Locating nodes in mobile sensor networks more accurately and faster

Zhang Shigeng†‡, Jiannong Cao†
Internet and Mobile Computing Lab†
Hong Kong Polytechnic University
Hong Kong
Email: {cssgzhang,csjcao}@comp.polyu.edu.hk

Chen Lijun‡, Chen DaoXu‡
State Key Laboratory for Novel Software Technology†
Nanjing University
Nanjing, P.R. China
Email: {chenlj,cdx}@nju.edu.cn

***Abstract*—Localization in mobile sensor networks is more challenging than in static sensor networks because mobility increases the uncertainty of nodes' positions. Most existing localization algorithms in mobile sensor networks use Sequential Monte Carlo (SMC) methods due to their simplicity in implementation. However, SMC methods are very time-consuming because they need to keep sampling and filtering until enough samples are obtained for representing the posterior distribution of a moving node's position. In this paper, we propose a localization algorithm that can reduce the computation cost of obtaining the samples and improve the location accuracy. A simple bounding-box method is used to reduce the scope of searching the candidate samples. Inaccurate position estimations of the common neighbor nodes is used to reduce the scope of finding the valid samples and thus improve the accuracy of the obtaineed location information. Our simulation results show that, comparing with existing algorithms, our algorithm can reduce the total computation cost and increase the location accuracy. In addition, our algorithm shows several other benefits: 1) it enables each determined node to know its maximum location error, 2) it achieves higher location accuracy under higher density of common nodes, and 3) even when there are only a few anchor nodes, most nodes can still get position estimations.**

***Index Terms*—localization; bounding-box; wireless sensor networks; mobility; Sequential Monte Carlo methods**

I. Introduction

Wireless sensor networks are becoming a reality and have been used in many applications such as habitat monitoring[14], long-term animal migration tracking [10] and animal state estimating and actuating [22]. In many applications, it is desired to know the position information about the sensor nodes. For example, the user needs to know where the data are collected in order to perform data analyses or to determine what actions should be taken. Also the node position information are used in geographical routing protocols such as GPSR [11] and clustering protocols such as EECS[24].

Traditional methods of obtaining the node position information include attaching a GPS receiver in each node or manually configure each node's position. As the scale of sensor networks becomes larger and larger, these methods are becoming unfeasible for their high cost and inconvenience. Many localization algorithms for sensor network have been proposed [8], [7], [12], [20], [16], [13], [19], [23], [3], [21], [18], [5]. These algorithms use some special nodes, called anchor nodes, which know their positions to facilitate the determination of the positions of the other nodes (called common nodes). However, these algorithms are designed for static sensor networks and are not applicable to mobile sensor networks because of the following limitations:

1) in some algorithms [20], [16], [13], [23], common nodes need to calculate the shortest paths to the anchor nodes by using flooding, which results in high communication cost.
2) in some algorithms [18], [7], [8], special hardware are required for the anchor nodes and for measuring the accurate distances between neighboring nodes.
3) some algorithms are centralized [20], [7], [5], [3], and will cause high communication cost for disseminating the results to all the common nodes.
4) some algorithms use iterative methods to improve location accuracy [1], [21], which will result in both high communication cost and high computation cost.

Localization algorithms specially designed for mobile sensor networks have also been proposed [9], [4], [2], [25], [17], but they all use the Sequential Monte Carlo (SMC) methods. SMC methods are preferred in mobile sensor networks because they are easy to implement and can exploit nodes' mobility to improve location accuracy. The idea of SMC methods is to represent the posterior distribution of a common node's position using a set of weighted samples. In each time unit, a common node predicts its new samples based on the samples obtained in the last time unit and filters out the invalid samples using the locations of the newly observed anchor nodes. After obtaining enough samples, the node uses the weighted average of all the samples as its position estimation. The procedure repeats and the node can continuously update its position estimation.

The main drawback of SMC methods is that they need to keep sampling and filtering until obtaining enough valid samples. This is very time consuming, and not suitable for sensor networks where nodes have limited computation ability. In this paper, we propose a localization algorithm which outperforms the existing algorithms in terms of computation cost and location accuracy. The proposed algorithm uses a simple bounding-box method to reduce the scope of searching for the candidate samples and uses inaccurate position estimations of the common neighbor nodes to improve the location accuracy. Our simulation results show that, comparing with the existing SMC-based localization algorithms, our algorithm can significanly reduce the total computation cost and increase the location accuracy.

Our algorithm have some other benefits. In addition to its location information, each determined node also knows the maximum location error. This may be required in some protocols such as geographical routing. Our algorithm performs better in location accuracy in networks with a large number of common neighbor nodes. In contrast, existing algorithms rely on the ancor nodes to achieve high accuracy. Finally, in our algorithm, even when there are only a few anchor nodes, most common nodes can still be determined.

The rest of this paper is organized as follows. Section II introduces necessary background knowledge, including the network model and SMC-based localization algorithms which we will compare with. Section III presents our algorithm and describe how it can reduce the computation cost and improve the location accuracy. Section IV analyzes the performances of our algorithm in terms of location accuracy and communication cost. Section V reports our simulation results and compares the performanec of our algorithm with exisitng algorithms introduced in section II. Finally, section VI concludes this paper.

II. Background

A. Network model

In this paper we consider mobile sensor networks where both the anchor nodes and common nodes are mobile. The time is assumed to be divided into discrete time units. Anchor nodes know their exact positions at any time while common nodes need to determine their positions in each time unit.

Both the anchor nodes and common nodes only have limited knowledge about their mobility. They only know the maximum speed v_{max}, which means that in each time unit a node can move in any direction with speed v where $0 < v \leq v_{max}$, but the exact value of v is unknown.

Two nodes can communicate with each other only if they are within the communication range defined by the radius r. If a node p can communicate with another node q, we say that q is a 1-hop neighbor of p. If q is an anchor node, we say that q is p's 1-hop anchor neighbor ; if q is a common node, we say q is p's 1-hop common neighbor. If there is another node m which cannot communicate with p but can communicate with q, then we say that m is a 2-hop neighbor of p. For simplicity in presentation and analysis, we assume that all messages are received instantly. As described above, we assume the unit disk model for connectivity. We will discuss the irregularity of communication models in section V.

The positions of the nodes in the network are assumed to be uniformly distributed in a 2-D deployment region. Assuming the area of the deployment region is s and there are totally n nodes including m anchor nodes, the density of all the nodes ρ_n and the density of anchor nodes ρ_s are defined as

$$\rho_n = \frac{n}{s}\ ,\ \rho_s = \frac{m}{s} \tag{1}$$

respectively.

The node degree n_d and anchor degree s_d represent the average number of the nodes and anchor nodes in a node's communication range and are defined as following[1]:

$$n_d = \rho_n * \pi r^2\ ,\ s_d = \rho_s * \pi r^2 \tag{2}$$

The probability that there are k nodes in a given area a follows a *Poisson distribution*:

$$Pr(k \text{ nodes in } a) = \frac{(\rho_n a)^k}{k!} e^{-\rho_n a} \tag{3}$$

Because we assume that all nodes' coordinates are uniformly distributed, using of Poisson distribution is reasonable as explained in [15]. We will use equation (3) to derive a lower bound on the expected location error in section IV.

B. Sequential Monte Carlo Localization(SMCL)

Sequential Monte Carlo methods are referred to a set of simulation-based methods which provide a convenient approach to compute the posterior distributions [6]. The posterior distribution is represented using a set of weighted samples, and the samples are updated gradually as time goes. As for localization in mobile sensor networks, the possible positions of a common node is represented using a set of weighted samples. In each time unit, the samples are updated based on the samples obtained in the last time unit, and are validated using the newly observed anchor nodes in the current time unit.

The Sequential Monte Carlo Localization (SMCL) algorithm, proposed in [9], is the first algorithm using SMC methods for localization in mobile sensor networks. For each common node, the operations in SMCL can be divided into 3 steps:

1) Initialization: N samples are randomly chosen from the deployment region to represent possible positions of p, denoted as $L_0 = \{l_0^1, l_0^2, \ldots, l_0^N\}$.
2) Prediction: N new samples L_t are drown from L_{t-1} using transition equation $p(l_t^i|l_{t-1}^i)$. The transition equation $p(l_t^i|l_{t-1}^i)$ is determined by the motion model or other constraints.
3) Filtering: weights of new samples are computed as $p(l_t^i|o_t)$, where o_t is the newly observed anchor nodes in the current time unit. Samples whose wights equal to 0 are dropped. If the number of samples after filtering step is less than N, go to step 2.

After obtaining N samples, p estimates its position as the weighted average of all the samples.

[1]We use the term "degree" for consistency with [9], [2], [17].

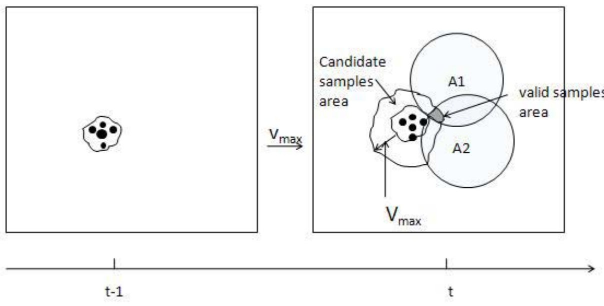

Fig. 1. How SMCL[9] works: 1) the left figure shows sample set L_{t-1} at time t-1; 2) Prediction based on L_{t-1} and filtering using newly observed anchor nodes.

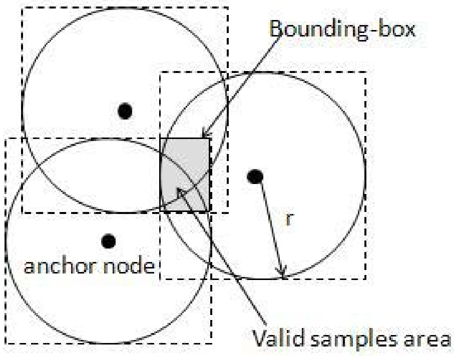

Fig. 2. MCB[2]: new sample candidates are only drawn from the shadowed area.

1) Transition equation: In SMCL[9], the transition equation is given by:

$$p(l_t|l_{t-1}) = \begin{cases} \frac{1}{\pi v_{max}^2} & \text{if } d(l_t, l_{t-1}) < v_{max} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $d(l_t, l_{t-1})$ is the distance between two samples l_t and l_{t-1}. This means that for each sample l_{t-1} in the last time unit, we randomly choose a point in the circle centering at l_{t-1} with radii v_{max} as a candidate sample in the current time unit.

2) Filtering: Candidate samples newly drawn in the prediction step may be inconsistent with the new observations of the anchor nodes in the current time unit denoted as o_t. Denote the possibility of observing o_t given l_t as $p(o_t|l_t)$. If $p(o_t|l_t)$ is 0, then l_t should be dropped. In SMCL, the filtering condition uses 1-hop and 2-hop anchor neighbors. Denote the set of all 1-hop anchor neighbors as S and the set of all 2-hop anchor neighbors as T, then the filtering condition of l_t is:

$$filter(l_t) = \forall s \in S, d(l_t, s) \le r \wedge \forall s \in T, r < d(l_t, s) \le 2r.$$

SMCL is a straightforward application of SMC methods in localization for mobile sensor networks. The probability for a candidate sample to be dropped after filtering step is high, especially when v_{max} or s_d is large. Consequently, it is very time-consuming, and especially un-suitable for wireless sensor networks where nodes only have limited computation ability.

C. Monte Carlo Localization Boxed(MCB)

Fig.1 illustrates how SMCL works. There are two areas used in SMCL: candidate samples area and valid samples area. The candidate samples area is used to draw new candidate samples and the valid samples area is used to filter out invalid candidate samples. When candidate samples area is large and valid samples area is small, candidate samples drawn in the prediction step have high probability to be dropped. From transition equation (4) and filtering condition (5) we know that the candidate samples area will be large when v_{max} is large and the valid samples area will be small when s_d is large. So SMCL will be very time-consuming when v_{max} or s_d is large.

To overcome this problem, [2] used a bounding-box method to reduce candidate samples area. The idea is to constrain candidate samples into a much smaller area. The perfect solution is only drawing candidate samples from the valid samples area. However, the valid samples area is hard to obtain. But we can construct an approximation of that area using a bounding-box. Suppose a common node can communicate with n 1-hop anchor neighbors, then a bounding-box $(x_{min}, x_{max}, y_{min}, y_{max})$ is built as following (see Fig.2):

$$\begin{aligned} x_{min} = max_{i=1}^n\{x_i - r\},\ x_{max} = min_{i=1}^n\{x_i + r\} \\ y_{min} = max_{i=1}^n\{y_i - r\},\ y_{max} = min_{i=1}^n\{y_i + r\} \end{aligned} \quad (5)$$

where (x_i, y_i) is the coordinate of the i'th 1-hop anchor neighbor. 2-hop anchor neighbors can be used to reduce the bounding-box further. When using 2-hop anchor nodes, we should replace r with $2r$ in the above formulas.

Using this method, the probability for a candidate sample to be reserved in the final sample set increases very much, so the computation cost is reduced. For scenarios in which it is difficult for SMCL to obtain enough samples (for example, when v_{max} or s_d is large) , MCB can also obtain enough samples, so achieve higher location accuracy than SMCL.

D. MSL and MSL*

In [17] the authors proposed two algorithms, MSL* and MSL, using position estimations of the common neighbor nodes to improve location accuracy. The main idea is to assign different weights to samples according to the relationship between this sample and the samples of the common neighbor nodes.

Similar to SMCL, MSL* and MSL also have three steps in processing: initialization, sampling, and re-sampling.

Initialization: Same as in SMCL and MCB.

Sampling: In order to be useful in both mobile sensor networks and static sensor networks, MSL* and MSL use a transition equation different from (4). The transition is given below:

$$p(l_t|l_{t-1}) = \begin{cases} \frac{1}{\pi (v_{max}+\alpha)^2} & \text{if } d(l_t, l_{t-1}) < v_{max} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where α is an empirically determined value. α is used to make the algorithm also suitable for static sensor networks. In [17] α is set to be $0.1r$ and we follow this setting in our simulations.

After a sample candidate is chosen, its weight is computed using 1-hop and 2-hop (anchor and common) neighbor nodes. For a sample s of a node p, its weight $w_s(p)$ is computed as the product of the partial-weights $w'_s(q)$ corresponding to each neighbor node q. That is,

$$w_s(p) = \prod_{q=1}^{k} w'_s(q) \tag{7}$$

where k is the number of 1-hop and 2-hop neighbor nodes of p. $w'_s(q)$ is computed as follows:

1) if q is p's 1-hop anchor neighbor node, then

$$w'_s(q) = \begin{cases} 1 & \text{if } d(s,q) \le r \\ 0 & \text{otherwise} \end{cases}$$

2) if q is p's 2-hop anchor neighbor node, then

$$w'_s(q) = \begin{cases} 1 & \text{if } r \le d(s,q) \le 2r \\ 0 & \text{otherwise} \end{cases}$$

3) if q is p's 1-hop common neighbor node, then

$$w'_s(q) = \sum_{q_i} w(q_i), \text{where } d(s,q_i) \le r + v_{max}$$

where q_i is q's sample in the last time unit and $w(q_i)$ is q_i's weight.

4) if q is p's 2-hop common neighbor node, then

$$w'_s(q) = \sum_{q_i} w(q_i), \text{where } r - v_{max} \le d(s,q_i) \le 2r + v_{max}$$

where q_i is q's sample in the last time unit and $w(q_i)$ is q_i's weight.

After N samples are obtained, their weights are normalized. The weight of the i-th sample is normalized as :

$$\frac{w_i(p)}{\sum_{j=1}^{N} w_j(p)}$$

Re-sampling: The resampling step in MSL* and MSL is different from that in SMCL and MCB. In MSL* and MSL, each sample in the last time unit may be included in the new sample set with a probability proportional to its weight. Samples with higher weights have greater chance to be reserved in the new sample set. New samples are drawn using transition equation (6) only when there are not enough samples inherited from sample set in the last time unit.

In MSL* a node uses sample sets of its common neighbor nodes to weight its samples which results in high communication cost. To reduce the communication cost, the authors propose a simplified version of MSL*, which is called MSL. Instead of using full set of samples, MSL only uses position estimations of common neighbor nodes to weight its samples. For each common node there is a value called *closeness* which is used to evaluate the quality of its position estimation. The closeness value of a node p with N samples is defined as

$$closeness_p = \frac{\sum_{i=1}^{N} w_i \sqrt{(x_i - x)^2 + (y_i - y)^2}}{N}$$

where (x, y) is p's position estimation and (x_i, y_i) is p's i-th sample. In MSL the partial-weight corresponding to a common neighbor node q is defined as

$$w'_s(q) = b^{-closeness_q}$$

where b is set to 7 in [17] and we follow this setting in our simulation. To further improve location accuracy, only those common neighbor nodes with lower closeness values than p are used in the weight computing step.

MSL* and MSL can achieve higher location accuracy than SMCL. However, MSL* and MSL need high node density and high anchor density to converge. Simultaneously, the property of reserving most samples in the last time unit makes MSL* and MSL more suitable for networks with low speed. In fact, the location accuracy using MSL* and MSL is lower than original SMCL when v_{max} is large, as reported in [17].

E. Other variants

There are some other variants of SMCL, for example, [4], [25]. In [4] distances between common nodes and anchor nodes were used in the filtering step in order to improve location accuracy. However, this method requires additional hardware to measure distances. We won't compare our algorithm with this algorithms because our algorithm has no such requirements. In [25] the authors proposed a multi-hop version of SMCL. In each time unit a common node obtains its shortest paths to all anchor nodes by flooding and using these distances to construct a candidate samples area and to do filtering. The good news of [25] is that it can alleviate high radio irregularity and can use much less anchors to localize all common nodes in the network. The bad news is that it will result in very high communication cost because it uses flooding.

In this paper we will compare our algorithm with SMCL, MCB, MSL* and MSL. We won't compare our algorithm with algorithms proposed in [4], [25] because they either require special hardware or use flooding.

III. Our algorithm

In this section we present our algorithm and explain how it reduces computation cost and increases location accuracy. Our algorithm also uses position estimations of common neighbor nodes to improve location accuracy. However, different from MSL* and MSL, it only uses location information of 1-hop common neighbor nodes but using this information in both prediction steps and filtering steps.

A. Reduce Bounding-box further

Let us recall how MCB reduces candidate samples area of SMCL. In MCB a bounding-box is built using 1-hop and 2-hop anchor neighbor nodes. Our first contribution is using 2-hop anchor neighbor nodes' negative effects to further reduce the bounding-box. See Fig.3. q is p's 2-hop anchor neighbor node, then the overlapping region of the bounding-box of p and the circle centering at q with radii r does't contain p, otherwise q will be p's 1-hop neighbor node. So that region should be eliminated.

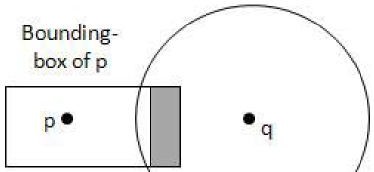

Fig. 3. Using 2-hop anchor neighbor nodes' negative effects to further reduce candidate samples area. The shadowed area should be eliminated.

Now suppose that each common node knows its maximum error in x-axis $errX$ and maximum error in y-axis $errY$ (how to compute them is explained in next subsection). Use $R_{t,p}$ and $E_{t,p}$ to denote node p's real position and estimated position in time unit t respectively. Suppose q is one of p's 1-hop common neighbor nodes in time unit t, then we have the following relationship:

$$|R_{t,p} - R_{t,q}| \leq r$$
$$|R_{t,q} - R_{t-1,q}| \leq v_{max}$$
$$|R_{t-1,q} - E_{t-1,q}|_x \leq errX_q$$
$$|R_{t-1,q} - E_{t-1,q}|_y \leq errY_q$$

Where $errX_q$ and $errY_q$ are q's maximum error in x-axis and y-axis in the last time unit respectively. Suppose that the bounding-box built using 1-hop and 2-hop anchor neighbor nodes is $(x_{min}, x_{max}, y_{min}, y_{max})$, then it can be updated as following (only take x_{min} as an example):

$$x_{min} = max_{q \in Q}\{x_{min}, E^x_{t-1,q} - v_{max} - r - errX_q\}$$

where Q is the set of p's 1-hop common neighbor nodes and $E^x_{t-1,q}$ is the x value of $E_{t-1,q}$.

p's position estimation in the last time unit can be used to further reduce the bounding-box. Take x_{min} as an example:

$$x_{min} = max\{x_{min}, E^x_{t-1,p} - v_{max} - errX_p\}$$

After the bounding-box is built, in prediction step candidate samples are randomly chosen from the bounding-box.

Position estimations of common neighbor nodes are also used in the filtering step in order to improve location accuracy. Now the filtering condition is defined as:

$$filter'(l_t) = filter(l_t) \wedge d(l_t, E_{t-1,q}) \leq v_{max} + r + err_q, \forall q \in Q$$

where Q is the set of p's 1-hop common neighbor nodes and $err_q = \sqrt{errX_q^2 + errY_q^2}$. It can be seen that when v_{max} becomes larger, the constraints introduced by common neighbor nodes become weaker.

B. Compute and broadcast $errX, errY$

After we have obtained N valid samples, we can compute the position estimation of a node p as the weighted average of all its samples. Using the position estimation and the

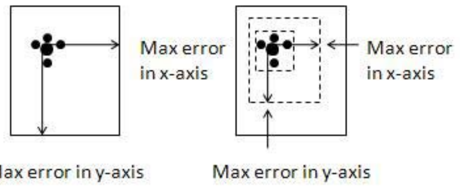

Fig. 4. Computing $errX$ and $errY$; the biggest dot is (x_e, y_e). The right figure shows a way to refine $errX$ and $errY$ with little risk.

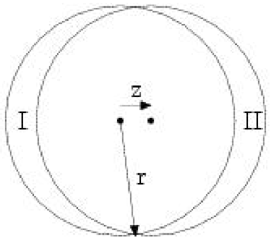

Fig. 5. Lower bound of expected location error[13]. The summation of area of I and area of II is about $4rz$.

bounding-box [2], we can compute $errX_p$ and $errY_p$. See Fig.4.

Suppose that a node p's position estimation is (x_e, y_e) and its bounding-box is $(x_{min}, x_{max}, y_{min}, y_{max})$, then it is obvious that $errX_p$ is $max(x_e - x_{min}, x_{max} - x_e)$ and $errY_p$ is $max(y_e - y_{min}, y_{max} - y_e)$. However, $errX_p$ and $errY_p$ can be further refined. Suppose that p's sample set is $\{(x_1, y_1), \ldots, (x_N, y_N)\}$. Let $x'_{min} = min\{x_1, \ldots, x_N\}$ and we can further refine x_{min} of the bounding-box to be $(x_{min} + x'_{min})/2$. Do the same to x_{max}, y_{min} and y_{max} we can get more refined $errX_p$ and $errY_p$. This may introduce some risks that the bounding-box doesn't contain p's real position. However, the possibility that this type of inconsistences occur is very small. In fact, in our simulation with unit dist connectivity model, the inconsistence only occurs for several times in the whole simulation.

After p gets (x_e, y_e) and $errX_p, errY_p$, it broadcasts them to its neighbor nodes and its common neighbor nodes will use that information in the next time unit.

IV. Algorithm analysis

A. location accuracy

Our algorithm uses only local connectivity information in each time unit's prediction step and filtering step. [13] proposed a method to estimate the lower bound on expected location error for this type of algorithms. Although [17] showed that this lower bound is not rigorous, it can still help us gain some insights of how accurate an algorithm of this type can achieve. We use it to derive a lower bound on expected location error for our algorithm.

The main idea is to treat the distance a common node can move without changing its connectivity as a random variable Z

[2]The bounding-box in the initialization step is set to be the deployment region.

and compute the expectation of Z. See Fig.5. The probability that a node can move a distance z without changing its connectivity information equals to the probability that there is at least one node in I or II. Using our network model (section II-A):

$$F(z) = 1 - e^{-\rho_n 4rz}$$

Because SMCL and MCB use only anchor neighbor nodes within two hops, the lower bound on expected location error for SMCL and MCB is:

$$\begin{aligned} E(z) &= \int_0^\infty z\, d(1 - e^{-\rho_s 12rz}) \\ &= \frac{\pi r}{12 s_d} \end{aligned}$$

Compared with SMCL, our algorithm also uses 1-hop common neighbor nodes. So for our algorithm, the expected location error is lower bounded by:

$$\begin{aligned} E(z) &= \int_0^\infty z\, d(1 - e^{-(\rho_s 8rz + \rho_n 4rz)}) \\ &= \frac{\pi r}{8 s_d + 4 n_d} \end{aligned}$$

However, the constraints introduced by common neighbor nodes ($v_{max} + r + err$) are much weaker than the constraints introduced by anchor neighbor nodes(r), so the real location error will be much higher than this lower bound.

MSL* and MSL use both common neighbor nodes and anchor neighbor nodes within 2 hops, and the lower bound is given by $\frac{\pi r}{12 n_d}$.

B. Communication cost

To find 1-hop anchor neighbor nodes, only anchor nodes need to broadcast their position information and the corresponding communication cost is $O(m)$ in each time unit. To find 2-hop anchor neighbor nodes, all nodes (including anchor nodes) should re-broadcast neighbor anchor nodes' position information they received. The corresponding cost is $O(n * sd)$. Similarly, to find 1-hop and 2-hop (anchor and common) neighbor nodes need $O(n)$ and $O(n * n_d)$ messages respectively.

SMCL and MCB only use 1-hop and 2-hop anchor neighbor nodes, so the total communication cost is $O(m + n * s_d)$. Our algorithm also uses 1-hop common neighbor nodes so the total communication cost is $O(n + n * s_d)$. The additional communication cost compared with SMCL and MCB is $O(n - m)$. For MSL*, it uses sample sets of 1-hop and 2-hop common neighbor nodes, so the communication cost is $O(m + n * s_d + (n - m) * N + n(n_d - s_d) * N)$ where N is the number of samples. The 3rd term is used to get sample sets of 1-hop common neighbor nodes and the 4th item is used to get sample sets of 2-hop common neighbor nodes. The communication cost of MSL is $O(n + n * n_d)$.

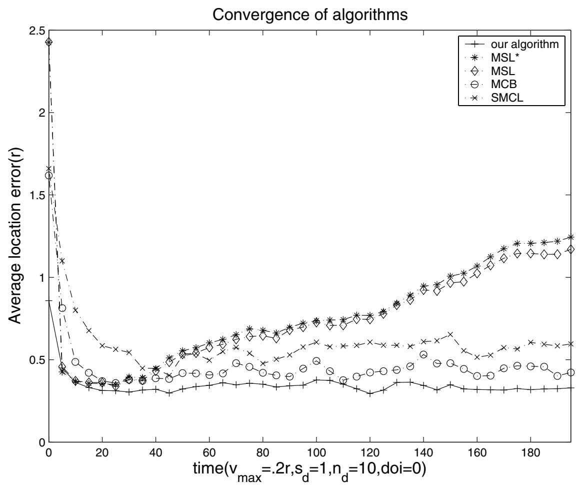

Fig. 6. Our algorithm converges faster than SMCL.

V. Simulation results

We obtained a copy of the simulator code from Hu and Evans[9] and modified it to implement five algorithms: SMCL, MCB, MSL*, MSL and our algorithm. The parameters are set to be consistent with [9], [2], [17]. All nodes are uniformly distributed in a 500m*500m region and r is set to 50m. Unless otherwise specified, the default value of parameters are: $n_d = 10$, $s_d = 1$, $v_{max} = .2r$. The number of samples is set to 50. We use the *degree of irregularity* (DOI) [26] model to test the effects of radio irregularity on location accuracy, and the default value of doi is 0. For each experiment, we randomly generate 10 networks and in each network we run 200 time units. The results shown here are the average value of the 10 networks.

The key metric for evaluating a localization algorithm is location accuracy. In this paper we use *location error* of a common node to measure its location accuracy. A common node's *location error* is defined as the distance between its real position and its estimated position measured as a multiple in r. For an algorithm, we define its *average location error* as the average of all its common nodes' *location error*.

A. Convergence

Fig.6 shows how average location error of different algorithms vary in each time unit. We can find out that our algorithm and MCB converge faster than SMCL. This is because in our algorithm and MCB most common nodes can get their position estimation in the first several time units, so the location error converges faster than in SMCL.

MSL* and MSL don't converge in this scenario. The reason may be that MSL* and MSL use common neighbors just like using anchor neighbors, which requires that position estimations of common nodes should be accurate enough. The position estimations of common nodes are not very accurate in this scenario, so MSL* and MSL don't converge.

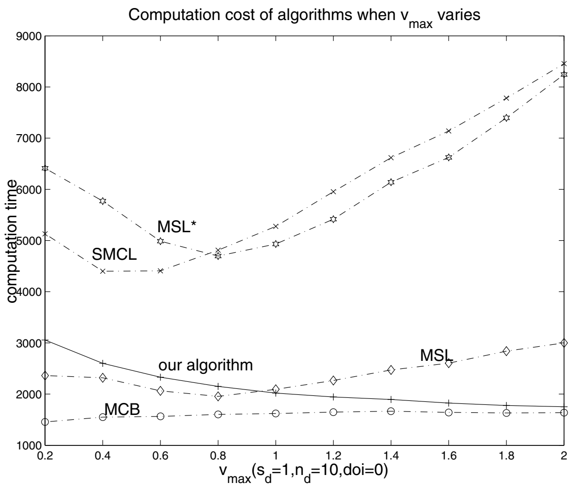

Fig. 7. Computation cost when v_{max} increases.

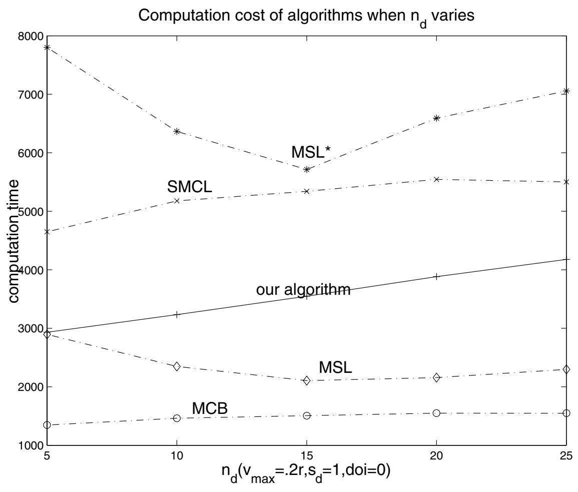

Fig. 9. Computation cost when n_d increases.

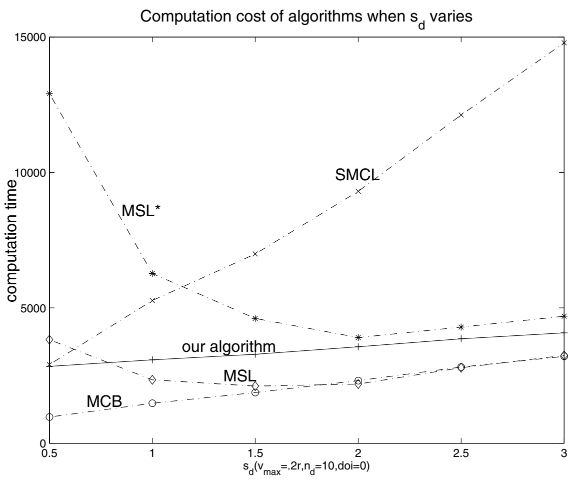

Fig. 8. Computation cost when s_d increases.

B. Computation cost

Our algorithm increases the probability that a sample candidate is reserved after filtering. However, using this probability as the metric of evaluating computation efficiency is unfair for SMCL and MCB because the filtering step in our algorithm needs more computation cost. So we use the average time needed to get a valid sample as the metric to evaluate computation efficiency. We count time needed to get a valid sample in the following way: to generate a sample candidate costs 1 computation unit; in the filtering step (or the weight computing step in MSL* and MSL), each comparison costs 1 computation unit. To be fair, we also count the time used to build bounding-box in our algorithm and MCB: each min or max operation costs 1 computation unit.

Fig.7 shows how v_{max} affects computation cost of different algorithms. In Fig.7 we can find out that, as we have pointed out in section II-B, the computation cost of SMCL increases when v_{max} increases. The computation cost of our algorithms reduces when v_{max} increases. This is because when v_{max} increases, the constraints introduced by common neighbors become weaker and more sample candidates are viewed as valid samples in the filtering step. Our algorithms is a little less computation efficient than MCB because we use common neighbors in the filtering step. However, the location accuracy of our algorithm is higher than MCB, which will be shown in next subsection.

Fig.8 shows how s_d affects the computation cost of different algorithms. Also as we have pointed out in section II-B, the computation cost of SMCL increases dramatically when s_d increases. This is due to two effects caused by the increase of s_d: it makes a sample candidate has high possibility to be dropped and it also need more computation time in the filtering step. Both our algorithm and MCB have less computation cost than SMCL because the sample candidates area is reduced dramatically using bounding-box.

However, because our algorithm also uses common neighbors in the filtering step, the computation cost will increase when n_d increases. Fig.9 shows how n_d affects the computation cost in different algorithms. We can see that computation cost only grows slightly when n_d grows. The reason is that although large n_d will increase computation cost in each filtering step, the probability that a sample candidate is a valid sample also increases because the sample area is also reduced when n_d increases. So the computation cost doesn't grow very much.

C. Location accuracy

According to our analysis in section IV-A, the location accuracy of our algorithm and MSL* (MSL) will be improved when n_d increases. Fig.10 shows how location accuracy is affected when n_d increases in different algorithms.

¿From Fig.10 we can find out that in our algorithm the average location error becomes smaller when n_d increases. n_d has almost no effect on the location accuracy in SMCL. There is a little improvement in location accuracy in SMCL when n_d increases from 5 to 10 because that some of 2-hop

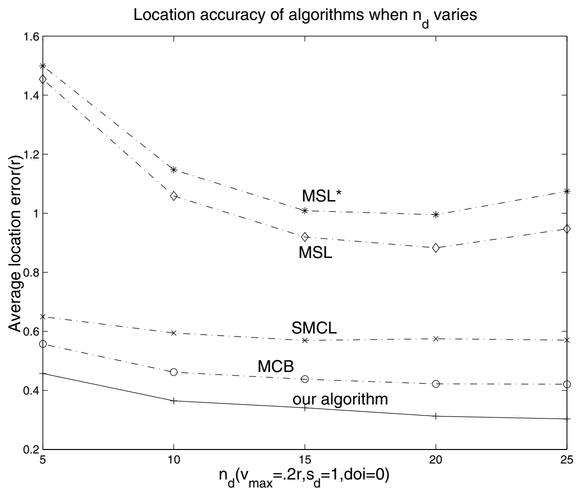

Fig. 10. Location accuracy improves when n_d increases.

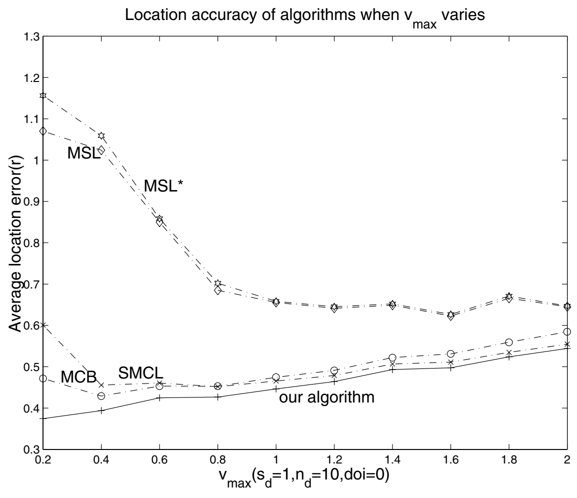

Fig. 12. Location accuracy degrades when v_{max} increases.

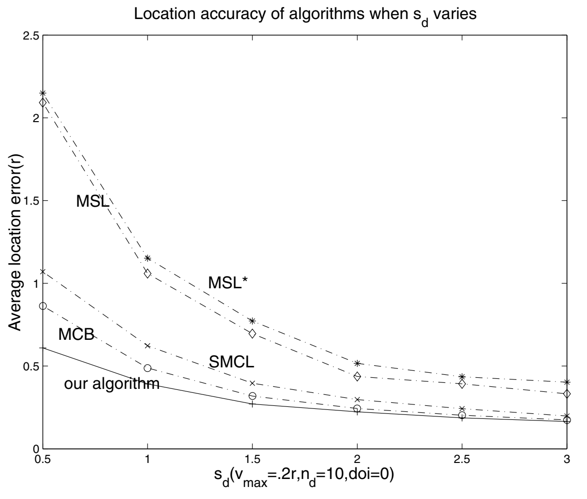

Fig. 11. Location accuracy improves when s_d increases.

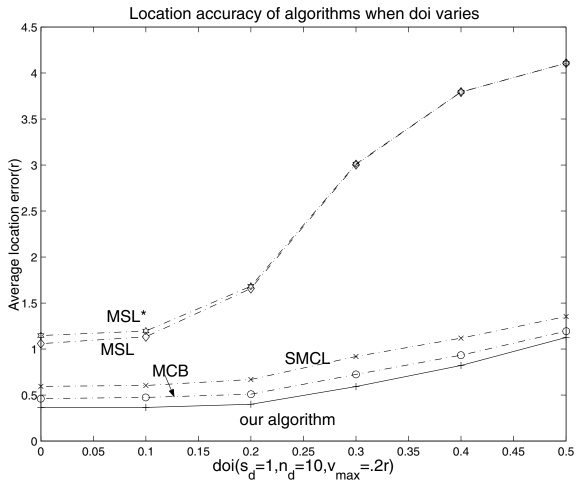

Fig. 13. Impact of Irregularity.

anchor neighbors cannot be found when n_d is less than 10. When n_d is higher than 10, n_d has no effects on location accuracy in SMCL. MCB has higher location accuracy than SMCL because it also uses position estimation of itself in last time unit to improve location accuracy.

As we have pointed out in section II-D, MSL* and MSL need high anchor density and high node density to converge. In the scenario presented here, location accuracy of MSL* and MSL are even lower than original SMCL.

Using the analyses in section IV-A, we know that the location accuracy will be improved when s_d increases. Fig.11 shows how s_d impacts location accuracy in different algorithms. From Fig.11 we can find out that location accuracy of all algorithms are improved when s_d increases. When s_d is large, the improvement introduced by common neighbors becomes smaller. However, our algorithm always outperforms SMCL and MCB. The location accuracy of our algorithm using anchor degree s_d approximately equals to location accuracy of SMCL using anchor degree $s_d + 0.5$.

Nodes' speed will also impact location accuracy of algorithms. Fig.12 shows how v_{max} effects location accuracy in different algorithms. We can see that when v_{max} increases, the location accuracy of our algorithm degrades but still always higher than other algorithms.

D. Impact of radio irregularity

We conduct simulations to test the impact of radio irregularity on location accuracy in different algorithms. The results are shown in Fig.13.

We use *degree of irregularity* (DOI) [26] to denote the maximum radio range variation in the direction of radio propagation. For example, if DOI=0.1, then the actual communication range in each direction is randomly chosen from [0.9r,1.1r]. From Fig.13 we can find out that both our algorithm and SMCL can tolerate small DOI (≤ 0.2). For MSL* and MSL, because they use both common neighbors and anchor neighbors in 2-hops, they are more sensitive to radio irregularity.

VI. Conclusions

We have presented an efficient localization algorithm for mobile sensor networks. The proposed algorithm improves the performance of existing SMC-based algorithms. The main drawback of these existing algorithms is high computation cost. In our algorithm, using the bounding-box method, we can reduce the scope of searching possible candidate samples, and thus reduce the time for finding the set of valid samples. Also, by using inaccurate position estimations of common neighbor nodes, the location accuracy can be improved. Simulation results show that our algorithm outperforms the existing SMC-based algorithms in terms of computation cost and location accuracy. We have also analytically derived the lower bound on the location error. In addition to the above benefits, the proposed algorithm has several other desirable features. Besides its location information, each determined node also knows the maximum location error. The algorithm can achieve higher location accuracy under higher node density. Also, even when there are only a few anchor nodes, most nodes can still get the position estimations.

Comparing with the existing SMC-based algorithms, the proposed algorithm incurs additional communication cost but the cost is rather reasonable: $O(n-m)$ in each time unit where n and m are the number of all the nodes and the number of anchor nodes in the network, respectively.

Acknowledgments

This research is partially supported by Hong Kong Research Grant Council under CERG grant PolyU 5103/06E, the Hong Kong Polytechnic University under the ICRG grant G-YF61, the China NSF grant (60573132,60673154), the China 973 project (2006CB303000), the China 863 project (2006AA01Z199) and the Jiangsu Provincial High-Tech Research Program (BK2006029). The authors also gratefully acknowledge the code sent by Hu and Evans [9] and the code sent by Masoomeh Rudafshani[17].

References

[1] J. Albowicz, A. Chen, and L. Zhang. Recursive position estimation in sensor networks. In *Proc. of 9th International Conference on Network Protocols(ICNP)*, pages 35–41, 2001.

[2] Aline Baggio and Koen Langendoen. Monte-carlo localization for mobile wireless sensor networks. In *MSN06*, pages 317–328, 2006.

[3] P. Biswas and Y Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proc. of 3rd International Symposium on Porcessing in Sensor Networks(IPSN)*, 2004.

[4] Bram Dil, Stefan Dulman, and Paul Havinga. Range-based localization in mobile sensor networks. In *ewsn06*, pages 164–179, 2006.

[5] Lance Doherty, Kristofer S.J. Pister, and Laurent El Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom'01)*, volume 3, pages 1655–1663, 2001.

[6] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *An Introduction to Sequential Monte Carlo Methods*, chapter 1, pages 3–14. Springer.

[7] David K. Goldenberg, Pascal Bihler, Ming Cao, Jia Fang, Brian D.O. Anderson, A. Stephen Morse, and Y. Richard Yang. Localization in sparse networks using sweeps. In *Proceedings of the 12th annual international conference on mobile computing and networksing (MobiCom'06)*, 2006.

[8] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th Annual International Conference on Mobie Computing and Networking(Mobicom'03)*, pages 81–95, 2003.

[9] Lingxuan Hu and David Evans. Localization for mobile sensor networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking(Mobicom'04)*, pages 45–57, Philadelphia, Pennsylvania, 2004. ACM Press.

[10] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design trdeoffs and early experiences with ZebraNet. In *Proceedings of the 10th international conference on architectural support for programming languages and operationg systems(ASPLOS-X)*, pages 96–107, October 2002.

[11] Brad Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking(Mobicom'00)*, pages 243–254, Boston, 2000. ACM Press.

[12] Mo Li and Yunhao Liu. Rendered path: Range-free localization in anisotropic sensor networks with holes. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking(Mobicom'07)*, pages 51–62, Montreal, Queébec, Canada, September 2007. ACM.

[13] H. Lim and J.C. Hou. Localization for anisotropic sensor networks. In *Proc. of 24th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM 2005)*, pages 138–149, 2005.

[14] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless Sensor Networks for Habitat Monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, September 2002.

[15] W. Mendenhall, R.L. Scheaffer, and D.D. Wackerly. *Mathematical statistics with applications*. Duxbury Press Boston, 1981.

[16] Dragos Niculescu and Badri Nath. Ad-hoc positioning system(aps). In *Proceedings of 2001 IEEE Global Telecommunications Conference(Globecom'01)*, pages 2926–2931, San Antonio, TX, USA, May 2001. IEEE Communications Society.

[17] Masoomeh Rudafshani and Suprakash Datta. Localization in wireless sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks(IPSN'07)*, pages 51–60, Cambridge, Massachusetts, USA, 2007. ACM.

[18] Andreas Savvides, Chih-Chieh Han, and Mani B. Srivastava. Dynamic fine-grained localization in ad-hoc sensor networks. In *Proceedings of the 7th Annual international Conference on Mobile Computing and Networking(Mobicom'01)*, pages 166–179, Rome, Italy, 2001. ACM.

[19] Y. Shang and W. Ruml. Improved MDS-Based localization. In *Proc. of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies(Infocom'04)*, pages 2640–2651, 2004.

[20] Y. Shang, W. Ruml, Y. Zhang, and Fromherz MPJ. Localization from mere connectivity. In *Proceedings of the 4th ACM Internatioal Symposium on Mobile Ad Hoc Networking and Computing(Mobihoc'03)*, 2003.

[21] Chen Wang and Li Xiao. Locating Sensors in Concave Areas. In *Proceedings of 25th IEEE International Conference on Computer Communications(Infocom'06)*, pages 1–12, April 2006.

[22] Tim Wark, Chris Crossman, Wen Hu, et al. The design and evaluation of a mobile sensor/actuator network for autonomous animal control. In *Proceedings of the 6th international conference on Information processing in sensor networks(IPSN'07)*, pages 206–215. ACM, 2007.

[23] Hongyi Wu, Chong Wang, and Nian-Feng Tzeng. Novel self-configurable positioning technique for multihop wireless networks. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 13(3):609–621, June 2005.

[24] Mao Ye, Chengfa Li, Guihai Chen, and Jie Wu. Eecs: an energy efficient clustering scheme in wireless sensor networks. In *24th ieee international performance, computing and communication conference(IPCCC2005)*, pages 535–540, April 2005.

[25] Jiyong Yi, Sungwon Yang, and Hojung Cha. Multi-hop-based monte carlo localization for mobile sensor networks. In *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07)*, pages 162–171, June 2007.

[26] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *Proc. of MobiSYS'04*, 2004.