KKU ENGINEERING JOURNAL 2016;43(S1):114-117 Research Article

KKU Engineering Journal

https://www.tci-thaijo.org/index.php/kkuenj/index

Multi-hop network localization in unit disk graph model under noisy measurement using tree-search algorithm with graph-properties-assist traversing selection

Phisan Kaewprapha*[1)], Nattakan Puttarak[2)] and Thaewa Tansarn[1)]

[1)]Department of Electrical and Computer Engineering, Faculty of Engineering,
Thammasat University, Pathum Thani, Thailand.
[2)]Department of Telecommunications Engineering, Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand.

Received April 2016
Accepted June 2016

Abstract

This paper proposes a heuristic approach to efficiently traverse through search space in wireless localization in unit disk graph model. The main idea is to accommodate graph characteristics as metrics for branch selection. Selecting appropriate branching order eliminates infeasible branches quicker than random selection. We show that graph properties such as connectivity, measured distances, and shortest paths to anchor nodes (nodes with known-locations), can drastically reduce the number of iterations (branching) required to traverse thru the possible realization of the wireless network. A normalized weight-sum function of those parameters is used as an evaluation function in selecting branching direction. We extensively perform experiments to find good weights for evaluation function of those graph characteristics. Additionally, we apply this algorithm in a more realistic environment where noise is observed during the measurement of distances. We use a modified version of our algorithm by relaxing feasible constraints to tolerate more discrepancy allowing error within a threshold. A tradeoff between complexity and the probability of finding the feasible solution is shown. The results show that the adding error does increase the complexity while maintaining the ability to find solution within a timely manner.

Keywords: Localization, Unit disk graph, Tree-search, Multi-hop network

1. Introduction

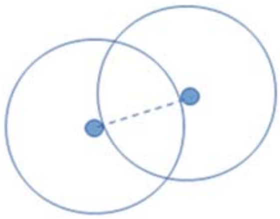
Network localization is a problem of obtaining geometric positions given some characteristics or network properties such as connectivity, distances, angles whose values are also obtained from physical measurements, for instance, signal strength, angle of arrival, time of arrival, etc. Knowing network's location is essential, especially in wireless sensor networks (WSNs). The sensed data may be meaningless if the geometric information is missing. Conventional usage of GPS may not be practical in some cases; multi-hop network, with some nodes are anchors and the rest of the nodes are unknown, are preferable settings and much gaining attraction.

The importance of multi-hop network localization, in general, draws attention not only from wireless sensors area but also researchers from related fields such as computation geometry, graph theory, optimization mainly to develop algorithms aiming at realizing a network, with difference objectives, for example, accuracy, robustness, and efficiency. Researchers work in a vast spectrum of problem formulations and assumptions, from the very theoretical ideal settings to more realistic assemble of real world application [1-8]. A well-known algorithm such as DV-hop [6] simplifies the problem settings and utilizes only hop count to estimate the distance. Since this problem can be formulated as an optimization problem (exact formulation shown in the next section) the problem itself has been shown as an NP-hard even at its simplest form, aka noise-free environment, much of the efforts are put into finding heuristic or approximation with certain bounds and certain objectives. [8] proposes an upper and lower bound with respect to the quality of the virtual realization. [2] relaxes the problem into convex optimization. [7] utilize virtual edge from implicit information such as non-connectivity relationship is also an important info. [3] focuses on the tree-search algorithm by utilizing explicit and implicit of graph information to accelerate tree-search algorithm by mapping locations into integer coordinate.

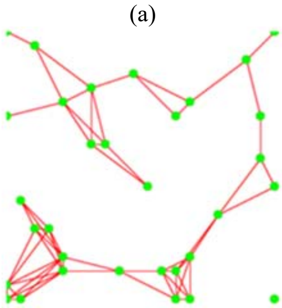
This paper proposes a heuristic approach to efficiently traverse through search space in wireless localization in unit disk graph model (as depicted in Figure 1). Graph characteristics such as connectivity, sum of distances and shortest paths to anchor nodes are used as metrics for branch selection. By ordering and selecting the right branching direction, infeasible branches are eliminated earlier as the search proceeds. We perform a simulation to find good weighting values for evaluation function of those graph characteristics. We then apply this algorithm in a more realistic environment where noise is observed during the

*Corresponding author. Tel.: +6684 726 5175
Email address: kphisan@engr.tu.ac.th
doi: 10.14456/kkuenj.2016.35

measurement of distances. Our approach is detailed in the next section.



(a)



(b)

Figure 1 (a) Unit disk graph modeling and (b) an instance of a uniformly distributed embedding network into 2-dimensional space

2. Localization in unit disk graph model

2.1 Formal definition

Unit disk graph is one of the widely-used models for wireless networks. A randomly placed network can be modeled as a graph consisting of vertices (nodes) V and edges E. An edge exists when two node u and v are within the distance r, e.g. $|E(u,v)| \leq r$. $|E(u,v)|$ is the actual distance (without noise) between u and v. Let $d(u,v)$ be the measured distance from the realization of node u and v; $d(u,v) = |E(u,v)|$ in the ideal case, without noise during the measurement. And $|X(u) - X(v)|$ is the distance calculated from an instance of realization of node u and v. Then the realization (solution) can be found by solving the following optimization problem:

$$\min \sum \big|\, |X(u)-X(v)| - d(u,v) \big| \text{ for all } E(u,v) \text{ in } V$$

Subject to some u,v with known locations $X(u,v) = x,y$ (1)

The feasible solution in the ideal solution is when

$$\sum \big|\, |X(u)-X(v)| - d(u,v) \big| = 0 \quad (2)$$

This solution always exists in the ideal case. We also assume that the locations of nodes are aligned to integer coordinate system. We develop our search algorithm under these assumptions. Note that constraints obtained from prior knowledge according to real-world scenarios can be added; for example, in urban area, navigation system may exclude area where buildings are located.

2.1.1 Tree-search algorithm

The search algorithm relies on the optimism that the exhaustive search (depth-first search) with proper branch selection using graph characteristics can eliminate branches that are infeasible early. The algorithm starts with a set of nodes with known locations denoted as a localized set *Re{V}*. At one of the anchor nodes, the algorithm expands the search tree by trying to add a neighboring node until all nodes are added to the *Re{V}* set. We define a function *Eva(u)* as a weight-sum of the following properties at node *u*

$$Eva(u) = w_1.DC(u)/|DC(u)|_{max} + w_2.SC(u)/|SC(u)|_{max} + w_3.SH(u)/|SH(u)|_{max} \quad (3)$$

wherew_1, $w_2$and w_3 are weight whose summation is 1. *DC(u)* is node degree, *SC(u)* is a sum of the distances and *SH(u)* is a sum of the shortest paths; all three parameters are calculated toward neighboring nodes of a node *u* that are in *Re{V}*. These parameters are also normalized by the maximum value among all *u* not in *Re{V}*.We also define a function *Ver(u)* that verifies the consistency between the realization *X(u)* versus the measured distances $|E(u,v)|$ for all v, where v are neighboring nodes of u that are in a realized set *Re{V}*, must meet the equation (2). The algorithm works as follows:

Table 1 Tree-search algorithm [3]

1. Start with all known nodes in a set *Re{V}* which will represents realized nodes and *Im{V}* contains nodes which are waiting to be realized.
2. Sort nodes in *Im{V}* in dependence order according to the *Eva(u)* and choose a node *u* with largest value of *Eva(u)*.
3. Generate all feasible realization of *u* centered at a neighboring node *v* in *Re{V}* and add all the instances to the search tree and put *u* in *Re{V}*
4. Repeat step 2 until all nodes in *Im{V}* are realized.

The algorithm terminates successfully as long as the first feasible solution is found during the branching through the search tree and return *Re{V}* as the solution. Note that the first solution found does not necessary imply the correct solution if a graph is not rigid; our results in [3] show that it is with high probability that the first solution found is the exact solution or close to the exact location. This is our key idea on reducing the complexity of the search algorithm. In section 3, we show some variation of weighting in equation (3) for best results.

2.1.2 Tree-search algorithm in noisy measurement

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 side information is insufficient). In a real-world scenario where noise is part of the measured data, however, exhausting thru all branches likely yields no feasible solution. We mitigate this issue by modifying our algorithm to accommodate inner to outer searching bounds by some certain threshold t depicted in Figure 2 where red dot circle represents conventional possible search space according to equation (2) whilst solid blue circles from inner to outer ones represent an area where integer coordinate will be searched. This relaxation can be added into the equation as follows:

$$\sum \big|\, |X(u)-X(v)| - d(u,v) \big| \leq t \quad (4)$$

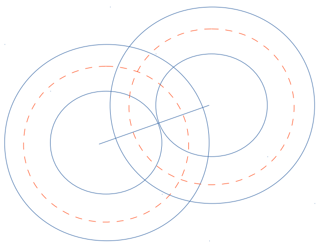


Figure 2 Extended search area bounded by inner and outer circles

3. Simulation setups and results

We perform simulation on a square area of size 15 by 15 unit with 36 nodes randomly placed within the area with communication radius of 4. Among 36 nodes, there are 4 anchors nodes at the corners and 2 anchors are placed randomly within the area. We first iterate through all possible weight and put some of important weight ratio to demonstrate the effectiveness of proper weight choices. We measure the iterations in the algorithm in Table 1; one iteration equals to a loop in step 2-4. This pseudo iteration is hardware dependence. We run each experiment against 1 million randomly-generated graph instances. Table 2 shows the results from variable weight. Extreme cases are shown when *DC*, *SC* and *SH* are 1. We also iterate through all possible weight mixture with 0.02 increments. Only the best weight results are shown in the table. Similar results are obtained when $w_1>w_2>w_3$. This is particularly making sense since the *Eva(u)* function is used for ordering and choosing a node. Since *DC*, the node degree, is the most effective metric, weight should be given to this parameter the most. The other parameters are supplemental when *DC* gives equal result.

Table 2 Average iteration at various weight ratio of *Eva(u)*

DC	*SC*	*SH*	Average iterations
1	0	0	141
0	1	0	7745
0	0	1	14000
0.8	1.4	0.06	85

Next we add additive Gaussian noise so that the measured distance includes noise

$$d(u,v) = |E(u,v)| + N(0, \sigma^2) \quad (5)$$

We use the best weight from Table 2 and vary the relaxing threshold *t* and noise level σ^2 as some proportion of *d* (e.g. σ^2 varies from 1% to 6% of the distance *d*). The results are shown in Table 3.

When noise is relatively small $\sigma^2 \in [0.01, 0.03]$ the threshold t=0.1 is capable of finding the solution with a slight increase of the average iterations. At $\sigma^2=0.5$, $t=0.1$ however, the probability of finding consistent realization drops drastically. Increasing the threshold t=0.2 does allow the solution to be found again with the increase of the average iterations. The patterns follow with t=0.3.

4. Discussion

In our experimental results, even though the algorithm is capable of finding solutions. It should be noted that we are using the assumption that the coordinate of the location is in integer domain; hence allowing us to discretize the search space and running search algorithm. If the coordinate is in fact embedded in real number coordinate, then this algorithm can still be applied with additional error from truncating real coordinate to integer coordinate system and a tradeoff between granularity and complexity will have to be taken into account. We also like to note that graph parameters used as evaluation function are chosen based on intuition that node with more connectivity, larger distances and closer to anchor nodes are more likely to be placed correctly and infeasible choices can be eliminated earlier when evaluated against *Ver(u)*. We found that our method is suitable for sparse graph as dense graph can benefit from conventional lateration method. Our future work is to utilize rigidity sub graph where localization can be done efficiently and build a graph in a bottom up manner. At this moment, our results may not be able to compare directly with other works in the area because our problem formulation and focus is slightly different; we are developing our algorithm to be capable of resolving general and more realistic model.

Table 3 Average Iteration 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

Regarding the complexity of this algorithm the worst case is in the order of $O(k^n)$ where *k* is the number of integer points lying inside the outer and inner boundary *t* shown in Figure 2. *k* can be estimated by the following relation between integer points inside a circle [9].

$$N(r) = \pi r^2 + E(r) \quad (6)$$

Where *N(r)* is the number of integer points inside a circle with radius *r*. And E(r) is small error with upper and lower bound.

Thus *k* takes the following form

$$k = N(r+t) - N(r-t) = 4\pi rt + E(r+t) - E(r-t) \quad (7)$$

Nevertheless, the simulation results show that the complexity is far from the worst case and the average case is another interesting problem to be investigated in the future.

5. Conclusions

We propose a heuristic ordered tree-algorithm to tackle multi-hop network localization by modeling the problem in integer coordinate unit disk graph model. Evaluation function serves as branch selection as the algorithm iterates through branches in the searching space. The right mixture of parameters used in evaluation function gives relatively impressive outcome and out performs single parameters. Noise-added environment is also studied; the relaxed version of the algorithm with appropriate threshold at certain noise level is able to obtain solution while maintaining the complexity.

6. References

[1] Aspnes J, Eren T, Goldenberg DK, Morse AS, Whiteley W, Yang YR, et al. A theory of network localization. IEEE Transactions on Mobile Computing 2006;5(12):1663-1678.

[2] Ji S, Sze KF, Zhou Z, So AMC, Ye Y. Beyond convex relaxation: A polynomial-time non-convex optimization approach to network localization. In: Cao G, Capone A, Subramaniam S, editors. Proceedings IEEE INFOCOM, 2013; 2013 Apr 14-19; Turin, Italy. New Jersey: IEEE; 2013. p. 2499-2507.

[3] Kaewprapha P, Li J, Puttarak N. Network localization on unit disk graphs. In: Zhang X, editor. Global Telecommunications Conference (GLOBECOM 2011); 2011 Dec 5-9; Houston, USA. New Jersey: IEEE; 2011. p. 1-5.

[4] Mao G, Fidan B, Anderson BD. Wireless sensor network localization techniques. Computer networks 2007;51(10):2529-2553.

[5] Nguyen C, Georgiou O, et al. Maximum likelihood based multihop localization in wireless sensor networks. In: Gomes N, Manikas A, editors. 2015 IEEE International Conference on Communications (ICC); 2015 June 8-12; London, UK. New Jersey: IEEE; 2015. p. 6663-6668.

[6] Niculescu D, Nath B. DV based positioning in ad hoc networks. Telecommunication Systems. 2003;22(1-4):267-280.

[7] Oliva G, Panzieri S, Pascucci F, Setola R. Noisy localization over unit disk graphs: The shadow edge approach. In: Anand D, editor. American Control Conference (ACC 2015); 2015 July 1-3; Chicago, USA. New Jersey : IEEE; 2015. p. 4436-4442.

[8] Pemmaraju SV, Pirwani IA. Good quality virtual realization of unit disk graphs. Journal of Computational Geometry 2011;2(1):69-91.

[9] Hardy GH, Ramanujan: Twelve Lectures on Subjects Suggested by His Life and Work, 3rd ed. Chelsea, New York: Chelsea Pub Co.; 1999.