n)^{k-1} \cdot 2^k}{\Tvol(S)} \right)^2
\leq \gamma \cdot \delta^{2(k-1)} \cdot (4 \log^2 n)^k \cdot \sum_{S \subseteq V(H), |S| = k} \left( \frac{1}{\Tvol(S)} \right)^2
\leq \gamma \cdot \delta^{2(k-1)} \cdot 8k! \cdot \beta^k \cdot n \cdot \log^3 n \leq n \cdot (\zeta \cdot k \cdot \log^3 n)^k
\]

Using Markov’s inequality we get \( \Pr[N_C \geq n^4 \cdot n \cdot (\zeta \cdot k \cdot \log^3 n)^k] \leq 1/n^4 \). Since \( H \) has \( n \) vertices, the maximum distance between a pair of vertices in \( H \) is bounded above by \( n \). Since the embedding in Step (2) is non-expansive, we can bound the total number of outer cells in each dimension by \( n^{1.5} \), which leads to a total of \( n^3 \) outer cells. Using the union bound we get

\[
\Pr \left[ \text{there exists an outer cell containing } n^4 \cdot n \cdot (\zeta \cdot k \cdot \log^3 n)^k \text{ size-} k \text{ subsets of } V(H) \right] \leq \frac{1}{n}
\]

Hence, with probability at least \( 1 - 1/n \), every outer cell contains fewer than \( n^4 \cdot n \cdot (\zeta \cdot k \cdot \log^3 n)^k \) size- \( k \) subsets of \( V(H) \). Using the fact that if there are \( N \) subsets of size \( k \) (in an outer cell), then there are at most \( kN^{1/k} \) points in it, we obtain that with probability at least \( 1 - 1/n \), every outer cell contains at most \( k(n^5 \cdot (\zeta \cdot k \cdot \log^3 n)^k)^{1/k} \) points. Since \( k = \log n \), using the fact that \( n^5/\log n = O(1) \), we get that with high probability, every outer cell has at most \( O(\log^5 n) \) points.

**Lemma 12.** With high probability, the maximum edge length in the final embedding is \( O(\log^3 n) \).

**Proof.** For any edge \( \{v_i, v_j\} \) in the cluster graph \( H \), after Step (2), \( ||v_i - v_j||_2 \leq 1 \), since the embedding constructed in Step (2) is non-expansive. After the random projection in Step (3), using the value of \( c = 4\sqrt{\log n} \) in Lemma 3, we see that \( ||v_i - v_j||_2 < \frac{4\sqrt{\log n}}{\sqrt{n}} \) for all edges \( \{v_i, v_j\} \in E(H) \), with probability at least \( 1 - 1/n \). Lemma 11 tells us that with high
probability, each outer grid cell is divided into at most \( O(\log^{2.5} n) \) inner grid cells along each of the two dimensions of the outer grid cell. Therefore, the scaling in Step (4) by a factor of at most \( \sqrt{n} \cdot \log^{2.5} n \), will cause each edge to have length at most \( O(\log^3 n) \), with high probability.

**Theorem 5.** With high probability, the quality of the embedding is \( O(\log^3 n) \).

**Proof.** Follows immediately from the high probability upper bound on the maximum edge length proved in Lemma 12 and the fact that every pair of vertices are separated by at least unit distance in “Step 4”.

## 5 Conclusions

The techniques used in this paper may not be able to provide an embedding of quality better than polylogarithmic in \( n \). These techniques were introduced in the context of the bandwidth minimization problem [14] and since nothing better than an \( O(\text{polylog}(n)) \) approximation is known for that problem, even on trees, perhaps we cannot hope for \( o(\text{polylog}(n)) \) quality embeddings of UDGs via these techniques. We have used the fact that our input graph has an \( O(1) \)-quality embedding in a critical way, for e.g., in constructing a growth-restricted approximation of the input graph. Perhaps we can take advantage of this feature of the input graph in other ways as well. One possibility is that a completely different approach, possibly analogous to the dynamic programming approach used by Gurari and Sudborough [15] for bandwidth minimization of graphs with constant bandwidth, could be used to obtain a constant-quality embedding of UDGs.

In one of the steps of our algorithm, we needed to construct a volume respecting embedding of the shortest path metric of a growth-restricted graph \( H \). For this, we used the algorithm of Krauthgamer et al. [19] that yields a \( (k, O(\log n)) \) volume respecting embedding for any \( k \) and any \( n \)-point metric space. In doing so, we did not exploit the special “growth restrictedness” of \( H \). It seems possible that the shortest path metric of a growth restricted graph has a volume respecting embedding with smaller distortion, for e.g., a \( (k, O(\sqrt{\log n})) \) volume respecting embedding. It is known that the class of graphs excluding a fixed minor has a \( (k, O(\sqrt{\log n})) \) volume respecting embedding [36] and there have been results showing that a metric space with constant doubling dimension admits a \( (k, O(\sqrt{\log n})) \) volume respecting embedding [19]. While we cannot use either of these results directly, it seems possible that ideas used in these results will lead to a \( (k, O(\sqrt{\log n})) \) volume respecting embedding for growth restricted graphs. Such an improvement would yield an \( O(\log^{2.5} n) \) quality embedding for UDGs. We leave this as an open question for future investigation.

Note that our algorithm makes no claims about the sum of edge lengths quality measure described in Section 1. For our problem, it was sufficient to construct a constant-sized neighborhood clique partition of the input UDG \( G \) in order to construct a growth-restricted approximation of \( G \). Such a partition may yield two cliques that have many edges between them and it is possible that two such cliques are embedded far away from each other, contributing a large term to the objective function. It seems that a different partitioning
and embedding scheme is needed to avoid this possibility. We have not investigated this problem and leave it as a question for future research.

Raghavan and Spinrad [34] claim that their algorithm for finding a maximum clique in a UDG without any geometry can be easily generalized to unit ball graphs (UBGs) in d-dimensional Euclidean space. In their paper, they show that ordering the edges in non-increasing order of length is a CNEEO for UDGs. However, this edge ordering is not a CNEEO for 3-dimensional UBGs and it is unclear how to generalize the Raghavan-Spinrad argument to higher dimensions. It is easy to see why a greedy ordering of edges by length does not produce a CNEEO even for $d = 3$. Afshani and Chan [1] also consider the problem of finding a maximum clique in 3-d UBGs and give several efficient constant-factor approximations to the maximum clique problem in 3-d UBGs with a realization. They note that the complexity of the clique problem for $d$-UBGs for $d > 2$, is unknown [1] even with the use of geometry.

In an orthogonal direction, it may be relevant to the wireless sensor networks application, if we could devise an efficient distributed algorithm (i.e., one that runs in poly-logarithmic number of rounds) for the best quality embedding problem, without sacrificing the quality of the embedding. We have made some progress toward this goal in this paper as shown by the fact that the cluster graph approximation step can be implemented in poly-logarithmic number of rounds using ideas from recent papers [20, 21]. However, at this point, it is unclear how to devise efficient distributed implementations for constructing a volume-respecting embedding and for Vempala’s random projection step.

Acknowledgements: We thank anonymous referees for their useful comments, criticisms, and for pointing out a flaw in an earlier manuscript.

References


