Anchor-Free Distributed Localization in Sensor Networks

Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, and Seth Teller
Tech Report #892, April 15, 2003
MIT Laboratory for Computer Science
http://nms.lcs.mit.edu/cricket/

April 8, 2003

Abstract

Many sensor network applications require that each node's sensor stream be annotated with its physical location in some common coordinate system. Manual measurement and configuration methods for obtaining location don't scale and are error-prone, and equipping sensors with GPS is often expensive and does not work in indoor and urban deployments. Sensor networks can therefore benefit from a self-configuring method where nodes cooperate with each other, estimate local distances to their neighbors, and converge to a consistent coordinate assignment.

This paper describes a fully decentralized algorithm called *AFL* (Anchor-Free Localization) where nodes start from a random initial coordinate assignment and converge to a consistent solution using only local node interactions. The key idea in AFL is *fold-freedom*, where nodes first configure into a topology that resembles a scaled and unfolded version of the true configuration, and then run a force-based relaxation procedure. We show using extensive simulations under a variety of network sizes, node densities, and distance estimation errors that our algorithm is superior to previously proposed methods that incrementally compute the coordinates of nodes in the network, in terms of its ability to compute correct coordinates under a wider variety of conditions and its robustness to measurement errors.

1 Introduction

Physical location is an important attribute of a sensor's data stream in a large number of sensor network applications. In addition, geographic information, for instance in the form of node coordinates in some common coordinate system, is a useful primitive in routing protocols such as geographic routing [16], information dissemination protocols such as directed diffusion using location attributes [15], and sensor query processing systems [17].

This paper presents a method to facilitate large-scale deployment of location-aware sensor networks. The main idea of this paper is to show that large networks of location-aware sensors can be made cooperatively self-configuring, that is, that each sensor can run an algorithm locally, interacting only with neighboring nodes, such that after a number of iterations all sensors will have reached a consensus about their coordinates in some coordinate system. By doing this in an automated manner, large-scale sensor networks can eliminate the cumbersome and unscalable process of manually configuring sensor nodes with their location.

In non-urban outdoor settings, nodes may obtain location information using an existing infrastructure such as GPS [13]. However, GPS receivers may be too expensive, too large, or too power-intensive for the desired application. In many outdoor urban environments, and most indoor environments, GPS is not available. One solution to this problem is an alternative location infrastructure such as Bat [11] or Cricket [20] that works in places that GPS does not. Another solution to these problems is to equip sensors with hardware capable of estimating distances to nearby nodes, and to have the sensors themselves self-configure into a consistent coordinate system.

In contrast to the GPS with its few dozen satellites and ground-based monitoring centers, alternative infrastructures such as Cricket employ hundreds or thousands of inexpensive position beacons to provide location information over extended areas. This large number of nodes raises a deployment issue: how can an extended deployment be efficiently configured for initial operation, and efficiently maintained for continuing operation? Autonomously operating location-aware sensor networks face the same problem.

This paper solves the following problem: *Given a set of nodes with unknown position coordinates, and a mechanism by which a node can estimate its distance to a few nearby (neighbor) nodes, determine the position coordinates of every node via local node-to-node communication.* Our solution to this problem is fully decentralized: all nodes start from a random initial coordinate assignment and use only local distance estimates to converge to a coordinate assignment that is consistent with the distance estimates by exchanging only local information. The resulting coordinate assignment has translation and orientation degrees of freedom, but is cor-

rectly scaled. A post-process could incorporate absolute position information into three or four nodes (from building conventions, survey data, or GPS) to remove the translation and orientation degrees of freedom.

Some previous work on this problem assumes that a non-negligible fraction of nodes in the network are *anchor nodes* that already know their location [2, 7, 21, 23]. In contrast, we pursued an anchor-free approach for three reasons. First, establishing anchors is a manual deployment task, and may be cumbersome. Second, the numerical stability of anchor-based approaches is questionable, since they give more weight to anchor position estimates, and errors in those estimates will have undue effect on the global solution. Finally, anchor-based approaches may not scale well, since to combat the instability described above, a large number of anchors may be required to configure an unbounded working area.

Another class of algorithms proceed *incrementally*, starting from a small core set of nodes that know their location, and adding nodes to an existing, configured network one at a time or in groups [21]. This can be done if a node attempting to join the existing network can successfully estimate its distances to three or four nodes that are already configured. We show in this paper that such incremental approaches have two major problems: first, they may not solve the problem even when a valid coordinate assignment exists, and second, errors in local distance estimates often tend to cascade, leading to large global error.

Our contribution is an algorithm called *AFL* (Anchor-Free Localization), a *concurrent* and *anchor-free* solution to the problem. We show that this combination has significant advantages over several previous approaches. However, realizing such an algorithm is a non-trivial problem, because concurrent approaches tend to fall into false minima, where each node believes it is in an optimal position but the global configuration is incorrect. In particular, the classic approach to obtaining a concurrent algorithm is to model the nodes as point masses connected using springs, and use force-directed relaxation methods to converge toward a minimum-energy configuration. Such force-directed methods are susceptible to severe false minima.

The key idea in AFL that alleviates the false-minimum problem is *fold-freedom*. Based on the observation that many false minima are caused because nodes operating on local information converge falsely to configurations where groups of nodes are topologically folded with respect to the true configuration, AFL seeks to first configure nodes into a "fold-free" configuration that is (loosely speaking) a scaled-up, unfolded and locally distorted version of the true configuration. After this is done, the nodes run a force-based relaxation procedure taking care to not seriously violate fold-freedom. The result is a correct solution, or graph embedding, for a large class of input networks in practice. While we don't yet have a proof of correctness for exactly when AFL works and when it does not, we show using extensive simulations under a range of network sizes, node connectivity and densities, and distance estimation errors that AFL outperforms incremental algorithms by both being able to converge to correct positions when incremental algorithms do not, and by being significantly more robust to errors in local distance estimates.

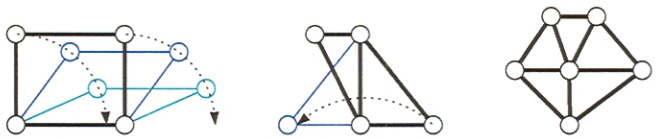

Figure 1: Examples of graphs that are not rigid (flexible as a bar-and-joint framework), rigid but not globally rigid (multiple embeddings), and globally rigid (one embedding up to rotation, translation, and reflection).

2 Terms and definitions

This section defines useful notation and terms, and formally defines the problem. For ease of exposition, we restrict our definitions to two dimensions, but AFL applies to three-dimensional node placement as well.

2.1 Problem definition

Consider N nodes labeled $1, \ldots, N$ at unknown distinct locations in some physical region. We assume that some mechanism exists through which each node can discover its *neighbor* nodes by establishing communication with those nodes, and can estimate the range (separation distance) to each of its neighbors. For example, neighbor information may be obtained using radio links, while range information may be obtained using radio coupled with ultrasonic or acoustic signals. Each discovered neighbor relationship contributes one undirected edge $e = (i, j)$ in a graph G over the nodes. We denote by r_{ij} the *range*, or estimated distance, between nodes i and j, and by d_{ij} the actual distance between nodes i and j.

Given a collection of N nodes, and the distance measurements of each node to its neighbors, the goal is to produce a set of coordinate assignments p_i that are *consistent* with all distance measurements, that is, an assignment of points p_i for all i and j such that the distance between i and j $\|p_j - p_i\| = d_{ij}$ for all $e \in G$. Note that this position assignment can be unique only up to an arbitrary rotation, translation, and possible reflection, but its scale is determined by the measured ranges.

However, for some graphs, the position assignment is not unique even up to rotation, translation, and reflection. Refer

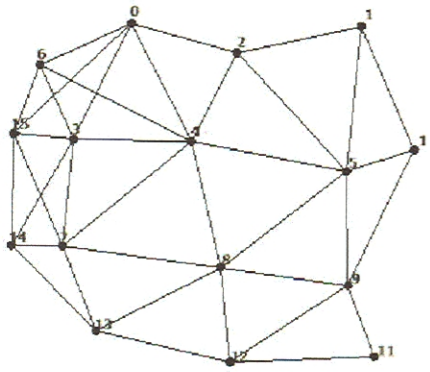

Figure 2: A graph consisting of 16 nodes, displayed at their true positions.

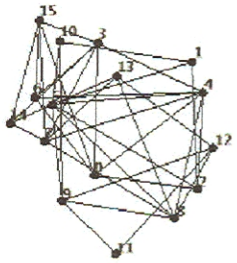

Figure 3: A modified version of the graph in Figure 15 where the average edge length error is 5%.

to Figure 1. If we treat the graph as a bar-and-joint framework, the graph should be *rigid* in the sense that it cannot be flexed while preserving the distances (as in a rectangle, for example). Even if the graph is rigid, it may be subject to "local flips". For example, if there are just two triangles sharing an edge, one triangle can be reflected through that edge without any distances changing. We call such a graph *rigid* but not *globally rigid*. For autolocalization to work given just edge lengths, we need a *globally rigid* graph that has exactly one embedding. We elaborate on this connection to rigidity theory, and known results about globally rigid graphs, in Section 3.2.

Even if the graph is globally rigid and has a unique embedding, it is NP-hard to find the correct embedding in general [24, 25]. Most solutions have failure modes where they fall into false minima or simply don't work for certain topologies (*e.g.*, incremental methods don't work well unless the node density is high). Furthermore, in practice, distance estimation errors occur, which may cascade in certain solutions to produce highly erroneous configurations.

2.2 Performance metric

In previous work on the localization problem, researchers have used the average percentage error of the calculated distances compared to the true distance between neighbors as a measure of the algorithm's performance [2, 21]. While instructive, this metric does not fully capture the intended goal of the configuration algorithm, which is to produce coordinate assignments that "resemble" the true configuration's topological properties. For example consider the graph of nodes in Figures 3 and 4. These figures show two modified versions of the graph in Figure 2 where the average error ratio is 5% compared to the original graph. Compared to the graph in Figure 3, it is visually obvious that the graph in Figure 4 is a much better approximation of the true configuration. Simply reporting the average position error does not capture the true behavior of auto-localization algorithms.

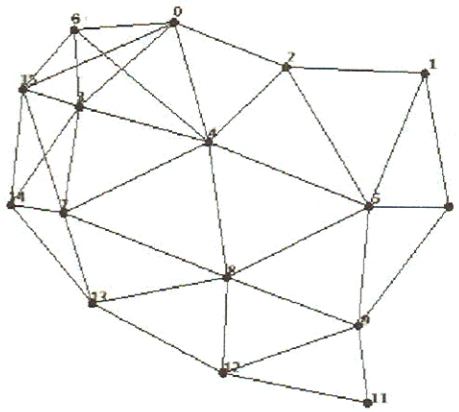

Figure 4: Another modified version of graph in Figure 15 where the average edge length error is 5%.

To capture this global structural property, we introduce a metric called the *Global Energy Ratio (GER)*. The global energy ratio is the root-mean-square normalized error value of the node-to-node distances, where the error e_{ij} is the difference between the true distance d_{ij} and the distance in the algorithm's result $\hat{d}_{ij}$, and $\hat{e}_{ij}$ is the normalized error, equal to $\frac{\hat{d}_{ij}-d_{ij}}{d_{ij}}$.

$$\text{GER} = \frac{\sqrt{\sum_{i,j:i<j} \hat{e}_{ij}^2}}{N(N-1)/2}. \quad (1)$$

This measure captures both the edge length errors and the structural error of the graph, because it has contributions from

both nodes that are neighbors as well as nodes that are not. As an example, the GER of the configuration in Figure 3 is 0.034, while the GER of the configuration in Figure 4 is 0.005. If all the estimated ranges r_{ij} between neighbor nodes are equal to the true d_{ij} values, and if the true configuration is rigid, then an ideal algorithm will produce a result whose GER = 0. Because $\hat{e}_{ij}$ compares $\hat{d}_{ij}$ with d_{ij} rather than with r_{ij}, the GER metric also captures the errors in the final configuration caused by erroneous range estimates.

3 Related work

Previous research has addressed various versions of the distributed localization problem. We characterize the distributed algorithms developed to solve this problem in two different ways. The first characterization is according to whether or not they rely on *anchor nodes*, which are nodes that are preconfigured with their true position. The second is based on whether they are *incremental* or *concurrent* algorithms.

Anchor-based algorithms. Algorithms that rely on anchor nodes assume that a certain minimum number or fraction of the nodes know their position, e.g., by manual configuration or using some other location mechanism. The final coordinate assignment of individual nodes will therefore be valid with respect to another possibly global coordinate system. Any positioning scheme built around such algorithms has the limitation that it needs another positioning scheme to bootstrap the anchor node positions, and cannot be easily applied to any context in which another location system is unavailable (e.g., strictly interior to a building). It turns out that in practice a large number of anchor nodes are needed for the resulting position errors to be acceptable [21].

Anchor-free algorithms. In contrast, anchor-free algorithms use local distance information to attempt to determine node coordinates when no nodes have pre-configured positions. Of course, any such coordinate system will not be unique and can be embedded into another global coordinate space in infinitely many ways, depending on global translation, rotation, and possibly flipping. This limitation is fundamental to the problem specification, and is not a limitation of the algorithm.

If the coordinates assignments must conform to another coordinate system such as GPS, any algorithm that does not use anchor nodes can easily be converted to a one that uses a small number of anchor nodes by adding a final transformation where all the node coordinates are transformed using three (in 2D) or four (in 3D) anchor nodes.

Incremental algorithms. These algorithms usually start with a core of three or four nodes with assigned coordinates. Then they repeatedly add appropriate nodes to this set by calculating the node's coordinates using the measured distances to previous nodes with already computed coordinates. These coordinate calculations are based on either simple trigonometric equations or some local optimization scheme.

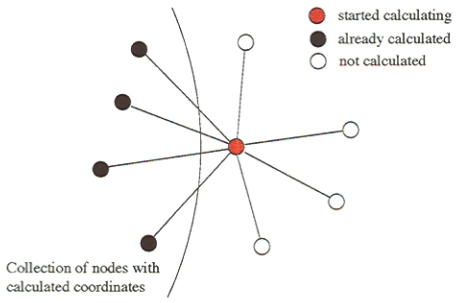

Figure 5: Nodes involved in a typical incremental optimization.

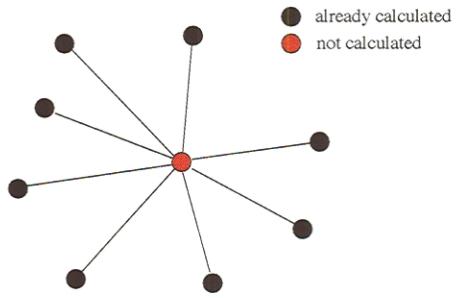

Figure 6: Nodes involved in a typical concurrent optimization.

A drawback of incremental algorithms is that they propagate measurement errors, resulting in poor overall coordinate assignments. Some incremental approaches apply a later global optimization phase to balance such error, but it remains difficult to jump out of local minima introduced by the local optimization in the incremental phase.

Concurrent algorithms. In these algorithms, all the nodes calculate and refine their coordinate information in parallel. Some of these algorithms use an iterative optimization scheme that reduces the difference between measured distances and the calculated distances based on current coordinate estimates.

Concurrent optimization schemes have a better chance of avoiding local minima compared to incremental schemes especially under measurement errors, because they continually balance global error and thereby try to avoid error propagation. For example, consider Figure 5, which shows node positions from a typical incremental optimization scheme; in contrast, Figure 6 shows the same set of nodes involved in typical concurrent optimization. As we can see, the layout of the nodes involved in these optimizations more frequently results in an incorrect coordinate assignment (or local minima) for the incremental scheme compared to the concurrent scheme. A more thorough experimental comparison can be found in Section 5.

Previously proposed concurrent algorithms almost always use anchor nodes. The anchor nodes with known position information help avoid local minima during the optimization. In contrast, AFL avoids the use of anchor nodes, while its initial phase of building a fold-free configuration helps avoid local minima during the optimization.

3.1 Previous Auto-localization Systems

Doherty *et al.* describe an anchor-based algorithm for localization using only connectivity constraints among beacons. They represent the connectivities as a set of convex position constraints, and use a centralized linear-programming algorithm to solve for the node positions.

Bulusu *et al.* describe a GPS-less scheme that uses the radio connectivity of a node to a set of anchor nodes to determine its coordinates [2]. The coordinates of non-anchor nodes are obtained by calculating the centroid of all the anchors in the nodes radio-range. This is a concurrent algorithm, but it does not use any optimization. In simulations, they achieve about 12% localization error with approximately 12 anchor nodes per non-anchor node ($\frac{R}{d} = 2$ where R is the radio range and d is the separation between anchors). The ratio of the anchor nodes to non-anchor nodes is rather large.

The ABC algorithm is an incremental algorithm that does not use anchor nodes [21]. ABC first selects three in-range nodes and assign them coordinates to satisfy the inter-node distances, then it incrementally calculates the coordinates of nodes using the distances to three nodes with already calculated coordinates. This simple incremental scheme results in error propagation. The authors report that with 5% range error, ABC results in about 60% average position error, which is larger than tolerable in many situations. This is a consequence of cascading errors in incremental solutions.

The Terrain algorithm, another anchor-based algorithm, builds on ABC [21]. Each anchor starts the ABC algorithm. Using the coordinates assigned using ABC, each node calculates the distances to at least three anchors. Then each node performs a concurrent optimization using the distances to the anchors and the anchor coordinates. The authors report about 25% position error (actual offset of the node position from the true position) with 5% range error. They also mention that position errors show a high variance and possibility of divergence during the optimization phase. A related algorithm for localizing nodes in an ad hoc network uses hop-count and radio strength as distance measures, but assumes nearly uniform node density and no occlusion [18].

Savarese *et al.* describes a two-phase, anchor-based, concurrent, localization algorithm [22]. The first phase of the algorithm, Hop-Terrain, is a variant of Terrain, and is robust against ranging errors. The second phase is a simulated-annealing based optimization. With 5% range errors, 10% of the nodes being anchors, and 12 neighbors per node, this algorithm results in about 12% average position error.

	Incremental	Concurrent
Anchor-based	Collaborative-multilater'n [23] AOA [19]	Terrain [21] Hop Terrain [22] GPS-less [2]
Anchor-free	ABC [21]	AFL (this paper)

Table 1: A characterization of localization algorithms.

Savvides *et al.* describe a *collaborative multilateration* scheme, an anchor-based localization algorithm [23]. Here, a node solves a set of over-constrained equations relating the distances among a set of anchors and a set of non-anchor nodes (including itself). For a sample graph of 300 nodes, the algorithm needs about 30 (10%) anchor nodes to calculate the location of the other nodes. *Iterative multilateration*, an incremental component of their algorithm, produces node position errors within 20cm of a node's actual positions, when the ranging error is small (2cm, Gaussian-distributed). This experiment consists of 50 nodes, with a 3m ranging system, deployed in a square grid of 15 × 15m, and with 10% of the nodes being anchors.

Niculescu *et al.* present an anchor-based, distributed algorithm that uses *angle-of-arrival* (AOA) for localization [19]. In this algorithm, nodes iteratively obtain position and orientation information starting from anchor (landmark) nodes. One potential problem with this approach is that using angle-of-arrival is expensive and obtaining precise angle estimates is often difficult.

Howard *et al.*'s localization scheme using spring-based relaxation is perhaps the closest to our work [14]. In their system, robots equipped with odometric equipment move through an environment, seeding beacons with approximate initial positions, from which the beacons run a distributed relaxation procedure. While the problem setup is different from ours because of the assumption of having active robots to seed the system, we discuss certain similarities in Section 4 when we describe the details of AFL.

Bulusu *et al.* study the performance characteristics of different RF-based beacon configuration algorithms and conclude that node density is an important determinant of performance [3]. This paper also contains a detailed survey of various beacon-based localization schemes.

Table 1 categorizes these algorithms according to the taxonomy developed earlier. Because AFL does not use anchor nodes, it can provide localization without an existing location system. AFL uses a concurrent optimization scheme that is robust against measurement errors.

3.2 Previous Geometric Work

Given an abstract graph with a specified length (positive real number) for each edge, when can the graph be embedded into 2D or 3D while satisfying the edge lengths? When is such an embedding unique (up to global translation, rotation, and reflection), and therefore a reliable reconstruction of the desired geometry? Both of these questions have received considerable attention in both the discrete geometry and computational geometry communities.

Deciding whether a graph with edge lengths can be embedded is NP-hard in general [25]. Basically, triangles form rigid structures but can be independently flipped (folded), and deciding whether a string of triangles can be folded left and right to make a particular length is equivalent to subset sum. Saxe [24] proved the stronger result that the problem is strongly NP-hard even for embedding into 1D.

A graph with specified edge lengths which has a unique embedding is called *globally rigid* [4, 12], a variation on the well-studied concepts in *rigidity theory* [5, 10, 9]. Because global rigidity can be expressed as the uniqueness of a solution to a system of algebraic constraints (specifying distances between some pairs of vertices), global rigidity is almost always a property of the underlying graph, not the specific edge lengths. ("Almost always" is a measure-theoretic notion, meaning "with probability 1" under any reasonable probability distribution.) A graph (without edge lengths) is *generically globally rigid* if, for almost any realizable assignment of lengths to the edges, it is globally rigid.

Hendrickson [12] showed that, for a graph to be generically globally rigid in d dimensions, it must be $(d+1)$-connected and the removal of any edge must leave the graph "generically rigid" [5, 10, 9]. Both of these properties can be checked in polynomial time. Connelly [4] proved that these two properties are not enough: they do not imply generic global rigidity in 3D. However, Hendrickson conjectures that these two properties are enough, exactly characterizing generic global rigidity, in 2D.

Embedding a graph with given edge lengths also arises in the context of reconstructing the geometry of molecular structures in an area called *distance geometry*; see e.g. [6]. In this context, distance measurements are substantially less accurate, and several techniques have been developed to refine estimates and reduce error bounds by combining several constraints. On the algorithmic side, Berger et al. [1] give efficient algorithms for embedding a graph with error-prone edge lengths, even when nearly half of the edges might have completely inaccurate lengths. However, these algorithms rely on every node having a constant fraction of the nodes as neighbors, for a total of $\Omega(n^2)$ links between n nodes, which does not scale in our context.

4 AFL algorithm

4.1 Overview

The AFL algorithm proceeds in two phases. The first phase is a heuristic that produces a fold-free graph embedding which "looks similar" to the original embedding. The second phase uses a mass-spring based optimization to correct and balance localized errors. We begin with a summary of the second phase to illustrate the need for and importance of the first phase.

To understand the importance of fold-freedom, consider the classical mass-spring optimization method. Here, we imagine each edge in the graph as a spring between two masses, with a rest length equal to the measured distance between the two nodes. If the current estimated distance between two nodes is greater than their true (measured) length, the spring incurs a force that pushes them apart. On the other hand, if the estimated distance is larger than the true distance, a force pulls them together. Different mass-spring schemes define the magnitude of these forces differently, but the optimization proceeds essentially in the same iterative manner: at each step, nodes move in the direction of the resultant force. At any node, the optimization stops when the resultant force acting on it is zero; the global optimization stops when every node has zero force acting on it. In the optimization, if the magnitude of the force between *every* pair of neighbor nodes is also zero (*i.e.*, the global energy of the system measured as the sum of squares of the forces is zero), then the optimization has reached the global minimum; otherwise, it has reached a local minimum.

Mass-spring optimization is used heavily in the field of *force-directed graph drawing* [8]. In force-directed graph drawing, the mass-spring model and optimization are used to find some local minima ideally representing "nice" drawings of graphs. Howard et al. [14] describe the use of this technique for general localization. In this paper, the authors mention that the mass-spring approach can converge to local minima rather than the global minimum. This is the fundamental problem with unconstrained mass-spring optimizations—when nodes start with a random initial coordinate assignment, mass-spring optimization has a high probability of converging to local minimum. For example, Figure 8 shows the graph we obtain by applying spring-mass optimization to the graph in Figure 7 with random initial coordinate assignments. In Section 5, we show that mass-spring based optimization indeed has a high probability of reaching a local minimum.

Through simulations we observed that local minima in a spring-based optimization are most often characterized by sections of the graph "folding over" with respect to the true configuration. Because folds involve *groups of nodes* that have all folded over, their local interactions are both correct and strong and there isn't enough resultant force exercised by

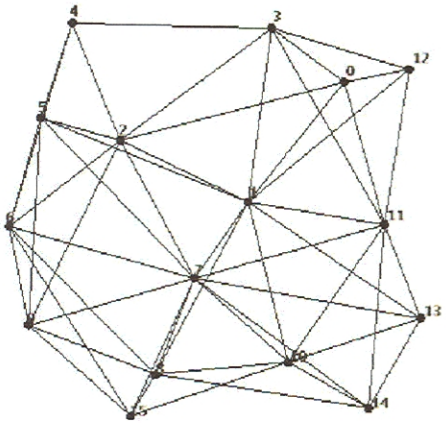

Figure 7: A graph with sixteen nodes with nodes at their true positions.

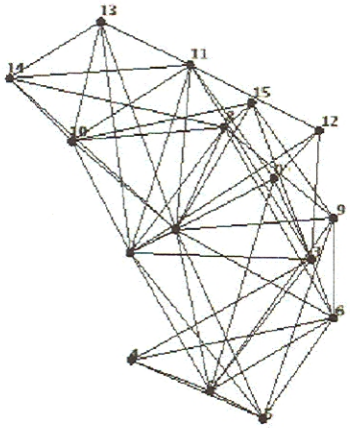

Figure 8: The graph obtained by applying mass-spring optimization to the graph in Figure 7 with a random initial coordinate assignment.

the nodes neighboring the fold to unfold the group and improving the global energy. Thus, the goal of the first phase of AFL is to design an initial "fold-free" coordinate assignment. In fact, our observation of folds causing local minima may shed light on a point made by Howard *et al.* [14], who found that local minima did not seem to occur frequently in their experimental setup. Their experiments were done with robots equipped with odometric equipment moving around to provide initial coordinates to beacons, and this in fact led to approximately fold-free initial configurations from which the force-directed optimizations work better.

4.2 Generating a fold-free configuration

The goal of the first phase of AFL is to embed the graph structurally similar to the original embedding. More specifically, the algorithm tries to avoid folds in the resulting graph compared to the original graph. We formally define a *fold-free embedding* of a graph to be one where every cycle of the embedding has the correct clockwise/counterclockwise orientation of nodes, modulo global reflection, with respect to the original graph.[1] We do not guarantee that our heuristic produces such an embedding, but it is our motivating principle. Our heuristic applies to both 2D and 3D graphs, but for clarity we focus on the 2D version; the 3D version is a simple extension. It operates in distributed fashion.

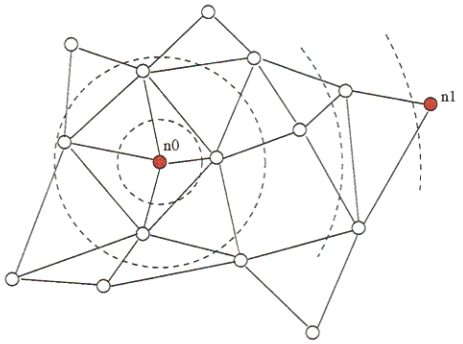

Figure 9: First step of the fold-free phase – electing n_1.

We start with some terminology and assumptions. We assume that each node has a unique identifier; the identifier of node i is denoted by ID_i. We use the phrase *hop-count between nodes i and j* to mean the number of nodes $h_{i,j}$ along the shortest radio path between nodes i and j. We assume symmetrical links between nodes, making the graph is undirected, so that $h_{i,j} = h_{j,i}$. In practice, this heuristic works on a neighbor graph that assumes only radio connectivity, without using accurate ranging information from other technologies like ultrasound.

The algorithm first elects five reference nodes. Four of these nodes n_1, n_2, n_3, and n_4 are selected such that they are on the periphery of the graph and the pair (n_1, n_2) is roughly perpendicular to the pair (n_3, n_4). The node n_5 is elected such that it is in the "middle" of the graph. These five nodes are elected in five steps.

- **Step 1.** Select an arbitrary node n_0—a simple way to achieve this in distributed fashion is to pick the node with smallest ID. Then, select the reference node n_1 to *maximize* $h_{0,1}$; *i.e.*, n_1 is a node that is the maximum hop-count away from node n_0 (Figure 9). Any ties are broken using the node's ID.
- **Step 2.** Select reference node n_2 to *maximize* $h_{1,2}$ (Figure 10). Any ties are broken using the node's ID.
- **Step 3.** Select reference node n_3 to *minimize* $|h_{1,3} -$

[1] This notion is similar to the *combinatorial embedding* used in planar graphs, and the *order type* used in point sets / complete graphs.

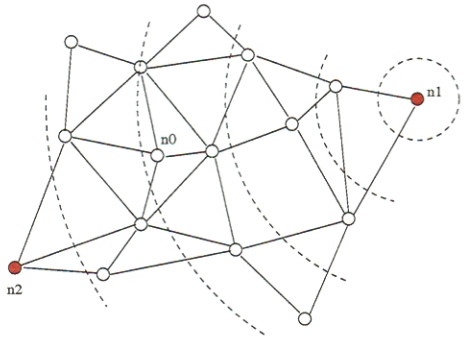

Figure 10: Second step of the fold-free phase – electing n_2.

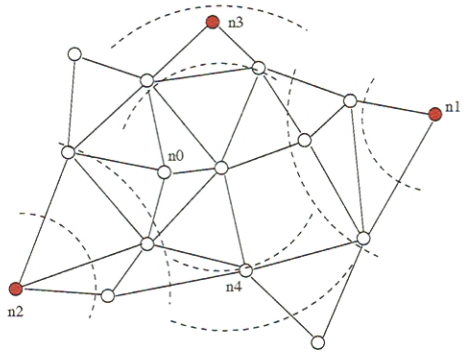

Figure 11: Third step of the fold-free phase – electing n_3.

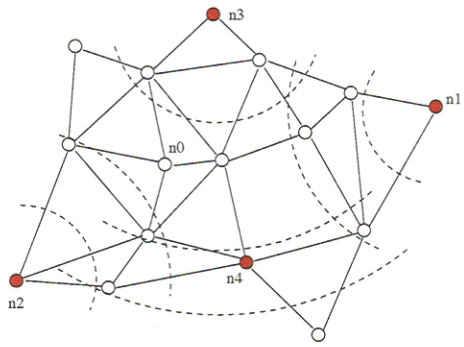

Figure 12: Fourth step of the fold-free phase – electing n_4.

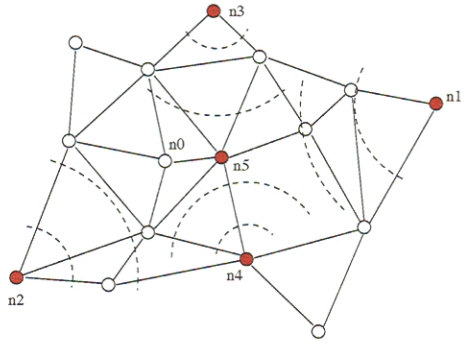

Figure 13: Fifth step of the fold-free phase – electing n_5.

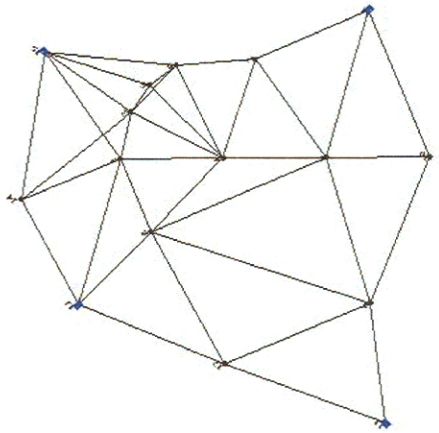

Figure 14: The graph obtained after running the fold-free phase on Figure 2(zoomed out).

$h_{2,3}|$. In general, several nodes may all have the same minimum value, and the tie-breaking rule is to pick the node that *maximizes* $h_{1,3} + h_{2,3}$ from the contenders.

This step selects a node that is roughly equidistant from nodes n_1 and n_2) and is "far away" from n_1 and n_2 (Figure 11).

- **Step 4.** As in the previous step, select reference node n_4 to *minimize* $|h_{1,4} - h_{2,4}|$. Now, break ties differently: from among several potential contender nodes, pick the node that *maximizes* $h_{3,4}$. This optimization selects a node roughly equidistant from nodes n_1 and n_2) while being farthest from node n_3 (Figure 12).

- **Step 5.** As in the previous step, select reference node n_5 to *minimize* $|h_{1,5} - h_{2,5}|$. From the contender nodes, pick the node that *minimizes* $|h_{3,5} - h_{4,5}|$. This optimization selects the node representing the rough "center" of the graph (Figure 13).

For all other nodes n_i, the heuristic uses the hop-counts $h_{1,i}$, $h_{2,i}$, $h_{3,i}$, $h_{4,i}$,and $h_{5,i}$ from the chosen reference nodes to approximate the polar coordinates $(\theta_i, \ldots)$. Here, R is the maximum radio range.

$$\theta_i = h_{5,i} \times R$$

$$\ldots = \ldots\left(\frac{h_{1,i} - h_{2,i}}{h_{3,i} - h_{4,i}}\right)$$

This coordinate assignment roughly approximates the true layout of the graph, especially for graphs that "radiate out" from a central point. Figures 2 and 14 show the shapes of a sample original embedding and the embedding we obtain by the first phase's approximate coordinate assignment. When calculating θ, the use of range R to represent one hop-count

results in a graph which is physically larger than the original graph; this property of the graph helps avoid local minima during the optimization phase.

4.3 Mass-spring optimization

The second phase of the AFL algorithm runs concurrently at each node. The nodes run the mass-spring optimization described below.

At any time, each node n_i has a current estimate $\hat{p}_i$ of its position. Each node n_i also periodically sends this position estimate to all its neighbors. Now, each node knows its own estimated position and the estimated position of all its neighbors.

Using these position estimates, each node n_i calculates the estimated distance $\hat{d}_{i,j}$ to each neighbor n_j. It also knows the *measured* distance $r_{i,j}$ to each neighbor n_j.

Let $\hat{v}_{i,j}$ represent the unit vector in the direction from $\hat{p}_i$ to $\hat{p}_j$. The force $\vec{F}_{i,j}$ in the direction $\hat{v}_{i,j}$ is given by

$$\vec{F}_{i,j} = \hat{v}_{i,j}(\hat{d}_{i,j} - r_{i,j}). \quad (2)$$

The resultant force on the node i is given by

$$\vec{F}_i = \sum_{i,j} \vec{F}_{i,j}.$$

The energy $E_{i,j}$ of nodes n_i and n_j due to the difference in the measured and estimated distances is the square of the magnitude of $\vec{F}_{i,j}$, and the total energy of node i is equal to

$$E_i = \sum_j E_{i,j} = \sum_j \left(\hat{d}_{i,j} - r_{i,j}\right)^2.$$

The total energy of the system E is given by

$$E = \sum_i E_i.$$

The energy E_i of each node n_i reduces when it moves by an infinitesimal amount in the direction of the resultant force $\vec{F}_i$. The exact amount by which each node n_i moves is important for two reasons. First we must ensure that the new position has a smaller energy than the original position; second, we have to ensure that such movement does not result in a local minima.

AFL can guarantee the first condition by calculating the energy at the new location before moving there to guarantee that the energy reduces. But there is no simple way to guarantee that the move does not result in a local minima. We empirically selected that each node moves by the amount $|\vec{F}_i|/(2m_i)$, inversely proportional to the number of neighbors of n_i.

Section 5 shows that AFL has a low probability of converging to local minima. Even if the graph reaches a local minimum, the fraction of nodes that get displaced tends to be small, thus causing only a small deformation in the resulting graph.

5 Simulation results

We simulated the performance of AFL varying graph size, node connectivity, and ranging error. We evaluated its performance against an incremental scheme and a pure mass-spring based approach that did not use fold-freedom. We wrote a Java3d-based simulator to experiment with, analyze, and visualize the performance and behavior of the different localization algorithms.[2]

All the simulations presented here are 2D simulations. We model ranging error using a uniform random distribution, as a fraction of the true distance between any two nodes. We select a single sample from the distribution to represent the error, rather than collecting multiple samples over time and averaging them. Our simulations therefore present a worst-case scenario, because averaging a number of samples whose errors are symmetric about the mean eliminates ranging error. In practice, the hard errors to overcome are one-sided [20], for which our experimental method is appropriate.

We select a range to represent the distance over which nodes can communicate. For a given range R, any two nodes whose distance is less than R are connected by an edge on the graph.

In all the simulations, we take necessary precautions to reduce the possibility of non-rigid graphs, this becomes very important when we do simulations with low connectivity. When deploying nodes we try to maintain a uniform local density by adding nodes to only those positions that have a number of neighbors below a certain threshold (we select this threshold based on the average connectivity of the graph).

5.1 Mass-spring without fold-freedom

In our first experiment, we study the performance of a pure mass-spring based algorithm without fold-freedom. We simulate graphs of 30, 100, and 300 nodes, with average per-node connectivities of four, eight and twelve for each case. For each combination of the graph size and the connectivity we run 20 simulations each on a random topology.

All of these $3 \cdot 4 \cdot 20 = 240$ *simulations resulted in local minima.* We detect the onset of a local-minimum condition when the square sum of distance errors do not change by more than 0.1%, at which time we check the GER to determine if the graph has reached a global or local minimum.

The results of these experiments validates our hypothesis that a pure mass-spring algorithm does not work without good ini-

[2] We plan to release the simulator and visualizer in the public domain.

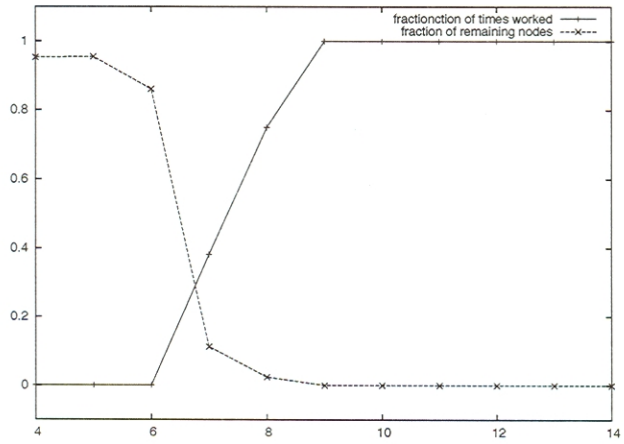

Figure 15: The fraction of the time a pure incremental scheme did not work and the fraction of nodes that could not be localized.

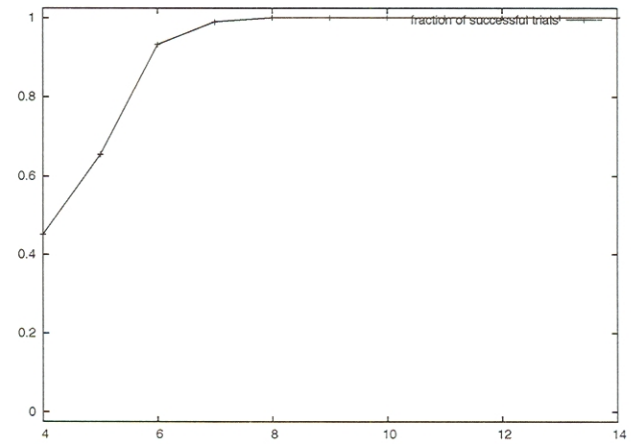

Figure 16: The fraction of the time our algorithm could not localize a graph.

tial position estimates. In all these cases, we were able to verify that the reasons for local minima were *folded configurations*.

5.2 AFL v. incremental scheme: Node connectivity

As discussed in Section 3, we can approach the anchor-free localization problem either incrementally or concurrently (as in AFL). The goal of our second set of experiments is to examine the performance of an incremental anchor-free scheme under different degrees of connectivity and under different ranging errors. Our hypothesis is that AFL's concurrent approach is able to work in more cases than the incremental approach.

We deploy a number of nodes in a square area. The node deployment is random; we select a random point in the square to place a node and check if the number of nodes within range of that node is less than n, the degree of connectivity we are trying obtain. Then, we adjust the graph by repeatedly removing any node that is connected to rest of the nodes by less than three links. This is essential for a fair analysis of the positioning scheme since a node has to be connected to at least three other points for a unique solution to its position.

With the node configuration in place, we examine if we can incrementally obtain the position information of the nodes, starting with three nodes that can all hear each other. We perform an exhaustive search on the set of nodes to see if there are *some* three starting nodes that allow us to incrementally solve for the location information of all the nodes. In this sense, our results are an absolute best-case for an incremental scheme, because the existence of even one triplet that works properly is considered a success for the experiment.

Figure 15 shows the fraction of time we could find some three nodes for incremental localization as a function of node connectivity. As we can see, even for highly connected networks with an average connectivity of seven, the incremental localization method fails most of the time. This graph also shows the average fraction of the nodes that cannot be localized in each case. As we can see, the pure incremental scheme fail to localize a large fraction of nodes even under average connectivities as high as seven.

We run AFL on the same set of graphs. Figure 16 gives the results of this experiment. As we can see our algorithm performs much better even for graphs with small connectivity. This demonstrates our hypothesis.

5.3 AFL v. incremental scheme: Ranging error

Our third set of experiments explores a large parameter space, varying both ranging error as well as average node connectivity. The primary goal of these experiments is to evaluate whether our hypothesis, that AFL's concurrent fold-free approach is more robust to ranging error than the incremental approach. We find that the robustness to error at any given error rate depends on the node connectivity, so our results are three-dimensional graphs showing surfaces.

Each simulation in this set of experiments is on a 250-node graph in average and we ran 50 simulations to obtain each point on the graphs described below.

Figure 17 shows the performance of the AFL algorithm under different error ratios and different connectivities. The resulting GER values are very small. The GER represents the sum of the errors among individual points on the graph. Hence, a small GER value must correspond to small change to the overall structure of the graph.

Figure 18 shows the ratio of the GERs of the incremental scheme and AFL. The AFL algorithm clearly outperforms the incremental version; this ratio is always larger than 4, and is

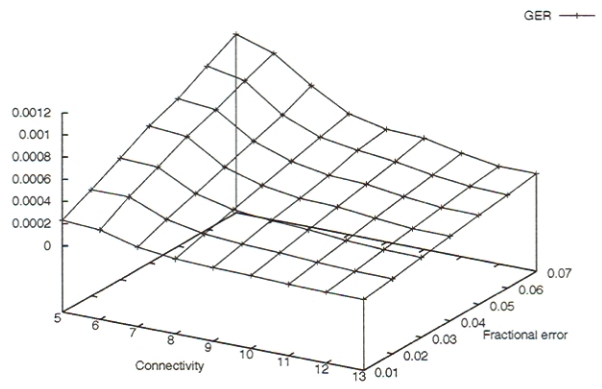

Figure 17: The value of global-error-ratio for AFL under different error and connectivities.

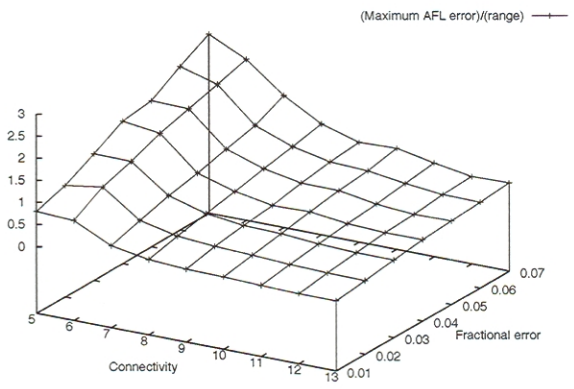

Figure 19: Maximum error between any two unconnected nodes as a fraction of the range.

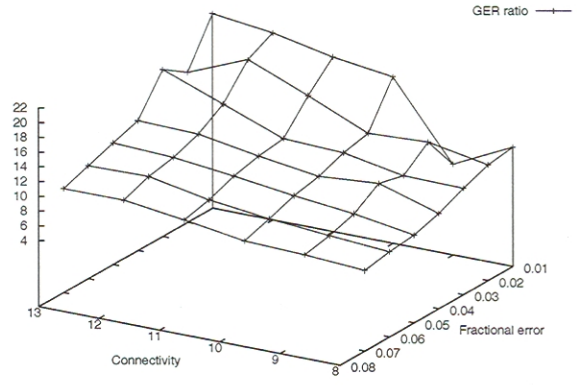

Figure 18: Ratio of GER of incremental scheme vs AFL.

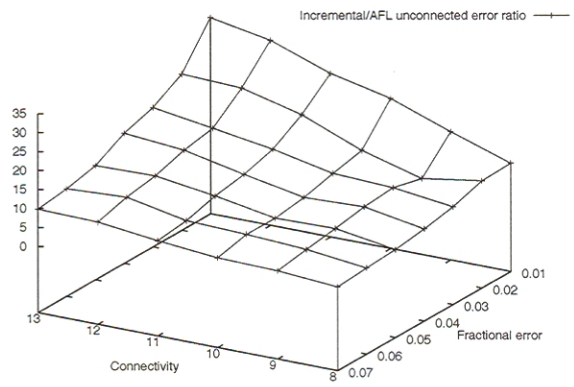

Figure 20: The ratio of the maximum errors between any two unconnected nodes in incremental and AFL respectively .

often larger by more than 10, an order of magnitude. The ratio increases with small increases in ranging error (which is never more than 1% in the experiments).

Figure 19 shows the *maximum* error between any two nodes after running the AFL algorithm. When the graph undergoes some physical deformation, this is identical to some points in the graph moving with respect to other points. Hence the maximum error between any two points corresponds to the maximum deformation the graph has undergone. Figure 19 shows the superior performance of AFL under ranging errors, since the maximum distance error between any two points is small most of the time. In most cases the absolute position error is smaller than the radio range, showing a degree of robustness to error that is significantly better than in previously published schemes.

Finally, Figure 20 shows the ratio of the maximum error between any two unconnected nodes in the incremental algorithm and AFL. As mentioned earlier, the maximum error between unconnected nodes is a good measure of the overall structural accuracy of the resulting graphs. Hence as far as total structural rigidity is concerned, AFL easily outperforms the incremental algorithm.

6 Conclusion

Many sensor network applications require that each node's sensor stream be annotated with its physical location in some common coordinate system. Manual measurement and configuration methods for obtaining location don't scale and are error-prone, and equipping sensors with GPS is often expensive and does not work in indoor and urban deployments. Sensor networks can therefore benefit from a self-configuring method where nodes cooperate with each other, estimate local distances to their neighbors, and converge to a consistent coordinate assignment.

This paper describes a fully decentralized anchor-free algorithm called AFL that solves this problem in many situations. In AFL, nodes start from a completely random initial coordinate assignment and converge to a consistent solution using only local node interactions. Nodes in AFL operate concurrently, rather than incrementally, and our simulation results show that this approach produces good coordinate assignments substantially more often than an incremental approach. For example, on random graphs based on RF connectivity, when the average node connectivity is to 7 or fewer neighbors, the incremental scheme almost never works whereas AFL does.

The key idea in AFL is *fold-freedom*, where nodes first configure into a topology that resembles a scaled and unfolded version of the true configuration, and then run a force-based relaxation procedure. Fold-freedom reduces the likelihood of lapsing into local minima by avoiding "folded" configurations, and is crucial to the ability of nodes in AFL to work concurrently. Our simulation results show that AFL is an order-of-magnitude better than incremental anchor-free approaches.

Several directions for future work present themselves. First, we would like to make precise claims about when AFL works and when it doesn't, by formally proving theorems about fold-free configurations. Second, we have started implementing AFL on a large location-aware beacon and sensor network, and look forward to evaluating its performance under real ranging and connectivity conditions. Third, we plan to compare AFL against anchor-based approaches insofar as position accuracy is concerned (even though they solve a different problem because they rely on the existence of anchors and require a large number of anchors for good performance). While a comparison against published simulation results of the other schemes shows AFL in good light, a direct comparison and analysis is needed.

We also believe that our method for obtaining fold-free configurations has applications beyond ranging, including in the design of self-configuring scalable routing systems for large wireless and sensor networks. This is because the polar (or equivalent Cartesian) coordinates resulting from the fold-free procedure forms a natural framework over which to run scalable geographic routing, without requiring any actual location system.

References

[1] BERGER, B., KLEINBERG, J., AND LEIGHTON, T. Reconstructing a three-dimensional model with arbitrary errors. In *Proceedings of the 28th ACM Symposium on Theory of Computing* (1996).

[2] BULUSU, N., HEIDEMANN, J., AND ESTRIN, D. GPS-less Low Cost Outdoor Localization For Very Small Devices. Tech. Rep. 00-729, Computer Science Department, University of Southern California, Apr. 2000.

[3] BULUSU, N., HEIDEMANN, J., ESTRIN, D., AND TRAN, T. Self-confi guring localization systems: Design and experimental evaluation. *ACM Transactions on Embedded Computing Systems* (May 2003). To appear.

[4] CONNELLY, R. On generic global rigidity. In *Applied Geometry and Discrete Mathematics*. American Mathematical Society, 1991, pp. 147–155.

[5] CONNELLY, R. Rigidity. In *Handbook of Convex Geometry*, vol. A. North-Holland, Amsterdam, 1993, pp. 223–271.

[6] CRIPPEN, G., AND HAVEL, T. *Distance Geometry and Molecular Conformation*. John Wiley & Sons, 1988.

[7] DOHERTY, L., PISTER, K., AND GHAOUI, L. Convex position estimation in wireless sensor networks. In *Proc. IEEE INFOCOM* (April 2001).

[8] FRUCHTERMAN, T., AND REINGOLD, E. Graph Drawing by Force-directed Placement. *Software - Practice and Experience (SPE) 21*, 11 (November 1991), 1129–1164.

[9] GRAVER, J., SERVATIUS, B., AND SERVATIUS, H. *Combinatorial Rigidity*. American Mathematical Society, 1993.

[10] GRAVER, J. E. *Counting on Frameworks: Mathematics to Aid the Design of Rigid Structures*. Mathematical Association of America, 2001.

[11] HARTER, A., HOPPER, A., STEGGLES, P., WARD, A., AND WEBSTER, P. The Anatomy of a Context-Aware Application. In *Proc. 5th ACM MOBICOM Conf.* (Seattle, WA, Aug. 1999).

[12] HENDRICKSON, B. Conditions for unique graph realizations. *SIAM Journal on Computing 21*, 1 (1992), 65–84.

[13] HOFFMANN-WELLENHOF, B., LICHTENEGGER, H., AND COLLINS, J. *Global Positioning System: Theory and Practice, Fourth Edition*. Springer-Verlag, 1997.

[14] HOWARD, A., MATARIC, M., AND SUKHATME, G. Relaxation on a mesh: A formalism for generalized localization. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (Wailea, Hawaii, Oct. 2001).

[15] INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Boston, MA, 2000), pp. 56–67.

[16] LI, J., JANNOTTI, J., DE COUTO, D., KARGER, D., AND MORRIS, R. A Scalable Location Service for Geographic Ad-Hoc Routing. In *Proc. 6th ACM MOBICOM Conf.* (Boston, MA, Aug. 2000).

[17] MADDEN, S., FRANKLIN, M., HELLERSTEIN, J., AND HONG, W. TAG: a Tiny AGregation Service for Ad-Hoc Sensor Networks. In *Proceedings of the Fifth USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (Boston, MA, December 2002).

[18] NAGPAL, R., SHROBE, H., AND BACHRACH, J. Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. In *Proc. International Workshop on Information Processing in Sensor Networks, IPSN 2003* (Palo Alto, CA, 2003).

[19] NICULESCU, D., AND NATH, B. Ad Hoc Positioning System (APS) Using AOA. In *Proc. of IEEE INFOCOM* (Salt Lake City, UT, April 2003), pp. 2037–2040.

[20] PRIYANTHA, N., CHAKRABORTY, A., AND BALAKRISHNAN, H. The Cricket Location-Support System. In *Proc. 6th ACM MOBICOM Conf.* (Boston, MA, Aug. 2000).

[21] SAVARESE, C., RABAEY, J., AND BEUTEL, J. Locationing in Distributed Ad-Hoc Wireless Sensor Networks. In *Proc. of ICASSP* (Salt Lake City, UT, May 2001), pp. 2037–2040.

[22] SAVARESE, C., RABAEY, J., AND LANGENDOEN, K. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In *USENIX Annual Technical Conference, General Track* (Monterey, CA, June 2002), pp. 317–327.

[23] SAVVIDES, A., HAN, C., AND SRIVASTAVA, M. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *Proc. 7th ACM MOBICOM Conf.* (July 2001), pp. 166–179.

[24] SAXE, J. B. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proceedings of the 17th Allerton Conference on Communications, Control, and Computing* (1979), pp. 480–489.

[25] YEMINI, Y. Some theoretical aspects of position-location problems. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science* (1979), pp. 1–8.