Unit Disk Graph Approximation*

Fabian Kuhn
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
kuhn@tik.ee.ethz.ch

Thomas Moscibroda
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
moscitho@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

ABSTRACT

Finding a good embedding of a unit disk graph given by its connectivity information is a problem of practical importance in a variety of fields. In wireless ad hoc and sensor networks, such an embedding can be used to obtain virtual coordinates. In this paper, we prove a non-approximability result for the problem of embedding a given unit disk graph. Particularly, we show that if *non-neighboring nodes* are not allowed to be closer to each other than distance 1, then two *neighbors* can be as far apart as $\sqrt{3/2} - \epsilon$, where ϵ goes to 0 as n goes to infinity, unless $P = NP$. We further show that finding a realization of a d-quasi unit disk graph with $d \geq 1/\sqrt{2}$ is NP-hard.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms

Algorithms, Theory

Keywords

virtual coordinates, embedding, unit disk graph, ad hoc networks, sensor networks

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

DIALM-POMC'04, October 1, 2004, Philadelphia, Pennsylvania, USA.

1. INTRODUCTION

In a *unit disk graph* [4], there is an edge between two nodes u and v if and only if the Euclidean distance between u and v is at most 1. Equivalently, each node is identified with a disk of unit radius $r = 1$ in the plane, and is connected to all nodes within (or on the edge of) its corresponding disk. Unit disk graphs have proven to be useful in modeling various physical real world problems. One prominent application of unit disk graphs can be found in the field of wireless networking, where a unit disk graph represents an idealized *multi-hop radio network*. Nodes are located in the Euclidean plane and are assumed to have identical (unit) transmission radii. They can communicate only if they are within mutual transmission range. Clearly, the unit disk graph model neatly captures this behavior and it is not surprising that it has become a standard when studying *ad hoc and sensor networks*.

When modeling ad hoc and sensor networks as a unit disk graph, it seems plausible that the *connectivity information* of the network graph contains sufficient information in order to obtain *topologically correct coordinate information*. Such coordinate information, in turn, would serve a number of needs arising in many applications of ad hoc and sensor networks.

To be more concrete, we are interested in assigning to each node a coordinate in the plane, such that nodes that are *neighbors* in the connectivity graph have at most Euclidean distance 1 in the plane, and nodes that are *not neighbors* in the connectivity graph have at least distance 1. In other words, we would like to *embed* a given unit disk graph in the plane. Unfortunately, it can be shown that this is impossible in polynomial time unless $P = NP$. In [3], Breu and Kirkpatrick show that given a graph G, deciding whether G is a unit disk graph is NP-complete. Finding a *realization* of a unit disk graph, that is, finding an embedding which fulfils all unit disk graph constraints, can be polynomially reduced to the recognition problem of [3] by a straightforward reduction. Assume there is an algorithm $\mathcal{A}$ which computes valid coordinates for a given unit disk graph G_U. Letting $\mathcal{A}$ run on an arbitrary graph G, we can subsequently verify whether the computed coordinates fulfil all conditions for a unit disk graph. G is a unit disk graph if and only if this is the case.

In this paper, we extend the result by Breu and Kirkpatrick by showing that a unit disk graph cannot even be *approximatively* embedded in the plane. In particular, we show that if we do not allow *non-neighboring nodes* to be closer to each other than distance 1, then, unless $P = NP$,

two *neighbors* can be as far apart as $\sqrt{3/2}-\epsilon$, where ϵ tends to 0 as the number of nodes n goes to infinity.

This inapproximability lower bound comes along with another impossibility result, namely, given a *d-quasi unit disk graph* G [8] with parameter $d \geq 1/\sqrt{2}$, it is NP-hard to find a realization (an embedding fulfilling all requirements of the quasi unit disk graph) of G. In a quasi unit disk graph, two nodes are connected by an edge if their distance is less than or equal to d, d being a parameter between 0 and 1. Furthermore, if the distance between two nodes is greater than 1, there is no edge between them. In the range between d and 1, the existence of an edge is not specified. Modeling ad hoc and sensor networks as a quasi unit disk graph (as opposed to a unit disk graph) has the advantage of being significantly closer to reality, while still being concise enough to permit stringent theoretical results [8, 1]. From a computational geometry point of view, the impossibility of finding a realization for a d-quasi unit disk graph with $d \geq 1/\sqrt{2}$ may be of independent interest.

Our lower bound results are of particular interest in the light of recent work on *virtual coordinates* for wireless ad hoc and sensor networks [12, 13, 10]. The reason is that the problems of finding a good embedding of a unit disk graph in the plane, and assigning virtual coordinates to nodes in wireless multi-hop networks – when being modelled as a unit disk graph – are equivalent.

For many higher-layer protocols, such as routing algorithms, it would be of great help if the nodes of an ad hoc or sensor network could be assigned virtual coordinates. As nodes may be tiny sensors in a sensor network (where equipment is restricted to the minimum due to limitations in energy consumption, weight, or cost), we may not generally assume that nodes are capable of sensing directions and/or distances to neighboring nodes. Hence, it is highly desirable to derive coordinates from *connectivity information* only, which can be collected even by the simplest nodes. In the literature on virtual coordinates, there have been several suggestions for heuristics [12, 13] without provable performance guarantees. The only existing approximation algorithm given in [10] achieves an approximation ratio of $O(\log^{2.5} n \cdot \sqrt{\log \log n})$. Our results show that none of the proposed heuristics can have a better approximation ratio than roughly $\sqrt{3/2}$. As a direct consequence, this rules out any kind of polynomial time approximation scheme (PTAS) for the problem. Independently, Lotker et. al. also showed that no PTAS exists for the problem of assigning virtual coordinates to the nodes of a unit disk graph [9]. However, in [9], no actual lower bound on the approximability of the problem is given.

The primary application of virtual coordinate approaches lies in the area of *routing* where virtual coordinates represent a simple way to embody the topology of the network. Based on virtual coordinates, geometric routing algorithm such as GFG/GPSR [2] or GOAFR [7] can be applied. Other than for routing algorithms in multi-hop radio networks, virtual coordinates have found prominent application in the context of *Internet mapping* [5, 11]. Here, the goal is to derive topological information about the Internet graph in order to improve anycast and peer-to-peer systems.

The paper is organized as follows. After introducing various definitions in Section 2, the central Section 3 states and proves the lower bounds. The paper is concluded in Section 4.

2. MODEL AND NOTATION

As indicated in the introduction, the lower bound construction is based to some extent on a generalization of the unit disk graph, the so-called d-quasi unit disk graph [8].

Definition 2.1 (Quasi Unit Disk Graph). *Let $V \in \mathbb{R}^2$ be a set of points in the 2-dimensional plane and let $d \in [0,1]$ be a parameter. The symmetric Euclidean graph $G = (V, E)$, such that for any pair $u, v \in V$*

- $dist(u,v) \leq d \Longrightarrow \{u,v\} \in E$
- $dist(u,v) > 1 \Longrightarrow \{u,v\} \notin E$

is called a d-quasi unit disk graph (d-QUDG).

Note that the definition of a quasi unit disk graph does not specify whether there is an edge between two nodes u and v having distance $d < dist(u,v) < 1$. Such an edge may be there, but it may not be there. Clearly, a unit disk graph is a special case of a d-quasi unit disk graph for $d = 1$.

As our goal is to find a good representation of the unit disk graph (given by its connectivity information) in the plane, we need to formalize the notion of such a representation. An *embedding* of a Graph $G = (V, E)$ in the Euclidean plane is a mapping $f : V \to \mathbb{R}^2$, such that each vertex v corresponds to a point (x, y) in the plane. An embedding which satisfies all constraints of a d-quasi unit disk graph G is called a *realization* of G.

Definition 2.2 (Realization). *A* realization *of a d-QUDG graph $G = (V, E)$ in the Euclidean plane is an embedding $r(G)$ of G such that $\{v_i, v_j\} \in E \Rightarrow dist(f(v_i), f(v_j)) \leq 1$ and $\{v_i, v_j\} \notin E \Rightarrow dist(f(v_i), f(v_j)) \geq d$, where $dist(u,v)$ denotes the Euclidean distance between two points.*

As shown in the introduction, finding a realization for unit disk graphs is NP-hard. Therefore, we resort to finding algorithms which compute an approximate realization, i.e., an embedding which may violate some unit disk constraints, but does not do so too much. The goal of an approximation algorithm is to map adjacent nodes to close-by coordinates, and non-adjacent nodes to distant coordinates. Hence, the measure for determining the quality of an approximation algorithm is based on the ratio between the most distant adjacent pair of nodes, to the closest non-adjacent pair of nodes. We formally evaluate approximation algorithms for virtual coordinates and UDG realizations by means of the so-called *quality* of an embedding as defined in [10].

Definition 2.3 (Quality). *Let $r(G)$ be an embedding of UDG $G = (V, E)$ in the plane. Let $\rho(u,v)$ denote the Euclidean distance between nodes u and v in $r(G)$. We define the* quality *of the embedding $r(G)$ as*

$$q(r(G)) := \frac{\max_{\{u,v\} \in E} \rho(u,v)}{\min_{\{\hat{u},\hat{v}\} \notin E} \rho(\hat{u},\hat{v})}.$$

Let $\mathcal{G}$ denote the family of all unit disk graphs. We consider algorithms which, given an input graph $G \in \mathcal{G}$, compute an embedding $r_{ALG}(G)$. We say that an algorithm achieves approximation ratio α if $q(r_{ALG}(G)) \leq \alpha$ for all $G \in \mathcal{G}$.

In the following, we are going to place an inherent bound on the approximability of virtual coordinates and unit disk

graph embeddings. In particular, we prove in the Section 3 that finding an embedding $r(G)$ for a unit disk graph G such that,

$$q(r(G)) \leq \sqrt{3/2} - \epsilon$$

where ϵ approaches 0 as n tends to infinity is NP-hard.

3. LOWER BOUND

In this central section, we prove the lower bound on the approximation ratio. More particularly, Theorem 3.1 shows that, given a unit disk graph G, it is hard to find a good embedding in the plane. More precisely, it states that not only it is impossible to find a unit disk graph embedding for G, but it is even impossible to embed G as a d-quasi unit disk graph for certain values of d.

THEOREM 3.1. *Given a unit disk graph $G = (V, E)$, it is NP-hard to find a realization of G as a d-QUDG with $d \geq \sqrt{2/3} + \epsilon$, where ϵ tends to 0 as n goes to infinity.*

As motivated in the introduction, this theorem has manifold implications. In the context of ad-hoc and sensor networks, for instance, it places a bound on the ability to derive virtual coordinates from the graph's connectivity properties only.

The proof is based on a reduction from an instance of 3-SAT. Without loss of generality, we can assume that each variable appears in at most 3 clauses [6]. We give a polynomial time construction of a graph $G_C = (V_C, E_C)$ from an instance C of SAT, such that the following holds:

- C is satisfiable $\Rightarrow$ G_C is realizable as UDG
- C is not satisfiable $\Rightarrow$ G_C is not realizable as d-QUDG with $d \geq \sqrt{2/3} + \epsilon$.

Hence, an approximation algorithm $\mathcal{A}$ with approximation ratio better than $\sqrt{2/3} + \epsilon$ could be used to decide in polynomial time whether the given instance of 3-SAT is satisfiable, thus implying P = NP.

3.1 The Reduction

The construction of $G_C = (V_C, E_C)$ was inspired by a construction in [3]. In the following, we summarize the reduction given in [3] to a level of detail which is necessary to understand our result.

We begin by constructing an intermediate (undirected) graph G_C^{SAT} from the instance C of 3-SAT, such that the clauses and literals of C correspond to vertices in G_C^{SAT}. There is an edge between a *literal vertex* and a *clause vertex* if the literal appears in the clause. The graph G_C^{SAT} is *orientable* if its edges can be directed such that the following two conditions hold. First, the outdegree of each clause vertex is at least 1. And secondly, for each variable, either the positive literal vertex or the negative literal vertex has indegree 0. Intuitively, the first condition secures that each clause is satisfied and the second condition guarantees that each variable can be set to either true or false. Using this notion of orientability, the following Lemma can easily be shown [3]:

LEMMA 3.2 ([3]). *C is satisfiable if and only if G_C^{SAT} is orientable.*

The next step is to draw G_C^{SAT} on a grid of size $O(|C| \cdot |V|)$, where C and V are the set of clauses and variables of the 3-SAT instance, respectively. See Figure 1 for an example of how clauses and literals are linked to each other by paths for the instance $C = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$. Each grid vertex is either unused, or is associated with a unique *component* of the drawing. Each component has up to 4 *terminals* located in the north, south, west, or east of the component. Each terminal is associated with a unique direction, i.e., a terminal can either be *directed away* or *directed towards* its component.

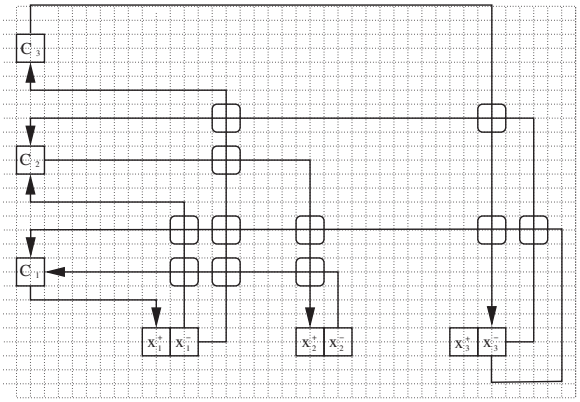

Figure 1: The grid drawing of G_C^{SAT}.

The grid drawing of G_C^{SAT} is called orientable if all terminals can be oriented subject to the condition that

1. For every variable, there is a literal component with *all* (potentially zero) terminals directed away from it.
2. For all other components, there is *at least one* terminal directed away from it.

The following Lemma states the equivalence of G_C^{SAT} and its drawing in the grid.

LEMMA 3.3 ([3]). *A grid drawing is orientable if and only if the underlying graph G_C^{SAT} is orientable, i.e., if the underlying instance C of 3-SAT is satisfiable.*

The basic idea of the reduction is to find a construction of G_C which simulates the grid drawing of G_C^{SAT}. In particular, we create a graph component for each of the different grid drawing components and connect them according to the grid drawing. We then show that G_C can be realized as a unit disk graph in case the grid drawing of G_C^{SAT} is orientable, and it cannot be realized as a d-quasi unit disk graph otherwise.

3.2 Cages and Chains

We begin by introducing the main building block of the constructed graph G_C, the *cages* and the *chains*, from which all components introduced in subsequent sections are built. A cage is a cycle of length k. Cages are hooked together (as shown in Figure 2) by merging two adjacent vertices on each cycle. The edge incident to these connecting vertices serves as a *hinge* for a path of independent vertices, the so-called *chain*. These chains come in two flavors. *Single-chains* have length $2t$, while *double-chains* have length $4t$, where the exact value of t is to be determined later.

Intuitively, the idea of this construction is as follows. Each cage has a maximum number of independent nodes which can be placed *within* the cage. If the chain between two adjacent cages is placed in the interior of one of the cages, it diminishes the available space inside that cage and may thereby squeeze out other chains (into the next neighboring cage) which may otherwise have been embedded in the cage. By connecting several such cages together, we are able to construct directable paths as imposed by the grid drawing of G_C^{SAT}.

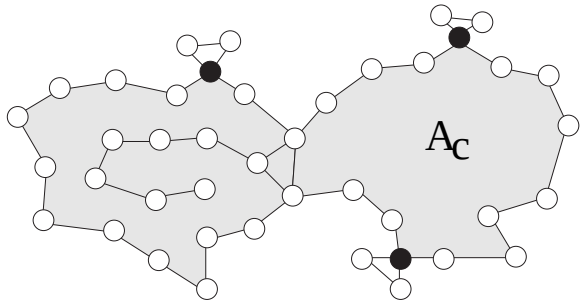

Figure 2: Two QUDG-cages connected by a hinge, the chain being embedded in the left cage. Both cages well-defined interior is shaded grey.

In order to ensure that this construction works, several properties must be guaranteed. Unlike in unit disk graphs, a realization of a cycle of length k in a d-QUDG does not necessarily have to be planar. Nonetheless, for our reduction to work, cages must have a *well-defined interior* into which chains can (and must) embed. Moreover, it must be ensured that chains embed entirely in one of its two adjacent cages, i.e., chains cannot leave a cage or be located outside of any cage. In the following, we are going to establish these and other vital properties.

The following two lemmas show that a cage does have a well-defined interior, and hence, the notion of a chain being *embedded* inside a cage makes sense. The first lemma was proven in [8].

LEMMA 3.4. *Let $e_1 = \{u_1, v_1\}$ and $e_2 = \{u_2, v_2\}$ be two intersecting edges in a d-QUDG G with parameter $d \geq 1/\sqrt{2}$. Then, at least one of the edges $\{u_1, u_2\}$, $\{u_1, v_2\}$, $\{v_1, u_2\}$, or $\{v_1, v_2\}$ exists in G.*

Using this lemma, it is now easy to show that, in spite of its possibly being non-planar, the realization of a cycle in a d-QUDG for $d \geq 1/\sqrt{2}$ must have a well-defined interior.

LEMMA 3.5. *Given a cycle C of length $k \geq 5$ of a d-QUDG with $d \geq 1/\sqrt{2}$, any realization $r(C)$ of C has a well-defined interior. The area A_C of this interior is at most*

$$A_C \leq k^2/4\pi.$$

PROOF. Let $r(C)$ be a realization of C. Consider two intersecting edges $\{u, v\}$ and $\{x, y\}$. By Lemma 3.4 and the fact that $k \geq 5$, *exactly* one of the edges $\{u, x\}$, $\{u, y\}$, $\{v, x\}$, or $\{v, y\}$ is in the cycle C. Assume without loss of generality that $\{u, x\} \in C$. Now, replace $\{u, x\}$ by a node x' located at the intersection of $\{u, v\}$ and $\{x, y\}$ and repeat the same procedure for all intersecting edges (see Figure 2 where nodes x' are represent by dark nodes). This yields a cycle of at least half the length of C with no intersecting edges. The ratio of area to perimeter is optimal in a circle and hence, the interior area of $r(C)$ is maximized if the vertices are realized as a regular k-gon. Because $r(C)$'s perimeter is at most k, the encircled area cannot be larger than $k^2/4\pi$. □

An immediate consequence of Lemmas 3.4 and 3.5 is that, if a part of a chain is embedded inside the interior of a cage, the remainder of the chain must also be embedded therein. A crossing edge would contradict the independence of at least one chain vertex and cycle vertex.

It is now time to establish a relationship between the perimeter k of a cage and the length $2t$ (or $4t$) of the chains. Intuitively, we want the following to hold: One the one hand, for unit disk graphs, we want cages to have an imaginary *capacity* of 2, i.e., exactly two single-chains or one double-chain may be embedded in a cage. If two single-chains or one double-chain embed in a cage, all other adjacent chains are forced from that cage into neighboring cages. This way, not only can information be propagated over paths, but it is also possible to build complex components such as the ones needed in Subsection 3.3. On the other hand, we want to guarantee that in a d-QUDG with $d \geq \sqrt{2/3} + O(1/k)$, cages do not have capacity more than 2, either. In other words, we want at most two single-chains or one double-chain to embed in the interior of a cage in a realization of G_C as a d-QUDG. With these properties, we ensure that realizing G_C as a unit disk graph and as a d-QUDG with $d \geq \sqrt{2/3} + O(1/k)$ are equally hard. The approximation lower bound then follows.

Obviously, a chain of length $2t$ contains t mutually independent vertices, each of which covers an area of at least $d^2\pi$ in a d-QUDG realization. Since we want $2t$ such independent unit disks to fit into a cage, we define t as

$$t := \left\lceil \frac{\eta_h k^2}{8\pi} - \beta k \right\rceil \tag{1}$$

where β is the smallest constant such that, for all k, at least $2t$ independent disks can be packed into the cage. For large enough k, β can be chosen smaller than 1. Subtracting the second term ensures that areas of less packing density along the cage's border (which is obviously in the order of the length of the perimeter $O(k)$) are taken into account. The constant $\eta_h := \frac{1}{6}\pi\sqrt{3}$ denotes the packing density of circles in the plane in a hexagonal lattice, which is the densest possible packing.

LEMMA 3.6. *Let C be a circle of length k. In any realization $r(C)$ of C as a d-QUDG with $d \geq \sqrt{2/3} + O(1/k)$, at most two single-chains each of length $2t$ or one double-chain of length $4t$ can be embedded in the interior of $r(C)$.*

PROOF. By Lemma 3.5, the area of the interior of a cage C is at most $A_C \leq k^2/4\pi$. We need to show that $3t$ independent disks (which corresponds to having three single-chains or both a double-chain and a single-chain) with radius $d \geq \sqrt{2/3} + \alpha k$ cannot be packed in C. The number of unit disks m_u and disks m_d with radius d which can maximally be packed in C is therefore

$$m_u = \frac{\eta_h A_C}{\pi} - \beta k \qquad m_d = \frac{\eta_h A_C}{d^2\pi} + \beta' k \tag{2}$$

where, again, the constants β and β' are used to account for border effects. We want to find the smallest d such that the two conditions

$$m_u \geq 2t \qquad \text{and} \qquad m_d \leq 3t$$

hold. Plugging (1) and (2) into the resulting system of inequalities and solving for d yields the desired result. □

Observe that a cage can indeed be realized having two single-chains or one double-chain embedded in its interior by packing the chains in a "snake-like" fashion. This is possible regardless from which direction the chains are entering the cage.

3.3 Components

In this section, we describe the realization of the various components shown in Figure 1, starting with the basic *wire*. We then go on to describe *clauses*, *variables* and *crossings*. Note that the components used in the proof of [3] serve the same purposes (and bear the same names) as the components described in the sequel. Their construction, particularly of the variable and crossing component, is different since it is built on multiple types of cages differing from the ones used in this paper.

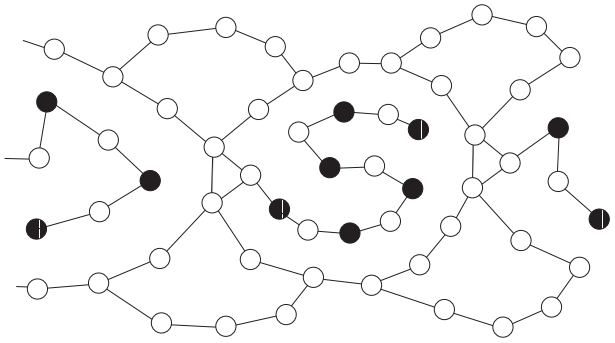

Figure 3: An oriented wire with mortar.

3.3.1 Wire:

A *wire* connects the various components of the grid drawing to one another and hence serves as the *directable path* between them. A wire consists of a sequence of cages hooked together, and a double-chain at each connecting link. In order to make sure that this chain indeed embeds in one of the two adjacent cages, we borrow an idea from [3] and add *mortar* around the hinge vertices, as shown in Figure 3. Mortar forces the chain to embed in one of the two cages. This intuition is concretized in the following lemma.

LEMMA 3.7. *In any realization $r(G)$ of G_C as a d-QUDG with $d \geq 1/\sqrt{2}$, the chain between two cages is embedded entirely in one of the cages.*

PROOF. Let v_1 be the first independent node in the chain. Let I and I' denote the interior areas of the two adjacent cages, and let I_M be the interior of a corresponding mortar. If v_1 is in I or I', the rest of the chain is embedded in the same cage by Lemma 3.4. Now, assume for contradiction that v_1 is located outside of I and I'. If v_1 is in I_M, one of the subsequent chain vertices must cross an edge of either the mortar or a cage as the capacity of I_M is too small. This leads to a contradiction with Lemma 3.4 and the assumption that each chain vertex is independent of all cage and mortar vertices. Similarly, if v_1 is outside of I, I', and I_M, there must be a crossing between one of the hinge edges and a cage edge. □

Mortars are also used in the remaining components where necessary. It is easy to see that for large enough k, the mortars of different "connections" do not overlap or interfere, because the number of vertices per mortar is constant. This is the case even if a cage has four adjacent cages, one in each direction. Further, note that *corners* (i.e., wire components where the direction of the grid-drawing path turns by 90 degrees) are built the same way as wires, the only difference being the placement of the hinge.

We write that a wire is oriented towards the clause if all chains on this wire are embedded in the adjacent cage closer to the corresponding clause component. The wire is oriented towards the variable if all chains are embedded in the adjacent cage closer to the variable component. The behavior of wires is summarized in the following lemma.

LEMMA 3.8. *In any realization $r(G)$ of G_C as a d-QUDG with*

$$d \;\geq\; \sqrt{2/3} + O(1/k),$$

a wire is uniquely oriented *towards either the clause component or the variable component.*

PROOF. Once one chain in the wire is embedded in the cage closer to the either the clause or the variable component, all other chains in the same wire must be oriented towards the same direction by Lemmas 3.6 and 3.7. □

For the description of the remaining components, we abstract cages as octagons (mortars are omitted for the sake of clarity). Chains are denoted by lines with one or two filled dots, corresponding to single-chains or double-chains, respectively.

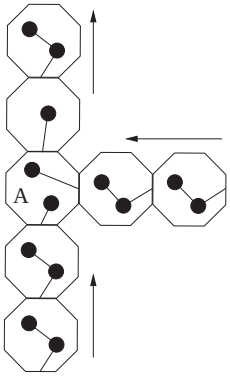

Figure 4: Clause component.

3.3.2 Clause component:

For the *clause component*, we can easily adapt the construction given in [3], merely replacing the various kinds of cages with our cage introduced in Section 3.2. As shown in Figure 4, the central cage A can contain exactly two of the

three adjacent single-chains. Hence, in accordance to the requirement imposed by G_C^{SAT}, at least one terminal must be directed away from the component. In case a clause contains less than three variables, some of the terminals may be *capped*. Such a capped terminal can easily be constructed by reducing the capacity of a cage, such that the double-chain must be directed towards the component.

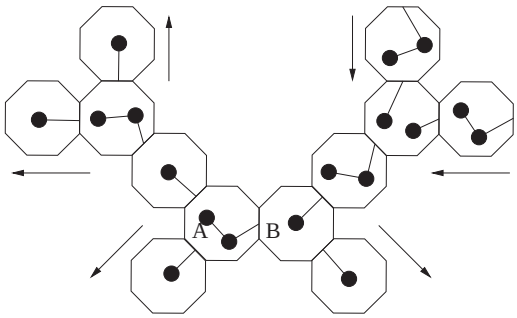

Figure 5: Variable component. The left half represents the positive literal, the right half the negative literal. All terminals of the positive literal are directed away from the component.

3.3.3 Variable component:

The *variable component* (see Figure 5) contains the two *literal components*. It must guarantee that all terminals are directed away from one of its two literal components, i.e., that the variable can be set either true or false. Assume that one negative terminal (i.e., on the right side) is directed towards the component. It follows that at least one single-chain is embedded in cage B, thus forcing the double-chain between A and B into A. This consequently forces all positive terminals to be directed away from the terminal. By symmetry, the same holds in the other direction as well.

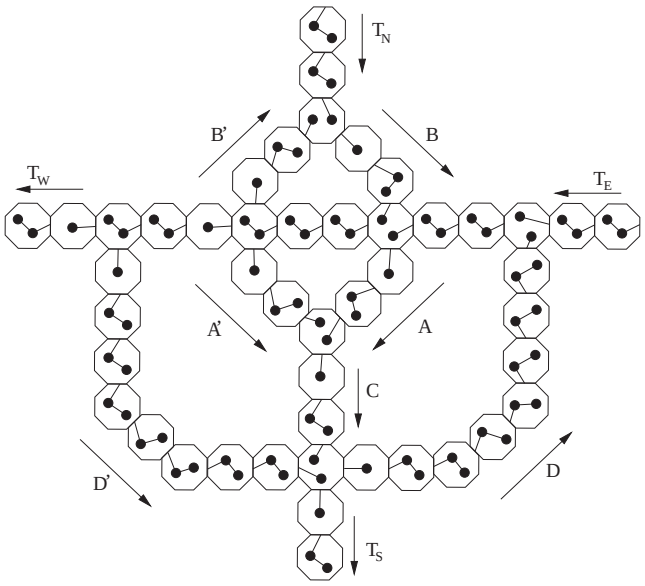

Figure 6: Crossing component.

3.3.4 Crossing component:

The most challenging component is the crossing component depicted in Figure 6. Note that the crossing component given in [3] cannot be used because it uses cages of greater capacity than two.

LEMMA 3.9. *The crossing component ensures that, if the T_S (resp. T_W, T_N, T_E) terminal is oriented towards the component, then T_N (resp. T_E, T_S, T_W) is oriented away from it. However, it is possible for T_N (T_E) and T_S (T_W) to be simultaneously directed away from the component.*

PROOF. Let $\mathcal{H}$ denote the horizontal path between T_W and T_E. Let $d(\mathcal{R}):\uparrow$ and $d(\mathcal{R}):\downarrow$ denote that wire $\mathcal{R}$ is directed in northern and southern direction, respectively. In the example drawn in Figure 6, the following properties hold: $d(\mathcal{A})$, $d(\mathcal{A}')$, $d(\mathcal{B})$, $d(\mathcal{C})$, $d(\mathcal{D}'):\downarrow$, and $d(\mathcal{B}')$, $d(\mathcal{D}):\uparrow$.

We start with the simpler, horizontal direction. Assume T_E is directed towards the component as in Figure 6. All chains on $\mathcal{H}$ must point to the left (because each cage has capacity 2) and it follows that T_W must be directed away from the crossing. By symmetry, the same holds in the opposite direction.

Now, consider the vertical direction and assume T_S is directed towards the crossing. It follows that $d(D):\uparrow$ or $d(D'):\uparrow$. Observe that it is not possible that *both* D and D' are directed north since this would lead to a contradiction on $\mathcal{H}$. Whether D or D' is directed upwards is imposed by the direction of the horizontal terminals. In case both horizontal terminals are directed away from the component, a realization arbitrarily "chooses" one of the two. Assuming w.l.o.g. that $d(D):\downarrow$ and $d(D'):\uparrow$, it follows $d(B'):\downarrow$ and $d(A'):\uparrow$. Combining $d(B'):\downarrow$ with $d(C):\uparrow$) yields $d(B):\uparrow$ and consequently, $d(A):\uparrow$. By $d(A):\uparrow$ and $d(A'):\uparrow$, we have the T_N is directed away from the crossing.

Last, we have to show that the crossing works if T_N is directed towards the component. Again, it is not possible to have both $d(D'):\uparrow$ *and* $d(D'):\uparrow$. In case $d(D'):\downarrow$ and $d(D'):\downarrow$, it clearly holds that T_S is directed south. Hence, it remains to analyze the cases for which exactly one of D and D' is directed upwards (again, a directed horizontal terminal decides which of the two is directed upwards). Assuming w.l.o.g. that $d(D):\downarrow$ and $d(D'):\uparrow$ forces the part of $\mathcal{H}$ between the two outermost cages to be directed towards the west and therefore, it follows $d(B'):\downarrow$ and $d(A'):\uparrow$. $d(A'):\uparrow$ and $d(T_N):\downarrow$ leads to $d(A):\downarrow$ and consequently $d(B):\downarrow$. In combination with $d(B'):\downarrow$, this results in $d(C):\downarrow$ and finally, $d(T_S):\downarrow$. The case $d(D):\uparrow$ and $d(D'):\downarrow$ is symmetric. □

Having described all components, it is now straightforward to conclude the lower bound proof.

PROOF OF THEOREM 3.1. From the above description of the components, it follows that G_C can be realized as a d-unit disk graph with $d \geq \sqrt{2/3} + \epsilon$ if and only if the underlying grid drawing is orientable. By Lemma 3.3, this is the case if and only if the underlying SAT instance C is satisfiable. The construction of G_C is clearly polynomial in the number of clauses and variables and hence, approximating the optimal $q(r(G))$ within $\sqrt{3/2} - \epsilon$ is NP-hard. □

Our lower bound reduction can be used to prove a slightly more general result which is summarized by the following corollary.

COROLLARY 3.10. *Given a d-quasi unit disk graph G, it is NP-hard to find a realization of G as a d′-quasi unit disk graph with* $d' \geq 1/\sqrt{2}$ *such that,* $d' \geq (\sqrt{2/3} + \epsilon)d$.

PROOF. The corrollary can be proven using exactly the same techniques as in the prove of Theorem 3.1. The restriction $d' \geq 1/\sqrt{2}$ stems from Lemma 3.4. For details, we refer to the full version of the paper. □

Note that for $d = 1$, Corollary 3.10 is equivalent to Theorem 3.1.

Finally, it is straightforward to obtain another impossibility result. It is a strengthening of [3] in the sense that the impossibility of finding a perfect embedding is not only restricted to unit disk graph, but even to quasi unit disk graphs for large enough d.

COROLLARY 3.11. *Given a graph G, it is NP-hard to determine whether G can be realized as a d-quasi unit disk graph with* $d \geq 1/\sqrt{2}$.

PROOF. The proof follows from the lower bound construction. Based on a 3SAT instance, we construct a graph G_U which is a unit disk graph if the 3SAT instance has a satisfying assignment. If the 3SAT instance is not satisfiable, G_U cannot be embedded as a $(\sqrt{2/3} + \epsilon)$-quasi UDG. It follows that it is NP hard to decide whether a given graph is a d-quasi UDG for $d \geq \sqrt{2/3} + \epsilon$. Applying the same reasoning to the construction of the proof of Corollary 3.10, yields graphs G_d for which it is NP hard to decide whether G_d is a d-quasi UDG for any $d \geq 1/\sqrt{2}$. □

4. CONCLUSION

In this paper, we have given the first inapproximability result for the problem of embedding a unit disk graph into the Euclidean plane. Besides being of theoretical interest in the field of computational geometry, our result has direct consequences for the study of virtual coordinates in ad hoc and sensor networks. It places a bound on how well virtual coordinates can be derived from connectivity information alone. Currently, the gap between the best known approximation algorithm achieving an approximation ratio of $O(\log^{2.5} n\sqrt{\log\log n})$ [10] and our lower bound of $\sqrt{3/2}-\epsilon$ is still glaring and we believe that neither of the current bounds is tight. Considering the great potential of virtual coordinates in a variety of applications — particularly in ad hoc and sensor networks —, diminishing the chasm between upper and lower bound promises to be an interesting field for future research.

5. REFERENCES

[1] L. Barrière, P. Fraigniaud, and L. Narayanan. Robust Position-based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges. In *Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 19–27. ACM Press, 2001.

[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad hoc Wireless Networks. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 1999.

[3] H. Breu and D. G. Kirkpatrick. Unit Disk Graph Recognition is NP-hard. *Computational Geometry. Theory and Applications*, 9(1-2):3–24, 1998.

[4] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[5] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical, Distributed Network Coordinates. *SIGCOMM Comput. Commun. Rev.*, 34(1):113–118, 2004.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[7] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proceedings of Symposium on Principles of Distributed Computing (PODC)*, 2003.

[8] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-Hoc Networks Beyond Unit Disk Graphs. In *Proceedings of 1st Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 69–78. ACM Press, 2003.

[9] Z. Lotker, M. M. de Albeniz, and S. Perennes. Range-Free Ranking in Sensors Networks and Its Applications to Localization. In *Proceedings of 3rd Ad-Hoc, Mobile, and Wireless Networks: Third International Conference (ADHOC-NOW)*, pages 158–171, 2004.

[10] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer. Virtual Coordinates for Ad hoc and Sensor Networks. In *Proceedings of 2nd Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2004.

[11] E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-based Approaches. In *Proceedings of Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[12] A. Rao, C. Papadimitriou, S. Ratnasamy, S. Shenker, and I. Stoica. Geographic Routing without Location Information. In *Proceedings of Mobile Computing and Networking (MOBICOM)*, 2003.

[13] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from Mere Connectivity. In *Proceedings of International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2003.