ADHOC 109
17 September 2004; Disk Used
ARTICLE IN PRESS
No. of Pages 14, DTD = 5.0.1

ELSEVIER
Available online at www.sciencedirect.com
SCIENCE DIRECT®
Ad Hoc Networks xxx (2004) xxx–xxx
Ad Hoc Networks
www.elsevier.com/locate/adhoc

Positioning using local maps ☆

Yi Shang [a,*], Wheeler Ruml [b], Markus P.J. Fromherz [b]

[a] Department of Computer Science and Computer Engineering, University of Missouri-Columbia, 201 EBW, Columbia, MO 65211, USA
[b] Palo Alto Research Center, Palo Alto, CA 94304, USA

Received 9 February 2004; received in revised form 20 June 2004; accepted 20 June 2004

Abstract

It is often useful to know the positions of nodes in a network. However, in a large network it is impractical to build a single global map. In this paper, we present a new approach for distributed localization called Positioning using Local Maps (PLM). Given a path between a starting node and a remote node we wish to localize, the nodes along the path each compute a map of their local neighborhood. Adjacent nodes then align their maps, and the relative position of the remote node can then be determined in the coordinate system of the starting node. Nodes with known positions can easily be incorporated to determine absolute coordinates. We instantiate the PLM framework using the previously proposed MDS-MAP(P) algorithm to generate the local maps. Through simulation experiments, we compare the resulting algorithm, MDS-MAP(D), with existing distributed methods and show improved performance on both uniform and irregular topologies.

Keywords: Localization; Relative position estimation; Multilateration; Multidimensional scaling

1. Introduction

Future wireless sensor networks may involve a large number of sensor nodes densely deployed over physical space. Many applications require knowing the positions of the nodes, sometimes their relative positions and sometimes even their absolute positions. Nodes could be equipped with a global positioning system (GPS) to provide them with their absolute positions, but this is currently a costly solution. With a network of thousands of nodes, it is unlikely that the position of each node can be pre-determined.

Localization has been a topic of active research in sensor networks in recent years [1]. Most existing methods are for absolute positioning, i.e., find-

☆ This work was partially supported by DARPA under contract F33615-01-C-1904.
* Corresponding author. Tel.: +1 573 884 7794; fax: +1 573 882 8318.
E-mail addresses: shangy@missouri.edu (Y. Shang), ruml@parc.com (W. Ruml), fromherz@parc.com (M.P.J. Fromherz).

1570-8705/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2004.06.001

ing the absolute positions of nodes in a global coordinate system. Techniques that use local distance information include the convex constraint satisfaction method [2], triangulation or multilateration methods [3,4], and collaborative multilateration [5]. These techniques require starting with anchor nodes with known positions.

Less work has been done on relative positioning without anchor nodes [6,7]. Relative locations are useful for many basic functions of sensor networks. Examples include Greedy Perimeter Stateless Routing (GPSR) [8], Geographic and Energy Aware Routing (GEAR) [9], and Information Driven Sensor Query (IDSQ) [10]. Application scenarios include answering queries such as: "Where does the loud noise come from?" or "In what direction is that vehicle on my left moving?" Answering such queries requires knowing the relative locations of sensors close to the target.

In this paper, we present a new approach to distributed localization called Positioning using Local Maps (PLM). The method estimates node locations based on local maps, i.e., positions of neighbor nodes in the relative coordinate systems of individual nodes. It can estimate the relative positions of nodes multiple hops away when there are no anchor nodes with known positions. When there are sufficient anchor nodes, e.g., 3 or more for 2-D space, the method can then determine the absolute positions of individual nodes in a distributed fashion.

After presenting the PLM framework, we will instantiate it using a particular method to compute the local maps called MDS-MAP(P). Through simulation, we compare the resulting algorithm, which we call MDS-MAP(D), with existing methods and demonstrate that it provides improved localization results on both regular and irregular network topologies.

2. Related work

In this paper, we focus on localization using connectivity information or local distance measures between neighboring nodes. Several techniques have previously been proposed for this setting. The GPS-less system by Bulusu et al. [11] 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 method needs a high beacon density to work well. Doherty's [2] convex constraint satisfaction method formulates the localization problem as a feasibility problem with convex constraints. It is a centralized method and needs well placed anchor nodes, preferably on the outer boundary, to work well.

Several distributed localization methods have been proposed based on triangulation or multilateration. The APS by Niculescu and Nath [3] is a typical example. The method is called *DV-Hop* when only connectivity information is used and *DV-Distance* when distance measures between neighboring nodes are used. *DV-Euclidean* is another method that uses the local geometry of the nodes [3]. Its performance rapidly degrades as range errors increase. For instance, it performs poorly when the range error is over 2% [12]. The Hop-TERRAIN method by Savarese et al. [4] is similar to APS, but with an additional refinement step. Savvides et al. [5] proposed a collaborative multilateration method that needs more anchors than the other methods to work well.

The techniques discussed above are for absolution positioning and need anchors to start with. The self-positioning algorithm (SPA) [6] has been proposed for relative positioning. This approach is capable of determining the exact relative locations of nodes, but only in the absence of range errors. When there are range errors, its performance rapidly degrades.

MDS-MAP [7,13] is a localization method based on multidimensional scaling which can often generate good relative maps. Multidimensional scaling (MDS) is a set of methods widely used for the analysis of similarity or dissimilarity of a set of objects and discover the spatial structures in the data [14]. MDS methods start with one or more distance matrices (or similarity matrices) and 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. There are several varieties of MDS and we focus on the classical MDS [15]. In classical MDS, proximities are treated as distances in an Euclidean space and optimal analytical solutions are derived from the proximity matrix efficiently through singular value decomposition. The main advantage of using MDS for node localization is that it can generate relatively more accurate position estimation even based on limited and error-prone distance information.

Compared to multilateration-based methods, MDS-MAP uses connectivity or local distance measures between unknown nodes together with those between unknown nodes and anchors, and thus can obtain better results. Compared to SPA, the MDS-MAP method is much more robust to range errors, and it can work with only connectivity information. MDS-MAP(P) [13] is a variant of MDS-MAP with improved performance on irregular topologies. MDS-MAP(P) first computes local maps of individual nodes and then merges them to form a relative map of the whole network. When there are enough anchors, the relative map is transformed to an absolute map. Although the local maps are computed in a distributed fashion and the merging can be done either sequentially or in parallel, a global map is obtained and the transformation of the global map is computed at a central location.

3. Distributed localization: the PLM approach

Rather than trying to build a single global map, in distributed localization we wish to estimate the positions of only certain nodes of interest. Positioning using Local Maps (PLM) uses only local maps along a path between two nodes to estimate their relative positions or the absolute position of a remote node of interest. This more controlled localization scheme is useful in a wide variety of scenarios. For example, if a node in a sensor network detects a target, the root node will care only about the positions of the nodes in vicinity of the target. Path-based localization provides enough information to answer queries such as "Where does the loud noise come from?" or "Tell me in what direction that vehicle is moving".

Specifically, to find the relative position of a remote (target) node in the coordinate system of a center (starting) node from a given communication path, PLM has the following three phases:

1. Build local maps.
   Each node on the path computes its local map. Various methods can be used to compute local maps, such as MDS-MAP, SPA, or semidefinite programming [16]. The local maps of individual nodes only need to be computed once, if the sensor nodes are static, and can be reused for localizing other nodes later. The communication path between two nodes can be discovered by various means, such as limited flooding or constraint-based routing.
2. Compute alignments of adjacent local maps.
   The local maps have different coordinate systems. Each pair of adjacent nodes on the path find the common nodes between their local maps and compute the parameters of the optimal linear transformation to match the common nodes.
3. Determine the position of the remote node in the coordinate system of the center node.
   Along the path from the remote node to the center node, a sequence of linear transformations (computed in Phase 2) are applied to the position of the remote node to obtain its relative position in the coordinate system of the center node.

In the alignment phase of PLM, the optimal linear transformation (minimizing conformation errors) is computed to transform the coordinates of the common nodes in one map to those in the other map. The transformation includes translation, reflection, orthogonal rotation, and scaling. Fig. 1 shows an example of computing the transformation of two local maps. The two maps are constructed by MDS-MAP using only connectivity information.

If instead one wishes to estimate the absolute positions of a node using anchor nodes with known positions, the PLM approach is as follows:

1. Each anchor broadcasts its position throughout the network. At each unknown node, the relative positions of each anchor in its local coordi-

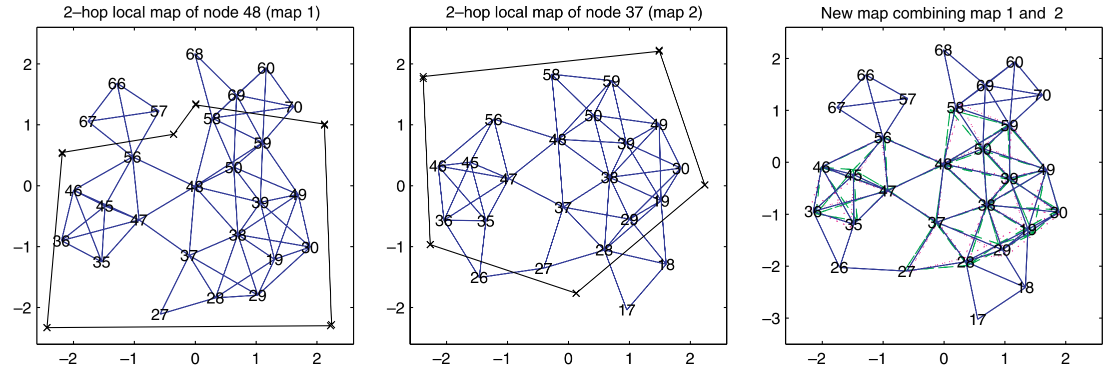

Fig. 1. An example of aligning two local maps based on their common nodes. Nodes inside the boxes in the first two diagrams are common nodes.

nate systems are computed based on a path, e.g., a shortest path, between each anchor and the unknown node.

2. At each unknown node, an optimal linear transformation that maps all anchors from their relative positions to their absolute positions is computed.
3. The absolute position of the unknown node is computed by applying the transformation to its relative position.
4. For each unknown node, its computed position is refined using the computed positions of its neighbors. It is an iterative refinement process. By fixing its neighbors' positions, a least squares minimization problem is solved to find the new position of the unknown node.

An alternative to Steps 2 and 3 is to calculate the distance of each anchor to the unknown node and then apply multilateration to determine the unknown's position based on the distance information and the anchors' absolute positions. Empirically, this alternative does not work as well. It inherits the deficiencies of multilateration on irregular networks, such as the C-shape networks shown later in Section 5.

4. An instantiation of PLM: MDS-MAP(D)

PLM is a general approach to distributed localization. In order to test its effectiveness, it must be instantiated with a particular choice of method for computing the local maps. MDS-MAP(D) implements PLM using the MDS-MAP(P) algorithm to construct local maps. Let $x = (x_i,\ i = 1, \ldots, N)$ represent the estimated coordinates of the N points (nodes); $d_{ij} = \|x_i - x_j\|_2$, the 2-norm of the difference of two points i and j, be their Euclidean distance; and p_{ij} be the empirically measured proximity of nodes i and j. If nodes i and j are within the radio range of each other, then p_{ij} is the distance measure between them if it is available and $p_{ij} = 1$ otherwise. Initially, p_{ij} does not exist for nodes i and j that are outside the radio range of each other.

The localization problem based on proximity information is finding x values such that $d_{ij} = p_{ij}$. When the proximity p_{ij} is just the connectivity or inaccurate local distance measurement, 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 error in the approximate distances between the nodes:

$$\min_x \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij} - p_{ij})^2 \text{ for all available } p_{ij} \tag{1}$$

This optimization problem is non-convex with many local minima. Traditional local optimization techniques, such as the Levenberg–Marquardt method, require a good initial candidate solution in order to return acceptable final results. Randomly generated initial location estimates usually lead to poor final solutions. Global search methods such as simulated annealing or genetic algorithms are generally too slow for practical applications. The contribution of the MDS-MAP method [7] is to use efficient MDS techniques to generate good initial points for the non-linear optimization problem.

In MDS-MAP(D), the local maps are constructed using distance information within a certain range, specified by a mapping range parameter 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 accuracy of the local maps. R_{lm} usually takes values 1, 2, or 3. The case of $R_{\mathrm{lm}} = 1$ only uses information among 1-hop neighbors and has the smallest computation and communication costs. Its result may be good for relatively regular network configurations, but usually is not good for randomly placed nodes. The result of $R_{\mathrm{lm}} = 3$ is better at the expense of higher computation and communication costs. We find that $R_{\mathrm{lm}} = 2$ usually provides a good trade-off.

Each node computes its local map using MDS-MAP through the following steps (see [17] for details): (a) find the shortest paths between all pairs of nodes in the local mapping range R_{lm}. The shortest path distances are used to construct a matrix of estimated inter-point distances for input to the MDS procedure. MDS is a well-known data analysis technique that estimates coordinates for a set of points, given the inter-point distances; (b) apply MDS to the distance matrix and construct a 2-D (or 3-D) local map. The classical formulation of MDS has an analytical solution that is quick to compute. Of course, because MDS is given only relative distances, the resulting estimated coordinates lie at an arbitrary rotation, reflection, and translation from the absolute coordinate system; and (c) minimize least squares error. Using the MDS solution as the initial point, we solve a general version of Eq. (1), i.e., performing least squares minimization (LSM) to encourage the distances between neighbor nodes to match the measured ones.

The objective function used in the LSM of Step (c) not only includes information between 1-hop neighbors, but may optionally include information between multi-hop neighbors, although these distances can be weighted less. We use a refinement range R_{ref}, defined in terms of hops, to specify what information is considered. $R_{\mathrm{ref}} = 1$ means only distance measures between 1-hop neighbors are used; $R_{\mathrm{ref}} = 2$ means estimated distances between nodes up to two hops away are used; and so on. For two nodes that are more than 1-hop apart, we use their shortest path distance. Different values of R_{ref} offer trade-offs between computational cost and solution quality. Thus, the objective function is as follows:

$$\min_x \sum_{i,j} w_{ij}(d_{ij} - p_{ij})^2 \text{ for all provided } p_{ij} \tag{2}$$

where w_{ij} are the weights. If $w_{ij} = 1$ for all pairs of nodes, then we minimize the sum of squared errors. If $w_{ij} = 1/p_{ij}^2$, then we minimize the sum of squared relative errors. Empirically, we have found that $R_{\mathrm{ref}} = 1$ is better than $R_{\mathrm{ref}} = 2$ when range errors are close to 0. When range errors are over 5%, $R_{\mathrm{ref}} = 2$ is better, and minimizing the relative error also improves the results slightly. Thus, in the experiments reported below we use $R_{\mathrm{ref}} = 2$ and $w_{ij} = 1/p_{ij}^2$.

The communication cost of MDS-MAP(D) is dominated by building local maps. For a node z to compute its local map, each node within the mapping range needs to send its connectivity information or local distance measures to z. The cost of computing the alignment of two local maps is small, where one local map needs to be sent from one node to the other node. The communication cost of transforming the position of the target node to the local coordinate system of the starting node is also small. Only the coordinates of the target node need to be sent along the path to the starting node. At each intermediate node, the coordinates are transformed using the local transformation matrix.

When computing the absolute positions of nodes using anchors, MDS-MAP(D) uses a new iterative method based on a mass-spring model in the last step to refine the node positions. In this procedure, mass points are connected to each other by springs of certain lengths. In the lowest energy state, the combined force of the springs is the smallest. In the refinement method of MDS-MAP(D), the connectivity or the distances between neighbor nodes represent the spring lengths. Imagining the positions of its neighbors to be static, each node changes its position to reduce the combined force that its neighbors put on it. It is a simple iterative process.

Specifically, assuming the current position of the node is z, the positions of its neighbors are y_i, $i = 1, \ldots, k$, and the distance estimates between z and y_i is q_i, z is updated in each iteration according to the average combined force as follows:

$$z = z + \frac{1}{k}\sum_{i=1}^{k}(\|y_i - z\|_2 - q_i)\frac{y_i - z}{\|y_i - z\|_2} \tag{3}$$

The number of iterations is set to 20 in our experiments.

This method is similar to the resolution of forces technique [18]. Typical multilateration techniques, such as APS, minimize the sum of squared errors of the computed distances and measured distances, whereas resolution of forces minimizes the combined absolute error. Minimizing squared error is optimal if the distance errors are normally distributed, and minimizing absolute error is better when the errors have a Laplace distribution. In [18], experimental data show that real distance measurement errors are more likely to be Laplace distributions, where the resolution of forces method performs well.

Our technique is simple, efficient, and gets good results. It differs from resolution of forces in two aspects. One is that we refine the position of a node using neighbor positions, whereas resolution of forces estimates the position of an unknown node using anchor positions and distance estimation to anchors. Our method can also be applied in the same situation. Secondly, the update rule is different. In resolution of forces, the new position of a node is the current position plus the resultant of all forces. It is not an iterative process. When the initial position is far away from the lowest energy point, its update rule can generate large position changes and may not converge to the lowest energy state, even if it runs iteratively. We found that it did not work well in our experiments and that our update rule is more robust.

5. Experimental evaluation

To evaluate its performance, we tested MDS-MAP(D) on both uniform and irregular topologies. Fig. 2 shows an example of an uniform topology with 200 nodes randomly placed inside a 10×10 square area and an example of an irregular topology with 70 nodes placed near the grid points

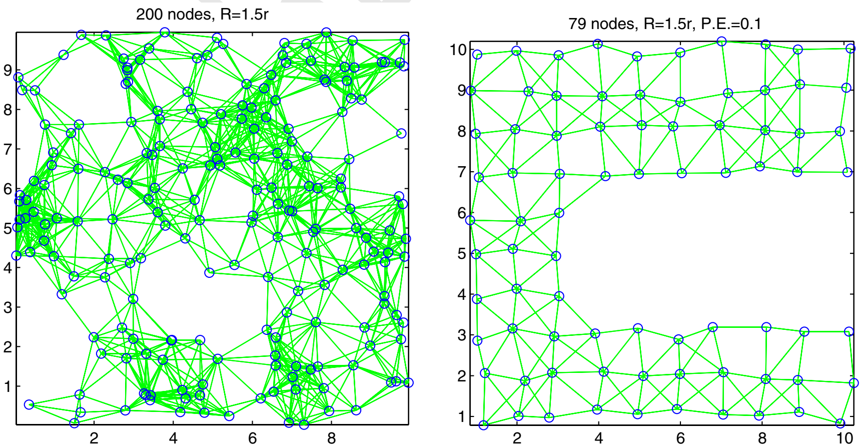

Fig. 2. Two example problems: (a) *random uniform* placement—200 nodes are randomly placed in a $10r \times 10r$ square; and (b) *regular C-shaped* placement—79 nodes are placed on a C-shape grid with $10\%r$ placement errors. The radio range is $1.5r$, where the placement unit length $r = 1$. The connectivity levels are 12.1 and 5.1, respectively.

of a C-shape grid. In the figures, circles represent nodes and edges represent connections between nodes that are within communication range of each other. The connectivity (average number of neighbors) is controlled by the radio range R.

To model the placement errors in the grid placements, we added Gaussian noise to the coordinates of the grid points. For example, 10%r placement error means the coordinates of nodes are the coordinates of corresponding grid points plus random variables drawn from Gaussian distribution $N(0, 10\%r)$. Similarly, the distance measures are modeled as the true distances plus Gaussian noise. For example, if the true distance is d^* and the range error is e_r, then the measured distance is $d^*(1+y)$, where y is drawn from $N(0, e_r)$.

In applying MDS-MAP(D) to the two examples, each unknown node independently computes its position estimation based on the anchor positions and the distance information. The results of all unknown nodes are shown in Figs. 3 and 4.

A variant of APS was used in the experiments. A linear system of multilateration is first solved and the solution is used as the initial point in solving a system of quadratic equations. Specifically the method has the following three steps:

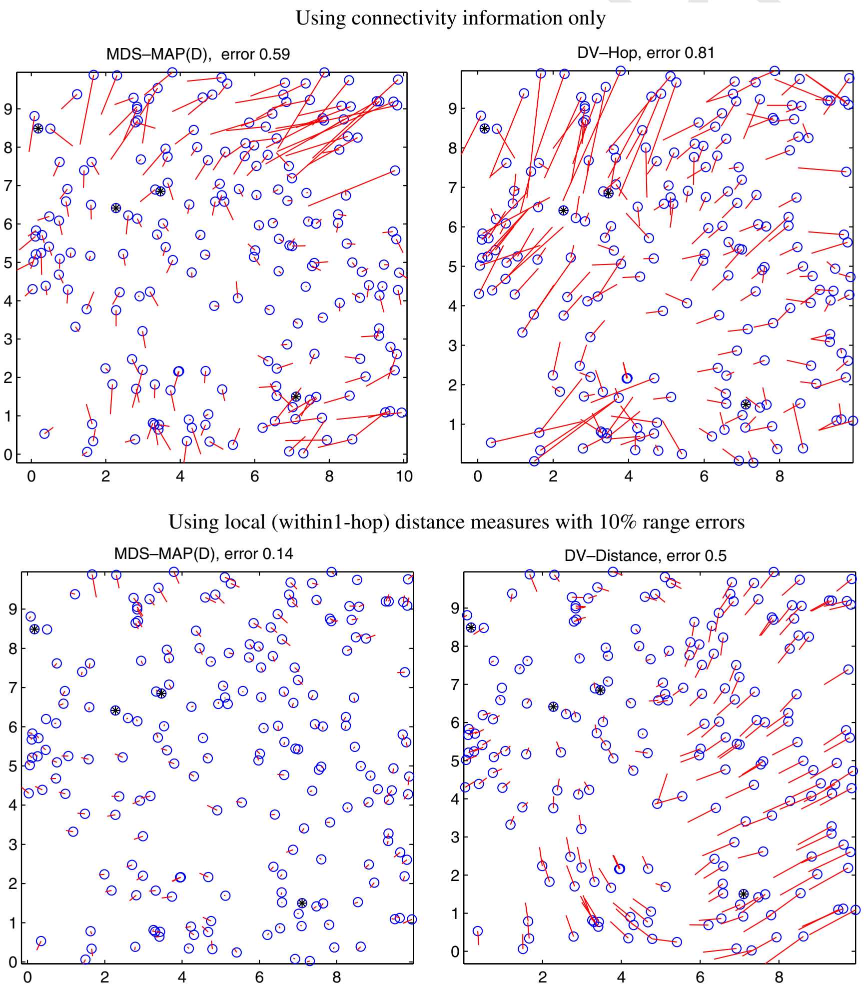

Fig. 3. Comparison of MDS-MAP(D) and APS on the *random uniform* example.

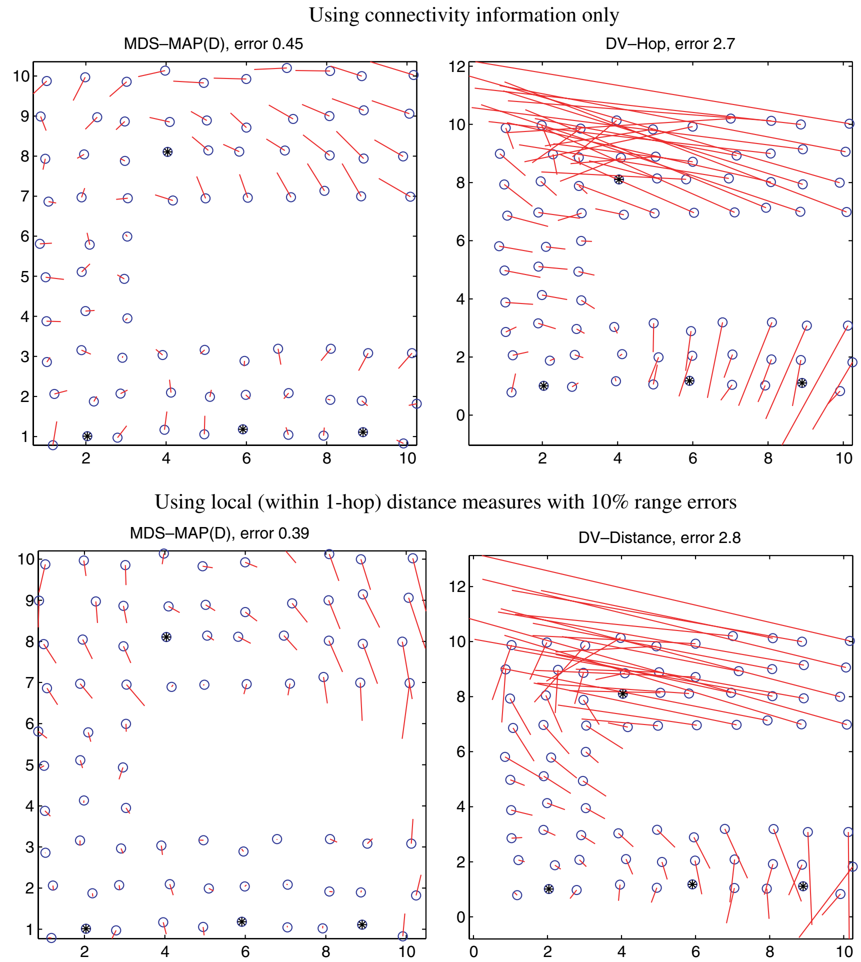

Fig. 4. Comparison of MDS-MAP(D) and APS on the *C-shape grid* example.

1. Each anchor k receives the positions (a_j,b_j), $j = 1,\ldots,m$, of all anchors and also computes the shortest path distance p_{kj} to each of the anchors.
2. Each anchor k computes its distance correction value, $c_k = \frac{\sum_{j=1}^{m} d_{kj}}{\sum_{j=1}^{m} p_{kj}}$, where d_{kj} is the Euclidean distance between two anchors k and j.
3. For each unknown node i, determine its position. First, we solve the following system of linear equations of two variables, x_i and y_i, by least-squares minimization. (Anchor 1 is used to linearize the equations.)

$$2(a_1 - a_j)x_i + 2(b_1 - b_j)y_i + a_j^2 + b_j^2 - a_1^2 - b_1^2 = (c_j p_{ij})^2 - (c_1 p_{i1})^2 \quad \text{for } j = 2,\ldots,m \tag{4}$$

Then, we use the solution as the initial point to solve the following system of quadratic equations by least-squares minimization.

$$(x_i - a_j)^2 + (y_i - b_j)^2 = (c_j p_{ij})^2 \quad \text{for } j = 1,\cdots,m \tag{5}$$

In our experiments, this approach works better than just solving the linear system [4,12], especially for networks with lower connectivity and fewer anchor nodes. In the rest of the paper, the APS mentioned in the experimental results refers to our variant of APS. APS using connectivity information is also referred to as DV-Hop, and APS using local distance measures as DV-Distance.

5.1. Examples

To give an intuitive feel for our results, we first present single specific example runs before reporting our more thorough averaged comparisons. Fig. 3 compares the results of MDS-MAP(D) and APS, both using the refinement method in Eq. (3), on the random uniform example. Four random anchor nodes (denoted by asterisks) are used. 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.

Using *connectivity* information only, the average error of MDS-MAP(D) is 0.59, which is 73% of the average error of DV-Hop (0.81). The example shows that MDS-MAP(D) localizes the unknown nodes within the convex hull of the anchors quite accurately, but does poorly on the group of nodes in the upper-right corner. Because the four anchors are not spread out and the distance estimation based on connectivity is very crude, APS could not generate good result overall. However, it does better on the group of nodes in the upper-right corner.

Using local distance measures with 10% range error, the average error of MDS-MAP(D) is very small, 0.14 (or 9%R since the radio range R is 1.5), which is only 28% of the error of DV-Distance (0.5). Note that the MDS-MAP(D) positioning error is comparable to the 10% range error. Now MDS-MAP(D) estimates the positions of the group of nodes in the upper-right corner very well due to more accurate local maps. Regarding APS, it does a better job on the nodes surrounded by the anchors than the ones in the right half of the network.

Fig. 4 compares the results of MDS-MAP(D) and APS on the C-shape grid placement example. It is interesting to compare the difference of the two methods on the group of nodes in the upper-right corner. APS fails badly on them because the shortest path distance estimation used in multilateration is far from the actual distance for most anchors. In addition, local distance measures do not help APS. MDS-MAP(D) is more tolerant to the placement of anchors and does a better job on these nodes. MDS-MAP(D) does an even better job on the nodes surrounded by the anchors. Overall, the average errors of MDS-MAP(D) are 0.45 and 0.39, respectively, for the cases of using *connectivity* information and using local distance measures with 10% range errors, much better than those of APS.

5.2. Performance comparison of MDS-MAP(D), APS, and MDS-MAP(P)

In the experiments, we compare the performance of MDS-MAP(D) and APS on two types of random networks, and use MDS-MAP(P) as a baseline. All three methods are run with or without the mass-spring refinement technique in Eq. (3). The random uniform networks are generated by placing 200 nodes randomly inside a 10×10 square area, and the random irregular networks are generated by placing 200 nodes randomly inside a C-shape area within a 10×10 square (similar to the C-shape of the grid example in Fig. 4. Fifty random trials were done for each data point.

Fig. 5 shows the results on the random uniform networks, using connectivity information or local distance measurement with 10% range error. The average position estimation error of 50 random trials for each case is plotted against the connectivity level. Five or ten random anchors are used.

The results show that when using connectivity information only, MDS-MAP(D) is worse than APS when the connectivity is below 10 and is much better when the connectivity is over 12. The reason is that when the connectivity is low, the local maps can be inaccurate and their alignment can be bad. When this happens, it is better not to use the local maps. With higher connectivity, the accurate local maps can provide much more information than the distance estimation technique used in APS. Both MDS-MAP(D) and APS get better results using 10 anchors than using five anchors.

The mass-spring refinement technique provides consistent, significant improvement on APS solutions for networks with various connectivity levels.

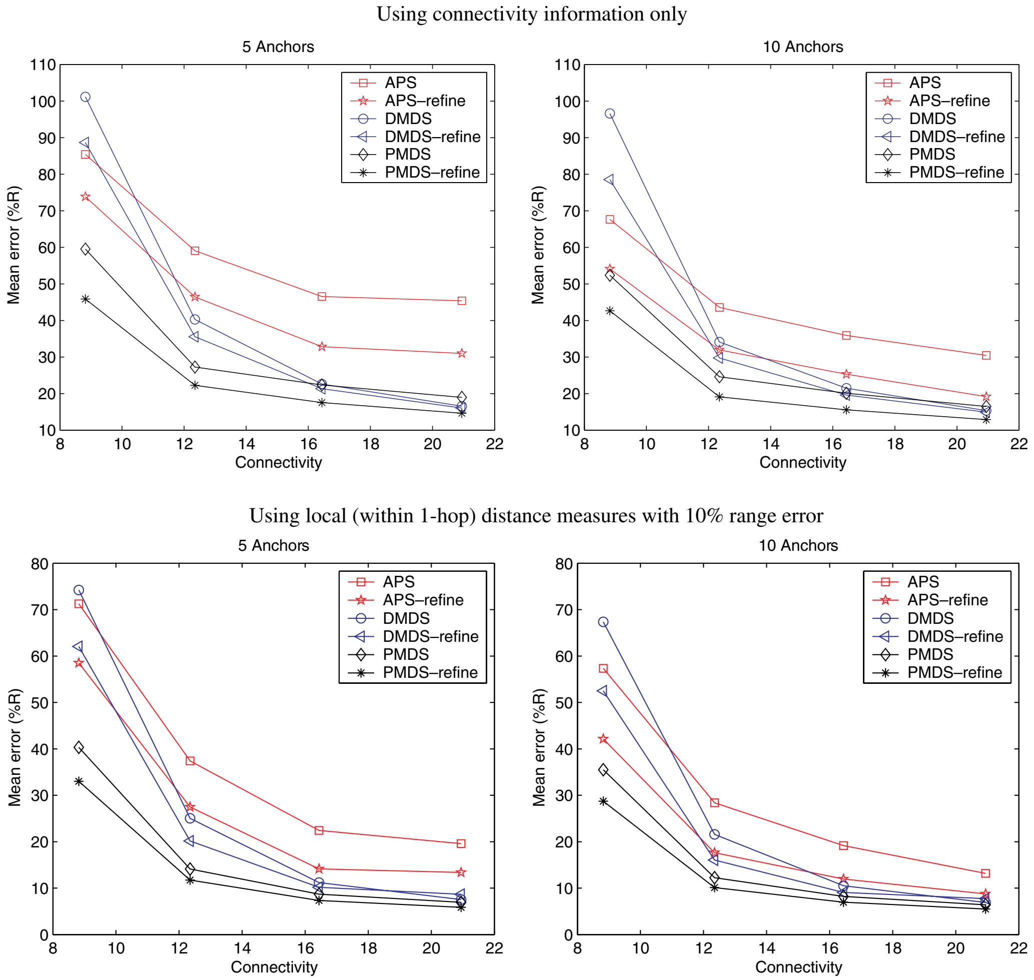

Fig. 5. Performance comparison of MDS-MAP(D), APS, and MDS-MAP(P) on the random uniform networks with 200 nodes using 5 or 10 random anchors.

The improvement of refinement for MDS-MAP(D) is also significant for low connectivity networks, but decreases as the connectivity increases due to the more accurate local maps computed with higher connectivity.

Compared to MDS-MAP(P), MDS-MAP(D) is much worse on low connectivity networks, but is comparable on higher connectivity ones. Both achieve good results of about 18%R (R is the radio range) positioning errors when the connectivity is from 16 to 21, using connectivity and five anchors. MDS-MAP(P) performs consistently better than APS. The positioning error of MDS-MAP(P) is about 60% of that of APS on lower connectivity networks, and goes down to about 50% on higher connectivity networks.

The results are similar when using local distance measures with 10% range error. All methods achieve better results than when using connectivity only. Positioning errors are reduced as the connectivity increases. Using five anchors, the positioning errors of MDS-MAP(D) and MDS-MAP(P) with refinement approach 6%R, whereas the error of APS with refinement approaches 12%R.

All three methods have the same coverage (the number of nodes being localized) since they localize all the nodes in the largest connected network in each test case, which is more than other existing methods such as Hop-TERRAIN and the Euclidean method.

Next, we compare their performance on the irregular topology. Fig. 6 shows the results on the random irregular networks, using connectivity information or local distance measurement with 10% range error. The results of using five random anchors are reported.

The results show that APS performs poorly no matter whether it uses connectivity or local distance measures. The reason is that the anchor distance estimation used by APS are very inaccurate for multi-hop nodes in this type of network. MDS-MAP(D) performs equally bad when the connectivity is low due to inaccurate local maps, but is much better when the connectivity increases. MDS-MAP(D) gets better results using the local distance measurement than just using connectivity. Its positioning errors approach 20%R and 15%R when using connectivity and local distance measurement, respectively, for higher connectivity levels.

Compared to MDS-MAP(P), the performance difference of MDS-MAP(D) is much smaller than on the uniform problems. MDS-MAP(P) does not work well either when the connectivity is low. They perform similarly when the connectivity is high because they use almost the same local maps and the patching process in MDS-MAP(P) becomes similar to the alignment processing in MDS-MAP(D).

Finally, we compare their performance for different range errors. Fig. 7 shows the results on the random uniform networks using connectivity or local distance measurement with 1% or 20% range error and five anchors.

In the 1% range error case, MDS-MAP(D) is better than APS for connectivity at 8.8, with or without refinement, which is different from the 10% or 20% range error case. This means that MDS-MAP(D) has more advantage with more accurate distance measures due to the better local maps that can be constructed. Generally, the relative performance of the three methods is similar to the 10% range error case shown in Fig. 5.

In the 20% range error case, MDS-MAP(D) becomes even worse than APS for connectivity at 8.8. However, MDS-MAP(D) is still better when the connectivity is over 12, and the relative performance difference is similar to the 10% range error case. An interesting observation is that at connectivity level 21, MDS-MAP(D) without refinement has noticeably better solutions than MDS-MAP(D) with refinement. This suggests that when some local distance measurement is very inaccurate, the refinement based on distances to

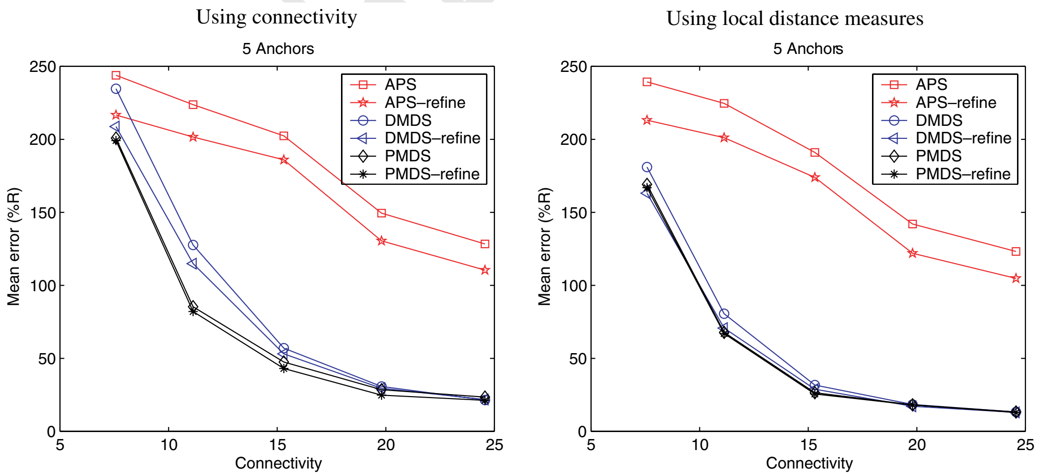

Fig. 6. Performance comparison of MDS-MAP(D), APS, and MDS-MAP(P) on the random irregular networks with 200 nodes using five random anchors.

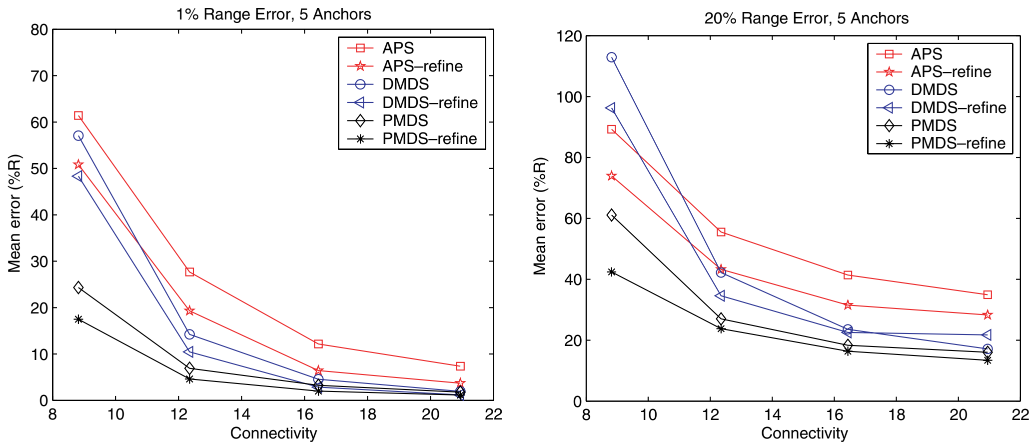

Fig. 7. Results of MDS-MAP(D), APS, and MDS-MAP(P) on the random uniform networks using local distance measures with 1% or 20% range error and five anchors.

1-hop neighbors may not be helpful when the connectivity is high.

In addition to random networks, we also tested MDS-MAP(D) on grid networks similar to the C-shape grid network in Fig. 2. Because of the regular structures of these type of networks, MDS-MAP(D) is able to build more accurate local maps for networks with low connectivity. Overall, MDS-MAP(D) performs much better on grid networks than on random networks and can achieve comparable positioning errors at much lower connectivity levels.

5.3. Relative position estimation

In this section, we present the results of MDS-MAP(D) in estimating the position of a remote (target) node relative to a center (starting) node. In each trial, a random network is first generated. Then a random center and a random target (remote) node a certain number of hops away are selected. By aligning the local map of the center node using the absolute positions of the nodes in the local map, the estimated relative position of the remote (target) nodes is compared with its true absolute position. The difference is the position

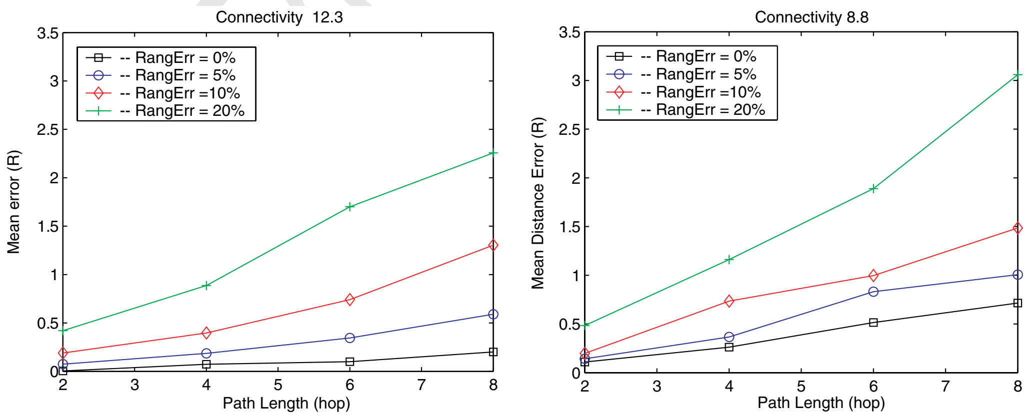

Fig. 8. MDS-MAP(D) results on *random uniform* networks with connectivity 12.3 and 8.8 using local distance measures with range errors from 0% to 20%.

estimation error. Again, 50 random trials were conducted for each data point.

Fig. 8 shows the result of MDS-MAP(D) on the 200-node random uniform networks using local distance measurements with range errors from 0% to 20%. The average positioning errors of the target (remote) nodes are plotted against the length of the communication path. The radio ranges are 1.5 and 1.25, respectively. As the range error increases, the positioning accuracy degrades, especially quickly for longer paths and networks with low connectivity. The algorithm performs reasonably well when the range error is less than 10%. In addition, we also tried the method on grid networks and obtained better results.

6. Conclusions

In this paper, we presented a new distributed localization approach called PLM, based on combining local maps along a path between two network nodes. We instantiated the approach using the MDS-MAP(P) algorithm to build the local maps, resulting in the distributed algorithm we call MDS-MAP(D). Each node computes its relative local map at most once. The alignment of one local map with another is also done at most once. Given a sequence of overlapping local maps along a path, a sequence of transformations can be used to compute the position of the remote node in the coordinate system of the center node. If the center node knows the absolute positions of three or more nodes that are in its local map, it can compute its own absolute position. Through simulation, we showed that MDS-MAP(D) performs well on both regular and irregular topologies when there is medium to high connectivity, e.g., more than ten for random networks and six for grid networks, and when the range errors are small, e.g., 10%. The algorithm significantly outperforms existing methods, e.g., APS, on cases with just a few anchor nodes, and especially on irregular networks.

PLM is a general approach and is independent of the way local maps are computed. For example, it can also build local maps by solving the localization problem as semidefinite programming problems, which can generate accurate local maps when range errors are small [16]. An important research direction is to study the properties of different methods for building relative maps and their performance in a local-map based localization framework.

References

[1] J. Hightower, G. Boriello, Location systems for ubiquitous computing, IEEE Computer 34 (8) (2001) 57–66.
[2] L. Doherty, L. El Ghaoui, K. Pister, Convex position estimation in wireless sensor networks, in: Proc. Infocom 2001, Anchorage, AK, April 2001.
[3] D. Niculescu, B. Nath, Ad-hoc positioning system, IEEE GlobeCom, November 2001.
[4] C. Savarese, J. Rabaey, K. Langendoen, Robust positioning algorithm for distributed ad-hoc wireless sensor networks, in: USENIX Technical Annual Conference, Monterey, CA, June 2002.
[5] A. Savvides, H. Park, M. Srivastava, The bits and flops of the n-hop multilateration primitive for node localization problems, in: First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta, GA, September 2002, pp. 112–121.
[6] S. Capkun, M. Hamdi, J.P. Hubaux, GPS-free positioning in mobile ad-hoc networks, in: 34th Hawaii International Conference on System Sciences (HICSS-34), Maui, Hawaii, January 2001, pp. 3481–3490.
[7] Y. Shang, W. Ruml, Y. Zhang, M. Fromherz, Localization from mere connectivity ACM MobiHoc, Annapolis, MD, 2003, pp. 201–212.
[8] B. Karp, H.T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in: Proceedings of the Sixth International Conference on Mobile Computing and Networks (ACM Mobicom), Boston, MA, 2000.
[9] Y. Yu, R. Govindan, D. Estrin, Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks, Technical report UCLA/CSD-TR-01-0023, UCLA Computer Science Department, May 2001.
[10] M. Chu, H. Haussecker, F. Zhao, Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks, International Journal on High Performance Computing Applications (2002).
[11] N. Bulusu, J. Heidemann, D. Estrin, GPS-less low-cost outdoor localization for very small devices, IEEE Personal Communications 7 (5) (2000) 28–34.
[12] K. Langendoen, N. Reijers, Distributed localization in wireless sensor networks: a quantitative comparison, Computer Networks (Elsevier), Special Issue on Wireless Sensor Networks (2003).
[13] Y. Shang, W. Ruml, Improved MDS-based localization, in: IEEE InfoCom, March 2004.

[14] I. Borg, P. Groenen, Modern multidimensional scaling, in: Theory and Applications, Springer-Verlag, New York, 1997.
[15] W.S. Torgeson, Multidimensional scaling of similarity, Psychometrika 30 (1965) 379–393.
[16] P. Biswas, Y. Ye, Semidefinite programming for ad hoc wireless sensor network localization, in: Proceedings of the IPSN'04, Berkeley, CA, April 2004.
[17] Y. Shang, W. Ruml, Y. Zhang, M. Fromherz, Localization from mere connectivity, IEEE Transactions on Parallel and Distributed Systems 15 (10) (2004).
[18] C.D. Whitehouse, The design of calamari: an ad-hoc localization system for sensor networks, Tech. Rep., University of California, Berkeley (MS Thesis), 2002.

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

Wheeler Ruml is a Member of the Research Staff at the Palo Alto Research Center. He received the A.B. and Ph.D. degrees from Harvard University in 1993 and 2002, respectively. 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 AAAI.

Markus Fromherz 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 received his Ph.D. in Computer Science in 1991 from the University of Zurich (Switzerland) and his M.S. in Computer Science in 1987 from ETH Zurich. He has published in the areas of software engineering, transformational program development, model-based computing, on-line scheduling, embedded constraint solving, and design optimization. He has also co-authored more than 30 patents. He is a member of the IEEE.