Localization from Connectivity in Sensor Networks

Yi Shang, *Member*, *IEEE*, Wheeler Ruml, Ying Zhang, *Member*, *IEEE*, and Markus Fromherz, *Member*, *IEEE*

Abstract—We propose an approach that uses connectivity information—who is within communications range of whom—to derive the locations of nodes in a network. The approach can take advantage of additional information, such as estimated distances between neighbors or known positions for certain anchor nodes, if it is available. It is based on multidimensional scaling (MDS), an efficient data analysis technique that takes $O(n^3)$ time for a network of n nodes. Unlike previous approaches, MDS takes full advantage of connectivity or distance information between nodes that have yet to be localized. Two methods are presented: a simple method that builds a global map using MDS and a more complicated one that builds small local maps and then patches them together to form a global map. Furthermore, least-squares optimization can be incorporated into the methods to further improve the solutions at the expense of additional computation. Through simulation studies on uniform as well as irregular networks, we show that the methods achieve more accurate solutions than previous methods, especially when there are few anchor nodes. They can even yield good relative maps when no anchor nodes are available.

Index Terms—Wireless sensor networks, optimization, position estimation.

1 Introduction

Large-scale networks with hundreds and even thousands of very small, battery-powered, and wirelessly connected sensor and actuator nodes are becoming a reality [3]. For example, future sensor networks will involve a very large number of nodes densely deployed over physical space. The nodes are typically highly resource-constrained (processor, memory, and power), have limited communication range, are prone to failure, and are put together in ad hoc networks.

Imagine a network of sensors sprinkled across a large building or an area such as a forest. Typical tasks for such networks are to send a message to a node at a given location (without knowing which node or nodes are there or how to get there), to retrieve sensor data (e.g., sound or temperature levels) from nodes in a given region, and to use the sensor nodes to track nearby events, such as vehicles moving through the sensor field. Most of these tasks require knowing the positions of the nodes or at least relative positions among them. For example, for a vehicle-tracking application, the sensor nodes would determine the positions of the tracked vehicles relative to their own positions.

With a network of thousands of nodes, it is unlikely that the position of each node can be precisely predetermined. While nodes could be equipped with a global positioning system (GPS) to provide them with their absolute position, this is currently a costly solution. Instead, for example, the sensor nodes might be dropped from an airplane in a rough grid pattern and would then have to determine their exact positions by putting themselves in relation to each other and possibly a few nearby beacons with known positions.

In this paper, we present an approach for computing the positions of nodes given only basic information that is likely to be already available, namely, which nodes are within communications range of which others. At the heart of the approach is multidimensional scaling (MDS), a data analysis technique that transforms proximity information into a geometric embedding. MDS is well-suited to node localization in communication networks, where the task is to use the distance information between nodes to determine the coordinates of nodes in a 2D or 3D space.

We present two methods based on this approach: a simple centralized one, called MDS-MAP(C), that builds a global map using classical MDS, and a more complicated one, MDS-MAP(P), that builds many small local maps and then patches them together to form a global map. MDS-MAP(C) has three steps: Starting with the given network connectivity information (or local distance measurements), we first compute the shortest-path distance between each pair of nodes. Then, we use classical MDS to derive node coordinates that fit those distances. Finally, we normalize the resulting coordinates to take into account any nodes whose positions are known.

MDS-MAP(C, R) is a variant of MDS-MAP(C) in which we add a refinement step to improve the solution computed by MDS. In the refinement, least-squares minimization is used to make the distances between neighboring nodes better match the provided measured distances. This optimization is more costly than classical MDS.

Like many existing methods, MDS-MAP(C) works well on networks with relatively uniform node density, but less well on more irregular networks, where the shortest path distance between two nodes does not correspond well to their Euclidean distance. To tackle this difficult problem, we present MDS-MAP(P). Its strategy is to build for each node a local map of the small subnetwork in the node's vicinity and then merge these local maps together to form a global

- *Y. Shang is with the Department of Computer Science, EBW 201, University of Missouri-Columbia, Columbia, MO 65211. E-mail: shangy@missouri.edu.*
- *W. Ruml, Y. Zhang, and M. Fromherz are with the Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304. E-mail: {ruml, yzhang, fromherz}@parc.com.*

Manuscript received 18 July 2003; revised 15 Jan. 2004; accepted 2 Mar. 2004. For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0116-0703.

map. This method avoids using shortest path distances between far away nodes and, thus, the smaller local maps constructed using local information are more accurate. Another advantage of the method is that it can be easily performed in a distributed fashion, which makes it appropriate for large-scale networks.

As with MDS-MAP(C), we can add a refinement step to MDS-MAP(P) to improve the global relative map. We call the method with refinement MDS-MAP(P, R).

As we will demonstrate, our approach often outperforms existing methods. Furthermore, it requires only connectivity information to produce a meaningful result. If the distances between neighboring nodes can be estimated, that information can be easily incorporated into the pairwise shortest-path computation during the first step of the algorithm. MDS yields coordinates that provide the best fit to the estimated pairwise distances, but which lie at an arbitrary rotation and translation. If the coordinates of any nodes are known, they can be used to derive the linear transformation of the MDS coordinates that allows the best match to the known positions. Usually, only three (or four) such "anchor nodes" are necessary to provide absolute positions for all the nodes in a 2D (or 3D) network.

The next section of the paper describes our approach in detail. Then, we will provide an overview of previous proposals before presenting our empirical evaluation.

2 Localization Using Multidimensional Scaling

We consider the node localization problem under two different scenarios. In the first, only proximity (or connectivity) information is available. Each node only knows what nodes are nearby, presumably by means of some local communication channel such as radio or sound, but not how far away these neighbors are or in what direction they lie. In the second scenario, the proximity information is enhanced by knowing the distances, perhaps with limited accuracy, between neighboring nodes.

In both cases, the network is represented as an undirected graph with vertices V and edges E. The vertices correspond to the nodes, of which there exist $m \geq 0$ special nodes with known positions, which we will call anchors. For the proximity-only case, the edges in the graph correspond to the connectivity information. For the case with known distances to neighbors, the edges are associated with values corresponding to the estimated distances. We assume that all the nodes being considered in the positioning problem form a connected graph.

Given a network graph of n nodes and estimated distances P between a subset of nodes (let p_{ij} represent the estimated distance between nodes i and j), the localization problem is to find the coordinates of the nodes $X = (X_1, X_2, \ldots, X_n)$ such that the Euclidean distances between the nodes, D, equal to P, i.e., $d_{ij} = p_{ij}$ for available p_{ij}, where $d_{ij} = ||X_i - X_j||_2$. When the estimates p_{ij} are just the connectivity or inaccurate local distance measurements, usually there is no exact solution to the overdetermined system of equations. Thus, the localization problem is often formulated as an optimization problem that minimizes the sum of squared errors. This optimization problem is generally nonconvex with many local minima. Traditional local optimization techniques, such as the Levenberg-Marquardt method, require good initial points in order to produce good solutions. Global search methods such as simulated annealing or genetic algorithms are generally too slow.

There are two possible outputs when solving the localization problem. One is a relative map and the other is an absolute map. The task of finding a relative map is to find an embedding of the nodes into either two or three-dimensional space that results in the same neighbor relationships as the underlying network. Such a relative map can provide correct and useful information even though it does not necessarily include accurate absolute coordinates for each node. Relative information may be all that is obtainable in situations in which powerful sensors or expensive infrastructure cannot be installed or when there are not enough anchors present to uniquely determine the absolute positions of the nodes. Furthermore, some applications only require relative positions of nodes, such as some direction-based routing algorithms [4], [5]. Sometimes, however, an absolute map is required. The task of finding an absolute map is to determine the absolute geographic coordinates of all the nodes. This is needed in applications such as geographic routing and target discovering and tracking [6], [7], [8], [9].

As we will show below, our method can generate either result, depending on the number of anchor nodes. The method first generates a relative map of the network and then transforms it to absolute positions if sufficient anchors are available. Before we describe the details of our method, we first introduce MDS, which is used in generating the relative map.

2.1 Multidimensional Scaling (MDS)

Imagine a small cloud of colored beads suspended in mid-air. To characterize the arrangement, one could measure the straight line distance between each pair of beads. If the cloud were shattered and the beads fell to the floor, one could imagine trying to recreate the arrangement based on the recorded interpoint distances. One would try to determine a location for each bead such that the distances in the new arrangement matched the desired distances. This recreation process is exactly the problem that MDS solves. Intuitively, it is clear that while the $O(n^2)$ distances will be more than enough to determine $O(n)$ coordinates, the result of MDS will be an arbitrarily rotated and flipped version of the true original layout because the interpoint distances make no reference to any absolute coordinates.

MDS has its origins in psychometrics and psychophysics. It can be seen as a set of data analysis techniques that display the structure of distance-like data as a geometrical picture [10]. MDS starts with one or more distance matrices (or similarity matrices) that are presumed to have been derived from points in a multidimensional space. It is usually used to find a placement of the points in a low-dimensional space, usually two or three-dimensional, where the distances between points resemble the original similarities. MDS is often used as part of exploratory data analysis or information visualization. By visualizing objects as points in a low-dimensional space, the complexity in the original data matrix can often be reduced while preserving the essential information. MDS is related to principal component analysis, factor analysis, and cluster analysis. MDS has been applied in many fields, such as machine learning [11] and computational chemistry [12].

There are many types of MDS techniques. They can be classified according to whether the similarity data is qualitative (nonmetric MDS) or quantitative (metric MDS).

They can also be classified according to the number of similarity matrices and the nature of the MDS model. Classical MDS uses one matrix. Replicated MDS uses several matrices, representing distances measurements taken from several subjects or under different conditions. Weighted MDS uses a distance model which assigns a different weight to each dimension. Finally, there is a distinction between deterministic and probabilistic MDS. In deterministic MDS, each object is represented as a single point in a multidimensional space, whereas, in probabilistic MDS, each object is represented as a probability distribution over the entire space.

We focus on classical metric MDS in this paper. Classical metric MDS is the simplest case of MDS: The data is quantitative and the proximities of objects are treated as distances in a Euclidean space [13]. The goal of metric MDS is to find a configuration of points in a multidimensional space such that the interpoint distances are related to the provided proximities by some transformation (e.g., a linear transformation). If the proximity data were measured without error in a Euclidean space, then classical metric MDS would exactly recreate the configuration of points. In practice, the technique tolerates error gracefully due to the overdetermined nature of the solution. This will be very helpful when we apply it to localization, as our distance estimates can be very rough indeed. Because classical metric MDS has an analytical solution, it can be performed efficiently on large matrices.

Let p_{ij} refer to the proximity measure between objects i and j. The Euclidean distance between two points $X_i = (x_{i1}, x_{i2}, \cdots, x_{im})$ and $X_j = (x_{j1}, x_{j2}, \cdots, x_{jm})$ in an m-dimensional space is

$$d_{ij} = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2}.$$

When the geometrical model of the objects fits the proximity data perfectly, the Euclidean distances are related to the proximities by a transformation $d_{ij} = f(p_{ij})$. In classical metric MDS, a linear transformation model is assumed, i.e., $d_{ij} = a + bp_{ij}$.

The distances D are determined so that they are as close to the proximities P as possible. There are a variety of ways to define "close." A common one is a least-squares definition, which is used by classical metric MDS. In this case, we define $I(P) = D + E$, where $I(P)$ is a linear transformation of the proximities and E is a matrix of errors (residuals). Since D is a function of the coordinates X, the goal of classical metric MDS is to calculate the X such that the sum of squares of E is minimized.

In classical metric MDS, the coordinates X can be computed from P through singular value decomposition (SVD) on the double centered squared P. Double centering a matrix is subtracting the row and column means of the matrix from its elements, adding the grand mean, and multiplying by -1/2. For an $n \times n$ P matrix for n points and m dimensions of each point, it can be shown that

$$-\frac{1}{2}\left(p_{ij}^2 - \frac{1}{n}\sum_{j=1}^{n} p_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} p_{ij}^2 + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^2\right) = \sum_{k=1}^{m} x_{ik}x_{jk}. \quad (1)$$

Let's call the double centered matrix on the left-hand side B. Performing singular value decomposition on B gives us $B = UVU'$ and coordinates $X = UV^{1/2}$.

B is an indefinite matrix. It may have negative as well as zero or positive roots. It can be shown that if we define V_r by selecting the r largest singular values and the corresponding singular vectors U_r and define $X_r = U_r V_r^{1/2}$, then $B_r = X_r X_r'$ is an optimal least squares approximation to B. In applying MDS to the node localization problem, the estimated distances between nodes form the P matrix. To compute the coordinates of nodes from P, we take the two largest singular and singular vectors of B for 2D networks and take the three largest singular and singular vectors of B for 3D networks. There is little overhead for computing the coordinates in 3D space as compared to 2D space, which is a nice property that previous triangulation-based localization methods do not have.

In nonmetric (also called ordinal) MDS [14], the goal is to establish a monotonic relationship between interpoint distances and the desired distances. Instead of trying to directly match the given distances, one is satisfied if the distances between the points in the solution fall in the same ranked order as the corresponding distances in the input matrix. The advantage of nonmetric MDS is that no assumptions need to be made about the underlying transformation function. The only assumption is that the data is measured at the ordinal level. Just as classical MDS, nonmetric MDS can also be applied to the localization problem. By adopting a more flexible model, the effects of a few highly incorrect measurements might be more easily tolerated.

2.2 MDS-MAP(C) and MDS-MAP(C, R)

Based on MDS, we have developed several localization methods, called MDS-MAP methods. The simplest MDS-MAP method, MDS-MAP(C), builds a global map using a single application of classical MDS. The parameter C refers to centralized as the connectivity information of the network is sent to a central location where the computation is carried out. The method with additional refinement to MDS-MAP(C) is called MDS-MAP(C, R), where the parameter R is for refinement.

MDS-MAP(C) consists of three steps:

1. Compute the shortest paths between all pairs of nodes in the region of consideration. The shortest path distances are used to construct the distance matrix for MDS.
2. Apply MDS to the distance matrix, retaining the first two (or three) largest eigenvalues and eigenvectors to construct a 2D (or 3D) relative map.
3. Given sufficient anchor nodes (three or more for 2D, four or more for 3D), transform the relative map to an absolute map based on the absolute positions of anchors.

In Step 1, we first assign distances to the edges in the connectivity graph. When the distance of a pair of neighbor nodes is known, the value of the corresponding edge is the measured distance. When we only have connectivity information, a simple approximation is to assign value 1 to all edges. Then, an all-pairs shortest-path algorithm, such as Dijkstra's or Floyd's, can be applied. The time complexity is $O(n^3)$, where n is the number of nodes.

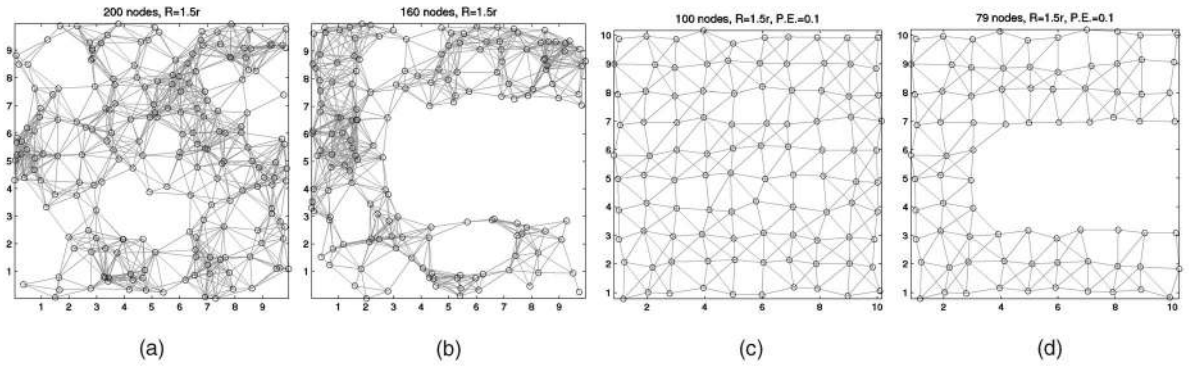

Fig. 1. Four example problems: (a) random uniform placement—200 nodes are randomly placed in a $10r \times 10r$ square. (b) Random irregular placement—160 nodes are randomly placed in an area of C shape within a $10r \times 10r$ square. (c) Regular uniform placement—100 nodes are placed on a grid with $10\%r$ placement errors. (d) Regular irregular placement—79 nodes are placed on a C shape grid with $10\%r$ placement errors. The radio range is $1.5r$, where the placement unit length $r = 1$. The average connectivity levels of the four problems are 12.1, 11.5, 6.0, and 5.1, respectively.

In Step 2, classical MDS is applied directly to the distance matrix. The core of classical MDS is singular value decomposition, which has complexity of $O(n^3)$. The result of MDS is a relative map that gives a location for each node. Although these locations may be accurate relative to one another, the entire map will be arbitrarily rotated and flipped relative to the true node positions.

In Step 3, the relative map is transformed through a linear transformation, which may include scaling, rotation, and reflection. The goal is to minimize the sum of the squares of the errors between the true positions of the anchors and their transformed positions in the MDS map. Computing the transformation parameters takes $O(m^3)$ time, where m is the number of anchors. Applying the transformation to the whole relative map takes $O(n)$ time.

In MDS-MAP(C, R), a refinement step is added between Steps 2 and 3 of MDS-MAP(C) to improve the relative map:

- Using the position estimates of nodes in the MDS solution as an initial solution, apply least-squares minimization to improve the match between the measured distances between neighboring nodes and their distances in the solution.

Our formation of the refinement is more general than previous methods [15], [16] in two ways: 1) In addition to the information between 1-hop neighbors, information between multihop neighbors is also used, but with different weights. 2) Instead of refining the coordinates of one node at a time while all other nodes remain fixed, the coordinates of all nodes in the relative map are variables in a single optimization. We use a refinement range R_{ref}, defined based on hops, to specify how much information is considered. $R_{ref} = 1$ means only information between 1-hop neighbors are used, $R_{ref} = 2$ means information of nodes within two hops is used, and so on. Different values of R_{ref} offer a trade off between computational cost and solution quality.

An important advantage of our approach is that MDS can provide better starting points for the least-squares minimization than other triangulation-based or heuristic methods [15]. The least-squares minimization problem is high-dimensional and has lots of local minima. Random starting points usually lead to very bad solutions. MDS is good at finding the right general topology of a network, which corresponds to a starting point in the basin of attraction of an optimal or near-optimal solution.

More formally, let $(x_i, y_i), i = 1, \cdots, N$ represent the coordinates of the N nodes in a 2D local map, d_{ij} be the Euclidean distance between two nodes i and j in a candidate solution, and p_{ij} be the measured proximity of nodes i and j. When only connectivity information is available, $p_{ij} = 1$ if i and j are 1-hop neighbors. When distance measurements between 1-hop neighbors are available, p_{ij} is the distance between i and j if they are 1-hop neighbors or the shortest path distance if i and j are further apart. The objective of the refinement step is

$$\min_{x_k, y_k} \sum_{i,j} w_{ij}(d_{ij} - p_{ij})^2, \qquad \text{for } k = 1, \cdots N, \tag{2}$$

where w_{ij} is the weight. In the experiments reported below, we set R_{ref} to 2; also, $w_{ij} = 1$ when i and j are one hop apart and $w_{ij} = 1/4$ when they are two hops apart.

For a 2D n-node network, the problem has $2n$ variables and no constraints. The Jacobian can be computed analytically. In our experiments, we use the Levenberg-Marquardt method (lsqnonlin in Matlab's optimization toolbox) to solve the problem. Usually, only the first few iterations of lsqnonlin give significant improvement. Thus, the maximum number of iteration is set to a small number, such as 20. Although this local optimization algorithm is fast, it is considerably slower than classical MDS. For 100-node networks, it is more than an order of magnitude slower. For larger networks, the time difference becomes larger.

We will use four example problems to illustrate the results of various MDS-MAP methods. Two have uniform topologies and the other two have irregular topologies. They are shown in Fig. 1. In the graphs, circles represent sensor nodes and edges represent connections between nodes that are within communication range of each other.

Fig. 2 shows the results of MDS-MAP(C) and MDS-MAP(C, R) using only connectivity information on the random uniform example. Both relative maps and absolute maps are shown for the two algorithms. Four random anchor nodes, denoted by asterisks, are used to estimate the transformation to absolute coordinates. (Note that the chosen nodes represent a rather unlucky selection as they are almost colinear.) The circles represent the true locations of the nodes and the solid lines represent the errors of the estimated positions from the true positions. The longer the line, the larger the error is. The average errors of MDS-MAP(C) and

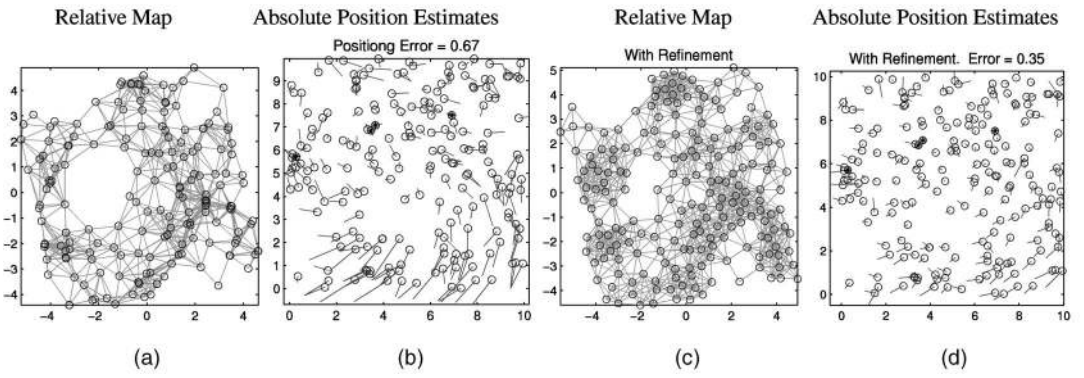

Fig. 2. Results of MDS-MAP(C) ((a) and (b)) and MDS-MAP(C, R) ((c) and (d)) on the example of random uniform placement using connectivity information only.

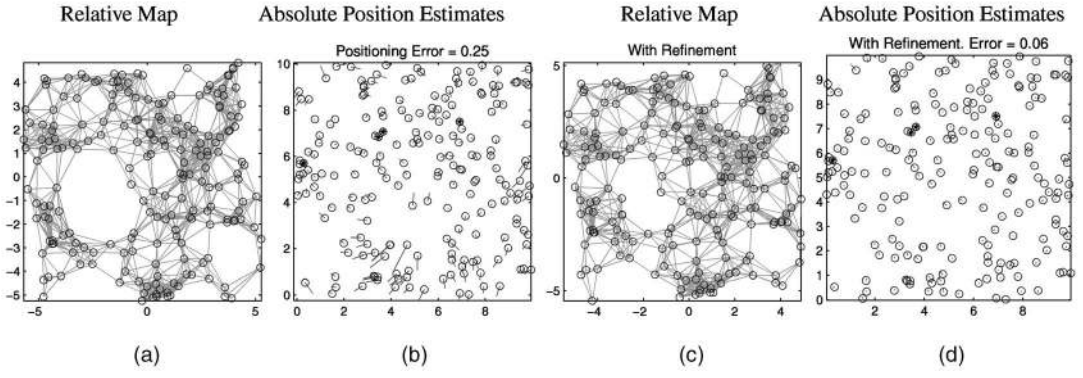

Fig. 3. Results of MDS-MAP(C) ((a) and (b)) and MDS-MAP(C, R) ((c) and (d)) on the example of random uniform placement using distances between neighboring nodes (5 percent distance measurement errors).

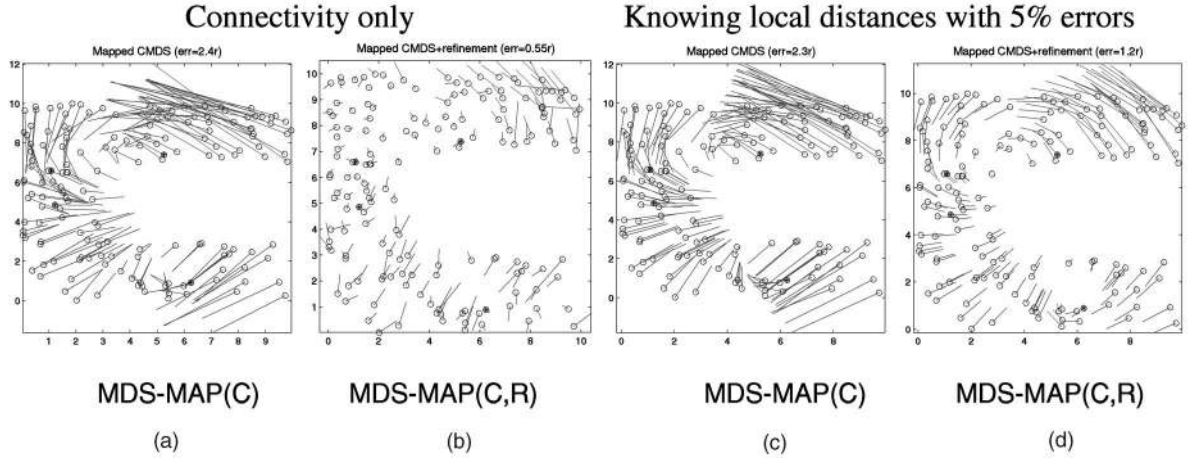

Fig. 4. Position estimation errors of MDS-MAP(C) and MDS-MAP(C, R) on the example of random irregular placement. (a) and (b) show results using connectivity information only, whereas (c) and (d) show results using distance measures between neighboring nodes with 5 percent distance errors.

MDS-MAP(C, R) are $0.67r$ and $0.35r$, respectively, where the field in which the nodes are places measures $10r$ by $10r$.

When distances between one-hop neighbors are known, the result of MDS-MAP(C) can be improved. Fig. 3 shows results on the same network, but when distances between one-hop neighbors are known with 5 percent measurement error. The estimates of MDS-MAP(C) based on the same four anchor nodes have an average error of $0.25r$, much better than the result when using connectivity only ($0.67r$). The result after refinement in MDS-MAP(C, R) is excellent. The average error is reduced to $0.06r$. r is the placement unit length and is set to 1 in the experiments.

Irregular topologies are much harder than uniform topologies. Fig. 4 shows the results on the random irregular example. Again, there are four random anchor nodes. The result of MDS-MAP(C) is poor. Although the result of MDS-MAP(C, R) is better than that of MDS-MAP(C), it is much worse than its result on the uniform example. MDS-MAP(C) does not work well because the shortest-path distance between two nodes in different wings of the network is much larger than their actual Euclidean distance. The error of MDS-MAP(C) using connectivity information is very large, $2.4r$. The refinement in MDS-MAP(C, R) is useful and reduces the error to $0.55r$.

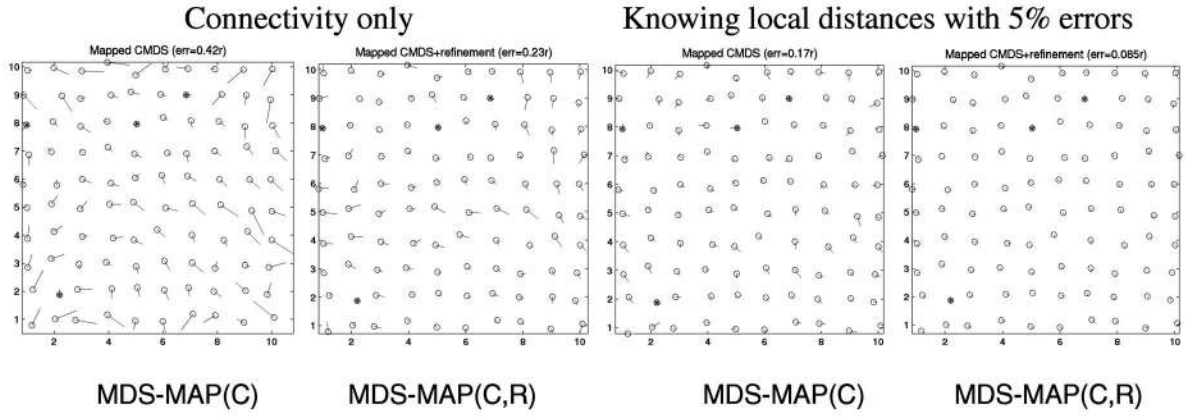

Fig. 5. Position estimation errors of MDS-MAP(C) and MDS-MAP(C, R) on the example of uniform grid placement.

The distances between one-hop neighbors are not helpful to the basic algorithm in the irregular case. Fig. 4 shows that, even when this information is provided, MDS-MAP(C) finds a poor solution with error $2.3r$, with MDS-MAP(C, R) improving the solution only slightly, reducing the error to $1.2r$, which is still large.

When the network has a relatively regular topology, such as nodes placed near grid points, MDS-MAP(C) and MDS-MAP(C, R) usually obtain good solutions. Fig. 5 shows the results on the uniform grid example. Again, there are four random anchor nodes. Comparing this grid example with the random example (Fig. 2), MDS-MAP(C) and MDS-MAP(C, R) obtain better solutions with lower connectivity levels. The connectivity levels of the two examples are 6.0 versus 12.1. Using connectivity information only, the errors of MDS-MAP(C) for the grid example versus the random example are $0.42r$ versus $0.67r$, whereas the errors of MDS-MAP(C, R) are $0.23r$ versus $0.35r$. Using local distances, their errors on the grid example are much smaller, $0.17r$ and $0.085r$, respectively.

It is a different story for the irregular grid example. As Fig. 6 shows, basic MDS-MAP(C) performs poorly, almost as poorly as on the random irregular placement example (Figs. 3 and 4). MDS-MAP(C) suffers from long-range distance estimation errors. In contrast, MDS-MAP(C, R) performs much better on the irregular grid example, especially when local distances are known. This is because the general topology of the solution obtained by MDS is accurate. Even though the distances between nodes are way off, by starting from the right topology, the refinement can successfully make the distances match the measured distances and thus generate a good relative map.

2.3 MDS-MAP Based on Patches of Local Maps: MDS-MAP(P) and MDS-MAP(P, R)

MDS-MAP(C) and MDS-MAP(C, R) do not work well on irregular networks because they rely on shortest-path distance estimation, which can have large errors for remote nodes. Another problem with these centralized methods is that they are not applied easily to large networks for which reading out the connectivity and distance information is potentially prohibitive. In such cases, in-network computation of coordinates would be much more attractive. MDS-MAP(P) addresses both of these problems.

MDS-MAP(P) is more complicated than MDS-MAP(C). It builds many local maps and then patches them together to form a global map. This method relies on local information and avoids using the distance estimation between remote notes. As we will show, it achieves better results on irregular networks. Another benefit of MDS-MAP(P) is that it can be easily executed in a distributed fashion. When we add refinement to improve the global map, we call the method MDS-MAP(P, R).

In MDS-MAP(P), individual nodes simultaneously compute their own local maps using their local information. Then, these maps can be incrementally merged to form a global map. The steps of MDS-MAP(P) are as follows:

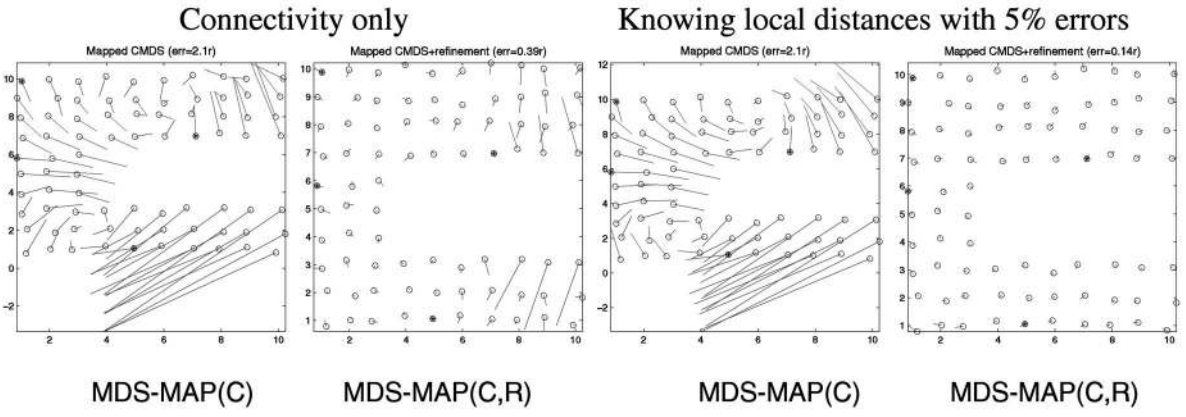

Fig. 6. Position estimation errors of MDS-MAP(C) and MDS-MAP(C, R) on the example of irregular grid placement.

TABLE 1
The Typical Time in Seconds Taken by the Major Steps of MDS-MAP(P, R) for Different Size Networks

Network Size	Computing local maps (summed over all nodes)		Merging all local maps	Refining global map
	classical MDS	Refinement		
50 nodes	0.08	4.74	0.59	0.49
100 nodes	0.25	18.1	1.68	2.9
200 nodes	0.49	36.3	5.0	19.3
300 nodes	0.91	55.79	12.4	84.6

1. Set the range for local maps, R_{lm}. For each node, neighbors within R_{lm} hops are involved in building its local map. We use $R_{lm} = 2$ in our experiments.
2. For each node, apply MDS-MAP(C, R) to the nodes within range R_{lm} to generate its local map.
3. Merge local maps. Local maps can be merged in various ways. We use a simple strategy: First, randomly pick a node and start with its local map; then, merge in the maps of neighboring nodes one by one. Each time, we choose the neighbor to merge whose local map shares the most nodes with the current map. Thus, the initial local map grows by incorporating other local maps and can eventually cover the entire network.
4. Given sufficient anchor nodes (three or more for 2D, four or more for 3D), transform the relative map to an absolute map based on the absolute positions of anchors.

Two maps are merged together based on the coordinates of their common nodes. The best linear transformation (minimizing discrepancy errors) is computed to transform the coordinates of the common nodes in one map to those in the other map. Given the coordinates of common nodes in maps A and B as matrices X_A and X_B, a linear transformation (translation, reflection, orthogonal rotation, and scaling) of X_B to best conform to X_A is determined. The "goodness-of-fit" criterion is the sum of squared errors, i.e., $\min_T ||T(X_B) - X_A||_2$, where $T(\cdot)$ is the linear transformation.

This method allows for parallel and distributed implementations in several ways. First, the computation of local maps can be done locally at each node in parallel with the others. Second, the local maps can be merged in parallel in different parts of the network. Because the method does not require anchor nodes in order to build a relative map of a subnetwork, it can be applied to many subnetworks in parallel. Third, the computation of absolute maps from anchor nodes could be applied to relative local maps and, thus, also be distributed in the network. For example, as soon as three or more anchors are present in a subnetwork, an absolute map could be computed. Furthermore, all local maps bordering on this absolute map could be absorbed in parallel into that map using the merger step. For large networks and a sufficient number of anchor nodes, it should never be necessary to compute a single global map anywhere. Distributed map merging has a number of benefits, including more balanced computation and communication among the nodes, faster construction of the global map, and distribution of map information in the network at multiple levels of granularity, giving the opportunity for better flexibility and robustness.

The amount of error generated when two maps are merged depends on several factors, including the accuracy of the two maps and the number of common nodes. The error will propagate when a linear sequence of maps is merged. In dense networks, the adjacent local maps usually have many common nodes and, thus, the error introduced in merging is small.

In MDS-MAP(P, R), a refinement step (as in MDS-MAP(C, R)) is added between Steps 3 and 4 of MDS-MAP(P) to improve the global relative map.

Table 1 shows the typical time taken by the major steps of MDS-MAP(P, R) in our prototype implementation. The program was run in Matlab 6.5 on a Dell Latitude C640 with a 2GHz Mobile Pentium 4M and 512MB RAM. All networks have connectivity 10. The data shows that MDS is very fast and the least-squares minimization of local maps is more than an order of magnitude slower. (This may be due in part to our use of the lsqnonlin function in Matlab, which calculates many terms that will have weight zero in the objective function.) The cost of merging local maps grows faster than linear due to the larger maps being manipulated. The cost of refining the global map grows the fastest and becomes dominant for large networks.

Again, using the four example problems from Fig. 1, we illustrate the performance of MDS-MAP(P) and MDS-MAP(P, R). Fig. 7 shows the results of MDS-MAP(P) and MDS-MAP(P, R) on the random uniform placement example. Using connectivity information only, the average error of MDS-MAP(P) is $0.40r$, about 60 percent of the error of MDS-MAP(C) in Fig. 2, and slightly worse than MDS-MAP(C, R). After refinement, the error of MDS-MAP(P, R) is $0.31r$, better than that of MDS-MAP(C, R). Using local distances, MDS-MAP(P) and MDS-MAP(P, R) obtain much better results. The error of MDS-MAP(P) is $0.16r$, better than the $0.25r$ error of MDS-MAP(C) in Fig. 3. After refinement, the error of MDS-MAP(P, R) is $0.06r$, at the level of the distance estimation errors.

Fig. 8 shows results on the random irregular placement example. MDS-MAP(P) returns a solution (error $1.2r$) better than MDS-MAP(C) (error $2.4r$), but the error is still very large. The refinement in MDS-MAP(P, R) helps to reduce the error to $0.43r$, better than the solution of MDS-MAP(C, R) (error $0.55r$). Using local distances, the solution of MDS-MAP(P) (error $0.72r$) is quite reasonable. The solution of MDS-MAP(P, R) is even better (error $0.29r$).

For networks with regular topologies, MDS-MAP(P) and MDS-MAP(P, R) usually obtain very good results. Figs. 9 and 10 show their results on the uniform and irregular grid examples with four random anchors. The results on the uniform network are excellent. The results on the irregular example are also very good.

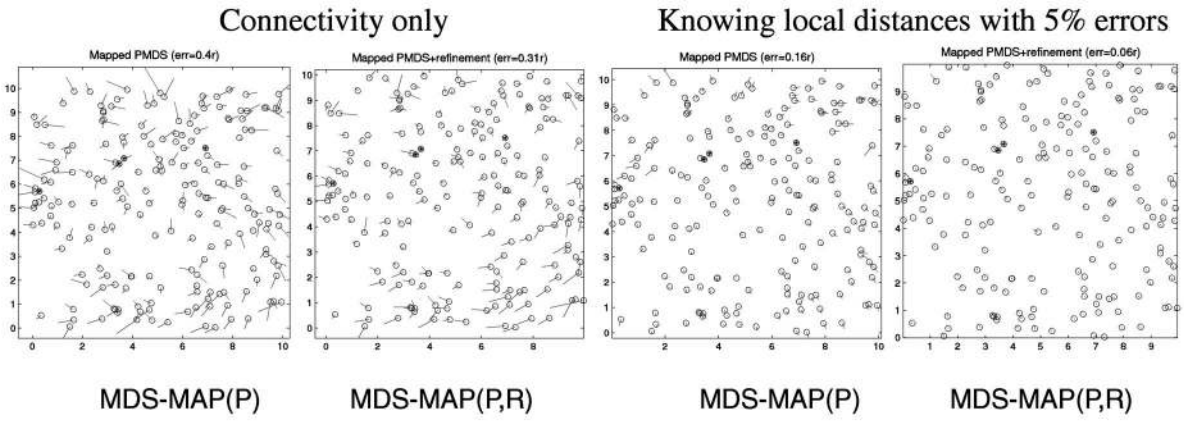

Fig. 7. Position estimation errors of MDS-MAP(P) and MDS-MAP(P, R) on the example of random uniform placement.

3 Related Work

Node localization has been a topic of active research in recent years. A detailed survey of the area is provided by Hightower and Borriello [17]. Many systems use some kind of range or distance information and rely on powerful beacon nodes with unusual capabilities, such as radio, laser ranging devices, or directional signals. Distance estimates can also be obtained from received signal strength indications (RSSI) or time-of-arrival (ToA) measurements. Due to nonuniform signal propagation, RSSI methods are not very reliable and accurate. ToA methods have better accuracy, but may require additional hardware at the nodes to receive a signal that has a smaller propagation speed than radio, such as ultrasound [18], [19]. Localization techniques for mobile robots often use additional odometric measurements [20], [21] which are not available on nodes in many networks.

Among existing localization methods for communication networks, several perform localization from connectivity information only. The GPS-less system by Bulusu et al. [22] employs a grid of beacon nodes with known positions. Each unknown node sets its position to the centroid of the beacons near the unknown. The position accuracy is about one-third of the separation distance between beacons. The method needs a high beacon density to work well.

Doherty et al.'s [23] convex constraint satisfaction approach localizes nodes based only on connectivity. The method formulates the localization problem with uniform communication as a feasibility problem with convex radial constraints. The method requires centralized computation. For the method to work well, it needs anchor nodes to be placed on the outer boundary, preferably at the corners, to make the constraints tight enough. When all anchors are located in the interior of the network, the position estimation of outer nodes can easily collapse toward the center, which leads to large estimation errors.

Several distributed localization methods are developed based on triangulation or multilateration. The "DV-hop" approach by Niculescu and Nath [24] is simple and efficient and among the best of triangulation-based methods. First, the anchors flood their location to all nodes in the network. Then, each unknown node performs a triangulation to three or more anchors to estimate its position. The method works well in dense and uniform topologies and poorly for sparse and irregular networks. The "DV-distance" method uses distance between neighboring nodes to reduces the location errors, but still works poorly on irregular networks.

Savarese et al. propose another distributed method [16] that consists of two phases: start-up and refinement. For the start-up phase, they use Hop-TERRAIN, an algorithm similar to DV-hop. Hop-TERRAIN is run once at the beginning to generate a rough initial estimate of the nodes' locations. Then, the refinement algorithm is run iteratively to improve and refine the position estimates. The algorithm is concerned only with nodes within a one-hop neighborhood and uses a least-squares triangulation method to determine a node's position based on its neighbors' positions and distances to them. The

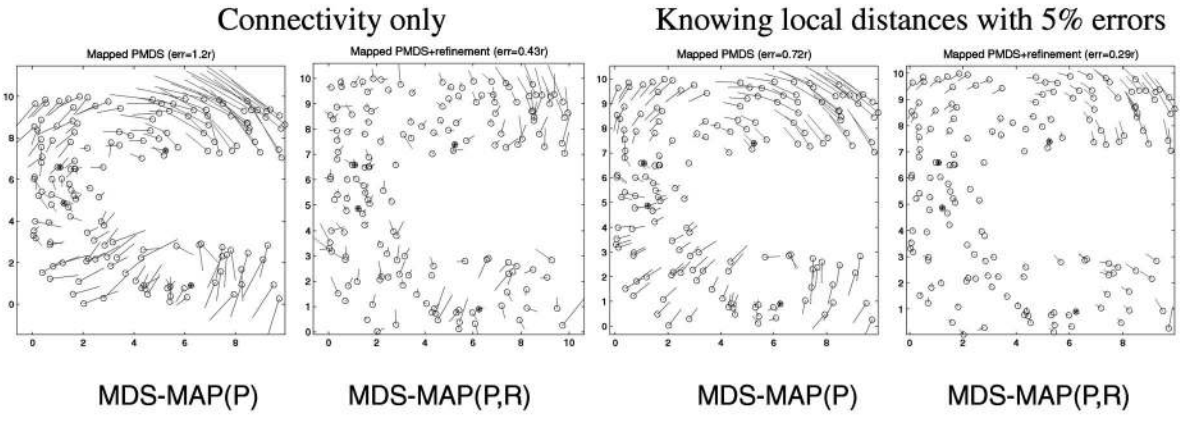

Fig. 8. Results of MDS-MAP(P) and MDS-MAP(P, R) on the random irregular example.

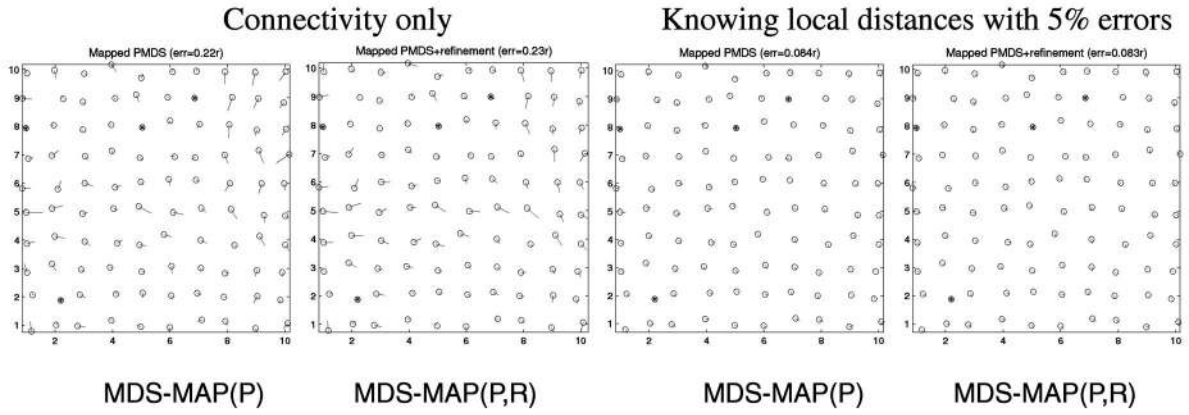

Fig. 9. Results of MDS-MAP(P) and MDS-MAP(P, R) on the uniform grid example.

Hop-TERRAIN phase is slightly worse than the DV-hop or DV-distance method. The refinement improves accuracy significantly, but reduces coverage since it only works for well-connected nodes.

Savvides et al. propose a collaborative multilateration method [15]. The method estimates node locations by using anchor locations that are several hops away and distances to neighboring nodes. The method has three main phases: 1) formation of a collaborative subtree, which only includes nodes whose positions can be uniquely determined, 2) computation of initial position estimates with respect to anchor nodes, and 3) position refinement by minimizing the residual between the measured distances between the nodes and the distances computed using the node location estimates. The method needs more anchors than the other methods to work well.

A key advantage of our approach over previous methods is the utilization of all connectivity or distance information among many nodes simultaneously. Most previous methods based on triangulation localize one unknown node at a time from the anchors and the information between nodes of unknown position is not utilized fully. Using all the information together, MDS generates correct topologies and better starting points for subsequent refinement techniques than previous methods. Without anchors, our approach can generate good relative maps using connectivity information only, an important capability not possessed by previous methods.

4 Experimental Results

In these experiments, we assess the average-case performance of MDS-MAP methods. For each of several different types of network, the algorithms are run on 30 randomly generated examples. The same types of networks are used as in Section 2 (Fig. 1): 1) uniform random, 200 nodes randomly placed inside a $10r \times 10r$ square, where $r = 1$ is the placement unit length, 2) irregular random, 160 nodes randomly placed inside an C-shaped area within a $10r \times 10r$ square; 1) uniform grid, 100 nodes placed according to a $10r \times 10r$ grid, and 2) irregular grid, 79 nodes placed according to a C-shaped grid within a $10r \times 10r$ square.

To model the errors in grid placements, we add Gaussian noise to the coordinates of nodes. For a $10\%r$ placement error, a random variable from $N(0, 10\%r)$ is added to each coordinate of a grid point. Thus, the nodes are not placed exactly on the grid points. The distance measure is modeled as the true distance blurred with Gaussian noise. Assume the true distance is d^* and range error is e_r; then, the measured distance is a random value drawing from a normal distribution $d^*(1 + N(0, e_r))$.

The connectivity (average number of neighbors) is controlled by specifying radio range R. The errors of position estimates are normalized to R (i.e., 50 percent position error means half of the range of the radio). We do not consider models of nonuniform radio propagation or widely varying ranging errors. Both modeling these

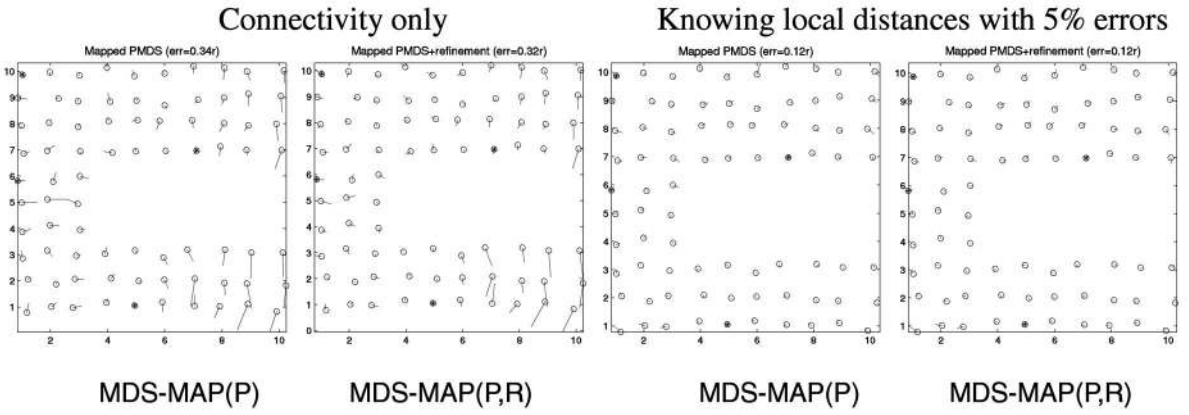

Fig. 10. Results of MDS-MAP(P) and MDS-MAP(P, R) on the irregular grid example.

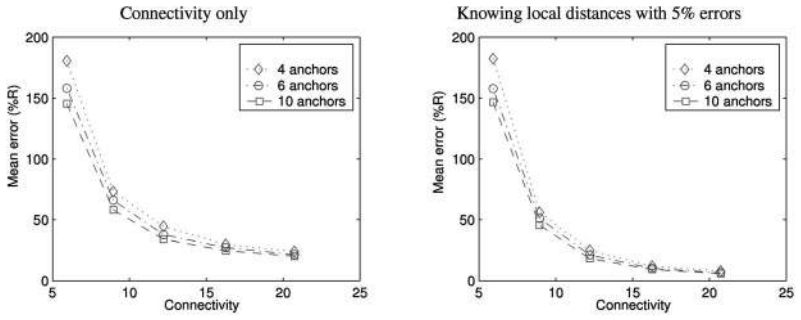

Fig. 11. Results of MDS-MAP(C) for 200-node random uniform networks.

phenomena and simulating their effects are very important directions for future work.

4.1 Random Placement

4.1.1 Uniform Networks

Fig. 11 shows the performance of MDS-MAP(C) as a function of connectivity and number of anchors, using 4, 6, or 10 random anchors. Position estimates by MDS-MAP(C) have an average error under $100\%R$ in scenarios with just four anchor nodes and an average connectivity level of 8.9 or greater. On the other hand, when the connectivity is low, e.g., 5.9, the errors can be large. Having good estimates of the distances between neighbors leads to much better solutions when the connectivity is high. When the connectivity level is 12.2 or greater, the errors are about half of those by MDS-MAP using proximity information only. On the other hand, when the connectivity is low, e.g., 5.9, knowing the local distance does not help much.

Fig. 12 shows the number of nodes participating in the location estimation in MDS-MAP(C). Recall that the largest connected subnetwork is extracted for processing. When the connectivity level is low, such as 5.9, about 7 percent of the nodes are not connected to the main subnetwork and their positions are not estimated. Among the nodes that are part of the main subnetwork, many of them have only one or two connections to their neighbors. The lack of sufficient information to determine the position of a node causes large errors in MDS-MAP(C) solutions. When the connectivity is over 12.2, the network is fully connected and all nodes are processed. Fig. 12 also shows the effects of increasing range errors on the estimation errors of MDS-MAP(C). Four random anchors are used. The range errors vary from 0 to 50 percent. Estimation error increases steadily as the range error increases. The errors with a larger connectivity (20.7) are more than $10\%R$ lower than those with a smaller connectivity (12.2) in most cases. The estimation error goes up faster after the range error is more than 30 percent. The large increase in error at range error 40 percent is due to an outlier that is very bad.

Next, we compare the performance of the four MDS-MAP methods. Fig. 13 shows their results on the random uniform networks with 200 nodes. The errors are plotted against the average connectivity level. The radio range (R) goes from $1.25r$ to $2.5r$, in increments of $0.25r$, which leads to average connectivity levels of 8.9, 12.2, 16.4, 20.9, 25.9, and 31.1. Three, four, six, or 10 random anchors are used.

When using only connectivity information, MDS-MAP(P) is consistently better than MDS-MAP(C), more than $10\%R$ better when the connectivity is low. MDS-MAP(C, R) and MDS-MAP(P, R) have comparable results and are better than MDS-MAP(P). Although more anchors lead to better results, the improvement with more than six anchors is small.

Using connectivity information only, MDS-MAP algorithms are much better than the convex optimization approach in [23] when the number of anchor nodes is low. For example, with 4 to 10 anchors in a 200-node random network, the convex optimization approach has an average estimation error of more than twice the radio range when the connectivity is 8.9 and above. The results are also better than Hop-TERRAIN [16], especially when the number of anchors is small. For example, with four anchors (2 percent of the network) and a connectivity level 12.2, MDS-MAP(P) using connectivity information only has an average error of about $27\%R$, whereas Hop-TERRAIN has an average error of about $90\%R$.

Using local distances with 5 percent error improves the performance of the MDS-MAP algorithms. Their errors are about half of those obtained using only proximity information. MDS-MAP(P) is comparable to MDS-MAP(C, R) and

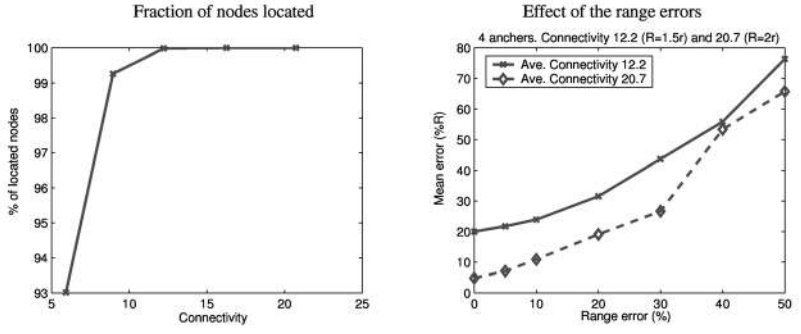

Fig. 12. The coverage of MDS-MAP(C) and the effect of the range error on the estimation error for 200-node random uniform networks.

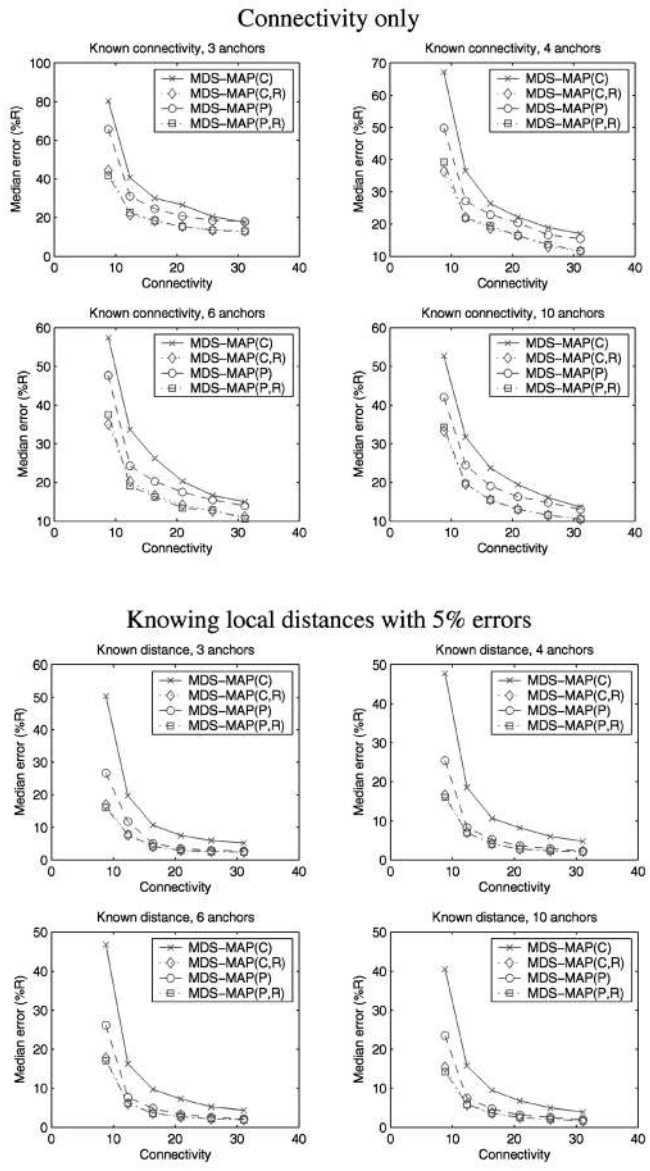

Fig. 13. Results of MDS-MAP methods on the random uniform networks and 3 to 10 anchors.

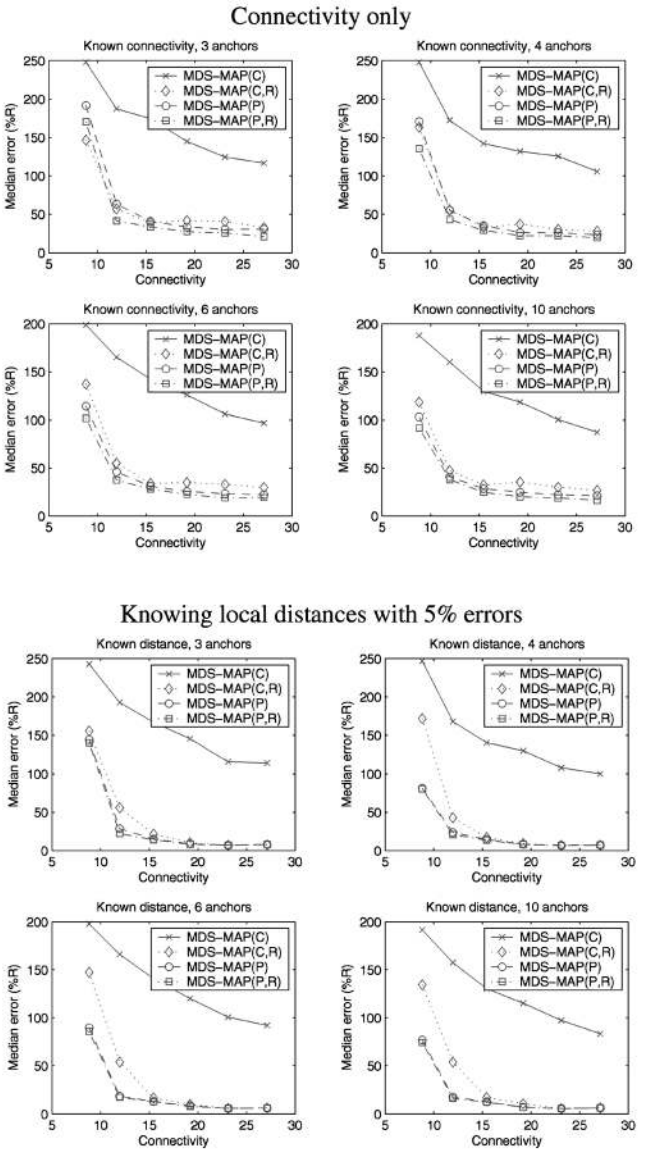

Fig. 14. Results of MDS-MAP methods on the random irregular networks and 3 to 10 anchors.

MDS-MAP(P, R) when the connectivity level is 12.2 and above.

MDS-MAP succeeds in localizing a higher fraction of the nodes in a network than most previous methods. MDS-MAP localizes all nodes in a connected network. The methods are not subject to constraints such as requiring that a node have at least three neighbors as in other methods. In our experiments, when the connectivity is 12.2 or more, the network is usually a connected graph and all nodes are located.

4.1.2 Irregular Networks

Irregular topologies are much harder than uniform topologies and previous methods reported very poor results on them [24]. Fig. 14 shows the performance of the MDS-MAP algorithms on the C-shaped random irregular networks with 160 nodes. The radio ranges (R) are the same as before, from $1.25r$ to $2.5r$, in increments of $0.25r$, which leads to average connectivity levels of 8.8, 12.0, 15.4, 19.2, 23.1, and 27.1.

On the C-shaped random irregular networks, MDS-MAP(C) returns bad solutions because the distance estimations for nodes in separate wings of the network are very different from their actual Euclidean distances.

The refinement in MDS-MAP(C, R) improves the solutions significantly. The result can be misleading because it seems that the refinement is the most important and does all the work. This is not the case. From a random starting point, the refinement usually does not do much and just returns a bad solution, because there are many local minima. Thanks to MDS, the relative map often has the right topology, which corresponds to a good starting point in the same basin as the optimal or near-optimal solution. This is why the refinement performs so well.

The results show that MDS-MAP(P) performs very well on these irregular networks, especially when the connectivity level is 12.0 or more, finding solutions just slightly worse than those returned by MDS-MAP(C, R) and MDS-MAP(P, R). The results of MDS-MAP(P, R) are slightly better than those of MDS-MAP(C, R).

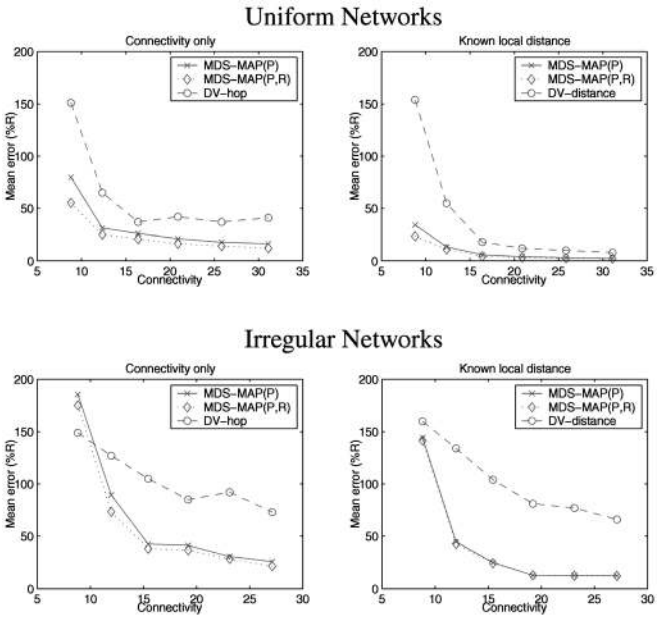

Fig. 15. Comparison of MDS-MAP(P), MDS-MAP(P, R), DV-hop, and DV-distance on random uniform (upper) and irregular networks (lower), using connectivity information (left) or local distances between 1-hop neighbors with 5 percent errors (right). Four anchors were randomly selected for each case.

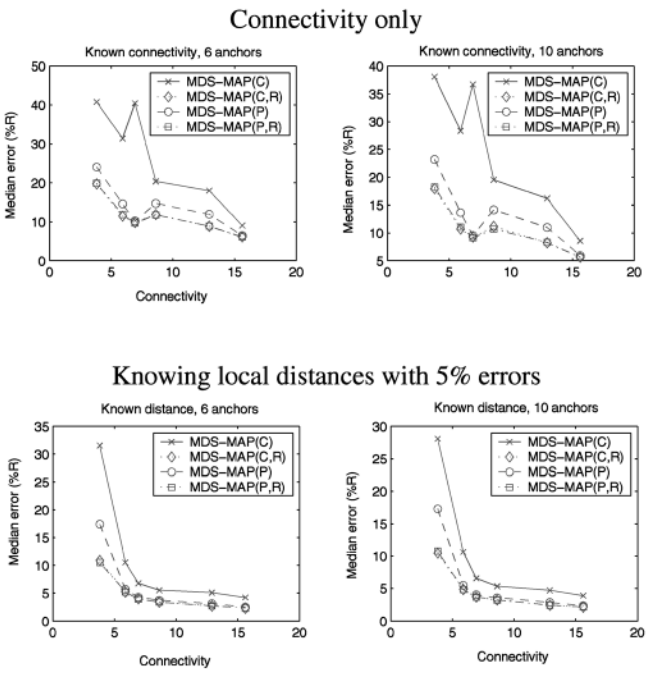

Fig. 16. Comparison of the MDS-MAP methods on uniform grid-like networks with 6 or 10 random anchors.

On networks with similar connectivity levels, the results of MDS-MAP(C) on the irregular networks are worse than those on the uniform networks. In contrast, MDS-MAP(C, R), MDS-MAP(P), and MDS-MAP(P, R) perform well when the connectivity is relatively high.

Having accurate estimates of local distances does not improve the performance of MDS-MAP(C), but helps MDS-MAP(P) and MDS-MAP(P, R) tremendously. The results of MDS-MAP(P) and MDS-MAP(P, R) are very close, indicating that the refinement step in MDS-MAP(P, R) does not do much.

Finally, we compare MDS-MAP(P) and MDS-MAP(P, R) with DV-hop and DV-distance [24]. The results are shown in Fig. 15. For the uniform networks, when using connectivity information only, the errors of MDS-MAP(P) and MDS-MAP(P, R) are consistently less than half of those of DV-hop. When using accurate local distances, the errors of MDS-MAP(P) and MDS-MAP(P, R) are just one quarter of those of DV-distance when the connectivity is low.

For the irregular networks, MDS-MAP(P) and MDS-MAP(P, R) are not better than DV-hop and DV-distance when the connectivity is low. That is because the local maps of MDS-MAP(P) are not very accurate for low connectivity. The error of a local map in an irregular network can significantly affect the global map. It is a different story when the connectivity is high. When using connectivity information only, the errors of MDS-MAP(P) and MDS-MAP(P, R) are about half of those of DV-hop. When using accurate local distances, the errors of MDS-MAP(P) and MDS-MAP(P, R) are just one quarter of those of DV-distance.

4.2 Grid Placement

We have done similar experiments on the grid networks. Fig. 16 compares the results on uniform grid networks with 6 or 10 random anchors. Unlike our results on the random networks, the estimation errors may increase when the connectivity increases in the connectivity-only case. This is due to the special structures of the grids. As the radio range increases, nodes lying at diagonally adjacent intersections come within range of each other. For example, when the radio range increases from $1.25r$ (connectivity 5.8) to $1.5r$ (connectivity 6.9), nodes that are actually either $1r$ or $1.414r$ apart both are 1-hop apart, increasing the error in the distance estimates.

In general, all the methods obtain much better results at a given connectivity level on the grid networks than on the random networks, either using connectivity or distance information. Better relative maps are computed given more regular, although fewer, connections.

MDS-MAP(C) is consistently the worst among the four methods. MDS-MAP(P) performs comparably with MDS-MAP(C, R) and MDS-MAP(P, R) when the connectivity is high. In general, the MDS-MAP methods perform well when the level of connectivity is over 6 for the grid placements and over 12 for the random placement.

Fig. 17 shows their results on irregular grid networks. MDS-MAP(C) performs poorly, whereas MDS-MAP(P) and MDS-MAP(P, R) perform well. The refinement in MDS-MAP(C, R) significantly improves the solutions of MDS-MAP(C), but it is still not as good as the other two methods.

5 Possible Extensions

As we have shown, the proposed algorithms work well for near-uniform radio propagation. However, in the real world, radio propagation indoors and in cluttered circumstances is far from uniform. Local distance estimation may also be poor. Further simulations will be needed to determine how robust MDS-based algorithms can be to such errors. Efficiently handling changes in a dynamic network may also be important in some applications [25], [26], [27].

As we have described it, MDS-MAP(P) builds local relative maps and merges these smaller maps to get a larger relative

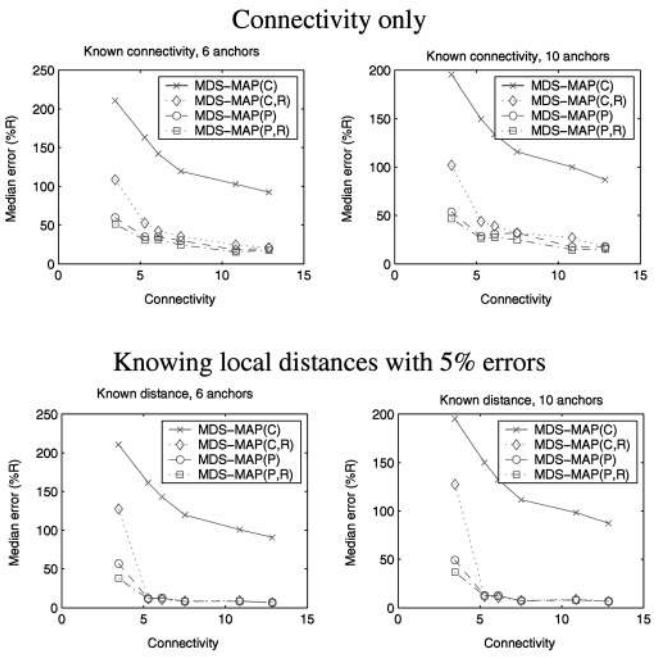

Fig. 17. Comparison of the MDS-MAP methods on irregular grid networks with 6 or 10 random anchors.

map. This is useful for applications that use relative maps. For applications that require absolute coordinates of nodes, waiting until a large map has formed before transforming to absolute coordinates may be a poor choice. Using the methods described here, distributed algorithms that compute absolute coordinates of individual nodes or subnetworks independently can be developed.

One interesting feature of MDS-MAP(P) is that it shows how information at different length scales can be used differently. Long distance shortest-path information is used only for rough layout decisions, while two-hop information is used to determine precise node positions. It would be interesting to develop a framework that precisely characterizes the contribution of each datum to the position estimation. The main question is whether an approach based on unified statistical inference could be as efficient as the special-purpose algorithms explored here.

MDS-MAP algorithms can be extended by applying more advanced MDS techniques. Instead of classical metric MDS, other MDS techniques such as ordinal MDS and MDS with missing data can be applied. We have done some experiments with ordinal MDS. Our results show that ordinal MDS is better than classical MDS when the connectivity level of the network is low and is comparable with classical MDS when the connectivity level is high. However, the computational cost of ordinal MDS is much higher than that of classical MDS. In another direction, developing MDS techniques that take advantage of anchor node locations directly might allow immediate computation of absolute coordinates.

Finally, it would be nice to obtain error bounds for MDS-MAP algorithms on regular and random networks. Such theoretical results may well guide the development of more efficient methods.

6 Conclusions

We presented a new approach for localization that works well with mere connectivity information. It can also incorporate distance information between neighboring nodes when it is available. The strength of the approach is that it can be used when there are few or no anchor nodes. Previous methods often require well-placed anchors to work well. Our approach does not have this limitation. It builds a relative map of the nodes even when no anchor nodes are available. With three or more anchor nodes, the relative map can be transformed into absolute coordinates. Extensive simulations using various network topologies and different levels of ranging error show that the method is effective and surpasses previous methods, especially when there are few anchors. An optional refinement step can be used to further improve the quality of the solution, at the expense of additional computation. A patching-based variation not only allows distributed and parallel computation, but also gives better solutions, especially on irregularly-shaped networks.

Acknowledgments

This work was supported in part by the US Defense Advanced Research Project Agency (DARPA) NEST program under contract F33615-01-C-1904. This paper is based on work reported at ACM MobiHoc 2003 [1] and IEEE Infocom 2004 [2].

References

[1] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from Mere Connectivity," *Proc. ACM MobiHoc*, pp. 201-212, June 2003.
[2] Y. Shang and W. Ruml, "Improved MDS-Based Localization," *Proc. IEEE Infocom*, 2004.
[3] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks," Technical Report ucla/csd-tr-02-0013, Computer Science Dept., Univ. of California, Los Angeles, 2002.
[4] E. Royer and C. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Comm.*, Apr. 1999.
[5] Y. Yu, R. Govindan, and D. Estrin, "Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," Technical Report ucla/csd-tr-01-0023, Computer Science Dept., Univ. of California, Los Angeles, May 2001.
[6] M. Chu, H. Haussecker, and F. Zhao, "Scalable Information-Driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks," *Int'l J. High Performance Computing Applications*, June 2002.
[7] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. Sixth Int'l Conf. Mobile Computing and Networks (ACM Mobicom)*, 2000.
[8] D.B. Johnson and D.B. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, T. Imielinski and H. Korth, eds., pp. 153-181, Kluwer Academic, 1996.
[9] B. Karp and H.T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. Sixth Int'l Conf. Mobile Computing and Networks (ACM Mobicom)*, 2000.
[10] I. Borg and P. Groenen, *Modern Multidimensional Scaling, Theory and Applications* New York: Springer-Verlag, 1997.
[11] J. Tenenbaum, V. de Silva, and J. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319-2323, 2000.
[12] W. Glunt, T.L. Hayden, and M. Raydan, "Molecular Conformation from Distance Matrices," *J. Computational Chemistry*, vol. 14, no. 1, pp. 114-120, 1993.

[13] W.S. Torgerson, "Multidimensional Scaling of Similarity," *Psychometrika,* vol. 30, pp. 379-393, 1965.
[14] R.N. Shepard, "Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function I & II," *Psychometrika,* vol. 27, pp. 125-140, 219-246, 1962.
[15] A. Savvides, H. Park, and M. Srivastava, "The Bits and Flops of the n-Hop Multilateration Primitive for Node Localization Problems," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02),* pp. 112-121, Sept. 2002.
[16] C. Savarese, J. Rabaey, and K. Langendoen, "Robust Positioning Algorithm for Distributed Ad-Hoc Wireless Sensor Networks," *Proc. USENIX Technical Ann. Conf.,* June 2002.
[17] J. Hightower and G. Boriello, "Location Systems for Ubiquitous Computing," *Computer,* vol. 34, no. 8, pp. 57-66, Aug. 2001.
[18] A. Savvides, C.C. Han, and M. Srivastava, "Dynamic Fine-Grained Localization in Ad Hoc Networks of Sensors," *Proc. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MOBICON),* July 2001.
[19] A. Nasipuri and K. Li, "A Directionality Based Location Discovery Scheme for Wireless Sensor Networks," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02),* pp. 105-111, Sept. 2002.
[20] A. Howard, M.J. Mataric, and G.S. Sukhatme, "Relaxation on a Mesh: A Formalism for Generalized Localization," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '01),* pp. 1055-1060, 2001.
[21] S.I. Roumeliotis and G.A. Bekey, "Synergetic Localization for Groups of Mobile Robots," *Proc. 39th IEEE Conf. Decision and Control,* Dec. 2000.
[22] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-Less Low-Cost Outdoor Localization for Very Small Devices," *IEEE Personal Comm.,* vol. 7, no. 5, pp. 28-34, Oct. 2000.
[23] L. Doherty, L. El Ghaoui, and K. Pister, "Convex Position Estimation in Wireless Sensor Networks," *Proc. Infocom 2001,* Apr. 2001.
[24] D. Niculescu and B. Nath, "Ad-Hoc Positioning System," *Proc. IEEE GlobeCom,* Nov. 2001.
[25] A. Cerpa and D. Estrin, "Ascent: Adaptive Self-Configuring Sensor Networks Topologies," *Proc. IEEE InfoComm,* June 2002.
[26] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *ACM Wireless Networks J.,* vol. 8, no. 5, Sept. 2002.
[27] Y. Xu, J. Heidemann, and D. Estrin, "Geography-Informed Energy Conservation for Ad Hoc Routing," *Proc. ACM MobiComm,* July 2001.

Yi Shang received the PhD degrees from the University of Illinois at Urbana-Champaign in 1997. He is an associate professor in the Department of Computer Science at the University of Missouri at Columbia. His research interests include nonlinear optimization, wireless sensor networks, intelligent systems, and distributed processing. He is a member of the IEEE, the IEEE Computer Society, and the ACM.

Wheeler Ruml received the AB and PhD degrees from Harvard University in 1993 and 2002, respectively. He is a member of the research staff at the Palo Alto Research Center. His research interests include heuristic search and optimization, cognitive modeling, and information visualization. He is a member of the ACM and a life member of the AAAI.

Ying Zhang received the PhD degree in computer science from University of British Columbia in 1994. She is a member of the research staff at the Palo Alto Research Center. Her current work is on control and communication architectures for massively distributed embedded systems, including sensor networks and modular reconfigurable robots. She is a member of the ACM and the IEEE.

Markus Fromherz received the PhD degree in computer science in 1991 from the University of Zurich (Switzerland) and the MS degree in computer science in 1987 from ETH Zurich. He is a principal scientist in the Systems and Practices Laboratory of the Palo Alto Research Center. His research interests are in the domain of intelligent embedded software, in particular, constraint-based modeling, model-based planning, scheduling, and control, and model-based design analysis and optimization. He has published in the areas of software engineering, transformational program development, model-based computing, online scheduling, embedded constraint solving, and design optimization. He has also coauthored more than 30 patents. He is a member of the IEEE.