Improved MDS-Based Localization

Yi Shang
Dept. of Computer Science
University of Missouri-Columbia
Columbia, MO 65211
Email: shangy@missouri.edu

Wheeler Ruml
Palo Alto Research Center
Palo Alto, CA 94304
Email: ruml@parc.com

***Abstract*— It is often useful to know the geographic positions of nodes in a communications network, but adding GPS receivers or other sophisticated sensors to every node can be expensive. MDS-MAP is a recent localization method based on multidimensional scaling (MDS). It uses connectivity information—who is within communications range of whom—to derive the locations of the nodes in the network, and can take advantage of additional data, such as estimated distances between neighbors or known positions for certain anchor nodes, if they are available. However, MDS-MAP is an inherently centralized algorithm and is therefore of limited utility in many applications. In this paper, we present a new variant of the MDS-MAP method, which we call MDS-MAP(P) standing for MDS-MAP using patches of relative maps, that can be executed in a distributed fashion. Using extensive simulations, we show that the new algorithm not only preserves the good performance of the original method on relatively uniform layouts, but also performs much better than the original on irregularly-shaped networks. The main idea is to build a local map at each node of the immediate vicinity and then merge these maps together to form a global map. This approach works much better for topologies in which the shortest path distance between two nodes does not correspond well to their Euclidean distance. We also discuss an optional refinement step that improves solution quality even further at the expense of additional computation.**

I. 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 [1]. For example, future sensor networks will involve a very large number of sensor nodes densely deployed over physical space. In particular, 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 find nodes with sensor data in a given range. Most of these tasks require knowing the positions of the nodes, or at least relative positions among them. With a network of thousands of nodes, it is unlikely that the position of each node has been pre-determined. Nodes could be equipped with a global positioning system (GPS) to provide them with absolute position, but this is currently a costly solution.

MDS-MAP [2] is a newly proposed localization method based on multidimensional scaling (MDS). It determines 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. If the distances between neighboring nodes can be measured, that information can be easily incorporated into the method. MDS-MAP is able to generate relative maps that represent the relative positions of nodes when there are no "anchor" nodes that have known absolute coordinates. When the positions of a sufficient number of anchor nodes are known, e.g., 3 anchors for 2-D localization and 4 anchors for 3-D, MDS-MAP then determines the absolute coordinates of all nodes in the network. MDS-MAP often outperforms previous methods when nodes are positioned relatively uniformly in space, especially when the number of anchors is low. MDS-MAP uses the distance or connectivity information between all nodes at the same time, whereas previous triangulation-based methods localize one unknown node at a time and only use the information between the unlocalized and anchor nodes.

However, like many existing methods, MDS-MAP does not work well on irregularly-shaped networks, where the shortest path distance between two nodes does not correlate well with their true Euclidean distance. In this paper, we build on the basic MDS-MAP method to develop a new algorithm that works well on both uniform and irregular networks. The main idea is to compute a local map using MDS for each node consisting only of nearby nodes, and then to merge these local-area maps together to form a global map. Thus we call the new technique MDS-MAP(P), which stands for MDS-MAP using patches of relative maps. This approach avoids using shortest path distances between far away nodes, and the smaller local maps constructed using local information are usually quite good. Another advantage of the new method is that it can be done in a distributed fashion, which makes it appropriate for large-scale networks.

An optional refinement step using least-square minimization may be used to refine the relative maps computed by MDS. MDS is often good at finding the right general layout of the network, but not the precise locations of nodes. That makes the MDS solution a good starting point for the local optimization done in the refinement step. This starting point is better

than those obtained by other methods such as collaborative multilateration [3]. The refinement improves solution quality but is much more expensive than MDS. MDS computes analytical solutions in $O(n^3)$, where n is the number of nodes, where the least-square minimization is solved by an iterative optimization method, which is about two orders of magnitude slower than MDS for 100-node networks and even more for larger networks. Thus, the refinement provides a trade-off between solution quality and computational cost.

The next section of the paper describes the new MDS-MAP(P) method in detail. Then, after an overview of previous proposals, we will present an extensive empirical evaluation. We will compare the new algorithm's performance with previous methods on various uniform and irregular networks, with node locations either chosen randomly or according to a rough grid layout.

II. Localization using MDS

A. Problem Formulation

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 zero or more may be special nodes, which we call anchors, whose positions are already known. 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. If an outlying node is not within communications range of any other nodes, we obviously have no way of estimating its position.

There are two possible outputs when solving the localization problem. One is a relative map and the other is an absolute map. 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 in 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 discovery and tracking [6], [7].

B. MDS-MAP

MDS-MAP is based on a well-established technique known as classical multidimensional scaling (MDS). MDS has its origins in psychometrics and psychophysics. It is a set of data analysis techniques that display the structure of distance-like data as a geometrical picture [8]. MDS starts with one or more matrices representing distances or similarities between objects and finds a placement of points in a low-dimensional space, usually two- or three-dimensional, where the distances between the 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.

There are many types of MDS techniques, including metric MDS and nonmetric MDS, replicated MDS, weighted MDS, deterministic and probabilistic MDS. In classical metric MDS, proximities are treated as distances in a Euclidean space [9]. Analytical solutions are derived from the proximity matrix efficiently through singular value decomposition and provide the best low-rank approximation (e.g., 2-D space) in the least squared error sense. In practice, the technique tolerates error gracefully, due to the overdetermined nature of the solution. Because classical metric MDS has a closed-form solution, it can be performed efficiently on large matrices.

In the basic MDS-MAP algorithm [2], given a network, the values of the edges are assigned to a constant, such as 1, when only connectivity information is available. Otherwise, if the distance of two neighbor nodes is known, the value of the corresponding edge is the measured distance. MDS-MAP consists of the following three steps:

1) Compute 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. The time complexity is $O(n^3)$, where n is the number of nodes.
2) Apply classical MDS to the distance matrix, retaining the first 2 (or 3) largest eigenvalues and eigenvectors to construct a 2-D (or 3-D) relative map. Again, the time complexity is $O(n^3)$,
3) Given sufficient anchor nodes (3 or more for 2-D networks, 4 or more for 3-D networks), the coordinates of the anchors in the relative map are mapped to their absolute coordinates through a linear transformation. The best linear transformation between the absolute positions of the anchors and their positions in the relative map is computed. Finding the transformation takes $O(m^3)$ time, where m is the number of anchors. Applying the transformation to all nodes takes $O(n)$ time.

Classical MDS requires the distance between every pair of nodes. The shortest path distance between two remote nodes provides an estimate of the true Euclidean distance. This estimate is fine when the networks are dense or uniform, but is not good for very irregular ones. When the estimation is off, the result of classical MDS is not good. The new MDS-MAP methods presented in the next section address this issue.

C. The MDS-MAP(P) Method

In the new MDS-MAP method, individual nodes compute their own local maps using their local information and then the local maps are merged to form a global map. We call it MDS-MAP(P).

In MDS-MAP(P), each node applies MDS-MAP to compute a local map that includes only relatively nearby nodes, e.g., those within two communication hops. Two maps are then merged together based on their common nodes. The best linear transformation (minimizing conformation errors) is computed to transform the coordinates of the common nodes in one map to those in the other map. This computation can be done efficiently.

The steps of MDS-MAP(P) are as follows:

1) Set the range for local maps, R_{lm}. For each node, neighbors within R_{lm} hops are involved in building its local map. The value of R_{lm} affects the amount of computation in building the local maps, as well as the quality. We use $R_{lm} = 2$ in the experiments reported here.
2) Compute local maps for individual nodes. For each node, do the following:
   a) Compute shortest paths between all pairs of nodes in its local mapping range R_{lm}. The shortest path distances are used to construct the distance matrix for MDS.
   b) Apply MDS to the distance matrix and retain the first 2 (or 3) largest eigenvalues and eigenvectors to construct a 2-D (or 3-D) local map.
   c) Refine the local map. Using the node coordinates in the MDS solution as the initial point, we perform least squares minimization to make the distances between nearby nodes match the measured ones. We discuss the exact formulation below. In our prototype implementation, this refinement step is more computationally expensive than MDS.

   The overall complexity of computing each local map is $O(k^3)$, where k is the average number of neighbors. Thus the complexity of computing n local maps is $O(k^3 n)$, where n is the number of nodes.
3) Merge local maps. Local maps can be merged sequentially or in parallel. There are various ways of merging local maps sequentially, such as randomly or according to certain order best for an application. In this paper, we use a simple strategy. For clarity, the process will be described from a centralized point of view, although it need not involve the entire network. First we randomly pick a node and make its local map the core map. Then we grow the core map by merging maps of neighboring nodes to the core map. Each time a neighbor's map with the maximal number of common nodes with the core map is selected. Eventually the core map covers the whole network. As we explain below, if the merges are chosen carefully, the complexity of this step is $O(k^3 n)$, where k is the average number of neighbors and n is the number of nodes.
4) Refine the global map (optional). Using the node coordinates in the global map as the initial solution, we apply least squares minimization to make the distances between neighboring nodes match the measured ones. This step is $O(n^3)$ and is much more expensive than the other steps for large networks.
5) Given sufficient anchor nodes (3 or more for 2-D networks, 4 or more for 3-D networks), transform the global map to an absolute map based on the absolute positions of anchors. For r anchors, the complexity of this step is $O(r^3 + n)$.

When the optional refinement step in Step 4 is used, we will refer to the algorithm as MDS-MAP(P,R). For large-size networks of constant density, k is limited by a constant and k and r are much smaller than n. Thus the complexity of MDS-MAP(P) is $O(n)$, linear in terms of the number of nodes in the network. This nice property makes MDS-MAP(P) suitable for large-scale networks.

Step 2(c) and 4 use similar least squares minimization techniques to refine relative maps. The minimization problem has many local minima, but relatively efficient local optimization techniques such as the Levenberg-Marquardt gradient descent method can be used. Generic global optimization techniques such as simulated annealing or genetic algorithms may also be applied, but they are slow. When using local optimization, if the starting point is not good, the solution of local optimization will not be good.

MDS can provide a very good starting point for local optimization. MDS is good at finding the right topology of the network, but not the precise locations of nodes, because MDS uses shortest path distances to approximate the distance between nodes more than 1 hop away and the approximation may not be accurate. Although this starting point is often much better than those obtained by other methods such as collaborative multilateration [3], errors in the $O(n^2)$ long-distance estimates can overwhelm the $O(n)$ short-range measurements. The refinement technique improves the relative maps by forcing them to conform more closely to the distances to nearby neighbors.

The exact objective function used during refinement measures not only distances between one-hop neighbors, but also distances between some multi-hop neighbors, although these distances are weighted less. We use a refinement range R_{ref}, defined in terms of hops, to specify what information is considered. $R_{ref} = 1$ means only distances between immediate neighbors are considered; $R_{ref} = 2$ means distances to all nodes within two hops are considered; and so on. Different values of R_{ref} offer trade-offs between computational cost and solution quality.

Specifically, let $(x_i, y_i), i = 1, \cdots, N$ represent the coordinates of the N nodes in a local map; $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ be the Euclidean distance between two nodes i and j; and p_{ij} be the proximity of nodes i and j. When only connectivity information is available, $p_{ij} = 1$ if i and j are 1-hop neighbors, $p_{ij} = 2$ if i and j are 2 hops apart,

and so on so forth. When distance measures between 1-hop neighbors are available, p_{ij} is the distance measure between i and j if they are 1-hop neighbors, and p_{ij} is the shortest path distance if i and j are more than 1 hop away. When there are actual distance estimates in some part of the networks and only connective information in the other part, the p_{ij} of hop counts are multiplied by the average hop distances. The objective of the refinement step is:

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

where w_{ij} is the weight. If $w_{ij} = 0$ for all i and j that are more than 1-hop away, then only the 1-hop connectivity or distance measures are used. In our experiments, we found that the information between 2-hop neighbors is also helpful. Thus we set R_{ref} to 2, and $w_{ij} = 1$ when i and j is 1-hop away and $w_{ij} = 1/4$ when they are two hops apart. The refinement improves the map by giving local information between neighbor nodes more weight than that between far away nodes, which may be less accurate.

In our experiments, we use the Levenberg-Marquardt method (lsqnonlin in Matlab's optimization toolbox) to solve the problem. For a 2-D n-node network, the problem has $2n$ variables and no constraints. The Jacobian can also be computed analytically. Usually only the first few iterations of lsqnonlin give significant improvement. Thus the maximum number of iteration is set to a small number, e.g., 10. Although this local optimization algorithm is fast, it is considerably slower than classical MDS. For 100-node networks, it is about two orders of magnitude slower. For larger networks, the time difference becomes larger.

After forming the local maps, they must be merged. We use an incremental greedy algorithm. Let $LN(p) = \{q | q$ is within R_{lm} hops from $p\}$ represent the set of nodes in p's neighborhood; and $LM(p)$ represent the local map of node p, which contains the coordinates of nodes in $LN(p)$. First, randomly select a node p and construct the set $C = \{p\}$ of nodes that have been considered. Let the set $D = LN(p)$ represent those nodes that have been localized, and $M = LM(p)$ be the current global map. Iterate the following steps until C contains all nodes.

1) Find $p' \in D - C$ such that the number of common nodes in D and $LN(p')$ is the largest.
2) Merge M and $LM(p')$. Set $C = C \cup \{p'\}$, $D = D \cup LN(p')$, and $M =$ merge $(M, LM(p'))$.

Two local maps are combined by minimizing the conformation difference of their common nodes subject to the best linear transformation. Because we do not mandate a rigid mapping, our method is more resilient to errors.

For each local map M, we maintain two lists of nodes, $OPEN(M)$ and $CLOSED(M)$. Nodes in $CLOSED(M)$ have been considered, i.e., their local maps have already been merged into M. The local maps of nodes in $OPEN(M)$ have not been merged. For node p,

$$\begin{aligned} CLOSED(LM(p)) &= \{p\} \\ OPEN(LM(p)) &= LN(p) - \{p\} \end{aligned}$$

The actual merging of two maps M and M' is straightforward:

1) Compute the intersection $I = (CLOSED(M) \cup OPEN(M)) \cap (CLOSED(M') \cup OPEN(M'))$.
2) Find a linear transformation T of the nodes in I from their coordinates in M' to those in M, such that the sum of squared errors between M and $T(M')$ is minimized. T have two options. One includes translation, reflection, and orthogonal rotation. The other includes scaling as well. In our experiments, we report results using the option without scaling, which works slightly better than the one with scaling. Without scaling, the information from both individual maps is better preserved in the combined map.
3) Update M and related data. Set

$$\begin{aligned} OPEN(M) = (&OPEN(M) \cup OPEN(M')) \\ &-(CLOSED(M) \cup CLOSED(M')) \\ CLOSED(M) = &CLOSED(M) \cup CLOSED(M') \end{aligned}$$

To get the coordinates of a node p in the new combined map M,

- if p is in the old M but not in M', use its coordinates in M;
- if p is in M' but not in the old M, use its coordinate in $T(M')$;
- otherwise, use the average of p's coordinates in the old M and $T(M')$.

Figure 1 shows an example of merging two local maps by this method. In the topmost panels, a polygon has been drawn around the nodes present in both local maps. The two maps were computed using connectivity information only. The linear transformation T for map 2 is

$$T(x, y) = (x\ y) \begin{pmatrix} 0.981 & -0.196 \\ 0.196 & 0.981 \end{pmatrix} + (0.016\ -0.835) \qquad (2)$$

When three or more anchors are present in a 2-D subnetwork, an absolute map can be computed. Since the algorithm does not require anchor nodes in building a relative map of a sub-network, it can be applied to many sub-networks in parallel. Distributed map merging has a number of benefits, including more balanced computation and communication among the nodes, faster construction of the global map, and information of multi-level granularity being distributed in the network, leading to better support for flexibility and robustness.

The communication cost in a distributed implementation of MDS-MAP(P) is proportional to the sizes of the local maps. The number of messages depends on the mapping range R_{lm} and whether local distance measures are available. For $R_{lm} = 2$ and only using connectivity, each node only needs to know

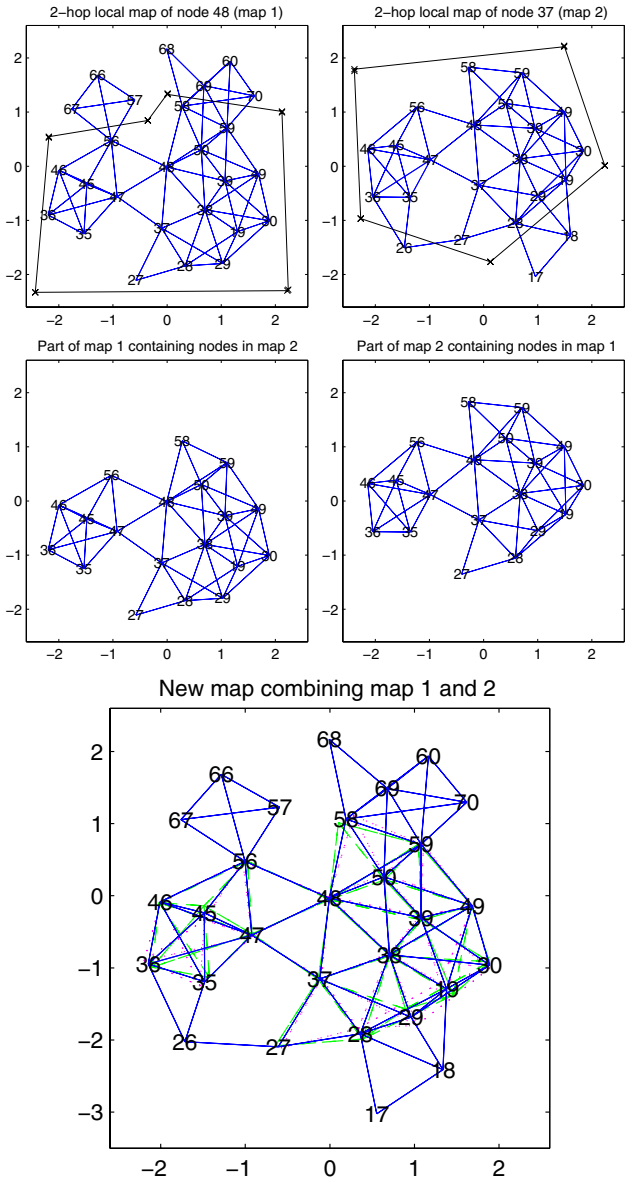

Fig. 1. An example of merging two local maps based on their common nodes.

the IDs of its 2-hop neighbors. First each node broadcasts its ID and each one records its 1-hop neighbors. Then each node broadcasts the IDs of its 1-hop neighbors. Now each node is ready to compute its own local map. In merging the local maps to form a global map, whether done sequentially or in parallel, the information of each node's local map has to arrive at the destination. Using a binary aggregation tree, each local map is sent through $O(\log n)$ hops. So the total communication cost is $O(n \log n)$.

To summarize, the key difference between MDS-MAP(P) and the basic MDS-MAP is that MDS-MAP(P) computes small relative maps using local information, instead of a global map using pair-wise distances between any two nodes. For MDS-MAP(P) to work, the local maps have to be accurate enough so that when they are merged together to form a global map, errors will not become too large. Empirically, we found that when the connectivity level is over 12 for random networks and over 6 for grid networks, good local maps can be constructed using nodes within 2 hops. Using only nodes within 1-hop distance does not work as well, especially when only connectivity information is available.

TABLE I

THE TYPICAL TIME IN SECONDS TAKEN BY THE MAJOR STEPS OF MDS-MAP(P,R) FOR DIFFERENT SIZE NETWORKS.

Network Size	Compute local maps		Merge local maps	Refine global map
	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

On the other hand, using nodes within 3-hop distance can produce local maps as good as using 2-hop neighbors, and sometimes slightly better. However, this is computationally more expensive. In addition, a larger local map means more information needs to be stored and transmitted. When the local maps are computed by sensor nodes in networks, each of them only have limited memory and communication bandwidth.

The rule of thumb is to set the size of the local map just large enough to get the desired accuracy. The right size depends on the topology of the network, the network density, and the accuracy of local distance measurement.

To give a concrete sense of the actual running time of the methods, Table I shows the typical time taken by the major steps of MDS-MAP(P,R). 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 on local maps is very fast and the refinement of local maps is about 2 orders of magnitude more expensive. (This may be due in part to our use of lsqnonlin in Matlab, which calculates many terms that will have weight zero in the objective function.) Note that the computation of local maps can be distributed to all the nodes in the network. Thus the computation at each node is not increased much as the network becomes larger. 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 quickly and becomes dominant for large networks. The desired extent of the global map will vary according to the application at hand.

D. Examples

We use four example problems to illustrate the behavior of these MDS-MAP-based algorithms. Two are uniform topologies and the other two are irregular topologies. They are shown in Figure 2. In the graphs, circles represent nodes and edges represent connections between nodes that are within communication range of each other.

1) Basic MDS-MAP *:* Figure 3 shows the results of the basic MDS-MAP method on the random uniform example. Four random anchor nodes (denoted by the star *) are used in the position estimation. (Note that this particular example

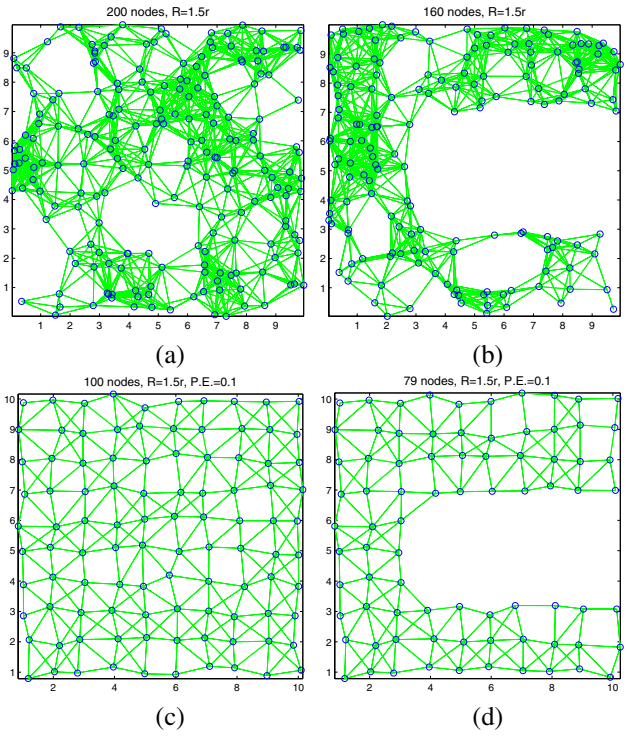

Fig. 2. Four example problems: (a) *random uniform* placement – 200 nodes are randomly placed in a $10r \times 10r$ square; (b) *random C-shaped* placement – 160 nodes are randomly placed in an area of C shape within a $10r \times 10r$ square; (c) *regular uniform* placement – 100 nodes are placed on a grid with $10\%r$ placement errors; and (d) *regular C-shaped* 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.

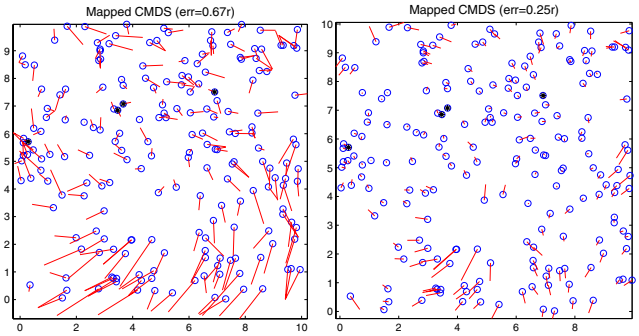

Fig. 3. Results of the basic MDS-MAP on the example of *random uniform* placement using connectivity only (left) or the *distance measures* between neighboring nodes with 5% errors (right). The same four random anchors are used and the position estimation errors are 0.67 and 0.25, respectively.

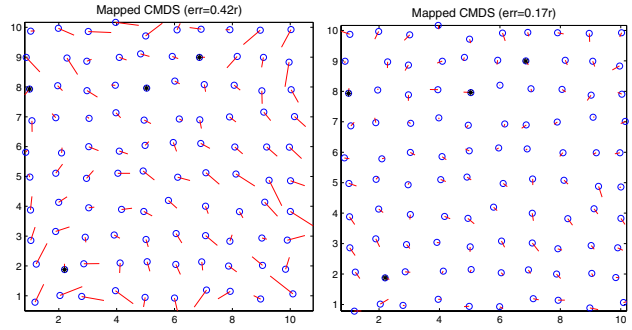

Fig. 4. Results of the basic MDS-MAP on the example of *grid* placement using connectivity only (left) or the *distance measures* between neighboring nodes with 5% errors (right). Four random anchors are used and the position estimation errors are 0.42 and 0.17, respectively.

shows a rather unlucky selection, as the four anchors are almost collinear.) The circles represent the true locations of the nodes and the lines connect the estimated positions with the true positions. The longer the line, the larger the error is. The results when using connectivity information and when using local distance measures are both shown. The example demonstrates that MDS-MAP works well on uniform topologies using only connectivity information and is better when accurate local distance measures are available. When the network has a regular topology, such as nodes being placed near grid points, MDS-MAP obtains very good solutions, as shown in Figure 4.

The basic MDS-MAP method performs badly on C-shaped topologies. Figure 5 shows the results of MDS-MAP on the random C-shaped example. Four random anchor nodes (denoted by *) are used. The average error of using connectivity is 2.4, very large. MDS-MAP does not work well because the shortest path distance between two nodes in the two wings is much bigger than their actual Euclidean distance. Accurate local distance measures do not help. The average error of using local distances with 5% error is 2.3, still very large.

The basic MDS-MAP method performs badly on irregular topologies even when they have regular internal structure, such as nodes being placed near grid points. Figure 6 shows the results of MDS-MAP on the *C-shaped grid* example. The basic MDS-MAP performs as poorly as it does on the random C-shaped example. Accurate local distance measures do not help either.

2) MDS-MAP(P) *and* MDS-MAP(P,R)*:* In this section, we use the examples to demonstrate that MDS-MAP(P) performs as well as the basic MDS-MAP method on uniform topologies and is much better on irregular topologies. When global refinement is added, which we call MDS-MAP(P,R), the solutions improves further.

Figure 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% of the error of the basic MDS-MAP. After refinement, the error of MDS-MAP(P,R) is $0.31r$, even better. Using local distance measures, MDS-MAP(P) and MDS-MAP(P,R) obtain better results. The error of MDS-MAP(P) is $0.16r$, much better than the basic MDS-MAP. After refinement, the error of MDS-MAP(P,R) is only $0.06r$.

Figure 8 shows the results on the random C-shaped placement example. Although MDS-MAP(P) (error $1.2r$) is better than the basic MDS-MAP (error $2.4r$), the error is still large. The global refinement in MDS-MAP(P,R) helps to reduce the error to $0.43r$. Using good local distance measures, the solution of MDS-MAP(P) (error $0.72r$) is improved and the

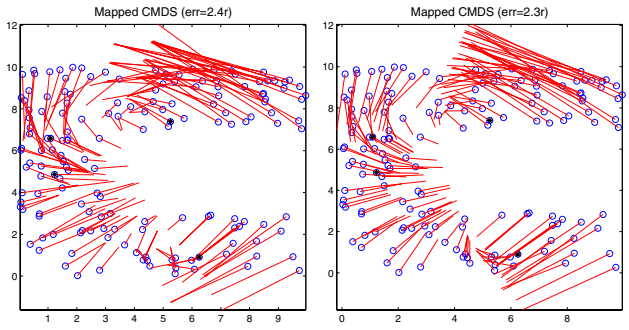

Fig. 5. Results of the basic MDS-MAP on the example of *random C-shaped* placement using connectivity only (left) or the *distance measures* between neighboring nodes with 5% errors (right). Four random anchors are used and the position estimation errors are 2.4 and 2.3, respectively.

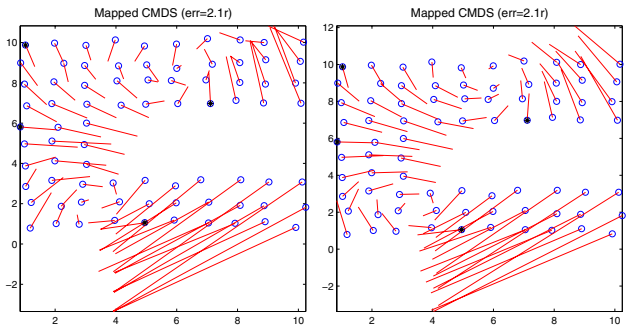

Fig. 6. Results of the basic MDS-MAP on the example of *C-shaped grid* placement using connectivity only (left) or the *distance measures* between neighboring nodes with 5% errors (right). Four random anchors are used and the position estimation errors are 2.1 for both cases.

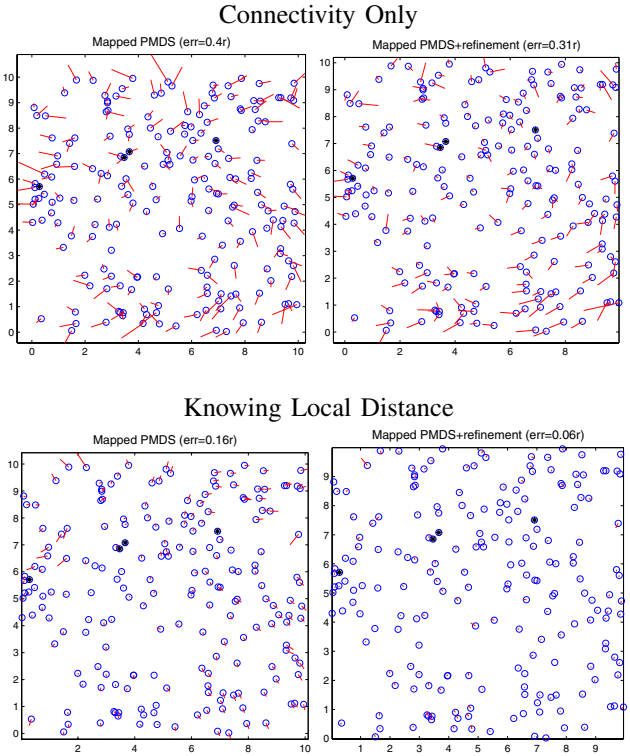

Fig. 7. Results of MDS-MAP(P) and MDS-MAP(P,R) on the example of *random uniform* placement. The upper two diagrams show their results using *connectivity* information only, whereas the lower two diagrams show their results using *distance measures* between neighboring nodes with 5% distance errors.

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) obtain very good results. For the uniform grid example, they find close to perfect solutions. For the C-shaped grid example, they also perform very well, as shown in Figure 9.

These specific examples have visually illustrated the performance of the methods. More extensive experimental results and further comparisons will be presented in Section IV.

III. 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 [10]. Many systems use some kind of range or distance information and many of them rely on powerful beacon nodes with unusual capabilities, such as radio or laser ranging devices. Others use distance or angle measures from a fixed set of reference points or anchor nodes. For example, the GPS-less system by Bulusu et al. [11] uses a grid of anchor nodes. 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.

Among those existing localization methods that use only connectivity information, Doherty's [12] convex constraint satisfaction method formulates the localization problem with uniform communication as a feasibility problem with convex radial constraints. The problem is solved by semi-definite programming (an interior point method). The method requires centralized computation. In addition, for the method to work well, it needs anchor nodes to be placed on the outer boundary, preferably at the corners. When the anchors are located in the interior of the network, there are many feasible solutions of the constraint satisfaction problem. The majority of them give large position estimation errors.

Most other methods are based on triangulation. These include DV-Hop and DV-distance methods by Niculescu and Nath [13], Hop-TERRAIN and refinement by Savarese et al. [14], and collaborative multilateration by Savvides et al. [3]. In the "DV-based" methods [13], 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 regular topologies. For sparse and irregular networks, the accuracy degrades to the radio range and is not very good. The "DV-distance" method uses distance between neighboring nodes to reduces the location error. These methods perform badly for irregular topologies.

The Hop-TERRAIN and refinement method [14] is similar to the "DV-based" methods. For the start-up phase, they use Hop-TERRAIN, an algorithm similar to DV-hop. Hop-

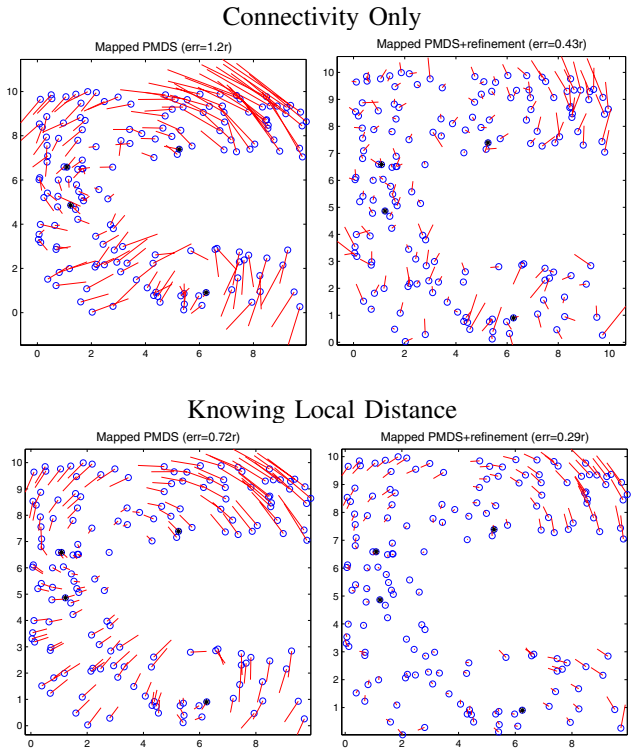

Fig. 8. Results of MDS-MAP(P) and MDS-MAP(P,R) on the example of *random C-shaped* placement. The upper two diagrams show their results using *connectivity* information only, whereas the lower two diagrams show their results using *distance measures* between neighboring nodes with 5% distance errors.

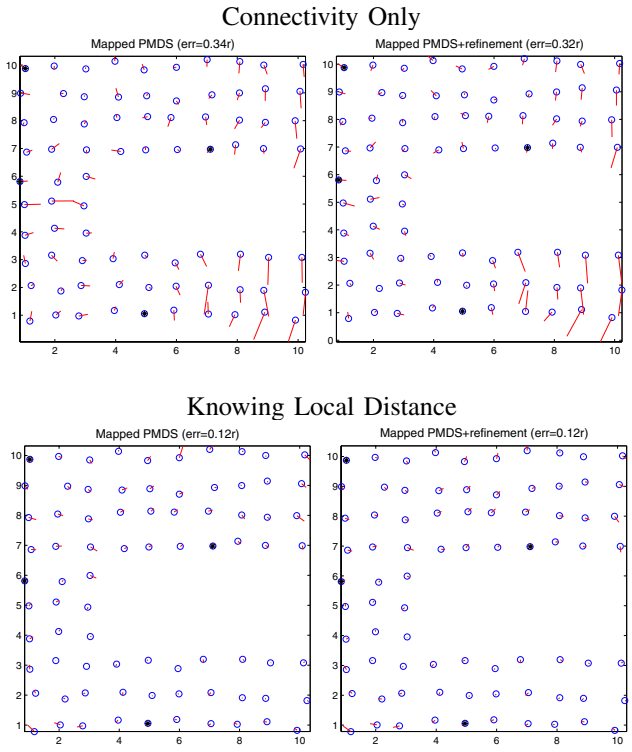

Fig. 9. Results of MDS-MAP(P) and MDS-MAP(P,R) on the example of *C-shaped grid* placement. The upper two diagrams show their results using *connectivity* information only, whereas the lower two diagrams show their results using *distance measures* between neighboring nodes with 5% distance errors.

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 method delivers localization accuracy comparable to that of the "DV-based" methods.

The collaborative multilateration method needs many anchors to work well [3]. The method estimates node locations by using anchor locations that are several hops away and distance measures to neighboring nodes. The method has three main phases: (1) formation of a collaborative subtree, which only includes nodes that can be uniquely determined, (2) computation of initial 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 coverage of multilateration is low when the number of anchors is small, and many nodes are not localized.

Some previous approaches also build local maps [15], [16]. They are worse than our methods because their local maps are built using triangulation or geometry and have larger errors.

What is the single most important reason that MDS-based methods achieve better accuracy than previous methods? It is the joint utilization of all connectivity information, or local distance measures if they are available, among all nodes, including the ones that have yet to be localized. Most previous methods are based on triangulation. A common problem of triangulation-based methods is that one node at a time is triangulated based on anchors and the information between nodes of unknown position is not utilized.

IV. 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 many randomly-generated examples. The same types of networks are used as in Section II-D: (a) uniform random, 200 nodes randomly placed inside a $10r \times 10r$ square, where $r = 1$ is the placement unit length; (b) C-shaped random, 160 nodes randomly placed inside an area of C shape within a $10r \times 10r$ square; (c) uniform grid, 100 nodes placed on a $10r \times 10r$ grid; and (d) C-shaped grid, 79 nodes placed on a C shape 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 of 0 mean and $10\%r$ standard deviation 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

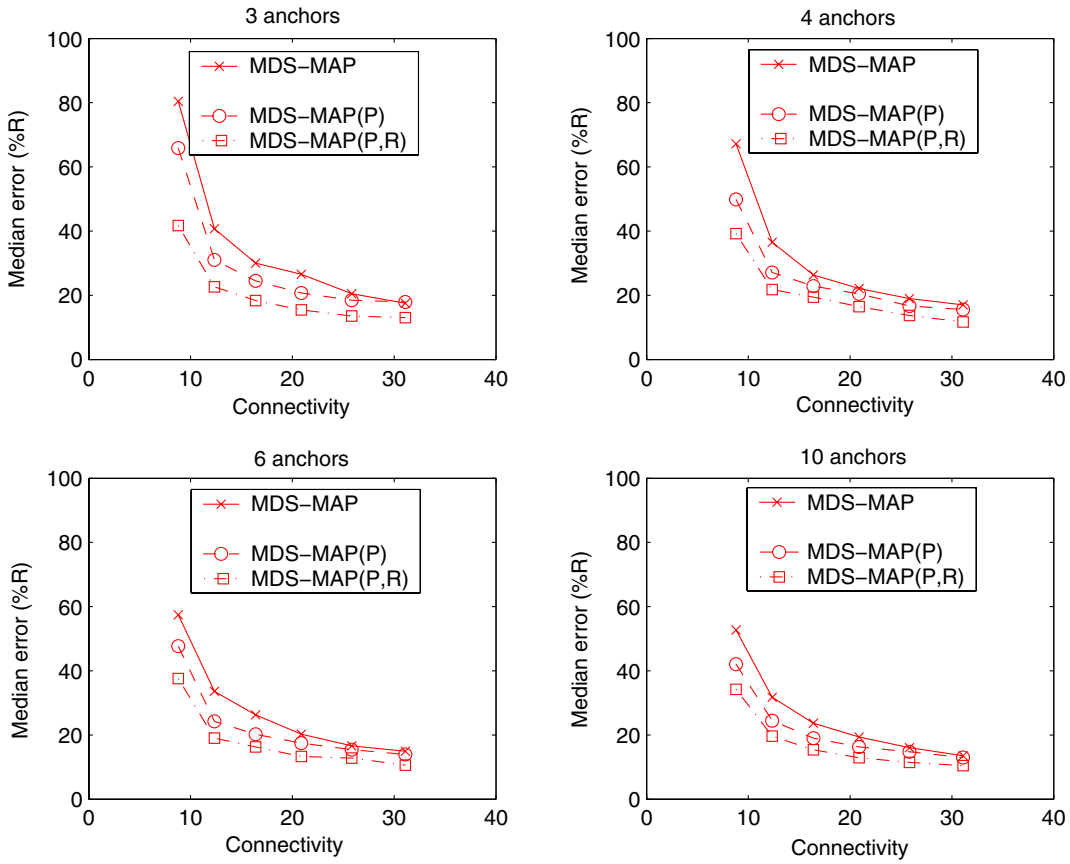

Fig. 10. Comparison of the basic MDS-MAP, MDS-MAP(P), and MDS-MAP(P,R) on *random uniform* networks with 3,4,6, and 10 anchors. Only *connectivity* information is used.

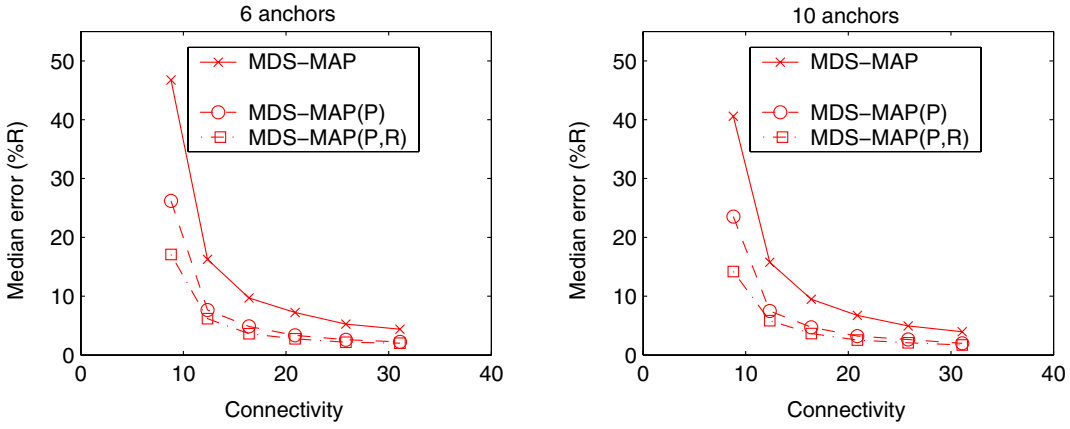

Fig. 11. Comparison of the basic MDS-MAP, MDS-MAP(P), and MDS-MAP(P,R) on *random uniform* networks with 6 and 10 anchors. Local *distance measures* between neighboring nodes with 5% errors are used.

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 radio range R.

Four examples of these test problems are shown in Figure 2 in the last section. The experiments were done in Matlab. The anchor nodes were selected randomly. Thirty random trials were conducted for each data point.

A. *Uniform Networks*

Figure 10 shows the results of the three MDS-MAP algorithms on the random uniform networks with 200 nodes. The errors are plotted against the average connectivity level. The radio ranges (R) are from $1.25r$ to $2.5r$, with an increment $0.25r$, which lead to average connectivity levels 8.8, 12.3, 16.4, 20.9, 25.9, and 31.1, respectively. Three, four, six, and

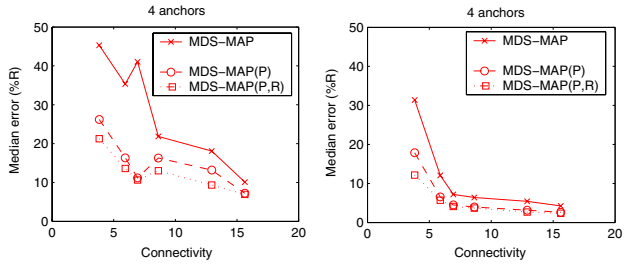

Fig. 12. Comparison of the basic MDS-MAP, MDS-MAP(P), and MDS-MAP(P,R) on *uniform grid* networks with 4 random anchors. The left diagram shows the results using *connectivity* only, whereas the right diagram shows the results using local *distance measures* between neighboring nodes with 5% errors.

ten random anchors are used.

For the case of using connectivity information only, MDS-MAP(P) is consistently better than the basic MDS-MAP and is more than $10\%R$ better when the connectivity is low. MDS-MAP(P,R) improves upon MDS-MAP(P). Although more anchors lead to better results, the improvement with more than 6 anchors is small.

Using connectivity information only, MDS-MAP algorithms are much better than the convex optimization approach in [12] 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 radio range is $1.25R$ and above. The results are also better than Hop-TERRAIN [14], especially when the number of anchors is small. For example, with 4 anchors (2%) and a connectivity level 12.3, 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 distance measures with 5% errors, all methods get much better results. The average error is roughly half of that obtained when using only proximity information. MDS-MAP(P) is comparable to MDS-MAP(P,R) when the connectivity level is 12.3 and above.

Unlike most previous methods, MDS-MAP localizes all nodes in a connected network and does not impose constraints such as a node having to possess three or more neighbors. However the nodes with only one or two neighbors are more likely to have larger errors. That is why the average errors for sparse networks are very large.

We have also done extensive experiments on grid networks. Figure 12 compares the results of the basic MDS-MAP, MDS-MAP(P), and MDS-MAP(P,R) on *uniform grid* networks with 4 random anchors. All three methods obtain much better results at lower connectivity levels on the grid networks than on the random networks, either using connectivity information only or local distance measures. The reason is that the local maps for the random networks are not as good as the ones for the grid networks.

MDS-MAP(P) does not need many anchors to achieve good solutions. For example, when there is sufficient connectivity, 3 anchors for the grid placements and 4 anchors for the random placement are usually enough to find good solutions.

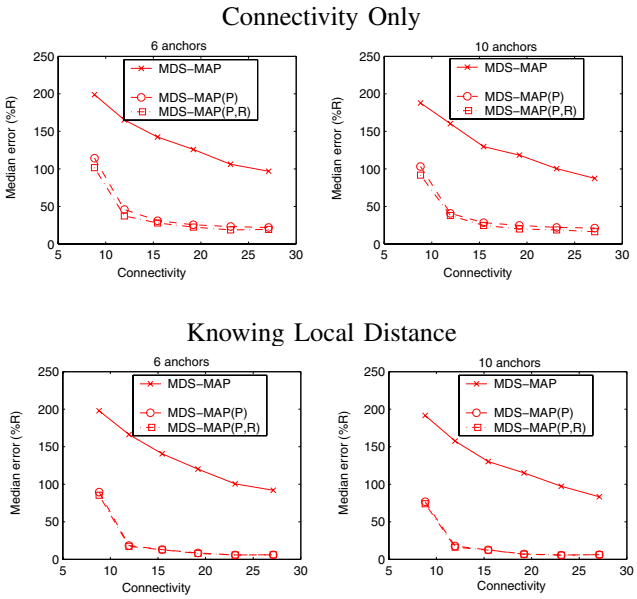

Fig. 13. Comparison of the basic MDS-MAP, MDS-MAP(P), and MDS-MAP(P,R) on *random C-shaped* networks with 6 and 10 anchors, using *connectivity* information (upper) or local *distance measures* between 1-hop neighbors with 5% errors (lower).

B. Irregular Networks

Irregular topologies are much harder than uniform topologies and previous methods reported very poor results on them [13].

Figure 13 shows the performance of the three MDS-MAP methods on the random C-shaped networks with 160 nodes. The radio ranges (R) are the same as before, from $1.25r$ to $2.5r$, with an increment $0.25r$, which leads to average connectivity levels 8.6, 12.0, 15.4, 19.2, 23.1, and 27.1, respectively.

On the random C-shaped networks, the basic MDS-MAP returns poor solutions because the distance estimations for nodes on separate wings are far from their actual Euclidean distances.

MDS-MAP(P) performs well on these C-shaped networks, especially when the connectivity level is 12.0 or more, and finds solutions just slightly worse than those by MDS-MAP(P,R). Again, although more anchors lead to better results, the improvement with more than 6 anchors is small.

Knowing good local distance measures does not make the basic MDS-MAP better, but helps MDS-MAP(P) and MDS-MAP(P,R), as shown in Figure 13. The results of MDS-MAP(P) and MDS-MAP(P,R) are very close, implying that the global refinement in MDS-MAP(P,R) does not do much in those circumstances.

On networks with similar connectivity levels, the results of the basic MDS-MAP on the C-shaped networks are much worse than those on the uniform networks. In contrast, MDS-MAP(P)

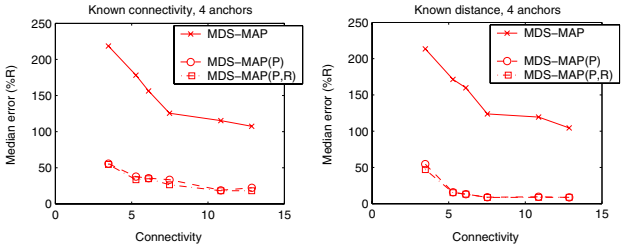

Fig. 14. Comparison of the basic MDS-MAP, MDS-MAP(P), and MDS-MAP(P,R) on *C-shaped grid* networks with 4 random anchors, using *connectivity* information (left) or local *distance measures* between 1-hop neighbors with 5% errors (right).

and MDS-MAP(P,R) perform quite well. They have larger errors for C-shaped networks than for uniform networks because some nodes on the main paths of the C-shaped networks have low connectivity, which leads to poor local maps. These poor local maps can affect the global map significantly.

Figure 14 shows their results on C-shaped grid networks. Again, MDS-MAP(P) and MDS-MAP(P,R) are much better than the basic MDS-MAP, and achieve good results.

Finally, we compare MDS-MAP(P) and MDS-MAP(P,R) with DV-hop and DV-distance on the random uniform and C-shaped networks. The results are shown in Figure 15. The results of DV-hop and DV-distance are similar to the ones in [13]. For the uniform networks, MDS-MAP(P) and MDS-MAP(P,R) are consistently much better than DV-hop and DV-distance. 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 local distance measures with 5% errors, 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. Their results become similar when the connectivity is high.

For the C-shaped 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 on the central path of an C-shaped network can significantly affect the global map. In contrast, in relatively uniform networks, the effects of local maps on the global map are much more constrained. When the connectivity is high, MDS-MAP(P) and MDS-MAP(P,R) are much better than DV-hop and DV-distance. 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 distance measures, the errors of MDS-MAP(P) and MDS-MAP(P,R) are just one quarter of those of DV-distance.

V. 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.

As we have described it, MDS-MAP(P) builds local relative maps and merges these smaller maps to get a larger relative map. This is useful for applications that use relative maps. For applications that requires 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 limited experiments with ordinal MDS (also known as nonmetric 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.

VI. Conclusions

We presented a new method, MDS-MAP(P), that improves the basic MDS-MAP algorithm significantly. It can be implemented in a distributed setting and performs exceptionally well on irregular topologies. The method builds a small relative map for each individual node. The relative maps are usually of high quality when connectivity is sufficiently high. For irregular networks, local information is much more accurate than estimates for distant nodes obtained using shortest path distance. By emphasizing this local information during a refinement step, MDS-MAP(P) obtains superior local maps. The relative maps can then be merged together to form a global map. A global refinement can be used to further improve the quality of the global map, which usually leads to smaller position estimation errors. The drawback of the global refinement is that it can be more computational expensive than MDS. Comparing the basic MDS-MAP and MDS-MAP(P), the former suffers from long-range distance estimation errors, whereas the latter from error propagation. A balance between the two may be struck to achieve the best result for a given network. Our experimental results show MDS-MAP(P) significantly outperforms existing methods for irregular topologies, particularly when the number of anchors is small.

Acknowledgment

This work was supported in part by DARPA under contract F33615-01-C-1904.

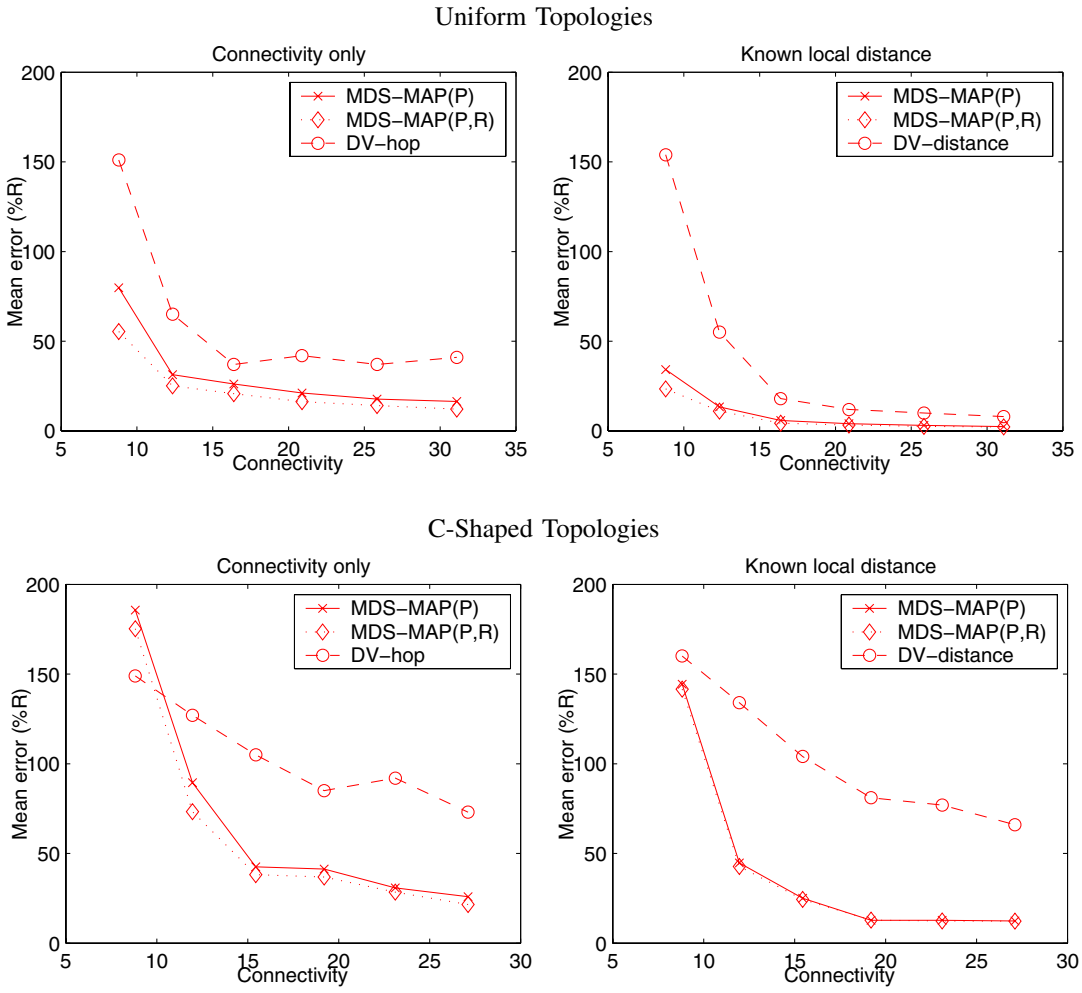

Fig. 15. Comparison of MDS-MAP(P), MDS-MAP(P,R), DV-hop, and DV-distance on 200-node *random uniform* networks (upper two diagrams) and 160-node *random C-shaped* networks (lower two diagrams), using *connectivity* information (left) or local *distance measures* between 1-hop neighbors with 5% errors (right). Four anchors were randomly selected for each case.

REFERENCES

[1] 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," UCLA Computer Science Department," Technical Report UCLA/CSD-TR-02-0013, 2002.

[2] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from mere connectivity," in *ACM MobiHoc*, Annapolis, MD, June 2003, pp. 201–212.

[3] A. Savvides, H. Park, and M. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, Sept. 2002, pp. 112–121.

[4] E. Royer and C. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, April 1999.

[5] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks," UCLA Computer Science Department," Technical Report UCLA/CSD-TR-01-0023, May 2001.

[6] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *Int. Journal on 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," in *Proc. 6th Int'l Conf. on Mobile Computing and Networks (ACM Mobicom)*, Boston, MA, 2000.

[8] I. Borg and P. Groenen, *Modern Multidimensional Scaling, Theory and Applications*. New York: Springer-Verlag, 1997.

[9] W. S. Torgeson, "Multidimensional scaling of similarity," *Psychometrika*, vol. 30, pp. 379–393, 1965.

[10] J. Hightower and G. Boriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, Aug. 2001.

[11] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, Oct. 2000.

[12] L. Doherty, L. E. Ghaoui, and K. Pister, "Convex position estimation in wireless sensor networks," in *Proc. Infocom 2001*, Anchorage, AK, April 2001.

[13] D. Niculescu and B. Nath, "Ad-hoc positioning system," in *IEEE GlobeCom*, Nov. 2001.

[14] C. Savarese, J. Rabaey, and K. Langendoen, "Robust positioning algorithm for distributed ad-hoc wireless sensor networks," in *USENIX Technical Annual Conf.*, Monterey, CA, June 2002.

[15] S. Capkun, M. Hamdi, and J. Hubaux, "GPS-free positioning in mobile ad-hoc networks," in *34th Hawaii Int'l Conf. on System Sciences (HICSS-34)*, Maui, Hawaii, Jan. 2001, pp. 3481–3490.

[16] C. Savarese, J. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks," in *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Salt Lake City, UT, May 2001, pp. 2037–2040.