Localization and Routing in Sensor Networks by Local Angle Information

Jehoshua Bruck
Department of Electrical Engineering
California Institute of Technology
Email: bruck@paradise.caltech.edu.
and
Jie Gao
Department of Computer Science, Stony Brook University
Email: jgao@cs.sunysb.edu.
and
Anxiao (Andrew) Jiang
Department of Computer Science, Texas A&M University
Email: ajiang@cs.tamu.edu.

Categories and Subject Descriptors: E.1 [**Data**]: Data Structures—*graphs and networks*; F.2.2 [**Theory of Computation**]: analysis of algorithms and problem complexity—*non-numerical algorithms and problems*

General Terms: Algorithms, Design, Theory

Additional Key Words and Phrases: Sensor networks, Wireless networks, Localization, Geographical routing, Embedding, Planar spanner subgraph

1. INTRODUCTION

The fast development of sensor networks in recent years has attracted a lot of interest in the networking community. Sensor networks are closely related to the geometric environment in which they are deployed. Sensor location information is indispensable for both sensor data integrity and network organization. The nature of sensor networks is data-centric. Individual sensors are not as interesting as their sensed data. But the data from sensors are meaningless if we do not know where the data are from. Location information can also help networking operations such as routing and topology control. For example, geographical routing uses node locations to aid routing, such that a node sends the message to the neighbor who is closest to the destination. With a dense and uniform sensor deployment, geographical routing

delivers messages to the destination with high success rate in a local and efficient manner.

Traditional approaches to obtain location information include global positioning systems (GPS) [Hofmann-Wellenhof et al. 2001]. But GPS is not appropriate for large-scale sensor localization due to its high cost, large form factor and outdoor constraints. There has been a lot of study on localization algorithms that induce the locations of sensor nodes from local measurements including distance and angle estimations between neighbors [Savvides et al. 2001; Savvides et al. 2003; Niculescu and Nath 2001; 2003; 2004; Shang et al. 2003; So and Ye 2005; Biswas and Ye 2004; Moore et al. 2004; Gotsman and Koren 2004; Doherty et al. 2001]. A detailed review of these methods will be presented in the related work section. Generally localization algorithms can be classified as anchor-based and anchor-free methods. Anchor-based methods assume a (sometimes large) number of anchor nodes know their positions already [Savvides et al. 2001; Savvides et al. 2003; Niculescu and Nath 2001; 2003; Shang et al. 2003; Niculescu and Nath 2004; Doherty et al. 2001]. Sensor nodes derive their locations by using distance measurements to anchor nodes. Anchor-free methods output the relative positioning of the sensors, subject to a global translation and rotation. In this paper we focus on anchor-free methods that deduct the geometry of the network from local measurements.

Existing anchor-free algorithms take either the connectivity graph [Rao et al. 2003; Shang et al. 2003], or the distances between neighboring sensor nodes [So and Ye 2005; Biswas and Ye 2004; Moore et al. 2004; Gotsman and Koren 2004] as input. The distance between two communicating nodes can be estimated by Received Signal Strength Indicator (RSSI) or Time of Arrival (ToA) techniques. One major challenge along this approach is localization ambiguity — when the localization solution is not unique, localization algorithms may come up with a different embedding that satisfies all the distance constraints but deviates far from the ground truth. This difficulty is also confirmed by the NP-hardness of unit disk graph embedding. With purely the connectivity information, determining whether a combinatorial graph is a unit-disk graph is NP-hard, and thus finding such an embedding in the plane (with neighboring nodes embedded within distance 1 and non-neighboring nodes more than distance 1 away) is also hard [Breu and Kirkpatrick 1998]. In fact, even a relaxed version of the problem is still hard. Kuhn *et al.* proved that finding an embedding such that non-neighboring pairs are at least 1 away and neighboring pairs are within $\sqrt{3/2}$ is NP-hard [Kuhn et al. 2004]. Distance measurements of neighboring nodes do not help either [Aspnes et al. 2004; Bădoiu et al. 2004].

Interestingly, not as much work has been done on using angle information for localization. Angles between adjacent edges can be measured by using multiple ultrasound receivers [Priyantha et al. 2000], or by using directional antennas and laser transmitters. Considering angle information adds one more dimension to the localization problem. At first sight, angle information tells us how the graph stretches out in different directions and thus could be helpful in removing incorrect folding during localization. Indeed, Efrat *et al.* [Efrat et al. 2006] show that incorporating angular information can significantly improve the performance of mass-spring relaxation for sensor localization. Thus we were initially motivated to examine whether

angle information, instead of distance information, can make unit disk graph embedding polynomially solvable. The results we obtained in this paper, however, are contrary to our initial intuition.

1.0.0.1 *Our contribution..* In this paper, we study what can and what cannot be done using the connectivity together with local angle information. Given a combinatorial unit disk graph G with the angles between adjacent edges specified, we want to find a valid embedding of G in the plane. That is, we want to assign Euclidean coordinates to the vertices of G such that G is the induced unit disk graph that meets the angle constraints. We prove that this problem is hard. In addition a few relaxed versions are also hard. We show that it is NP-hard to find a $\sqrt{2}$-approximate embedding where non-neighboring nodes are embedded at least $\sqrt{2}/2$ away, or a topologically-equivalent embedding where two edges cross in the embedded graph if and only if they cross in a valid embedding.

Despite these negative results, we show two positive results that angle information is useful. First, angle information, though not sufficient to derive the global geometry, is sufficient for a topology control problem for which existing solutions all assume location information. The problem we study is to find a planar spanner subgraph that approximates the original unit disk graph. Given a unit disk graph, we would like to prune edges such that the remaining graph is planar and the shortest path between any two nodes in the subgraph is at most a constant factor longer than that in the original graph. Spanner subgraphs are useful in topology control and geographical routing. In particular, geographical routing uses face routing on a planar subgraph to guide a packet out of the local minima. Existing work on constructing planar spanner subgraphs of unit disk graphs (please refer to [Rajaraman 2002] for an overview) all assume that locations are already known. Here we show that with local angle information we can find a subgraph G' of G such that for *any* valid embedding $\mathcal{E}$ of G, the graph $\mathcal{E}(G')$ induced by the same embedding is a planar spanner of $\mathcal{E}(G)$. No two edges cross in $\mathcal{E}(G')$ and the shortest path distance between two nodes in $\mathcal{E}(G')$ is at most a constant factor of that in $\mathcal{E}(G)$.

The significance of this result is in two folds. First, to identify the edges of a planar spanner subgraph, one does not need the node locations, only the local angle information suffices! Further, any straight line embedding of the combinatorial graph G' in the plane, not necessarily a valid embedding of the unit disk graph (which is NP-hard to compute), gives a set of virtual coordinates for sensor nodes with which geographical routing is guaranteed to deliver a packet to its destination if such a path exists. Thus, just for the purpose of geographical routing, using accurate location information is unnecessary. Secondly, this observation can be useful in practice to improve the robustness of planar spanner subtraction, especially when localization is not accurate.

For practical localization, we propose an embedding algorithm with local angle information that gives surprisingly good results. We first formulate the embedding problem by a linear program with relaxed constraints such that any valid embedding must be a feasible solution to the LP. Through simulations, we show that the LP finds an almost identical set of locations as the original ones, even when the graph is sparse. We also show that the method is robust to both noisy measurements of angles and different models of sensor networks (e.g., quasi-unit disk graph models).

A planar spanner derived based on local angle information equipped with the virtual coordinates obtained through this practical embedding enables geographical routing and approximate shortest path routing with demonstrated performance almost the same as that of using the real locations.

2. RELATED WORK

In this section we give a quick overview of existing localization algorithms and the use of location information in geographical routing.

2.1 Localization algorithms

A number of anchor-based localization algorithms use iterative triangulation or its variants. Anchor nodes obtain their locations by GPS or as pre-specified. Then trilateration is used to find the locations of other sensors progressively [Savvides et al. 2001; Savvides et al. 2003; Niculescu and Nath 2001]. If the distances from a sensor p to three anchors are known, the location of p is determined and p becomes a new anchor node. Similar methods can also be done by using angles [Niculescu and Nath 2003; 2004]. There are two issues that need special care for these incremental solutions. One is to deal with cascading error accumulation in large-scale networks. One can adopt optimization techniques such as mass-spring relaxation to smooth out error distribution, or, adopt robust statistics to handle outliers in input measurements [Li et al. 2005]. The other issue is to handle insufficient number of initial anchor nodes. If the number of anchors is too small or the anchors are not well distributed, some nodes may not be able to find three neighboring anchor nodes to locate themselves. In this case, one can use distance estimations to anchor nodes via multi-hop paths [Niculescu and Nath 2001] or adopt collaborative multilateration by solving a larger optimization problem [Savvides et al. 2001; Savvides et al. 2003].

Using range information and local optimization such as mass-spring relaxation techniques can often get stuck at local minima with part of the network flip over the rest and generate a network layout far away from the ground truth [Efrat et al. 2006]. To deal with localization ambiguity, Moore *et al.* [Moore et al. 2004] proposed to use robust quadrilaterals as the basic iterative operation. A quadrilateral on four nodes with all pairs of edges is a globally rigid component with a unique realization. The global layout is obtained by gluing locally identified robust quadrilaterals. Similarly, with ideas from rigidity theory to improve iterative multilateration on sparse networks, Goldenberg *et al.* proposed recording, propagating and verifying multiple possible locations of sensors to discover the truth network layout [Goldenberg et al. 2006].

Global optimization techniques for sensor network localization include multidimensional scaling (MDS) [Borg and Groenen 1997; Shang et al. 2003] and semidefinite programming [So and Ye 2005; Biswas and Ye 2004]. They formulate the problem as solving a global optimization problem for the sensor locations such that the distance constraints are satisfied. The results are typically better than local optimization algorithms. But these are centralized solutions. Comparison of different localization algorithms have been evaluated in real sensor deployment [Whitehouse et al. 2005; Whitehouse and Culler 2006].

From a theoretical point of view not much is known on approximation algorithm for unit disk graph embedding. So far the only known theoretical result is an algorithm with an upper bound $O(\log^{2.5} n\sqrt{\log\log n})$ on the ratio of the longest distance between neighboring pairs to the shortest distance between non-neighboring pairs [Moscibroda et al. 2004].

2.2 Geographical routing

Node location information enables geographical routing. A source node knows the location of the destination and uses it for guidance [Karp and Kung 2000; Bose et al. 1999; Kuhn et al. 2003]. In the simplest form (greedy forwarding), a message is forwarded to the neighbor whose Euclidean distance to the destination is the minimum among all neighbors. When a message gets stuck at a node whose neighbors are all further away from the destination, it uses perimeter routing (or face routing) to route along the faces of a planar subgraph until either the destination is reached or greedy forwarding can be performed again. Both the node location information and a correctly constructed planar subgraph are needed.

Due to the hardness of the localization problem, recent researchers proposed to use virtual coordinates in replace of the real coordinates. The idea is first proposed by Rao *et al.* [Rao et al. 2003], where they construct a set of virtual coordinates by using only the connectivity for geographical routing. But when a message gets stuck at a local minima, the only way for it to reach the destination is to be flooded to the whole network. The results in the second half of this paper can be considered as computing virtual coordinates for routing. We use more information, the local angle information, and produce an embedded planar spanner subgraph together with a set of virtual coordinates such that stuck messages can be routed to the destination by perimeter routing.

3. PRELIMINARIES

We start with some definitions on unit disk graphs and embeddings. Throughout the paper we assume that the UDG is connected since otherwise we'll work on each connected component separately.

Definition 3.1. *A* unit-disk graph *is a combinatorial (unweighted) graph induced by a set of points in the Euclidean plane such that two points have an edge in between if and only if their distance is no more than 1.*

We emphasize here that by the notion of unit-disk graph we mean the combinatorial graph without the embedding. Such a unit-disk graph is induced by a set of points in the Euclidean plane but the configuration of the nodes in $\mathbb{R}^2$ is unknown. An embedding of such a combinatorial graph in the Euclidean plane may or may not be the same as the original (unknown) configuration. For an embedding $\mathcal{E}$, we denote by $\mathcal{E}(p)$ the embedded point of a node p. The Euclidean distance between two nodes p, q in an embedding $\mathcal{E}$ is denoted by $d(\mathcal{E}(p), \mathcal{E}(q))$. We will sometimes abuse the notations and use p to represent $\mathcal{E}(p)$ when the context is clear.

In this paper we study embedding problems by using local angle information. Specifically, besides the combinatorial unit disk graph we are also given the angles between angularly adjacent edges (All angles are measured counterclockwise). See

Figure 1. With the local angles constrained there is still freedom to choose the lengths of the edges.

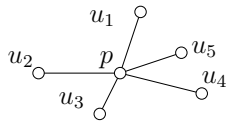


Fig. 1. For each node p in the unit disk graph, assume that $u_1, \cdots, u_k$ are p's neighbors ordered counterclockwise. In this paper we assume that the angles between edges pu_i and pu_{i+1} are given.

Definition 3.2. *An* α-approximate embedding $\mathcal{E}$ *of a graph* G *with angle information is an embedding of the vertices such that the distance between two nodes* $d(\mathcal{E}(u), \mathcal{E}(v)) \leq 1$ *if* u, v *have an edge between them in* G*, and* $d(\mathcal{E}(u), \mathcal{E}(v)) > 1/\alpha$ *if* u, v *do not have an edge between them in* G*, where* $\alpha \geq 1$*. The angle between any two adjacent edges* uv, uw *is as specified. A* valid embedding *is an* α*-approximate embedding with* $\alpha = 1$*.*

We observe that by local angle information, we can decide whether two edges cross in a valid embedding of the unit disk graph. Thus when we say two edges cross in a unit disk graph G, we actually mean that they cross in any valid embedding of G.

Lemma 3.3. *If we know the angles between adjacent edges of a unit disk graph, we can determine all pairs of crossing edges in a valid embedding.*

Proof. In particular, if two edges AB, CD intersect with each other, there must be a node that is connected with all the other three nodes [Breu and Kirkpatrick 1998; Gao et al. 2001]. Suppose B is connected with the other three nodes. Then AB, CD cross each other if and only if AB is located inside the cone defined by $\angle CBD < \pi$ and A, B are on different sides of the line defined by CD.

First we can decide if AB is located inside the cone defined by $\angle CBD < \pi$ easily by the angle information. Further, if AB is located inside the cone defined by $\angle CBD$ and A, B are on the same side of the line defined by CD, then A is inside the triangle BCD. See Figure 2 (ii). Then A is connected to B, C, D due to plane geometry. This situation can be identified with only angle information since BA must be outside the cone defined by $\angle CAD$. □

The above lemma implies that we can identify all crossing edges in a valid embedding with local angle information. Thus one relaxation of a valid embedding is to require that the topology of the embedded graph is equivalent with a valid embedding, i.e., only the edges that cross in a valid embedding are allowed to cross.

Definition 3.4. *A* topologically equivalent embedding $\mathcal{E}$ *of a graph* G *with angle information is an embedding of the vertices such that two edges cross in* $\mathcal{E}$ *if and only if they cross in a valid embedding. The angle between any two adjacent edges* uv, uw *is as specified.*

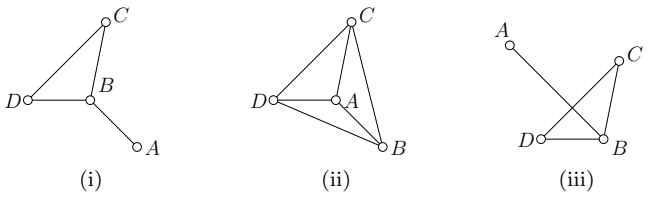


Fig. 2. (i) The edge AB is not located inside the angle $\angle CBD$ and thus AB, CD cannot cross each other; (ii) AB is located inside the cone defined by $\angle CBD$ and A, B are on the same side of the line defined by CD, then BA must be outside the cone defined by $\angle CAD$; (ii) A correct crossing between AB and CD.

Remark. We notice that without loss of generality we can assume that in a topologically equivalent embedding the neighboring nodes are embedded no further than distance 1. This is because we can always do proper global scaling that does not change the topology of the embedded graph.

Theorem 3.5. *A $\sqrt{2}$-approximate embedding is a topologically equivalent embedding.*

Proof. Assume that there are two edges AB, CD that cross each other in a $\sqrt{2}$-approximate embedding $\mathcal{E}$. Also assume that $\mathcal{E}^*$ is a valid embedding. If the following two claims are true, then $\mathcal{E}$ is topologically equivalent with $\mathcal{E}^*$.
Claim 1: If AB, CD cross in a valid embedding $\mathcal{E}^*$, then they must also cross in a $\sqrt{2}$-approximate embedding $\mathcal{E}$.
Proof of claim 1. If AB, CD cross in a valid embedding $\mathcal{E}^*$, then one node must be connected to all the three other nodes. There are three possible cases, as illustrated by Figure 3. For case (ii) and (iii), if the angles between adjacent edges

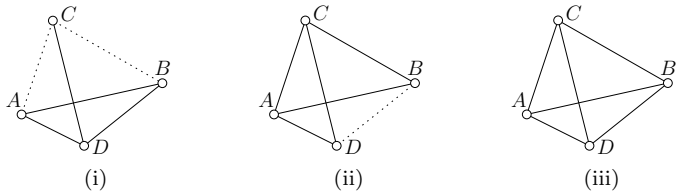


Fig. 3. In a valid embedding of the unit disk graph G, if two edges AB, CD cross each other, there are only three possible cases.

are fixed as specified, the configuration of the four nodes is unique up to a global rigid motion and scaling. Thus AB, CD cross in any embedding preserving the local angles. For case (i), we argue that in a $\sqrt{2}$-approximate embedding AB, CD must also cross each other. In a valid embedding $\mathcal{E}^*$ as in Figure 4 (i), AC must be longer than both AD and CD. Thus the angle $\angle CDA > \pi/3$. Similarly $\angle BDC > \pi/3$. Thus $\angle BDA > 2\pi/3$. If in a $\sqrt{2}$-approximate embedding $\mathcal{E}$, AB does not cross CD, then C is embedded inside the triangle ADB, as shown in Figure 4 (ii). First $\angle BCA > \angle BDA > 2\pi/3$. On the other hand, $d(\mathcal{E}(A), \mathcal{E}(C)) > \sqrt{2}/2$,

$d(\mathcal{E}(B), \mathcal{E}(C)) > \sqrt{2}/2$, $d(\mathcal{E}(A), \mathcal{E}(B)) \leq 1$. Thus,

$$d(\mathcal{E}(A), \mathcal{E}(C))^2 + d(\mathcal{E}(B), \mathcal{E}(C))^2 > 1 \geq d(\mathcal{E}(A), \mathcal{E}(B))^2.$$

So $\angle BCA < \pi/2$. This leads to a contradiction.

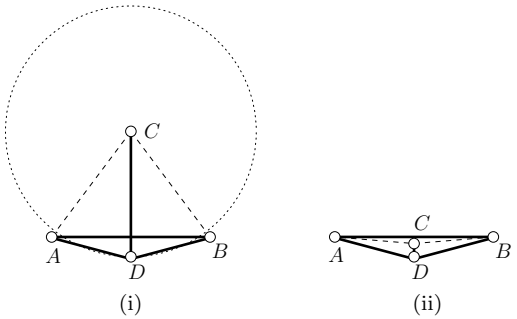


Fig. 4. (i) A valid embedding $\mathcal{E}^*$; (ii) A $\sqrt{2}$-approximate embedding $\mathcal{E}$.

Claim 2: If AB, CD cross in a $\sqrt{2}$-approximate embedding $\mathcal{E}$, then they must also cross in a valid embedding $\mathcal{E}^*$.

Proof of claim 2. There are six possible cases based on how the nodes are connected with each other in G. See Figure 5. We argue that none of the cases has both properties that

—AB, CD intersect each other in $\mathcal{E}$ and
—AB, CD do not intersect each other in $\mathcal{E}^*$.

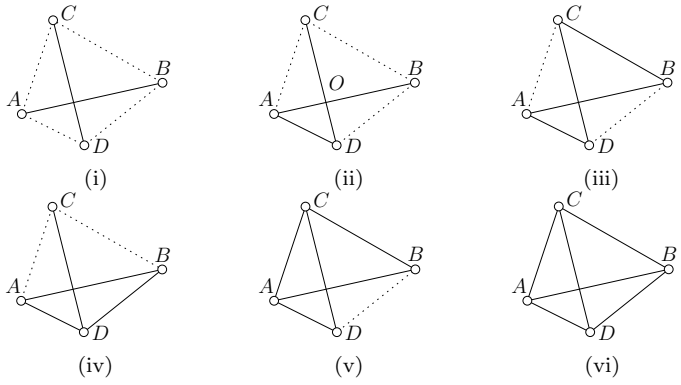


Fig. 5. A $\sqrt{2}$-approximate embedding $\mathcal{E}$. Solid lines are edges in G.

(1) For case (i) in Figure 5, let's take a look at triangle $\triangle ACD$ under embedding $\mathcal{E}$. We know that $d(\mathcal{E}(A), \mathcal{E}(C)) > \sqrt{2}/2$, $d(\mathcal{E}(A), \mathcal{E}(D)) > \sqrt{2}/2$, $d(\mathcal{E}(C), \mathcal{E}(D)) < 1$. So the angle $\angle CAD < \pi/2$. Similarly, $\angle ACB < \pi/2$, $\angle CBD < \pi/2$, $\angle BDA < \pi/2$. This leads to a contradiction since the sum of the inner angles of a 4-gon must be 2π. So this case can never happen in $\mathcal{E}$.
(2) Case (ii) cannot happen for a $\sqrt{2}$-approximate embedding $\mathcal{E}$. The intuition is that if the two edges do not cross in a valid embedding, then the angle $\angle COB \leq \pi/6$. This contradicts with the fact that $d(\mathcal{E}(B), \mathcal{E}(C)) > \sqrt{2}/2$. The details are in Appendix 8.
(3) Case (iii) cannot happen. By the angle constraint, the two edges AB, CD must cross in any planar embedding. But in a valid embedding there must be a node that is connected to three other nodes. This leads to a contradiction.
(4) For cases (iv), (v) and (vi), AB and CD cross in any valid embedding.

Therefore if two edges do not cross in a valid embedding, they cannot cross each other in any $\sqrt{2}$-approximate embedding. This shows that an $\sqrt{2}$-approximate embedding is a topologically equivalent embedding. □

4. THE HARDNESS OF UDG EMBEDDING WITH ANGLES

As shown in the last section, by using local angle information we can decide on all crossing edges in a valid embedding. However, local angle information is not sufficient to determine a valid embedding. It turns out that the problem of finding a valid embedding by using the network connectivity and the local angle information is still hard. In fact it is even NP-hard to find a topologically equivalent embedding or a $\sqrt{2}$-approximate embedding. In this section we show a polynomial reduction from the 3SAT problem, such that any 3SAT instance can be turned into a problem of embedding a unit disk graph in the plane with angle constraints. Therefore, as long as we can solve the UDG embedding problem we can solve 3SAT in polynomial time, which then establishes the hardness result.

A 3SAT problem consists of a set of Boolean variables and a set of clauses such that each clause is composed of at most 3 literals, which are either negated or un-negated variables. The 3SAT problem is to find an assignment to the variables such that all the clauses are satisfied (i.e., having value 1). For example, one instance of a 3SAT problem has three variables x_1, x_2, x_3 and three clauses: $\overline{x_1} \vee x_2 \vee \overline{x_3}$, $x_2 \vee \overline{x_3}$, $x_1 \vee \overline{x_2} \vee x_3$. One asks for the question whether there exists a 0, 1 assignment to the variables x_1, x_2, x_3 such that all the clauses have value 1; and if so the instance is called satisfiable.

A 3SAT instance C can be represented as a graph G_C where the set of clauses and variables are drawn in the plane as boxes, and there is a path connecting a clause with a variable (or its negated version) if the variable appears in the clause. Please see Figure 6 for an example. Such a graph can be drawn on a grid in polynomial time [Breu and Kirkpatrick 1998]. Breu and Kirkpatrick proved the NP-hardness of unit disk graph embedding by a reduction from a 3SAT problem [Breu and Kirkpatrick 1998]. We use a different reduction to show with angle information the unit disk graph embedding is still hard. Specifically, given *any* 3SAT instance C represented by a graph G_C, we will realize G_C by a unit-disk graph with angle constraints such that there is a topologically equivalent embedding if and only if

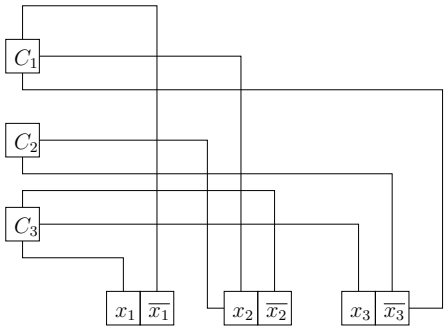


Fig. 6. The graph G_C of a 3SAT instance $(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$.

the corresponding 3SAT problem is satisfiable. Now if there exists a polynomial algorithm to solve the UDG embedding with angles, one can apply the polynomial algorithm to solve 3SAT. This will establish that the UDG embedding with angles is NP-complete.

The rest of the section is devoted to the following task. Given any 3SAT instance C, come up with a unit disk graph G and angle constraints as the graph representation of C such that G has a UDG embedding if and only if the 3SAT instance C is satisfiable.

4.1 Realization of G_C by unit disk graphs

We first present a set of building blocks by using unit disk graphs. These building blocks will be used to realize a 3SAT instance.

—**Spring**. A spring is a line segment with length between ℓ and 2ℓ. It can be realized by a set of $2\ell+1$ nodes placed on a straight line with only edges between adjacent pairs of nodes, as shown in Figure 7 (ii). The angles between two edges adjacent to one node is fixed as π. In particular, each edge in a unit disk graph has length at most 1, so a chain of $2\ell + 1$ nodes have length at most 2ℓ. For 3 adjacent nodes a, b, c on the chain, since a cannot communicate with c, their distance must be at least 1 away. Thus a chain of $2\ell+1$ nodes is no shorter than ℓ.

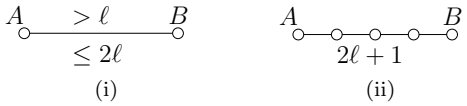


Fig. 7. (i) A spring; (ii) The realization of a spring by unit-disk graphs.

—**Amplifier**. An amplifier is a triangle with fixed inner angles. Thus the ratio between the edge lengths of the triangle is fixed. For a number ℓ we use an amplifier to get the number $\ell' = c \cdot \ell$ for any $c > 0$. An amplifier can be realized by a unit disk graph with pre-specified angles between adjacent edges. In particular, each edge in the triangle is realized as a chain of nodes. There might be possible

edges between nodes in different chains, depending on the magnitude of the inner angles. See Figure 8 for an example.

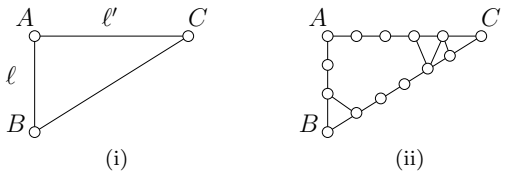


Fig. 8. (i) An amplifier; (ii) The realization of an amplifier by unit-disk graphs.

—**Propagator and Crossing Propagator**. A propagator is a rectangle (with all inner angles specified as $\pi/2$). The lengths of the opposing sides of the rectangle are thus the same. It can be implemented by a cycle of nodes with corresponding angle constraints. A crossing propagator is a pair of crossing rectangles. See Fig. 9.

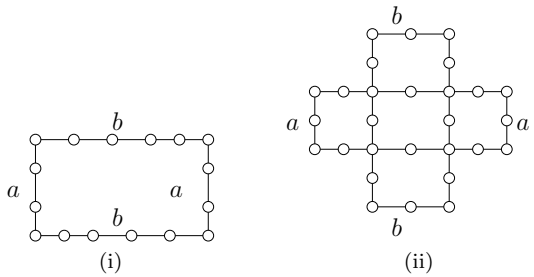


Fig. 9. (i) Propagator; (ii) Crossing propagator.

Now we are ready to explain how to realize the graph G_C for a 3SAT instance C by using unit disk graphs with angle constraints. The graph G_C consists of three components: clauses, variables and wires to connect them.

—**0/1 block (variable component)**. By using the above building blocks, we can construct a 0/1 block that has only two types of valid embedding. In short, we construct a concave cycle with one upper "tooth" and one bottom "tooth" with no edges in between. With propagators and amplifiers, we are able to force the teeth to be relatively large. Since the teeth cannot overlap (to keep the embedding topologically correct), there are basically two ways to embed the concave cycle, either by putting the upper tooth to the left of the bottom tooth, or the other way around. Please see Figure 10 for the two types of embedding with the two teeth highlighted in thick lines. The concave cycle is bounded by $AEFGHDCKLIJB$, the upper tooth is the part of the cycle $EFGH$, the bottom tooth is the part of the cycle $JILK$. The rest of the stuff on the peripheral of the concave cycle, consisting of propagators and amplifiers, makes sure that the size of the teeth is relatively large.

Now suppose the length of $AB = CD$ is ℓ, we use amplifiers and propagators such that the length of $BC = DA = 11\ell/6$. There are two squares $EFGH$, $IJKL$ inside the rectangle $ABCD$. Both of them have side length $2\ell/3$. The two squares do not have edges in between. Thus any embedding without incorrect crossings will have to embed the graph in two ways, either by putting the square $EFGH$ to the left of $IJKL$ or the other way around. In the first case, the length of the path AE is no more than $\ell/2$, the length of HD is at least $2\ell/3$. In the second case, the length of the path AE is at least $2\ell/3$, and the length of HD is no more than $\ell/2$. The segments AE, HD, BJ, KC are springs, thus their lengths can be stretched and shrunk by a factor no more than 2.

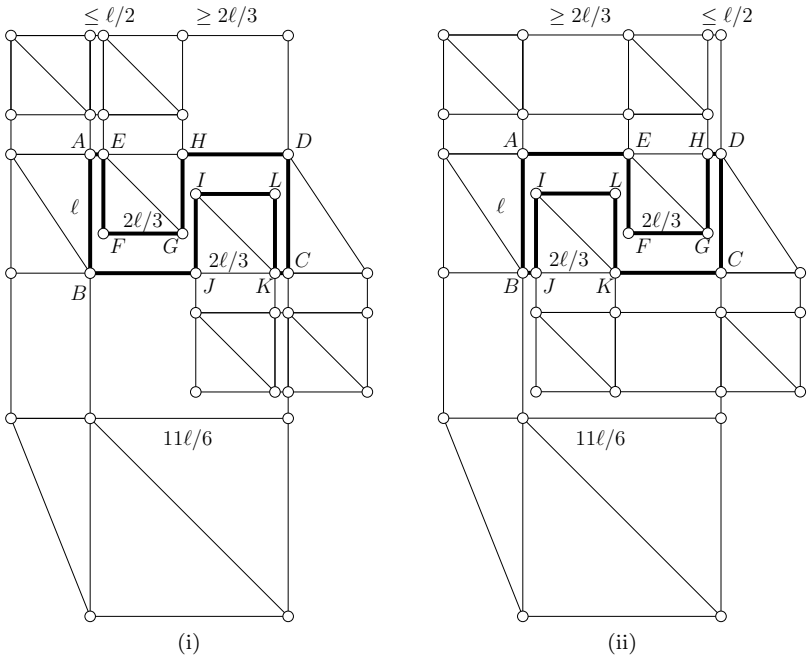


Fig. 10. The only two embedding of a concave cycle without incorrect crossings.

A variable component, including x and its negation $\overline{x}$, is implemented by a 0/1 block. In fact, we use the length of AE to represent the value of a variable x and the length of HD to represent its negated version $\overline{x}$. A variable v is assigned 1 if the length of AE is less than $\ell/2$, and 0 if the length of AE is at least $2\ell/3$. Correspondingly we use the length of HD to represent the negated variable $\overline{v}$. By the previous construction the value of a variable is always different from the value of its negated version.

—**Clause components.** A clause component puts constraints on the input variables. In particular, it enforces a total maximum length on the concatenation of springs whose lengths represent the assignments of input variables. See Figure 11 (i) for an example. If a clause is composed of three variables, then the outer rectangle has height $11\ell/6$. Thus at least one of the variable has length less

than $\ell/2$. That is, the clause is satisfied if at least one variable is assigned value 1. The clauses with two or one variables are designed similarly with a maximum height of $7\ell/6$ and $\ell/2$ respectively. See Figure 11 (ii) and (iii). To enforce the maximum height of a clause component, we use propagators and amplifiers.

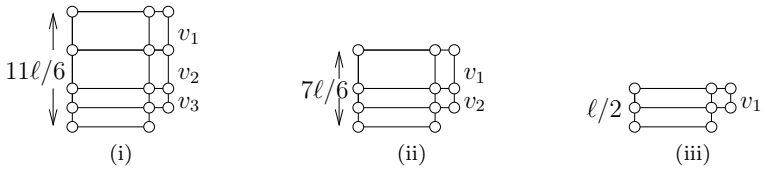


Fig. 11. Clause components (i) $(v_1 \vee v_2 \vee v_3)$; (ii) $(v_1 \vee v_2)$; (iii) v_1.

—**Wires.** The wires connect the variables with the clauses. See Figure 12. The width of the wires indicates the assignment of a variable. If the width of a wire is no more than $\ell/2$, this means the variable connected by the wire is assigned '1'. If the width of a wire is at least $2\ell/3$, the variable connected by the wire is assigned '0' in G_C. The wires are built by propagators.

Now we put all the components together and show a realization of the graph G_C (Figure 6) for a 3SAT instance C by a unit disk graph in Figure 12. This figure omits the details in the variable components. Intuitively, the hardness of the problem comes from that the ways to embed the 0/1 blocks are affected by each other through the constraints enforced by the clauses.

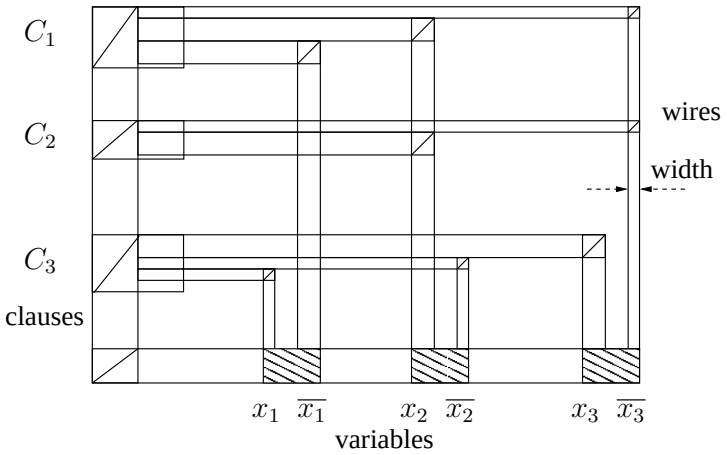


Fig. 12. The realization of a 3SAT instance $(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$ by a unit disk graph. Shaded areas are 0/1 blocks for variables. In this example $x_1 = 1$, $x_2 = 0$, $x_3 = 0$. The instance is satisfied.

Input	Hardness	ref.
UDG graph only	NP-hard	[Breu and Kirkpatrick 1998; Kuhn et al. 2004]
$O(1)$-hop distances	NP-hard	[Aspnes et al. 2004]
$O(1)$-hop angles	NP-hard	this paper
$O(1)$-hop angles & distances	in P	this paper
$\Omega(n^2)$ pairs distances	in P	[Biswas and Ye 2004; So and Ye 2005]
all pairs angles	in P	this paper

Fig. 13. A summary of the hardness of finding a valid embedding of a UDG.

4.2 Hardness results

Now we are ready to prove the NP-hardness of unit disk graph embedding with local angle information.

Theorem 4.1. *It is NP-hard to find a topologically equivalent embedding of a unit-disk graph with local angle constraints.*

PROOF. By the construction of G_C for a 3SAT instance G_C, we can see that the the instance C can be satisfied if and only if we can find an embedding of G_C in the plane that has the same topology and preserves all the local angles. Since 3SAT is NP-hard, It is also NP-hard to find a topologically equivalent embedding. □

Corollary 4.2. *It is NP-hard to find a valid embedding of a unit-disk graph with local angle constraint.*

PROOF. The proof is similar with the above theorem. For a graph G_C of a satisfiable 3SAT instance C, we can find an embedding $\mathcal{E}$ of G_C with no incorrect crossings. Further we can do proper scaling and local arrangement of $\mathcal{E}$ such that $\mathcal{E}$ is a valid embedding. □

Corollary 4.3. *It is NP-hard to find an α-approximate embedding of a unit-disk graph with local angle constraints, for $\alpha < \sqrt{2}$.*

PROOF. We construct a graph G_C for a 3SAT instance C. By Theorem 3.5, a $\sqrt{2}$-approximate embedding is a topologically equivalent embedding. Thus if we have a $\sqrt{2}$-approximate embedding $\mathcal{E}$ of G_C, then C is satisfiable. The other direction can be proved similarly as the above proof. □

4.3 A summary of hardness of localization

Localization by using only angles between adjacent edges in a unit disk graph is shown to be NP-hard. However, if we have more information, localization can be solved easily from a theoretical point of view. For example, if we have the angles between all pairs of nodes in the graph, then the graph is basically determined up to a scaling factor. For another example, if we have both the lengths of the edges and the angles between adjacent edges in a unit disk graph, the graph is uniquely determined. A short summary of the hardness results on localization is shown in Figure 13.

5. PLANAR SPANNER CONSTRUCTION

In the previous section we've shown that by using the communication graph and local angle information, It is NP-hard to find a valid embedding of a unit disk graph. On the positive side we'll show that by local angle information we can find a planar spanner subgraph whose embedding in the plane can be used for geographical routing with guaranteed delivery.

A planar graph is a graph that can be embedded in the plane with no edge crossings. A c-spanner G' of a graph G is a subgraph of G such that the shortest path distance of u, v in G' is at most c times the shortest path distance of u, v in G, where the shortest path distance is the sum of the Euclidean length of all the edges on the shortest path. c is the spanning ratio of G'. A spanner with a constant spanning ratio is usually called a spanner. In this section we'll show that one can construct a planar spanner for a unit disk graph by using only the angles between adjacent edges. Recall that the location information is not available. Thus when we say a planar spanner we mean a subgraph G' of the input unit disk graph G such that for *any* valid embedding $\mathcal{E}(G)$, the subgraph G' on the same embedding $\mathcal{E}(G')$ is a planar spanner. Finding a spanner subgraph can be easily done without the location information, however, finding a spanner subgraph that has a planar embedding for any valid embedding of the UDG does not seem to be intuitive. The idea is to find a planar subgraph that is guaranteed to contain a restricted Delaunay graph, i.e., a subgraph of the Delaunay triangulation with all the edges longer than 1 deleted [Gao et al. 2001].

A Delaunay triangulation on a point set in $\mathbb{R}^2$ is a triangulation with "empty-circle" property: the circumcircle of any triangle has no other points inside. A restricted Delaunay graph, defined as the subgraph of the Delaunay triangulation with all the edges longer than 1 deleted, is known to be a 2.42-spanner of the unit disk graph [Gao et al. 2001; Li et al. 2002]. Now we claim that with local angle information we can find a subgraph G' of G that is planar and contains all the edges of a restricted Delaunay graph. Thus G' is a planar spanner subgraph of G with spanning ratio 2.42.

Suppose two edges AB, CD cross each other in a unit disk graph, then only one of them can possibly be a Delaunay edge due to the planar property. We show that we can decide which one is *not* a Delaunay edge by using the local angle information. To be specific, there are only three possible cases of a pair of crossing edges, as shown in Figure 3. Notice that in cases (ii) and (iii), because of the given angle information, the positions of the four nodes are unique up to a rigid motion and a scaling factor. Since the Delaunay triangulation is invariant under global scaling, there is only one possible Delaunay triangulation, which can be decided by only the angles.

For case (i), node C is at least of distance 1 away from nodes A, B. See Figure 14. We take the bisectors of the edge AD, BD, ℓ_1, ℓ_2, that intersect at a point O. O is also the center of the circumcircle of $\triangle ABD$. The lines ℓ_1, ℓ_2 divide the plane into four quadrants. Node C must be inside the same quadrant with node D since $d(\mathcal{E}(C), \mathcal{E}(D)) \leq 1 < d(\mathcal{E}(C), \mathcal{E}(A))$, $d(\mathcal{E}(C), \mathcal{E}(D)) \leq 1 < d(\mathcal{E}(C), \mathcal{E}(B))$. Thus C is inside the circumcircle of $\triangle ABD$. This implies that the edge AB is not a Delaunay edge, since it violates the "empty-circle" property of the Delaunay

triangulation.

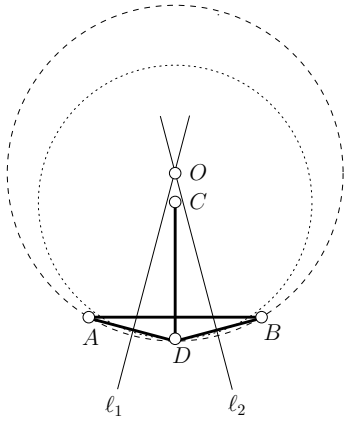


Fig. 14. Thick lines are edges in the unit disk graph. Node C must lie in the circumcircle of triangle $\triangle ABD$.

By the above argument, one can decide a non-Delaunay edge between a pair of crossing edges in a unit disk graph. Thus we can eliminate crossings by always deleting non-Delaunay edges. In the end we'll have a planar subgraph G' such that all the Delaunay edges with length no more than 1 are kept. That is, G' contains the restricted Delaunay graph, which is a constant spanner.

Theorem 5.1. *Given a unit disk graph and the angles between adjacent edges, one can construct a planar spanner subgraph with spanning ratio* 2.42.

We should also notice that there are possibly infinitely many valid embeddings of a particular unit disk graph that satisfies the angle constraints. However, the planar spanner we found is the same for all such embedded graphs. This is a little counter-intuitive since Delaunay triangulation has been considered to be very delicate – a tiny movement of a single point can possibly change the whole graph structure. Yet we show that the restricted Delaunay graph has some kind of robustness. Further, such a planar spanner subgraph can help us with efficient routing in a sensor network. In particular, it can be used to produce a set of virtual coordinates for efficient geographical routing, or a set of distributed labels for approximate shortest path routing.

5.1 Geographical routing with guaranteed delivery

It is known that any planar graph has a straight line realization in the plane [Fáry 1948; Bryant 1989]. By using a straight line embedding of the planar subgraph G', each node is assigned an Euclidean coordinate that can be used in geographical routing [Karp and Kung 2000; Bose et al. 1999]. Although in our case the location information cannot be obtained unless $P = NP$, the embedded planar subgraph provides a set of virtual coordinates that are equally good for geographical routing. The virtual coordinates guarantee the delivery of a packet if possible at all.

5.2 Approximate shortest path routing

In general, graph labelling is to assign a set of distributed labels to the vertices such that the shortest path can be inferred by using only the labels of the source and destination. In particular, one can compute a set of labels, each with size at most $O(\sqrt{n}\log n)$, on the vertices of a planar graph with n vertices, due to the fact that a planar graph enjoys a $O(\sqrt{n})$ balanced separator [Gavoille et al. 2001]. The basic idea is to partition the graph recursively into pieces by small-size separators. The number of recursions is $\log n$. For a separator of a subgraph P, we compute and store distributedly the shortest path trees of P centered at all nodes of the separator. Each node has a label with size $O(\sqrt{n}\log n)$. Therefore with the planar spanner G' of the unit disk graph, we can use the above graph labelling algorithm to construct a set of labels with size $O(\sqrt{n}\log n)$ such that one can find a 2.42-approximate shortest path of G by using only the labels of the source and the destination.

6. A PRACTICAL SOLUTION TO UDG EMBEDDING AND ROUTING WITH ANGLES

Embedding a unit-disk graph is NP-hard, and it is so even when the restriction is relaxed to be finding a topologically equivalent embedding. In practice, however, we still hope to use the local angle information to find localization that well approximates the true sensor network. The planar spanner of a sensor network is certainly very useful for geographical routing and approximate shortest path routing; yet before the routing works, the spanner firstly needs to be realized in the plane where edges are embedded as straight-line segments not crossing each other. There are currently known straight-line embedding algorithms for planar graphs [Fáry 1948; Bryant 1989]; however, when such algorithms are applied to planar spanners of UDG, they distort the edge lengths and the relative positions among nodes extremely severely, and thus are not effective in practice. In this section, we show that we can construct an embedding method based on linear programming, which produces very good localization solutions; the solutions lead to nearly optimal routing performance as well. We also demonstrate the robustness of the general embedding method to noisy measurements of angles and to more general topological models of sensor networks. This shows that using local angle information to do localization and routing is practically good for sensor networks.

6.1 UDG embedding based on LP

We first consider unit-disk graphs, and formulate the embedding problem by solving a linear program. We include as many constraints as possible such that the optimization problem remains an LP. We take the length of each edge e, $\ell(e)$, as a variable. We arbitrarily pick an edge and make the x-axis be parallel to it. By the fact that we know the angle between any two adjacent edges, the *absolute angle* of every edge e — the counterclockwise angle between the positive x-axis and e — can be uniquely determined. We see every edge as the superposition of two directed edges of opposite directions, whose absolute angles differ by π. Then a valid UDG embedding satisfies the following constraints.

—**Edge-length constraint**. $\forall$ edge e, we have

$$0 \leq \ell(e) \leq 1. \tag{1}$$

—**Cycle constraint**. For any cycle that consists of edges $\{e_1, e_2, \cdots, e_p\}$, where for $1 \leq i \leq p$, the absolute angle of e_i is θ_{e_i}, there exist two constraints

$$\sum_{i=1}^{p} \ell(e_i) \cos \theta_{e_i} = 0, \tag{2}$$

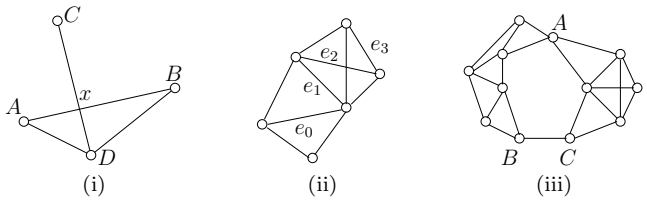
$$\sum_{i=1}^{p} \ell(e_i) \sin \theta_{e_i} = 0. \tag{3}$$

—**Non-adjacent node pair constraint**. For any two adjacent edges e_1, e_2 whose three endpoints do not induce a triangle subgraph, we have

$$\ell(e_1) + \ell(e_2) > 1. \tag{4}$$

—**Crossing-edge constraint**. For any two edges AB and CD crossing each other, one of the four nodes must be connected to all the other three. Let's say D is connected to A, B and C, and AB crosses CD at the point x (see Fig. 15(i)). Then there exists the constraint

$$\ell(CD) \geq |xD| = \ell(AD) \frac{\sin \angle DAB}{\sin(\angle ADC + \angle DAB)}. \tag{5}$$



(i) (ii) (iii)

Fig. 15. (i) Crossing-edge constraint. (ii) A subgraph where any two edges are related through a sequence of triangles. (iii) Two rigid subgraphs sharing node A and connected by edge BC.

The above constraints serve as the linear constraints in our linear program. A feasible solution to the LP gives us an embedding of the UDG, since we can use the edge lengths of a spanning tree and the angle information to determine the node positions. There are many ways to select the objective function; as a heuristic, we choose it to be maximizing the minimum length of all edges.

When the UDG has lots of edges, the large number of variables and constraints in the LP will lead to high complexity. In such cases, we can almost always use the following method to significantly reduce the complexity. First we reduce the number of variables. For any three edges AB, BC and CA that form a triangle, since the values of $\angle ABC$, $\angle BCA$ and $\angle CAB$ are given, the three edge lengths have fixed ratios. So we can regard only $\ell(AB)$ as a variable, and represent the lengths of BC and CA respectively by $c_1 \cdot \ell(AB)$ and $c_2 \cdot \ell(AB)$, for some constants c_1 and

c_2. Thus three variables are reduced to one variable. Similarly, if a subgraph of the UDG satisfies the condition that for any two of its edges e_0 and e_p, there exist edges $e_1, e_2, \cdots, e_{p-1}$ such that e_{i-1} and e_i are contained in a triangle for $1 \leq i \leq p$ (see Fig. 15(ii) for an example), then all the edge lengths in this subgraph have fixed ratios — therefore they can be represented with only one variable. We call such a subgraph a *rigid subgraph*. To push this approach further, we observe that if several rigid subgraphs share common nodes or are connected by edges, then every cycle that travels through multiple rigid subgraphs enables us to derive two equations like the *cycle constraint* described before. If there are enough such equations, the ratios among the sizes of those subgraphs and the lengths of the connecting edges can be uniquely determined — then those subgraphs and the edges between them unite and form a larger rigid subgraph, all of whose edge lengths can be represented with only one variable. (For example, see Fig. 15(iii), where two rigid subgraphs share the node A and are also connected by an edge BC. All the edge lengths there have determined ratios between themselves and therefore can be represented with only one variable.) The improvement by this approach is large. For example, when 1000 nodes are placed in a 18×18 square with a uniform distribution, the largest connected component typically contains more than 4500 edges; by the above approach, the number of variables in the LP can nearly always be reduced to be less than 30. Then the number of linear constraints can also be reduced.

The above method not only reduces complexity, but also gives us additional constraints for further guarantee on the quality of the embedding. For any two non-adjacent nodes A and B in a rigid subgraph, let $\ell(e)$ denote the edge length in the subgraph specially chosen to be the variable, then $|AB| = c \cdot \ell(e)$ for some constant c. We include the constraint $c \cdot \ell(e) > 1$ in the LP.

We have implemented the embedding algorithm and measured its performance on a variety of inputs. In the first experiment, we placed n nodes in a 15×15 square with a uniform distribution, and embed the largest connected component. The results are shown in the top part of Fig. 16, where each result is averaged over 50 experiments. In Fig. 16, *distance violation* is the number of non-adjacent node pairs that mistakenly have distance less than or equal to 1 in the embedding. d_{error} is the *minimum distance* between two non-adjacent embedded nodes that mistakenly have distance less than or equal to 1 in the embedding. (So $d_{error} \leq 1$ if such a pair of nodes exist; if no such node pair exists, we let $d_{error} = 1$). *Extra crossing* is the number of edge pairs that do not cross in the true UDG but mistakenly cross each other in the embedding. Note that the other criteria for embedding are guaranteed to be satisfied by the LP method: the *edge-length constraint* guarantees that every edge has length at most 1; the *cycle constraint* guarantees that all the angles between adjacent edges are as specified; the *crossing-edge constraint* guarantees that any two edges that cross in the true UDG also cross in the embedding. In Fig. 16 some additional properties are displayed as well, where *order of graph* is the number of nodes in the embedded UDG, and *node degree* is the average degree of nodes. A typical embedding result is shown in Fig. 17.

In a second experiment, we place nodes in an annulus with external radius 7.5 and internal radius 2.5. The results are shown in the bottom part of Fig. 16. A typical embedding result is shown in Fig. 18.

	network in square				
	order of graph	node degree	distance violation	d_{error}	extra crossing
$n = 200$	33.22	3.6422	0.80	0.9728	0.00
$n = 400$	337.96	5.4512	9.68	0.7642	0.50
$n = 600$	596.82	7.9110	6.50	0.8714	0.68
$n = 800$	799.64	10.5237	1.60	0.9568	0.10
$n = 1000$	999.94	13.1944	0.68	0.9601	0.00
	network in annulus				
	order of graph	node degree	distance violation	d_{error}	extra crossing
$n = 200$	59.76	4.1810	1.70	0.9368	0.00
$n = 400$	397.30	7.4084	6.62	0.8426	0.42
$n = 600$	599.88	11.0106	0.90	0.9570	0.08
$n = 800$	799.88	14.6423	0.10	0.9909	0.00
$n = 1000$	1000.00	18.2822	0.00	1.0000	0.00

Fig. 16. Performance of embedding unit disk graphs deployed in a square and an annulus. Each result is averaged over 50 experiments.

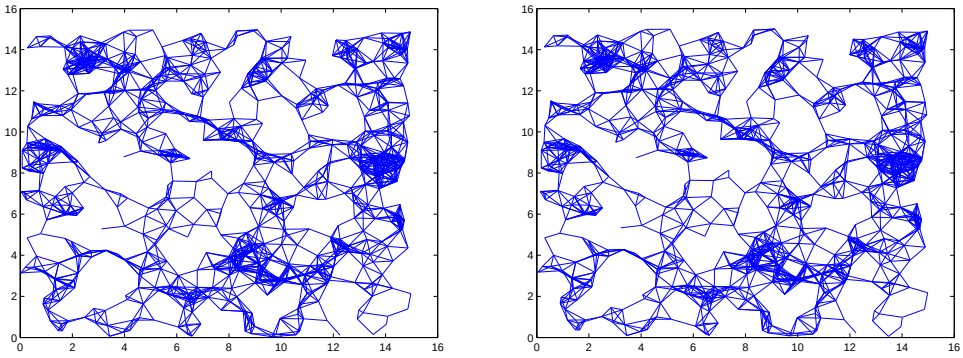


Fig. 17. The unit disk graph of 597 nodes randomly deployed inside a 15×15 square. Top: the original UDG. Bottom: embedding by LP.

We can clearly see that the results are very good. Compared to previous results on embedding in the literature, our results can be seen to have superb performance without using landmarks [Biswas and Ye 2004] or edge-length information [Gotsman and Koren 2004], even when the edges in the unit disk graphs are sparse. The number of non-adjacent node pairs having distance less than or equal to 1 in the embedding is very small, and even for such node pairs, their distances are close to 1. The number of incorrect edge crossings in the embedded graphs is very close to 0. We have also conducted experiments with many other inputs and in areas of other shapes, and the results have been consistently very good. Therefore the LP-based method does produce an almost truthful localization for sensor networks.

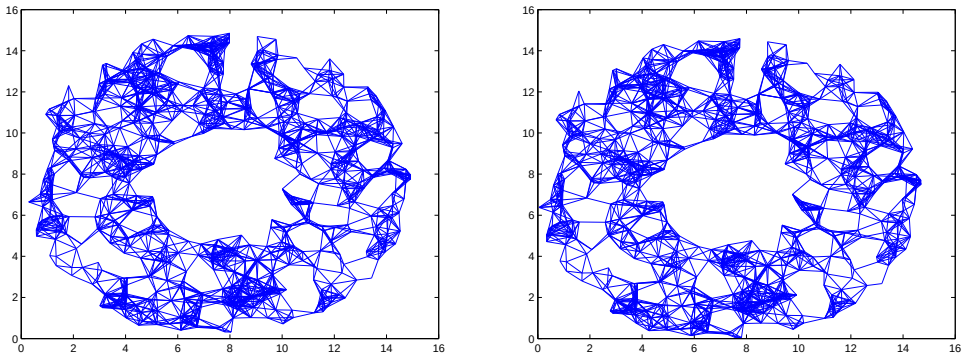


Fig. 18. The unit disk graph of 600 nodes randomly deployed inside an annulus. Top: the original UDG. Bottom: embedding by LP.

6.2 Geographical routing and approximate shortest path routing

In this section we examine the performance of routing schemes on the embedding of a unit disk graph by the linear program. In particular, given a unit disk graph with angle constraints, we find an embedding by the LP. Further, we embed the planar spanner constructed in the previous section using only local angle information. In particular we exclude the edges not in the spanner from the embedded UDG; if two edges still cross, we arbitrarily exclude one (this second step is heuristic). We run a particular geographical routing protocol (GPSR) and the approximate shortest path routing on this embedded UDG and its planar subgraph and compare the performance with that on the original (true) embedding.

Geographical routing and the approximate shortest path routing have their special requirements that differ from the criteria commonly used for localization. Geographical routing constantly makes local decisions on choosing the next hop, so it is important that the ranking of the distances from nearby nodes to any faraway destination is well maintained by the embedding. The graph-labelling-based approximate shortest path routing routes along shortest paths in planar spanners, so the distances between all pairs of nodes, adjacent or not, need to be well maintained in the embedding. Those requirements are global structures of a localization and differ from the comparatively more local criteria commonly used for localization — whether the node distance passes the threshold of 1, or whether two edges incorrectly cross or not cross. The success of the two routing algorithms in the embedded graphs shows the power of local angle information for routing, which reaches beyond the common objectives of network localization.

We experiment on sensor networks embedded with the LP approach, and compare its routing performance to that of the sensor networks with true coordinates. In the first experiment, we place n nodes in a 15×15 square with a uniform distribution, and embed the largest connected component. Then 20 source-destination node pairs are randomly selected, and routing is performed for each pair. We measure the Euclidean length (resp., number of hops) of a routing path, as well as that of the routing path with the same source-destination pair in the graph with true coordinates; we call the ratio between them the *length distortion* (resp., *hop distor-*

tion), and denote it by D_l (resp., D_h). (Note that the Euclidean length of a routing path performed on the embedded graph should still be measured based on the true Euclidean lengths of its edges.) In the second experiment nodes are placed in an annulus with external radius 7.5 and internal radius 2.5, while other conditions are unchanged. The results for GPSR and approximate shortest path routing (ASPR) are shown in Fig. 19, where each result is averaged over 50 experiments and 20 source-destination pairs in each experiment.

	network in square				
	$n =$ 200	$n =$ 400	$n =$ 600	$n =$ 800	$n =$ 1000
GPSR D_l	1.1549	1.0011	1.0000	1.0000	1.0000
GPSR D_h	1.1403	1.0007	1.0000	1.0000	1.0000
ASPR D_l	1.0000	1.0000	1.0001	1.0000	1.0000
ASPR D_h	1.0000	1.0000	1.0000	1.0000	1.0000
	network in annulus				
	$n =$ 200	$n =$ 400	$n =$ 600	$n =$ 800	$n =$ 1000
GPSR D_l	1.0580	1.0078	1.0014	1.0000	1.0000
GPSR D_h	1.0575	1.0099	1.0012	1.0000	1.0000
ASPR D_l	1.0000	1.0001	1.0000	1.0000	1.0000
ASPR D_h	1.0000	0.9989	1.0000	1.0000	1.0000

Fig. 19. Length distortion and hop distortion for GPSR and ASPR, averaged over 50 experiments and 20 source-destination pairs per experiment.

Fig. 19 shows that for GPSR and ASPR, they both have the same routing performance in the embedded networks as in the true networks, both in terms of length and hops. In fact, a detailed study showed us that most of the time, the routing routes in the embedded networks are identical to their counterparts in the true networks. We have also conducted experiments for many other inputs and in areas of other shapes, and the results have been consistently as good. Thus not only does the LP give very good local embedding, i.e., neighboring nodes are close and non-neighboring nodes are far away, but it also gives a quite accurate global view such that geographical routing and approximate shortest path routing on the embedded graph are almost identical to those on the original (true) embedding.

6.3 Handling Noises and Quasi-UDGs

In this subsection, we address the localization problem with noisy angle measurements and with sensor networks modelled as quasi-unit disk graphs. The simulation we have shown so far assumes that the angles are measured accurately. In practice measurement errors are inevitable. The modelling of sensor networks as UDG can be inaccurate, too, because network links can be lost due to noise, signal interference or obstacles, and the transmission ranges of directional antennas are not circles [Zhou et al. 2004]. A more realistic model for sensor networks is called quasi-unit disk graphs, where a pair of nodes have an edge for sure if their distance is no more than $r \leq 1$, do not have an edge if their distance is more than 1 apart,

and may or may not have an edge if their distance is between r and 1 [Kuhn and Zollinger 2003].

We extend the LP-based embedding method to deal with noisy angle measurements and quasi-unit disk graph models. We will show by simulation that this embedding algorithm by LP is robust to measurement errors and network models. First of all, as the noisy angles bring inconsistency, it is beneficial to correct them and make them approach their true values. A linear program is used to correct the angles by using the interior-angle constraint of polygons. That is, for a polygon of k edges, its interior angles $\theta_1, \theta_2, \cdots, \theta_k$ should have a sum of $(k-2)\cdot 180$ degrees. Because of the existence of noise/errors, we need to relax the interior-angle constraint to be:

$$|\sum_{i}^{k} \theta_i - (k-2)\cdot 180| \leq \varepsilon.$$

The interior-angle constraint is a linear constraint. Here ε is the *error variable* associated with the polygon. For the polygons $p_1, p_2, \cdots$ in the network, let $\varepsilon_1, \varepsilon_2, \cdots$ denote their corresponding error variables. In practice, we only use triangles and quadrilaterals for low computational complexity. In addition to the interior-angle constraint, it is useful that the angles do not deviate from their original measurements by too much, because the original measurements are good estimations of the angles. With a little abuse of notation, let $\theta_1^0, \theta_2^0, \cdots$ denote the measured noisy angles, and let $\theta_1, \theta_2, \cdots$ denote their corrected values. Then for $i = 1, 2, \cdots$, we have the linear constraint $|\theta_i - \theta_i^0| \leq \epsilon_i$, where ϵ_i is the *error variable*. The optimization objective of the linear program is set to be:

$$\text{Minimize } \sum_i \varepsilon_i + \lambda \sum_i \epsilon_i.$$

Here λ is a constant parameter. (We set $\lambda = 0.2$.) The above linear program corrects the angles' values. This LP has a very sparse coefficient matrix, so it can be solved very efficiently. It can also be efficiently computed in a distributed way by the sensors, since every constraint is a local constraint, and the optimization objective is a simple summation of the local error variables.

As the second step, the corrected angles are used to compute the edge lengths by the same LP method as before. However, since the angles here are not accurate and the network is not a UDG, it is necessary to modify the LP formulation slightly. Specifically, the *cycle constraint* is modified to be:

$$|\sum_{i=1}^{p} \ell(e_i)\cos\theta_{e_i}| \leq \varepsilon,$$

and

$$|\sum_{i=1}^{p} \ell(e_i)\sin\theta_{e_i}| \leq \epsilon,$$

where ε and ϵ are error variables. In each cycle, one edge is seen to have absolute angle 0, and then the other edges' absolute angles are easily computed based on the interior angles of the cycle. For simplicity, only cycles of five or fewer edges are

used. The *non-adjacent node pair constraint* is modified to be $\ell(e_1) + \ell(e_2) > r$ due to the Quasi-UDG property. The *edge-length constraint* is maintained, and the *crossing-edge constraint* is discarded. Let $\varepsilon_1, \epsilon_1, \varepsilon_2, \epsilon_2, \cdots$ be the error variables of the different cycle constraints. Then, the objective function is set to be minimizing $\sum_i(\varepsilon_i + \epsilon_i)$. A solution of the LP gives the edge lengths. Then we randomly choose a spanning tree of the network, and use its edge lengths and angles to determine node positions. The random spanning tree is generated a few times, and the one that gives comparatively better embedding performance is picked. In the final step, the node positions are refined using a small number of iterations as follows. When the angle between two adjacent edges, the length of the two edges, and the positions of two of their three nodes are known, the position of the third node can be determined. In each iteration, a node sets its new position as the average of its current position and the positions determined by such edges within two hops. The above linear program can also be solved efficiently, and it can be computed efficiently in a distributed way.

In the following experiment, we assume that each node measures the direction of an incident edge with an independent Gaussian error $N(0, \sigma^2)$. (So the measurement error of a local angle has a variance of about $2\sigma^2$.) For the quasi-UDG model, we assume that for two nodes whose distance d is between r and 1, there is an edge with probability $\frac{1-d}{1-r}$. Such a model has the property that nearby nodes are more likely to have edges. We place n nodes in a 10×10 square with a random uniform distribution, and embed the largest component. Some typical embedding results are shown in Fig. 20. They show that the embedding is very robust to angle measurement errors and the quasi-UDG models.

To quantitatively evaluate the embedding performance, we measure the distance between the nodes' true positions and their embedding positions. The results are shown in Fig. 21, where each result is averaged over 50 experiments. We can see that the embedding performance is consistently very good. Here σ changes from 1° (small measurement error) to 15° (large measurement error), r changes from 0.9 (relatively close to the UDG model) to 0.3 (very far away from the UDG model), and the average degree changes from 6.2 (sparse network) to 12.3 (dense network). The average distance between the nodes' true positions and their embedding positions is consistently less than 0.3. Since the maximum communication range of a node is 1 and the sensor field size is 10×10, we can see that the embedding performance is very robust to angle measurement errors, network models and node density variations.

It is shown in Fig. 21 that the embedding performance improves when σ decreases, r increases or the average degree increases. That is because in all the three cases, the constraints in the LP-based algorithm become stronger.

We also evaluate how the embedding influences the routing performance. Since for quasi-unit disk graphs, connected plane subgraphs (including planar spanners) may not exist and GPSR is not applicable, we focus on *greedy forwarding*, which is a widely used geographic routing method. We uniformly sample source-destination pairs, and compare the *length distortion* D_l and the *hop distortion* D_h (as defined before) for the embedded network and the network with true coordinates. For simplicity, we compare only those source-destination pairs where greedy forwarding succeeds in both networks. The results are shown in Fig. 22, where each result is

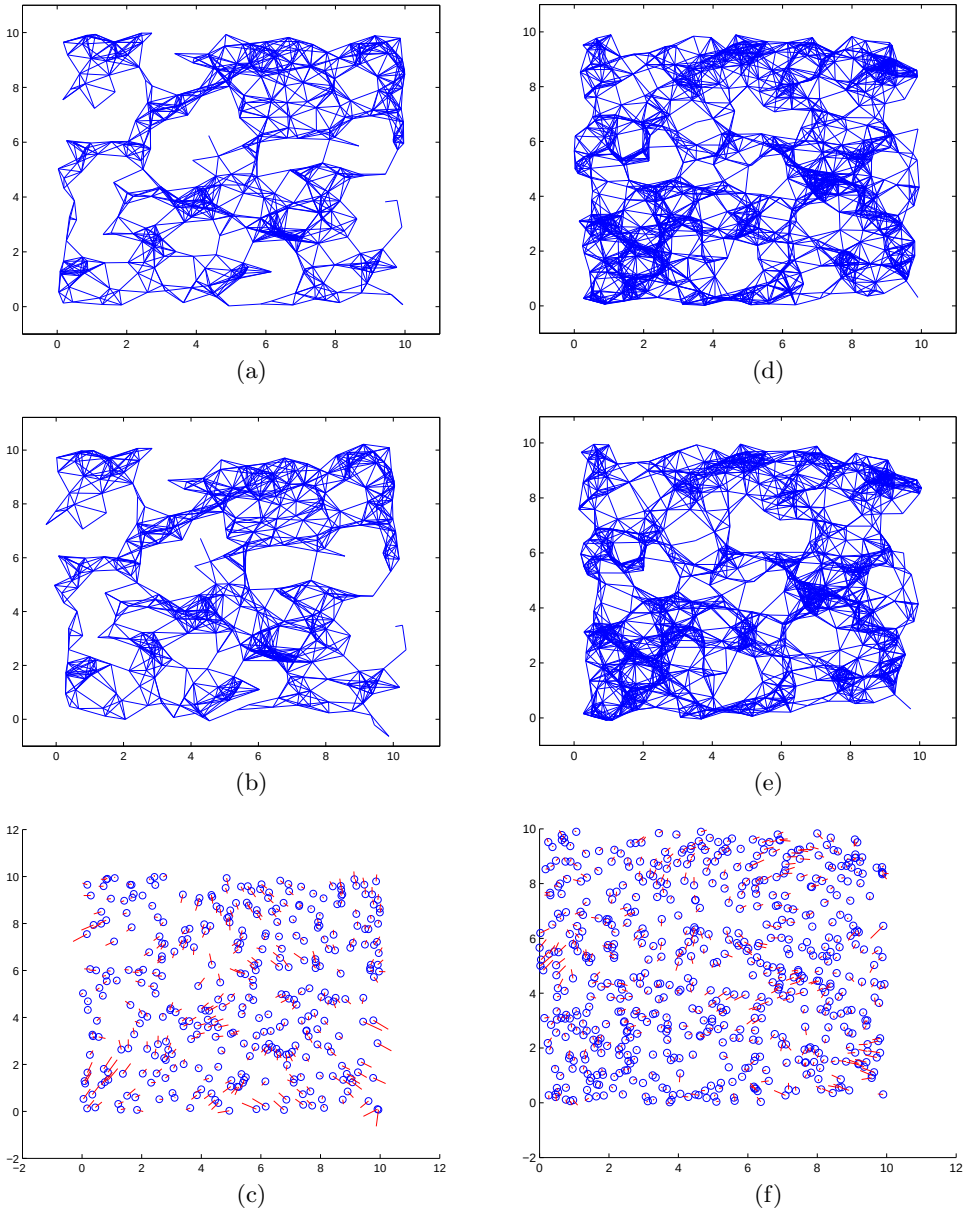


Fig. 20. Embedding quasi-unit disk graphs with noisy angle measurements. The figures (a), (b) and (c) correspond to a network with 350 nodes, where $\sigma = 5°$ and $r = 0.9$. The figures (d), (e) and (f) correspond to another network with 600 nodes, where $\sigma = 11°$ and $r = 0.6$. Figures (a) and (d): the original quasi-UDGs. Figures (b) and (e): embedding by LP. Figures (c) and (f): the difference between the nodes' original positions (circles) and embedding positions (the other end of the bars).

averaged over 50 experiments and 20 source-destination pairs in each experiment. We see that both the average distance distortion and the average hop distortion are very close to 1. In fact, in more than 80% of the cases, the two networks generate the same routing path. This shows that the embedding is good both locally and globally, and it is very useful for geographic routing.

7. SUMMARY AND FUTURE WORK

In this paper we study the embedding of unit disk graphs in the plane with angle constraints. We show theoretically that this problem is actually NP-hard. We also propose a solution based on linear programming that gives very good results in practice. This work raises a few open questions. For example, It is unknown whether one can find an algorithm that gives a good approximate embedding with theoretical bounds in the worst case. The localization algorithm here is for a sensor network in a plane, while in practice sensors may not exist in the same plane.

This paper has focused on the theoretical understanding of localization with angle information. It would be natural future work to adapt some of the ideas in real system design and evaluate the tradeoffs of the benefit with angle information versus the additional overhead of hardware requirements.

Acknowledgements: This work was supported in part by the Lee Center for Advanced Networking at the California Institute of Technology, and by NSF grant CCR-TC-0209042. A preliminary version appeared in the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05), May, 2005. Work was done when J. Gao was at Center for the Mathematics of Information, California Institute of Technology and A. Jiang was with Department of Electrical Engineering, California Institute of Technology.

REFERENCES

Aspnes, J., Goldenberg, D., and Yang, Y. R. 2004. On the computational complexity of sensor network localization. In *The 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*. 32–44.

Biswas, P. and Ye, Y. 2004. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*. 46–54.

Borg, I. and Groenen, P. 1997. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag.

Bose, P., Morin, P., Stojmenovic, I., and Urrutia, J. 1999. Routing with guaranteed delivery in ad hoc wireless networks. In *DialM '99: Proceedings of the 3rd International Workshop on Discrete Algorithms and Mehtods for Mobile Computing and Communications*. 48–55.

Breu, H. and Kirkpatrick, D. G. 1998. Unit disk graph recognition is NP-hard. *Computational Geometry. Theory and Applications 9,* 1-2, 3–24.

Bryant, V. W. 1989. Straight line representation of planar graphs. *Elementary Mathematics 44*, 64–66.

Bădoiu, M., Demaine, E. D., Hajiaghayi, M. T., and Indyk, P. 2004. Low-dimensional embedding with extra information. In *Proceedings of the 20th Annual Symposium on Computational Geometry*. 320–329.

Doherty, L., Ghaoui, L. E., and Pister, S. J. 2001. Convex position estimation in wireless sensor networks. In *IEEE Infocom*. Vol. 3. 1655–1663.

Efrat, A., Erten, C., Forrester, D., Iyer, A., and Kobourov, S. G. 2006. Force-directed approaches to sensor localization. In *Proc. of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 108–118.

Fáry, I. 1948. On straight line representations of planar graphs. *Acta Scientiarum Mathematicarum (Szeged) 11*, 229–233.

Gao, J., Guibas, L. J., Hershberger, J., Zhang, L., and Zhu, A. 2001. Geometric spanner for routing in mobile networks. In *MobiHoc '01: Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 45–55.

Gavoille, C., Peleg, D., Pérennes, S., and Raz, R. 2001. Distance labeling in graphs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*. 210–219.

Goldenberg, D. K., Bihler, P., Yang, Y. R., Cao, M., Fang, J., Morse, A. S., and Anderson, B. D. O. 2006. Localization in sparse networks using sweeps. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*. ACM Press, New York, NY, USA, 110–121.

Gotsman, C. and Koren, Y. 2004. Distributed graph layout for sensor networks. In *Proceedings of the International Symposium on Grpah Drawing*.

Hofmann-Wellenhof, B., Lichtenegger, H., and Collins, J. 2001. *Global Positioning Systems: Theory and Practice*, 5 ed. Springer.

Karp, B. and Kung, H. 2000. GPSR: Greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking*. 243–254.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2004. Unit disk graph approximation. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*. 17–23.

Kuhn, F., Wattenhofer, R., Zhang, Y., and Zollinger, A. 2003. Geometric ad-hoc routing: of theory and practice. In *PODC '03: Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing*. ACM Press, New York, NY, USA, 63–72.

Kuhn, F. and Zollinger, A. 2003. Ad-hoc networks beyond unit disk graphs. In *Proceedings of the 2003 Joint Workshop on Foundations of Mobile Computing*. 69–78.

Li, X.-Y., Calinescu, G., and Wan, P.-J. 2002. Distributed construction of planar spanner and routing for ad hoc networks. In *IEEE INFOCOM*. 1268 – 1277.

Li, Z., Trappe, W., Zhang, Y., and Nath, B. 2005. Robust statistical methods for securing wireless localization in sensor networks. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press, Piscataway, NJ, USA, 12.

Moore, D., Leonard, J., Rus, D., and Teller, S. 2004. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 50–61.

Moscibroda, T., O'Dell, R., Wattenhofer, M., and Wattenhofer, R. 2004. Virtual coordinates for ad hoc and sensor networks. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*. 8–16.

Niculescu, D. and Nath, B. 2001. Ad hoc positioning system (APS). In *IEEE GLOBECOM*. 2926–2931.

Niculescu, D. and Nath, B. 2003. *Ad hoc* positioning system (APS) using AOA. In *IEEE INFOCOM*. Vol. 22. 1734–1743.

Niculescu, D. and Nath, B. 2004. Error characteristics of ad hoc positioning systems (APS). In *MobiHoc '04: Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 20–30.

Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. 2000. The cricket location-support system. In *MobiCom '00: Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking*. 32–43.

Rajaraman, R. 2002. Topology control and routing in ad hoc networks: a survey. *SIGACT News 33,* 2, 60–73.

Rao, A., Papadimitriou, C., Shenker, S., and Stoica, I. 2003. Geographic routing without location information. In *MobiCom '03: Proceedings of the 9th ACM Annual International Conference on Mobile Computing and Networking*. 96–108.

Savvides, A., Han, C.-C., and Strivastava, M. B. 2001. Dynamic fine-grained localization in *ad-hoc* networks of sensors. In *MobiCom '01: Proceedings of the 7th ACM Annual International Conference on Mobile Computing and Networking*. 166–179.

Savvides, A., Park, H., and Strivastava, M. B. 2003. The n-hop multilateration primitive for node localization problems. *Mobile Networks and Applications 8*, 443–451.

Shang, Y., Ruml, W., Zhang, Y., and Fromherz, M. P. J. 2003. Localization from mere connectivity. In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 201–212.

So, A. M.-C. and Ye, Y. 2005. Theory of semidefinite programming for sensor network localization. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 405–414.

Whitehouse, K. and Culler, D. 2006. A robustness analysis of multi-hop ranging-based localization approximations. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*. ACM Press, New York, NY, USA, 317–325.

Whitehouse, K., Karlof, C., Woo, A., Jiang, F., and Culler, D. 2005. The effects of ranging noise on multihop localization: an empirical study. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press, Piscataway, NJ, USA, 10.

Zhou, G., He, T., Krishnamurthy, S., and Stankovic, J. A. 2004. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM Press, New York, NY, USA, 125–138.

8. APPENDIX

Now we prove that a $\sqrt{2}$-approximate embedding is topologically equivalent with a valid embedding.

If there are four nodes A, B, C, D such that in the unit-disk graph G there are edges AB, CD, AD, we show that It is impossible to have AB, CD cross in a $\sqrt{2}$-approximate embedding $\mathcal{E}$, but not cross in a valid embedding $\mathcal{E}^*$.

Without loss of generality we assume that edge AB is no shorter than CD and the embedding $\mathcal{E}$ looks like Figure 23 (i). First, if AB, CD cross in $\mathcal{E}$, then $\angle BAD + \angle CDA < \pi$. Otherwise AB, CD will never cross in any embedding preserving the angles. Notice that the angle θ between line AB, CD does not change for any embedding preserving the angles. We argue that θ is at most $\pi/6$ if we can find a valid embedding $\mathcal{E}^*$ such that AB, CD do not cross. See Figure 23 (ii). Specifically, in a valid embedding $\mathcal{E}^*$ there are no edges AC in the unit disk graph. Thus $\mathcal{E}^*(C)$ is outside the unit disk centered at $\mathcal{E}^*(A)$. $\mathcal{E}^*(B)$, $\mathcal{E}^*(D)$ are inside the unit disk centered at $\mathcal{E}^*(A)$. It is not hard to see that the angle θ achieves the maximum $\pi/6$ when $\mathcal{E}^*(A)$. $\mathcal{E}^*(B)$, $\mathcal{E}^*(D)$ are exactly of distance 1 pairwise apart and D is arbitrarily close to C such that CD is arbitrarily close to the tangent at C. So $\theta \leq \pi/6$.

In a $\sqrt{2}$-approximate embedding $\mathcal{E}$, suppose O is the intersection of edges AB, CD. $\angle BOC = \theta \leq \pi/6$. Since the length of BC, BD, CA are all greater than $\sqrt{2}/2$, the angles $\angle ACB$, $\angle CBD$ are both less than $\pi/2$. Thus the angle $\angle BCD$, $\angle CBA$ are less than $\pi/2$ as well. Assume without loss of generality that BO is longer than CO. We take the perpendicular line through B to the line CO and denote the intersection as P. P must be on the interior of line segment CO since $\angle OCB < \pi/2$. Thus the length of BC achieves the maximum when CO has the same length of

BO. Thus $d(\mathcal{E}(B), \mathcal{E}(C)) \leq 2d(\mathcal{E}(B), \mathcal{E}(O)) \sin(\pi/12) \leq 2\sin(\pi/12) \approx 0.52$. This contradicts with the assumption that BC has length at least $\sqrt{2}/2$.

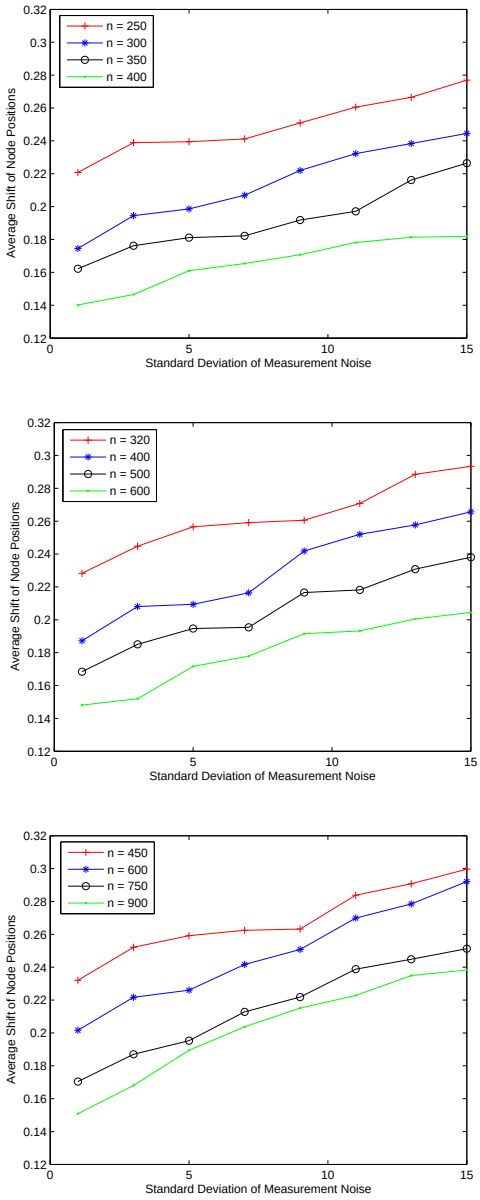


Fig. 21. Embedding accuracy versus angle measurement error, quasi-UDG model and node density. The x-axis is σ (degrees). The y-axis is the average distance between the nodes' true positions and their embedding positions. Top: $r = 0.9$. When n increases from 250 to 300, 350, 400, the order of the embedded graph increases from 246.1 to 297.6, 350.0, 400.0, and its average degree increases from 6.45 to 7.76, 9.02, 10.32. Middle: $r = 0.6$. When n increases from 320 to 400, 500, 600, the order of the embedded graph increases from 312.8 to 397.7, 499.9, 599.9, and its average degree increases from 6.30 to 7.67, 8.66, 11.44. Bottom: $r = 0.3$. When n increases from 450 to 600, 750, 900, the order of the embedded graph increases from 442.2 to 599.1, 749.0, 899.9, and its average degree increases from 6.23 to 8.23, 10.24, 12.30.

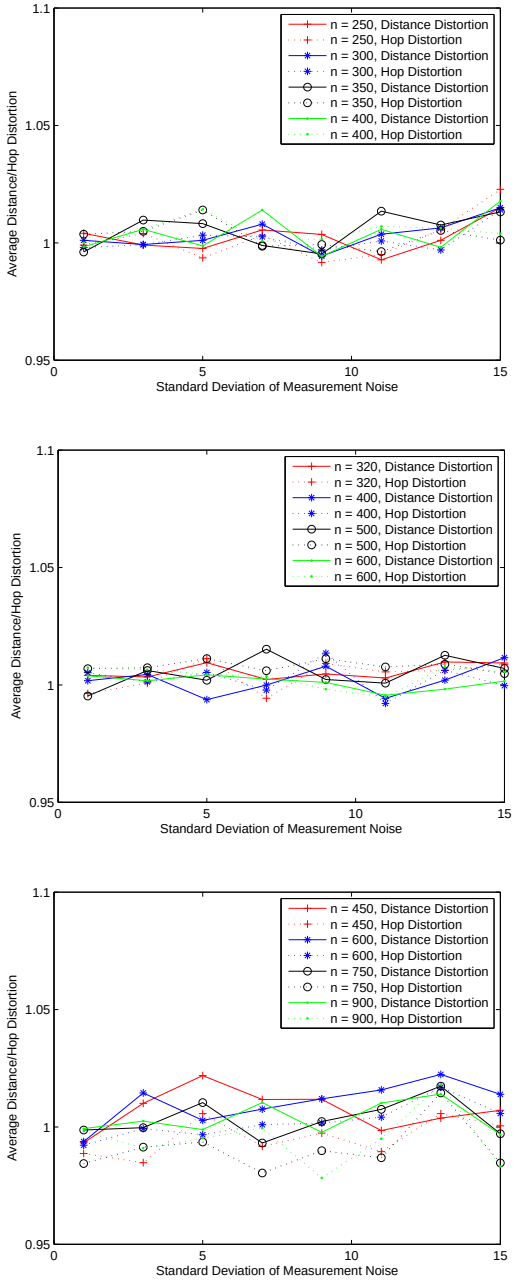


Fig. 22. Distance distortion and hop distortion for greedy-forwarding routing versus angle measurement error, quasi-UDG model and node density. The x-axis is σ (degrees). The y-axis is the average distance distortion and the average hop distortion. Top: $r = 0.9$; Middle: $r = 0.6$; Bottom: $r = 0.3$.

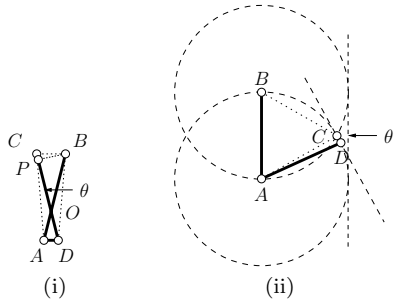


Fig. 23. (i) A $\sqrt{2}$-approximate embedding $\mathcal{E}$; (ii) A valid embedding $\mathcal{E}^*$.