Quantitative Uncertainty-Based Incremental Localization and Anchor Selection in Wireless Sensor Networks

†Zhiheng Xie, ‡Mingyi Hong, †Hengchang Liu, †Jingyuan Li
‡Kangyuan Zhu and †John A. Stankovic
†Computer Science Department, University of Virginia
‡System Engineering Department, University of Virginia
{zx3n, mh4tk, hl4d, jl3sz, kz3y, stankovic}@virginia.edu

ABSTRACT

Previous localization solutions in wireless sensor networks mainly focus on using various techniques to estimate node positions. In this paper, we argue that quantifying the uncertainty of these estimates is equally important in practice. By using the quantitative uncertainty of measurements and estimates, we can derive more accurate estimates by better fusing the measurements, provide confidence information for confidence-based applications, and know how to select the best anchor nodes so as to minimize the total mean square errors of the whole network. This paper quantifies the estimation uncertainty as an error covariance matrix, and presents an efficient incremental centralized algorithm—INOVA and a decentralized algorithm—OSE-COV for calculating the error covariance matrix. Furthermore, we present how to use the error covariance matrix to infer the confidence region of each node's estimate, and provide an optimal strategy for the anchor selection problem. Extensive simulation results show that INOVA significantly improves the computation efficiency when the network changes dynamically; the confidence region inference is accurate when the measurement number to node number ratio is more than 2; and the optimal anchor selection strategy reduces the total mean square error by four times as much as the variation-based algorithm in best case.

Categories and Subject Descriptors

I.6.4 [**Computing Methodologies**]: SIMULATION AND MODELING—*Model Validation and Analysis*

General Terms

Algorithms

Keywords

wireless sensor network, localization, uncertainty

1. INTRODUCTION

MSWiM'11, October 31–November 4, 2011, Miami, Florida, USA.

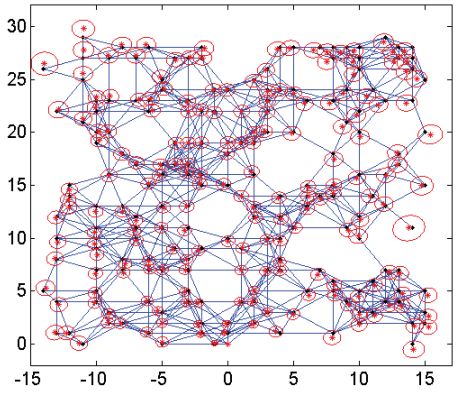

Figure 1: Confidence-based Localization (at 90% confidence level)

Localizing nodes in a wireless sensor network is an important problem that has produced many solutions. Typically, each solution makes various assumptions and provides different levels of accuracy. However, most of these solutions suffer from two common drawbacks: they do not quantify the uncertainty of measurements, and they do not provide quantitative uncertainty for their location estimates. The former problem causes the estimation to be non-optimal, while the latter results in the impossibility for inferring the confidence information at the node level. A preferred localization scheme is shown in Figure 1, where not only the location estimates (the red "*"), which fuse measurements according to their quantitative uncertainty, are provided, but also the confidence region (the red ellipses) for each node is rendered.

In this paper, we emphasize the importance of quantitative uncertainty for three reasons. First, by taking the quantitative uncertainty of the measurements into account, we can achieve better localization results. For example, in Figure 2(a), Nodes 1, 2 and 3 are the anchor nodes, Node R is to be localized and its real location is at the blue point. The arrow lines are the measurements, and the dashed circles represent the uncertainty of these measurements. If we do not take the uncertainty into account and just average these three measurements (trilateration and triangulation fall into this category), the estimate of Node R would be $E1$; but if we weight the accurate measurements more and the inaccurate ones less (by using the relative measurement graph model as this paper does), the estimate would be $E2$, which is more accurate than $E1$.

Second, the quantitative uncertainty can provide confi-

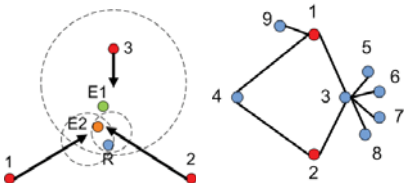

Figure 2: (a) Uncertainty Based Localization; (b) Anchor Node Selection

dence regions for confidence-based applications, which would take different actions based on different confidence-levels. Take a firefighter retreat selection application for example [**?**]. If a WSN is dynamically deployed in a building on fire to help select a safe retreat route (by sensing the surrounding environment), the system would only choose a safe route if it is within a 95% confidence-level region.

A third usage of quantitative uncertainty is to guide the selection of anchor nodes. Anchor nodes are those whose global locations are precisely known, e.g. either by GPS or manual input. Many WSN localization solutions (including our method) rely on a few anchor nodes to deduce other non-anchor node positions. However, in a large scale WSN, which nodes should be set as anchors so that the entire network's localization accuracy can be significantly improved? If the quantitative correlated uncertainty (e.g. the covariance matrix) for all nodes' location estimation is provided, the nodes with both large estimation variation and high correlations with other nodes should be selected. Take Figure 2(b) for example. Given the quantitative uncertainty, we would choose Node 3 as an anchor because it has both large variation (although not largest) and high correlation to other nodes, otherwise we may intuitively choose Node 4 since it is farthest away from the anchor nodes 1 and 2.

This paper models the quantitative uncertainty as the error covariance matrix for both measurements and estimates, and uses the Relative Measurement Graph (RMG) [2, 3] to calculate the location estimates and the covariance matrix. This model assumes that a set of relative position measurements (both the relative distances and angles) and their corresponding error covariance matrices are known. A RMG can be built, where the vertex set is all the sensor nodes, the directed edge set is all the node pairs which have relative measurements, and the edge function is to map each edge to its corresponding measurement and the error covariance matrix. Then, the location estimates and the corresponding covariance matrix for all these estimates can be derived by the Best Linear Unbiased Estimator (BLUE) [13].

Although the BLUE method mathematically solves the localization with quantitative uncertainty, its high computational complexity ($O(mn^2d^3)$, m is the measurement number, n is the node number and d is the variable dimension) and one-time localization scheme prevents it from being used in a large network, where measurements are incrementally generated and the number of nodes dynamically changes. Additionally, it does not provide any distributed algorithm to calculate the error covariance matrix, which is required by a peer-to-peer architecture.

This paper not only solves the above calculation problems, but also shows how to make use of a covariance matrix to infer the confidence region and help anchor selection. In summary, the contributions of this paper are: first, a more efficient incremental centralized algorithm–INOVA is developed, which reduces the computational complexity from $O(mn^2d^3)$ to $O(n^3d^3)$, and keeps a fixed computational time as measurements are incrementally generated. Second, by extending the location estimation distributed algorithm—OSE [2], a decentralized algorithm for calculating the error covariance matrix, named OSE-COV, is provided. The advantage of OSE-COV is that it reuses the scheme of the OSE algorithm, so that the location estimates and the corresponding error covariance matrix can be simultaneously obtained by using the same protocol. Third, we show that the location estimates derived by BLUE follow the multivariate normal distribution, and thus the confidence region can be derived based on the estimate and the corresponding covariance matrix. Finally, an optimal anchor selection algorithm is derived. It decides which node should be selected as an anchor so that the total mean square error of the whole network is minimized. Note that all above work is generic, and can be used for other problems, such as time synchronization [12](1D) or motion consensus [2] (3D) problems.

2. OPERATIONAL SCENARIO

Since the details of our solution are quite mathematical, we begin by presenting an operational scenario. Consider an expedition team that moves through a large forest or other wild area dropping sensor nodes. Once deployed, each of these nodes communicates with neighbors and using a technique such as TDOA [22, 20] and AOA [18, 20] to create a set of measurements and the corresponding error covariance matrices (each node can infer the error covariance matrix based on its device's accuracy property which is pre-studied, e.g. the error variation could be constant or proportional to the measured distance, and x-axis's variation could be independent with y-axis's) for the relative position between them. All these measurements and the corresponding error covariance matrices are wirelessly transmitted to the base station. Then using the RMG model and the INOVA algorithm, the location estimates and the corresponding error covariance matrix for all these nodes are obtained. If the above centralized architecture is not available in some place, the distributed algorithms OSE and OSE-COV can be used to estimate the location and the covariance matrix, respectively.

After deployment, nodes may continuously generate more relative measurements, and some may move, leave or join the network at run time, then INOVA can quickly respond to these dynamic behaviors, and update the estimates in real time. If some confidence-based application needs to know nodes' real location region at some specific confidence level, a confidence region picture like Figure 1 (the red part) can be rendered based on the inference theory in Section 6. Furthermore, based on each node's estimation error covariance matrix, we can judge whether the current estimation accuracy is good or not. If it is not good enough, we can set one or more nodes to be anchor nodes by using the optimal anchor selection strategy in Section 7, so that the total mean square error of the network is guaranteed to be minimized after each selection.

After the basic RMG model described in Section 3, we provide our centralized algorithm INOVA and the distributed algorithm OSE-COV in Sections 4 and 5, respectively. The confidence region inference and the anchor selection pro-

blem are presented in Sections 6 and 7. The evaluations of all above conclusions are shown in Section 8, followed by the related work and the conclusion in Sections 9 and 10.

3. RELATIVE MEASUREMENT GRAPH

Estimating each node's vector-valued variable based on a set of noisy linear relative measurements, such as problems of time synchronization [12], 2D or 3D WSN localization [2], or motion consensus [2], can be solved by the relative measurement graph model. Assume the vector-valued variable for each node is $x_i \in \mathbb{R}^d, i = 1, 2, \cdots, n$, and a set of independent relative measurements about these variables are obtained, denoted as $\zeta_{uv} = x_u - x_v + \epsilon_{uv} \in \mathbb{R}^d, u, v \in \{1, 2, \cdots, n\}$, where $\epsilon_{uv} \in \mathbb{R}^d$ is the random error vector with zero mean and a covariance matrix $P_{uv} = E[\epsilon_{uv}\epsilon_{uv}^T] \in \mathbb{R}^{d\times d}$, representing the measurement uncertainty. By stacking all the node variables into one vector $\mathbf{x} = [x_1^T, x_2^T, \cdots, x_n^T]^T \in \mathbb{R}^{nd}$, all the measurements into one vector $\mathbf{z} = [\zeta_1^T, \zeta_2^T, \cdots, \zeta_m^T]^T \in \mathbb{R}^{md}$ and all the measurement errors into a vector $\epsilon = [\epsilon_1^T, \epsilon_2^T, \cdots, \epsilon_m^T]^T \in \mathbb{R}^{md}$, the measurement equations can be expressed as

$$\mathbf{z} = \mathcal{A}^T\mathbf{x} + \epsilon \tag{1}$$

where $\mathcal{A} = \mathbf{A} \otimes \mathbf{I}_d$, $\mathbf{A}$ is the incidence matrix of the relative measurement graph G, $\mathbf{I}_d$ is $d \times d$ identity matrix, and $\otimes$ denotes the Kronecker product. The relative measurement graph $G = (V, E, F)$ is a directed graph, constructed by including all the nodes as its vertex set V, all the node pairs which have measurements as its directed edge set E (the direction is from u to v if the node pair has the measurement of ζ_{uv}), and a function $F : E \to (Z, P)$ which maps an edge $e_{uv} \in E$ to the corresponding measurement $\zeta_{uv} \in Z$ (Z is the set of all measurements) and the error covariance matrix $P_{uv} \in P$ (P is the set of all error covariance matrices). The incidence matrix $\mathbf{A}$ is a $n \times m$ matrix, where each row corresponds to each node, each column corresponds to each edge. The element a_{ue} of A is 1 if Edge e leaves from Node u, -1 if e is directed toward u, and 0 if Node u is not involved in Edge e. For example, the left figure in Figure 3 can be expressed as

$$\underbrace{\begin{bmatrix}\zeta_1\\ \zeta_2\\ \zeta_3\\ \zeta_4\\ \zeta_5\end{bmatrix}}_{\mathbf{z}} = \underbrace{\begin{bmatrix} I & -I & 0 & 0\\ I & -I & 0 & 0\\ 0 & I & 0 & -I\\ 0 & I & -I & 0\\ 0 & 0 & -I & I\end{bmatrix}}_{\mathcal{A}^T} \underbrace{\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\\ \epsilon_4\\ \epsilon_5\end{bmatrix}}_{\epsilon}$$

By partitioning $\mathbf{x}$ into an anchor node variable vector $\mathbf{x}_r$, which includes all anchor nodes' constant positions, and a non-anchor node variable vector $\mathbf{x}_b$, which includes all non-anchor nodes' position variables, and partitioning $\mathcal{A}$ into an anchor node incidence matrix and a non-anchor node incidence matrix, Equation 1 can be rewritten as

$$\overline{\mathbf{z}} = \mathcal{A}_b^T\mathbf{x}_b + \epsilon \tag{2}$$

where $\overline{\mathbf{z}} = \mathbf{z} - \mathcal{A}_r^T\mathbf{x}_r$. For the example in Figure 3, since Nodes 1 and 4 are anchors, the equation can be written as

$$\underbrace{\begin{bmatrix}\zeta_1 - x_1\\ \zeta_2 - x_1\\ \zeta_3 + x_4\\ \zeta_4\\ \zeta_5 - x_4\end{bmatrix}}_{\overline{\mathbf{z}}} = \underbrace{\begin{bmatrix} -I & 0\\ -I & 0\\ I & 0\\ I & -I\\ 0 & -I\end{bmatrix}}_{\mathcal{A}_b^T} \underbrace{\begin{bmatrix}x_2\\ x_3\end{bmatrix}}_{\mathbf{x}_b} + \underbrace{\begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\\ \epsilon_4\\ \epsilon_5\end{bmatrix}}_{\epsilon}$$

Obtaining an estimate of $\mathbf{x}_b$ becomes a classical estimation problem, which can be solved by BLUE. If G is a weakly connected graph (where there always exist undirected paths between any node pair), the best estimate for $\mathbf{x}_b$ in the linear combination space of all the measurements is uniquely determined by $\hat{\mathbf{x}}_b$, the solution of the following linear system

$$\mathcal{L}\hat{\mathbf{x}}_b = \mathbf{b} \tag{3}$$

where $\mathcal{L} = \mathcal{A}_b\mathcal{P}^{-1}\mathcal{A}_b^T$ (called *G's Kirchhoff Matrix*), $\mathbf{b} = \mathcal{A}_b\mathcal{P}^{-1}\overline{\mathbf{z}}$, and $\mathcal{P}$ is a block diagonal matrix consisting of all the covariance matrices P_i of $\epsilon_i, i = 1, 2, \cdots, m$. The covariance matrix of the estimation errors is given by

$$\mathbf{\Sigma} = \mathcal{L}^{-1} \tag{4}$$

Although the above centralized solution is elegant, it has several drawbacks. First, the time complexity is high. Assuming there are n nodes and m measurements, the anchor number is constant, and the dimension of the variable vector is d, the time complexity for calculating the covariance matrix $\mathbf{\Sigma}$ is $T(\mathbf{\Sigma}) = O(n^2md^3)$, since $T(\mathcal{L}) = T(\mathcal{P}^{-1}) + T(\mathcal{A}_b\mathcal{P}^{-1}) + T((\mathcal{A}_b\mathcal{P}^{-1})\mathcal{A}_b^T) = md^3 + nmd^3 + n^2md^3 = O(mn^2d^3)$, and $T(\mathcal{L}^{-1}) = n^3d^3$ ($n <= m$). The time complexity for calculating nodes' location estimates is $T(\hat{\mathbf{x}}_b) = O(n^2md^3)$, since $T(\hat{\mathbf{x}}_b) = T(\mathcal{L}^{-1}) + T(\mathbf{b}) + T(\mathcal{L}^{-1}\mathbf{b}) = (md^3 + nmd^3 + n^2md^3 + n^3d^3) + nmd^3 + n^2d^3 = O(n^2md^3)$. So the time complexity for calculating both $\hat{\mathbf{x}}$ and $\mathbf{\Sigma}$ simultaneously is $T(\hat{\mathbf{x}}_b, \mathbf{\Sigma}) = T(\hat{\mathbf{x}}_b) = md^3 + 2nmd^3 + 2n^2d^3 + n^3d^3 = O(mn^2d^3)$. The computational complexity depending on the measurement number m makes this solution not scale well, since m would be a very large number as the node number increases (the possible different edge number of a graph is of the order of $O(n^2)$, and the same edge may be measured multiple times in practice).

Second, this method follows a one-time calculation scheme, which means it calculates $\hat{\mathbf{x}}_b$ and $\mathbf{\Sigma}$ for a fixed RMG, if the RMG is changed, all the calculations must be re-executed. However, in practice, it is very likely that new measurements are continuously generated, and sensor nodes may leave or join the network dynamically. Therefore, if the network size is large, the high one-time calculation time complexity would prevent it from providing the latest estimation in real time.

4. THE INCREMENTAL NODE-VOLTAGE ANALYSIS METHOD

To overcome the drawbacks of high computational complexity and one-time calculation scheme of the above method, we propose a new calculation scheme, called the incremental node-voltage analysis (INOVA) method, which derives the same results as BLUE, but reduces the time complexity from $O(mn^2d^3)$ to $O(n^3d^3)$, and can incrementally calculate $\hat{\mathbf{x}}_b$ and $\mathbf{\Sigma}$ as new measurements are generated, or nodes join or leave the network dynamically. Although this

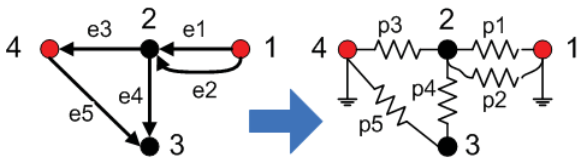

Figure 3: The Analogy Between RMG and The Generalized Electrical Network. The red nodes are anchor nodes.

method can be derived by analyzing the structure of matrices in Equation 3, we choose to deduce it based on the electrical network analogy for the following two reasons: 1) it can provide more insights of the decomposition operations; and 2) our subsequent distributed algorithm in Section 5 will also use this analogy.

It has been shown in [3] that a RMG $G = (V, E, F)$ is analogous to a generalized electrical network $\mathcal{G} = (V, \mathcal{E}, \mathcal{F})$, where V is the node set the same as in G, $\mathcal{E}$ is the edge set the same as E, but with no directions, and $\mathcal{F} : \mathcal{E} \rightarrow \mathcal{R}$ is an edge function that assigns each edge e_i a matrix valued resistance R_i which is numerically equal to the measurement error covariance matrix P_i in RMG. The generalized current J is defined as a $d \times d$ matrix associated with an edge with certain direction, and satisfies the Kirchhoff's Current Law, i.e. for each node in a generalized electrical network, the net generalized current flowing out or into that node is $\mathbf{0}$. The generalized voltage potential difference U_{uv} between two nodes u and v is defined as a $d \times d$ matrix $U_{uv} = R_{uv} \times J_{uv}$. Similarly, it satisfies Kirchhoff's Voltage Law, i.e. for any loop in a generalized electrical network, the sum of the generalized voltage potential difference in the clockwise or the counterclockwise direction is $\mathbf{0}$. Like the traditional electrical network, each node is associated with a voltage potential if there are currents or voltages imposed to the generalized electrical network. For the anchor nodes in $\mathcal{G}$, they are considered being connected to the ground, and thus their voltage potentials are always $\mathbf{0}$.

One important conclusion from [3] is that each node's estimation covariance matrix Σ_{ii} (the ith diagonal block element of network's covariance matrix $\mathbf{\Sigma}$) is numerically equal to the effective resistance R_i^{eff} between that node and the ground in $\mathcal{G}$. Hence, to calculate Σ_{ii}, we can impose an identity generalized current $\mathbf{I}$ to that node, then Node i's voltage potential U_{ii} is numerically equal to R_i^{eff}, i.e. $U_{ii} = R_i^{eff} \times \mathbf{I} = R_i^{eff} = \Sigma_{ii}$. We call the generalized electrical network imposed by a current $\mathbf{I}$ on Node i as *Node i's Current Graph, denoted as $\mathcal{G}^i$*. Therefore, to calculate Σ_{ii}, we can use node-voltage analysis method from electrical theory, i.e. set n_b voltage potential variables $\mathbf{U}_i = [U_{i1}^T, U_{i2}^T, \cdots, U_{in_b}^T]^T$ for every non-anchor node in $\mathcal{G}^i$, and then build the balance equations based on Kirchhoff's Current Law for each node.

$$\underset{dn_b \times dn_b}{\mathbf{L}} \underset{dn_b \times d}{\mathbf{U}_i} = \underset{dn_b \times d}{\mathbf{H}_i} \tag{5}$$

where $\mathbf{L} = [L_{uv}] \in \mathbb{R}^{dn_b \times dn_b}, u, v \in \{1, 2, \cdots, n_b\}$,

$$\underset{d \times d}{L_{uv}} = \begin{cases} \sum_{\{u,q\} \in \mathcal{E}} P_{uq}^{-1} \text{ (or } P_{qu}^{-1}) & \text{if } u = v \\ -\sum P_{uv}^{-1} \text{(or } P_{vu}^{-1}) & \text{if } u \neq v \wedge \{u, v\} \in \mathcal{E} \\ \mathbf{0} & \text{if } \{u, v\} \notin \mathcal{E} \end{cases} \tag{6}$$

$$\underset{dn_b \times d}{\mathbf{H}_i} = [\underset{d \times d}{\mathbf{0}_1}, \underset{d \times d}{\mathbf{0}_2}, \cdots, \underset{d \times d}{\mathbf{I}_i}, \cdots, \underset{d \times d}{\mathbf{0}_{n_b}}]^T$$

The subscript of $\mathbf{H}_i$'s element denotes the index of that element. Actually, it is not hard to show that $\mathbf{L}$ in Equation 5 is equal to $\mathcal{L}$ in Equation 3. In addition, if we consider all nodes' $\Sigma_{ii}, i = 1, 2, \cdots, n_b$ together, we can get the following equations:

$$\underset{dn_b \times dn_b}{\mathcal{L}} \underset{dn_b \times dn_b}{\mathbf{U}} = \underset{dn_b \times dn_b}{\mathbf{H}} \tag{7}$$

where $\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_{n_b}]$ and $\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_{n_b}] = \mathbf{I}$ (size $dn_b \times dn_b$). Obviously, the solution is $\mathbf{U} = \mathcal{L}^{-1} = \mathbf{\Sigma}$. Another observation is that *the covariance matrix Σ_{ij} of Nodes i and j is numerically equal to the voltage potential U_{ij} of Node i in Node j's Current Graph $\mathcal{G}^j$, or the voltage potential U_{ji} of Node j in Node i's Current Graph $\mathcal{G}^i$ (since Σ is a symmetric matrix).*

Moreover, we can also directly build $\mathbf{b} = [b_i]$ in Equation 3 as follows.

$$b_i = \sum_{\substack{(u,v) \in E \\ u = i \vee v = i}} P_{uv}^{-1}[(-1)^{s_{uv}} \zeta_{uv} + c_u + c_v] \tag{8}$$

where $s_{uv} = 0$ if $u = i$, and $s_{uv} = 1$ if $v = i$; $c_u = x_u$ if Node u is an anchor node, otherwise $c_u = \mathbf{0}$ (size $d \times 1$); similarly, $c_v = x_v$ if Node v is an anchor node, otherwise $c_v = \mathbf{0}$.

We can directly build $\mathcal{L}$ and $\mathbf{b}$, and thus $\mathbf{\Sigma}$ and $\hat{\mathbf{x}}_b$ can be calculated. If the network is fixed, everything is fine. But what if the network changes? Do we need to redo all these calculations? There are two types of dynamic changes for a WSN: new measurements being generated and nodes quitting or joining the network. For the former problem, when a new measurement ζ_{uv} is generated, only two Nodes u and v are involved. So we only need to modify the balance equations for these two nodes if they are not anchor nodes, i.e. modifying four elements in $\mathcal{L}$ and two elements in $\mathbf{b}$ as follows (if any of them is an anchor, nothing needs to be done for that node).

Operation 1. *When a new measurement ζ_{uv} is generated, do $L_{uv}- = P_{uv}^{-1}, L_{vu}- = P_{uv}^{-1}, L_{uu}+ = P_{uv}^{-1}, L_{vv}+ = P_{uv}^{-1}, b_u+ = P_{uv}^{-1}\zeta_{uv} + c_v, b_u- = P_{uv}^{-1}\zeta_{uv} - c_u$*

For the event of a node leaving or joining the network, it only impacts that node and its neighbors. Especially for node joining, the new node should have measurements with the existing nodes in order to keep the graph weakly connected, otherwise, our system would not include this node. The corresponding operations are:

Operation 2. *If a node joins the network (with some measurements to some already existing nodes), a new column and a new row with all $\mathbf{0}$'s are appended to $\mathcal{L}$, and one element $\mathbf{0}$ is appended to $\mathbf{b}$. Then Operation 1 is taken for this node's new measurements.*

Operation 3. *If Node u leaves the network, for each of Node u's neighbors q, if the measurement is ζ_{qu}, then do $L_{qq}- = P_{uq}^{-1}$, $b_q+ = \zeta_{qu} - c_u$; if the measurement is ζ_{uq}, then do $L_{qq}- = P_{qu}^{-1}$ and $b_q- = \zeta_{uq} + c_u$. After that, delete Row u and Column u in $\mathcal{L}$, and the uth element of $\mathbf{b}$.*

In summary, the initial inputs of INOVA are the measurement set Z, the corresponding error covariance matrix set P and the anchor nodes' locations $\mathbf{x}_r$. The outputs are nodes' location estimates $\hat{\mathbf{x}}_b$ and the covariance matrix $\mathbf{\Sigma}$. After the initial localization, if there are some dynamic changes at

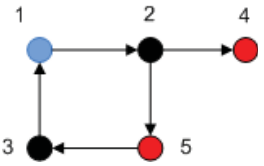

Figure 4: Node 1's Two-hop RMG

run time, the inputs are the set of new measurements, the set of nodes which leave, or the set of newly joining nodes with the corresponding measurements, and the outputs are the updated $\hat{\mathbf{x}}_b$ and $\mathbf{\Sigma}$. The whole algorithm is described as follows:

1. Calculate $P_i^{-1}, i = 1, 2, \cdots, m$. $(T(P_i^{-1}) = md^3)$.
2. Scan all measurements once, build up $\mathcal{L}$ and $\mathbf{b}$ by using Equations 6 and 8, respectively. $(T(\mathcal{L}) = 4md, T(\mathbf{b}) = 2md^2)$
3. Calculate $\mathbf{\Sigma} = \mathcal{L}^{-1}$. $(T(\mathbf{\Sigma}) = n_b^3 d^3)$.
4. Calculate $\hat{\mathbf{x}}_b = \mathcal{L}^{-1}\mathbf{b}$. $(T(\hat{\mathbf{x}}_b) = n_b^2 d^3)$.
5. If a new measurement is generated, do Operation 1 $(T(opt1) = O(d^3))$; if a new node joins the network with constant measurements, do Operation 2 $(T(opt2) = O(d^3))$; if a node having constant number of neighbors leaves the network, do Operation 3 $(T(opt3) = O(d^3))$. Then repeat Steps 3 and 4 $(T(\mathbf{\Sigma}) + T(\hat{\mathbf{x}}_b) = n_b^3 d^3)$.

The time complexity for INOVA to localize a fixed network is $T(INOVA) = O(n^3 d^3)$, and the time complexity for one dynamic change is also $O(n^3 d^3)$ (recall that n is the node number, m is the measurement number and d is the variable dimension). However, for the BLUE method, either for a fixed network or responding to a dynamic change, it all needs $O(mn^2 d^3)$. The improvement of INOVA is significant for two reasons: first, in a large dense network, m is much larger than n, because the possible edge number is n^2, and multiple measurements may be taken for the same node pair. Second, if we look into the details of these two processes, we find the key difference is how to build $\mathcal{L}$. In BLUE, it always needs $O(mn^2 d^3)$ time; while in INOVA, initially, it needs $O(n^3 d^3)$ time, but after that, when the network dynamically changes, the time is constant. Therefore, INOVA has much better performance when the topology changes (see Section 8.1).

5. A DISTRIBUTED ALGORITHM

In some scenarios (e.g. in wild areas), the centralized architecture may not be available, and thus a distributed scheme is required. Prabir Barooah et al. proposed a distributed algorithm OSE to estimate each node's location [2]. In OSE, each node assumes its two-hop neighbor's estimations are correct, iteratively estimate its own location based on its two-hop RMG, and exchanges the estimates among neighbors. After a number of iterations, each node's estimate converges to the estimate calculated by BLUE. The essence of OSE is to use a method, called Asynchronous Filtered Weighted Additive Schwarz (AFWAS) [2], to solve linear Equation 3 for the whole network in a distributed way.

However, OSE can only estimate node positions but not the covariance matrix. Therefore, we propose a similar distributed scheme called OSE-COV to estimate the covariance matrix $\mathbf{\Sigma}$ of the whole network. The differences of OSE-COV are: first, OSE-COV uses INOVA for its local calculation instead of BLUE; second, OSE-COV is to solve n_b linear systems of Equation 5 (while OSE only solves one), and each node initially does not know how many linear systems there are.

In OSE-COV, the tasks of each node u are to discover the existence of all the other nodes, and estimate its voltage potential Σ_{uk} in Node k's current graph $\mathcal{G}^k$, $\forall k = 1, 2, \cdots, n$. Each node u maintains its two-hop RMG $G_u(2)$ (it could be also extended to more than 2 hops, but the communication cost would be large). In each iteration, Node u $(u = 1, 2, \cdots, n)$ updates its known node list $S_u^{(i)}$ by merging its two-hop neighbors' $S_w^{(i)}$ $(w \in V_u(2))$, solves $\hat{\Sigma}_{uk}^{(i)}, \forall k \in S_u^{(i)}$ locally based on $G_u(2)$ by using node-voltage analysis and assuming their two-hop neighbors' $\hat{\Sigma}_{wk}^{(i-2)}$ are correct, and broadcasts $S_u^{(i)}$, $\hat{\Sigma}_u^{(i)}$ and its neighbor's estimates to its one-hop neighbors. Let $G_u(1) = (V_u(1), E_u(1), F_u(1))$ be Node u's one-hop RMG, where $V_u(1)$ consists of u and all its one-hop neighbors, $E_u(1)$ consists of all the edges (from the set E) between these nodes, and $F_u(1)$ is the corresponding mapping from edges to measurements; $G_u(2) = (V_u(2), E_u(2), F_u(2))$ is Node u's two-hop RMG; $A_u(1) = V_u(1) - \{u\}$ is the set of Node u's one-hop neighbors; $A_u(2) = V_u(2) - V_u(1)$ is the set of Node u's two-hop neighbors; $S_u^{(i)} = \{k | \text{Node } u \text{ knows the existence of Node } k \text{ at } i\text{th iteration}\}$ is Node u's known node list; $\hat{\Sigma}_u^{(i)}$ $(d \times d | S_u^{(i)}|)$ denotes Nodes u's estimates for Σ_{uk}, $\forall k \in S_u^{(i)}$; $X[i]$ denotes the ith element of set X; and we define a node as a voltage anchor node in $\mathcal{G}^k$ if its voltage potential is known in Node k's current graph. The whole algorithm is described as follows.

1. Initially, by broadcasting twice, each node $u \in V$ gets its two-hop RMG $G_u(2)$. Node u assumes its neighbors $A_u(1) \cup A_u(2)$ know the existence of $V_u(2)$, thus sets their known node lists as $S_v^{(0)} = V_u(2)$ and $S_w^{(-1)} = V_u(2)$, where $v \in A_u(1)$ and $w \in A_u(2)$, and picks up arbitrary estimates $\hat{\Sigma}_v^{(0)}, v \in A_u(1)$ and $\hat{\Sigma}_w^{(-1)}, w \in A_u(2)$ for them.
2. At the ith iteration, if Node u is an anchor, it updates its known nodes list $S_u^{(i)} = S_{A_u(1)}^{(i-1)} \cup S_{A_u(2)}^{(i-2)}$, where $S_{A_u(1)}^{(i-1)} = \bigcup_{v \in A_u(1)} S_v^{(i-1)}$ and $S_{A_u(2)}^{(i-2)} = \bigcup_{w \in A_u(2)} S_w^{(i-2)}$, and updates its estimate $\hat{\Sigma}_u^{(i)} = \mathbf{0}$ $(d \times d | S_u^{(i)}|)$. If Node u is not an anchor, it updates its known nodes list $S_u^{(i)} = S_{A_u(2)}^{(i-2)}$. Then Node u considers all its two-hop neighbors as voltage anchors (i.e. assumes their estimates $\hat{\Sigma}_w^{(i-2)}, \forall w \in A_u(2)$ are correct), and combines all their estimates $\hat{\Sigma}_w, w \in A_u(2)$ into one matrix $Cov_u^{(i)} = [(Cov_u^{(i)})_{jk}]$ as follows:

$$(Cov_u^{(i)})_{jk} = \begin{cases} \Sigma_{(A_u(2)[j])(S_u^{(i)}[k])} & \text{if } S_u^{(i)}[k] \in S_{A_u(2)[j]}^{(i-2)} \\ \text{arbitrary value} & \text{otherwise} \end{cases} \quad (9)$$

Then, Node u builds up $|S_u^{(i)}|$ groups of current balance equations as Equation 5 for $\mathcal{G}^k, \forall k \in S_u^{(i)}$, which can be written together as

$$\mathcal{L}_u(2)\mathbf{U}_u^{(i)} = \mathbf{C}_u^{(i)} - \mathcal{A}_{ub}(2)\mathcal{P}_u(2)^{-1}\mathcal{A}_{ur}(2)^T Cov_u^{(i)} \quad (10)$$

where $\mathcal{L}_u(2)$ ($d|V_u(1)| \times d|V_u(1)|$), $\mathcal{A}_{ub}(2)$ ($d|V_u(1)| \times d|E_u(2)|$), $\mathcal{A}_{ur}(2)$ ($d|A_u(2)| \times d|E_u(2)|$), and $\mathcal{P}_u(2)$ ($d|E_u(2)| \times d|E_u(2)|$) are the Kirchhoff Matrix, non-anchor node incidence matrix, anchor node incidence matrix and the measurement error diagonal matrix of $G_u(2)$, respectively. $\mathbf{U}_u^{(i)}$ ($d|V_u(1)| \times d|S_u^{(i)}|$) is the variable matrix, where the block element of ith row and jth column represents the estimate of Node $V_u(1)[i]$'s voltage potential in Node $S_u^{(i)}[j]$'s Current Graph, i.e. $\hat{\Sigma}_{(V_u(1)[i])(S_u^{(i)}[j])}$. $\mathbf{C}_u^{(i)}$ ($d|V_u(1)| \times d|S_u^{(i)}|$) is defined as:

$$(\mathbf{C}_u^{(i)})_{jk} = \begin{cases} \mathbf{I} & \text{if } V_u(1)[j] = S_u^{(i)}[k] \\ \mathbf{0} & \text{otherwise} \end{cases} \quad (11)$$

After solving $\mathbf{U}_u^{(i)}$, Node u only keeps the row corresponding to itself, denoted as $\mathcal{Y}_u$ ($d \times d|S_u^{(i)}|$), and then updates its estimates as $\hat{\Sigma}_u^{(i)} = \lambda\mathcal{Y}_u + (1-\lambda)\hat{\Sigma}_u^{(i-1)}$ (if the size of $\hat{\Sigma}_u^{(i-1)}$ is less than $\mathcal{Y}_u$, append $\hat{\Sigma}_u^{(i-1)}$ with the missing elements in $\mathcal{Y}_u$), where $0 < \lambda \leq 1$ is a pre-defined parameter (we use $\lambda = 0.9$ in this paper). Then, the new estimates $\hat{\Sigma}_u^{(i)}$ plus the known node list $S_u^{(i)}$, and the estimates $\hat{\Sigma}_v^{(i-1)}$ plus their known node lists $S_v^{(i-1)}, v \in A_u(1)$, previously received from its one-hop neighbors are broadcasted to all its one-hop neighbors.

3. When each node u receives broadcasts from its one-hop neighbors, it updates corresponding $S_v^{(i)}$, $S_w^{(i-1)}$, $\hat{\Sigma}_v^{(i)}$ and $\hat{\Sigma}_w^{(i-1)}$, where $v \in A_u(1)$ and $w \in A_u(2)$. Node u waits for a timeout or a predefined number of messages, and then it begins the $(i+1)$th iteration.

The above distributed algorithm can be stopped after a timeout or a predefined number of iterations. Take Figure 4 for example. Figure 4 shows the two-hop RMG $G_1(2)$ of Node 1. Nodes 4 and 5 are Node 1's two-hop neighbors, and considered as the voltage anchors in $G_1(2)$. Assume at the ith iteration, Node 1 receives the known node lists of Nodes 2, 3, 4 and 5 as $S_2^{(i-1)}$, $S_3^{(i-1)}$, $S_4^{(i-2)}$ and $S_5^{(i-2)}$, respectively, and their corresponding covariance matrix estimates as $\hat{\Sigma}_2^{(i-1)}$, $\hat{\Sigma}_3^{(i-1)}$, $\hat{\Sigma}_4^{(i-2)}$ and $\hat{\Sigma}_5^{(i-2)}$, respectively. Assume $S_4^{(i-2)} = (1,2,3,4,5)$, $S_5^{(i-2)} = (1,2,3,4,5,6)$, $\hat{\mathbf{\Sigma}}_4^{(i-2)} = [\hat{\Sigma}_{41}^{(i-2)}, \hat{\Sigma}_{42}^{(i-2)}, \hat{\Sigma}_{43}^{(i-2)}, \hat{\Sigma}_{44}^{(i-2)}, \hat{\Sigma}_{45}^{(i-2)}]$ and $\hat{\mathbf{\Sigma}}_5^{(i-2)} = [\hat{\Sigma}_{51}^{(i-2)}, \hat{\Sigma}_{52}^{(i-2)}, \hat{\Sigma}_{53}^{(i-2)}, \hat{\Sigma}_{54}^{(i-2)}, \hat{\Sigma}_{55}^{(i-2)}, \hat{\Sigma}_{56}^{(i-2)}]$. This means by the $i-2$th iteration, Node 4 knows the existences of Nodes 1, 2, 3, 4 and 5, and estimates its voltage potentials in Current Graphs $\mathcal{G}^1$, $\mathcal{G}^2$, $\mathcal{G}^3$, $\mathcal{G}^4$ and $\mathcal{G}^5$ as $\hat{\Sigma}_{41}^{(i-2)}$, $\hat{\Sigma}_{42}^{(i-2)}$, $\hat{\Sigma}_{43}^{(i-2)}$, $\hat{\Sigma}_{44}^{(i-2)}$ and $\hat{\Sigma}_{45}^{(i-2)}$, respectively. It is similar for Node 5, although Node 5 knows one more node (Node 6) than Node 4. Then Node 1 updates its known node list as $S_1^{(i)} = (1,2,3,4,5,6)$, builds $\mathbf{C}_1^{(i)}$ and $Cov_1^{(i)}$ as:

$$\mathbf{C}_1^{(i)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$Cov_1^{(i)} = \begin{bmatrix} \hat{\Sigma}_{41}^{(i-2)} & \hat{\Sigma}_{42}^{(i-2)} & \hat{\Sigma}_{43}^{(i-2)} & \hat{\Sigma}_{44}^{(i-2)} & \hat{\Sigma}_{45}^{(i-2)} & \mathbf{0} \\ \hat{\Sigma}_{51}^{(i-2)} & \hat{\Sigma}_{52}^{(i-2)} & \hat{\Sigma}_{53}^{(i-2)} & \hat{\Sigma}_{54}^{(i-2)} & \hat{\Sigma}_{55}^{(i-2)} & \hat{\Sigma}_{56}^{(i-2)} \end{bmatrix}$$

After that, Node 1 solves $\mathbf{U}_1^{(i)}$ for Nodes 1, 2 and 3 based on Equation 10, abandons all other nodes' solutions except for itself (the 1st row of $\mathbf{U}_1^{(i)}$, denoted as $\mathcal{Y}_1$), and updates $\hat{\Sigma}_1^{(i)} = \lambda\mathcal{Y}_1 + (1-\lambda)\hat{\Sigma}_1^{(i-1)}$. Then Node 1 broadcasts $S_1^{(i)}$, $S_2^{(i-1)}$, $S_3^{(i-1)}$, $\hat{\Sigma}_1^{(i)}$, $\hat{\Sigma}_2^{(i-1)}$ and $\hat{\Sigma}_3^{(i-1)}$ to all its one-hop neighbors.

Since OSE-COV does not require synchronization between nodes, it works under unreliable links. The convergence of OSE-COV can be proved by using the AWAS framework [9]. Another benefit of OSE-COV is that it has the same communication scheme as OSE. Hence at each iteration, all the exchange information of OSE and OSE-COV can be broadcasted together, and the location estimates and the covariance matrix estimates can be calculated simultaneously. To calculate the communication costs of OSE-COV, we assume 4 bytes are needed to represent a real number, 4 bytes for a node's address, and one node has v one-hop neighbors and w two-hop neighbors. Then for each iteration, Node u needs to broadcast $4(v+1)$ bytes for its own address and its neighbors' addresses, maximum $4(v+1)n$ bytes for the known node list of itself and its one-hop neighbors, maximum $4(v+1)nd^2$ bytes for itself and its one-hop neighbors covariance matrix estimates, $3v$ bytes for time stamps of these estimates. Therefore, the number of packets for one broadcast is

$$N_{tx}^{COV}(u) = \frac{(4nd^2+4n+7)v + 4nd^2 + 4n + 4}{max\_payload} \quad (12)$$

If we combine OSE and OSE-COV together, we only need to add $4d(v+1)$ bytes for the location estimates of Node u and its one-hop neighbors.

6. CONFIDENCE REGION INFERENCE

As described in Section 1, some confidence-based applications require not only the position estimates, but also the corresponding confidence regions. Since a node's position estimate from BLUE is a linear combination of all measurements, based on the large number theorem from probability theory, it is likely that it follows a multivariate normal distribution. If this conclusion is correct, then we can deduce its confidence region at some confidence level by using the properties of the multivariate normal distribution.

One version of central limit theorem, named Lindeberg's condition [1], says that for a sequence of independent random variables $X_k, k = 1,2,\cdots,n$, with $E(X_k) = \mu_k$ and $Var(X_k) = \sigma_k^2$, denoting $s_n^2 = \sum_{k=1}^{n} \sigma_k^2$, if $\max_{k=1,2,\cdots,n} \frac{\sigma_k^2}{s_n^2} \to 0$, as $n \to \infty$, i.e. none of σ_k^2 dominates s_n^2, then $Z_n = \sum_{k=1}^{n}(X_k - \mu_k)/s_n$ converges to a standard normal distribution $\mathcal{N}(0,1)$ when $n \to \infty$.

In BLUE, each node's position estimate is $\hat{x}_u = \sum_{k=1}^{m} W_k^u \zeta_k$, where W_k^u is the weight matrix of the measurement ζ_k Node u (or the generalized current of edge e_k in $\mathcal{G}^u$). Its corresponding covariance matrix is $\Sigma_{uu} = \sum_{k=1}^{m} W_k^u P_k (W_k^u)^T$. Let $y_k^u = W_k^u \zeta_k$, and then $\Sigma_{uu} = \sum_{k=1}^{m} COV(y_k^u)$. Define

$$Z^u = \Sigma_{uu}^{-\frac{1}{2}} \sum_{k=1}^{m} (y_k^u - E(y_k^u)) \quad (13)$$

If P_k is a diagonal matrix and $\forall k = 1,2,\cdots,m, COV(y_k^u)$ does not dominate Σ_{uu}, then Z^u tends to be a standard multivariate normal random vector, i.e. $Z^u \backsim \mathcal{N}_d(\mathbf{0}, \mathbf{I})$, since each element $z_i^u, i = 1,2,\cdots,d$ of Z^u tends to be a standard normal random variable under Lindeberg's condition.

In BLUE, the latter condition ($COV(y_k^u)$ does not dominate) is satisfied, because in practice usually the uncertainty of measurements generated by one device is similar. For the first condition, based on our simulation, even if P_k is not a diagonal matrix, Z^u still tends to be $\mathcal{N}_d(\mathbf{0}, \mathbf{I})$. Consequently, we can derive that

$$\begin{aligned} Z^u &= \Sigma_{uu}^{-\frac{1}{2}}(\sum_{k=1}^{m}(y_k^u) - \sum_{k=1}^{m}E(y_k^u)) = \Sigma_{uu}^{-\frac{1}{2}}(\hat{x}_u - x_u) \\ \Longrightarrow \quad & \hat{x}_u \backsim \mathcal{N}_d(x_u, \Sigma_{uu}) \end{aligned} \tag{14}$$

This means each node estimate from BLUE follows the multivariate normal distribution which is centered at the real position with the covariance matrix Σ_{uu}. Then we can further infer (see [1]) that

$$(\hat{x}_u - x_u)^T\Sigma_{uu}^{-1}(\hat{x}_u - x_u) \backsim \chi_d^2(\text{with } d \text{ d.f.}) \tag{15}$$

Therefore, the solid ellipsoid $\{x_u : (\hat{x}_u - x_u)^T\Sigma_{uu}^{-1}(\hat{x}_u - x_u) \leq \chi_d^2(\alpha)\}$ is the $100(1-\alpha)\%$ confidence region of x_u, where $\chi_d^2(\alpha)$ denotes the upper (100α)th percentile of the χ_d^2 distribution. For instance, after estimating node positions based on BLUE, Node u's estimate is $\hat{x}_u = [1, 2]^T$ and the covariance matrix is $\Sigma_{uu} = \begin{bmatrix} 1.6 & 0.25 \\ 0.25 & 1.2 \end{bmatrix}$. Then its 90% confidence region for x_u is the solid ellipsoid $\{x_u : \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix} - x_u\right)^T \begin{bmatrix} 1.6 & 0.25 \\ 0.25 & 1.2 \end{bmatrix}^{-1} \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix} - x_u\right) \leq 9.21\}$, since $\chi_2^2(0.01) = 9.21$.

7. OPTIMAL ANCHOR SELECTION

Anchor nodes are defined as the nodes which have accurate position information under a global coordinate frame. Any relative measurement based localization scheme must rely on one or more anchor nodes. Not only the anchor number, but also which nodes are selected to be anchors would impact the localization accuracy of the whole network. Taking Figure 2(b) for example, if Nodes 1 and 2 are already anchor nodes, obviously setting Node 4 as an anchor node is better than setting Node 9, since the node further away from the anchor nodes usually has large uncertainty, and thus more uncertainty is eliminated by setting Node 4 as an anchor. Additionally, selecting Node 3 is better than selecting Node 4, since it highly correlates to a large number of other nodes. Although the uncertainty of Node 3 is not as much as Node 4, the total uncertainty eliminated by setting Node 3 as an anchor is more (since the uncertainty of Nodes 5, 6, 7 and 8 is also reduced). Although this example provides some intuitive sense on how to select a good anchor, in large networks, anchor selection becomes much more complicated.

Without giving any quantitative uncertainty, the most straight forward strategy is to choose the node which has the max-min distance from all the anchor nodes, i.e. the node which has the maximum distance of the set of nodes' minimum distances to all the anchors is selected to be an anchor. This strategy assumes that the measurement uncertainty monotonically increases as the measured distance increases (e.g. the ToA method in [14]). For example, in Figure 2(b), assume the measured distances of node pairs are $Z = \{\zeta_{13} = 4.1, \zeta_{23} = 4, \zeta_{14} = 5.1, \zeta_{24} = 5, \zeta_{19} = 1, \zeta_{35} = 1, \zeta_{36} = 1, \zeta_{37} = 1, \zeta_{38} = 1\}$. Then the set of nodes' minimum distances to all anchors is $D_{min} = \{D_1 = 0, D_2 = 0, D_3 = 4, D_4 = 5, D_5 = D_6 = D_7 = D_8 = D_9 = 1\}$. Hence Node 4 should be selected.

When the quantitative uncertainty—the covariance matrix of all estimates—is given, the metric for selecting the best anchor node becomes selecting the node as an anchor so that the resultant trace of the covariance matrix is the smallest, that is

$$K = \underset{i=1,2,\cdots,n}{\arg\min}\, trace(\mathbf{\Sigma}_i^{New}) \tag{16}$$

Where $\mathbf{\Sigma}_i^{New}$ denotes the new covariance matrix after selecting Node i as an anchor node. The trace of the covariance matrix represents the sum of mean square errors of all the node estimates. Hence the above metric satisfies the minimum mean square error(MMSE) requirement. To reduce $trace(\mathbf{\Sigma})$ as much as possible, one naive method is to select the node which has the maximum variation, i.e. selecting Node $k = \underset{i=1,2,\cdots,n}{\arg\max}\, trace(\Sigma_{ii})$. By using this method, the $trace(\mathbf{\Sigma})$ is reduced by at least as much as $trace(\Sigma_{kk})$. Although this method is very simple, it does not take the correlation between nodes into account. Actually, some node which is highly correlated to many other nodes should have the priority to be the anchor node, as Node 3 in Figure 2(b). Following we give the optimal anchor selection strategy which satisfies Equation 16 and its proof. To select multiple anchors, we can just repeat this strategy.

THEOREM 1. *If only one node can be set as an anchor, $trace(\mathbf{\Sigma}^{New})$ is minimized only when Node $K = \underset{i=1,2,\cdots,n}{\arg\max}$ $trace\left(\sum_{i=1}^{n_b}(\Sigma_{ik}\Sigma_{kk}^{-1}\Sigma_{ki})\right)$ is selected.*

Proof. To set Node k as an anchor is equivalent to get a new measurement between Node k and the global origin, and the measurement error covariance matrix $P_k = \mathbf{0}$. Assume $\mathbf{\Sigma}(t)$ and $\mathbf{\Sigma}(t+1)$ are the covariance matrices before and after selecting Node k as an anchor node, respectively. Then, $\mathbf{\Sigma}(t+1) = \mathcal{L}^{-1}(t+1) = (\mathcal{A}_b(t+1)\mathcal{P}^{-1}(t+1)\mathcal{A}_b^T(t+1))^{-1}$, where $\mathcal{A}_b(t+1) = \begin{bmatrix} \mathcal{A}_b(t) & H_k \end{bmatrix}$, $H_k = [\mathbf{0}_1, \mathbf{0}_2, \cdots, \mathbf{I}_k, \cdots, \mathbf{0}_{n_b}]^T$ and $\mathcal{P}(t+1) = \begin{bmatrix} \mathcal{P}(t) & \mathbf{0} \\ \mathbf{0} & P_k \end{bmatrix}$. Note P_k should be equal to $\mathbf{0}$, however, we first assume $P_k \to \mathbf{0}$, but $P_k \neq \mathbf{0}$, and P_k^{-1} exist. Then

$$\mathcal{L}(t+1) = \begin{bmatrix} \mathcal{A}_b(t) & H_k \end{bmatrix} \begin{bmatrix} \mathcal{P}(t) & \mathbf{0} \\ \mathbf{0} & P_k \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{A}_b^T(t) \\ H_k^T \end{bmatrix}$$

$$= \mathcal{A}_b(t)\mathcal{P}^{-1}(t)\mathcal{A}_b^T(t) + H_kP_k^{-1}H_k^T = \mathbf{\Sigma}^{-1}(t) + H_kP_k^{-1}H_k^T$$

By using the formula

$$(\mathbf{A} + \mathbf{BD}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1}$$

we obtain

$$\begin{aligned} \mathbf{\Sigma}(t+1) &= \mathcal{L}^{-1}(t+1) \\ &= \mathbf{\Sigma}(t) - \mathbf{\Sigma}(t)H_k(P_k + H_k^T\mathbf{\Sigma}(t)H_k)^{-1}H_k^T\mathbf{\Sigma}(t) \\ &\overset{P_k=\mathbf{0}}{=} \mathbf{\Sigma}(t) - \mathbf{\Sigma}(t)H_k(H_k^T\mathbf{\Sigma}(t)H_k)^{-1}H_k^T\mathbf{\Sigma}(t) \\ &= \mathbf{\Sigma}(t) - \mathbf{\Delta}_k \end{aligned}$$

Therefore, $trace(\mathbf{\Sigma}(t+1)) = trace(\mathbf{\Sigma}(t)) - trace(\mathbf{\Delta}_k)$. To minimize $trace(\mathbf{\Sigma}(t+1))$ is equivalent to maximize $trace(\mathbf{\Delta}_k) = trace\left(\sum_{i=1}^{n_b}(\Sigma_{ik}\Sigma_{kk}^{-1}\Sigma_{ki})\right)$. **Q.E.D.**

To calculate $trace(\mathbf{\Sigma}(t+1))$ for all k's, the time complexity is $O(d^3n^2)$, and to select the maximum $trace(\mathbf{\Sigma}(t+1))$,

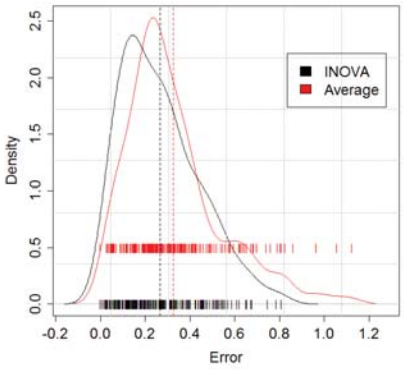

Figure 5: Error Density of INOVA and Average-Based Methods

it takes $O(n)$ time. Therefore, the total time complexity is $O(d^3n^2)$. One benefit of this method is that it is compatible with the distributed algorithm in Section 5. Since each node maintains one row of the covariance matrix, i.e. $\Sigma_{ki}, i = 1, 2, \cdots, n$, and $\Sigma_{ki} = \Sigma_{ik}$. Hence nodes can calculate $trace(\mathbf{\Delta}_k)$ locally, and then the whole network selects the node with maximum $trace(\mathbf{\Delta}_k)$ as the anchor node.

8. EVALUATION

In this section, we evaluate the localization accuracy and time complexity of INOVA, the convergence of the distributed algorithm–OSE-COV, the correctness of the confidence region inference, and the performance of the optimal anchor selection strategy. All the evaluations are based on Matlab simulation on a PC with dual Intel(R) Core(TM)2 CPU 6600 @2.40GHz, 2G memory and the Windows XP SP3 operating system.

As the default configuration in Figure 1, 200 nodes are randomly deployed in a square area with 30 by 30 grids (each grid is a 1×1 square), and 1000 relative measurements (the blue lines) are randomly generated among them. The relative measurement ζ_{ij} between Nodes i and j is determined by an angel measurement θ_{ij} and a distance measurement r_{ij}, where θ_{ij} and r_{ij} are random variables, and $\theta_{ij} = \theta^*_{ij} + \epsilon^\theta_{ij}$ and $r_{ij} = r^*_{ij} + \epsilon^r_{ij}$. θ^*_{ij} and r^*_{ij} are the true angle and distance between Nodes i and j. $\epsilon^\theta_{ij} \backsim \mathcal{N}(0, \sigma^2_\theta)$ and $\epsilon^r_{ij} \backsim \mathcal{N}(0, \sigma^2_r)$, where $\sigma_\theta = 10^\circ$ and $\sigma_r = 0.15 r^*_{ij}$. Therefore, the measurement between a node pair i and j is $\zeta_{ij} = [r_{ij}\cos\theta_{ij}, r_{ij}\sin\theta_{ij}]^T$, and the error covariance matrix is

$$P_{ij} = \begin{bmatrix} y^2\sigma^2_{\theta_{ij}} + \sigma^2_r\cos^2\theta^*_{ij} & -xy\sigma^2_{\theta_{ij}} + \sigma^2_r\sin(2\theta^*_{ij})/2 \\ -xy\sigma^2_{\theta_{ij}} + \sigma^2_r\sin(2\theta^*_{ij})/2 & x^2\sigma^2_{\theta_{ij}} + \sigma^2_r\sin^2\theta^*_{ij} \end{bmatrix}$$

where $x = r^*_{ij}\cos\theta^*_{ij}$ and $y = r^*_{ij}\sin\theta^*_{ij}$. The parameters of node number, measurement number and the area size can be changed during the simulation process.

8.1 Localization Accuracy and Efficiency

We first compare the localization accuracy of INOVA with the traditional average-based methods, and then evaluate the execution time of INOVA and BLUE for calculating both position estimates and the covariance matrix. As described in Section 1, INOVA and BLUE should have better localization accuracy than the traditional average-based localization schemes, such as triangulation or multilateration. This is because INOVA better fuses the measurements by quantitatively differentiating the measurement uncertainty. Figure 5 shows the error density of INOVA versus average-based methods (AVG). The error is defined as the Euclidean distance of the estimated location and the real location. From the plot, we can see that the peak of INOVA error is at the left side of AVG, and the average error of AVG is 0.325, while that of INOVA is 0.267, which is 18% better.

Figure 6 shows the execution time for INOVA and BLUE as new measurements are incrementally generated. The result shows that initially, to localize a network with 500 nodes and 2000 measurements, BLUE takes 13.5 seconds, while INOVA only needs 8.8 seconds. After that, 200 measurements are incrementally generated in each iteration. The execution time for BLUE increases very quickly, while INOVA almost keeps a constant time. When there are 4000 measurements in the network, the computational time of INOVA is only 0.5 second, which is 70 times as fast as BLUE. Similarly, in Figure 7, which fixes the measurement number at 2000, but incrementally adds 50 nodes each time, the execution time of BLUE shows a large rate of rise, while INOVA only increases slightly. When there are 1100 nodes in the network, the computational time of INOVA is 5 seconds, which is 12 times as fast as BLUE.

This experiment shows that in a large network where measurements are continuously generated or nodes dynamically join or leave the network, INOVA can provide real-time estimates, while BLUE fails. Based on this result, we can even apply INOVA to a mobile network. When a node moves, this node is considered to leave the network, and a new node joins the new position with new measurements. Since INOVA has very low execution time, it can respond to these mobile behaviors very quickly.

8.2 Convergence Speed of OSE-COV

In this section, we evaluate the convergence speed for the OSE-COV distributed algorithm. The network in Figure 1 is used, which has 200 nodes and 1000 measurements, and Node 1 at the position $[0, 0]^T$ is set to be the anchor node. Figure 8 shows the convergence degree during 100 iterations. The y-axis is the difference ratio, which is defined as $\sum_{i=1}^{dn_b}\sum_{j=1}^{dn_b}(\sigma_{ij} - \hat{\sigma}_{ij})^2 / \sum_{i=1}^{dn_b}\sum_{j=1}^{dn_b}\sigma^2_{ij}$, where σ_{ij} denote the element of the ith row and jth column of $\mathbf{\Sigma}$ obtained by BLUE or INOVA, while $\hat{\sigma}_{ij}$ denotes the corresponding element estimated by OSE-COV. The result shows that OSE-COV converges as the iteration number increases. Actually, to further improve the convergence speed, we can initialize each node's estimate with a better value, i.e. nodes start to execute OSE-COV only when its neighbors have the estimates.

8.3 Confidence Region Inference Accuracy

Based on the analysis in Section 6, each node's estimate $\hat{x}_u$ is a multivariate normal random variable, and $(\hat{x}_u - x_u)^T\Sigma^{-1}_{uu}(\hat{x}_u - x_u) \backsim \chi^2_d$(with d d.f.). To test the correctness, we first examine the normality of the estimates. Figure 9 shows the chi-square plot for all node's estimates in Figure 1. Chi-square plot shows the relationship between the statistic distance $(\hat{x}_u - x_u)^T\Sigma^{-1}_{uu}(\hat{x}_u - x_u)$ and the quantiles of chi-square distribution. The closer the result is to the line $y = x$, the more likely the sample x_i's are from a multivariate normal distribution. The plot in Figure 9 is very closed to $y = x$, and thus our inference is correct.

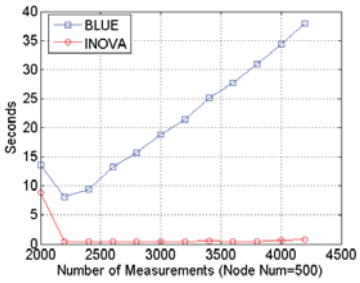

Figure 6: Execution Time for Incremental Measurements

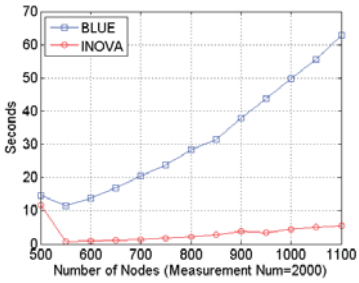

Figure 7: Execution Time for Incremental New Nodes

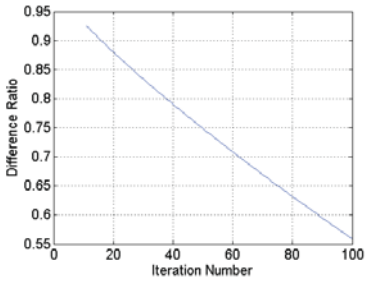

Figure 8: Convergence Speed for COV-OSE Distributed Algorithm

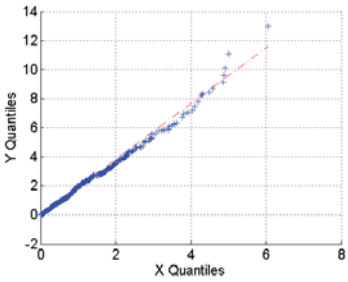

Figure 9: Chi-square Plot for All Node's Estimates

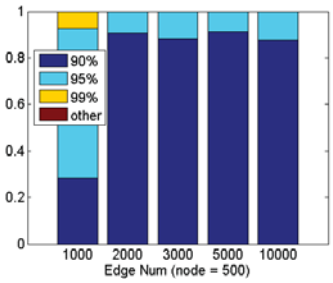

Figure 10: Confidence Region Accuracy

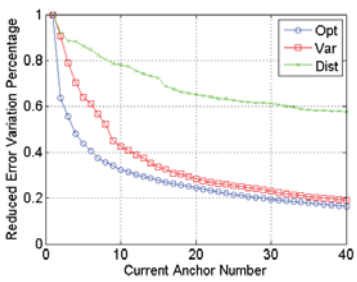

Figure 11: Anchor Selection Strategies Comparison

To show the correctness of our second inference (about the confidence region), we use three confidence levels–90%, 95% and 99%, and the corresponding chi-square quantiles are $\chi^2_2(0.1) = 4.61, \chi^2_2(0.05) = 5.99, \chi^2_2(0.01) = 9.21$. Based on the theorem in Section 6, 90% of node's real positions should fall in the ellipse $(\hat{x}_u - x_u)^T \Sigma_{uu}^{-1} (\hat{x}_u - x_u) \leq 4.61$, 95% of nodes' real positions should be within the ellipse $(\hat{x}_u - x_u)^T \Sigma_{uu}^{-1} (\hat{x}_u - x_u) \leq 5.99$, and 99% of them should be in $(\hat{x}_u - x_u)^T \Sigma_{uu}^{-1} (\hat{x}_u - x_u) \leq 9.21$. Figure 10 shows the statistical results for 5 different networks, which all have 500 nodes, but with 1000, 2000, 3000, 5000 and 10000 measurements, respectively. Except the first network (the ratio of measurement number to node number is low), all others have the consistent results with our inference. We also studied other networks of different sizes. It seems when the ratio of measurement number to node number is bigger than 2, the results are quite consistent and stable. Therefore, in practice, if the network is dense, we can use the covariance matrix to infer the confidence region accurately. Note that all the measurements are not from a multivariate normal distribution (as described at the beginning of Section 8). Therefore, there is no requirement on the population where the measurements are from.

8.4 Anchor Selection Strategies Comparison

In this section, we evaluate the performance of the three anchor selection strategies described in Section 7—the distance-based strategy (Dist), the variation-based strategy (Var) and the optimal strategy (Opt). The experiment shows how much percentage of total error variation is reduced as the anchor nodes are set one after another by using one specific strategy. In Figure 11, the x-axis is the current anchor number, and the y-axis is the percentage of the current total error variation to that at the beginning. The results show that Dist performs worst, while Opt is the best. We also observe that the first few selections reduce the total error variation a lot, where the reduced error percentage of Opt is four times as much as Var. But as more and more anchors are selected, the reduction rate decreases. This is because the nodes with largest uncertainty are selected at the beginning, leaving the nodes with little uncertainty. Another observation is that as more and more nodes are set to be anchors, method Var trends to have the similar performance as Opt. This is because when multiple anchors are selected, the anchors Var selected are very likely to be similar as Opt, although the selection order may differ.

9. RELATED WORK

In the wireless sensor networks field, localization approaches can be divided into two classes: range-based and range-free solutions. Range-based approaches use special devices, such as ultrasound [22, 24] or radio[6, 15], and various techniques, such as Time of Arrival (ToA) [15, 16], Time Difference of Arrival (TDoA) [6, 5, 22, 24], Angle of Arrival (AOA) [4, 18], or Phase Difference of Arrival (PDoA) [17] to estimate the distances or bearing among nodes, and then apply a triangulation, lateration or multilateration algorithm for final location estimations.

On the other hand, range-free localization solutions do not directly measure the distances or angles among nodes, instead, they make use of proximity information or events to infer node positions. When the density of sensor nodes is high, a number of solutions [19, 21] make the assumption of the shortest path (hop counted) between two nodes approximating the Euclidean distance of them, and thus can localize nodes with a few anchors. Other solutions make use of events to localize nodes, since these events are associated with the coordinate information [25, 26], distance information [23], or node sequence information [29]. Other works [11, 30] use relative RSSI difference to iteratively shrink node possible positions and finally localize the nodes.

However, no matter which technique is used, these me-

thods suffer from several drawbacks: first, they do not differentiate the uncertainty of measurements, or at least do not well quantify the uncertainty [29, 28, 10, 27]. Therefore, their deduction is not theoretically optimal. Second, they do not provide the confidence of the estimates, so their estimates cannot be used for confidence-based applications. Third, there is no way for them to guide the anchor selection.

Prabir Barooah et al. present a quantitative uncertainty-based localization scheme, the Relative Measurement Graph model, which takes the quantitative uncertainty of measurements into account, and provides the covariance matrix for the estimates [2, 3]. However, their work is not adaptive to the incremental measurements generation or dynamic node number changes. Hence, their approach cannot provide real time estimation for these dynamic behaviors. Additionally, they only focus on the location estimation, but not the covariance matrix. So they do not provide any distributed algorithm for covariance matrix calculation, and do not make use of covariance matrix for confidence region inference and anchor selection.

In robotics field, there is a quantitative uncertainty-based localization solution called SLAM [7, 8], which can optimally (or approximate optimally) localize both a robot and landmarks simultaneously, make use of incremental measurements, and provide confidence of the estimates. SLAM uses the state space model, and periodically makes predictions and updates on the robot's location estimate as well as the landmark location estimates as the robot moves on. Different from SLAM, our work does not assume a state space model and also does not require a mobile agent. Additionally, we provide a distributed way for estimating both node locations and the corresponding covariance matrix. Therefore, our work is more suitable for general WSN scheme.

10. CONCLUSION

This paper emphasizes the importance of the quantitative uncertainty in WSN localization. The quantitative uncertainty is modeled as the error covariance matrix of the estimates, and can be used to infer the confidence region for location estimates and assist anchor selection so as to reduce the total mean square estimation error. To calculate the covariance matrix, we present a centralized algorithm, called INOVA, which has low time complexity and is adaptive to the dynamic changes of a network. In addition, a distributed algorithm—OSE-COV is presented for peer-to-peer architectures.

11. REFERENCES

[1] R. B. Ash and C. A. Doléans-Dade. *Probability and Measure Theory (2nd ed.)*. San Diego: Harcourt/Academic Press, 2000.

[2] P. Barooah and J. Hespanha. Estimation on graphs from relative measurements. *Control Systems Magazine, IEEE*, 27(4):57 –74, 2007.

[3] P. Barooah and J. Hespanha. Estimation from relative measurements: Electrical analogy and large graphs. *Signal Processing, IEEE Transactions on*, 56(6):2181 –2193, 2008.

[4] H.-l. Chang, J.-b. Tian, T.-T. Lai, H.-H. Chu, and P. Huang. Spinning beacons for precise indoor localization. In *SenSys*, 2008.

[5] X. Cheng, H. Shu, Q. Liang, and D. H.-C. Du. Silent positioning in underwater acoustic sensor networks. *IEEE Transactions on Vehicular Technology*, 2008.

[6] X. Cheng, A. Thaeler, G. Xue, and D. Chen. Tps: A time-based positioning scheme for outdoor wireless sensor networks. In *INFOCOM*, 2004.

[7] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE Robotics And Automation Magazine*, 2006.

[8] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (slam): Part ii - state of the art. *IEEE Robotics And Automation Magazine*, 2006.

[9] A. Frommer, H. Schwandt, Daniel, and D. B. Szyld. Asynchronous weighted additive schwarz methods. *Electronic Transactions on Numerical Analysis*, 5:48–61, 1997.

[10] D. K. Goldenberg, P. Bihler, M. Cao, J. Fang, B. D. O. Anderson, A. S. Morse, and Y. R. Yang. Localization in sparse networks using sweeps. In *MobiCom*, 2006.

[11] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *MobiCom*, 2003.

[12] R. K. Jeremy, R. M. Karp, J. Elson, C. H. Papadimitriou, and S. Shenker. Global synchronization in sensornets. In *LATIN*, pages 609–624, 2004.

[13] S. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, 1993.

[14] Y. Kwon, K. Mechitov, S. Sundresh, W. Kim, and G. Agha. Resilient localization for sensor networks in outdoor environments. In *ICDCS*, 2005.

[15] S. Lanzisera, D. T. Lin, and K. S. J. Pister. Rf time of flight ranging for wireless sensor network localization. In *WISES*, 2006.

[16] H. Liu, J. Li, Z. Xie, S. Lin, K. Whitehouse, J. Stankovic, and D. Siu. Automatic and robust breadcrumb system deployment for indoor firefighter applications. In *MobiSys*, 2010.

[17] J. Liu, Y. Zhang, and F. Zhao. Robust distributed node localization with error management. In *MobiHoc*, 2006.

[18] M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, A. Lédeczi, G. Balogh, and K. Molnár. Radio interferometric geolocation. In *SenSys*, 2005.

[19] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. In *INFOCOM*, 2003.

[20] D. Niculescu and B. Nath. Dv based positioning in ad hoc networks. *Journal of Telecommunication System*, 2003.

[21] G. Oberholzer, P. Sommer, and R. Wattenhofer. Spiderbat: Augmenting wireless sensor networks with distance and angle information. In *IPSN*, 2011.

[22] L. I. On, R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *IPSN*, 2003.

[23] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *MobiCom*, 2000.

[24] K. Römer. The lighthouse location system for smart dust. In *MobiSys*, 2003.

[25] A. Savvides, C.-C. Han, and M. B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MobiCom*, 2001.

[26] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke. A high-accuracy, low-cost localization system for wireless sensor networks. In *SenSys*, 2005.

[27] R. Stoleru, P. Vicaire, T. He, and J. A. Stankovic. Stardust: A flexible architecture for passive localization in wireless sensor networks. In *SenSys*, 2006.

[28] W. Xi, Y. He, Y. Liu, J. Zhao, L. Mo, Z. Yang, J. Wang, and X. Li. Locating sensors in the wild: pursuit of ranging quality. In *SenSys '10*, pages 295–308, New York, NY, USA, 2010. ACM.

[29] Z. Yang and Y. Liu. Quality of trilateration: Confidence based iterative localization. In *ICDCS*, 2008.

[30] Z. Zhong and T. He. Msp: multi-sequence positioning of wireless sensor nodes. In *SenSys*, 2007.

[31] Z. Zhong and T. He. Achieving range-free localization beyond connectivity. In *SenSys*, 2009.