Virtual Coordinates for Ad hoc and Sensor Networks*

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

Regina O'Dell
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
bregina@tik.ee.ethz.ch

Mirjam Wattenhofer
Department of Computer Science
ETH Zurich, Switzerland
mirjam.wattenhofer@inf.ethz.ch

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

ABSTRACT

In many applications of wireless ad hoc and sensor networks, position-awareness is of great importance. Often, as in the case of geometric routing, it is sufficient to have virtual coordinates, rather than real coordinates. In this paper, we address the problem of obtaining virtual coordinates based on connectivity information. In particular, we propose the first approximation algorithm for this problem and discuss implementational aspects.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

virtual coordinates, unit disk graphs, metric embedding

1. INTRODUCTION

Wireless multi-hop radio networks are playing an increasingly vital role in a wide range of applications, such as monitoring, surveillance, and data-gathering. What most of these application scenarios have in common is that *position-awareness* is a key issue. Especially in sensor networks, positioning is indispensable: Sensing the environment is useful only in the context of *where* the data has been measured. It is not surprising that the need for accurate and easily obtainable positioning information has sparked a buzz of activity in the wireless research community, and has resulted in a vast literature on the topic.

The importance of position information has been plainly shown by the advent of GPS [13], spawning a multi-billion dollar market. While serving the needs of a variety of transportation, industry, and recreational applications, attaching a GPS receiver to nodes in an ad hoc or sensor network is often undesirable or even impossible. In comparison to a sensor node, a GPS receiver is clumsy, expensive, and energy-inefficient. Moreover, GPS reception might be obstructed by climatic conditions, or if nodes are deployed indoors, there is no reception at all.

One way out of this dilemma is to equip only a few designated nodes (so-called *anchor* or *landmark* nodes) with GPS, and let all others derive their position through connectivity information. A typical approach, for example, is to let non-anchor nodes guess their position by counting the hop-distance to different anchors. A multiplicity of such "hybrid" positioning algorithms have been proposed in literature, [22, 20, 12, 26, 25, 2] to name a few. One characteristic inherent to all these approaches is that the solution quality is determined by the anchor density.

Clearly, in absence of any anchors, nodes are clueless about their real coordinates. Nonetheless, intrepid researches [24] have recently proposed to have *no anchors* at all. The underlying motivation for this paradigmatic shift is the observation that for many applications, it is not necessary to have real coordinates; often it is sufficient to have *virtual coordinates* only – two nodes having similar coordinates implies that they are physically close together.

The most prominent application of virtual coordinates is *geometric routing*. Algorithms such as GOAFR [16] and GFG/GPSR [5, 14] enable routing without routing tables and appear to be extremely resource-frugal and scalable, only at the expense that nodes need to know their coordinates. Introducing virtual coordinates can solve this problem. It is argued [2] that geo-routing in combination with virtual coordinates behaves better than other routing approaches, particularly in case of high mobility since they tend to be more stable. Other applications of virtual coordinates include locality-sensitive queries or obtaining meta-

*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.

information on the network. Finally, virtual coordinates can be used as an efficient means for anycast services ("Which of the service nodes is closest to me?").

As it has become customary, we model the multi-hop radio network as a unit disk graph (UDG). In a UDG $G = (V, E)$, there is an edge $\{u, v\} \in E$ iff the Euclidean distance between u and v is at most 1. In this paper, we consider the *connectivity-based* approach, in which nodes can solely derive connectivity information (as opposed to measuring angles or distances to their neighbors). The usefulness and importance of this approach has been documented in a number of papers, e.g. [23, 2].

Our algorithm can naturally be extended along two orthogonal directions. The first direction is that we can also include anchors to arrive at the *positioning problem*, where the goal is to obtain *real coordinates*. The other direction is to integrate inter-node distance measurements where we simply skip the first step of our algorithm, thus circumventing the computationally most expensive part. An exciting example of such a *distance-based virtual coordinates* problem is *Internet mapping* (such as [7, 21]), the goal of which is to obtain topological information about the Internet graph in order to enhance anycast or peer-to-peer systems.

Despite being a clear-cut and well-formulated optimization problem, all previous work on virtual coordinates is of heuristic nature only. To the best of our knowledge this paper presents the first "hard" provable results. In particular, we give an approximation algorithm which achieves an approximation ratio of $O(\log^{2.5} n\sqrt{\log\log n})$, where n is the number of nodes in the network.

The remainder of this paper is organized as follows. Section 2 gives an overview over relevant previous work. After introducing the notation in Section 3, we define and analyze the algorithm in Sections 4 and 5, respectively. Details on the algorithm's implementation are given in subsequent Section 6, before Section 7 concludes the work.

2. RELATED WORK

Despite there being a plethora of positioning papers (where a node approximates its actual coordinate), comparatively little has been published on the companion problem of finding virtual coordinates. For the one-dimensional Euclidean line, Bischoff et al. have given an optimal algorithm in [2]. As for the two-dimensional case, some of the most notable works are by Rao et al. [24], Shang et al. [27], as well as Doherty et al. [8] and recently Biswas and Ye [3]. The key contribution of our approach lies in the fact that (with the exception of [2]), all these algorithms evaluate their success via simulation on some random graphs whilst we provide theoretic upper bounds on the quality of our algorithm.

Extending connectivity-based approaches to positioning problems has been discussed in [27]. In particular, [27] describes the application of an affine transformation of the coordinate system so as to (approximately) match the relative with the known absolute coordinates. The integration of inter-node distance measurements into the computation of virtual coordinates has been studied in [8, 27].

From a theoretical point of view, a homogenous sensor network can be modelled by a unit disk graph. Computing virtual coordinates for a network is then turned into finding a representation of a given UDG. In that context, researchers have already established fundamental lower bounds. In particular, Breu and Kirkpatrick [6] have proven that it is NP-hard to recognize whether a given graph is a UDG. Consequently, it is also NP-hard to determine a set of virtual coordinates that satisfy all of the UDG constraints for any given unit disk graph. More recently, Kuhn et al. [15] have shown that even approximating the constraints to within a factor of $\sqrt{3/2}$ is NP-hard.

Research on computing coordinates for graphs is not restricted to the wireless networks community. Whereas we have derived our motivation from the positioning point of view, our methods to solve the problem stem from seemingly unrelated areas. The task of embedding graphs can be approached from many different angles and numerous alternatives have been explored for quite some time. We will briefly survey some of these ideas, all stemming from the large research area of the geometry of graphs.

The first of these alternative approaches is the so-called *topological* approach, which was the most intensively studied one for any length of time. It is mainly concerned with graph planarity and embeddability of graphs on 2-dimensional manifolds, where some criteria the embedding should satisfy are: a small number of edge crossings, evenly distributed vertices and edges, short edges, few edge bends, a small layout area or volume, and a good angular resolution. We refer to [28] for readers interested in a more detailed discussion on this topic. Though this approach is not applicable directly for the virtual coordinates problem we are convinced that a better understanding of methods used in this area of research would also lead to better algorithms for problems arising in the area of wireless networks, as seen in the work of [24].

Lately, a new approach, the so-called *metric* approach, which is the angle we take in this work, has enjoyed great popularity among researchers in the area of the geometry of graphs. Its main concern is to find a representation of a graph in a geometric space, such that the metric of the representation has small edge-distortion compared to the given metric. The *edge-distortion* of an embedding is a factor c such that for all two vertices in the graph it holds that if their distance in the graph is d, then their distance in the embedding is between d and d/c. Linial, London and Rabinovich [17] showed how small-distortion embeddings can be obtained and how those embeddings can be used in the design of algorithms. Whereas the graphs in [17] are embedded in rather high-dimensional ($\log n$) spaces, Badoiu [1] gave an algorithm which finds good embeddings in the 2-dimensional plane under the l_1 norm. Feige [10] extended the notion of distortion to sets, rather than just pairs, of vertices. Note, that for all these approaches to work one needs to have a metric on a graph, not merely a graph. This is one of the key problems of applying methods from this research area to the virtual coordinates problem. Vempala [29] uses these approaches to find an embedding of a graph on a d-dimensional grid, so as to minimize the maximum edge length. The given algorithm, which our work is based on, obtains an $O(\log^{2.5+2/d} n)$ approximation in polynomial time. In this paper, we adapt Vempala's techniques for our purpose. Other methods used in ours as well as Vempala's paper are random projection ([30, 4]) and spreading constraints ([9, 4]).

3. NOTATION

In this section, we will formalize a virtual coordinates algorithm as a computation of an embedding. We also intro-

duce various notations used in the sequel of the paper. An *embedding* of a graph $G = (V, E)$ in the Euclidean plane is a mapping $f : V \to \mathbb{R}^2$, i.e., each vertex v is identified with a point (x, y) in the plane. In analogy to the term *edge*, we call a pair of vertices $\{v_i, v_j\}$ a *non-edge* if $\{v_i, v_j\} \notin E$.

An *independent set* of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that for all $v_i, v_j \in S$, $\{v_i, v_j\} \notin E$. An independent set is *maximal* if no additional vertex can be added to S such that the resulting set is still an independent set. In this paper, we write IS to denote an independent set in the given graph G.

A *semi-metric space* (or simply a *semi-metric*) is a pair (S, ρ), where S is a set of points and $\rho : S \times S \to [0, \infty)$ is a *distance function* satisfying the properties $\rho(p, q) = \rho(q, p)$ and $\rho(p, r) + \rho(r, q) \leq \rho(p, q)$ for all $p, q, r \in S$. Note that we do not impose the additional constraint $\rho(p, q) = 0 \Leftrightarrow p = q$ to be a full metric. However, for readability and since this constraint is inconsequential to the discussion, we refer to a semi-metric as a metric. A finite metric space can be represented as a complete graph on $|S|$ vertices, with edge lengths between pairs of vertices equal to the distance between the respective points in the metric space.

An embedding of a finite metric space (S, ρ) in an L-dimensional Euclidean space $(\mathbb{R}^L, \sigma)$ is a mapping $f : S \to \mathbb{R}^L$. In this paper we only consider *contracting* mappings, that is for all $x, y \in S$, $\rho(x, y) \geq \sigma(f(x), f(y))$. Let us say that the volume $Evol(S)$ of a set S of k vertices in $\mathbb{R}^L$ is the $(k-1)$-dimensional volume of the simplex spanned by S. Then the volume $Vol(S)$ of a finite metric space with k points is defined to be the maximum $Evol(f(S))$ over all contracting mappings f from S to $\mathbb{R}^L$. The *distortion* of an embedding f is the ratio

$$\left(\frac{Vol(S)}{Evol(f(S))}\right),$$

and the embedding is (k, D)-*volume respecting* if for every subset $S \subseteq V$ of at most k vertices we have

$$\left(\frac{Vol(S)}{Evol(f(S))}\right)^{1/(k-1)} \leq D. \tag{1}$$

In the rest of the paper, we are merely concerned with graphs G which are unit disk graphs and consequently call an embedding of G that satisfies all UDG constraints a *realization* of G. Formally, a realization is defined as follows.

DEFINITION 3.1 (REALIZATION). *A* realization *of a unit disk graph $G = (V, E)$ in the Euclidean plane is an embedding $r(G)$ of G such that $\{v_i, v_j\} \in E \Leftrightarrow \rho(f(v_i), f(v_j)) \leq 1$ where ρ is the Euclidean distance between two points.*

It is clear that there is a realization for every UDG graph G. This paper addresses the problem of finding such an embedding. Since finding an exact solution is NP-complete [6, 15], we want to find an embedding which reasonably approximates such a realization. In particular, we want to map adjacent vertices in the graph to points in the plane which are close together. On the other hand, non-adjacent vertices should be embedded far from one another in the plane. This intuition naturally leads to a quality measure based on the ratio between the longest edge to the shortest non-edge in the embedding. Therefore, we formally define the *quality of an embedding* as follows.

DEFINITION 3.2 (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*

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

Let $\mathcal{G}$ be the family of all unit disk graphs. An algorithm ALG for the virtual coordinate problem is an algorithm which, given an input graph $G \in \mathcal{G}$, computes an embedding $r_{ALG}(G)$. The algorithm achieves approximation ratio α if $q(r_{ALG}(G)) \leq \alpha$ for all $G \in \mathcal{G}$.

4. THE ALGORITHM

The algorithm is presented in detail in Section 4.2. While trying to give the reader a more intuitive understanding of the algorithm we now explain the ideas behind each of its four main stages separately. At the end of each stage we explain how it connects to the next one. To give a first rough idea we shortly summarize the main ideas: In the first stage we solve a set of linear constraints which gives indications on the distances between any two vertices. Those distances are used thereafter to embed the vertices in an n-dimensional space, where the embedding is done in such a way that independent sets of vertices have a large volume. Given this high-dimensional embedding we project the points to a randomly chosen 2-dimensional plane. Finally, some of the points are placed on grid points, while the others are placed around them.

4.1 Overview

4.1.1 Linear Constraints

The main goal of the first stage of the algorithm is to compute a metric on the input graph, which satisfies certain additional constraints. The key difficulty is that the problem of describing the UDG conditions is inherently non-convex. As such, we need to replace these problematic constraints with something more managable. Therefore, we make use of *spreading constraints* (as in Even et al. [9]),

$$\sum_{v \in S} \rho(u, v) \geq c \cdot |S|^{3/2} \quad \forall S \subset V, \forall u \in V \tag{2}$$

for some constant c. In our case, we will not be interested in all subsets $S \subset V$, but only in all the independent sets $IS \subset V$ of G.

First we give some intuition on what spreading constraints achieve in general and then specifically why we impose them merely on independent sets. The motivation for using spreading constraints is to model the property that in any region of diameter R there are at most $O(R^2)$ points. That is, the points are enforced to spread out, instead of clustering at a point. Now, in relation with unit disk graphs, we obviously do not want *all* points to be spread out, but rather want to model the fact that non-edges are "far apart" while edges are allowed to be "close". This becomes intuitively clear when we think of a clique of size n. Surely, a clique is a unit disk graph, but one that cannot fulfill the spreading constraints. So instead of formulating the spreading constraints on all sets of vertices, we replace them by saying that sets of non-edges, i.e. independent sets, must be sufficiently far apart. Serendipitously, all independent sets IS of a unit disk graph

G satisfy Eq. (2). To see this, observe that for any given realization of G, the number of independent nodes in a region of radius R is at most κR^2 (κ being a constant) since all pair-wise distances are greater than 1.

We are ready to formulate a set of convex constraints[1] to describe the essential properties of a UDG $G = (V, E)$. For the sake of readability, we set $x_{uv} = \rho(u, v)$.

(LP) subject to:

$$x_{uv} \leq 1 \qquad \forall \{u, v\} \in E \qquad (3)$$

$$x_{uv} \leq \sqrt{n} \qquad \forall u, v \in V \qquad (4)$$

$$x_{uv} \geq 0 \qquad \forall u, v \in V \qquad (5)$$

$$x_{uv} + x_{uk} \geq x_{vk} \qquad \forall u, v, k \in V \qquad (6)$$

$$\sum_{v \in IS} x_{uv} \geq \kappa |IS|^{3/2} \qquad \forall IS \subset V, \forall u \in V. \qquad (7)$$

Constraints (5) and (6) ensure that we have a metric which we need in order to embed the nodes into Euclidean space. From the definition of unit disk graphs and the discussion of the spreading constraints above, it immediately follows that if G is a UDG, then there exists a feasible solution to (LP). We show in Section 6.1 that it can be found in polynomial time.

Observe that we are not actually computing any coordinates of the points yet, only their pairwise Euclidean distances. This is perhaps the key conceptual difference between our approach and that of previous virtual coordinates algorithms, such as [8, 27, 24], where the point coordinates, which these algorithms attempt to approximate, are included in the initial equations.

Now that we have computed a metric with the desired properties, we would like to embed it into a geometric space. Preferably, we would like to embed it in the 2-dimensional plane, but in general this cannot be done directly without large edge-distortion. Instead, we embed the vertices in an high- (n-)dimensional space where we cannot only guarantee that the distortion is small but also that the embedding is *volume respecting*.

4.1.2 Volume-Respecting Embedding

Feige [10] introduced a powerful strengthening of the notion of the edge-distortion of an embedding, concerning embeddings into Euclidean spaces. Whereas the usual distortion of an embedding is determined by looking at pairs of points, that is, distances, Feige's volume respecting embedding takes into account all k-tuples for some $k \geq 2$, that is volumes.

A volume respecting embedding has many useful properties when projecting the points from the high-dimensional space to a random lower dimensional one. Intuitively, large volumes have large projections and hence the points spread fairly well when projecting to a random lower dimension subspace. This leads us to the third stage of the algorithm. Once a volume respecting embedding in $\mathbb{R}^n$ is computed, the points of the embedding are projected to a randomly chosen 2-dimensional plane.

4.1.3 Random Projection

Random projection is the technique of projecting a set of points from a high-dimensional space to a low-dimensional subspace. Lately, this technique has enjoyed great popularity in various research areas. For our application, two observations about random projections are crucial. Firstly, the length of a vector when projected from $\mathbb{R}^L$ to a random line in $\mathbb{R}^L$ scales by roughly $1/\sqrt{L}$ and is concentrated around this expectation. Secondly, the probability that a set of k points is projected to a small interval is inversely proportional to the volume of the points. Hence, together with the fact that our embedding was volume respecting, we get that the projected points spread quite well in the 2-dimensional plane. In fact, we prove in Section 5.4 that if we partition the plane into a grid with cell-width $1/\sqrt{n}$ then at most $O(\log^4 n \log\log n)$ independent points lie in a cell with high probability. This leaves us to show what is to be done with the points in one cell in the final stage of the algorithm.

4.1.4 Final Embedding

In order to guarantee that the smallest non-edge is not too short we need the points within a cell to be evenly spread out and not to cluster at a single point. For that reason, we compute a maximal independent set MIS of vertices within a grid cell and assign those vertices to grid points of a refined grid. The width of a cell in the refined grid is $1/\sqrt{nM}$ along each dimension, M being the maximum number of independent points in any cell.

We now have to embed all the vertices in G which are not in MIS. The idea is that we can assign each node u not in MIS to exactly one vertex in MIS. Then, for each $v \in MIS$, we place its assigned vertices uniformly around v, respecting the neighborly relations of the assigned vertices.

4.2 The Algorithm in Pseudocode

Algorithms 1-4 succinctly describe each stage of our algorithm to compute the virtual coordinates of a given (unit disk) graph. The output of each phase serves as the input of the next one.

In the following, let us denote the position of vertex u in the volume respecting embedding v_u^b, the position after the random projection r_u and the final position p_u.

Algorithm 1 Solve (LP)

Input: $G = (V, E)$
Output: distance matrix $X = (x_{uv})$
1: solve (LP) and return set of distances x_{uv} between each pair of vertices u, v

Algorithm 2 Volume-Respecting Embedding

Input: distance matrix $X = (x_{uv})$
Output: positions $v_u^n \in \mathbb{R}^n$ for all $u \in V$
1: find a $(\log n, \log^2 n)$-volume respecting Euclidean embedding in $\mathbb{R}^n$ using the "random subsets embedding" in [10]

5. ANALYSIS

This entire section is devoted to proving the following main theorem.

THEOREM 5.1. *The quality of the embedding computed by the algorithm is in $O(\log^{2.5} n \sqrt{\log\log n})$ with high probability.*

[1] basically, a linear program without an optimization function

Algorithm 3 Random Projection

Input: positions $v_u^n \in \mathbb{R}^n$ for all $u \in V$
Output: positions $r_u \in \mathbb{R}^2$ for all $u \in V$

1: independently choose two random vectors $l_1, l_2 \in \mathbb{R}^n$ of unit length (lines passing through the origin)
2: for all $u \in V$, project v_u^n to each of the lines, that is $r_u \leftarrow (v_u^n \cdot l_1, v_u^n \cdot l_2)$

Algorithm 4 Final Embedding

Input: positions $r_u \in \mathbb{R}^2$ for all $u \in V$
Output: positions $p_u \in \mathbb{R}^2$ for all $u \in V$

1: enclose the projected points in a grid $C_{\sqrt{n}}$ of cell-width $1/\sqrt{n}$
2: let C be a cell in $C_{\sqrt{n}}$ and compute a maximal independent set MIS_C in C, then $M \leftarrow \max_C |MIS_C|$
3: refine $C_{\sqrt{n}}$ by subdividing each cell into M subcells, denote the refined grid by $C_{\sqrt{nM}}$
4: **for** each cell C in $C_{\sqrt{n}}$ **do**
5: assign all points in MIS_C arbitrarily to grid-points in $C_{\sqrt{nM}}$ which lie in C
6: assign all points in C not in MIS_C to points in MIS_C, such that there is an edge in G between assigned points
7: **for** all points u in MIS_C **do**
8: place points assigned to u on a circle of diameter $2/3$ and center u, where vertices which are neighbored in G may be placed on the same position, and non-neighbored vertices are equally distantly distributed on the circle
9: **end for**
10: **end for**
11: scale up $C_{\sqrt{nM}}$ by a factor $\sqrt{nM}$ along each dimension, such that the distance between any two grid points is one

5.1 Volume Respecting Embeddings

Once we have a set of inter-point distances, we want to ensure that the high-dimensional embedding does not distort the encoded UDG properties. The volume-respecting property of Feige's embedding ensures that independent vertices remain spread out.

THEOREM 5.2 ([10]). *For any k and any connected graph G, a $(k, \beta\sqrt{\log n}\sqrt{k \log k + \log n})$-volume respecting embedding can be found in polynomial time for some large enough constant β.*

Unfortunately, computing the volume $Vol(S)$ of a set S exactly is far too tedious in general. Yet, Feige showed that it can be approximated quite well by making use of the notion of the *tree volume* $Tvol(S)$ of S, which is the product of the edge lengths in a minimum spanning tree of S.

THEOREM 5.3 ([10]). *For $S \subset V$, $|S| = k$,*

$$Vol(S) \leq \frac{Tvol(S)}{(k-1)!} \leq Vol(S) 2^{(k-2)/2}.$$

5.2 Tree Volume

Since we want to ensure that the independent sets are neatly spread out, we need to derive bounds on the tree volume of those sets in order to use Feige's volume-respecting embedding. In particular, we want to show the following.

LEMMA 5.4. *Let IS denote an independent set of vertices in G then*

$$\sum_{IS \subset V,\, |IS|=k} \frac{1}{Tvol(IS)^2} \leq (1/\kappa^2)^{k-1} n \log^{k-1} n.$$

Before proving the lemma above we start with a general lemma on metric spaces and tree volumes. Observe that in our case, the set S is an independent set which, by virtue of being a solution to (LP), is also a (finite) metric space.

LEMMA 5.5. *For any finite metric space (S, x) with $S = \{u_1, \ldots, u_k\}$,*

$$\frac{1}{Tvol(S)^2} \leq \sum_{\pi} \left(x_{u_{\pi(1)}u_{\pi(2)}}, \ldots, x_{u_{\pi(k-1)}u_{\pi(k)}}\right)^{-2}$$

where the summation is taken over all permutations $\pi \in S_k$.

PROOF. We prove the lemma by induction on k, as similarly done in [10]. For $k = 2$ there is only one spanning tree and two possible permutations π, hence we get $1/Tvol(S)^2 \leq 2/x_{u_1u_2}^2$, which is clearly true.

Assume that the statement is correct for sets of size k, that is if $|S| = k$ then

$$Tvol(S)^2 \geq 1/\sum_{\pi} \left(x_{u_{\pi(1)}u_{\pi(2)}}, \ldots, x_{u_{\pi(k-1)}u_{\pi(k)}}\right)^{-2} \quad (8)$$

for all $\pi \in S_k$. Now consider the set S' where we added a vertex u_{k+1} to S and let us denote the permutations on $\{1, \ldots, k+1\}$ by σ. The tree volume of S' is at least the tree volume of S times the minimum distance between u_{k+1} and S, formally: $Tvol(S') \geq Tvol(S) \cdot \min_i x_{u_{k+1}u_i}$. Let $\min_i x_{u_iu_{k+1}}$ be $x_{u_iu_{k+1}}$. Then, assuming (8), this leaves us to show that

$$x_{u_iu_{k+1}}^2 \cdot \sum_{\sigma} \left(x_{u_{\sigma(1)}u_{\sigma(2)}}, \ldots, x_{u_{\sigma(k)}u_{\sigma(k+1)}}\right)^{-2} \geq \sum_{\pi} \left(x_{u_{\pi(1)}u_{\pi(2)}}, \ldots, x_{u_{\pi(k-1)}u_{\pi(k)}}\right)^{-2}.$$

Each permutation π corresponds to $k+1$ different permutations σ, depending on where $k+1$ is inserted. We now fix a permutation π and show that the above inequality holds for the fixed permutation and the corresponding $k+1$ permutations σ. Clearly, the inequality then also holds in the stated form. W.l.o.g., we let the fixed permutation be the identity permutation and denote $(x_{u_1u_2}, \ldots, x_{u_{k-1}u_k})^{-2}$ by A^2. We want to show that

$$x_{u_iu_{k+1}} A^2 \cdot \left(\frac{1}{x_{u_1u_{k+1}}^2} + \frac{1}{x_{u_ku_{k+1}}^2} + \sum_{j=2}^{k-1} \frac{x_{u_ju_{i+j}}^2}{x_{u_ju_{k+1}}^2 x_{u_{j+1}u_{k+1}}^2} \right) \geq A^2.$$

By the triangle inequality and simple arithmetic we get $x_{u_ju_{j+1}}^2/2 \geq x_{u_ju_{k+1}}^2 - x_{u_{k+1}u_{j+1}}^2$, which we use for $j < i$. In case $j \geq i$ we use $x_{u_ju_{j+1}}^2/2 \geq x_{u_{k+1}u_{j+1}}^2 - x_{u_ju_{k+1}}^2$. Then the terms inside the parentheses cancel each other out except for two terms of the form $1/(2 \cdot x_{u_{k+1}u_i})$, yielding the desired inequality. □

We will also need the following spreading constraint property.

LEMMA 5.6. *Let (x_{uv}) be a feasible solution vector of (LP). For any vertex u in G and any set of independent vertices IS, such that $\{IS \cup u\}$ is an independent set in G, it holds that*

$$\sum_{u_i \in IS} \frac{1}{x_{uu_i}^2} \leq \log n/\kappa^2.$$

PROOF. Fix u and consider an arbitrary set of vertices $IS = \{u_1, \ldots, u_m\}$, which together with u forms an independent set in G. Then any subset, in particular $\{u, u_1, u_2\}$, is also independent. Therefore, using the spreading constraints of (LP), $x_{uu_1} + x_{uu_2} \geq \kappa 2^{3/2}$. Hence, w.l.o.g., $x_{uu_1} \leq \kappa 2^{1/2}$ and $x_{uu_2} \geq \kappa 2^{1/2}$.

Now consider all independent sets of the form $\{u, u_1, u_i\}$, $1 < i \leq m$. Since we could upper bound the length of x_{uu_1}, we can lower bound all the x_{uu_i}, $1 < i \leq m$, by $2^{1/2}$. By applying the same reasoning as above to subsets of $IS \cup u$ of increasing size, we get that at least $m - j$ of the distances x_{uu_i} are at most $\kappa(j+1)^{1/2}$, for each $1 \leq j \leq m-1$. Considering furthermore that the minimal distance between two independent vertices is at least κ, it follows $\kappa^2 \sum_{u_i \in IS} 1/{x_{uu_i}}^2 \leq 1/m + 1/(m-1) + \ldots 1/2 = H(n) - 1 \leq \log n$, where $H(n)$ is the harmonic number and $m \leq n$. □

PROOF PROOF OF LEMMA 5.4. We use Lemma 5.5 and rewrite the sum in the following way (setting, for readability, $IS_i = \{u_1, \ldots, u_i\}$ an independent set in G):

$$\sum_{IS \subset V, |IS|=k} \frac{1}{Tvol(IS)^2} \leq$$

$$\sum_{u_1 \in V} \sum_{u_2 \in V \setminus IS_1} \cdots \sum_{u_k \in V \setminus IS_{k-1}} \left(\frac{1}{x_{u_1 u_2}, \ldots, x_{u_{k-1} u_k}} \right)^2$$

$$= \sum_{\substack{u_1, u_2 \in V \\ IS_2}} \left(\frac{1}{x_{u_1 u_2}^2} \sum_{\substack{u_3 \in V \\ IS_3}} \left(\frac{1}{x_{u_2 u_3}^2} \cdots \sum_{\substack{u_k \in V \\ IS_k}} \frac{1}{x_{u_{k-1} u_k}^2} \right) \cdots \right).$$

Using Lemma 5.6 we can upper bound the single sums, except for the outer one, by $\log n/\kappa^2$. The outer sum can be upper bounded by $n \log n$ and the lemma follows. □

5.3 Random Projection

While the volume-respecting embedding helps in keeping non-edges far apart, we also want the edges to be close together. In this section, we prove that the edge length will be bounded after the random projection step.

For starters, we will need the well-known lemma given below [30].

LEMMA 5.7. *Let $v \in \mathbb{R}^n$. For a random unit vector l,*

$$\mathbb{P}\left(|v \cdot l| \leq \frac{c}{\sqrt{n}} |v| \right) \geq 1 - e^{-c^2/4}.$$

$$\mathbb{P}\left(|v \cdot l| \leq \frac{1}{c\sqrt{n}} |v| \right) \in O(1/c).$$

The next lemma gives a connection between random projection and the volume of a set. It will become important when we count the number of nodes that fall into any given cell of the final grid.

LEMMA 5.8 ([10]). *Let S be a set of vectors $v_1, \ldots, v_k \in \mathbb{R}^n$. For $c > 0$, consider the event that the projection of S on a random unit vector l is of length at most c. The probability of this event is bounded by*

$$\mathbb{P}\left(\max_i (v_i \cdot l) - \min_j (v_j \cdot l) \leq c \right) = O\left(\frac{c^{k-1} n^{k-1/2}}{(k-1)! Evol(S)} \right).$$

LEMMA 5.9. *The length $|r_u - r_v|$ of an edge after the projection step is at most $O(|r_u - r_v| \sqrt{\log n}/\sqrt{n})$, with high probability.*

PROOF. By Lemma 5.7 with probability $1 - e^{-c^2/4}$ a projected vector v has length at most $\frac{c}{\sqrt{n}}|v|$. Choosing $c = \sqrt{\log n}$, the length of a projected vector is at most

$$|r_u - r_v| \sqrt{\log n}/\sqrt{n}$$

with probability $1 - (1/n)^{1/4}$. □

5.4 Putting Things Together

LEMMA 5.10. *The number of independent vertices that fall in any cell of the outer grid $C_{\sqrt{n}}$ is in $O(\log^4 n \log\log n)$ with high probability.*

PROOF. Let N_C be the number of independent sets IS of size k that fall into an arbitrary grid cell C of width $1/\sqrt{n}$. Let X_{IS}^i be the indicator random variable which is 1 if all the vectors in IS fall in C along l_i. Following the reasoning in [29], we can express the expected value of N_C in terms of X_{IS}^1 as follows:

$$\begin{aligned}
\mathbb{E}[N_C] &= \sum_{|IS|=k} \mathbb{E}[X_{IS}^1]^2 \\
&= \sum_{|IS|=k} \mathbb{P}(X_{IS}^1 = 1)^2 \\
&\overset{\text{(Lem. 5.8)}}{\leq} \sum_{|IS|=k} \left(\frac{c^k n^{k/2}}{(k-1)! Evol(IS)} \right)^2 \\
&\overset{(c = 1/\sqrt{n})}{=} \sum_{|IS|=k} \left(\frac{1}{(k-1)! Evol(IS)} \right)^2 \\
&\overset{\text{Eq. (1)}}{\leq} \sum_{|IS|=k} \left(\frac{(\beta \log n \sqrt{\log\log n})^k}{(k-1)! Vol(IS)} \right)^2 \\
&\overset{\text{(Thm. 5.3)}}{\leq} \sum_{|IS|=k} \left(\frac{(\beta \log n \sqrt{\log\log n})^k 2^k (k-1)!}{(k-1)! Tvol(IS)} \right)^2 \\
&= (2\beta \log n \sqrt{\log\log n})^{2k} \sum_{|IS|=k} \frac{1}{Tvol(IS)^2} \\
&\overset{\text{(Lem. 5.4)}}{\leq} n(2/\kappa^2 \beta \log^3 n \log\log n)^k.
\end{aligned} \tag{9}$$

Using Markov's inequality, we have

$$\mathbb{P}(N_C \geq n^3 n(2/\kappa^2 \beta \log^3 n \log\log n)^k) \leq 1/n^3. \tag{10}$$

We can bound the number of cells by n^2 using Lemma 5.9. Hence, with probability $1 - 1/n$ the number of sets of independent point that fall in any of the n^2 cells is at most the value computed in (9). By this we can bound the maximum number of independent points that fall in any of the cells of the outer grid in the following way: if there are N subsets of size k the number of elements is at most $kN^{1/k}$. In our case,

we get that the maximum number of independent points in an outer grid cell is with high probability at most

$$k(n^4(2/\kappa^2\beta\log^3 n\log\log n)^k)^{1/k} \in O(\log^4 n\log\log n)$$

by choosing $k = \log n$. □

PROPOSITION 5.11. *The maximum edge length is in* $O(\log^{2.5} n\sqrt{\log\log n})$.

PROOF. We have seen in Lemma 5.9 that an edge can have length at most $O(|r_u - r_v|\sqrt{\log n}/\sqrt{n})$. Hence, using the fact that the embedding is non-expansive, an edge $\{u, v\}$ spans at most $O(x_{uv}\sqrt{\log n})$ cells along each dimension. By Lemma 5.10, each cell of the grid $C_{\sqrt{n}}$ is divided into $O(\log^4 n\log\log n)$ cells, or $O(\log^2 n\sqrt{\log\log n})$ cells along each dimension. Keeping in mind that, by condition (3) in (LP), $x_{uv} \leq 1$, the total number of cells in the final grid $C_{\sqrt{nM}}$ that are spanned by an edge along each dimension is at most $O(\log^{2.5} n\sqrt{\log\log n})$. The final embedding has no impact on the big-oh notation. □

PROPOSITION 5.12. *The minimum distance between two non-neighbored vertices is at least* 1/3.

PROOF. A vertex v in a UDG can have at most 5 neighbors which are mutually independent. For v in the independent set of a grid cell, 5 vertices are placed on a circle of diameter 2/3, meaning that the distance between any pair is at least $\pi \cdot 2/3 \cdot 1/5 > 2/5 > 1/3$. The distance between v and its neighbors is at least 1/3, the distance between neighbors of v and any other vertex is also at least 1/3 by construction. □

PROOF OF THEOREM 5.1. The theorem follows immediately from Definition 3.2 and Propositions 5.11 and 5.12. □

6. IMPLEMENTING THE ALGORITHM

6.1 Polynomial Running Time

In this section we show that the algorithm can be implemented in such a way that its running time is polynomial in n. In particular, we show how the set of constraints (LP) can be solved in polynomial time, despite being exponential in number.

To that end, we will construct a *separation oracle* which will guarantee that we only need to solve polynomially many constraints polynomially often [11]. A separation oracle for (LP) takes as input a set of lengths $\{x_{uv}\}$ and checks whether this set satisfies all constraints. If that is the case, a feasible solution is found, otherwise, the oracle returns one constraint which was violated by the input and a new solution satisfying all up to now reported constraints is computed. Constraints (3), (4), (5), and (6) can be checked explicitly in polynomial time. We cannot check the spreading constraints (7) explicitly since there are exponentially many, but we can make use of the following two observations.

The fist observations is, that if a set violates the spreading constraints, then the set of smallest edge weights also violates them. Hence, instead of checking the spreading constraints on all (independent) sets, we merely check them on those of smallest weight. For this reason, we have to compute the closest independent set of size k around vertex u for all $u \in V$ and check whether this set satisfies the spreading constraints. The second observation we make is, that we do not actually have to compute the closest independent set, but it suffices to approximate it. If an r-approximation to the closest set satisfies the constraint $r \cdot \kappa|IS|^{3/2}$ (which we check with the oracle) then clearly the closest set satisfies the constraint $\kappa|IS|^{3/2}$ and hence by the reasoning above *all* sets satisfy this constraint.

The point of Algorithm 5 is then to approximate the closest (in terms of distance) independent set around a node u for a given set size.

Algorithm 5 Minimal Weighted Independent Set

Input: distance matrix, G, vertex u, k
Output: independent set IS

```
maxdist ← max_{v∈V} x_uv
for i = 1 to (maxdist − 1) do
    V_i = {v | i < x_uv ≤ i + 1}
    IS_i ← maximal independent set of V_i
end for
s_even ← arg min_s s.t. Σ_{i=1}^{⌈s/2−1⌉} |IS_2i| ≥ k
s_odd ← arg min_s s.t. Σ_{i=1}^{⌈s/2⌉} |IS_2i−1| ≥ k
s ← min{s_even, s_odd}
if s = s_even then
    IS ← ⋃_{i=1}^{⌈s_even/2−1⌉} IS_2i
else
    IS ← ⋃_{i=1}^{⌈s_odd/2⌉} IS_2i−1
end if
repeatedly remove farthest v ∈ IS until |IS| = k.
```

LEMMA 6.1. *Algorithm 5 finds an independent set of weight at most a constant times the optimal.*

PROOF. Let IS^* denote the optimal minimum weighted independent set of size k as seen from node u, that is,

$$w(IS) := \sum_{v\in IS} x_{uv} \text{ , with } |IS \setminus \{u\}| = k$$

is minimal for IS^*. Let IS be the set returned by Algorithm 5. Let further m be the maximal index of V_i such that a vertex in IS^* is in layer V_i. Then

$$\sum_{i=1}^{\lceil m/2-1\rceil} |IS_{2i}| + \sum_{i=1}^{\lceil m/2\rceil} |IS_{2i-1}| \geq k/5,$$

since for each maximal independent set IS_i in a UDG it holds that $|IS_i| \geq |V_i \cap IS^*|/5$ (using the natural greedy algorithm [19]) and by definition of m. Hence,

$$\max\left\{\sum_{i=1}^{\lceil m/2-1\rceil} |IS_{2i}|, \sum_{i=1}^{\lceil m/2\rceil} |IS_{2i-1}|\right\} \geq k/10.$$

W.l.o.g. let the sum over the even indices be the maximum. Then

$$\sum_{i=1}^{\lceil s_{even}/2-1\rceil} w(IS_{2i}) \leq \frac{k}{10}\cdot m + \frac{9}{10}k\cdot 4m = 4km,$$

since at least $k/10$ vertices have distance at most m to u, whereas the remaining $9k/10$ vertices have distance at most $4m$ to u. (The input graph G is a connected unit disk graph and hence in each IS_i there is at least one vertex.) Now, since the value of m is unknown, we may have chosen the

final IS to be the sum over the odd indices (line 8), instead of the sum over the even indices. Then, $s_{odd} \leq s_{even} \leq 4m$ and

$$\sum_{i=1}^{\lceil s_{odd}/2 \rceil} w(IS_{2i-1}) \leq 4km$$

as above. The optimal solution must have at least one vertex per two layers. Hence, we can lower bound the weight of the optimal independent set by

$$w(IS^*) \geq \gamma' \sum_{i=1}^{m-1} i^2 \geq \gamma' \gamma m^3,$$

for constants γ' and γ, since there are at least m layers, and the distance of nodes in layer i is at least i.

It remains to establish a relationship between m and k. As the number of independent vertices in each layer i is bounded by the layer's circumference, which is linear in i, it holds that $|V_i \cap IS^*| \leq 15i$. Because IS^* contains exactly k nodes, at most $15i$ from each level i, it follows that $\sum_{i=1}^{m-1} 15i \geq k$ and therefore, m is at least $\sqrt{k}$ times a constant. The lemma now follows from $w(IS) \leq \beta k^{3/2}$, and $w(IS^*) \geq \alpha k^{3/2}$, with α and β being small constants. □

Note that we cannot compute a maximum independent set on G in polynomial time but a constant approximation thereof. This has implications only for Lemma 5.4 and this only in an additional multiplicative constant of 5.

COROLLARY 6.2. *The linear programming relaxation can be solved in polynomial time.*

PROOF. By the discussion above and Lemma 6.1. □

Together with Theorem 5.2, we obtain the following result.

THEOREM 6.3. *The virtual coordinates algorithm can be solved in polynomial time.*

6.2 Practical Considerations

To see the algorithm's actual performance (as opposed to its theoretical validity), we have also implemented it in a simulation environment. To that end, we have made a few practical changes which we describe below. A sample output is shown in Figure 1.

6.2.1 Faster Implementation

While the algorithm presented in this paper has indeed polynomial running time, in practical settings, it counts for a lot what the degree of that polynomial is. As we have presented it, the algorithm needs to take into account all independent sets of a graph G. In our alternative algorithm, we first compute a maximal independent set IS on G, use a modified version of the algorithm as presented in Section 4.2, and finally place the remaining nodes around the grid as in Algorithm 4.

The advantage of this version is that the input size for the linear program and the other steps are considerably smaller. The price we pay is in accuracy, since we only consider a single (maximal) independent set in the first step as opposed to every.

6.2.2 Mobility

Centralized algorithms do not fare well in the context of dynamic networks. We would not want a single movement to induce another run of the entire algorithm. Instead, we suggest using a "spring-like" algorithm (such as phase 1 of [24]) to adapt to local changes (the idea is that edges represent springs and we try to find the position with minimal potential spring energy). After a certain period of mobility, the centralized algorithm would be run to re-calibrate the network with more accurate coordinates.

6.2.3 Routing

Since some nodes can (and very likely will) have the same coordinate, we cannot use the standard geometric greedy routing algorithm (such as in [24]) on our computed virtual coordinates. There are two ways around this. One is to slightly perturb the coordinates of the nodes placed on the same position, so that they are some small ϵ apart instead. This will have a negligible impact on the quality of the approximation, yet might also cause more dead ends in the routing process. Another solution is to implement 2-hop or neighbor-of-neighbor (NoN) greedy routing where the next hop is chosen in the 2-hop instead of the 1-hop neighborhood. While this may seem a minor detail, it has recently been shown to be fruitful in general (see [18] and references therein).

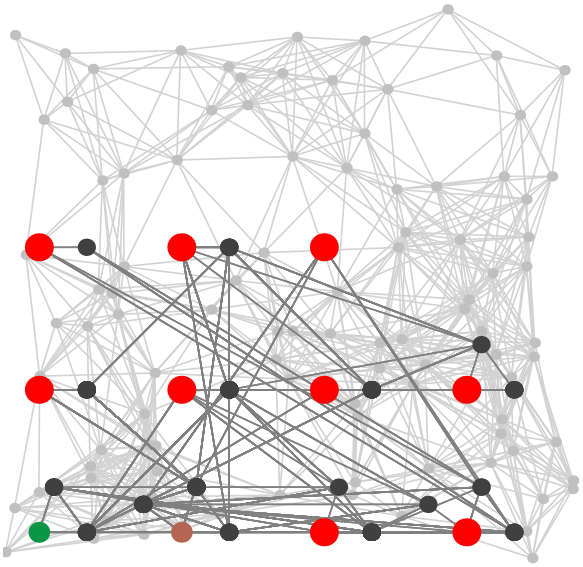

Figure 1: The output of our algorithm contrasted with the original graph. There are 100 nodes in an area of 4 by 4 radio units. The large red dots represent the initial maximal independent set MIS, the smaller black dots are the remaining nodes placed around their associated MIS node.

7. CONCLUSION

In this paper, we have given the first approximation algorithm for the connectivity-based virtual coordinates problem. While in its current state, the algorithm is hardly practical in wireless ad hoc or sensor networks, it is a first step beyond mere heuristics and simulation results. We believe

that it is only by gaining a thorough theoretical understanding of the various underlying problems that the full potential of virtual coordinates in ad hoc and sensor networks will ultimately be tapped.

8. ACKNOWLEDGMENTS

The authors would like to acknowledge Jiri Matoušek for pointing us to the relevant literature. We also thank Fabian Kuhn for helpful discussions.

9. REFERENCES

[1] M. Badoiu. Approximation Algorithm for Embedding Metrics into a Two-dimensional Space. In *Proc. of the ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2003.

[2] R. Bischoff and R. Wattenhofer. Analyzing Connectivity-based, Multi-hop Ad hoc Positioning. In *Proc. of the IEEE Intl. Conf. on Pervasive Computing and Communications (PerCom)*, 2004.

[3] P. Biswas and Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. In *Proc. of Intl. Symp. on Information Processing in Sensor Networks (IPSN)*, 2004.

[4] A. Blum, G. Konjevod, R. Ravi, and S. Vempala. Semi-definite Relaxations for Minimum Bandwidth and other Vertex-ordering Problems. *Theor. Comput. Sci.*, 235(1):25–42, 2000.

[5] 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.

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

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

[8] L. Doherty, L. E. Ghaoui, and K. Pister. Convex Position Estimation in Wireless Sensor Networks. In *Proc. of Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.

[9] G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer Approximation Algorithms via Spreading Metrics. In *Proc. of the IEEE Symp. on Foundations of Computer Science (FOCS)*, 1995.

[10] U. Feige. Approximating the Bandwidth via Volume Respecting Embeddings. *J. of Computer and System Sciences*, 60(3):510–539, 2000.

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

[12] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large Scale Sensor Networks. In *Proc. of Mobile Computing and Networking (MobiCom)*, 2003.

[13] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning Systems: Theory and Practice*. Springer, 5th edition, 2001.

[14] B. Karp and H. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. of Mobile Computing and Networking (MobiCom)*, 2000.

[15] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit Disk Graph Approximation. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 2004.

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

[17] N. Linial, E. London, and Y. Rabinovich. The Geometry of Graphs and some of Its Algorithmic Applications. *Combinatorica*, 15(2):215–245, 1995.

[18] G. Manku, M. Naor, and U. Wieder. Know Thy Neighbor's Neighbor: The Power of Lookahead in Randomized P2P Networks. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, 2004.

[19] M. V. Marathe, H. Breu, H. B. H. III, S. S. Ravi, and D. J. Rosenkrantz. Simple Heuristics for Unit Disk Graphs. *Networks*, 25:59–68, 1995.

[20] R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. In *Proc. of Information Processing in Sensor Networks (IPSN)*, 2003.

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

[22] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS). In *Proc. of IEEE Global Communications (GLOBECOM)*, 2001.

[23] D. Niculescu and B. Nath. Error characteristics of ad hoc positioning systems. In *Proc. of Intl. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

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

[25] C. Savarese, J. Rabaey, and K. Langendoen. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In *Proc. of USENIX Technical Conference*, 2002.

[26] A. Savvides, C.-C. Han, and M. Srivastava. Dynamic Fine-Grained Localization in Ad-Hoc networks of Sensors. In *Proc. of Mobile Computing and Networking (MobiCom)*, 2001.

[27] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from Mere Connectivity. In *Proc. of Intl. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[28] I. G. Tollis, R. Tamassia, G. D. Battista, and P. Eades. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1998.

[29] S. Vempala. Random Projection: A New Approach to VLSI Layout. In *Proc. of the IEEE Symp. on Foundations of Computer Science (FOCS)*, 1998.

[30] S. Vempala. *The Random Projection Method*. Dimacs Series in Discrete Mathematics and Theoretical Computer Science, 2004.