An Efficient Search Algorithm for Multi-hop Network Localization in a Sparse Unit Disk Graph Model

Phisan Kaewpraphа[1], **Thaewa Tansarn**[2], and **Nattakan Puttarak**[3], Non-members

ABSTRACT

We consider a network localization problem by modeling this as a unit disk graph where nodes are randomly placed with uniform distribution in an area.The connectivity between nodes is defined when the distances fall within a unit range. Under a condition that certain nodes know their locations (anchor nodes), this paper proposes a heuristic approach to find a realization for the rest of the network by applying a tree search algorithm in a depth-first search manner. Our contribution is to put together a priori information and constraints such as graph properties in order to speed up the search. An evaluation function is formed and used to prune down the search space. This evaluation function is used to select the order of the unknown nodes to iterate. This paper also extends the idea further by accommodating a variety of other properties of graphs into the evaluation function. The results show that node degrees, node distances and shortest paths to anchor nodes drastically reduce the number of iterations required for realizing a feasible localization instance both in noise-free and noisy environments. Finally, some preliminary complexity analysis is also given.

Keywords: Localization, Tree Search, Unit Disk Graph, Wireless Sensor Networks

1. INTRODUCTION

In sensor network deployments and applications, geographical information along with measured data from the sensors, e.g., representing sensed data on a map is crucial. Measuring data such as traffic flow will be less meaningful if not geographically represented. It is common that stationary sensing nodes are placed in known locations. In different circumstances, sensor network deployments may be done in a random manner. For example, ad-hoc sensor nodes can be air dropped into some specific area. Determining their locations is not as simple as one would expect.

Establishing a communication network is certainly a challenging research area. However, finding the geographically location of each deployed device is as challenging as the communication part. Although devices could be GPS-equipped, the cost of doing so could be prohibitive in practical scenarios. Hence, a more realistic assumption is that some of the nodes know their locations while the majority of them do not. Each sensor node may or may not have a direct connection with the anchor nodes and often rely on multiple-hop communications to the anchors. This specific instance is known as a multi-hop network.

A single-hop localization such as a GPS system where location can be computed using a lateration technique (finding the intersection of circles drawn around reference nodes). However, a multi-hop network can be modeled and solved in different ways. Researchers have studied these kind of problems for a few decades and this is still a very active research areas [1-4], as will be discussed in the literature review section. It is also noteworthy that our studies are based on 2-dimensional space, although 3 or higher dimensions could employ our aproach.

There are several research directions and assumptions regarding network localization problems. Some approaches are based on real-word scenarios where physical properties such as noise and the nature of signal propagation are taken into account. Others work with simplified models so that fundamental understanding and knowledge can be derived [4-6]. Even if the problem is formulated in a simplified form where a network is represented as graph and distance measured is exact, finding a feasible solution can be difficult [7].

Our approach to investigate this problem started from an abstract setting where we model the problem as a connected graph in its simplest form. Then we examined heuristic techniques to tackle this kind of localization problem. Next, we added more realistic assumptions and constraints to develop real-world scenarios while maintaining acceptable complexity. In this paper, we propose a centralized algorithm capable of computationally solving a problem based on the use of a simple yet effective tree search algorithm.

The complexity of the original exhaustive search

Manuscript received on September 15, 2016 ; revised on March 6, 2017.

Final manuscript received on March 6, 2017.

[1,2] The authors are with Department of Electrical and Computer Engineering, Faculty of Engineering, Thammasat University, Thailand, E-mail: kphisan@engr.tu.ac.th, debugger_man@hotmail.com

[3] The author is with Telecommunications Engineering Department, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand, E-mail: kpnattak@kmitl.ac.th

space for a realized instance of a network can be improved by utilizing a priori knowledge of properties derived from the graph itself. The results show that a computationally prohibitive search space can be greatly reduced when some information from the graph is used to assist the search and eliminate some search paths sooner with high probability. The right order of search paths (or the order of traversal) significantly affects the converging time to the final solution.

Our method can be applied in several real-world scenarios. Since this algorithm works well in sparse wireless networks with a few anchor nodes and limited communication range, localization in an ultra-wideband sensor network is one of the deployment scenarios. Sensed data along with measured distances among nodes will be passed to a gateway which forwards data to a central server for post-processing. The data from sensors and locations will be extracted in a centralized manner.

The paper is organized as follows. We discuss some related work and background in Section 2. Then in Section 3 we describe our algorithm and the rationale behind the chosen properties and present simulation results in Section 4. Finally the discussion and conclusions are given in Section 5 and 6, respectively.

2. RELATED WORKS

Network localization, especially for wireless sensor networks, can be viewed as two-step processes. The first step is to obtain measured data regarding spatial relationships between nodes (their coordinates x, y). The relationship is naturally obtained in a polar form (r, θ). If we know both parameters in polar form, the solution is found. Hence the challenge is when only one of the parameters is available. For example, omni- directional antennas are unable to sense the direction or angle of a received signal.

Physically, the distances or angles can be obtained by different measurement techniques. It can be measured by the time of arrival of the propagated signal (TOA), by the received signal strength (RSS), by the angle of arrival (AOA) or by other means [4]. These measured data solely or with other types of information are the inputs for the different approaches used to infer the actual locations. We often call this process the network realization. Such techniques include tri-lateration, or bearing measurement [4-6].

In a more complex environment such as multi-hop network localization where multiple hop communication is required to reach the other end, connectivity and graphical properties such as hop-count are used as inputs to solve this problem, examples of this approach can be found in [8,11].

Apart from real-world scenarios, theoretical work has also been done [15-16] where the clean state of the network is studied and some fundamental theories have been established.

In both theoretical and physical problem settings for network localization, it can be universally defined as an optimization problem, where the discrepancy between the measured distances and the calculated distances from the realized instance is minimized. Direct approaches such as some variations of non-linear optimization techniques have also been proposed. The problem is also approximated as linear or convex optimization, which can be tackled by various techniques, e.g., maximum likelihood estimation, semi-definite programming, the spring mass model, belief-propagation and the message passing algorithm [8-14].

Our work, as mentioned earlier, falls into the area of a clean state of the network in its simplest form, while making use of links and connectivity information to solve the problem in a manner similar to [8-14], with a different approach. Our method is based on a tree search algorithm inspired by [17]. Our original proposed version can be found in [2] and that will be briefly explained in the next section. This paper is an extension of our previously published work [3]. In this study, we added preliminary analysis of the complexity bounds for our algorithm as well as open problems that need to be addressed.

2.1 Theory of Network Localization

We briefly discuss the theory of network localization in this section as it has some implication from the fundamental result that we can use to explain the effect of our heuristic approach. In [14], the network is modeled as graph, we follow the same approach. Below are some useful results:

- In network localization, the network can be solved if it has a unique realization. The condition is that the graph must be globally rigid. This is the case when we use only edge information. We showed in our previous work that this is not a necessary condition if we use implicit information about non-connected nodes [1].
- Solving a unit disk graph localization is NP-hard, however our approach can reduce the number of iterations on average.
- If a graph is dense enough, it will be solved efficiently using a tri-lateration method. This implies that our approach is useful for a sparse network.

In the next section we explain our method and improvements that increase the speed of finding a solution.

3. TREE SEARCH LOCALIZATION

3.1 Problem Formulation

We model a group of communication devices forming a network by establishing communication to neighboring devices as a graph G consisting of a set

of vertices (we use node interchangeably) V and a set of edges (we use connection or link) E connecting between two nodes. $V(i)$ refers to an i^{th} node labeled as node i, where i is an integer from 1 to n. $E(i,j)$ represents an edge between nodes i and j.

Additionally, we assume that locations are aligned in a 2-dimensional integer coordinate system where node positions are integer numbers. Hence the distance squared will also be an integer. This assumption enables us to apply a search algorithm with a finite number of branches. Granularity and complexity are the trade-offs that needs to be considered. So far, small to medium sized networks with 200 or fewer participating nodes are studied. Nodes are placed in a random uniformly distribution in a unit square area. Two nodes establish connectivity if and only if they are within a certain distance r apart from each other. We define $D(i,j) = |(E(i,j))|$ to be the distance between nodes i and j obtained from the measurements and let $X(i)$ denote the 2-dimensional coordinate for a node i. Then $|X(i) - X(j)|$ represents the calculated Euclidian distance between i and j according to the instance of the realization. We can establish a realization of a graph as an optimization problem with constraints as follows:

Minimize:

$$\Sigma \| X(i) - X(j)| - |E(i,j)\|, \; E(i,j) \; \in \; E \qquad (1)$$

Subject to:

$$X(i) = x, y \qquad (2)$$

for some nodes i that are in a set of anchor nodes.

Our search algorithm seeks an exact solution where $||X(i) - X(j)| - |E(i,j)|| = 0$. In the other words, $|X(i) - X(j)| = |E(i,j)|$. Fig.1 depicts an example of our network instance. The grid size is 20x20 units with 40 randomly placed nodes. The radius is 5 units in distance.

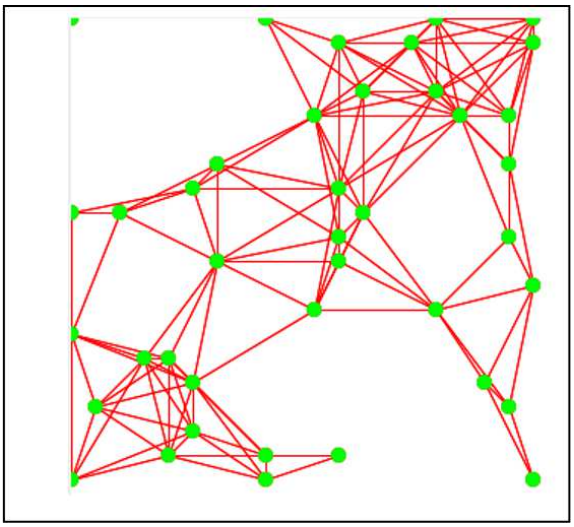

Fig.1: *An instance of random unit disk graph.*

3.2 Tree Search Algorithm

Proposed in [1], the algorithm can be briefly described as follows: Starting from a set of anchor nodes, called a "realized network instance", while the other nodes that are not placed into this realized instance are kept in a separate set. The search tree starts from this instance. The algorithm traverses the search tree by adding a new node that is not in the set and checks if this new node is compatible with the objective function in Eq.(1), e.g., $|X(i) - X(j)| - |E(i,j)| = 0$ if they are neighbors and $|X(i) - X(j)| > 1$ if they are not neighboring nodes.

If the new node is not compatible, then the algorithm discards the instance and moves on to the next path in the tree in a depth-first search style. If the node is successfully placed, violating no constraint, the new node will be added to the current instance of the realized graph. The algorithm continues until all nodes have been localized. Table 1 summarizes the method. Starting with a set of anchor nodes and let $R\{V\}$ denote a set of nodes that are realized. $R\{V\}$ is empty initially (or containing only anchor nodes) and realized nodes will be added to the set as the algorithm tries to place nodes into feasible locations that are consistent with measured data. Another set called $I\{V\}$ represents unrealized nodes. During each iteration, the algorithm picks a node from $I\{V\}$ according to the $Eva(u)$ function, which is used to select a node with a better chance of forming a final feasible solution as the search algorithm continues.

The algorithm relies on the probability that a new node selected will be more likely to converge to the final solution faster, while discarding the poor search paths sooner. Our algorithm lies on the same concept like decision tree used in computer chess games [17]. We iterate through the search tree and eliminate infeasible branches, while applying an evaluation function to increase the chance of eliminating infeasible branches sooner. Hence, this speeds up the search. It is similar to chess or board games where an evaluation function is used to select the best move to win. However, our evaluation function is used to select a move that will likely result in reaching the solution faster (e.g., fewer iterations or walk-throughs). Another difference is that our algorithm is a depth-first search that is efficient in term of memory usage, while games may require breadth-first search approach.

Our contribution in this paper is mainly the development of better ways to exploit intrinsic/implicit properties of a graph. Then, we use these properties as part of an evaluation function to determine the order of nodes being added or placed during the process of constructing a graph. Properties of graphs such as connectivity, node-to-node distance, connectivity counts from a node to its neighbors (node's degree) are used as a mean for further enhancement/speeding up the search algorithm. We also give out preliminary

analysis of the complexity of this algorithm.

Table 1: *Tree-search algorithm [1].*

1. Start with all known nodes in a set $R\{V\}$ which represents realized nodes and $I\{V\}$ contains nodes which are waiting to be realized.
2. Sort nodes in $I\{V\}$ in descendant order according to the $Eva(u)$ and choose a node u with the largest value of $Eva(u)$.
3. Generate all feasible realizations of u centered at a neighboring node v in $R\{V\}$, add all the instances to the search tree and put u in $R\{V\}$
4. Repeat step 2 until all nodes in $I\{V\}$ are realized (empty).

3.3 Search Constraints

In a conventional problem formulation for network localization, the only constraint used for realizing the network is the connectivity and distances such as $||X(i) - X(j)| - |E(i,j)|| = 0$. We included an assumption that the implicit information such as no connection between nodes can imply that the nodes are not within the range and this can be used to discard iterations of graph instances that conflict with this constraint, e.g., $|X(i) - X(j)| > 1$. A similar approach is done in [13] by transforming this information into virtual links.This a priori constraint significantly reduces the size of the search space, especially for sparse networks. A similar concept was applied to urban outdoor localization of mobile devices where buildings are excluded from feasible locations for the devices.

3.4 Graph Properties

We look into graph properties that lead to the selection of a new node with high probability of eliminating unsuccessful paths. Those properties were examined and applied individually. These properties are defined for a given node that is being placed:

- Nc: Number of links whose end nodes locations are known. We consider a new node that has more links to the known/placed nodes by counting neighboring nodes whose locations have been realized.
- D: Degree of a new node: similar to Nc but considers all neighboring nodes regardless of their realization status by counting the number of neighbor nodes, including those that are not realized.
- Sd: Sum of distances from the realized nodes.
- Ss: Sum of shortest paths between a new node and all anchor nodes.

Fig. 2, 3, and 4 demonstrate these parameters. Fig. 2 depicts the case where node A is chosen to be placed because it has more connectivity to the instance of the realized network rather than a node B with more total connectivity. Fig. 3 shows that the node degree of B is greater than A. Hence, B will be examined first. Fig. 4 shows that the distance in the realized graph from A is greater than B. Thus, A is chosen first.

There are other possible properties such as the sum of the shortest paths to all nodes or to all realized nodes. Max/min or average or count of those parameters can also be studied. Such parameters will be studied extensively in our future work.

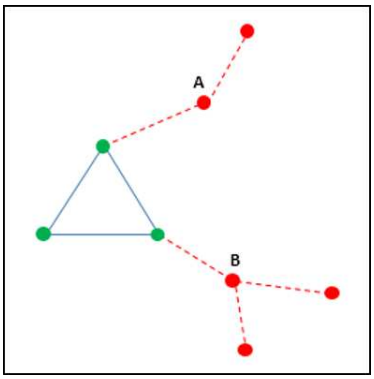

Fig.2: *Example of most connected node to a realized graph.*

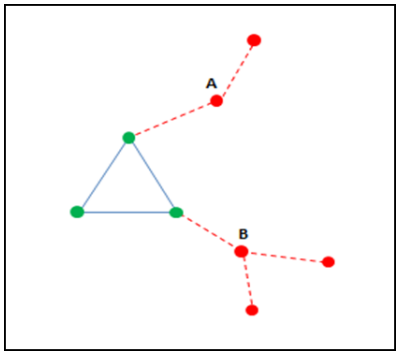

Fig.3: *Example of node degree preference.*

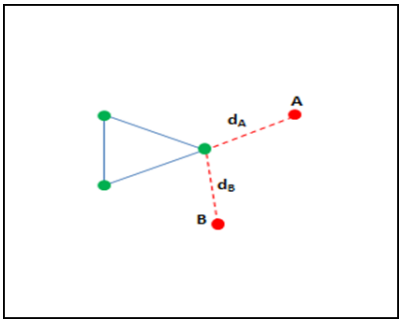

Fig.4: *Distances between unrealized nodes and realized graph.*

3.5 Evaluation Function

Our evaluation function is in a very general form, the sum of the functions of each individual property.

$$Eva(n) = \Sigma Fi(Pi) \quad (3)$$

where $Eva(n)$ is the evaluation function of a node n. $Fi()$ is a function of a parameter Pi. We currently use the simplest form by employing a linear weight to each parameter. By including the parameters studied in this paper, we obtained the following evaluation function.

$$Eva(n) = wl * Nc + w2 * D + w3 * Sd + w4 * Ss \quad (4)$$

where $w1, w2, w3$ and $w3$ are the coefficients for those parameters. Any node n with a higher $Eva(n)$ will be placed into a realized graph first.

3.6 Tree-search Algorithm in Noisy Measurement

We also consider the case where noise is part of the measurement. Thus, the measured distances are contaminated by noise. The original search algorithm, unfortunately, will not be able to find the exact solution because the added distance causes the error in the measured data and the distance obtained while placing a node into the graph during the runtime of the tree-search algorithm.

We modify the search algorithm by relaxing the constraint. As noise varies, we varied the threshold t so that the magnitude of the error between the measured distance and the calculated distance is within this threshold t. Any search result that is compatible with this new relaxed constraint will be regarded as a feasible solution.

Previously, under the ideal assumption that noise-free distance measurement is obtained, it is a matter of how much complexity is needed to find the exact solution (if a unique solution exists) or the first feasible solution found (in case there are ambiguities as the graph is not rigid or additional information is insufficient). In the latter case when noise is present, even though the feasible solution is obtained after performing the algorithm, there is no guarantee of an exact solution.

Fig. 5 depicts the extended bound t where the red dotted circles represent the original possible search boundary according to Eq. (1) and (2). The solid blue circles from inner to outer regions represent an area in which an integer coordinate will be sought. Relaxation can be added into the equation as follows:

$$\Sigma||X(u) - X(v)| - d(u,v)| \ \leq \ t \quad (5)$$

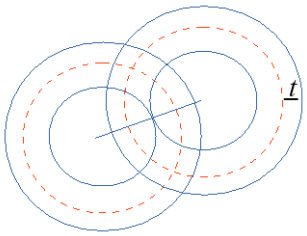

Fig.5: *Extended Search Boundaries with a threshold t.*

4. SIMULATION SETUPS AND RESULTS

We prepared and performed the simulations for both noise-free and noisy environment as follows:

4.1 Simulation Setup

The search space of any random network can be huge and there is no known analytical closed-form results regarding the performance of this approach. Thus we demonstrate the idea that the algorithm is able to operate efficiently in a real-word scenario for a moderate network size. We tested the algorithm by implementing the search in parallel using multi-threading on multiple processing cores available on modern CPUs. We simulated our algorithm on a system equipped with an Intel i7-5960X consisting of 8 computing cores and 16 logical threads running at 4.0 Ghz with 32GB of RAM. The algorithm was implemented using Java JDK 1.8. Each of the simulation settings was run multiple times. Eevery time, a new random instance graph was used.

We simulate 1 million samples of graphs per setting. The number of 1 million experiments was obtained in a Monte Carlo fashion. The average running time for each experiment ranged from 10 minutes to 21 hours.

To extend the effect of our algorithm, we choose to work with a sparse network similar to the one depicted in Fig. 1. Assuming that the unit length is in metric system (meter), we simulate the network in an area of 15×15 m containing 37 nodes with each node had communication range of 1 m. We put 4 anchor nodes at the corners of the graph. Firstly, we ran our algorithm with randomly placed nodes; the average performance was 30,000 iterations. This was because the graph is sparse.

The simulation was done for each individual parameter as well as for the combined evaluation parameters. The theoretical lower bound for the number of iterations was 33 (as there were 33 nodes). We counted placement of a new node as 1 iteration.

4.2 Simulation Setup in Noisy Environments

Similarly, in a noisy environment, we performed a simulation using the same settings with a node radius of 4. There were 4 anchors nodes at the corners and

we also add 2 anchor nodes randomly along with other ordinary nodes. We modified our evaluation function as in equation (4) by normalizing each parameter of the evaluation function, the maximum value of each parameter of each instance of the random network.

$$Eva(n)=w1\frac{Nc}{|Nc|\max}+w2\frac{D}{|D|\max}+w3\frac{Sd}{|Sd|\max}+w4\frac{Ss}{|Ss|\max} \quad (6)$$

Under this new equation, we then found a new set of weights and parameters and chose the ones that performed best before applying them in a noisy environment.

4.3 Performance of Individual Parameters

By running each individual parameter separately, we obtained the results shown in Fig. 6. The parameter Nc effectively reduced the number of iterations. The rationale behind this is that a node having more connected links to the already-realized nodes had a smaller valid search space. Other parameters may not be obvious. For example, the sum of the distances can be interpreted as follows: placing a new node that is further away allows the algorithm to expand and cover the entire graph faster. This helps to eliminate the infeasible moves sooner. The Shortest Path parameter (Ss) did not perform as well as expected when it was used alone. The parameter, Node Degree (D), performed better than the Ss, however, if D was used alone, it may cause the algorithm to pick a new node with a higher node degree that is not relevant to the current instance of the graph being realized. This is because D also counts links that are connected to unrealized nodes.

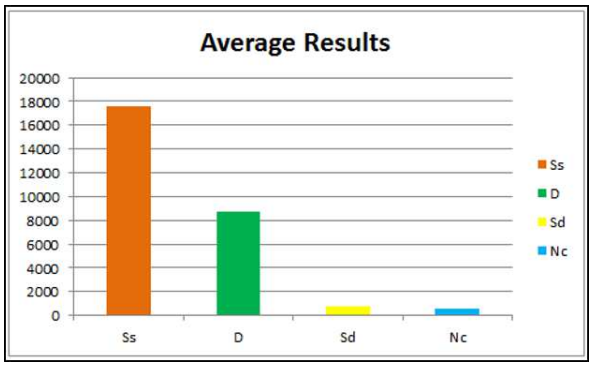

Fig.6: *Average Iterations for Individual Parameters.*

4.4 Performance of Combined Evaluation Function

Next, we use combined evaluation functions. A coefficient is assigned from the observed data from the previous simulation. Since the parameter Nc performed well, we assign more weight to it followed by the other parameters D, Sd and Ss accordingly. We assign the weight $w1, w2, w3$ and $w4$ values of 4, 2, 1 and -0.1 respectively.

The parameter Ss (sum of the shortest paths) had a negative weight because a shorter path implies that it is closer to anchor nodes and that helps to reduce the number of search paths. The results showed that combining parameters indeed improved the overall performance though the gain diminished as shown in Fig.7. Moreover, Nc alone required on average 530 iterations. When adding D (node degree), the performance improved and the number of iterations was reduced to 390. Similarly, adding Sd (sum of distance) brought the total iterations down to 298, whereas Ss (sum of shortest paths) reduced it further to 290. Hence the complexity was reduced by 46% compared to using a single parameter.

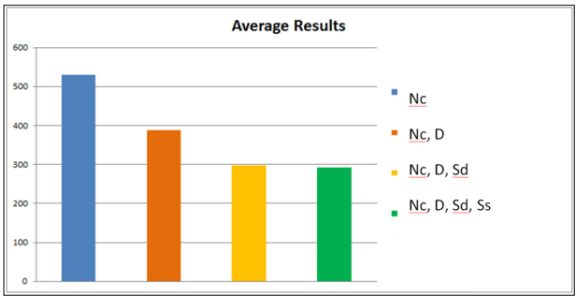

Fig.7: *Average Iterations for Combined Parameters.*

It is notable that when a parameter is used individually, even though it performs better than a randomly select node, it performs poorer than using them in a combined evaluation function.

4.5 Performance under Noisy Measurement

As shown in [18], with the evaluation function and search algorithm presented in Table 1, we varied the weight of the function to find its optimum, i.e., the value that provides the lowest average number of iterations. In Table 2, the optimal iterations obtained was 85 with $w1$=0.8, $w2$=1.4 and $w4$= -0.06. The combined evaluation function performed significantly better than each individual parameter alone. This weights were used as we investigated the effect of noise on the algorithm.

We looked into two performance metrics that affected by the additive noise. These are the increases in the average number of iterations and the algorithm's capability of finding a feasible solution.

For simplicity, we assumed that the noise during the measurement followed a normal distribution so we applied additive Gaussian noise. Eq. (5) can be written as follows:

$$d(u,v) = |E(u,v)| + N(0,\sigma^2) \quad (7)$$

We chose Gaussian distributed noise because the

Table 2: *Average iteration at various weight ratios of Eva(u).*

Nc	***Sd***	***SS***	Iterations			
			Mean	Min	Max	SD
1	0	0	141	34	22000	446
0	1	0	7745	34	138000	32700
0	0	1	14000	34	75600	621000
0.8	1.4	-0.06	85	34	127	1511

measurement was done over a wireless communication channel and a Gaussian distribution is common. The other channel characteristics such as fading could be considered. We plan to investigate this in the future.

We used the optimum weight from Table 2 as noise varied, then we modulated the relaxation threshold, t, against σ^2 and t was proportional to the measured distanced (e.g., σ^2 varied from 1% to 6% of the distance, d). The results are shown in Table 3 [18].

Table 3: *Average Iterations at various noise levels.*

***t* (threshold)**	σ^2	Average iterations	% Solution found
0.1	0.01-0.03	96	100
0.1	0.04	1154	19.6
0.1	0.05	103	0.4
0.2	0.04	175	99.8
0.2	0.05	186	95.2
0.2	0.06	214	73.5
0.2	0.07	205	37.3
0.2	0.08	186	11.2
0.2	0.09	104	1.9
0.2	0.10	-	0
0.3	0.09	1148	57.3
0.3	0.10	1054	31.9

According to the result, when the noise level was relatively small i.e., $\sigma^2[0.01, 0.03]$, the relative threshold, e.g., $t = 0.1$ was sufficient. t allows the algorithm to tolerate some level of uncertainty. Thus, the algorithm reached a feasible solution eventually. There was a trade-off as we notice that there was a slight increase in the number of iterations. As the noise level increased, the success rate of finding consistent realization dropped drastically. The increase in the complexity was also noticeable when $\sigma^2 = 0.5$ and $t = 0.1$. If we increased the threshold to support the increase in noise, e.g., $t = 0.2$, this allowed the solution to be found. The patterns continued with $t = 0.3$, which can tolerate more noise with a significant increase in complexity.

5. DISCUSSION

In our experimental results, the algorithm was capable of finding solutions. It should be noted that we were using the assumption that the coordinates of the location are in an integer domain, hence allowing us to discretize the search space while running the search algorithm. If the coordinate is in fact embedded with real number coordinates, then this algorithm can still be mapped into our settings by assuming that the quantized real value location causes some additional error due to the truncation of real coordinates to an integer coordinate system. A trade-off between granularity and complexity will have to be made.

It is nothworthy that since our algorithm was based on centralized processing, the communication overhead was minimal as the information regarding location could be transmitted along with measured data.

We also note that graph parameters used as evaluation functions were chosen based on the idea that nodes with more connectivity, larger distances and closer proximity to the anchor nodes are more likely to be placed correctly and the infeasible choices can be eliminated earlier by evaluating $Eva(u)$. We found that our current method was useful primarily for sparse graphs. In dense graphs, the problem can be solved more efficiently using other algorithms such as the conventional tri-lateration method.

In this section, a preliminary approximation of its complexity is given. The lower bound of this algorithm is equal to the number of nodes, in the other words, we can place one node per iteration. The upper bound is when the first node has k possible coordinate points to be placed without considering any constraints. Then the worst case is expressed on the order of $O(k^n)$ where k is the number of integer points lying inside the outer and inner boundary t shown in Fig.5. k can be estimated by the following relation between integer points inside a circle [18].

$$N(r) = \pi r^2 + E(r) \tag{8}$$

where $N(r)$ is the number of integer points inside a circle with radius r. E(r) is small error with an upper and lower bound.

$$k = N(r+t) - N(r-t) = 4\pi rt + E(r+t) - E(r-t) \tag{9}$$

Equation (9) represents an estimation of a loose upper bound of the complexity of the algorithm.

Deriving the upper bound from the radius alone still results in a loose boundary. We would like to give a conceptual sketch to develop a better upper bound. Consider any node in a network. It is possible to find the shortest path between one end from its adjacent node to an anchor node at the other end as

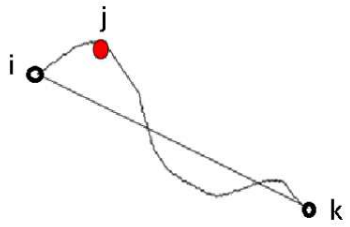

Fig.8: *Node is a member in a shortest path between 2 ends.*

depicted in Fig. 8. Let $s(i,j,k)$ be the shortest path from node i,j and k. Normally $s(i,j,k) \leq d(i,k)$. The actual path i to k via node j is usually longer than the direct distance between i and k. The parameters $d(i,k)$ and $s(i,j,k)$ provide another bound to the search space of node j. Then, this particular node j can no longer be placed freely within the vicinity of one of its anchor node. It will be bounded inside an area under the triangles formulated from $d(i,k)$ and $s(i,j,k)$ as depicted in Fig. 9.

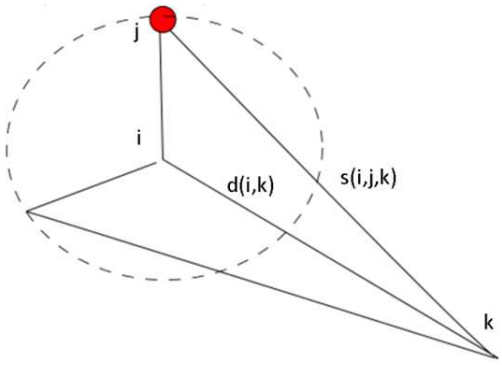

Fig.9: *Max stretched distance while placing node.*

The expression for the upper bound developed based on this idea should be further investigated in our future work. Our primary finding is that the average distance between two random nodes in a unit square is approximately 0.368 [20]. And there is a bound regarding the maximum shortest path (diameter) of a unit disk graph. This diameter is bounded by [21]:

$$(1 + 0((\ln \ln n / \ln n)^{1/d}))/\lambda \qquad (10)$$

where λ is the maximum radius of the unit disk. Once the 3 sides of the triangle are known or estimated, the bound can be formulated. Nevertheless, the simulation results show that the average complexity is far better than the worst case. Hence finding tighter bounds is also an open problem, which is another interesting problem to be investigated in the future.

One of our directions for the future work consists of utilizing a rigid sub graph where localization can be done efficiently and to build a graph in a bottom up manner. Another direction is to find a tighter analysis of the current complexity of this algorithm. At this moment, our results may not be directly compared with other works since our problem formulation and focus is slightly different. We are developing our algorithm for the purpose of developing a general and more realistic model. At the same time, we may add more useful constraints and conditions that are already embedded in the properties of the graph itself, but have not been utilized.

6. CONCLUSIONS

We propose an evaluation function to be used in our tree search algorithm for solving network localization problems. An evaluation function is a function derived from a combination of graph properties. The results show that those properties, when applied individually, can reduce the number of iterations. The number of connections to the realized graph, Nc, significantly reduces the number of iterations required. Furthermore, the combination of the parameters D, Sd and Ss can significantly reduce the average number of iterations in noise-free and noisy environments by 46%. In the case of noisy environments, the threshold,t, can be adjusted for the algorithm to tolerate noise at different levels. We also provide an outline for complexity analysis.

ACKNOWLEDGEMENT

This research was partially supported by Faculty of Engineering, Thammasat University, Thailand. We thank the editors at ECTI-CIT who assisted us during the submission process. We would also like to thank all the reviewers for their valuable comments.

References

[1] P. Kaewprapha, J. Li and N. Puttarak, "Network Localization on Unit Disk Graphs," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, Houston, TX, USA, pp. 1-5, 2011.

[2] P. Kaewprapha, "Localization in Wireless Sensor Networks: Solvability Improvement Technique Using Priori Information from Sensing Data and Network Properties in Unit Disk Graph Model", *Advanced Materials Research*, Vols. 931-932, pp. 999-1003, 2014.

[3] P. Kaewprapha, and N. Puttarak, "Network Localization using Tree Search Algorithm: A heuristic search via graph properties", *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Chiang Mai, Thailand, pp.1-5, 2016.

[4] L. Cheng, C. Wu, Y. Zhang, H. Wu, M. Li, and C. Maple, "A Survey of Localization in Wireless

Sensor Network," *International Journal of Distributed Sensor Networks*, January 2012.

[5] G. Mao, B. Fidan and B. D.O. Anderson, "Wireless sensor network localization techniques," *Computer networks*, Vol.51, Issue 10, pp. 2529-2553, 11 July 2007.

[6] A. Pal, "Localization algorithms in wireless sensor networks: Current approaches and future challenges," *Network Protocols and Algorithms*, Vol. 2, No. 1, pp.45-73, 2010.

[7] H. Breu and D. G. Kirkpatrick, "Unit disk graph recognition is NP-hard," *Computational Geometry*, Vol. 9, Issues 1, pp.3-24, Jan. 1998.

[8] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," *Telecommunication Systems*, pp. 267-280, 2003.

[9] A. T. Ihler, J. W. Fisher, R. L. Moses and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," in *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809-819, April 2005.

[10] S. Lee, H. Woo, and C Lee, "Wireless sensor network localization with connectivity-based refinement using mass spring and Kalman filtering," *EURASIP J Wireless Commun. and Networking*, pp. 1-11, 2012.

[11] J. Xiang and W. W. Tan, "An improved DV-hop algorithm based on iterative computation for wireless sensor network localization," *2013 IEEE International Workshop on Electromagnetics, Applications and Student Innovation Competition*, pp.171-174, 1-3 Aug. 2013.

[12] S. Ji, K. F. Sze, Z. Zhou, A. M. C. So and Y. Ye, "Beyond convex relaxation: A polynomial-time non-convex optimization approach to network localization", *2013 Proceedings IEEE INFOCOM*, Turin, pp. 2499-2507, 2013.

[13] G. Oliva, S. Panzieri, F. Pascucci and R. Setola, "Sensor Networks Localization: Extending Trilateration via Shadow Edges," in *IEEE Transactions on Automatic Control*, vol.60, no.10, pp.2752-2755, Oct. 2015.

[14] Nguyen C, Georgiou O, et al. Maximum likelihood based multihop localization in wireless sensor networks. In: Communications (ICC), 2015 IEEE International Conference on. IEEE; 2015. p. 6663-6668.

[15] 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," in *IEEE Transactions on Mobile Computing*, vol.5, no.12, pp.1663-1678, Dec. 2006.

[16] S. V. Pemmaraju and I. A. Pirwani, "Good quality virtual realization of unit disk graphs," *Springer Berlin Heidelberg*, pp. 311-322, 2007.

[17] M. Campbell, A. J. Hoane , F. Hsu, "Deep Blue," *Artifical Intelligence*, Vol. 134, Issues 1, pp. 57-83, 2002.

[18] Kaewprapha, P., Puttarak, N., Tansarn, T., "Multi-hop network localization in unit disk graph model under noisy measurement using tree-search algorithm with graph-properties-assist traversing selection," KKU-IENC, Proceeding of, Khon Kaen, Thailand, August 2016.

[19] G.H. Hardy, *Ramanujan. Twelve Lectures on Subjects Suggested by His Life and Work*, 3rd ed., Chelsea, New York, 1999, pp.67.

[20] Santaló, L. A., "Integral geometry and geometric probability. Encyclopedia of Mathematics and its Applications," Vol. 1. Addison-Wesley Publishing Co., Reading, Mass.-London-Amsterdam, 1976.

[21] R. B. Ellis, J. L. Martin, and C. H. Yan, "Random geometric graph diameter in the unit ball," *Algorithmica* , pp. 421-438, 2007.

Phisan Kaewprapha received his Ph.D. in Electrical Engineering from Lehigh University (USA), in 2012. He is a lecturer at the Department of Electrical and Computer Engineering, Faculty of Engineering, Thammasat University Thailand. His area of interests includes computer and data communication networks, network localization, wireless networking, parallel and distritbuted computing, big data and machine learning.

Thaewa Tansarn received his Bachelor of Engineering degree in Electronics and Telecommunications Engineering from Rajamangala University of Technology and is currently a master degree student at the Department of Electrical and Computer Engineering, Faculty of Engineering, Thammasat University, Thailand. He also worked as a software engineer at Toyota Tsusho Electronics Thailand. His research interests includes network localization, microcontroller and embededed systems.

Nattakan Puttarak received her Ph.D in Electrical Engineering from Lehigh University (USA), in 2011, for her work in coding for data storage. She is currently a lecturer at the Telecommunications Engineering Department, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand. Her research interests are in coding theory in the area of data storage, including both the mainstream systems of hard drives and the emerging technology of flash drives. This includes designing new error correction coding strategies and decoding algorithms to combat disk failure and recover lost data for small-scale disk arrays as well as large-scale data centers, analyzing their performance, and identifying best practices.