

LOCALIZATION OF SPARSE SENSOR NETWORKS USING LAYOUT INFORMATION

Sameer Sundresh, YoungMin Kwon, Kirill Mechitov, Wooyoung Kim and Gul Agha

Department of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin Ave., Urbana, IL 61801, USA

{sundresh|ykwon4|mechitov|wooyoung|agha}@cs.uiuc.edu

ABSTRACT

Localization is the process by which sensor networks associate spatial position information with individual sensors' measurements. While manual surveying is sufficient for small-scale prototypes, it is too time-consuming and costly for the large-scale deployments anticipated in the near future. Our experiments with medium-scale outdoor sensor network deployments show that sparsity of ranging measurements is a key factor limiting the accuracy of localization; often, several solutions are equally consistent with the data. Fortunately, layout information can usually be obtained at little extra cost; for example, if it is used to guide the deployment process, or by analyzing a photograph of the network. We have developed an algorithm based on subgraph isomorphism which uses the known layout information in conjunction with ranging measurements to find a family of localization solutions for a sensor network deployment. Although subgraph isomorphism is in general NP-complete, the more specific cases that occur in real-world scenarios are usually tractable. Experiments with a 50-node network show that our algorithm is very efficient in practice.

1. INTRODUCTION

Localization is the process by which sensor networks associate spatial position information with individual sensors' measurements. While manual surveying is sufficient for small-scale prototypes, it is too time-consuming and costly for the large-scale deployments anticipated in the near future. Automated localization algorithms have been developed in an attempt to solve this problem; these localization procedures run in two steps, ranging followed by solving for node positions. A ranging service is used to estimate the distances between sensor nodes in the network; our experiments use the acoustic TDoA ranging technique which we developed in [1]. Once these measurements have been gathered, a localization solver must be run to estimate node positions.

Our experiments with medium-scale outdoor sensor network deployments show that sparsity of ranging measurements is a key factor limiting the accuracy of localization; often, several solutions are equally consistent with the data. For example, Table 2 summarizes the number of ranging measurements obtained in five experiments with a 50-node sensor network. Figure 1 graphically depicts which nodes were able to make reliable ranging measurements to each other. Under these conditions, measurement data is quite sparse (averaging 2.3 measurement neighbors per sensor

node), and existing localization algorithms such as Least Squares Scaling fail to converge (Figure 8).

Fortunately, layout information can usually be obtained at little extra cost; for example, if it is used to guide the deployment process, or by analyzing a photograph of the network. We have developed an algorithm based on subgraph isomorphism which uses known layout information in conjunction with ranging measurements to find a family of localization solutions for a sensor network deployment. Although subgraph isomorphism is in general NP-complete (since it subsumes CLIQUE and HAMILTON CIRCUIT [2]), more specific cases such as bounded-valence graph isomorphism [3] and constant-model subgraph isomorphism [4] can be solved in polynomial time, and there exist several algorithms which have generally been observed to operate efficiently in practice [5, 6, 7].

Once distances have been gathered into a *measurement graph*, our algorithm attempts to match sensor nodes to positions in the known *layout graph* which are consistent with the measurements. Figure 1 shows the offset grid layout which we used for our experiments, along with the measurement graph from one experiment. In this case, the layout graph defines the distances between the discrete possible positions of nodes in the grid. While random or irregular known layouts can be easy to match, a regular grid contains many more symmetries and ambiguities, thus better testing the effectiveness of our algorithm. For example, it is trivial to localize two nodes relative to each other if there is a unique distance between them.

Our experiments with the 50-node network show that while sparse measurement data often have several localization results consistent with the known layout, our algorithm can find them very quickly. This contrasts with previous localization techniques which only attempt to find one solution, either by arbitrarily choosing a "best" result or simply not localizing even slightly underconstrained nodes. By enumerating multiple results, our algorithm reveals the amount of uncertainty about node positions; for example, we may discover that a particular node has three candidate positions. This information can then be used to guide additional measurements to improve results. Alternatively, a higher level service may decide that the results are good enough if the variation across solutions is within application-specific bounds.

2. BACKGROUND

The Global Positioning System (GPS) is by far the most popular standard for electronic outdoor localization [8]. However, GPS

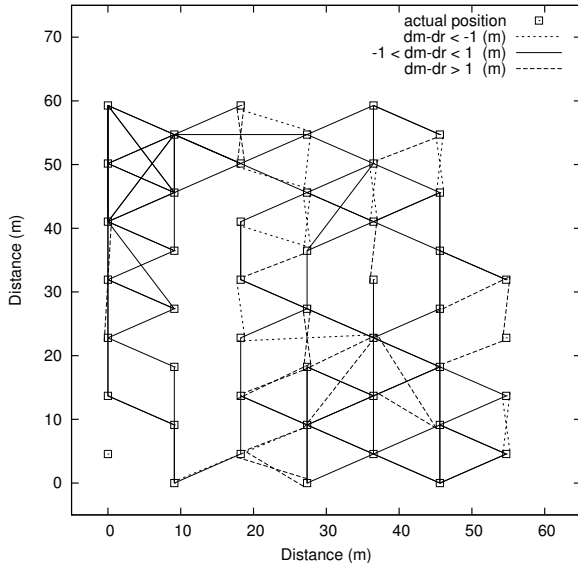


Fig. 1. Node layout and ranging measurements from Experiment 06/14/04-1 (also see Table 2); dr and dm are real and measured distances, respectively. Sensors were deployed on a field with 15-23cm tall grass and somewhat uneven ground.

units are typically either too costly or imprecise for wireless sensor nodes ($\pm 6.3m$ error with 95% confidence when selective availability is turned off [9]). As such, many hybrid methods employ a two-tiered approach in which a set of *anchor* nodes is used to localize non-anchor nodes. Anchor nodes are assumed to know their own locations, while the remaining sensor nodes estimate their distances to anchors and perform multilateration to determine their own locations. These systems primarily differ in how distances to anchor nodes are measured.

The Ad-hoc Positioning System (APS) is a family of distributed localization algorithms based on trilateration [10]. The basic idea is to perform multi-hop propagation of distances to anchors throughout the network, so that every node can trilaterate its position. Four different distance metrics were developed, ranging from minimum hop count and sum of hop lengths—for isotropic networks with uniform node density—to local geometric constructions, which require higher anchor density in order to control error propagation. Another variant of APS relies on sensor nodes able to measure the angle-of-arrival of a signal from an anchor [11].

A related approach to localization is the robust quadrilaterals method of [12]. In this technique, sensor nodes are first localized within certain 4-node complete graphs. These so-called *robust quadrilaterals* must satisfy the constraints that all measurements are (approximately) consistent, and all angles are large enough that a bounded measurement error could not result in a flip or discontinuous flex ambiguity which radically changes the localization results. Robust quadrilaterals which have 3 nodes in common are then iteratively merged into a rigid structure to localize the whole network. Nodes not involved in a robust quadrilateral are simply not localized in order to avoid the possibility of erroneous localization.

Convex optimization has been proposed as the basis for a con-

straint-based localization scheme [13]. In this algorithm, measured data such as RF communication range or angular information from optical devices are used to constrain the feasible node positions. Semidefinite programming (SDP) is then used to find a solution to the localization problem.

Multidimensional Scaling (MDS) has been proposed as the basis for a centralized robust localization algorithm given a set of distance measurements [14]. One problem with this centralized approach is that it requires distances between all pairs of nodes. As a remedy, distributed approaches have been proposed. One approach is to apply a local MDS-MAP for each node along with its hop count-limited neighborhood [15]. Neighborhoods are then incrementally merged into a global coordinate system. Instead of using classical MDS, the local map may be computed by directly minimizing the discrepancies between measurements and distances in the estimated map [16]. Another alternative technique is LSS [17], which seeks a configuration of nodes from a possibly incomplete set of distances between them by minimizing an unweighted error function. The application of this technique to localization was studied in [1].

One common limitation of the above schemes is that only one “best” solution is computed. If positions are underconstrained due to sparse measurements, this choice may be arbitrary – influenced by initial conditions, error propagation, and artifacts of the chosen utility function – or pessimistic, returning no information at all about such a node’s location. Subgraph isomorphism algorithms, on the other hand, have been designed to enumerate some or all solutions.

Ullman’s algorithm is a well-known efficient solution to subgraph isomorphism; given two graphs with p_α and p_β nodes, respectively, it performs a depth-first search on a $p_\alpha \times p_\beta$ bit matrix with pruning and backtracking to enumerate the isomorphisms between one graph and the subgraphs of another [5].

Due to physical measurement range restrictions, localization is a *bounded-degree subgraph isomorphism* problem. If this were not the case, measurement sparsity would not be an issue, and existing localization algorithms such as LSS would be sufficient. Luks’s paper [3], aptly titled “Isomorphism of graphs of bounded valence can be tested in polynomial time,” constructs an intricate group-theoretic algorithm which shows that this related problem is tractable.

A sensor network’s physical layout is often known a priori, either due to a regular structure or because it is used in the deployment of many sensor networks (or even one network that must repeatedly be collected and re-deployed for logistical reasons). Thus, the layout and any associated one-time preprocessing may be considered constant. The constant-model subgraph isomorphism algorithm of [4] uses a decision tree of worst-case size $O(l_v(1 + l_e^2)^M)$ to compute subgraph isomorphisms via decision tree traversal in time $O(M^2 l_e + M l_v)$, where M is the number of vertices in the (fixed) model graph, l_v the number of different vertex labels, and l_e the number of different edge labels. More generally, this result shows that subgraph isomorphism decision procedures with efficient running times can be created given a known layout.

Our localization algorithm is based on VF2, a memory-efficient subgraph isomorphism algorithm based on depth-first search with pruning and backtracking; unlike Ullman’s algorithm, VF2 only maintains a search data structure of size linear in the number of nodes in the graphs [6]. As a result of careful pruning criteria and good cache efficiency, VF2 demonstrates exception-

ally good performance in comparison to other subgraph isomorphism algorithms [7].

3. DEFINITIONS

We start with the standard definition of a *labeled undirected graph* to describe the distances between sensor nodes.

Definition 1 (Labeled Undirected Graph). A labeled undirected graph $LG_{VL,EL} = (V, E, L_V, L_E)$ consists of a set V of vertices, a relation $E \subseteq \{\{u, w\} : u, w \in V\}$, and a pair of functions which assign labels to the vertices and edges, respectively:

$$\begin{aligned} L_V : V &\rightarrow VL \\ L_E : E &\rightarrow EL \end{aligned}$$

A measurement graph consists of the set of sensor nodes, labeled by their ids, together with edges representing the measured distances between sensors.

Definition 2 (Measurement Graph). A ranging measurement graph MG is a labeled undirected graph $LG_{\mathbb{Z}^+, \mathbb{R}^+}$ subject to the following constraints:

1. Each node is labeled by a unique positive integer id.
2. Each edge is labeled with the approximate distance between its endpoints.

In contrast to the measurement graph, the layout information graph consists of the *positions* of sensors – generally not labeled by node ids – together with edges representing the actual distances between different positions. If *anchor nodes* are employed, their positions in the layout graph are labeled by their ids. Since ranging hardware characteristics limit the maximum edge length in the measurement graph, only positions within some threshold distance M of each other need be connected by an edge; this is useful in practice to limit the size of the layout graph data structure. The layout graph can easily be generated from a set of position coordinates in \mathbb{R}^2 or \mathbb{R}^3 ; furthermore, it may be defined procedurally rather than explicitly, as discussed in Section 4.

Definition 3 (Layout Information Graph). The layout information graph for a particular deployment scenario LI is a labeled undirected graph LG_{VL, \mathbb{R}^+} where the set of vertex labels $VL = \mathbb{Z}^+ \cup \{*\}$, subject to the constraints:

1. Each node is labeled by a unique positive integer id or the wildcard, $*$.
2. Each pair of nodes within distance M of each other is connected by an edge.
3. Each edge is labeled with the real distance between its endpoints.

Some sort of hardware-software ranging service is required to estimate distances between sensor nodes. Since measurements are not exact, a comparison operator is necessary to determine whether a particular measurement is likely to represent a particular distance in the known layout.

Definition 4 (Ranging Service). We assume that a ranging service R provides operations for ranging measurement and comparison of measured values against actual distances:

$$\text{measure}_R : \rightarrow M, \text{ i.e. a procedure which coordinates ranging and creates a measurement graph.}$$

$$\approx_R : \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \{\text{true}, \text{false}\}$$

In our experiments, we used an error threshold T , e.g. 1m, to define $x \approx_R y \Leftrightarrow |x - y| < T$.

Given the above, we can now define what exactly we mean by a subgraph isomorphism, and how localization is a special case of the subgraph isomorphism problem.

Definition 5 (Subgraph Isomorphism). Given a pair of labeled undirected graphs $A_{VL,EL}$ and $B_{VL,EL}$, where A is the model graph and B the proposed subgraph, and a pair of relations \approx_V and \approx_E on vertex and edge labels (see below), respectively, a subgraph isomorphism is a 1-1 mapping $SI : V_B \rightarrow V_A$ subject to the constraints (note: $L_{V/A}$ means A 's L_V , etc.):

1. If $c \in V_B$ then $L_{V/B}(c) \approx_V L_{V/A}(SI(c))$, i.e. corresponding vertices have equivalent labels.
2. If $\{c, d\} \in E_B$ then $\{SI(c), SI(d)\} \in E_A$, i.e. each edge in B has a corresponding edge in A .
3. If $\{c, d\} \in E_B$ then $L_{E/B}(\{c, d\}) \approx_E L_{E/A}(\{SI(c), SI(d)\})$, i.e. corresponding edges have equivalent labels.

The subgraph isomorphism problem is: given A and B , find SI ; note that depending on the data, there may be 0, 1 or more possible solutions.

Definition 6 (Localization as Subgraph Isomorphism). Given a ranging service R , a layout graph LI , and the measurement graph MG produced by measure_R , the localization problem is a special instance of the subgraph isomorphism problem where:

$$\begin{array}{lcl} A & \text{is} & LI \\ B & \text{is} & MG \\ x \approx_V y & \text{iff} & (x = *) \vee (y = *) \vee (x =_{\mathbb{Z}^+} y) \\ \approx_E & \text{is} & \approx_R \end{array}$$

We will now construct a localization algorithm to satisfy the above problem statement.

4. LOCALIZATION ALGORITHM

4.1. Fundamental Behavior

Our localization algorithm is based on the VF2 subgraph isomorphism algorithm [6]. The data structures used for localization are described in Figure 2.

First of all, the measurement graph is represented in a particular traversal order using the grammar described in Figure 3 – vertices are enumerated in some sequence, and the second vertex of each edge includes the 1-based sequence number of the first such vertex; for example, see Figure 4. The traversal order employed is described by the prepare-graph procedure in Figure 5. We perform a “most-connected first” traversal: always choose to localize the node which has the most already-localized neighbors; if there is a tie, choose such a node with the most unlocalized neighbors. The reasoning is that the node with the most localized neighbors will have the fewest number of feasible positions, thus decreasing the branching factor of the search space; similarly, the tie-breaker attempts to constrain the largest number of thus far unlocalized nodes.

Find-next-localization-result in Figure 6 is the main localization procedure. In general, it iteratively localizes nodes such that (1) no two nodes occupy the same position, and (2) all distances

<code>graph[2 V + 2 E]</code> : array of numbers (measured graph)
<code>graph-index</code> : index into graph
<code>localized-so-far[V]</code> : array of (x, y, position-number)
<code>next-node-seq</code> : index into localized-so-far
<code>num-nodes</code> : total number of nodes, V
<code>num-fixed-nodes</code> : to avoid redundant solutions
<code>occupancy-hash-table[2 V]</code> : array of sequence numbers

Fig. 2. Data structures used for localization.

<code><graph></code>	::=	<code><node-info>...<node-info></code>
<code><node-info></code>	::=	<code><node-id><nbr-info>...<nbr-info>0</code>
<code><nbr-info></code>	::=	<code><nbr-seq><nbr-dist></code>
<code><nbr-seq></code>	::=	<i>1-based sequence number of a neighbor traversed before this node</i>
<code><nbr-dist></code>	::=	<i>measured distance to neighbor</i>

Fig. 3. Grammar of the graph data structure.

between localized node positions are consistent with the measured distances. Occupancy checks are performed using a hash table. The array `localized-so-far` is accessed like a stack, and maps node sequence numbers in the graph traversal to (x, y) coordinates. The position-number datum stores which possible positions of a node have already been tried; this is used for backtracking. The possible positions for a node n are determined by considering all points in the known layout which are approximately the measured distance away from n 's first already-localized neighbor. Since we do not fix a representation for the layout graph, it may be defined procedurally; for example, this is useful for specifying infinite regular grids. `Graph-index` points to the graph description of the next node to place, while `next-node-seq` contains that node's traversal sequence number, and hence points to its entry in `localized-so-far`. If all nodes can be localized without violating the measured distance and position occupancy constraints, the result is returned to the caller; whenever the search reaches a dead end or the caller requests another solution, we backtrack.

Figure 7 describes the procedures which direct the localization process. `Begin-localization` attempts to uniquely localize a core set of up to 3 nodes in order to eliminate rotational and mirror symmetries before calling `find-next-localization-result`. `Continue-localization` backtracks one step so that we can try a new position for the last-traversed node (`find-next-localization-result` may further cascade this backtracking).

From Figure 2, we see that memory usage is

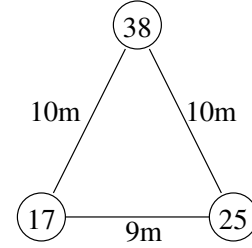
$$k_V|V| + k_E|E| + k_1 \in O(|V| + |E|) \quad (1)$$

where the k 's depend the sizes of the atomic data types used:

$$\begin{aligned} k_V &= \langle \text{id} \rangle + 1 + \langle x \rangle + \langle y \rangle + \langle \text{position} \rangle + 2\langle \text{node-count} \rangle \\ k_E &= \langle \text{node-count} \rangle + \langle \text{measurement} \rangle \\ k_1 &= \langle \text{graph-index} \rangle + 2\langle \text{node-count} \rangle + 1 \end{aligned}$$

4.2. Mitigating Errors

Due to low-cost hardware and environmental interference, ranging measurements in sensor networks are prone to inaccuracy. Our TDoA acoustic ranging service, described in [1], exhibits two



is represented by: 17 0 38 1 10 0 25 1 9 2 10 0

Fig. 4. A simple graph and its grammatical representation. Vertex labels indicate sensor ids, not traversal order.

```

proc prepare-graph:
  seq ← 1, i ← 0
  while some nodes in V are not marked,
    u ← the unmarked node with the most marked
        neighbors; tie-breaker: # unmarked neighbors
    graph[i] ← u's id; i++
    foreach {u, w} ∈ E, if w is marked,
      graph[i] ← w.seq
      graph[i+1] ← distance between u and w
      i ← i + 2
    mark(u)
    u.seq ← seq; seq++
  end-while

```

Fig. 5. Procedure to prepare the measurement graph in “most-connected first” traversal order.

```

proc find-next-localization-result:
  while num-fixed-nodes < next-node-seq ≤ |V|,
    if there are more possible positions for next-node-seq
      (x, y) ← next possible position
      localized-so-far[next-node-seq].position-number++
      if (x, y) is occupied, skip
      foreach localized neighbor n of node-seq,
        if distance((x, y), (xn, yn)) ≈ measurement, skip
      localized-so-far[next-node-seq].(x, y) ← (x, y)
      graph-index ← scan forward to next node
    else backtrack:
      localized-so-far[next-node-seq].position-number ← 0
      next-node-seq --
      graph-index ← scan back to previous node
      (skip to here)
  end-while
  return |V| + 1 = next-node-seq

```

Fig. 6. `Find-next-localization-result` is the main localization procedure. Returns true if a solution is found, false if not.

proc begin-localization:

*determine whether the first $k \in \{0..3\}$ nodes
can be uniquely localized*
place the first k nodes in localized-so-far[0..k]
next-node-seq $\leftarrow k$
graph-index \leftarrow scan past the first k nodes
num-fixed-nodes $\leftarrow k$
return find-next-localization-result()

proc continue-localization:

next-node-seq \leftarrow
graph-index \leftarrow scan back to last node
return find-next-localization-result()

Fig. 7. Procedures which direct the enumeration of localization results. Continue-localization can be called repeatedly until no more solutions are found.

Length	# Consistent	# Total	% Consistent
2	4	16	25.0
3	24	64	37.5
4	116	256	45.3
5	490	1024	47.9
6	2010	4096	49.1
7	8120	16384	49.6
8	32632	65536	49.8

Table 1. Number of cycles consistent with offset grid layout, assuming all edges are restricted to 9m, 10.06m, 16.22m or 18m.

kinds of errors: zero-mean Gaussian and outliers. The first case is addressed by tolerating a certain amount of error when comparing measurements to inter-node distances occurring in the layout. For example, if we allow an error of $\pm 1m$, a measurement graph edge labeled 9.3m may be matched with a layout graph edge labeled 9m or 10.06m, but not one labeled 16.22m.

Outliers can be subclassified as bidirectional measurement inconsistency, measurements which are not consistent with any distance occurring in the layout, and measurements which are consistent with an incorrect distance in the layout. Measurements of the first two types can easily be discarded. The strategy can be extended to also catch outliers of the third variety by checking that bounded-size subgraphs containing an edge are consistent with the layout. For example, if the layout information mandates that a node may have at most 4 neighbors at a distance of 16.22m, but the measurements claim a node has 5 such neighbors, we know that at least one of those edges is erroneous. Another example is checking that cycles in the measurement graph are consistent with the known layout. The number of consistent and total possible cycles for the offset grid layout are depicted in Table 1. Short cycles are particularly useful for finding inconsistencies, and asymptotically it appears that a cycle of k random measurements which are individually consistent with the grid spacing has a 50% chance of being totally consistent with the grid.

5. EXPERIMENTAL RESULTS

Our experiments consist of two phases: first, we deployed a 50-node sensor network in a 3000m² grassy field and collected acous-

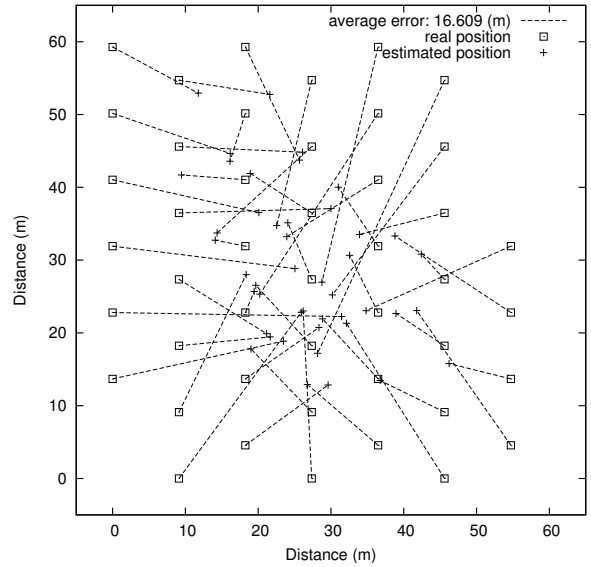


Fig. 8. Least Squares Scaling localization results for Experiment 06/14/04-1. The gradient descent algorithm failed to converge on a reasonable solution after 10 hours of execution on a 1.7GHz Pentium 4.

tic ranging measurements; second, we ran our localization algorithm on the measurement graph given the known deployment layout. The layout is illustrated by the node positions in Figure 1: we used a 7×7 offset grid, with 9m inter-node spacing and 4.5m row offsets. While random or irregular known layouts can be easy to match, a regular grid contains many more symmetries and ambiguities, thus better testing the effectiveness of our algorithm. The acoustic ranging service is based on the Time Difference of Arrival (TDoA) method: a sensor node sends a radio packet followed by an acoustic chirp; other nodes measure the time between reception of the two signals and use the speed of sound to estimate distance [1].

From Table 2, it is evident that the ranging measurements are quite sparse, with an average of 2.3 measurement edges per sensor node. Before exploring the results with subgraph isomorphism, let us consider how this measurement sparsity affects localization when layout information is not used. The Least Squares Scaling (LSS) localization technique minimizes the following error metric via the gradient descent minimization method,

$$E = \sum_{\{i,j\} \in E_{MG}} \left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - L_{E/MG}(\{i,j\}) \right)^2$$

where (x_i, y_i) is the estimated position of node i . Note that ranging measurement sparsity creates many deep local minima, such that the gradient descent algorithm is unable to find a reasonable solution. Figure 8 shows the results of LSS localization on ranging dataset 06/14/04-1 after running the gradient descent minimization procedure for 10 hours on a 1.7GHz Pentium 4 PC.

Table 2 summarizes our ranging data sets and the behavior of our localization algorithm. *Measured edges* is the number of different pairs of nodes between which a reliable ranging measurement was recorded (note that our acoustic ranging service includes

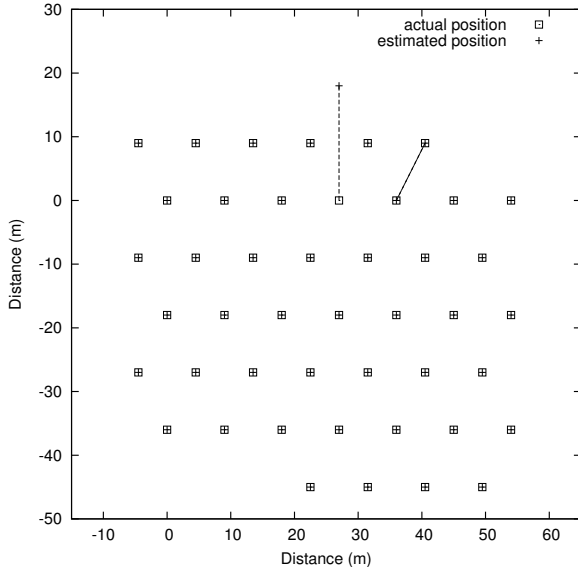


Fig. 9. Localization solution #4/10 for Experiment 06/14/04-1.

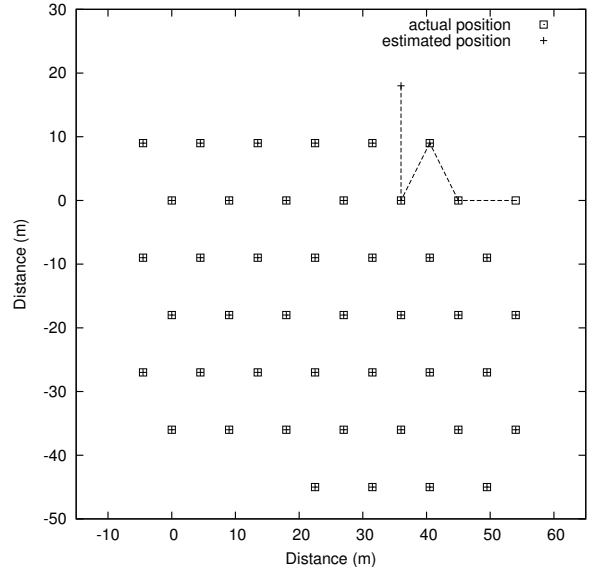


Fig. 10. Localization solution #10/10 for Experiment 06/14/04-1.

lower-level signal processing and statistical filtering techniques to eliminate a large number of bad measurements [1]). *Consistent edges* is the number which remained after filtering based on the criteria described in Section 4.2. The localization algorithm was implemented in C, compiled with GCC version 2.95.3 without any optimization flags, and run on a Pentium 4 2.53GHz PC with Linux 2.4.23. The *time* command built into bash 2.05a was used to measure the running time of a C program which executes the localization algorithm 1000 times on the input data set; this time was then divided by 1000 to compute the average execution times listed in Table 2. The fact that the number of consistent *solutions* is sometimes much greater than one indicates that some nodes' locations were not adequately constrained based on the measurements; previous algorithms which do not make use of layout information would have either not localized these nodes at all, or chosen some arbitrary location which does not truly reflect the information in the data set. Despite this uncertainty, our algorithm was able to find a consistent location in each solution for every node which was involved in at least one ranging measurement. Note that in the case of Experiment 06/13/04-5, the ranging measurement graph actually consisted of two unconnected components, which could not be localized relative to each other. This experiment used a higher threshold than the others in an attempt to reduce errors, however, this also reduced the number of measurements in an area of uneven ground and slightly taller grass.

Figures 9 and 10 show two representative incorrect localization solutions which our algorithm found were consistent with the ranging measurements in Experiment 06/14/04-1 and the known grid constraint (it also found the correct solution, which was, incidentally, #7/10). As is evident from the figures, most of the uncertainty is in the upper-right area of the sensor field. This is because nodes near the edge of the sensor field have fewer neighbors, and hence tend to have fewer measurements to constrain their positions. In Figure 9, we see that one node is flipped across the edge of the sensor field, while another pair of nodes are swapped; if we

had described our layout exactly, rather than as an infinite grid, this first error would have been caught automatically. Figure 10 is particularly interesting, because we see a *domino effect* in the errors. Due to measurement edges and the occupancy constraint, the node positioned above the top essentially *pulls* the other erroneously-positioned nodes with it, so again, a few extra constraints could possibly have eliminated most of the location errors. Across all experiments, 195 out of 216 node occurrences were uniquely localized. Furthermore, all except for 5 node occurrences (on average one per experiment) were constrained to at most 5 possible positions, as seen in Figure 11.

Figures 12 and 13 depict the number of localization solutions found by our algorithm over the course of time, in terms of iterations of the while loop in find-next-localization-result (Figure 6). Although different loop iterations will of course have different running times depending on the control flow, this still gives a macro-scale picture of the search behavior. In general, we see a stair-step pattern, with two different regimes: flat, where we are not finding any solutions, and steep, where many solutions are being encountered quickly. The flat regions encompass iteratively localizing the earlier nodes in the graph traversal and backtracking as we find conflicts. The step regions, on the other hand, indicate that the algorithm is only performing a small amount of backtracking, and mostly finding new solutions by repositioning the nodes near the end of the traversal. In the case of 06/13/04-4, we see a distinct 3-step pattern; in this case, it is evident that nodes occurring earlier in the traversal were underconstrained and in fact had three different possible consistent arrangements. Note that the runs end with flat regions, which can sometimes be quite long, because the algorithm is checking to make sure that there are no more consistent solutions. Thus, if a complete enumeration of solutions is not important, criteria such as running time, number of solutions found, or average rate of solution discovery could be used to shorten localization time while still considering a large number of possible solutions.

Experiment	Measured Edges	Consistent Edges	CPU time	Iterations	Solutions	Nodes Localized	Uniquely Localized
06/13/04-3	90	81	1.540 ms	561	17	34	32
06/13/04-4	127	118	13.490 ms	1853	187	47	42
06/13/04-5	88	71 + 13	15.650 + 0.910 ms	949 + 145	152 × 16	35 + 9	31 + 6
06/14/04-1	101	97	10.930 ms	2441	10	45	40
06/14/04-2	97	97	1.190 ms	737	4	46	44

Table 2. Number of different localization consistent with the ranging measurements, as found by our algorithm. The measurement graph of 06/13/04-5 contained two unconnected components which were localized separately.

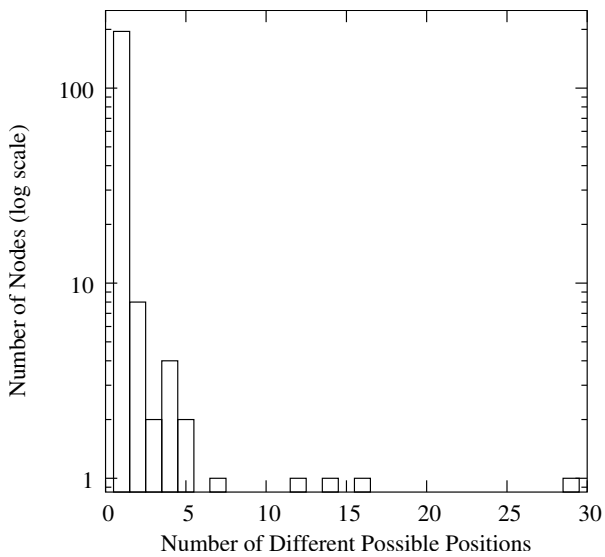


Fig. 11. Histogram describing the variation in node positions across multiple consistent solutions. 195 out of 216 node occurrences across all experiments are uniquely localized.

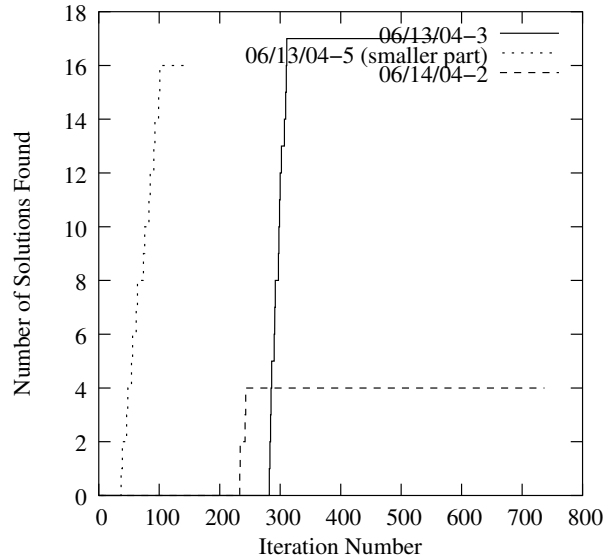


Fig. 12. Number of solutions found vs. iteration number for data sets with short running times.

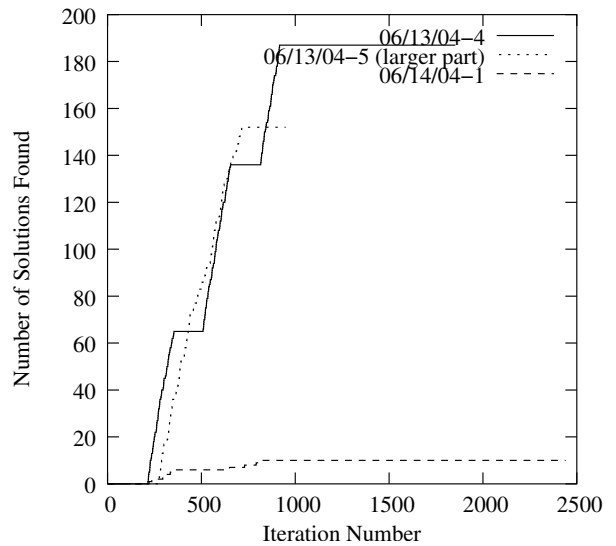


Fig. 13. Number of solutions found vs. iteration number for data sets with long running times.

We can calculate memory requirements from Equation (1) and Table 2. Using single bytes for all data types except graph-index, which requires two bytes, we need only 570 bytes of data for experiment 06/13/04-4, which involved the most measurements and nodes. This modest data memory usage along with the simplicity of our algorithm allow all memory accesses to remain in L1 cache, thus contributing to the good performance observed in practice.

6. CONCLUSION

In this report, we have recognized the practical problem of ranging measurement sparsity as well as the availability of network layout information to assist in localizing sensor networks with sparse measurements. We have then cast localization as a special case of subgraph isomorphism, and shown how to adapt an existing high-performance subgraph isomorphism algorithm to solve the localization problem. Experimental results show that our approach is very efficient on real measurement data, and indeed supersedes existing localization algorithms by exposing uncertainty in the form of multiple consistent solutions, rather than hiding it by arbitrarily choosing one solution or totally failing to localize a large number of nodes. Since our implementation operates quickly and uses a very small amount of memory (570 bytes to localize a 47-node network), in the future we will study the possibility of hosting localization computations entirely on sensor nodes themselves, effectively using the sensor network as an autonomous, scalable parallel computing platform. Furthermore, we will study other general approaches of introducing known layout constraints into localization computations, and how they compare with the technique discussed above.

7. REFERENCES

- [1] YoungMin Kwon, Kirill Mechitov, Sameer Sundresh, WooYoung Kim, and Gul Agha, "Resilient localization for sensor networks in outdoor environments," Tech. Rep. UTUCDCS-R-2004-2449, Department of Computer Science, University of Illinois at Urbana Champaign, 2004.
- [2] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover, 1998.
- [3] E. M. Luks, "Isomorphism of graphs of bounded valence can be tested in polynomial time," in *Journal of Computer and System Sciences* 25, 1982, pp. 42–65.
- [4] B. T. Messmer and H. Bunke, "Subgraph isomorphism in polynomial time," Tech. Rep. IAM 95-003, Institute of Computer Science and Applied Mathematics, University of Bern, 1995.
- [5] J. R. Ullman, "An algorithm for subgraph isomorphism," *Journal of the Association for Computing Machinery*, vol. 23, no. 1, pp. 31–42, January 1976.
- [6] P. Foggia, C. Sansone, and M. Vento, "An improved algorithm for matching large graphs," in *The 3rd IAPR-TC15 Workshop on Graph-based Representations*, Ischia, 2001.
- [7] P. Foggia, C. Sansone, and M. Vento, "A performance comparison of five algorithms for graph isomorphism," in *Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, Ischia, 2001.
- [8] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins, *Global Positioning System: Theory and Practice*, Springer-Verlag, 4th edition, 1997.
- [9] Interagency GPS Executive Board, "<http://www.igeb.gov/sa/>," .
- [10] Dragos Niculescu and Badri Nath, "Ad hoc positioning system (APS)," in *GLOBECOM*, November 2001.
- [11] Dragos Niculescu and Badri Nath, "Ad-hoc positioning system using AoA," in *Proceedings of the IEEE/INFOCOM 2003*, San Francisco, CA, April 2003.
- [12] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys'04)*, November 2004.
- [13] Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui, "Convex position estimation in wireless sensor networks," in *Proceedings of the 20th Conference of the IEEE Communications Society (IEEE INFOCOM)*, 2001, pp. 1655–1663.
- [14] Yi Shang and Wheeler Ruml, "Improved MDS-based localization," in *IEEE INFOCOM*, 2004.
- [15] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P.J. Fromherz, "Localization from mere connectivity," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Annapolis, Maryland, USA, 2003, pp. 201–212, ACM Press, <http://doi.acm.org/10.1145/778415.778439>.
- [16] Xiang Ji and Hongyuan Zha, "Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling," in *Proceedings of the 23rd Conference of the IEEE Communications Society (IEEE INFOCOM)*, 2004, (to appear).
- [17] Peter M. Lee, "Multivariate Analysis: Lecture Notes. Chapter 8: Multidimensional Scaling," <http://www.york.ac.uk/depts/maths/teaching/pml/mva/tex/8.ps>.