<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Solutions for connectivity-based sensor network localization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Qiao, D; Pang, GKH</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>The 2011 IEEE International Conference on Mechatronics and Automation (ICMA 2011), Beijing, China, 7-10 August 2011. In Proceedings of ICMA, 2011, p. 1056-1062</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>2011</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/135858">http://hdl.handle.net/10722/135858</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.; Proceedings of the IEEE International Conference on Mechatronics and Automation. Copyright © IEEE.; ©2011 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.</td>
</tr>
</tbody>
</table>
Abstract—This paper compares the solutions obtained by various methods in the literature for sensor network localization based on connectivity. The deficiencies of some of those solutions are discussed. It is argued that the actual problem should be represented as an optimization problem with both convex and non-convex constraints. A new method is proposed which utilizes multi-dimensional scaling (MDS) to provide an initial solution on the location of the unknown nodes and then searches for a solution to satisfy all the constraints of the problem. The final solution can reach the most suitable configuration of the unknown nodes because all the information on the constraints (convex and non-convex) related to connectivity will have been used. Compared with other constraint models that only consider the nodes that have connections, this method considers not only the connection constraints, but also the disconnection constraints. Simulation results have shown that better solution can be obtained through the use of this method when compared with those produced by other methods.

Keywords – Multiple dimensional scaling (MDS); nonlinear programming; non-convex constraints; localization; connectivity.

I. INTRODUCTION

Position estimation is necessary in many applications such as remote patient monitoring, package and personnel tracking, environment monitoring and wildlife habitat monitoring. In these systems, there could be hundreds or even thousands of low-cost sensor nodes, which can take some simple measurements. Based on either the signal strength or the connectivity among the nodes, we would like to estimate the location of these nodes in the sensor network. It is necessary to accurately localize the sensors in order to measure data which is geographically meaningful. This localization issue has been studied by many researchers and there are many different methods and algorithms [1–4] dealing with this situation.

In a typical sensor network, a few nodes have known positions, and they are called the anchors. However, the positions of the majority of the nodes need to be estimated using their relationships to the anchors and other unknown nodes. Based on whether the distances between nodes in a sensor network are known or not, the localization methods can be grouped into two categories: range-based and range-free. Range-based methods can be applied to the situation in which the distances between each pair of nodes are estimated or measured. The information is then communicated to a centralized station in the sensor network and algorithms such as MDS [3] compute the location of each sensor in the network. Usually, the distance between each pair of nodes is estimated by the signal strength received between them, and this information is very noisy in practice. On the other hand, range-free methods, which can also be called connectivity-based methods, assume that the distances between any two nodes are unknown. However, connectivity information between them is known. If the distance between any two nodes in the network is within a range, connectivity between the two nodes is said to be established. Although the actual distance is not known, this would provide many connection-imposed proximity constraints to the problem. These connectivity-based methods only require very simple and low-cost hardware. Yet, they can give adequate position estimation based on just connectivity information among the nodes. This paper will focus on the various connectivity-based methods for localization and propose how improvements can be made based on the previous methods.

II. PROBLEM DEFINITION

A formal definition of the connectivity-based localization problem is given next. Let $G = (V, E)$ be a given network, where $V$ denotes the nodes of the network and $E$ denotes the edge of the network. Let $V$ be partitioned into two sets:

\[ V_a = \{1, ..., m\} \text{ of anchors}; \]
\[ V_b = \{m + 1, ..., m + n\} \text{ of sensors}. \]

$E$ is also partitioned into two sets:

\[ E_{ab} = \{(i, j) \in E : i \in V_a, j \in V_b\} \] which are the edges between a sensor and an anchor;
\[ E_{bb} = \{(i, j) \in E : i, j \in V_b\} \] which are the edges between two sensors.

For each anchor $i \in V_a$, the position $a_i \in \mathbb{R}^2$ is assumed to be known. For each sensor $i \in V_b$, the position $b_i \in \mathbb{R}^2$ is assumed to be unknown.

Let $C_{ab} = \{(i, j, k) : i \in V_a, j \in V_b, k \in \{0,1\}\}$ be the connectivity information between a sensor and an anchor.
Also let \( C_{bb} = \{(i, j, k) : i, j \in V_a, k \in \{0, 1\} \} \) be the connectivity information between two sensors.

The value \( k \) in \( C_{ab} \) or \( C_{bb} \) is binary (either 0 or 1):
\[ k = 0 \text{ if there is no connection between node } i \text{ and } j, \]
\[ k = 1 \text{ if there is connection between node } i \text{ and } j. \]

Let \( a \) be a vector containing the positions of the anchors
\[ a = (a_i)_{i \in V_a} \in \mathbb{R}^{2n} \]

The goal of the network localization problem is to determine the coordinates of all the sensors (unknown nodes)
\[ b = (b_i)_{i \in V_a} \in \mathbb{R}^{2n} \]
such that \( b \) satisfies the following constraints:

Let \( R \) be the maximum distance (called the range) within which connectivity can be established.

If \( k = 1 \)
\[ \left\| a_i - b_j \right\|_2^2 \leq R^2 \text{ for } (i, j) \in E_{ab} \]
\[ \left\| b_i - b_j \right\|_2^2 \leq R^2 \text{ for } (i, j) \in E_{bb} \]
else \( k = 0 \)
\[ \left\| a_i - b_j \right\|_2^2 > R^2 \text{ for } (i, j) \in E_{ab} \]
\[ \left\| b_i - b_j \right\|_2^2 > R^2 \text{ for } (i, j) \in E_{bb} \]

III. RELATED WORK

Current connectivity-based localization algorithms on sensor networks include the centroid method [5], the approximate point in triangulation (APIT) [6], the multidimensional scaling–MAP (MDS–MAP) [3], DV–Hop [7] and the convex position estimation (CPE) [2].

A. Centroid

Centroid localization is probably the earliest and simplest approach. A proximity-based and coarse approach is proposed by Bulusu and Heidemann [5]. Every unknown node receives several nearby anchors’ information. The location information of the anchors is used, and the estimated location of the unknown node is assumed to be the average of the location of all the nearby anchors. The following formula is used:
\[ (X_{en}, Y_{en}) = \left( \frac{X_{a1} + ... + X_{a_k}}{k}, \frac{Y_{a1} + ... + Y_{a_k}}{k} \right) \]
where \((X_{a1}, Y_{a1}) \) … \((X_{a_k}, Y_{a_k})\) is the location of the \( k \) anchors that the unknown node \( i \) can contact; \((X_{en}, Y_{en})\) is the estimated location of node \( i \).

B. APIT

APIT (Approximate Point-In-Trimanluation test) was first proposed by He et al. [6]. The area around an unknown node is split into several pieces by some triangles with their vertexes being anchors it can hear. It tries to find whether the unknown nodes are in these triangles, and therefore determines in which piece the unknown node is located. In the process of checking whether it is in a triangle, the unknown node utilizes information obtained by its nearby unknown nodes. The strength of the signal transmitted by the anchors is also used in this process. In order to improve accuracy, APIT requires the anchors to have a larger communication range than normal nodes have. In their simulation, they have used anchors that have a communication range that is ten times larger.

C. Multi-dimensional scaling (MDS)

The basic MDS method [3, 8] can estimate the positions of all the unknown nodes by using the distance information between any two nodes. An extension of MDS [3, 9] for the connectivity-based localization problem has also been developed. First, a rough estimate of the relative node distance is made. Then, the relative positions are obtained by using a Singular Value Decomposition on the estimated distance information matrix. Finally, absolute positions of the unknown nodes are estimated based on the relative positions and the positions of the anchors. The computation complexity of this method is about \( O(n^3) \) time for a sensor network of \( n \) nodes.

MDS has also been modified for the connectivity-based localization problems based on the hop count information to replace the estimated distance between a pair of nodes [10]. One hop is one direct connection between two nodes. The hop count between two nodes represents the distance, more or less.

An example is shown in Figure 1. The connections are represented by lines between two nodes. The hop count between node 1 and node 5 is the number of connections between them, which is 3. Hence, the hop count between any two nodes can be obtained and used in MDS to estimate the locations of the nodes.

D. DV–HOP

Another well-known localization algorithm is DV–Hop (distance vector–hop) [7]. The idea of DV–Hop is to transform the distance to all anchors from hops to units of length measurement using the average size of a hop. DV–Hop was first proposed by Niculescu [11], and improved by many researchers. Anchors broadcast their location information to other anchors, and such information will be flooded with the hop count increment. Every anchor knows the hop count from any other anchor, and uses this information to estimate the average hop size. The distance between an anchor to an
unknown node is computed by the hop size and the hop count between them. At last, triangulation is used when an unknown node knows the distances to at least three anchors.

![Figure 2 A sensor network example for DV-hop](image)

In the example in Figure 2, anchors broadcast their position information to other anchors, and hop counting will be carried out in this process. The minimal hop count from A3 to A1 is 5. The minimal hop count from A3 to A2 is 6. Then the average hop distance (AHD) of anchor A3 is $AHD = \frac{(15+18)}{(5+6)} = 3$. There are many methods to find the AHD of an unknown node. Here, the closest anchor’s AHD as AHD for the unknown node is used.

$A3$ is the closet anchor to node 1, so we use the AHD of $A3$ to estimate the distance between node 1 and all the anchors.

- $d_{1,A1} = 3 \times AHD = 9$
- $d_{1,A2} = 4 \times AHD = 12$
- $d_{1,A3} = 2 \times AHD = 6$

At last, triangulation will be used to localize node 1.

**E. Convex Constraints in Localization[2]**

The connectivity between two nodes would tell whether the distance between these two nodes is less than a certain communication range [2]. The convex position estimation (CPE) uses this information in convex optimization and narrows the possible area by the solutions of the optimization.

Many researchers have formulated the connectivity-based localization problem as an optimization problem with some convex constraints. When two nodes are connected, the distance between them must be within a range distance. All the connections are then expressed by semi-definite inequalities. As all the constraints are convex, this method is called semi-definite programming or convex programming (SDPCC).

For examples, the convex constraints that represent the connectivity among nodes are ($k = 1$):

$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R$

for node $i (x_i, y_i)$ in connection with node $j (x_j, y_j)$.

Here, $R$ denotes the communication range defined for a pair of nodes.

**IV. Optimization Problem with Both Convex and Non-convex Constraints**

The CPE method has only formulated the optimization problem based on the convex constraints. However, in the sensor network, there are always nodes that are far away from each other and not in connection. Therefore, the solution should also consider the non-convex constraints which require the distance between some pairs of nodes to exceed the range value. In this section, we would like to consider both the convex and non-convex constraints in the search for a solution to the problem. The proposed method is called Nonlinear Programming with Convex and Non-convex Constraints (NPCC).

The NPCC method would combine MDS with a search for the positions of unknown nodes. The situation is formulated as an optimization problem as follows:

- If the connectivity information between a pair of node is known ($k = 1$), which means the two nodes are in connection, then there is a convex constraint in the solution to the problem. That is,

  $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R$

  for node $i$ in connection with node $j$.

- If a pair of nodes is disconnected ($k = 0$), then there is a non-convex constraint in the solution to the problem.

  $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} > R$

  for node $i$ not in connection with node $j$.

The constraints are formed by the connectivity information and an optimization algorithm is used to find a solution that minimizes a target function. In our method, the optimization is carried out in four steps with four different target functions. The first target is to minimize the summation of all the X coordinates of the estimated nodes. The second target is to minimize the summation of all Y coordinates of the estimated nodes. The third target is to maximize the summation of all the X coordinates of the estimated nodes. Finally, the fourth target is to maximize the summation of all Y coordinates of the estimated nodes. These four results would provide a rectangle that bounds the feasible region for the solution [2] and the center of the rectangle would be the final solution. In order to facilitate the search for the solution in the optimization, the result from the MDS is used as the starting point. Figure 3 shows the flowchart of the method.
V. SIMULATION RESULTS AND PERFORMANCE EVALUATION

We first compare the solutions from various established connectivity-based methods (centroid localization, APIT, MDS, DV–Hop, and SDPCC) and also show the result from our method (NPCC). In the first example, there are 8 anchors and 17 unknown nodes (i.e., \( m = 8, n = 17 \), 25 nodes in total) randomly placed in a square region of side length 10. The range of communication (\( R \)) is 3. In other words, if any two nodes are within a distance of 3, we assume they are connected. Otherwise, they are not. The average degree of connection is about 4.5 in this example.

Figure 5 gives the results from the six methods. The first subfigure gives the original node locations used to setup the problem. The anchors are shown in green squares. In the other subfigures showing the solutions of the six methods, the lines (either in green or in red colors) represent the connections between the estimated nodes. The red lines indicate the connection which is not supposed to exist (i.e., the connectivity between the two nodes should not exist actually).

The following table summarizes the results from the solutions obtained by various methods.

Note that the original network has 56 convex constraints and 244 non-convex constraints. The centroid localization, MDS, and DV–Hop have both convex and non-convex constraints unsatisfied. “Error on D” indicates the number of unsatisfied non-convex constraints, while “Error on C” means the number of unsatisfied convex constraints. SDPCC and APIT satisfy all the convex constraints. In general, APIT cannot guarantee that all convex constraints will be satisfied. In a later example with 104 nodes, APIT does not satisfy all the convex constraints. **Note that only the NPCC method has all the convex and non-convex constraints satisfied.** (i.e., the values of \( k \) in \( C_{ab} \) and \( C_{ba} \) are satisfied in the solution).

It should be mentioned that the result obtained by APIT is not a feasible solution. This is because APIT can estimate position of only 3 nodes from the 17 unknown nodes. Although the average error of the 3 nodes is small, we do not compare APIT with other methods because it cannot work out the location of all the unknown nodes. In the table showing the average error of centroid localization, MDS, DV–Hop, SDPCC and NPCC, NPCC is much more accurate than other methods.

In the procedure of locating the nodes into small split areas, APIT needs the nodes to obtain received signal strength (RSS) readings of the signal transmitted from anchors. However, our simulation does not have any assumption about RSS. Here, to compare the best performance of APIT, we assume the locating procedure has 100% accuracy. Both centroid localization and APIT require that the anchors have much larger communication range than the normal nodes have. For the centroid localization, the anchor-to-node range radio (also called ANR) is 2. For APIT, the ANR is assumed to be 3. Those values enable the centroid localization and APIT to give an excellent performance in the simulation. MDS only needs 3 anchors for locating the relative coordinates. Since 8 anchors are given in this simulation, we choose a suitable set of 3 anchors to give the node locations.

For SDPCC, the convex constraints which form a semi-definite programming problem are solved using the Mosek Optimization Toolbox [12]. The mathematic model of NPCC is a typical nonlinear programming problem which contains not only convex constraints but also non-convex constraints. An algorithm called “active-set” is used to solve the problem. The solution is obtained by the Matlab nonlinear constrained optimization function fmincon().

In the second example, the sensor network has 24 anchors and 80 unknown nodes (i.e., \( m = 24, n = 80 \), 104 nodes in total) randomly placed in a square region of side length 10. The range of communication (\( R \)) is 1.4. The average degree of connection is 4.8. Figure 6 gives the results from the six methods. The first subfigure gives the original node locations used to set up the problem.

The original network has 284 convex constraints and 5072 non-convex constraints. The following table summarizes the results from the solutions obtained by various methods. The solutions obtained from centroid localization, APIT, MDS, DV–Hop and SDPCC have both convex and non-convex constraints unsatisfied. “Error on D” indicates the number of unsatisfied non-convex constraints, while “Error on C” means the number of unsatisfied convex constraints. **Only the NPCC...**
method has all the convex and non-convex constraints satisfied. The following table summarizes the results from the solutions obtained by various methods:

<table>
<thead>
<tr>
<th>104 nodes</th>
<th># Connections (C)</th>
<th># Disconnections (D)</th>
<th>Error on D</th>
<th>Error on C</th>
<th>Average error</th>
</tr>
</thead>
<tbody>
<tr>
<td>centroid</td>
<td>284</td>
<td>5072</td>
<td>220</td>
<td>66</td>
<td>0.8585</td>
</tr>
<tr>
<td>APIT</td>
<td>284</td>
<td>5072</td>
<td>2235</td>
<td>48</td>
<td>0.3159</td>
</tr>
<tr>
<td>MDS</td>
<td>284</td>
<td>5072</td>
<td>148</td>
<td>13</td>
<td>1.3644</td>
</tr>
<tr>
<td>DV–Hop</td>
<td>284</td>
<td>5072</td>
<td>94</td>
<td>36</td>
<td>0.6917</td>
</tr>
<tr>
<td>SDPCC</td>
<td>284</td>
<td>5072</td>
<td>119</td>
<td>18</td>
<td>0.5665</td>
</tr>
<tr>
<td>NPCC</td>
<td>284</td>
<td>5072</td>
<td>0</td>
<td>0</td>
<td>0.2999</td>
</tr>
</tbody>
</table>

Note: For centroid localization, ANR = 2. For APIT, ANR = 3.

APIT has no result on 26 unknown nodes.

The sensor network localization problem based on connectivity is actually a typical nonlinear programming problem, which contains both convex constraints and non-convex constraints. A method for formulating this connectivity-based problem as nonlinear programming is proposed. To help with the search for a solution, the result from the MDS method is used as a starting point. This heuristic is obtained from our experience after many trials and simulations. More results will be presented in another paper.

In this paper, the results are obtained based on only a small and medium sized network. More results and analysis will be presented in the future. It must be mentioned that when compared with other current methods such as centroid localization, APIT, MDS, DV–Hop, and SDPCC, ours is the only method which can satisfy all the convex and non-convex constraints. A comparison of the average error of the various solutions with the original setup also shows that NPCC method has the best accuracy.

**REFERENCES**

Figure 5  The original location of the 25 nodes and the solution of the method: centroid localization; APIT; MDS; DV-hop; SDPCC; NPCC. In ‘An example of a sensor network’, the small circles are the original node locations, and the small squares are the anchors. The green lines represent the connections between the nodes, with a range of 3. In other figures, the small triangles are the estimated locations of the unknown nodes. The connections between the estimated nodes are indicated by the green lines and the red lines. The red lines indicate the connection which is not supposed to exist (i.e. the connectivity between the two nodes should not exist actually).
In ‘An example of a sensor network’, the small circles are the original node locations, and the small squares are the anchors. The green lines represent the connections between the nodes, with a range of 1.4. In other figures, the small triangles are the estimated locations of the unknown nodes. The connections between the estimated nodes are indicated by the green lines and the red lines. The red lines indicate the connection which is not supposed to exist (i.e. the connectivity between the two nodes should not exist actually).