Minimum Cost Localization Problem in Wireless Sensor Networks

Minsu Huang, Siyuan Chen, Yu Wang
Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA.
Email:{mhuang4,schen40,yu.wang}@uncc.edu

***Abstract*—Localization is a fundamental problem in wireless sensor networks. Current localization algorithms mainly focus on checking the localizability of a network and/or how to localize as many nodes as possible given a static set of anchor nodes and distance measurements. In this paper, we study a new optimization problem, *minimum cost localization problem*, which aims to localize all sensors in a network using the minimum number (or total cost) of anchor nodes given the distance measurements. We show this problem is very challenging and then present a set of greedy algorithms using both trilateration and local sweep operations to address the problem. Extensive simulations have been conducted and demonstrate the efficiency of our algorithms.**

I. Introduction

Location information can be used in many sensor network applications, such as event detecting, target tracking, environmental monitoring, and network deployment. On the other hand, location information can benefit networking protocols to enhance the performance of sensor networks in different ways, such as delivering packets using position-based routing, controlling network topology and coverage with geometric methods, or balancing traffics in routing using location information. However, manually configuring individual node's position or providing each sensor with a Global Positioning System (GPS) to obtain its location is expensive and infeasible for most sensor networks. Therefore, localization problem is a fundamental task in designing wireless sensor networks.

The main task of localization in wireless sensor networks is to obtain the precise location of each sensor in the 2-dimensional (2D) plane. To achieve this goal, several special nodes (called anchor nodes[1]), who know their own global locations via either GPS or manual configuration, are needed. The rest of sensors will determine their locations by measuring the Euclidean distances to their neighbors using different distance ranging methods (such as radio signal strength or time difference of arrival).

Given positions of anchor nodes, and distance measurements among all pair of neighbors, to find the positions of all sensors is still a very challenging task. In some cases, it is even impossible. A network is *localizable* if there is exactly one set of positions in the 2D plane for all nodes that is consistent with all available information about distances and positions. There is a strong connection between network localizability and mathematical rigidity theory [1]. Recent theoretical works [2]–[4] show that the network is localizable if and only if the graph is globally rigid. However, the problem of realizing globally rigid weighted graphs (i.e., the network localization problem) is NP hard [4].

A significant amount of localization algorithms [5]–[8], [10]–[15] have been developed to localize sensor nodes by exchanging information with anchor nodes. *Trilateration* is a basic localization technique and has been widely used in practice [5], [6]. To accurately and uniquely determine the location of a node in a 2D plane by trilateration, distance measurements to at least three anchor nodes (or sensor nodes who already know their positions) are needed. By iteratively applying trilateration, it is possible to identify localizable nodes in a network. However, as pointed out by [7], [8], trilateration has a clear deficiency: it can only recognize a subset of sensors even when the network is globally rigid.

To overcome the limitation of trilateration, there are some recent works on new techniques which aim to localize more sensors beyond trilateration. Yang *et al.* [8] proposed a localization method based on detection of wheel structures to further improve the performance of localization. Their method is based on the following claim made by them that *all nodes in a wheel structure with three anchor nodes are uniquely localizable*. However, in this paper, we show a counter-example in which nodes on a wheel structure cannot be uniquely localized. Goldenberg *et al.* [7] introduced a localization method for sparse networks using sweep techniques. Their method uses all possible positions of sensors in each positioning step and prunes incompatible ones whenever possible. Therefore, the possible positions could increase exponentially with the number of sensors. This limits the advantage of sweep method.

All existing localization methods try to localize more sensor nodes in a network without guarantee of localizing all nodes. They usually assume that there are enough anchor nodes to achieve the goal, or the set of anchor nodes are pre-decided before deployment of the network. However, in this paper, we focus on studying a new localization problem, called *minimum cost localization problem* (MCLP). MLCP is an optimization problem which aims to localize all nodes in a network using minimum anchor nodes. This is an important problem, since the cost of manually configuring an anchor node or equipping it with a GPS device is expensive in many cases. In MCLP, we concentrate on the selection of an anchor set such that (1) the whole network could be localized, and (2) the total cost

This work is supported in part by the US National Science Foundation (NSF) under Grant No. CNS-0721666 and No. CNS-0915331.

[1]They are also called reference nodes or beacon nodes in other papers.

of setting up these anchors is minimized. This is completely different from previous works on localization and has never been studied before. Notice that Khan *et al.* [9] recently also proposed a localization method using the minimal number of anchor nodes. However, they assume that the sensing range of each sensor can be enlarged to guarantee certain triangulation, thus only three anchor nodes are needed to localize all sensors in 2D plane. In our problem, the sensing range of each sensor is fixed, therefore, their method does not work.

The rest of this paper is organized as follows. In Section II, we summarize related works in localization for sensor networks. In Section III, we formally define the minimum cost localization problem and discuss its hardness. In Section IV, we propose four greedy algorithms to solve the MLCP using trilateration and/or sweep operations. Section V presents the simulation results of all proposed algorithms. Finally, Section VI concludes the paper by pointing out some possible future directions.

II. Related Works

A. Trilateration

Trilateration is the most basic technique for positioning system and has been used for thousands of years. It uses the known locations of multiple anchor nodes and the measured distance to each anchor node. To determine the accurate location of a node in a 2D sensor network using trilateration alone, it needs to hear from at least three anchors. However, in a sparse sensor network, many sensors may not be able to directly communicate with enough anchor nodes to compute their positions. Fortunately, sensors can also learn the distances among themselves using different distance ranging techniques (such as received signal strength (RSS), time of arrival (ToA), or time difference of arrival (TDoA)). In many localization algorithms [5], [6] designed for wireless sensor networks, iterative trilateration (or multilateration) is used to localize nodes via multihop. The basic idea is as follows: nodes measure distances to their neighbors and share their position information with their neighbors to collaboratively compute their positions. If a sensor node whose position has already been uniquely determined, it can act as a new anchor node to localize other nodes by sharing its position with its neighbors. This iterative process continues until there are no nodes can be further localized.

B. Rigidity Theory and Localizability

Even trilateration can compute the position of a sensor node using range measurements via anchor nodes when enough measurements are available, in practice it is still possible that many nodes' positions cannot be uniquely determined with limited measurements. An important question is under what conditions the network localization problem is solvable (*i.e.*, each node has a unique position solution). There is a strong connection between the problem of unique network localization and a mathematical topic known as rigidity theory [1]. Recently, several new results [2]–[4] have been published in sensor network area.

The network localization problem with distance information is to determine locations p_i of all nodes $v_i \in V$ in the real 2D space R^2 given the graph of network $G = (V, E)$, the positions of anchor nodes in R^2, and the distance between each neighbor pair $(v_i, v_j) \in E$. The localization problem is said to be *solvable* or the network is said to be *localizable* if there is exactly one set of positions in R^2 for all unknown nodes that is consistent with all available information of distances and positions. The localization problem can also be formed as a point formation $F_p = (\{p_1, p_2, \cdots, p_n\}, L)$ where p_i is node i's position and L is a set of links whose internode distances are given (including both the distances measured from unknown nodes to anchor nodes and the distances among unknown nodes). Then in [2], [4], the following theorem gives the conditions for a network to be localizable.

Theorem 1: [2], [4] For a network in R^2, if there are at least 3 anchor nodes in general position, the network is uniquely localizable if and only if the point formation for the graph G is *globally rigid.*

Here we say a set of points is in general position if any three points do not lie on a line. We then give a rough definition of global rigidity. Consider a point formation (and its corresponding graph) with edges connecting some of them to represent distance constraints. If there is no other point formation which consists of different points but preserves all distance constraints, then we call this point formation or its corresponding graph *globally rigid*.

Results from rigidity theory give efficient ways (polynomial algorithms) to check whether a graph is globally rigid. The following theorem gives a sufficient and necessary condition for global rigidity test in 2D space.

Theorem 2: [16] A graph with $n \geqslant 4$ vertices is generically globally rigid in R^2 if and only if it is redundantly rigid and 3-connected in two dimensions R^2.

Here a graph is redundantly rigid if the removal of any single edge results in a graph that is also generically rigid. This condition can be checked in polynomial time.

Even though the global rigidity can be determined efficiently, the problem of realizing globally rigid weighted graphs (the network localization problem) is still NP hard. Aspnes *et al.* [4] proved this by giving a polynomial-time reduction of the set-partition problem to the globally rigid weighted graph realizing problem. For more details about the complexity of localization problem, please refer to [4].

C. Beyond Trilateration

Recently, Yang *et al.* [8] proposed a localization method based on detection of wheel structures to further improve the performance beyond trilateration. Here a wheel graph W_n is a graph with n nodes, formed by connecting a single node to all nodes of an $(n-1)$-cycle. In [2], the wheel graph has been proved to be globally rigid. Then [8] claimed that *all nodes in a wheel structure with three anchor nodes are uniquely localizable*. Based on this observation, their method uses detection of wheel structure to identify more localizable

nodes than simple trilateration. However, we will show a counter-example in Section IV-B in which nodes on a wheel structure cannot be uniquely localized due to a possible flip.

Goldenberg *et al.* [7] introduced a localization method for sparse networks using sweep techniques. In trilateration, only a part of nodes can be uniquely localizable and there are still some nodes whose positions can not be uniquely decided. However, some of such nodes can be localized up to a set of possible locations. The idea of sweeping method is recording all possible positions in each positioning step and pruning incompatible ones whenever possible. One drawback of sweeping method is that the possible positions could increase exponentially with large number of nodes.

There are also other types of localization methods, such as using multidimensional scaling [10], [11] or mobile anchors [12]–[15]. However, all the previous localization methods try to localize more sensor nodes in a network without guarantee of localizing all nodes. Instead, in this paper, we study an optimization problem which aims to localize all nodes in a network using minimum anchor nodes. Recently, Khan *et al.* [9] also proposed a localization method to localize all nodes using the minimal number of anchor nodes. However, they assume that the sensing range of each sensor can be enlarged to guarantee certain triangulation, thus, three anchor nodes are enough to localize all sensors in 2D plane. Instead, in our study, the sensing range of each sensor is fixed. There are also anchor-free localization algorithms [17]–[19] proposed in the literature, which compute the relative positions of all sensors without the help from anchor nodes. However, these methods either require high density of sensors to preform boundary detection and landmark selection or need sensors equipped with motion actuator. The results from these methods are usually relative or roughly estimated positions instead of accurate positions, which are good enough for certain applications such as location-based routing. But in many sensor applications, more accurate locations are needed, thus we focus on anchor-based localization in this paper.

III. Minimum Cost Localization Problem

Assume that a sensor network is modeled as a graph $G = (V, E)$, where V is the set of n sensors $v_1, \cdots, v_n$ and E is the set of links. Here if a link $v_i v_j \in E$, the distance between sensors v_i and v_j can be measured or estimated via wireless communication. Each sensor could have different sensing range. Hereafter, we assume that no three sensors are collinear[2]. A subset of sensors $B \subset V$ are anchor nodes whose positions are known at the beginning of localization. The remaining sensors will rely on distance measurements in E and the positions of anchor nodes B to determine their locations during the localization procedure. We further assume that there is a unit cost to make a sensor node as an anchor node (e.g., equip it with a GPS device or perform a manual measurement). Then the *minimum cost localization problem* can be formally defined as follows:

Definition 1: **Minimum Cost Localization Problem (MCLP):** Given a sensor network G, find a subset of sensors B to be anchor nodes such that (1) all sensors can be localized and realized (i.e., their positions can be calculated) given the graph, the length of all links, and positions of all anchor nodes; and (2) the total number of anchor nodes $|B|$ is minimized.

Hardness of MCLP: It is clear that MCLP always has a feasible solution, since in the worst case every sensor is selected as anchor node, i.e., $B = V$. However, finding the optimal solution of such problem is very challenging. Even though the solvable of localization problem (i.e., global rigidity testing) is computable in polynomial time, Aspnes *et al.* [4] and Eren *et al.* [2] showed that realizable globally rigid weighted graph realization is still NP-hard even in unit disk graphs. Thus, to check whether a solution in MCLP can realize all sensors in the plane is still NP-hard. Therefore, MCLP is also a hard computational problem, even for a simple graph model (such as unit disk graph).

Next, we discuss the possible lower and upper bounds on the size of the optimal solution B_{opt} of MCLP. First, it is obvious that all sensors with node degree less than 3 should be included in B. When a sensor only has two neighbors, it cannot determine its location from the other nodes. Let $V_{<3}$ be the union of such sensors, then $|V_{<3}|$ is an obvious lower bound of $|B_{opt}|$. Notice that there exists a network in which $|B_{opt}| = |V_{<3}|$, such as a circular network where every sensor has degree 2. On the other hand, if all node degrees in G are larger or equal to 3, the upper bound of $|B_{opt}|$ can be given by the size of the minimum 3-dominating set $M3DS_{opt}$ of V. Here, the minimum k-dominating set of V (denoted by $MkDS_{opt}$) is a subset of nodes so that (1) every node not in $MkDS_{opt}$ must have k neighbors in $MkDS_{opt}$; and (2) $|MkDS_{opt}|$ is minimized. Finding the minimum k-dominating set (MKDS) is a NP-hard problem even in unit disk graphs. This also implies that analysis on optimal solution of MCLP problem is extremely hard. The best constant approximation of MKDS problem is $max\{\frac{5}{k}, 1\}$ from [20].

IV. Greedy Algorithms for MCLP

Since the problem of MCLP is computationally challenging, in this section, we propose several greedy algorithms to approximately solve MCLP. Our greedy algorithms differ from each other depending on how many hops of information is used at each node. To explain the algorithms easily, we define the status of each sensor using different colors. A white node is a node whose location is undecided yet, a black node is an anchor node whose position is known, and a green node is a non-anchor node whose position is already obtained via localization. Initially, all nodes are white. The purpose of our algorithms is to find the smallest set of anchors (black nodes) to get the positions of all nodes (coloring all nodes in either black or green).

[2]Notice that our proposed greedy methods can handle collinear sensors easily by testing the reference nodes' positions during trilateration or local sweep. If the reference nodes are collinear, one more reference node is needed for locating the position.

Algorithm 1 MARK$(u, color)$

1: $s(u) = color$.
2: **for all** u's neighbor v and $s(v) = white$ **do**
3: $r(v) = r(v) + 1$.
4: **end for**
5: **if** any u's white neighbor v and $r(v) \geq 3$ **then**
6: **if** $color = black$ or $green$ **then**
7: MARK$(v, green)$.
8: **end if**
9: **if** $color = blue$ **then**
10: MARK$(v, blue)$.
11: **end if**
12: **end if**

Algorithm 2 Greedy Localization Algorithm

1: **for** each $v \in V$ **do**
2: $s(v) = white$ and $r(v) = 0$.
3: **end for**
4: **for** each $v \in V$ **do**
5: **if** the degree of $v \leq 2$ **then**
6: MARK$(v, black)$.
7: **end if**
8: **end for**
9: **while** $\exists v$ such that $s(v) = white$ **do**
10: u = GREEDY-SELECTIONi.
11: MARK$(u, black)$.
12: **end while**

Algorithm 3 GREEDY-SELECTION1

1: $c_0^{max} = c_1^{max} = c_2^{max} = -1$.
2: **for all** v and $s(v) = white$ **do**
3: Let $c_i(v)$ be the number of v's neighbors with rank i.
4: **if** $c_2(v) > c_2^{max}$ **then**
5: $c_0^{max} = c_0(v), c_1^{max} = c_1(v), c_2^{max} = c_2(v)$.
6: $ID^{max} = v$.
7: **else if** $c_2(v) = c_2^{max}$ **then**
8: **if** $c_1(v) > c_1^{max}$ **then**
9: $c_0^{max} = c_0(v), c_1^{max} = c_1(v), c_2^{max} = c_2(v)$.
10: $ID^{max} = v$.
11: **else if** $c_1(v) = c_1^{max}$ **then**
12: **if** $c_0(v) > c_0^{max}$ **then**
13: $c_0^{max} = c_0(v), c_1^{max} = c_1(v), c_2^{max} = c_2(v)$.
14: $ID^{max} = v$.
15: **end if**
16: **end if**
17: **end if**
18: **end for**
19: Return ID^{max}.

Algorithm 4 GREEDY-SELECTION2

1: **for all** v and $s(v) = white$ **do**
2: MARK$(v, blue)$.
3: Let $c(v)$ be the number of blue nodes.
4: **for all** v and $s(v) = blue$ **do**
5: $s(v) = white$.
6: Let $r(v)$ be the number of its black and green neighbors.
7: **end for**
8: **end for**
9: Return v with the maximum $c(v)$ (tie is broken by ID).

A. Greedy Algorithms based on Trilateration

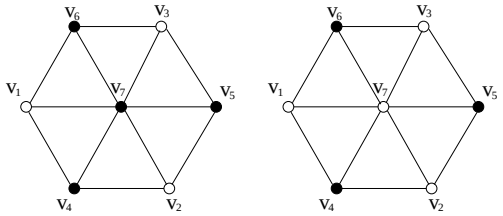
We first introduce a simple greedy algorithm only based on trilateration. Recall that in trilateration position of a white node can be calculated (becomes a green node) if it has three non-white neighbors. For each white node v, we maintain a rank $r(v)$ to indicate the number of its located (non-white) neighbors. When we mark a node u as either black or green, we first update its white neighbors' ranks and employ trilateration to locate as many nodes as possible. When a node v has three located neighbors (i.e., $r(v) = 3$), it can be marked as green. This MARK procedure could be done recursively, as shown in Algorithm 1. Lines 9-11 is the same procedure except for using blue instead of green or black, which will be used by Algorithm 4 later for estimation of the number of localizable nodes by making one node as an anchor node.

The basic idea of the greedy algorithm is as follows: (1) All nodes with degree less than three are colored black first, since they cannot be located by other nodes. Notice that when these nodes are marked as black, the MARK procedure will color as many surrounding nodes as possible. (2) In each step, we greedy select a white node which can benefit the localization procedure most in next step if it is marked as black, and color it as black. The whole procedure is given in Algorithm 2. The algorithm will terminate when all nodes are colored.

It is clear the ordering of picking the white node to color as black in each step is crucial and will affect the quality of the final output. We now introduce two ways to define the benefit of marking a white node as black.

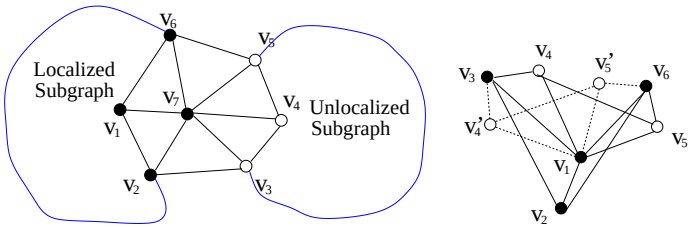
The first one (as shown in Algorithm 3) is purely based on information of its one-hop neighbors. Basically, for a white node v we first consider the number of its white neighbors with rank 2 (denoted by $c_2(v)$). White nodes with rank 2 are critical, since with one more located neighbor, their locations can be computed by trilateration. Thus, we pick the white node v with largest $c_2(v)$. When it is a tie, we consider the number of white neighbors with rank 1 or rank 0. ID is used for the last tie-break. Hereafter, we denote this greedy algorithm (consists of Algorithms 1,2,3) as *Greedy-Tri-1*. The limitation of this method is that the scope of estimated benefit of coloring a node in black is limited within its one-hop neighborhood. However, the real benefit of adding a new anchor node could be beyond its immediate neighbors.

The second way to define the benefit of coloring a node in black is to compute how many nodes can be located via iterative trilateration. To do so, for each white node v, our algorithm (as shown in Algorithm 4) runs a fake MARK procedure to color it in blue and recursively color other nodes



(a) greedy solution (b) optimal solution

Fig. 1. Greedy algorithm does not lead to the optimal solution.



(a) Deficiency of Trilateration (b) Flip of a Wheel

Fig. 2. (a) An example of deficiency of trilateration, where trilateration cannot localize the network while the network is localizable with a wheel structure. (b) An example where the wheel structure cannot realize the nodes' positions due to a possible flip. In this example, both v_4, v_5 and v_4', v_5' are feasible positions which are consistent with all distance measurements.

in blue using trilateration. We count the number of all blue nodes marked by node v, denoted by $c(v)$. The node with largest $c(v)$ will be selected as the next anchor node. Notice that we need restore the white color and right ranks for all blue nodes in each step after fake MARK procedure. This method basically estimates all possible benefit from a node via iterative trilateration. Thus, it performs better than *Greedy-Tri-1* method. Hereafter, we denote this new greedy algorithm (consists of Algorithms 1,2,4) as *Greedy-Tri-2*.

Both of greedy algorithms may not generate the optimal solution for MCLP. Figure 1 illustrates such an example with a 7-sensor network. Running *Greedy-Tri-1* on the network, the result is shown in Figure 1(a). v_7 is first marked as black since it has 6 white neighbors with rank 0. Then all white nodes have the same amount of white neighbors at each rank. Thus, v_6 is marked since it has the highest ID among them. Then v_5 is colored next since it has the highest ID among nodes have 1 white neighbor with rank 2. And v_3 is colored green by v_5. At last v_4 is colored since it now has two neighbors with rank 2. After that, v_1 and v_2 will be colored in green. This gives the solution as shown in Figure 1(a). If we run *Greedy-Tri-2* on the network, the same solution will be generated. However, the optimal solution of MCLP on this network should be three nodes like v_1, v_2, v_3 or v_4, v_5, v_6, as shown in Figure 1(b). After coloring these three nodes, v_7 can be marked as green, and then remaining nodes can also be marked as green.

B. Greedy Algorithms based on Local Sweeps

Recent research [7], [8] shows that trilateration has its own limitation. Figure 2(a) illustrates an example from [8] where trilateration can not propagate to the network while it actually should be localizable due to the globally rigidity of the network. In this example, the left part of the network is already marked black or green by trilateration, and the remaining part is connected via a wheel structure. Based on trilateration, nodes v_3, v_4 and v_5 cannot be localized since they all have only one or two colored neighbors. However, since the wheel structure is globally rigid, it should be able to localize all nodes on the wheel. Therefore, in [8] the authors propose a method to detect such wheel structures and use them to perform localization beyond trilateration.

Even though the wheel structure is localizable due to its globally rigid, it may not lead to unique realization of node positions as claimed in [8]. Figure 2(b) shows such an example. In this example, v_2 to v_6 are all neighbors of v_1 and form a circle, together with v_1 they form a wheel structure. Even assuming that the other nodes already know their positions, v_4 and v_5 cannot decide their positions due to a possible flip at v_4' and v_5'. Notice that this flip does not violate the distance measurements among all nodes. Therefore, *simply detecting wheel structure (as in [8]) is not sufficient to realize all nodes*.

In order to overcome the problem of wheel structure above, we can use sweep operations to check the consistency of possible positions of nodes in a local neighborhood and localize them if possible. Basically, we first compute all pairs of possible positions of two neighboring nodes, then calculate the corresponding distances between them. If there is only one pair of possible positions which can match the real distance measurement, these two nodes can then be realized. To simplify the operation, we limit such sweeps in two- or three-hop ranges from the processing node. This is different from the method used in [7], since they do not limit the range and type of sweep operations which leads to possible exponential growth of complexity.

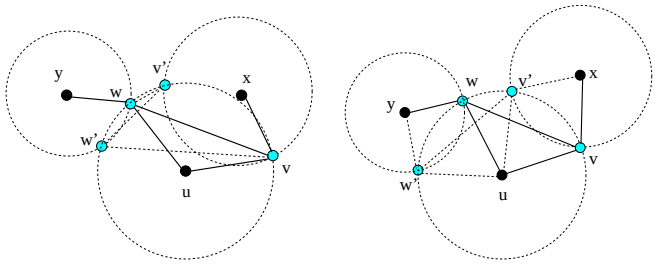
Next we present two greedy algorithms which use local sweeps in two-hop or three-hop neighborhood to localize nodes beyond trilateration. The key idea is using two neighboring nodes whose ranks are both 2 to localize each other. Since when a node's rank reaches 2, its location has been limited to two possible positions. The distance between these two nodes will be used to eliminate the bogus positions.

Figure 3 illustrates examples for the first method. In this method, when we consider to mark a node u as a new black node, we check the number of new nodes can be realized not only via trilateration but also using a local sweep in two-hop neighborhood of u. If there exist two white neighbors w and v with rank of 2 and they are neighbors to each other, u can check whether there is a pair of possible positions of these two nodes which can uniquely satisfy the distance measurement between them. If yes as shown in Figure 3(a), we consider both v and w can be marked as green by u. Otherwise, as shown in Figure 3(b), they can not be realized by u. Algorithm 5 shows the modified MARK process. Hereafter, we call this method (consists of Algorithms 5,2,4) *Greedy-Sweep-1*.

The second method based on sweep is an extension of *Greedy-Sweep-1*. It further considers the nodes in u's three-

Algorithm 5 MARK1$(u, color)$

1: All lines (Lines 1-12) in Algorithm 1 except for changing MARK to MARK1.
2: **if** any two u's white neighbors v and w satisfying $r(v) = r(w) = 2$ and they are neighbor to each other **then**
3: **if** both v and w have unique positions to guarantee the consistence of distance measurement **then**
4: **if** $color = black$ or $green$ **then**
5: MARK1$(v, green)$ and MARK1$(w, green)$.
6: **end if**
7: **if** $color = blue$ **then**
8: MARK1$(v, blue)$ and MARK1$(w, blue)$.
9: **end if**
10: **end if**
11: **end if**

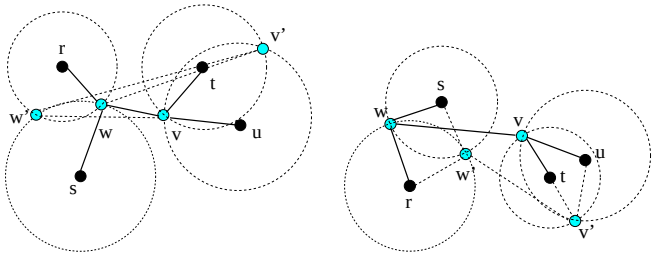


(a) unique match (b) no unique match

Fig. 3. Greedy method based on local sweep in two-hop neighborhood (*Greedy-Sweep-1*): Consider marking u as a new black node. When both w and v are neighbors of u with rank of 2, they now have only four combinations of possible positions. If there is a distance measurement between w and v. u can check which pair of positions uniquely satisfies the measurement. (a) There is a unique match. (b) There is no unique match.

Algorithm 6 MARK2$(u, color)$

1: All lines (Lines 1-11) in Algorithm 5 except for changing MARK1 to MARK2.
2: **if** any u's white neighbors v and any v's white neighbor w satisfying $r(v) = r(w) = 2$ and they are neighbor to each other **then**
3: **if** both v and w have unique positions to guarantee the consistence of distance measurement **then**
4: **if** $color = black$ or $green$ **then**
5: MARK2$(v, green)$ and MARK2$(w, green)$.
6: **end if**
7: **if** $color = blue$ **then**
8: MARK2$(v, blue)$ and MARK2$(w, blue)$.
9: **end if**
10: **end if**
11: **end if**



(a) unique match (b) no unique match

Fig. 4. Greedy method based on local sweep in three-hop neighborhood (*Greedy-Sweep-2*): Consider marking u as a new black node. When v is a neighbor of u and w is a neighbor of v, both have rank of 2, they now have only four combinations of possible positions. Node u can check which pair of positions uniquely satisfies the measurement. (a) There is a unique match. (b) There is no unique match.

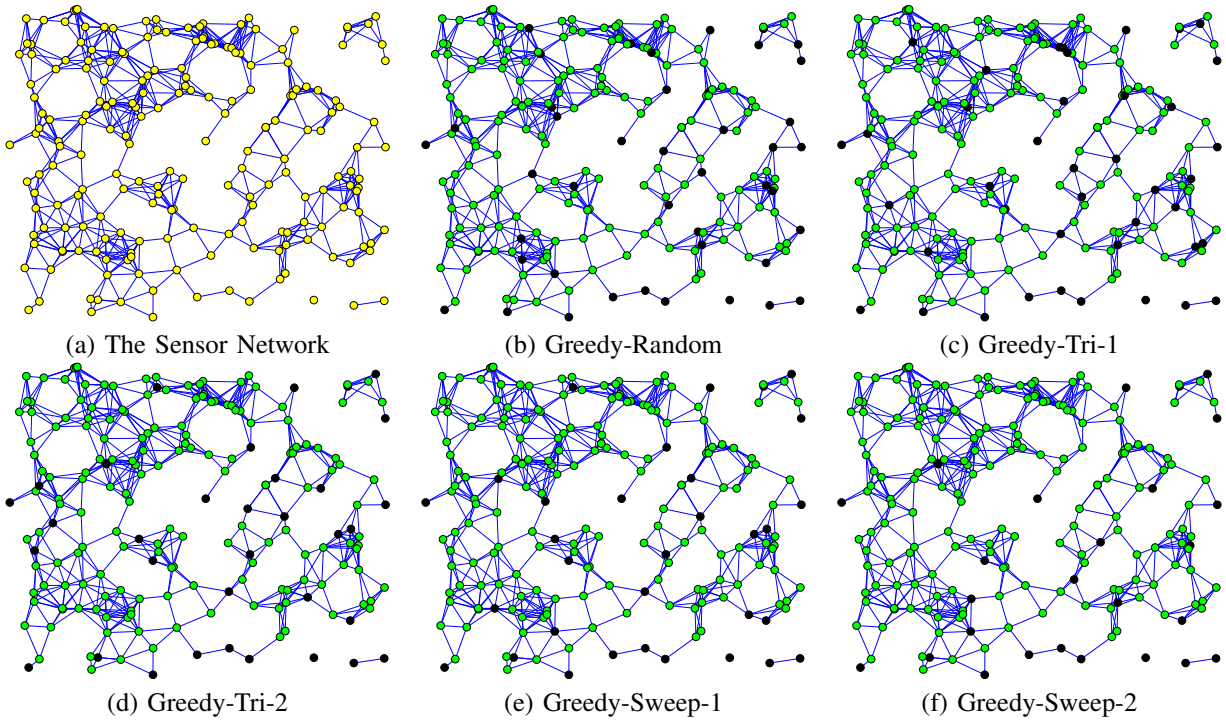
hop neighborhood when we process u. Figure 4 illustrates the examples. Node u will check its one-hop neighbor v and v's neighbor w about their possible positions. Algorithm 6 presents the corresponding MARK procedure used in this method. We call this method (consists of Algorithms 6,2,4) *Greedy-Sweep-2*. It is clear that in each step *Greedy-Sweep-2* can localize more sensors than previous methods. Therefore, the number of anchors used in its output is the least.

V. Simulation Results

To evaluate our proposed methods for MCLP, we conduct extensive simulations on random generated sensor networks. In our simulations, we deploy 50 to 1000 sensors uniformly in a 1200×1000 rectangle region. We use both unit disk graph model and random graph model to generate network topology (i.e., distance measurements among sensors). In unit disk graph model, we set the transmission range of each sensor as 80. If the distance between two sensors are smaller or equal to 80, we assume there is distance measurement between them. Figure 5(a) shows an example of such topology with 200 sensors. In the random graph model, whether there is an edge between a pair of nodes is decided randomly with a prefixed probability. In this case, the sensing range of each node may be various. For all simulation settings, we repeat the simulation for 100 times, the results presented here are the average results over these 100 simulations.

In our simulations, we implement five algorithms, namely, *Greedy-Random*, *Greedy-Tri-1*, *Greedy-Tri-2*, *Greedy-Sweep-1*, and *Greedy-Sweep-2*. In *Greedy-Random* algorithm, trilateration is recursively used to localize as many sensors as possible, and when there is no more nodes can be localized, a random node is picked to become the next anchor node, then this procedure is repeated until every sensor is localized. Thus, *Greedy-Random* is just like Algorithm 2 except for changing Line 10 to randomly pick a white node. The only metric of our evaluation is the number of anchor nodes (black nodes) selected by each algorithm. It is obvious that the less anchor nodes selected the better. Notice that it is impossible to obtain the optimal solution of MCLP for comparison even using the exhaustive search on all anchor sets, since checking whether a solution in MCLP can realize all sensors is still NP-hard.

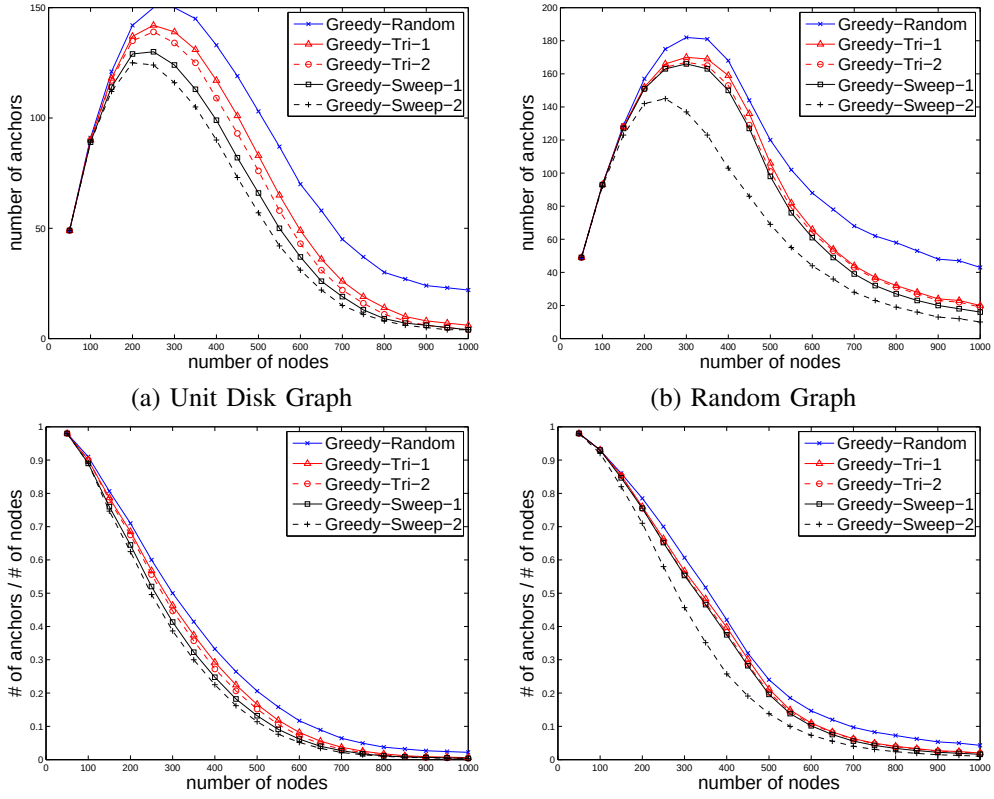
Figure 5 shows a group of results for all algorithms on the same unit disk graph with 200 sensors. In the results, black nodes are anchor nodes and green nodes are sensors whose location is realized by other nodes. In this particular example, *Greedy-Random* selects 43 anchor nodes, while our proposed greedy methods select 42, 34, 33, and 27, respectively. It is



(a) The Sensor Network (b) Greedy-Random (c) Greedy-Tri-1

(d) Greedy-Tri-2 (e) Greedy-Sweep-1 (f) Greedy-Sweep-2

Fig. 5. An example of MCLP: different algorithms generated different results. Here, black nodes are the anchors selected by our algorithms and green nodes are sensors whose locations are realized by localization algorithms. Running these five algorithms on this particular 200-node network, the number of black nodes in their results are 43, 42, 34, 33, and 27, respectively.



(a) Unit Disk Graph (b) Random Graph

(c) Unit Disk Graph (d) Random Graph

Fig. 6. Results of different algorithms of MCLP on different networks (the number of nodes increase from 50 to 1000). The upper row shows the average number of black nodes selected by the algorithm, while the lower row shows the average percentage of black nodes selected by the algorithm.

clear that these greedy algorithms perform better than pure greedy-random algorithm. In addition, using sweeping operation beyond the trilateration can improve the performance.

We run our algorithms on both unit disk graph model and random graph model. Notice that in random graph model there could be no distance measurement even between two nearby sensors. The results are plotted in Figure 6.

Figures 6(a) and 6(b) show the number of black nodes used by each algorithm for different models with different node densities. From these results, we have the following observations. First, in all the results, greedy algorithms with local sweep use less anchors than greedy algorithms with pure trilateration, and they all use less anchors than greedy-random algorithm. For example, with random network with 400 nodes, *Greedy-Sweep-2* uses 38% less anchors than *Greedy-Random* and 33% less anchors than *Greedy-Tri-2*. Second, if an algorithm uses more information (i.e., information from larger neighborhood), it can achieve better performance. For example, *Greedy-Sweep-2* is better than *Greedy-Sweep-1*. For random network with 400 nodes, *Greedy-Sweep-2* uses 31% less anchors than *Greedy-Sweep-1*. Third, when the network become denser, the number of anchors first increases and then decreases. This is reasonable. Initially, when the network is sparse, network with more sensors needs more anchor nodes to be localized. However, when the network becomes dense enough, the number of anchors will drop since most of sensors can be localized via their neighbors. Fourth, it is interesting that the improvement of sweep-based algorithms is more clear when the network is neither too sparse nor too dense. In addition, its improvement in random graphs is much larger than in unit disk graphs. This is due to the existence of special nonuniform substructure in random graphs.

It is also interesting to see the percentage of anchor nodes needed to localize a network. Figures 6(c) and 6(d) show the trend of such a percentage when the network density increases. It is clear that the percentage always decreases with increasing of density. In other words, less percentage of anchors nodes is needed for localization in denser networks. For example, when the network only has 100 nodes, over 90% of them need to be anchor nodes. But when the network has 1000 nodes, using only around 1% of them as anchor nodes can localize the whole network.

For all simulation results, the above conclusions are consistent for both unit graph model and random graph model. However, it is also clear from these results that random graph model needs more anchor nodes than unit graph model does.

Finally, instead of uniformly deployed sensor networks in a rectangle region, we consider sensor networks with special shapes or with different sizes of holes (as shown in Figure 7). The results are plotted in Figure 8. The performances of our algorithms are still consistent over these networks.

VI. Conclusion

In this paper, we formally define the minimum cost localization problem (MCLP) to find the minimum anchor set to localize the whole network. The problem is computationally challenging problem and has never been studied. We present four different greedy algorithms to find the anchor set for a given network. Extensive simulations have been conducted and demonstrated the efficiency of our algorithms. All proposed algorithms are given in centralized formats, however, they can be easily implemented in a distributed fashion (the propagation of localization is limited to a local region). We leave such implementations and their evaluations as our future work. In addition, finding more efficient algorithms which can achieve constant approximation of MCLP is also an interesting direction. However, such problem is an extremely challenging task since even to check whether a solution (a set of anchors) can realize all sensors is still NP-hard.

References

[1] J. Graver, B. Servatius, and H. Servatius, *Combinatorial Rigidity*, Graduate Studies in Math., AMS, 1993.

[2] T. Eren, D.K. Goldenberg, W Whiteley, Y.R. Yang, A.S. Morse, B.D.O. Anderson, and P.N. Belhumeur, "Rigidity, computation, and randomization in network localization," in *Proc. of IEEE INFOCOM*, 2004.

[3] D.K. Goldenberg, A. Krishnamurthy, W.C. Maness, Y.R. Yang, A.S. Morse, and A. Savvides, "Network localization in partially localizable networks," in *Proc. of IEEE INFOCOM*, 2005.

[4] J. Aspnes, T. Eren, D.K. Goldenberg, A.S. Morse, W Whiteley, Y.R. Yang, B.D.O. Anderson, and P.N. Belhumeur, "A theory of network localization," *IEEE Trans. on Mob. Comp.*, 5(12):1-15, 2006.

[5] A. Savvides, C.C. Han, and M.B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *ACM MobiCom*, 2001.

[6] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proc. of ACM Int'l W. on Wireless sensor networks & applications*, 2002.

[7] D. K. Goldenberg, P. Bihler, M. Cao, J. Fang, B.D. O. Anderson, A. Stephen Morse, and Y. Richard Yang, "Localization in sparse networks using sweeps," in *Proc. of ACM MobiCom*, 2006.

[8] Z. Yang, Y. Liu, and X.-Y. Li, "Beyond trilateration: On the localizability of wireless ad-hoc networks," in *Proc. of IEEE INFOCOM*, 2009.

[9] U. Khan, S. Kar, and J. Moura, "Distributed sensor localization in random environments using minimal number of anchor nodes," *IEEE Trans. on Signal Processing*, 57(5):2000-2016, 2009.

[10] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *Proc. of IEEE INFOCOM*, 2004.

[11] Y. Shang and W. Ruml, "Improved MDS-based localization," in *Proc. of IEEE INFOCOM*, 2004.

[12] K.-F. Ssu, C.-H. Ou, and H. C. Jiau, "Localization with mobile anchor points in wireless sensor networks," *IEEE Trans. on Vehicular Technology*, 54(3):1187-1198, 2005.

[13] C.-H. Wu, W. Sheng, and Y. Zhang, "Mobile sensor networks self localization based on multi-dimensional scaling," in *Proc. of IEEE Int'l Conf. on Robotics and Automation*, 2007.

[14] M. Erol, L. F. M. Vieira, and M. Gerla, "AUV-aided localization for underwater sensor networks," in *Proc. of Int'l Conf. on Wireless Algorithms, Systems and Applications*, 2007.

[15] M. Erol, L. F. M. Vieira, and M. Gerla, "Localization with Dive'n'Rise (DNR) beacons for underwater acoustic sensor networks," in *Proc. of ACM Workshop on Underwater Networks*, 2007.

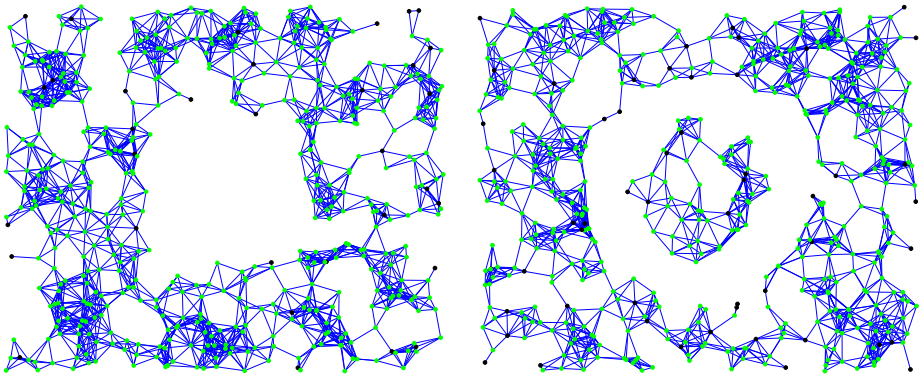
[16] B. Jackson and T. Jordan, "Connected rigidity martoids and unique realizations of graphs," *J. of Combinatorial Theory*, 94(1):1-29, 2005.

[17] A. Youssef, A. Agrawala, and M. Younis, "Accurate anchor-free node localization in wireless sensor networks," in *Proc. of IEEE Workshop on Information Assurance in Wireless Sensor Networks*, 2005.

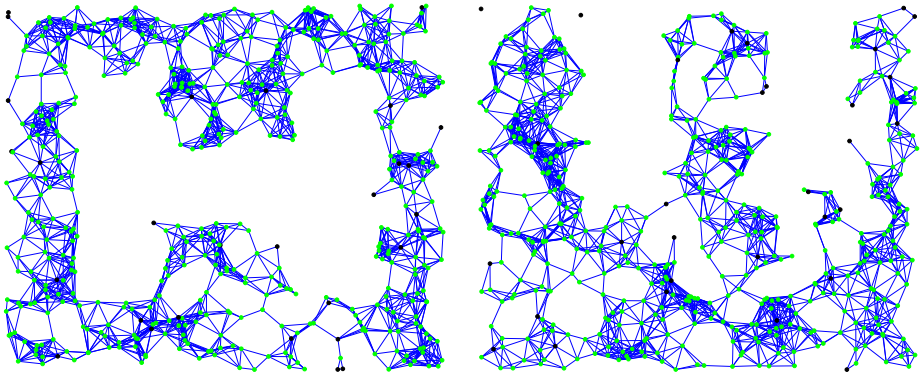
[18] H. Akcan, V. Kriakov, H. Brönnimann, and A. Delis, "Gps-free node localization in mobile wireless sensor networks," in *Proc. of ACM int'l workshop on data engineering for wireless and mobile access*, 2006.

[19] S. Lederer, Y. Wang, and J. Gao, "Connectivity-based localization of large-scale sensor networks with complex shape," *ACM Trans. Sen. Netw.*, 5(4):1-32, 2009.

[20] W. Shang, F. Yao, P. Wan, and X. Hu, "On minimum m-connected k-dominating set problem in unit disc graphs," *J Comb Optim*, 16:99-106, 2008.

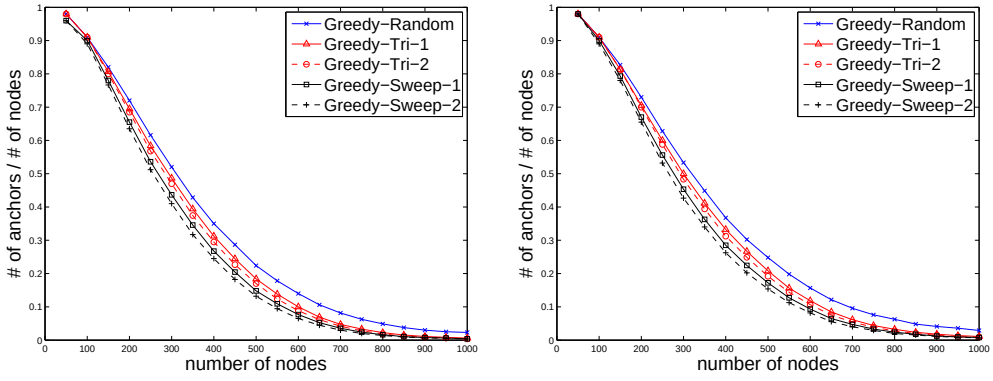


(a) Network with Square Hole (b) Network with "O" Hole

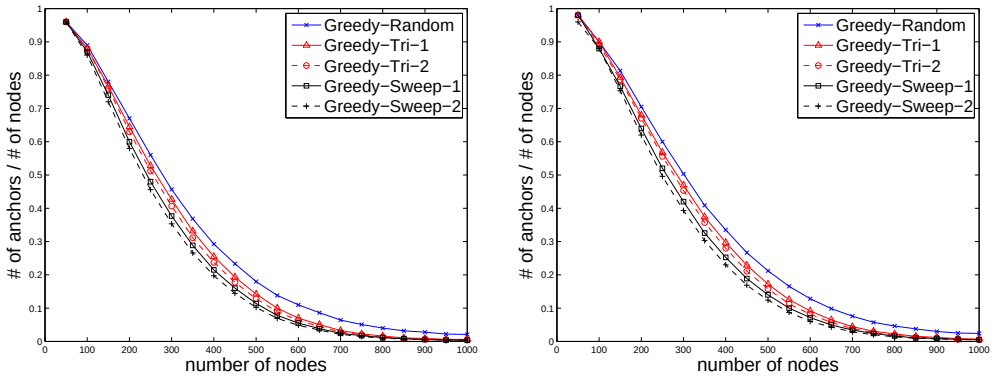


(c) Network with "H" Hole (d) "E"-Shape Network

Fig. 7. Different shapes of sensor networks used in the last set of simulations.



(a) Network with Square Hole (b) Network with "O" Hole



(c) Network with "H" Hole (d) "E"-Shape Network

Fig. 8. Results of different algorithms of MCLP on networks with different shapes (the number of nodes increase from 50 to 1000). Plots show the average percentage of black nodes selected by the algorithm.