PROBABILISTIC LOCALIZATION OF MOBILE AD HOC NETWORKS

by

RUI HUANG

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2006

To my grandmother 葛蕙芳 and my uncle 傅锐 (my father figure)

who set the example and who made me who I am.

ACKNOWLEDGEMENTS

I would like to thank my family for their patience and support throughout the years.

I am grateful to all the teachers who taught me including all my committee members for their guidance.

Finally, I would like to express my deep gratitude to Dr. Gergely V. Záruba for his wisdom and encouragement to help me complete this work.

November 15, 2006

ABSTRACT

PROBABILISTIC LOCALIZATION OF MOBILE AD HOC NETWORKS

Publication No. ______

RUI HUANG, Ph.D.

The University of Texas at Arlington, 2006

Supervising Professor: Gergely V. Záruba and Sajal K. Das

The mobile ad hoc network localization problem deals with estimating the physical location of the nodes that do not have a direct way (e.g., GPS) to determine their own locations. Being an enabling technology that is considered essential to the success of the future implementation of ad hoc networks in the real world, localization is a fundamental problem that needs to be solved with the best possible accuracy and efficiency. For this research, we study the localization problem in its various incarnations such as localization through static beacons, mobile beacons, dynamically deployed beacons and link longevity estimation based on relative locations. We will show the fundamental difficulty of the localization problem using computational complexity theories. We will also propose a probabilistic framework that serves as an approximation framework for this difficult problem. We will demonstrate the effectiveness of this framework via analysis and simulation.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND MOTIVATION

1.1 Introduction

Mobile Ad Hoc Networks (MANETs) are infrastructure-less networks that form on the fly as network nodes move in and out of each other's transmission range. Since MANET serves as an abstract model and concept that can be seen as a superset of diverse sub-areas such as sensor networks, mesh networks or an enabler for pervasive computing, it has attracted heavy research interest in the past several years. A major advantage of MANETs over regular wired or wireless networks is in their infrastructure-less nature as they can potentially be deployed more rapidly and less expensively than infrastructure-based networks. However, the lack of an underlying explicit infrastructure also becomes a major disadvantage in adapting MANETs to a wider array of applications, since existing network algorithms and protocols are not "plug-in" solutions for such dynamic networks. New algorithms need to be, and are being designed for such fundamental network tasks as addressing, topology discovery and routing.

Location discovery is emerging as one of the more important tasks as it has been observed and shown that (semi-) accurate location information can greatly improve the performance of other MANET tasks such as routing, energy conservation, or maintaining network security. For instance, algorithms such as LAR [42], GRID [51], and GOAFR+ [46] rely on the location information to provide more stable routes during unicast route discovery. The availability of location information is also required for geocast (multicast based on geographic information [37]) algorithms such as LBM [43], GeoGRID [52] and PBM [56]. To minimize the power consumption, the GAF algorithm [83] uses location information to effectively modify the network density by turning off certain nodes at particular instances. Furthermore, in [31], the authors have shown that wormhole attacks can be effectively prevented when location information is available. As more algorithms are being proposed to exploit the location information in the network, it is clear that obtaining such information efficiently and accurately becomes of greater importance.

This work deals with localization problem in general. In particular, we propose a probabilistic approach to a number of sub-problems within the localization domain. In this chapter, We first establish the importance of location discovery under various flavors of MANET by surveying some of its important applications. We then discuss the underlying difficulty of the localization problem, and the various measurement types available to solve this problem. Finally, we describe the overall organizational structure that will serve as the road map to the rest of this work.

1.2 Survey of Applications

There have been numerous algorithms proposed for MANETs that rely on localization data. In this section, we provide a brief survey of them; we divided them into four categories based on their functionalities: unicast routing, multicast routing, energy consideration, and network security.

1.2.1 Unicast Routing

Routing is a specially challenging task for MANETs because their frequent infrastructural change implies the underlying instability of any established routes. As such, routes are needed to be frequently rediscovered, reestablished, and repaired. In general, routing (i.e., route discovery and repair) involves flooding the routing control packets throughout the network. Flooding can often be quite expensive in terms of delay and bandwidth usage it incurs, both of which can greatly affect the network performance.

Thus, there is a strong incentive to design efficient routing algorithms that minimize the overhead caused by any unnecessary flooding. Unicast routing based on location information, often called *geometric routing* or *location based routing*, has shown to be a viable solution to this problem.

Location-Aided Routing (LAR) [42] protocol is the first MANET routing algorithm proposed that uses location data. In LAR, every node is assumed to know its own location, and each individual location is then broadcast throughout the network. Thus, at any time t, every node knows the locations of any other nodes at some previous time $< t$. Based on this location information and an estimated velocity, a node can derive an estimated location range, called "expected zone", of a target node at the current time. The routing request packets can be directed to search for the target node only at this expected zone. Global flooding is performed only after the location based routing request has failed. Limiting route discovery to a smaller expected zone with LAR reduces the number of routing requests compared to the standard flooding scheme, in which the routing request packets have to cover the entire network.

GRID [51] protocol uses location information as a way to form geographical clusters within the network. Based on node locations and their residency within a pre-determined grid system, nodes within the same grid block are grouped into a cluster. A cluster head ("gateway" in [51]) is then selected for each grid block. The cluster head is responsible to service the routing packets. Furthermore, the cluster head can monitor the status of existing routes and reroute packets as deemed necessary. Since the cluster formation effectively simplifies the network topology, the routing overhead is reduced. A critical requirement of forming such clusters is the availability of node location information.

In [46], the authors provide some theoretic bound to the geometric routing problem and propose an algorithm called GOAFR+. Assuming that node locations are known, GOAFR+ first tries to greedily route the packet by forwarding it to the neighbor located closest to the destination. However, such greedy selection does not guarantee message delivery since the intermediate node closest to the destination might not have a route to it. In such cases, GOAFR+ explores the boundaries of the faces of a planarized network graph by employing the local right hand rule (i.e., always turn right) to escape the local minimum. This method of escaping local minimums is also called “parameter routing,” which is used in a number of other location based routing protocols as well.

Performance-wise, simulations performed by [42] and [51] show up to 50% of reduction in routing packets when using geometric routing compared to standard flooding. Since the overhead of flooding is proportional to network density, it is observed that the amount of this performance increase becomes more significant when network density is increased. Furthermore, although the routing performance is impacted by the localization error, such impact is observed to be minimal. This indicates that in the case of routing highly precise location data is not required. After all, location data is used by routing algorithms to give a direction that guides the routing packets; imprecise location data can still be used as long as the general direction is valid.

1.2.2 Multicast Routing

Similar to unicast routing, multicast routing can also benefit from location data. Multicast routing using geometric information is often referred to in the literature as *geocast routing*. The Location-Based Multicast (LBM) algorithm [43] is a multicast extension to the unicast Location-Aided Routing (LAR). Like LAR, which forwards the routing requests according to the location of the destination node, LBM forwards the requests according to the direction of the geocast region that contains all the multicast destinations. GeoGRID [52] is the multicast extension to GRID [51]. Like in GRID, location information is used by GeoGRID to identify the grid block where nodes reside. Multicast is done through the gateway node selected at each grid block. Based on the location of the source node and the geocast region, LBM and GeoGRID define a "forwarding region" that contains the intermediate nodes responsible for forwarding requests. The size and shape of the forwarding region have a direct impact to the overall performance. Shapes such as rectangles and cones have been proposed in [43].

While the standard shapes such as rectangles and cones work well in most cases, there are situations where viable routes exist only outside the forwarding region. For instance, a network can be partitioned into two sub-networks connected only through a narrow linkage due to some obstacles (e.g., two islands connected by a bridge). When the source and the destination are in separate partitions, a geometrically defined forwarding region is unlikely to cover the linkage. To prevent routing failure in such case, a routing zone based on Voronoi diagrams was proposed in [77], which partitions the network graph based on the proximity of the nodes. Again, the proximity information relies on localization information.

The Position-Based Multicast (PBM) protocol proposed in [56] attempts to optimize the multicast tree it generates by minimizing the overall path length and the overall bandwidth usage; two often contradictory objectives. To minimize the overall path length, PMB takes a greedy approach using location information. At each intermediate node, the packet is forwarded to a set of neighbors based on their overall distances to the multicast destinations. In particular, a set of the neighbors with the minimum overall distance to every destination is selected as the next set of forwarding nodes. To take in account of the bandwidth usage, the greedy selection also weighs in the size of the forwarding set in order to minimize that as well. PBM also uses parameter routing to deal with local minimums. Both greedy routing and parameter routing employed by PBM rely on the location information.

1.2.3 Power Management

MANET is often being used as the model for sensor networks. Due to the recent emergence of interest in pervasive computing, sensor networks have been receiving heavy research efforts. One of the major challenges of sensor networks is power management. Since sensors are commonly small in size and battery powered, conserving the energy would prolong their service time and thus the lifespan of the entire network. The Geographical Adaptive Fidelity (GAF) algorithm [83] is a network topology management algorithm with reducing energy consumption as its primary objective. The idea behind GAF is that there are often a large number of nodes that are redundant during packet routing in MANET. If the redundant nodes can be identified, they can then turn off their radio to save energy. For GAF, the identification of redundant nodes is accomplished by analyzing the relative location information among the neighboring nodes. More specifically, GAF divides the network into virtual grids such that all nodes in grid block A are the neighbors of all nodes in grid block B. This way, all nodes within the same virtual grid block can be considered equivalent. To conserve energy during packet routing, GAF only turns on the radio for one of the nodes in each grid block. The active node is periodically "round-robinned" to achieve load-balancing. Analysis and simulations performed in [83] show that GAF can reduce overall energy consumption by 40% to 60%.

1.2.4 Security

In [31], the authors proposed a technique called "packet leashes" to defend against wormhole attacks in MANETs. A wormhole attack is a type of security breach where an adversary intercepts incoming packets and tunnels them to another part of the network via a single long-range directional wireless link or through a direct wired link. From there, the adversary can retransmit the packets to the network. Note that this type of "capture-and-retransmit" attack can be immune to common packet encryption methods since adversaries do not need to read the packet content. Wormhole attacks can severely disrupt ad hoc routing protocols such as AODV or DSR, and cause a denial of service to the network. The core of "packet leashes" is based on two assumptions: i) all nodes know their own locations, and ii) all nodes are synchronized. To enable packet leashes, the sender node encloses its location and transmission time-stamp within the packet. At the receiver node, the packet leash is validated against the receiver's own location and clock. In particular, the sender location information gives the distance from the original sender to the receiver, and the time-stamp gives the transmission duration of the packet. Based on the transmission duration and signal propagation model, factored in some error tolerance, the receiver can validate the estimated distance the packet has traveled against the true distance to tell if the packet is indeed coming from the original sender or an imposter at some other location. Thus, the location information and time-stamp provide a virtual leash to limit the effective range of the packet so that it cannot be exploited by wormhole attackers.

From the previous discussion of the location-dependent algorithms that encompass a number of different domains, it is quite obvious that providing location information (i.e., localization) to MANET is becoming an increasingly important task. In fact, localization is now widely regarded as an "enabling technology" for MANET that needs to be addressed before other location-dependent techniques can be realized in the real world [66].

1.3 Challenges

A direct way of obtaining location information at MANET nodes is to install GPS receivers on each node. However, this is currently impractical as GPS receivers are still relatively expensive, power-hungry, and require clear line of sight (i.e., making indoor usage impossible) to several earth-bound satellites. In sensor networks devices are imagined as small as possible while operating on a very restricted power source, thus it may not be feasible to install GPS receivers onto all sensor nodes. *Localization* in MANET refers to the problem of finding the locations of those non-GPS enabled nodes based on limited information such as some known beacon locations and ranging distances or angles among the neighbors. The localization problem is hard for a number of reasons as explained in the following subsections.

1.3.1 Geometric limitations

To pinpoint its exact location, a node needs to know the locations of at least three beacons together with its distance from each of these beacons. Alternatively, nodes could calculate their own location based on a distance and an (absolute) angle measurement from one beacon. Even if obtaining such measurements were possible and the measurements were exact, guaranteeing that (several) beacons surround each node is impossible as MANETs may be randomly deployed and in general only a small percentage of nodes are indeed beacons. Thus, many localization algorithms take advantage of multi-hop information, i.e., estimating node locations based on other nodes' location estimates.

1.3.2 Measurement Availability

For localization algorithms that require distance or angle measurements, certain sensory devices will need to be available to provide such readings. However, it is likely that not all nodes have the same sensory capacity. In other words, there is a need for the localization algorithm to work in a heterogeneous environment with different location sensory capabilities.

1.3.3 Measurement Error and Error Propagation

Even when distance or angle of arrival measurement devices are available, there is a general consensus that those measurements are prone to errors. For instance, a distance measurement based on received signal strength indication (RSSI) reading is prone to multi-path fading and far field scattering. The error can be especially high when there are a significant number of obstacles in-between the sender and the receiver. Furthermore, since many localization algorithms require measurements from nodes several hops away, the measurement error is likely to aggregate along the path and may eventually completely throw off the location estimate.

In spite of its hardness, there has been an increasing amount of research effort targeting the localization problem in recent years. The amount of effort is well justified because localization is considered an enabling technology that needs to be resolved with the best possible outcome upon which other location-dependent technologies for MANETs can be successfully employed. Thus, researchers have been working on the problem in both hardware (i.e., improving the devices measurement accuracy) and software (i.e, improving the localization algorithm).

1.4 Common Measurement Types

In this section, we survey the types of measurement data currently available to the localization problem. There are five general types of measurement as follows: i) connectivity only, ii) RSSI (radio signal strength indicator) ranging, iii) ToA (time of arrival) ranging, iv) AoA (angle of arrival), and v) Interferometric ranging.

1.4.1 Connectivity Only Measurement

At a minimum, a node can detect connectivity to its neighbors, i.e., its one-hop neighborhood. The connectivity only measurement is a binary reading between two nodes of either “true” or “false” indicating whether they are neighbors. Based on this connectivity information, one can derive the general proximity of the nodes and use it as a way to localize the network.

1.4.2 RSSI Ranging Measurement

A node can be localized using multilateration if the distances (i.e., the ranges) to three or more known locations are obtained. The distances can be obtained, e.g., by measuring RSSI (received signal strength indication) or ToA (time of arrival). In RSSI, the receiver measures the received signal strength and compares it with the transmitted signal strength. The difference (in dB) is then applied to the inverse of the signal propagation model to provide a distance measurement. Sensors that measure RSSI are widely available to mobile devices. Indeed, most off-the-shelf technologies implicitly provide such information (e.g., most WiFi, Bluetooth and IEEE802.15.4 chipsets do). The drawback of RSSI based measurements is that they can be very inaccurate because an exact model of the propagation environment is often unavailable.

In the outdoor environment with minimal obstacles, signal propagation decay is proportional to d^{n_p}, where d is the distance the signal has traveled and n_p is a path loss exponent. However, in the actual environment where obstacles exist, multipath signals and shadowing become two major sources of noises that impact the actual RSSI. In general, those noises are commonly modeled as a random process during localization.

Let $P_{i,j}$ be the RSSI (in dB) obtained at the receiver node j from the sender node i. $P_{i,j}$ is commonly modeled as a Normal distribution [66]

$$P_{i,j} = N(\bar{P}_{i,j}, \sigma_{dB}^2) \tag{1.1}$$

where $\bar{P}_{i,j}$ is the mean power in dB and σ_{dB}^2 is the variance caused by noise factor such as shadowing. $\bar{P}_{i,j}$ is further defined as the power reduction from a reference location:

$$\bar{P}_{i,j} = P_0 - 10n_p log_{10}(d_{i,j}/d_0) \tag{1.2}$$

where P_0 is the power at a reference location at the distance d_0 (commonly $d_0 = 1m$). n_p is an environment-dependent path loss exponent that is assumed to be known from prior measurements (theoretically $n_p = 2$). $d_{i,j}$ is the Euclidean distance between nodes i and j.

1.4.3 ToA Ranging Measurement

Although ToA is used for radio signals in GPS, it is mostly used in context of acoustic signals in inexpensive ToA tracking (as propagation speeds are five orders of magnitude less). ToA measures the time acoustic signals travel from the sender to the receiver. The distance between nodes is obtained by multiplying this time with the signal propagation speed. In spite of the additive noise and multipath, in general distance measures based on ToA are more accurate than RSSI based measures. However, special acoustic transceivers have to be employed on each node and synchronization among the nodes needs to be established. Sensor network clock synchronization algorithms accurate to the order of $10\mu s$ have been reported [74]. As mentioned earlier, ToA may also be used together with radio signals, but current technology is not mature enough to provide a satisfactory precision over smaller distances inexpensively.

Let i be the sender node and j be the receiver node, ToA measurement $T_{i,j}$ is often modeled as a Normal distribution [66]:

$$T_{i,j} = N(d_{i,j}/c, \sigma_T^2)$$

where $d_{i,j}$ is the euclidean distance between i and j, c is the signal propagation speed, and σ_T^2 is the variance caused by noise.

1.4.4 AoA Measurement

A node can be localized if the angles between itself and two beacons are known. Thus, it is possible to localize the network based on the angle information (i.e., bearing, or angle of arrival (AoA). Currently, there is no off-the-self device that offers AoA sensing capability. However, a number of prototype devices are available. For instance, Cricket Compass [69] is a small form device that uses ultrasonic measurements and fixed beacons to obtain acoustic signal orientations. In [61], a rotating directional antenna is attached to an 801.11b base station. By measuring the maximum received signal strength, a median error of 22° can be obtained from the sensor. The challenge here is to design an AoA sensing device that has small form factor and low energy consumption. In [14], the authors outline a solution with a ring of charge-coupled devices (CCDs) to measure AoA with relatively low energy consumption.

In general, AoA is also modeled as a Normal distribution. Let the true angle between the sender i and j be $a_{i,j}$, the AoA measurement between i and j is therefore

$$A_{i,j} = N(a_{i,j}, \sigma_a^2)$$

where σ_a^2 is the angle variance. Theoretical results for acoustic-based AoA estimation show a standard deviation σ_a between 2° to 6°, depending on the range [65]. An RSSI based AoA method with σ_a on the order of 3° has been reported [2].

1.4.5 Interferometric Ranging Measurement

Interferometric ranging is a "widely used technique in both radio and optical astronomy to determine the precise angular position of celestial bodies as well as objects on the ground [47]." Interferometric ranging exploits the property that the relative phase offset between two receivers determines their distances to the two simultaneous senders. Due to the recent advance in hardware, it is now possible to implement interferometric ranging sensors in a much smaller form factor so that it can be used for localization [55]. Because there has been relatively little work on localization using interferometric ranging, we devote a separate chapter (Chapter 3) to study some of its fundamental properties.

1.5 Content Overview and Organization

In this work, we propose a probabilistic localization framework for a number of related yet different sub-problems of localization. In Chapter 2, we introduce the general ad hoc localization problem, in which the nodes collaborate together to derive their locations from a small subset of beacon nodes. Our solution to this problem involves a probabilistic framework based on particle filters that take into account imprecise location information. The framework also supports localization of networks with heterogeneous sensor types. We also introduce some of the basic theoretic concepts of localizability and error bounds in collaborative localization.

In Chapter 3, we proceed to study localization using interferometric ranging in detail. In particular, we provide some fundamental complexity proofs of this problem. We show that localization using interferometric ranging is an NP-Hard problem. We also propose an iterative localization algorithm designed to work with interferometric ranging sensors.

We consider the mobility issue in ad hoc networks in Chapter 4. Because of node movement, the established communication link between two mobile nodes is temporary and can be disconnected. Since the link disconnection affects routing, it is desirable to predict its occurrence so that backup routes can be established ahead of time without interrupting the service. We present three measurement based link longevity predictors based on extended Kalman filters.

We cover the dynamic beacon deployment for localization in Chapter 5. The dynamic beacon deployment problem looks for a strategy to deploy beacons one after another so that the entire network can be localized. We show that a superset of this problem is NP-Complete, and thus the beacon deployment problem itself is likely to be NP-Complete as well. We then propose a greedy approximation algorithm with logarithmic approximation ratio to this problem. In reality, the beacons need to be deployed in an online fashion, in which the algorithm does not know the impact of the deployed beacon until after it has been deployed. Thus, we have to modify the greedy algorithm to work in the online environment. We also propose a deployment strategy that explicitly considers the localization error using the concept of Cramer Rao bounds (CRB).

Other than the traditional collaborative methods, a mobile beacon can be used to localize networks. In Chapter 6, we apply our probabilistic framework of particle filters to the localization problem using mobile beacons. We adapt the localization protocol so that the filtering runs exclusively on the mobile beacon itself. Thus, we alleviate the sensors from the computational resource needed to perform localization. It also provides a more secure environment for localization.

While there have been various localization algorithms proposed for the mobile beacon problem, relatively little work has been done in the path planning of the mobile beacon. In Chapter 7, we study both the static and dynamic path planning problem for localization. For the static path planning, we propose two new path types that provide better localization coverage and accuracy than previous ones without increasing the path length. We also study the dynamic path planning using a similar greedy strategy as the beacon deployment problem. We show that while the static paths would work better in uniformly deployed sensor networks, the dynamic paths are better suited to heavily clustered networks.

CHAPTER 2

GENERAL AD HOC LOCALIZATION PROBLEM

In this chapter, we study the localization problem in its most general and well-studied form. We assume that a small subset of the nodes know their locations, for instance, via an onboard GPS receiver, and the question is to derive the locations of all other nodes using some kind of measurement such as RSSI ranging. This problem has been heavily studied because it can serve as an abstract model of many real-world localization problems in the field of sensor networks, ad hoc networks and pervasive computing. Furthermore, this highly abstract model has the advantage of being closely related to the graph realization problem, which has been studied in the past 20 years in the domain of graph theory.

2.1 Problem Definition

A network is commonly modeled as a graph, in which network devices are represented as vertices and communication links between devices as edges. Depending on the actual network scenario, the graph can be either directed (indicating an asymmetric behavior of the communication links) or undirected (indicating symmetric communication links). As typical of other works in localization, we only consider the undirected graphs in the network graph, although some results can be extended to directed graphs as well. Within the context of a network graph, we define the *general ad hoc localization problem (GAHLP)* as the task of estimating the physical location of all nodes given that only a subset of nodes (i.e., *beacons* or *anchors*) know their exact location. More formally, given a network graph $G = (V, E)$ where a subset of the nodes $\{V\}$ are location-aware

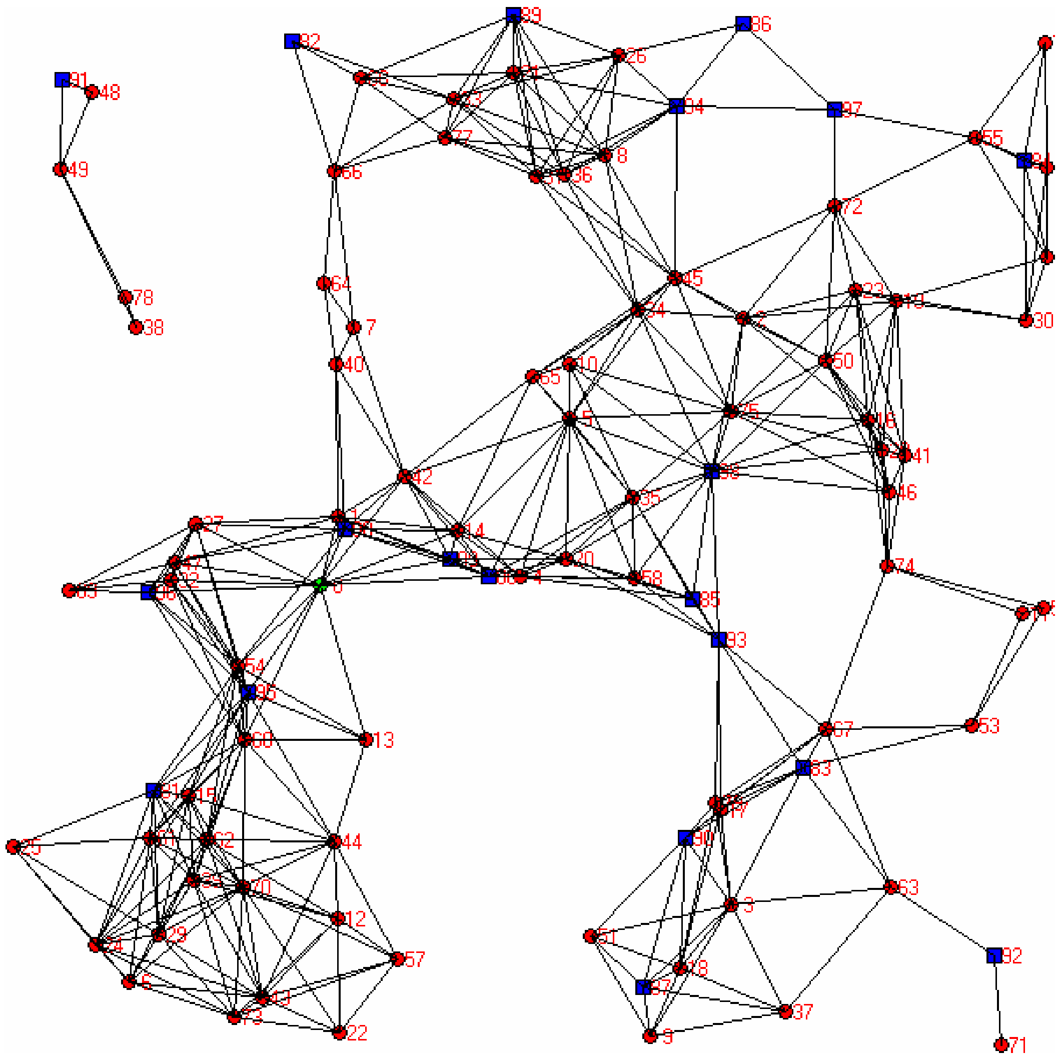


Figure 2.1. A symmetric network of 80 regular nodes (circles) and 20 beacons (squares). A link exists between two nodes if they are neighbors. The node connectivity (nodal degree) of this network is 7.6.

beacons $\{V_{beacon}\}$, the objective of a *general ad hoc localization algorithm (GAHLA)* is to find the locations of *regular* nodes $\{V\} - \{V_{beacon}\}$.

Figure.2.1 shows such a network graph with 80 regular nodes and 20 beacons. The beacons and regular nodes are depicted as squares and circles, respectively.

This is the localization problem formulated in the most general form, and it is also the most widely studied by researchers. There are two variations of this problem: i) when regular nodes are stationary, and ii) when the regular nodes are mobile. The mobile scenario is generally more difficult because node movement could invalidate previously obtained localization result. In the mobile case, either the localization algorithm has to run continuously or a separate correction algorithm is needed to accommodate the location change.

In this chapter, we study the GAHLP in the following three sub-areas: i) theoretic results derived from graph theory, ii) impact of different measurement types and noises, and iii) the actual localization algorithms.

2.2 Theoretic Hardness

The solution of GAHLP depends on the measurement type. Four out of the five types of measurements listed in Chapter 1, namely connectivity (i.e., a binary state indicating if two nodes are neighbors), RSSI and ToA ranging (i.e., the distance between two neighbors), and angle of arrival (AoA), have been previously studied in their computational hardness. However, GAHLP, even in stationary networks, is very hard for all the measurement types. Its hardness can be illustrated by a number of recent results in graph theory. In this section, we give a brief overview of the primary results in this field.

For the localization problem, the analysis of theoretic hardness asks two questions: i) how difficult it is to tell whether a network *can* be localizable (i.e., the localizability), and ii) how difficult it is to actually localize it (i.e., the realization). The branches of graph theory that study these two questions are called *rigidity theory* and *graph realization*. Rigidity theory deals with the first question that asks under what condition a graph will be rigid in the sense that there exists a unique localization result given the measurements. The test of graph rigidity is important because if the graph cannot be localized with a unique localization then it would be pointless trying to localize it. If the graph passes the localizability test, graph realization deals with the actual algorithm that localizes the network in Euclidean space. Note that it is a common practice in the theoretic analysis to assume that measurements are 100% accurate.

2.2.1 Localizability

We first consider the localizability test under the distance measurement (i.e., ranging).

Definition 2.2.1. *In d dimensions, a set of points are said to be in* general positions *if they do not lie in a proper subspace (i.e., three points in the plane do not lie on a line, and four points in space do not lie in a plane) [3].*

In the localizability test, we assume that the nodes are in *general positions* within the network graph. This assumption eliminates the special cases where some nodes are located at "unfortunate" locations. Those special cases are rare since with probability of 1 the nodes are in general positions within the continuous space. The next two definitions contain the qualifier "generically," meaning that we are considering nodes in general positions.

Definition 2.2.2. *A graph is* generically locally rigid *(also called* first-order rigid*) if it does not admit a continuous deformation other than global rotation, translation, and reflection [25, 3].*

Thus, the rigidity in general refers to the situation in a graph where there are no continuous motions of the vertices satisfying distance constraints on edges. While a locally rigid graph does not admit a continuous deformation, it can still admit discontinuous deformation such as in Figure 2.2, which would make it unlocalizable. To achieve localizability, we need additional conditions:

Definition 2.2.3. *A graph is* generically globally rigid *(also called* second-order rigid *or* redundantly rigid*) if the removal of any one edge would result in a graph that is also generically locally rigid [3].*

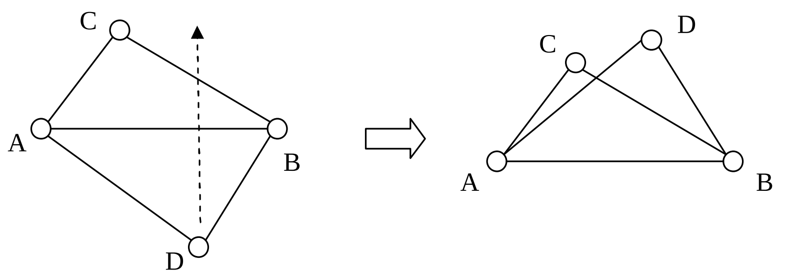


Figure 2.2. A locally rigid graph can still be unlocalizable.

Definition 2.2.4. *A graph is $k-$connected if it remains connected upon removal of any set of less than k vertices [3].*

Based on the above definitions, the following theorem gives the necessary and sufficient condition for distance-constrained network localizability in two dimensions.

Theorem 2.2.5. *The network is localizable in two dimensions if and only if the network graph is redundantly rigid and triconnected [29, 4].*

Theorem 2.2.5 holds only for two dimensions. The sufficient condition for higher dimension is currently unknown. To test the localizability, there exists a polynomial time algorithm ($O(n^2)$, where n is the number of vertices) that tests for the first-order rigidity (see [29] for one implementation). However, it is a known NP-Complete problem to test for the second-order rigidity of a graph [72]. A related but even more difficult problem is the *node localizability*, which asks if a particular node (instead of the entire network) is localizable. No sufficient condition of node localizability is currently known even in the two dimensional case, and thus no deterministic algorithm exists.

2.2.2 Realization

Given a network that is localizable, the graph realization problem asks for an algorithm to actually localize the network. The hardness of graph realization has been shown under the measurement of distance, angle and connectivity.

Theorem 2.2.6. *Graph realization is NP-Hard under distance measurement.*

Proved in [18] using a reduction from the known NP-Complete Set Partition problem.

Theorem 2.2.7. *Graph realization is NP-Hard under angle measurement.*

Proved in [8] using a reduction from the known NP-Complete 3SAT problem.

Theorem 2.2.8. *Graph realization is NP-Hard under unit disk connectivity measurement.*

Proved in [7, 45] using a reduction from the known NP-Complete 3SAT problem.

In Chapter 3, we will show that the graph realization is also NP-Hard under interferometric ranging measurement.

The above theorems clearly indicate the difficulty of the localization problem regardless of the measurement types. Based on the theoretic hardness of the problem, it is clear that any localization algorithm needs to be some form of stochastic optimization.

2.3 Measurement Noise and Cramer-Rao Bounds (CRB)

The above theoretic results indicate the general intractability of GAHLP even in the ideal case where measurements (such as edge distances) are 100% accurate. Thus, all proposed solutions have been some forms of stochastic optimization. Unfortunately, measurements in the real world are a far-cry from being accurate, and any optimization method has to deal with not only different measurement types but also noise. In this section, we study the impact of noise on the localization problem.

The localization inaccuracy attributed to the measurement types and noise can be mathematically qualified using Cramer-Rao Bounds (CRB) [65]. The CRB is a lower bound on the covariance of any unbiased location estimator that uses measurements such as RSSI, ToA, or AoA. Thus, the CRB indicates a lower bound of the estimation accuracy of a given network scenario regardless of the localization algorithm. In other words, with CRB we have a way to tell the best *any* localization algorithm can do given a particular network, measurement type and measurement noise scenario. CRB formulas of individual measurement types such as RSSI, ToA and AoA under most common noise models (mostly Gaussian) are currently known.

To illustrate how a CRB formula is derived, let us use RSSI as an example. Recall from Chapter 1 that RSSI can be modeled as a Normal distribution

$$P_{i,j} = N(\bar{P}_{i,j}, \sigma_{dB}^2) \tag{2.1}$$

where

$$\bar{P}_{i,j} = P_0 - 10n_p log_{10}(d_{i,j}/d_0) \tag{2.2}$$

Given an instance of the GAHLP with n unlocalized nodes and m beacons, the CRB of a problem instance can be obtained by constructing a covariance matrix (also called Fisher Information Matrix (FIM)) of all unknowns. In the two dimensional case, the problem of n unlocalized nodes constitutes $2n$ unknowns, namely, $x_1, x_2, ..., x_n, y_1, y_2, ..., y_n$. When the measurement model of RSSI is given as in equation (2.1) and (2.2) and assuming a bi-directional connectivity where node i can hear node j *iff* j can hear i, the FIM for RSSI is in the form of

$$FIM = \begin{pmatrix} F_{xx} & F_{xy} \\ F_{yx} & F_{yy} \end{pmatrix}$$

where $F_{yx} = F_{xy}^T$. Each element in the FIM can be obtained by taking the partial derivatives of the probability density function given by the specific measurement model with respect to the corresponding unknowns. In the case of RSSI measurements, the element is given as

$$[F_{xx}]_{k,l} = \begin{cases} \gamma \sum_{i \in H(k)} (x_k - x_i)^2 / d_{k,i}^4 & k = l \\ -\gamma I_{H(k)}(l)(x_k - x_l)^2 / d_{k,l}^4 & k \neq l \end{cases}$$

$$[F_{xy}]_{k,l} = \begin{cases} \gamma \sum_{i \in H(k)} (x_k - x_i)(y_k - y_i) / d_{k,i}^4 & k = l \\ -\gamma I_{H(k)}(l)(x_k - x_l)(y_k - y_l) / d_{k,l}^4 & k \neq l \end{cases}$$

$$[F_{yy}]_{k,l} = \begin{cases} \gamma \sum_{i \in H(k)} (y_k - y_i)^2 / d_{k,i}^4 & k = l \\ -\gamma I_{H(k)}(l)(y_k - y_l)^2 / d_{k,l}^4 & k \neq l \end{cases}$$

Here, γ is a constant derived from the parameters in the measurement model of Equation (2.1) and (2.2), where $\gamma = (\frac{10n_p}{\sigma_{dB} log10})^2$. $H(k)$ is the set of neighboring nodes of a node k, that is, $i \in H(k)$ *iff* node i can hear k. $I_{H(k)}(l)$ is a binary indicator function that returns 1 if l is in $H(k)$ and 0 otherwise.

To obtain the CRB matrix, we invert the FIM and examine its diagonal elements. Let the σ_i^2 be the location variance of the ith unknown node, where $\sigma_i^2 = Var(x_i) + Var(y_i)$. Then, the CRB gives a lower bound of σ_i^2 as

$$\sigma_i^2 \geq (FIM^{-1})_{i,i} + (FIM^{-1})_{i+n,i+n}$$

For the sample network given in Figure 2.1, the CRB of the localization error is shown in Figure 2.3 as rings of radius $= \sigma_i$. Here, we assume the measurement model to be RSSI with the path loss exponent $n_p = 1$ and the standard deviation of the noise $\sigma_{dB} = 0.7$. A ring with smaller radius (i.e., a smaller CRB) signals that more accurate localization result can be theoretically obtained. Conversely, a larger ring indicates a larger localization variance and thus a less accurate result. In the figure, two types of nodes do not have rings. First, all beacons have a CRB of 0. There are also regular nodes that have infinite CRB indicating that those nodes are theoretically impossible to localize. The latter case can be seen at nodes 38, 48, 49 and 78 in the top left corner. At the minimum, three beacons are needed to localize a connected network. However, those nodes at the top left corner are isolated to a different partition. Since they are connected to only one beacon (node 91), those nodes clearly cannot be localized. Other than those cases, the CRB rings at the main network partition clearly show the level of localization difficulty under various scenarios. In general, we observe that nodes closer to the beacons tend to have a smaller CRB than the ones that are several hops away. Even smaller CRB can be obtained when a node is closer to more than one beacon. All of the above observations match our common intuition about localization difficulty.

The CRB of ToA and AoA measurements can be similarly calculated [66, 65]. It is important to note that CRB is essentially a theoretic bound that depends on the measurement model. In the real world, its usefulness is limited by how accurate the measurement model reflects the reality. Nevertheless, CRB can be a useful tool in comparing various localization algorithms. It can be used to validate how close a particular algorithm can come to this theoretic lower bound and to see if there is any room for improvement in the algorithm design. In Chapter 5 and 7, we also provide an innovative usage of CRB in which we incorporate CRB as part of the algorithm.

2.4 Survey of Localization Algorithms

While there are various ways of classifying general ad hoc localization algorithms (GAHLA), we feel it is more logical to classify them according to the measurement assumptions as the four types: i) connectivity-only, ii) range-based, iii) angle-based, and iv) hybrid. Note that we omit the localization algorithms that use interferometric ranging here because they will be covered in Chapter 3. A comparison between the more well-

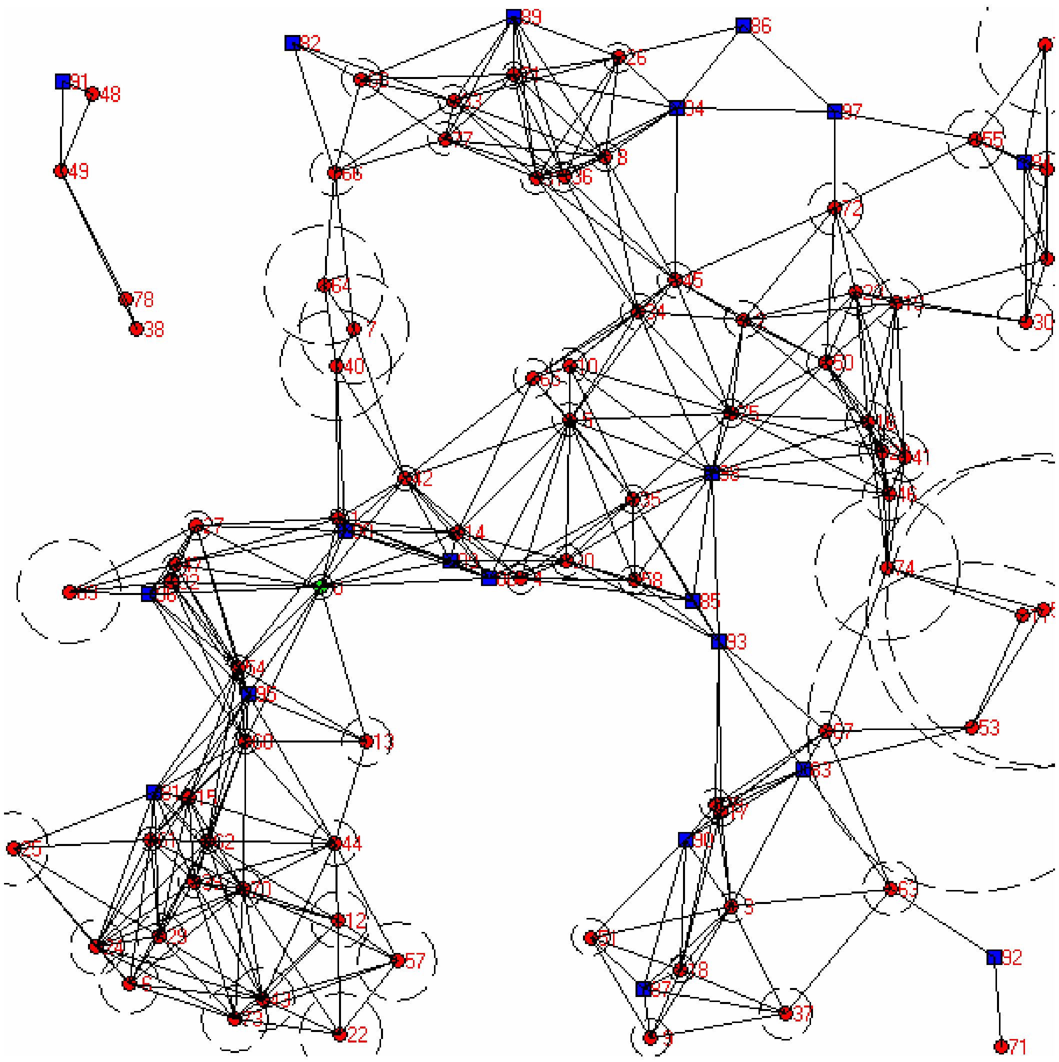


Figure 2.3. The CRB of the sample network is depicted as rings of the radius = σ_i. There are two exceptions: i) beacons, depicted as squares, have 0 CRB, and ii) some regular nodes have infinite CRB (such as node 38, 48, 49 and 78 at the top left corner) indicating that they cannot be localized.

known algorithms such as DV-Hop, Euclidean and Multi-lateralization can be obtained from [49]. The comparison here is done in the context of specific constraints of sensor networks, such as error tolerance and energy efficiency; results indicate that there is no single algorithm that performs “best” and that there is room for further improvement.

2.4.1 Connectivity-Based Algorithms

A number of localization methods rely on connectivity information only. These types of methods are also referred to as “range-free” methods in the literature. The Centroid method [11] estimates the location of an unknown node as the average of its neighbors’ locations. The APIT method [27] estimates the node location by isolating the area using various triangles formed by beacons. The DV-Hop method [58] counts the hop numbers to beacons and uses them as crude estimates for distances. Range-free methods require no additional hardware, but they generally only work well when networks are dense. Sparse networks by nature contain less connectivity information, and thus they are more difficult to localize accurately.

2.4.2 RSSI and ToA Range-Based Algorithms

Range-based methods include the ad hoc positioning system (APS) methods such as DV-Distance and Euclidean proposed in [58, 59]. In [70], ranging data are exchanged between the neighbors to refine the initial location guess. While those methods compute the absolute node locations, the GPS-Free method [12] calculates the relative node locations from the distance measurements. Compared to range-free methods, range-based methods give more accurate location estimates when ranging data is reliable. However, depending on the deployment environment, ranging techniques based on RSSI tend to be error-prone and strong filtering is required. The ranging error could ultimately destroy the localization accuracy if it is allowed to propagate through the network unbounded.

Different methods generally exploit the trade-off between the estimation accuracy and the estimation coverage. For instance, given the same network scenario, the Euclidean method is capable of generating more accurate location estimates of a smaller subset of nodes, whereas the DV-Hop method has better coverage but worse accuracy. Regardless of the tradeoff, a common characteristic shared by distance-based GAHLAs is that they require a relatively high network density in order to achieve better results. Based on the extensive simulation of DV-Distance, Euclidean and multilateration methods performed in [14], it can be concluded that those distance-based GAHLAs “require an average degree of 11-12 nodes within the ranging neighborhood in order to achieve 90% localization coverage with 5% accuracy [14].”

2.4.3 Angle-Based Algorithms

Even though the future of AoA sensing devices is still unclear, some works have been published on localization using angle information. Simulation studies in [14] also show that when AoA (angle of arrival) of the signals is used in addition to the distance measurement, the localization accuracy and coverage can be drastically improved. This should not come as a surprising conclusion, as nodes need to communicate with only one neighbor to perform localization if they can obtain both AoA and distance measurements. The work in [14] also presents three variations of a weighted mean square error algorithm that localizes the nodes, each of which is designed to work with one of the three measurement types: i) distance-only measure, ii) distance plus a more accurate AoA measure (up to $8°$ of precision) and iii) distance plus a less accurate AoA measure (up to $60°$ of precision). The less accurate AoA measurement method is sometimes referred to as *sectoring*. Simulations in [14] show that the localization accuracy and coverage can be greatly improved with such coarse sectoring measurement as well.

2.4.4 Hybrid Algorithms

A combination of the above techniques can be employed to form a hybrid method. For instance, a hybrid method is proposed in [1] that uses both APS and MDS (multi-dimensional scaling).

2.5 Probabilistic Localization

The localization algorithm proposed in this chapter is a probabilistic method employing importance sampling techniques (particle filters). Here, each unknown node's location is viewed as a probability distribution over the deployment area. The goal of the localization algorithm is to shape the distribution based on a sequence of measurement until the distribution becomes focused and collapses onto a small area. The probabilistic method and particle filters have been used in visual target tracking [34] and computer vision location systems [20, 80] in the context of robotics. The particle filter method is also used in [54] to obtain the mobile node location based on received signal strengths from several known-location base stations in wireless cellular networks. The probability grid system in [76] is a centralized probabilistic localization algorithm that updates the distribution based on a grid system.

Perhaps the closest to our work are the Monte Carlo localization (MCL) method [30] and the in-door location tracking algorithm [84]. In [30], a similar probabilistic and particle filter approach was taken to localize mobile nodes in ad hoc networks. As nodes move in and out of range of each other, the probability distribution is updated via particle filtering based on the connectivity information. However, our work differs with theirs in that we consider sensory based (RSSI or AoA) localization while the localization in [30] is range-free. Given the different measurement models, the filtering process is also completely different. Furthermore, our method works for both stationary and mobile networks, while the model in [30] is designed for mobile networks only. Finally, localization in [30] only occurs when a regular node hears from a beacon directly, while our algorithm allows collaborative localization among regular nodes.

The indoor location tracking problem in [84] deals with a known environment thus different obstacles can be represented in a floor-plan and thus a signal strength (RSSI) map can be obtained via measurements and calculations ahead of time. The location tracking problem then becomes a decision-making problem, where a solution may use a measurement model that compares the current RSSI with the signal strength map to find the location with the highest probability of matching the current RSSI reading. While roots are similar, the solution described in this work is designed for out-door environments and infrastructure-less networks where major continuous obstacles (such as walls) are assumed to be minimum, and fairly reliable distance estimates can be obtained from RSSI readings and the signal propagation model. The probability distributions of location estimates are updated solely from the distance and location estimates from neighbors.

To our best knowledge, our work is the first that incorporates multiple sensory data for localization using the same algorithmic framework. The need for such collaboration is established in [14], but the authors only consider the limited case when both ranging and angle readings are available at the same node. Our work is more general in that we consider the cases when the nodes can either have ranging, angle, both, or just connectivity-only. The same particle filter framework allows the nodes with different sensory capacities to collaborate during localization.

2.6 Particle Filter Framework for Location Estimation

In this chapter, we propose a localization method based on Bayesian filters using Monte Carlo sampling (also known as particle filters) introduced in [26]. Our method can be considered as a probabilistic approach in which the estimated location of each node is regarded as a probability distribution captured by samples, thus the term particles. The distribution of particles (the probability distribution of a node's location over the area) is continuously updated as the node receives location estimates from its neighbors along with certain types of sensory readings such as RSSI and AoA. (More precisely particles represent samples drawn from a continuous probability density function, thus using the term "probability distribution" is justified.) Essentially, nodes estimate their own locations by exchanging the location distributions directly with their neighbors.

Figure 2.4 demonstrates how our method solves the AHLP in a simple scenario (where RSSI based distance measurements are used). Here, nodes 2, 3, and 4 are beacon nodes, while nodes 0 and 1 are regular nodes. Of the beacons, node 0 can receive signals only from nodes 1 and 4, and node 1 can receive signals from only nodes 0, 2, and 3. From the signal strength readings, the regular nodes estimate their distances to their neighbors. The probability distribution of the estimated location is represented by the particles (dots) in the graph. In sub-figure (a), where node 1 is removed, node 0 can only receive signals from node 4; thus as the particle distribution indicates, the probability distribution where node 0 is most likely located concentrates on a circle around node 4. In sub-figure (b), where node 0 is removed, node 1 can receive signals from node 2 and 3; thus the most likely locations for node 1 center around two areas where "transmission circles" around node 2 and 3 intersect. Intuitively, in order to localize itself, a regular node needs to receive location information from a minimum of three beacons either directly or indirectly. In both case (a) and case (b), the exact location of node 0 and 1 cannot be deduced because they do not receive location information from all three beacons. In (c) and (d), where all nodes are available, node 0 and 1 are able to communicate to each other and exchange their particle distributions. Thus, their probability densities will represent their actual locations much closer even though neither node receives location information from all three beacons directly.

2.6.1 Classic Monte Carlo Sampling-Based Bayesian Filtering

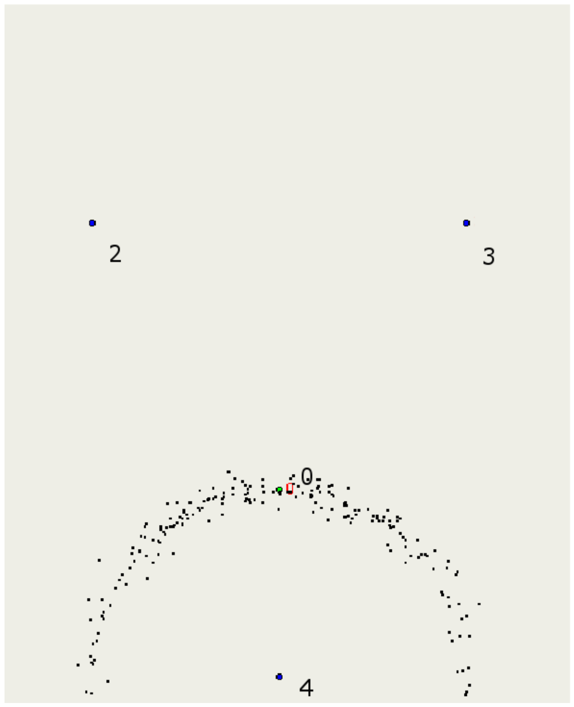
This section describes the theoretical background behind Bayesian filtering and how it can be applied to location estimation using RSSI. Let us envision a grid system superimposed over the entire tracking area, and let the state s_t be the location of the node to be tracked in the grid system at time t . Our goal is to estimate the posterior probability distribution, $p(s_t|d_1, \ldots, d_t)$, of potential states, s_t, using the RSSI measurements, $d_1, \ldots, d_t$. The calculation of the distribution is performed recursively using a Bayes filter:

$$p(s_t|d_1, \ldots, d_t) = \frac{p(d_t|s_t) \cdot p(s_t|d_1, \ldots, d_{t-1})}{p(d_t|d_1, \ldots, d_{t-1})}$$

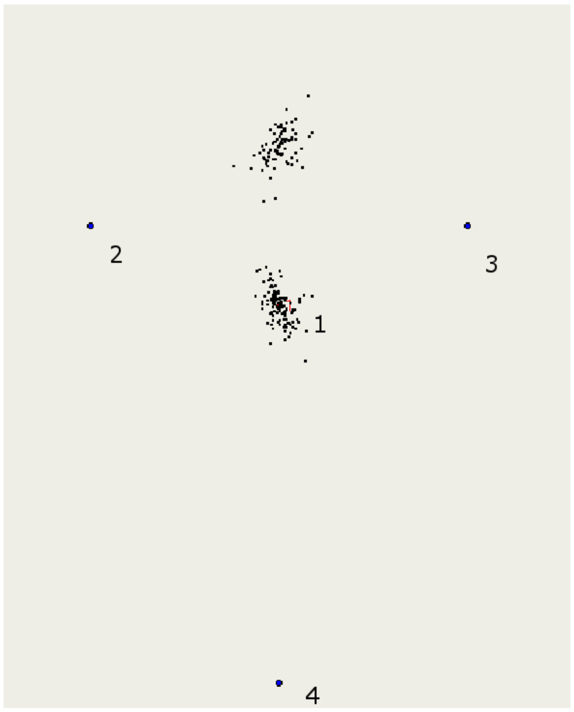
Assuming that the Markov assumption holds, i.e., $p(s_t|s_{t-1}, \ldots, s_0, d_{t-1}, \ldots, d_1) = p(s_t|s_{t-1})$, the above equation can be transformed into the recursive form:

$$p(s_t|d_1, \ldots, d_t) = \frac{p(d_t|s_t) \cdot \int p(s_t|s_{t-1}) \cdot p(s_{t-1}|d_1, \ldots, d_{t-1}) ds_{t-1}}{p(d_t|d_1, \ldots, d_{t-1})},$$

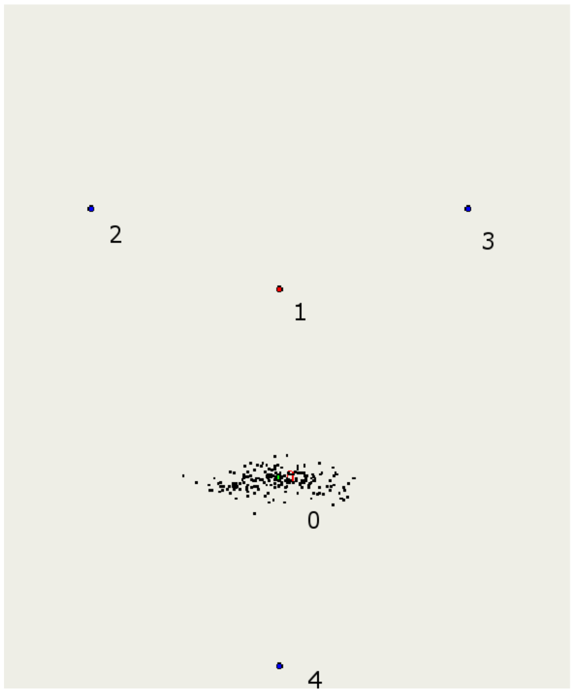
where $p(d_t|d_1, \ldots, d_{t-1})$ is a normalization constant. In the case of the localization of a mobile node from RSSI measurements, the Markov assumption requires that the state contains all available information that could assist in predicting the next state and thus, an estimate of the non-random motion parameters of the nodes is required as part of the state description. Starting with an initial, prior probability distribution, $p(s_0)$, a system model, $p(s_t|s_{t-1})$, representing the motion of the mobile node (the mobility model), and the measurement model, $p(d|s)$, it is then possible to drive new estimates of the probability distribution over time, integrating one new measurement at a time. Each recursive update of the filter can be broken into two stages:



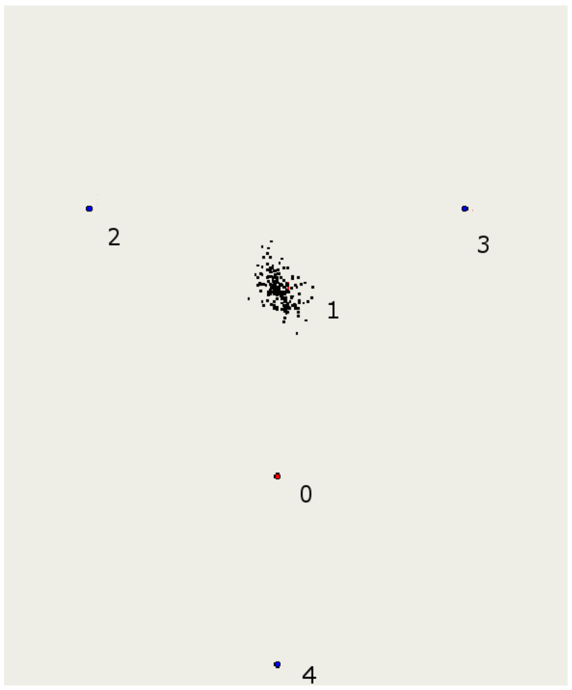
(a)



(b)



(c)



(d)

Figure 2.4. Location distribution in simple scenarios with distance measures. Four cases include (a) particle distribution of node 0 when node 1 is *not* present, (b) particle distribution of node 1 when node 0 is *not* present, (c) particle distribution of node 0 when node 1 *is* present, and (d) particle distribution of node 1 when node 0 *is* present.

Prediction: Use the system model to predict the state distribution based on previous readings

$$p(s_t|d_1,\ldots,d_{t-1}) = \int p(s_t|s_{t-1}) \cdot p(s_{t-1}|d_1,\ldots,d_{t-1})ds_{t-1}$$

Update: Use the measurement model to update the estimate

$$p(s_t|d_1,\ldots,d_t) = \frac{p(d_t|s_t)}{p(d_t|d_1,\ldots,d_{t-1})}p(s_t|d_1,\ldots,d_{t-1})$$

To address the complexity of the integration step and the problem of representing and updating a probability function defined on a continuous state space (which therefore has an infinite number of states), the approach presented here uses a sequential Monte Carlo filter to perform Bayesian filtering on a sample representation. The distribution is represented by a set of weighted random samples and all filtering steps are performed using Monte Carlo sampling operations. Since we have no prior knowledge of the state we are in, the initial sample distribution, $p_N(s_0)$, is represented by a set of uniformly distributed samples with equal weights, $\{(s_0^{(i)}, w_0^{(i)})|i \in [1, N], w_0^{(i)} = 1/N\}$ and the filtering steps are performed as follows:

Prediction: For each sample, $(s_{t-1}^{(i)}, w_{t-1}^{(i)})$, in the sample set, randomly generate a replacement sample $(\tilde{s}_t^{(i)}, w_t^{(i)})$ according to the system (mobility) model $p(s_t|s_{t-1})$. This results in a new set of samples corresponding to $p(s_t|d_1,\ldots,d_t)$:

$$\{(\tilde{s}_t^{(i)}, w_t^{(i)})|i \in [1, N], w_t^{(i)} = 1/N\}$$

Update: For each sample, $(\tilde{s}_t^{(i)}, w_t^{(i)})$, set the importance weight to the measurement probability of the actual measurement, $\tilde{w}_t^{(i)} = p(d_t|\tilde{s}_t^{(i)})$. Normalize the weights such that $\sum_i \eta \cdot \tilde{w}_t^{(i)} = 1.0$, and draw N random samples for the sample set $\{(\tilde{s}_t^{(i)}, \eta\cdot \tilde{w}_t^{(i)})|i \in [1, N]\}$

according to the normalized weight distribution. Set the weights of the new samples to $1/N$, resulting in a new set of samples $\{(s_t^{(i)}, w_t^{(i)}) | i \in [1, N], w_t^{(i)} = 1/N\}$ corresponding to the posterior distribution $p(s_t | d_1, \ldots, d_t)$.

2.6.2 Modified Particle Filtering for Location Estimations

To apply the filter to the localization problem a system model and a measurement model must be provided. We select a simplistic model that assumes at any point in time that the node moves with a random velocity drawn from a Normal distribution with a mean of $0m/s$ and a fixed standard deviation σ. No information about the environment is included in this model, and as a consequence, the filter permits the estimates to move along arbitrary paths. Thus, our system model is simply $p(s_t | s_{t-1}) = N(0, \sigma)$, where N is a Normal distribution. Note that while such a system model should work well in stationary networks and networks where user mobility is extremely uncertain; there could be better models for mobile networks. In reality, mobile nodes follow a certain kind of movement profile instead of random motion. The system model should closely resemble the current movement profile of the node. However, since it is difficult to obtain a reliable movement profile when the location is unknown, the assumption of random movement is probably the best we can make at this stage. (Note that a more accurate mobility model would greatly improve the estimate of the filter.)

When a reading, m, is obtained from the RSSI or AoA sensor, the particle filter undergoes a **correction step**, in which the measurement is used to correct the output of the system model. In particular, the correction step modifies the particle distribution X so that it becomes more consistent with the current measurement. In our case the correction is calculated based on both the measurement reading as well as the location distribution of the neighbors. In other words, when node u receives a measurement reading m from node v, the correction step updates the location distribution of X_u based on m and X_v. The correction step is challenging due to the fact that both m and X_v are imprecise. The measurement reading m could be noisy due to environmental and sensory characteristics. The location distribution X_v could also be imprecise unless v is a beacon. Thus, the correction step needs to modify the particle distribution X_u so that it becomes more consistent with m while taking into account the inherent impreciseness of X_u and X_v.

After each correction step, the estimated location of the node is obtained by estimating the "mode" of the particle distribution. In the particle distribution, we define the mode by comparing the distances between samples. The particle that is the closest to all other particles (i.e., the mode of the density) is selected to be the most likely estimated location at the current time. For stationary networks, we can monitor the expected location over multiple updates and set the stop condition when the change of the location becomes sufficiently small for the application using the location data. Alternatively, we can monitor the filter's uncertainty (in terms of variance in particles' locations) and set the stop condition when it falls below a certain threshold. For mobile networks, the particle filter can run continuously to keep track of locations as nodes move.

Algorithm 1 shows the pseudo-code for the particle filter update algorithm. Note that the method to find the new location x'_u depends on the sensory capacity of the receivers. We can now consider three different types of behavior for the correction step depending on what sensors are available: RSSI, AoA, or none (connectivity-only). We then analyze both the RSSI and the AoA methods in the context of the AHLP with and without measurement noise. Finally, we explain the *Decompress* step, which is necessary to reduce the communication overhead.

Algorithm 1 Particle Filter Update

```
$X_u$ ←Uniform particle distribution over the deployment area
**repeat**
  **for all** neighbors $v$ in the neighbor set **do**
    Receive($id_v$, $X_v$, $m$)
    Decompress $X_v$
    **for all** $x_u \in X_u$ **do**
      Randomly select a $x_v \in X_v$
      Find location $x'_u$ based on $x_v$, $m$, $stdev(X_u)$ and $stdev(X_v)$
      Update $x_u$ with $N(x'_u, ((stdev(X_u) + stdev(X_v))/2)^2)$
    **end for**
  **end for**
**until** $var(X_u)$ is below a threshold
```

2.6.3 RSSI Sensor Availability

When an RSSI sensor is available, we obtain a distance estimate from the inverse of the signal propagation model $P_{i,j}$. A Gaussian noise can be added to the model, but we disregard it when calculating the inverse and let it be reduced by the particle filter. (This does not mean that we are ignoring noise in our evaluation as noise will indeed be added to the RSSI measurements in the simulation model).

As shown in Figure 2.5(a), we let node v be the sender and node u be the receiver. For each particle x_u in the current location distribution X_u, we randomly select a particle x_v in the sender's location distribution X_v and calculate their distance $D^{(x_u,x_v)}$. We then measure the difference between $D^{(x_u,x_v)}$ and $D^{(RSSI)}$, and select a new location for re-sampling based on the difference as well as the variances of the particle distribution X_v and X_u. For instance, before the update step x_u and x_v are located at point A and B, respectively. Thus, $D^{(x_u,x_v)} = |AB|$. Let A' be the location of x_u based on the RSSI reading on the same line, i.e., $D^{(RSSI)} = |A'B|$. Intuitively, if the location estimate given by the distribution X_v is accurate and the actual location for node v is indeed at x_v, then the new location for particle x_u should be at point A'. Conversely, if the location

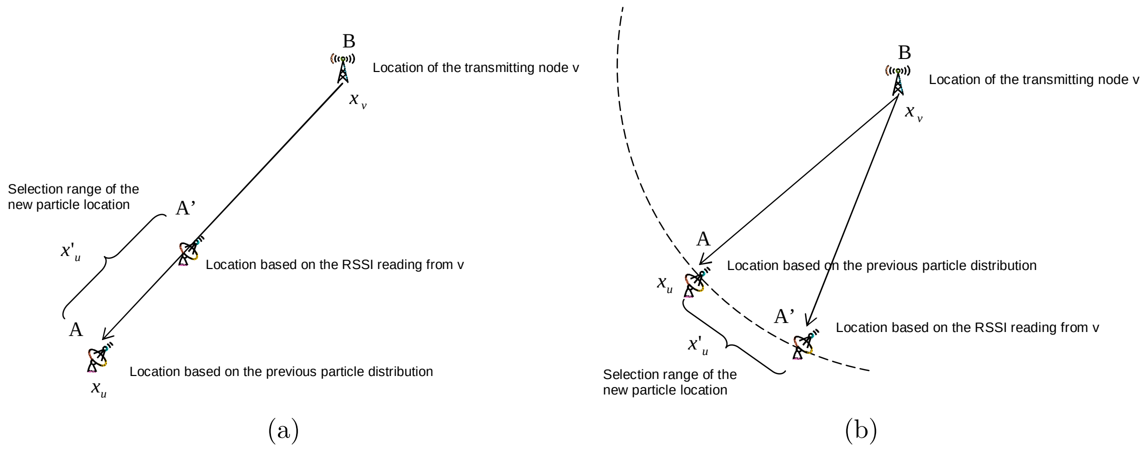

B
Location of the transmitting node v
x_v
Selection range of the new particle location
A'
x'_u
Location based on the RSSI reading from v
A
Location based on the previous particle distribution
x_u
(a)
B
Location of the transmitting node v
x_v
A
Location based on the previous particle distribution
x_u
A'
Location based on the RSSI reading from v
x'_u
Selection range of the new particle location
(b)

Figure 2.5. Particle filter update of (a) RSSI and (b) AoA.

estimate of the distribution X_u is accurate, the new location for x_u should stay at A. Therefore, we select the new location based on the perceived accuracy, i.e., the variances, of the distributions of X_u and X_v. Let the standard deviation of a distribution X be $stdev(X)$. We select the new location of x_u, x'_u, along the line $|AA'|$ such that

$$\frac{|Ax'_u|}{|x'_uA'|} = \frac{stdev(X_u)}{stdev(X_v)}$$

A new particle is then randomly re-sampled using a Normal distribution centered at x'_u with the standard deviation being the average of those of X_u and X_v. We consider the standard deviations of both X_u and X_v during re-sampling because the spread of both distributions affects the spread of the updated distribution X'_u.

2.6.4 AoA Sensor Availability

When an AoA sensor is available, we compare the standard deviations of the sender's and receiver's distribution as before, but in this case, we modify the receiver's particles based on the arrival angle. Again, let v be the sender and u be the receiver. Shown in Figure 2.5(b), for each particle x_u in the current location distribution X_u and particle x_v in the sender's location distribution X_v, the angle of arrival between x_u and x_v can be calculated along with the distance $D^{(x_u,x_v)} = |AB|$. We draw a line through B according to the current angle of arrival reading and select a point A' by maintaining the distance so that $|A'B| = |AB|$. The new particle location x'_u is then located on the arc between point A and A' with the radius being $|AB|$. If the location estimate from v is more accurate, then x'_u should be closer to point A' along the arc. Conversely, if the location estimate from u is more accurate, then x_u should be closer to A. Thus, the new location of x_u, x'_u, is the following:

$$\frac{\angle ABx'_u}{\angle x'_uBA'} = \frac{stdev(X_u)}{stdev(X_v)}$$

Similarly to the way done at the RSSI sensors, a new particle is re-sampled from the selected location based on a Normal distribution with the standard deviation equaling the average of $stdev(X_u)$ and $stdev(X_v)$. When both RSSI and AoA sensors are available, the aforementioned update steps can be effectively combined. In such case, the particles are first updated based on the RSSI reading followed by an update based on the AoA readings. Thus, both sensor readings are applied to the location estimation.

2.6.5 Connectivity-Only Nodes

We also consider a third type of nodes: connectivity-only nodes, where neither RSSI nor AoA sensors is available. Those nodes have to rely on pure connectivity information to estimate their locations. One approach is to use a variation of the Centroid method [11], where nodes estimate their locations by simply averaging all their neighbors' locations. Intuitively, such a method should work reasonably well when the network is well connected (so that there are more location data from the neighbors to work with) and the beacon ratio is high (so that the location data from the neighbors are more accurate). We adopt the same idea here but adapt it to the context of particle filtering.

We consider the same scenario as before, where node u receives a location update message from a neighbor v, Let x_u be a random particle within the location distribution X_u before the update. Let x_v be a random particle at neighbor's distribution X_v. Since there is no sensor reading on ranging or angle, we can't tell exactly the distance or the direction between u and v; but we know they are sufficiently close since the nodes are in each other's transmission range. In this case, we update particle x'_u's location (to between x_u and x_v) as:

$$\frac{|x_u x'_u|}{|x'_u x_v|} = \frac{stdev(X_u)}{stdev(X_v)} \cdot c$$

Again, when X_u is perceived as more accurate (i.e., when $stdev(X_u)$ is smaller), x'_u becomes closer to the previous location x_u; otherwise, x'_u moves closer to x_v. We multiply the weighting factor with a constant $0 < c < 1$, so that the new location is closer the previous location. Thus, we are able to retain the location information through a sequence of location updates from different neighbors.

2.6.6 Analysis

In this section, we attempt to infer, by simple geometric analysis, whether RSSI or AoA sensors would be preferred for localization. We also consider the presence of noise. In particular, our goal is to derive a relationship between the noise parameters of each sensor type, so that their impact can be compared during simulation.

Let us first consider the perfect scenario where no measurement noise interferes with the sensor reading. Recall, that to precisely locate a node, at least three RSSI readings from different beacons are required, while only two AoA readings are needed. When both measurement types are available, only one RSSI reading and one AoA reading from the

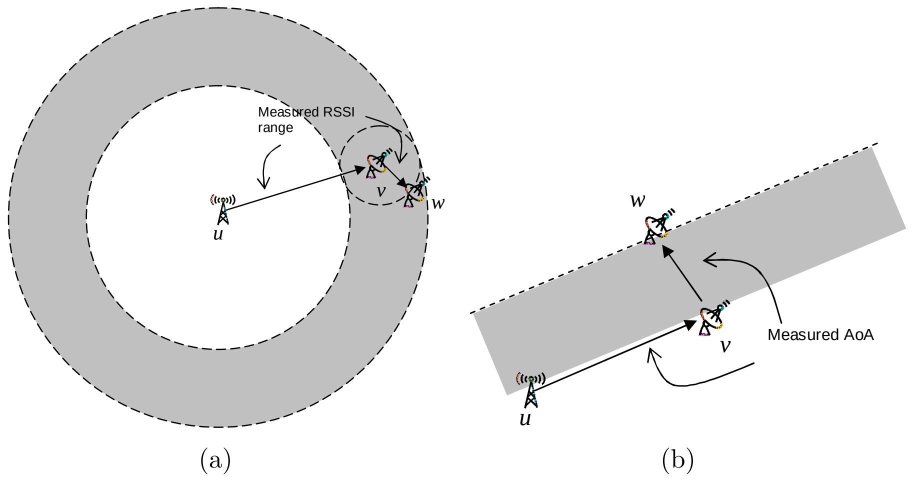


Figure 2.6. Location range for 2 hops of (a) RSSI and (b) AoA. The shaded area indicates the range of all possible locations.

same beacon are required to locate the node. In such cases, AoA readings should provide better coverage (i.e., locating more nodes) than RSSI readings.

In networks where connectivity is low and/or beacon ratio is low, it is likely that nodes have to estimate their location based on other nodes' estimates. Let us consider how well location data is propagated in all sensor type cases. Let node u be a beacon and v and w be regular nodes. Let u connect to v and v connect to w, but let there be no connection between u and w. Therefore, w has to infer its possible location based on the indirect information from v. In the case of RSSI sensors, the possible range of w is a hollow disk with inner radius $(r - s)$ and outer radius $(r + s)$, where r is the distance between u and v, and s is the distance between v and w (Figure. 2.6(a)). In the case of AoA sensors, the possible location of the intermediate node v is a beam with the origin at u's location. To derive the possible locations of w, we have to draw a beam with the origin at every possible point of the previous beam. Thus, the possible range of w is an area bounded only by the beams and the network boundaries (Figure. 2.6(b)).

From the above discussion we can infer that RSSI sensors are better suited for localization when location data need to propagate through multiple hops. AoA sensors, however, are better when beacons are only a single hop away. Thus, when networks are sparse and/or beacon ratio is low RSSI sensors are better suited. When networks are dense and/or beacon ratio is high, AoA sensors would be a better choice.

When noise is added to the measurement readings, the precise location of nodes cannot be obtained even when the minimal geometric requirements are met. Instead, location needs to be estimated and the accuracy of this estimate is affected by the noise (and thus the noise model in simulation and mathematical evaluations). Furthermore, unlike the ideal scenario where the minimal geometric requirement is sufficient to localize the node, the accuracy of the estimated location improves with readings from additional neighbors. As in other related works [58, 60], we qualify the noise in terms of a *noise ratio*, which presents noise as a percentage of the unbiased measure. In the case of RSSI, a noise ratio of x over the unbiased distance D would depend on the actual reading: $D' = D \cdot Uniform(1-x, 1+x)$. In other words, the distance measured at the receiver contains an error in the range $(-xD, xD)$, uniformly distributed. The error in the distance measurement accounts for the power loss due to factors such as multi-path fading and far field scattering. Modeling noise in this way with a uniform distribution is a drastic simplification; however we have chosen this simplified model so that our result can be effectively compared to results of other GAHLAs (as this method has been excessively used in the AHLP literature). The AoA measurement noise is modeled in a similar way using a uniform distribution; with a noise ratio of y, the measured AoA at the receiver is then $A' = A \cdot Uniform(1-y, 1+y)$.

To compare the estimates based on RSSI and AoA under noisy environment conditions, we need to establish a relationship between the noise ratios x and y. In other words, with a given x for RSSI, a corresponding y for AoA needs to be selected to gen-

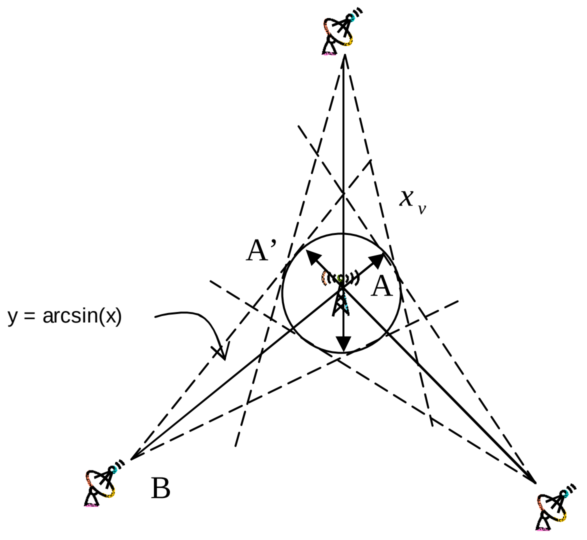


Figure 2.7. Finding similar error range between RSSI and AoA.

erate a more or less similar noisy environment. Figure 2.7 shows how such a relationship could be established. Here, we consider a scenario in which a node is localized from three beacons equal distances away. Let the actual location of the node be A and the location of one of the beacons be B; thus, the actual distance between them is $D = |AB|$. Let us first consider the RSSI sensor, where x is the noise ratio. The range of all possible estimated locations should be within a circle of radius $D \cdot x$ centered at the actual location A.

Now consider the case of the AoA sensor. To replicate a similar range from the same three neighbors by an AoA sensor, the closest range that can be possibly formed is to project the AoA error, y, along the line AB so that the entire circle is covered (by selecting the beam angle appropriately). Doing so with all three beacons, the actual location error range forms an outer hexagon that encloses the circle. Here, $sin(y) = |AA'|/|AB|$, where $|AA'|/|AB| = x$. Thus, a relationship can be established between the RSSI noise parameter x and the AoA noise parameter y in that $y = arcsin(x)$. Note that this relationship does not generate an identical error range between the two sensor types; in fact, the error range from AoA will always be greater. However, since it is not feasible to find the noise parameter that generates an identical error range that would work for all cases, our simplification can be justified. After all, our intention is to provide a relatively similar error range so we can reasonably compare the results from the two sensor types.

2.6.7 Compressing and Decompressing Particle Filter Distribution

The previous sections make the assumption that a complete set of particles is received from each of the neighbors. Since the complete distribution consists of a large number of particles with their location data, doing so is obviously not very practical. Therefore, we propose a simple yet effective compression mechanism that allows a representative for the particle distribution to be transmitted in a compact form.

Given a particle distribution X, we locate the most likely value, $\hat{x}$, as the particle in the distribution that has the minimum overall distance to other particles, i.e., $\hat{x} = \arg\min_{x \in X} \left(\sum_{y \in X} |x - y| \right)$. In other words, $\hat{x}$ is the most representative particle of the entire distribution. From $\hat{x}$, we count the number of particles n within a predefined range r. We then calculate the variance, σ^2 within those n particles. Thus, we obtain a quadruple $(\hat{x}, r, n, \sigma^2)$. From there, we remove the n particles in the previous quadruple from the distribution and repeat the process of finding the expected value, a larger range (explained later) and the variance. By continuing the same process until all particles have been covered, we obtain a sequences of quadruples that approximates the original particle distribution. When the quadruples are received by the receivers, a decompressing step is executed to reproduce the distribution by randomly generating particles based on the expected value, range, particle number and variance for each quadruple. Using this method the particle distribution can keep all of its “modes”, even when the distribution shows several likely location areas for the node.

Algorithm 2 Range Increase for Particle Compression

```
$Q \leftarrow$ number of quadruples desired
$R \leftarrow$ max range that covers the entire area
$minQuota \leftarrow |X|/Q$
$rIncrement \leftarrow R^{1/3}/Q$
$xCount \leftarrow 0$
$r \leftarrow 0$
$curRange \leftarrow 0$
$q' \leftarrow 1$
**for** $q = 1$ to $Q$ **do**
  $maxRange \leftarrow q \cdot rIncrement^{3/2}$
  **while** $curRange < maxRange$ AND the number of particles in $curRange + xCount < minQuota \cdot q$ **do**
    $curRange \leftarrow q' \cdot rIncrement^{3/2}$
    $q' \leftarrow q' + 1$
  **end while**
  $r_q \leftarrow curRange$
  $xCount \leftarrow xCount +$ number of particles in $curRange$
**end for**
```

For each broadcast, a fixed number of aforementioned quadruples are transmitted. Figure 2 shows the algorithm used to progressively increase the range r for each quadruple. The algorithm starts with an initial range of $R^{1/2}/Q$ and a minimum quota of particle size $|X|/Q$ for each quadruple. As each quadruple is defined, a running sum, $xCount$, keeps track of the total number of particles covered thus far. At each step, the range is incremented exponentially at each quadruple by $r := r^{3/2}$, unless the running sum already exceeds the minimum quota. The algorithm guarantees that all particles are covered by the predefined number of quadruples, and the overall trends of the original distribution are maintained. Meanwhile, by using a quota limit with the exponential range increment, denser particle areas are preserved in more detail. Our experiments in section 2.7.6 will show that the compression method reduces the amount of data exchange by nearly 90 percent without a significant increase in the location estimates' error. Figure 2.8 shows a typical compressed distribution, where circles represent ranges.

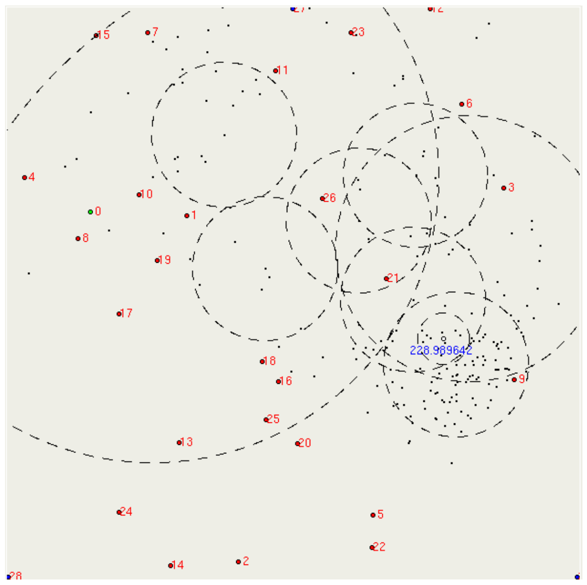


Figure 2.8. Compressing the particle filter distribution.

2.7 Simulation Results

We have conducted a number of simulation experiments to validate the effectiveness of our solution. During the following discussions we will concentrate on the performance of the filter based on the AoA reading (pure AoA as well as mixed sensors of both AoA and RSSI co-existing in the same network); detailed results on the performance of the pure RSSI based filter can be found in [33]. While our particle filter framework has no such restriction, we assume a network in which all nodes have an identical transmission power. Thus, we can effectively control the network connectivity by varying the transmission range. A certain percentage of nodes (simulation factor) are designated as beacons that know their coordinates. When a node is located within the transmission range of another node, we assume that it is capable of receiving a signal from the sender. The received signal strength (RSSI) depends on the distance to the sender (based on the employed signal propagation model) and a noise model. The measured AoA reading is affected by a noise model and its parameters as well.

The signal propagation model we use is the general free space propagation model of $P = c \cdot d^{-2}$, where the power of the received signal P is inversely proportional to the second power of the distance d, and c is constant that includes transmission power and frequency among others. When the received signal strength P is below a threshold P_{min}, it is considered too weak to be captured by the receiver thus the link breaks. Note, that the selection of c and P_{min} does not affect the overall simulation results, as long as the same values are used in the observation model of the filters. For the particle filter itself, we use a total number of 200 particles at each node. We randomly place 100 nodes into an isotropic (square) network. Noise is added to both RSSI and AoA readings using the models outlined in Section 2.6.6 .

Regardless of the sensor types available, nodes localize themselves by running our particle filter framework using the location exchanges among neighbors. The location information is exchanged between the neighbors at half-second time intervals. In other words, on average a node is able to obtain the location information from all of its neighbors every half second. The 0.5 second time interval is further randomized by a truncated Normal distribution $N(0.5, 0.5)$ to simulate the unpredictability of message arrival time in real networks. We simulate each type of scenario 50 times; the results are averaged, and a 95% confidence interval is calculated and displayed (using vertical bars in our figures). Estimation errors in our graphs are given as a ratio to the transmission range, i.e., an average error of 1 means that in average the location estimate's error is the same as the nominal transmission range. We start by showing simulation results on stationary multihop networks, and then move to mobile ad hoc networks.

2.7.1 Localization Error Characteristics

To study the localization error characteristics, we plot the error obtained using four different localization methods: particle filter using RSSI ranging, DV-Hop, DV-

Distance and Euclidean for the sample network of Figure 2.1. The result is shown in Figure 2.9. Note that while the particle filter method gives an estimate for *all* nodes, the DV-Hop, DV-Distance and Euclidean methods only give estimate for a subset of nodes that they can localize. The figures do not show the error for those unlocalized nodes. From the localized nodes in the figures, we observe that the error of the DV-based methods (Figure 2.9(a) and 2.9(b)) tend to spread out among all nodes. This is because the DV-based methods attempt to localize from the global information (the hop count for DV-Hop and the hop distance for DV-Distance). Since the global information only provides an average case, the localization is rather crude and thus results into similar error characteristics regardless of beacon proximity. The particle filter (Figure 2.9(d) and the Euclidean (Figure 2.9(c)) methods are based on the local information and thus are more fine-grained. They produce more accurate localization when the nodes are closer to beacons. Higher error occurs at the area where the beacon placement is not ideal such as the lower left corner of the sample network.

2.7.2 Connectivity

The primary performance metric for any location algorithm is the estimation error, which indicates how close the estimated location is to the actual location. We compare the estimation error of our particle filter algorithm, including several combinations of RSSI, AoA, and connectivity-only nodes, against results of the existing methods. Here, we use a noise parameter of $x = 20\%$, and a beacon ratio of 10%. One advantage of our method is that it produces the location estimate along with a variance indicating its quality. Thus, by varying the variance threshold, we are able to control the effective estimation coverage; the lower the coverage, the better the estimation accuracy. The estimation error in our figures is therefore plotted against the desired coverage.

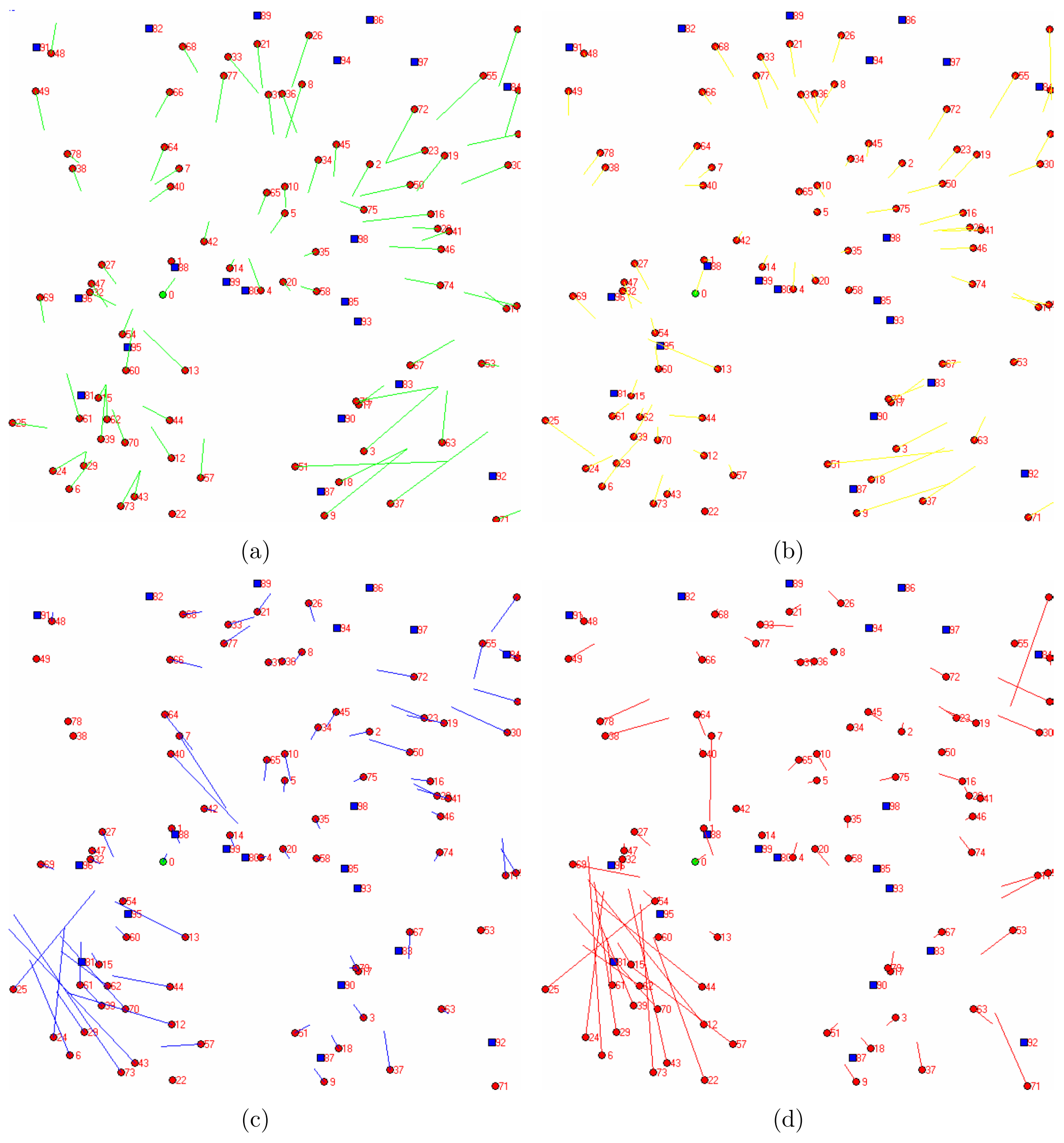


Figure 2.9. Localization error characteristics. The lines show the difference between the estimated location and the real location of (a) DV-Hop, (b) DV-Dist, (c) Euclidean, and (d) Particle Filter.

Figure 2.10 shows the estimation error against coverage using four different variations of the particle filter algorithm while varying network connectivity. For each variation, we modify the sensor types within the network as follows: (a) 100% of the nodes have RSSI only capacity, (b) 100% of the nodes have AoA only capacity, (c) 50% of the nodes have RSSI and 50% of the nodes have AoA capacity, and (d) 100% of the nodes have both RSSI and AoA capacity. As shown in Figure 2.10, network connectivity plays an important role determining the localization accuracy. In particular, nodes in sparser networks tend to be more difficult to be localized (i.e., resulting in higher estimation error) primarily due to two reasons. First, it is less likely for the nodes in sparser networks to be one-hop away from the beacons. Thus, more nodes would have to rely on location data from beacons several hops away. Furthermore, as the network becomes sparser, it is likely to become disconnected. The location data from the beacons might not be able to propagate to all nodes. In the extreme cases, which are much more common in sparser networks, there might be nodes that do not meet the minimum geometric requirement of localization; therefore, those nodes cannot be localized.

In Figure 2.10(a), where the localization relies solely on the RSSI sensor data, the estimation error from the particle filter method is plotted along with the results from DV-Hop, DV-Distance and Euclidean methods that produce location estimates with a fixed coverage. Thus, their estimation errors are shown as single data points in the figure. Comparing the single data points of the above methods, the comparable plot (when degree = 4.61) from the particle filter method closely follows the single data points from the DV and Euclidean methods. In general, algorithms such as Euclidean would trade-off coverage for accuracy, while algorithms such as DV-Hop would trade-off accuracy for coverage. Instead of relying on different GAHLAs for different trade-off objectives, our particle filter algorithm is capable of exploiting such trade-off by selecting the coverage based on the filter variance. Such characteristics make the particle filter

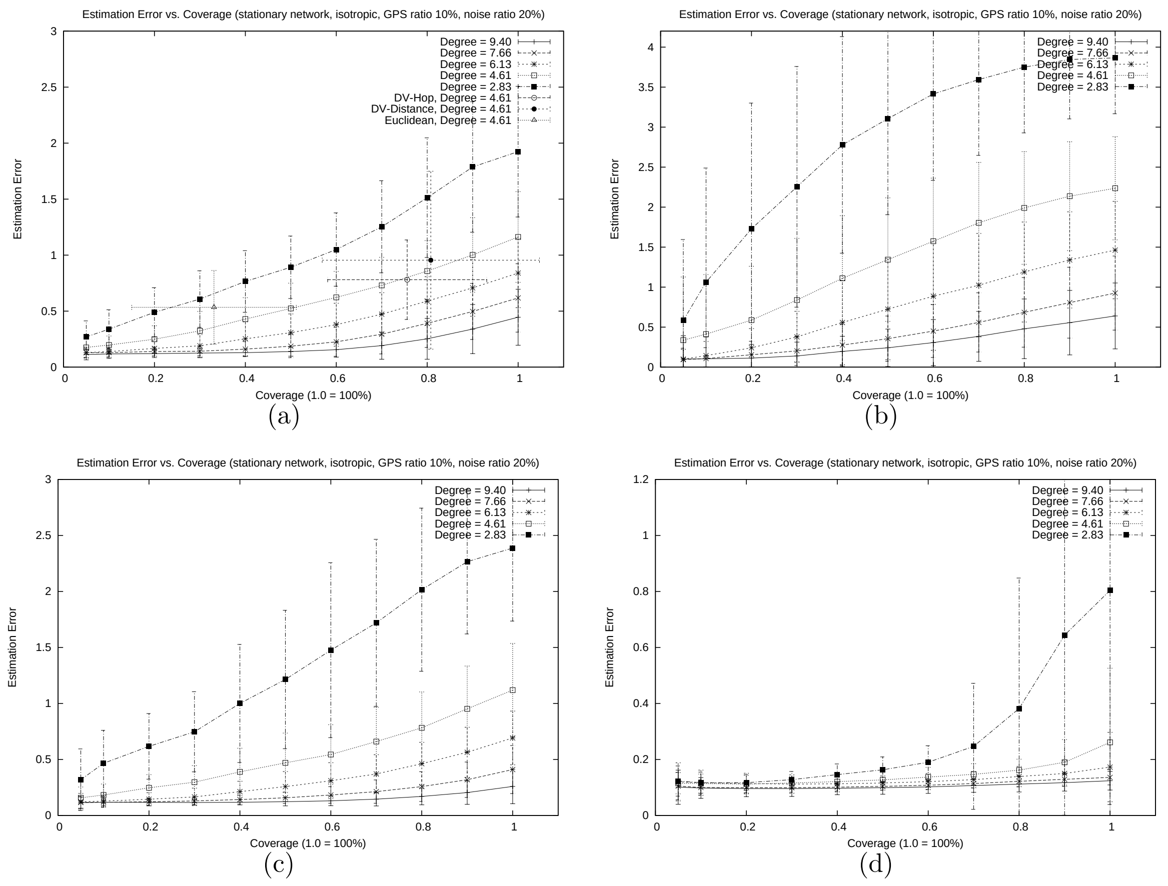


Figure 2.10. Effect of network connectivity on the estimation error. Four cases include (a) RSSI 100%, AoA 0%, (b) RSSI 0%, AoA 100%, (c) RSSI 50%, AoA 50%, and (d) RSSI 100%, AoA 100%.

solution versatile in adapting to the different localization requirements. For instance, for applications that prefer accuracy over coverage, only those estimates with smaller variances can be considered valid estimates. Conversely, for applications that prefer better coverage, estimates with larger variances can also be considered as valid.

Figure 2.10(b) shows result where only AoA sensors are available. It can be observed that the estimation error becomes high when the network is sparse. As explained in the aforementioned analysis, this can be attributed to the fact that AoA sensors are less capable of propagating location information through multi-hops. However, when both RSSI and AoA sensors are implemented (as shown in Figure 2.10(c) and 2.10(d)), the estimation error can be drastically reduced. In particular, when the average degree is greater than 4.61, the 50% RSSI and 50% AoA sensor combination outperforms the cases of 100% RSSI or AoA sensors (Figure 2.10(c)). This indicates that mixing different sensor data can be very beneficial in ad hoc localization. When all nodes have both RSSI and AoA capacity, the performance is even better (Figure 2.10(d)). This result confirms the results of [14], where the authors claim that localization results can be drastically improved when both RSSI and AoA capability are available at all nodes.

2.7.3 Beacon Ratio

The effect of the beacon ratio on the estimation error is shown in Figure 2.11. Again, we consider four different types of sensor configuration. As expected, a higher beacon ratio lowers the overall estimation error in all cases. Furthermore, when half of the nodes have RSSI sensor capacity and the other half have AoA capacity (Figure 2.11(c)), the result is better than using just one sensor type (Figure 2.11(a) and 2.11(b)). This further proves the advantage of using mixed type of sensors in ad hoc localization.

2.7.4 Noise

The effect of the noise ratio x on the location estimates is shown in Figure 2.12. In general and as expected, a lower noise ratio leads to a lower estimation error. However, there is an exception in the cases when only RSSI sensors are used and the coverage is high (Figs.2.12(a) and 2.12(c)). In those cases, higher noise ratio, such as 0.2 versus 0.1 and 0.01, would actually result in lower estimation error when the coverage exceeds 30%. This is indeed a side-effect of the noise model (recall that the noise model is based on a simple uniform distribution). A higher noise ratio means a node could hear from more neighbors because the uniform noise sometimes increases the actual transmission range.

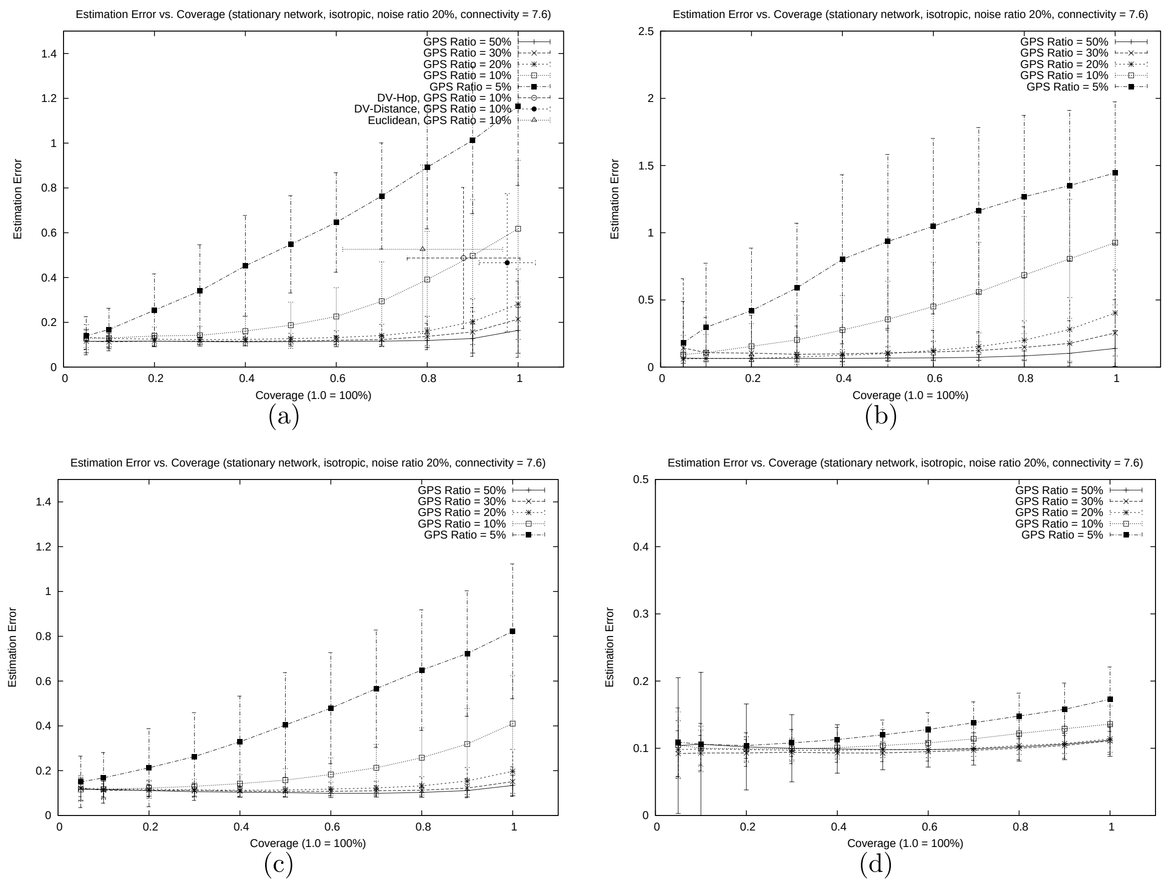


Figure 2.11. Effect of beacon ratio on the estimation error. Four cases include (a) RSSI 100%, AoA 0%, (b) RSSI 0%, AoA 100%, (c) RSSI 50%, AoA 50%, and (d) RSSI 100%, AoA 100%.

Therefore, when the noise ratio is high, even though the distance estimates become less accurate, more neighbors can be heard. In other words, more nodes are likely to obtain their general locations, but those location estimates are not very accurate.

Another observation to be made is that the estimation error difference between various noise ratios decreases as the actual estimation error increases. For instance, the difference between the four noise ratios is quite small when the coverage is 100% and only AoA sensors are used (Figure 2.12(b)). We can also observe that when only AoA sensors are used, fewer nodes can localize themselves simply because of the geometric limitation

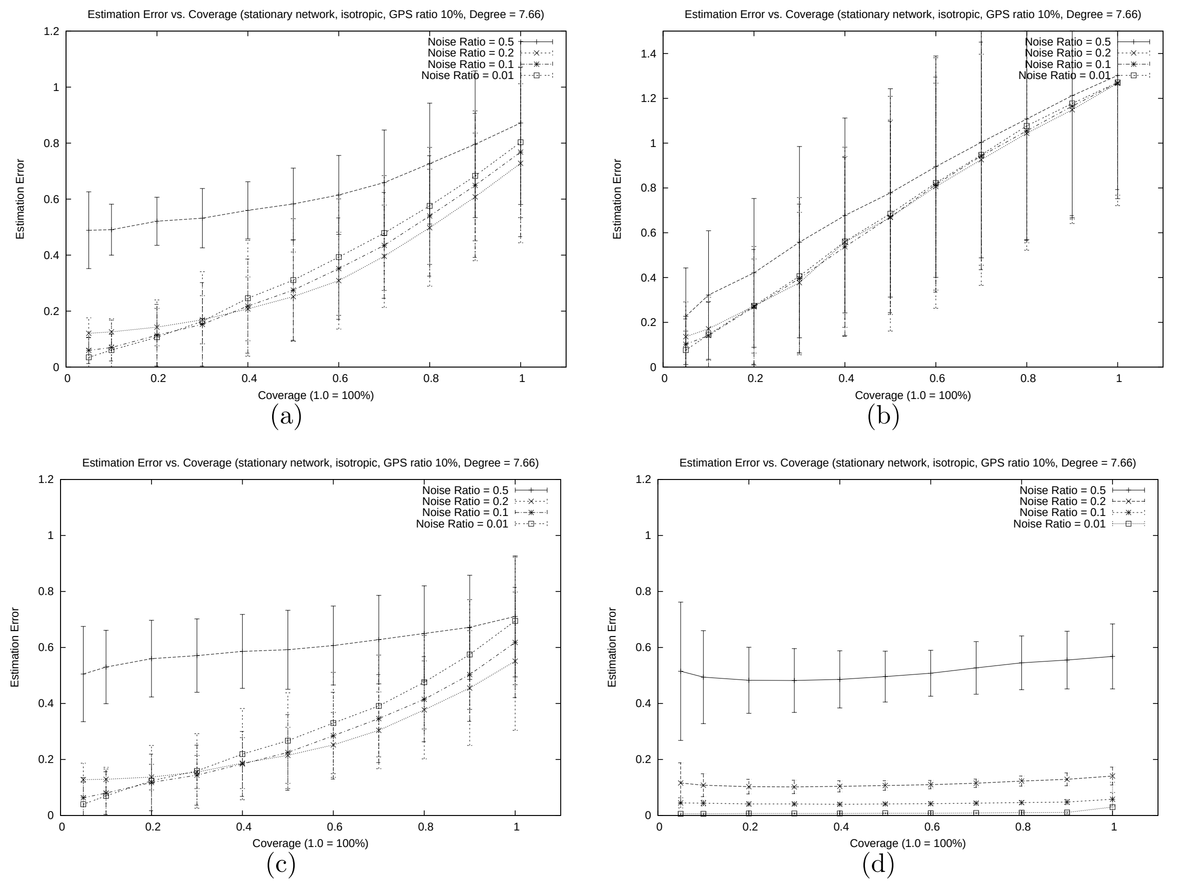


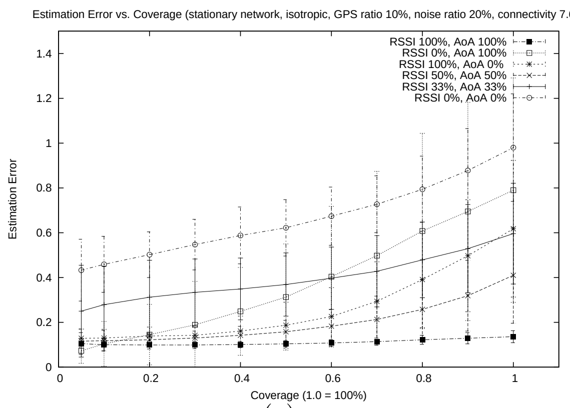
Figure 2.12. Effect of noise ratio on the estimation error. Four cases include (a) RSSI 100%, AoA 0%, (b) RSSI 0%, AoA 100%, (c) RSSI 50%, AoA 50%, and (d) RSSI 100%, AoA 100%.

of the AoA method. Thus, the average localization error for all nodes increases, and such increase is attributed more to the geometric limitation than to the signal noise. Thus, the effect of the noise ratio becomes less apparent.

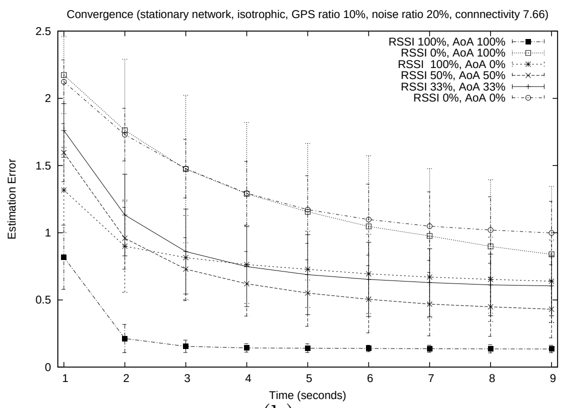
2.7.5 Mixed Sensor Types

To further evaluate the localization result of mixed sensor types, we added two additional types of sensor configurations that include connectivity-only nodes. In particular, we consider networks with one third of the nodes having RSSI, one third having AoA sensors, and one third connectivity-only. We will also consider networks in which all nodes are connectivity-only. For these experiments we use a moderately connected (average degree 7.66) network, a beacon ratio of 10%, and a noise ratio of 20%. Figure 2.13 shows the localization performance of the network with the above configuration versus the other four types of sensor configurations we have seen earlier. In terms of overall estimation error versus coverage (Figure 2.13(a)), networks with one third connectivity-only nodes perform reasonably well. Compared to networks with 100% RSSI capacity, the networks with one third connectivity-only nodes have higher localization error when the coverage is low, but they catch up when the coverage increases. This means that while the estimated locations of individual nodes might not be as accurate as in the networks with 100% RSSI sensors, more nodes are capable of identifying their rough locations. When all nodes are connectivity-only nodes, our particle filter framework still gives reasonable estimates. About half of the nodes are localized within 60% of the transmission range, and all the nodes are localized within 100% of the transmission range.

Figure 2.13(b) shows how the estimation error behaves for different types of sensor configurations as a function of time. For each of the simulation runs, we capture estimated locations at every second and compare them to the actual locations. As expected, the estimated locations become more accurate as more information is exchanged among neighbors. As such, location information from beacons eventually propagates throughout the network and allows regular nodes to localize themselves. Note that the convergence happens smoothly in all sensor configurations. This can be attributed to our filter update methods that rely on the variances of the particle distributions. The differences between the variances allow the nodes with more accurate location information to impact the nodes with less accurate location information, and not vice versa.



(a)



(b)

Figure 2.13. Results on mixed sensor types. The figure shows (a) estimation error and (b) convergence.

2.7.6 Compression vs. No Compression

Figures 2.14(a) and 2.14(b) demonstrate the effectiveness of our compression algorithm on transmitting the particle distribution. The same scenario of localizing using RSSI ranging was repeated when a complete particle distribution is transmitted instead of the compressed version. The results are compared side by side. While the localization algorithm works better when the complete distribution is sent, the differences are rather minimal. While the original particle distribution consists of 200 particles, each of which contains two decimal numbers to designate the location, the compressed version consists of 10 quadruples, each of which contains five decimal numbers. The compression method achieves a total bandwidth saving of 87.5% at each location exchange. Given that the network bandwidth can be expensive, one can easily justify the minimal tradeoff of performance using our compression scheme.

2.7.7 Results on Mobile Networks

Previous works on the AHLP generally do not contain extensive simulation and analysis when the network is indeed mobile (as the definition of MANETs implies). As

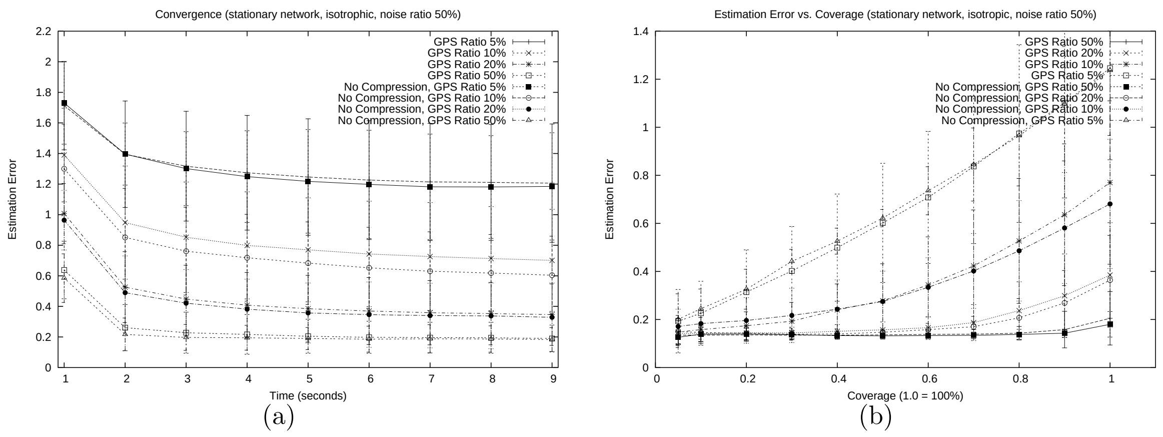


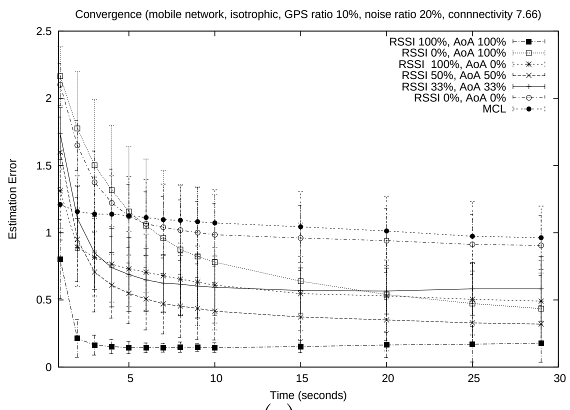
Figure 2.14. Effect of compressing particle distributions. The figure shows (a) filter convergence and (b) estimation error.

discussed earlier, most previous methods are specifically designed to work in stationary sensor networks, in which it is sufficient to complete one round of localization and there is no requirement for further adjustment when the topology changes. Thus, adapting them to work in mobile networks can be quite challenging (see Section 2.8.2 for a more detailed discussion on this). In the worst case, the entire localization scheme has to be redone every time the network changes. Our method however, is specifically designed to work in mobile networks. In our case, since the entire GAHLA relies on simple location exchange between the neighbors and there is no complicated multi-phase operation, we can let the same particle filter framework run continuously as nodes move about.

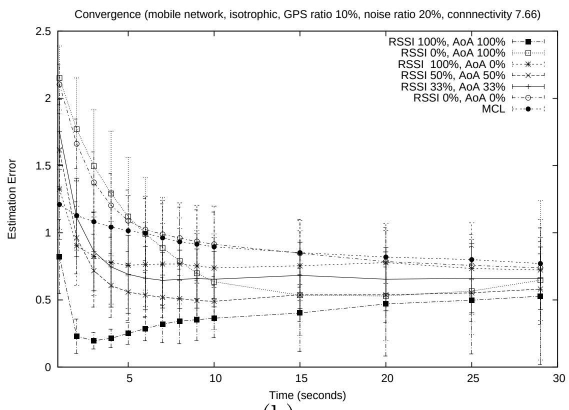
For our mobile node localization simulations we set the population to 100 nodes with an original average degree of 7.6. We use the epoch-based mobility model of [57] to simulate node movement, which is widely accepted as a good mobility model for ad hoc networks (in general it is deemed more realistic than a Brownian motion model). The entire movement path of the node is defined by a sequence of “epochs,” i.e., $(e_1, e_2, \cdots, e_n)$. The duration of each epoch is I.I.D. exponentially distributed with a mean of $1/\lambda$; within each epoch nodes move with a constant velocity vector. At the end of each epoch, nodes randomly select a new velocity vector. The direction of the movement is I.I.D. uniform between 0 and 2π. The absolute value of the velocity is I.I.D. normal with mean μ and a variance of σ^2. Our simulation uses a fixed mean and a fixed variance such that $\mu = \sigma$. The results are obtained by using two different means and standard deviations of 5m/s and 20m/s. The expected amount of time a node maintains its current velocity is set to 5 seconds ($\lambda = 5$).

Figure 2.15 shows the filter convergence on mobile networks for the first 30 seconds. In both low (5m/s), and highly (20m/s) mobile networks, the estimation error drops quickly in the first several seconds. This represents the phase when the nodes localize themselves initially. In slowly moving networks (Figure 2.15(a)), the estimation error stays around the same level after the initial localization, which indicates that the localization process is happening quickly enough to adapt to the location change. In the networks with rapid node movements (Figure 2.15(b)) however, rapid location change increases the overall estimation error. It is interesting to note that network mobility tends to have less impact in the case when AoA sensors and/or connectivity-only nodes are used. In fact, while the stationary networks with RSSI-only nodes outperform the stationary networks with AoA-only nodes, the opposite is true for mobile networks. This indicates that for our particle filter framework nodes with AoA sensors tend to adapt to mobility faster than nodes with RSSI sensors.

For comparison, we also implemented the Monte Carlo localization (MCL) method proposed in [30], which is another probabilistic localization method based on particle filtering. MCL is designed for mobile networks and relies only on the connectivity information, and thus behaves much like connectivity-only node scenario in our algorithm. The main difference is that MCL uses the beacon locations (up to two hops away) as the measurement, whereas our algorithm uses both beacon *and* regular node locations (one hop away). As shown in Figure 2.15, MCL has lower initial localization error because



(a)



(b)

Figure 2.15. Filter convergence in mobile networks. Two cases are considered: (a) average speed = 5 m/s and (b) average speed = 20 m/s.

the location information from the beacons two hops away can be used directly. Our algorithm uses the location information from the beacons more than one hop away indirectly via the exchange of the location distributions among the neighbors, and thus it would take more iterations to converge. However, our algorithm has the advantage that the location information from the beacons is implicitly contained in the location distribution of each node. Via the exchange of location distributions, the beacon information essentially propagates freely and there is no limiting factor of two hops like MCL. Thus, we observe in Figure 2.15 that once particles collapse, our algorithm has lower localization error than MCL.

In mobile networks, the frequency of filter updates can have an impact on the localization accuracy. Since the nodes are constantly moving, the current location estimation can become obsolete very quickly. Thus, the particle filters need to be updated at a sufficient frequency to keep up with the node movement. The effect of the filter update frequency on the localization error is shown in Figure 2.16. Here, we use an average node movement speed of 20 m/s and varies the filter update frequency from 0.1 to 0.5 seconds (previous simulations use 0.5 seconds). As expected, localization error generally increases

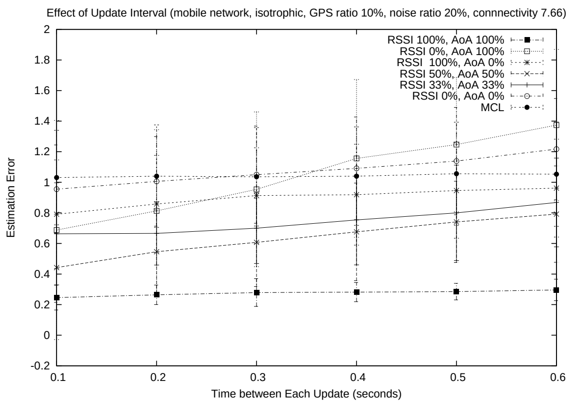


Figure 2.16. Effect of update interval size in mobile networks (average speed = 20 m/s).

with larger interval size. However, the increase is minimal in the case of 100% RSSI and 100% AoA as well as MCL, since they converge faster than others. (We argue that by increasing the update rate appropriately, an arbitrarily rapid mobility could be tracked. Indeed, the update rate should be determined based on the current local mobility rate of nodes.)

2.8 Discussion

Most previous general ad hoc localization algorithms (GAHLA) attempt to pinpoint each node at a single location. While this is the ultimate goal of any GAHLA, it is fundamentally impossible for most topological scenarios. Thus, the challenge in the AHLP really lies in *estimating* the location of nodes, especially those that are more than one hop away from the beacons. In such cases, a node might not know its *exact* location, but it may estimate a number of *possible* locations (thus obtaining partial information). A probabilistic approach provides a natural way of representing such partial information. In particular, the node location may not be presented as a fixed value but as a probability distribution. Initially, a node unaware of its own location has its location distribution uniformly spread over the entire deployment area. As the localization process proceeds, the location distribution is updated and it is expected to converge to a more concentrated location estimate (ideally to a single high probability location). In this work, we have applied such a probabilistic model using various measurements and particle filtering as the method of updating the location distributions. We showed that the probabilistic method achieves much better balance of the trade-off between the estimation error and coverage when considering the limitation of DV-Hop/DV-Distance (high estimation error) and Euclidean (low coverage). Our method has the following advantages over most existing GAHLAs:

1. *Provide a measure of estimation quality.* DV based algorithms can generate location estimates covering only a subset of nodes. The coverage of the estimates depends on the nature of the algorithm. There is always trade-off between the coverage and the quality of the estimates. Some algorithms (such as DV-Hop) give better coverage, while others (such as Euclidean) give better estimates. Our method, however, generates location estimates for all nodes in the network. Each estimate has a variance associated with it serving as a quality measure. Thus, the coverage of our estimates is not a fixed value but a function of the variances. In practice, certain applications might desire better estimation quality while other might desire better coverage. For instance, location-aided routing protocols [42] might only need a very rough location estimation of the destination node when the request is still far away. But as the request moves closer to the destination node, more accurate location estimation is needed. Previously, different GAHLAs had to be applied separately to accomplish the two objectives. Our method produces results satisfying both scenarios and does it in the same probabilistic framework.

2. *Single phase operation.* Many algorithms employ multiple phases during the localization process. For instance, DV-Hop requires a first phase to calculate per-hop distance and a second phase to propagate the result. Multilateration methods [71]

require three phrases: initial estimation, grouping, and refinement. Our method, however, has the advantage of a single phase operation. From the implementation point of view, our algorithm can be easily implemented in a distributed asynchronous fashion because nodes do not have to collectively maintain the state information of "which phase are we in?" From the functional point of view, the probabilistic nature of our method simplifies the algorithm by eliminating the need for multiple phases. In multilateration methods, an initial estimate is obtained based on a certain measure (distance or hops) to beacons followed by the phase of further refinement. The initial location estimate suffers because information from regular nodes is not used. The refinement phase is needed so that information from regular nodes can be incorporated into the estimates. Our method does not need separate phases, as the information from regular nodes is automatically applied as soon as it becomes available. In particular, regular nodes become more and more certain of their locations (their location variances decrease) allowing their estimates to be used by neighboring nodes.

3. *Simple communication model and fast convergence.* Our method employs a simple computation and communication model which relies solely on local broadcast (broadcast to neighbors only). This allows our method to be naturally integrated into periodical Hello messages (as generally used by mobile nodes in ad hoc networks to declare their existence); we do not require a new type of control message. Furthermore, our simulations show that compared to existing methods such as APS, our method generally converges with less message-overhead.
4. *Mobile ready.* Because our algorithm eliminates multiple phases and uses a simple communication model, it can be applied directly to mobile networks. While previous works do not generally provide simulation result for mobile scenarios, we

demonstrate via simulation that our method can be effectively used in mobile ad hoc networks.

5. *Extensibility.* The core of our algorithm is a probabilistic framework based on particle filtering that is extremely versatile. The framework can be easily extended to different signal and network models. For instance, unlike DV-Hop, our method does not assume that all nodes have the same transmission range. Unlike Centeriod or APIT, our method does not require a greater range for GPS nodes, which allows it to work in homogeneous networks. The framework is not tied to particular signal propagation model or a particular sensory data.

To the best of our knowledge, our work is the first that incorporates multiple sensory data for localization using the same algorithmic framework and enables localization for nodes in both static as well as mobile multihop networks. The need for such algorithms is established in [14], but the authors only consider the limited case when both RSSI and AoA are available at nodes. Our work is more general in that we consider the cases when the nodes can either have RSSI, AoA, both, or no sensory data at all. The same particle filter framework allows the nodes with different sensory capacities to collaborate in the localization.

2.9 Summary

In this chapter, we proposed a particle filter framework that solves the localization problem in both stationary and mobile ad hoc networks. Compared to previous localization algorithms, our framework is general enough to accommodate different sensory capacities regardless of the availability of ranging (such as RSSI) or sectoring (such as AoA). More importantly, our framework allows the networks consisting of nodes with different sensor types to collaborate in the localization process. By leveraging the filter variances, regular nodes localize themselves by simple location data exchanges with their neighbors without having to go through an initial localization phase and a refinement phase; the filter variances also give a measure of the estimation accuracy. The differentiation in estimation accuracy could be very useful as different applications might have different accuracy requirements.

Our analysis and simulation studies showed that in stationary networks AoA sensors, by themselves, do not work well when the network connectivity or the beacon ratio is low. While using only AoA sensors would result in poorer localization, good results can be achieved by combining AoA only sensors and RSSI only sensors in the same network. In fact, networks where 50% of the nodes have only AoA sensors and 50% of the nodes have only RSSI sensors can achive better localization result than networks dominated by one type of sensors. Furthermore, our analysis shows that employing AoA sensors is highly beneficial in mobile networks, especially when the network mobility is high. Simulation studies also validate the effectiveness of incorporating different sensor capacities (RSSI, AoA, or connectivity-only) using the same particle filter framework. With our framework, reasonable results can be obtained even if one third of the nodes are connectivity-only.

While this work addresses network mobility, we believe that there is room for further evaluation. In particular, our current framework assumes that particle filters are continuously updated with the same rate. The update rate however could be tied to the mobility in such a way that the location updates are executed on demand. This way, when the node movement is minimal, we could eliminate the unnecessary updates, and thus reduce the network traffic and save the power consumption by switching off the sensors. Furthermore, we are currently using a rather simplistic movement model in the particle filter (Gaussian displacement). This model could be improved by learning movement patterns (models) and their parameters (such as velocity and acceleration); we are targeting our research at those enhancements for mobile networks in the near future.

CHAPTER 3

LOCALIZATION USING INTERFEROMETRIC RANGING

3.1 Problem Definition

Chapter 2 introduced the general localization problem in ad hoc networks based on the measurement data of connectivity, TOA and RSSI ranging, or AoA. In most cases, the nodes collaborate to derive the location based on the anchor locations and the sensory data observed. Unfortunately, it has been shown that the localization problem is NP-Complete when localized from either ranging [18], angle [8] or unit-disk connectivity [45]. Thus, the localization problem is treated in many cases as a stochastic optimization problem, for which a distributed solution is more desirable.

Other than the already investigated sensory types, a new sensory type called interferometric ranging has recently been proposed [55]. Interferometric ranging is a "widely used technique in both radio and optical astronomy to determine the precise angular position of celestial bodies as well as objects on the ground [47]." Its original design is to work with large scale systems where objects are thousands of miles apart. However, there have been recent advances in hardware design that allow interferometric ranging to be performed on cheaper hardware, making it a promising new technique for localizing ad hoc and sensor networks. Interferometric ranging exploits the property that the relative phase offset between two receivers determines the distances between the two senders. By synchronizing the transmission at the two senders, the distance difference (also called the *q-range*) can be measured very accurately using interferometric ranging.

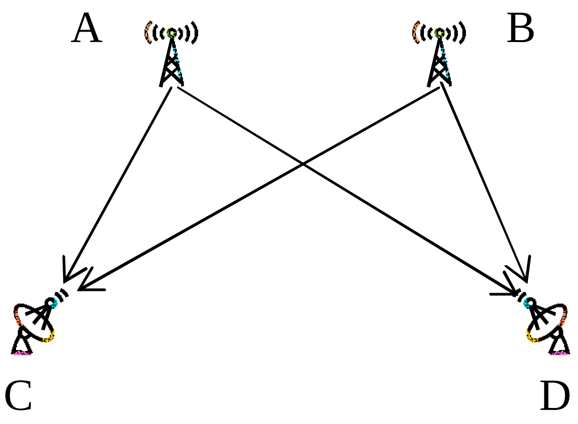


Figure 3.1. The interferometric ranging measurement. The measurement gives a q-range $d_{ABCD} = d_{AD} - d_{BD} + d_{BC} - d_{AC}$. Here, node A and B are the senders, and node C and D are the receivers.

Definition 3.1.1. *A* q-range *obtained from interferometric ranging from two senders A and B, and two receivers C and D is the distance difference $d_{ABCD} = d_{AD} - d_{BD} + d_{BC} - d_{AC} + e$, where e is the measurement error (Figure 3.1).*

Definition 3.1.2. *Given a q-range d_{ABCD}, A, B, C and D are also referred to as the* components *of the q-range, in which A and B are the* senders *and C and D are the* receivers*.*

Note that based on the above definitions, A is interchangeable with B since both are senders, and C is interchangeable with D since both are receivers. Thus, $d_{ABCD} = -d_{BACD} = -d_{ABDC} = d_{BADC}$.

A major advantage of interferometric ranging is that the measurement could be extremely accurate compared to noise-prone RSSI readings. In a recent experiment [55], in which 16 nodes are deployed in a 4x4 grid over a 18x18 meters flat grassy area with no obstruction, the maximum q-range error was shown to be around 0.1 meters while the medium error was less than 0.04 meters. However, interferometric ranging is more difficult to implement partially due to the following reasons:

1. Precise time synchronization is needed at all four components of a q-range.

2. Frequencies of the transmissions need to be precisely calibrated.
3. A significantly larger number of measurements are needed for localization than using direct ranging techniques.
4. Since each measurement involves four nodes, more collaboration is required between nodes.

Those difficulties rooted in the physical characteristics of interferometric ranging devices affect the algorithmic design of the localization algorithm. In this chapter, we concentrate on the algorithmic aspects of the problem, and in particular, the last two difficulties. In other words, we only consider how to localize the network from a set of given q-ranges, and we do not consider how those q-ranges are obtained. A detailed overview of the physical characteristics of interferometric ranging is given in [55].

In this chapter, we will start by reviewing some fundamental complexity results on localization using interferometric ranging. We show in Section 3.3 that there is a polynomial time algorithm that checks for a necessary condition under which a network can be localized. We also show that the localization problem itself is NP-Complete when using interferometric ranging as the measurement. In Section 3.4, we further derive a sufficient condition that guarantees a unique localization of a node based on local interferometric readings. Using the condition, we propose an iterative localization algorithm that localizes the network from a small number of seeding anchors. The performance of the localization algorithm and its error propagation behavior are validated using simulations in Section 3.5.

3.2 Previous Works

The large number of measurements required for localization using interferometric ranging are illustrated by the following theorem (and proof) given in [47].

Theorem 3.2.1. *In a network of n nodes, there is a maximum of $n(n-3)/2$ independent interferometric measurements that can be obtained [47].*

Theorem 3.2.1 shows the number of measurements available using interferometric measurements is $O(n^2)$. Considering the localization problem in relative coordinates, for a network of n nodes there are $2n-3$ unknowns in 2 dimensions and $3n-6$ unknowns in 3 dimensions [1]. Thus, the smallest network that can be localized using interferometric measurements is a fully-connected network with a population of $n = 6$, where there are 9 independent measurements available to cover 9 unknowns.

Furthermore, for interferometric ranging not all measurements are useful. Some measurements are dependent on others, and only independent measurements are useful in localization. For instance, for the four nodes A, B, C and D in Figure 3.1, if all nodes are completely connected (i.e., any two can be the senders or the receivers), then there are only two independent q-ranges, e.g., $d_{ABCD} = d_{AD} - d_{BD} + d_{BC} - d_{AC}$ (when A and B are the senders) and $d_{ADBC} = d_{AC} - d_{DC} + d_{DB} - d_{AB}$ (when A and D are the senders). All other q-ranges are dependent upon those two, that is, they can be expressed as linear combinations of those two (a different basis set could be picked as well).

Given a set of interferometric measurements (i.e., q-ranges), a localization algorithm attempts to find the sensor locations that satisfy the measurements. There have been a limited number of localization algorithms proposed for interferometric ranging. A genetic algorithm approach was taken in [55]. An algorithm was proposed in [67] that uses both interferometric and RSSI ranging. Both algorithms try to optimize for a global solution given an entire set of interferometric measurements. Intuitively, finding a global solution to the localization problem is often difficult because of the large search space and the large number of constraints given by the interferometric measurements. Thus,

[1]This is because the relative coordinates are invariant under translation, rotation and reflection. Thus, in 2 dimensions, we have $2n-3$ degrees of freedom, where translation, rotation and reflection each reduce one degree of freedom.

it is desirable to find the solutions in some subspaces first and then incrementally build up to the global solution. We make use of an iterative approach to localize the network, which was first introduced in [18].

3.3 Complexity Results

In this section, we provide some complexity results on localization using interferometric ranging. We will show when using interferometric readings as the measurement the complexity of the localization problem is NP-Complete. In the remainder of this section dealing with the complexity, we assume that the q-range measurement error e is insignificant, and all q-ranges give the precise distance difference. We will reconsider the measurement error in the subsequent sections.

First, we extend Theorem 3.2.1 to give a polynomial time algorithm that determines the number of independent interferometric readings.

Theorem 3.3.1. *Given a network of n nodes and a set of interferometric readings (q-ranges), there is a polynomial time algorithm that determines how many of them are independent (the set's dimension).*

Proof. We start by denoting each node in the network with an integer ID starting from 0 to $n-1$, and let A, B, C and D be variables containing a unique ID. Considering a vector space consisting of all possible q-ranges, the algorithm needs to identify the total number of independent q-ranges from a given set of vectors $\{d_{ABCD}\}$. To do this, one can use the Gaussian elimination method *if* the given vectors can be written as some linear combination of a set of basis vectors. The classification algorithm given in [47] provides a way to accomplish this, as follows.

Given a vector $d_{ABCD} = d_{AD} - d_{BD} + d_{BC} - d_{AC}$ in a vector space, it has been shown in [47] that any vector d_{ABCD} should satisfy the condition $A < B, A < C < D, B \neq$

$C, B \neq D$ in order to be independent since $d_{ABCD} + d_{BACD} = 0$ and $d_{ABCD} - d_{CDAB} = 0$. Thus, we can convert any q-ranges not satisfying the above condition to a q-range that does. Furthermore, any such vector d_{ABCD} that satisfies the above condition belongs to one of the following six classes:

- Class 0: $\{012D|2 < D\}$
- Class 1: $\{0B1D|1 < B < D\}$
- Class 2: $\{01CD|2 < C < D\}$
- Class 3: $\{0B1D|1 < D < B\}$
- Class 4: $\{0BCD|1 < B, 1 < C < D, B \neq C, B \neq D\}$
- Class 5: $\{ABCD|0 < A < B, A < C < D, B \neq C, B \neq D\}$

Among the six classes, Class 0 and Class 1 form a basis set that only contains independent vectors. Vectors in Class 2 through 5 can be written as linear combinations of those in the first two classes as follows:

- Class 2: $d_{01CD} = -d_{012C} + d_{012D}$
- Class 3: $d_{0B1D} = -d_{01DB} + d_{0D1B}$
- Class 4: $d_{0BCD} = -d_{0B1C} + d_{0B1D}$
- Class 5: $d_{ABCD} = -d_{0ACD} + d_{0BCD}$

Using the above algorithm, *any* given q-range in the set $\{d_{ABCD}\}$ can be written as a linear combination of the q-ranges in Class 0 and Class 1 (i.e., a basis set), with -1, 1, or 0 as the coefficient at each term.

Given a set of N q-range vectors, we construct a M-by-N matrix $\mathbb{A}$, where M is the total number of Class 0 and Class 1 vectors. For every q-range i, we run the above reduction algorithm to reduce it to a linear combination of Class 0 and Class 1, and then insert the coefficients (-1, 1, or 0) into the ith column of the matrix $\mathbb{A}$. We can then use Gaussian elimination on matrix $\mathbb{A}$ (or some other techniques in linear algebra to find the rank), which will indicate the number of independent columns of $\mathbb{A}$. Since for a network of n nodes there are $2n-3$ unknowns in 2 dimensions and $3n-6$ unknowns in 3 dimensions, we can compare the total count with $2n-3$ or $3n-6$ to determine whether the network is localizable in 2 dimensions or 3 dimensions.

For each q-range in a total of N q-ranges, the algorithm to find the coefficients in a column of $\mathbb{A}$ runs in constant time. The complexity of constructing the matrix $\mathbb{A}$ is therefore $O(N)$. From the constructed matrix $\mathbb{A}$, the complexity of running Gaussian elimination is $O(N^3)$. Thus, the overall complexity is $O(N + N^3) = O(N^3)$, polynomial time. $\square$

As argued in the previous section, the problem of localizing a network of n nodes in 2 dimensions involves $2n-3$ unknowns. Therefore, if there are at least $2n-3$ independent q-ranges, then we will have a system of $2n-3$ independent equations to sufficiently solve for the $2n-3$ unknowns. Thus, the polynomial time algorithm enables us to check for a necessary condition of network localizablity under interferometric readings. However, since each q-range is not a linear but a quadratic equation, having $2n-3$ independent q-ranges is only a necessary but not sufficient condition of network localizablity. Since testing the network localizablity using edge weights (distances) is a known NP-Complete problem [72], it is likely that network localizablity using interferometric ranging is also NP-Complete.

The results above assume a general graph. In reality, ad hoc and sensor networks resemble unit disk graphs. How many independent q-ranges are we expecting in unit disk graphs then? Simulations on randomly deployed unit disk graphs show that the number of independent q-ranges increases linearly with the network connectivity (average nodal degree) as depicted in Figure 3.2.

The following theorem shows that even with a known localizable network, the localization itself is NP-Complete when using interferometric ranging as the measurement.

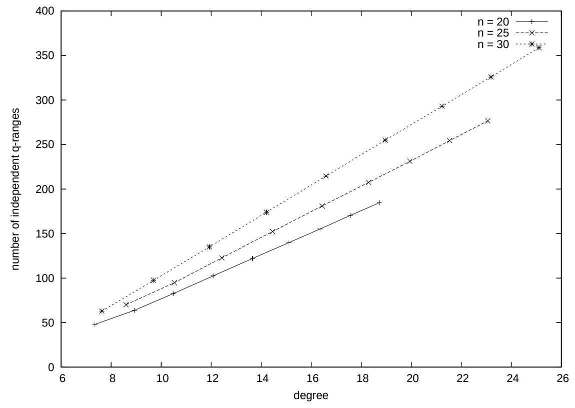


Figure 3.2. The number of q-ranges available in a unit disk graph. In a randomly placed unit disk graph of size n, the number of independent q-ranges increase linearly to the network connectivity (average node degree).

Theorem 3.3.2. *Given a network that is localizable using a set of interferometric readings (q-ranges), the actual localization of the network is an NP-Complete problem.*

Proof. When a solution instance (certificate) is given (i.e., when all node locations are known), there is a polynomial time algorithm to validate such an instance by simply looping through all the q-ranges. Thus, the localization problem is in NP. We now show the problem is NP-Complete.

We reduce our proof from the NP-Complete realization problem of wheel graphs. A wheel graph, W_n, is a graph of n nodes in which (without losing generality due to node numbering) nodes 1 through $n-1$ form a cycle (not necessarily on a circle), and node 0 (hub) is connected to all nodes. The edges on the cycle are called the *rim edges*, and edges connecting from the hub are called *spokes*. Figure 3.3(a) shows such a graph with $n = 6$.

It has been shown in [18] that it is NP-Complete to localize the wheel graph W_n when the edge weights (including spokes and rim edges) are known. To show that it is also NP-Complete to localize a network using interferometric readings (q-ranges), we construct a polynomial time reduction from W_n. We then claim that the reduced graph,

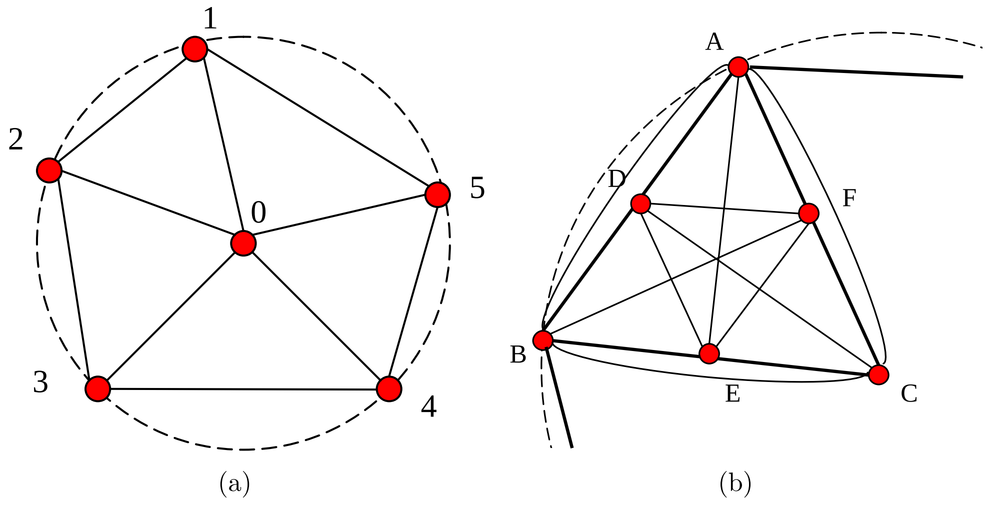


Figure 3.3. Wheel graph and its reduction. The figure shows (a) wheel graph W_6 and (b) reduction from wheel graph.

called $\hat{W}_n$, is localizable using the q-ranges obtainable from the edge weights of W_n. And finally we show that such reduction leads to the conclusion that localizing using q-ranges is NP-Complete.

The reduction from W_n to $\hat{W}_n$ works as follows. Observe that W_n consists of a sequence of triangles formed by two spokes and a rim edge. On every edge of each triangle, we create an additional node equal distance away from both end points. We then add additional edges to completely connect all six nodes (three original nodes and three newly created nodes) within the triangle. For instance, in Figure 3.3(b), we create three new nodes D, E, and F within $\triangle ABC$ such that $d_{AD} = d_{DB} = \frac{d_{AB}}{2}$, $d_{BE} = d_{EC} = \frac{d_{BC}}{2}$ and $d_{AF} = d_{FC} = \frac{d_{AC}}{2}$. The subgraph consisting of those six nodes are completely connected. Clearly, this step is polynomial in n.

We claim that the graph $\hat{W}_n$ is localizable in 2 dimensions using interferometric ranging. To support our claim we need to determine how many interferometric readings (q-ranges) are present in $\hat{W}_n$. First, observe that by our construction, the weight of *every* newly created edge in the triangle subgraph can be determined geometrically from three edge weights d_{AB}, d_{BC} and d_{AC} that are given by the original W_n graph. Thus, within each triangle subgraph of $n = 6$, we have *all* available q-ranges for a complete graph of $n = 6$. Conversely, from those available q-ranges, we can derive the original edge weights d_{AB}, d_{BC} and d_{AC}. For instance, we can calculate d_{AC} from the following four q-ranges:

$$\begin{aligned} d_{AECB} &= d_{AB} - d_{EB} + d_{EC} - d_{AC} \\ &= d_{AB} - \frac{d_{BC}}{2} + \frac{d_{BC}}{2} - d_{AC} \\ &= d_{AB} - d_{AC} \end{aligned} \tag{3.1}$$

$$\begin{aligned} d_{CDAB} &= d_{BC} - d_{DB} + d_{AD} - d_{AC} \\ &= d_{BC} - \frac{d_{AB}}{2} + \frac{d_{AB}}{2} - d_{AC} \\ &= d_{BC} - d_{AC} \end{aligned} \tag{3.2}$$

$$\begin{aligned} d_{BADF} &= d_{BF} - d_{AF} + d_{AD} - d_{BD} \\ &= d_{BF} - \frac{d_{AC}}{2} + \frac{d_{AB}}{2} - \frac{d_{AB}}{2} \\ &= d_{BF} - \frac{d_{AC}}{2} \end{aligned} \tag{3.3}$$

$$\begin{aligned} d_{BECF} &= d_{BF} - d_{EF} + d_{EC} - d_{BC} \\ &= d_{BF} - \frac{d_{AB}}{2} + \frac{d_{BC}}{2} - d_{BC} \\ &= d_{BF} - \frac{d_{AB}}{2} - \frac{d_{BC}}{2} \end{aligned} \tag{3.4}$$

Combining (3.1) with (3.2), and (3.3) with (3.4), we have

$$d_{AECB} + d_{CDAB} = d_{AB} + d_{BC} - 2 \cdot d_{AC} \tag{3.5}$$

$$\frac{d_{AC}}{2} = -d_{BECF} + d_{BADF} - \frac{d_{AB}}{2} - \frac{d_{BC}}{2} \tag{3.6}$$

Combining (3.5) with (3.6), and solving for d_{AC}, we have

$$d_{AC} = \frac{-2 \cdot d_{BECF} + 2 \cdot d_{BADF} - d_{AECB} - d_{CDAB}}{3}$$

It has been shown in [18] that W_n is localizable when all of its edge weights are given. Since, using the q-ranges of $\hat{W}_n$ available to us, we can derive that every edge weight of W_n, W_n is localizable using the q-ranges. If W_n is localizable using q-ranges, so is $\hat{W}_n$ because by our reduction the location of every newly added node in $\hat{W}_n$ can be uniquely determined from the location of the existing nodes in W_n. Thus, $\hat{W}_n$ is localizable using q-ranges.

If there is a polynomial time algorithm, $\mathcal{A}$, that performs the actual localization of $\hat{W}_n$ using q-ranges, the original wheel graph W_n can then be localized under edge weights by running our polynomial time reduction to produce $\hat{W}_n$ and then running $\mathcal{A}$ to localize $\hat{W}_n$, all in polynomial time. However, since localizing the wheel graph W_n using edge weights is NP-Complete as shown in [18], $\mathcal{A}$ does not to exist unless $P = NP$. Thus, localization using interferometric ranging (q-ranges) is NP-Complete. $\square$

3.4 Iterative Localization Using Interferometric Ranging

Based on the above complexity result, it is clear that localization using interferometric ranging is fundamentally intractable. Any algorithm that solves this problem will need to be some form of a heuristic (e.g., described as an optimization process). However, even as an optimization problem, the problem is difficult because of the large search space. In this section, we try to provide an optimization solution by localizing the nodes incrementally based on a sufficient (but not necessary) condition of node-localizability in terms of interferometric ranging.

Lemma 3.4.1. *A node i can be localized using interferometric ranging with high probability under the following two conditions:*

1. *i is a component of at least three mutually independent q-ranges,* **and**
2. *all of the other three components in each q-range are localized.*

Proof. Let node i be a component in the three q-ranges. For each of the three q-ranges, let the other three components be node A, B and C. Thus, the q-range is in one of the following forms, depending on whether i is a sender or a receiver:

- $d_{ABiC} = d_{AC} - d_{BC} + d_{Bi} - d_{Ai}$
- $d_{iABC} = d_{iC} - d_{AC} + d_{AB} - d_{iB}$

Since A, B and C are already localized, the distances d_{AB}, d_{AC} and d_{BC} can be calculated from the node locations. Thus, the q-range can be reduced to the following:

- $c_1 = d_{ABiC} + d_{BC} - d_{AC} = d_{Bi} - d_{Ai}$
- $c_2 = d_{iABC} + d_{AC} - d_{AB} = -d_{iB} + d_{iC}$

Here, c_1 and c_2 are two constant values. Thus, each q-range reduces to a partial (either a left-side or a right-side) hyperbola on which the location of node i resides. Since the three q-ranges are mutually independent, each of the partial hyperbolas they generate is unique. Ignoring the special cases that might cause multiple overlapping points among the three hyperbolas (such as when all the focus points of the hyperbolas are collinear), with high probability the intersection of three unique partial hyperbolas is a unique point. Thus, node i can be localized. $\square$

Lemma 3.4.2. *Any three q-ranges with node i as a common receiver (sender) are independent if*

1. *each q-range has a distinct pair of senders (receivers),* **and**
2. *there are in total at least four distinct senders (receivers).*

Proof. First consider the trivial case where the dependency is in two q-ranges. Two types of dependencies exist in this case: $d_{ABCD} + d_{BACD} = 0$ and $d_{ABCD} - d_{CDAB} = 0$. Based on the condition given, $d_{ABCD} + d_{BACD} = 0$ would not happen because they do not have a distinct pair of senders. $d_{ABCD} - d_{CDAB} = 0$ would not happen due to the lack of a common receiver (sender).

Now consider the case when the three q-ranges are dependent. Since there are four distinct senders (receivers) in the three q-ranges, not *all* of the four senders (receivers) can appear multiple times in the q-ranges. Let j be the sender (receiver) that appears only once in the q-ranges. Thus, there is only one q-range that includes the distance between node i and j, d_{ij}. However, since j only appears once, the other two q-ranges do not include d_{ij}. Since d_{ij} is a unique term, the q-range that includes j cannot be written as a linear combination of the other two.

Thus, all three q-ranges must be independent. □

Lemma 3.4.1 and 3.4.2 give the condition under which a node can be localized with high probability using interferometric ranging from its neighbors. If such condition is satisfied, the node can be localized with only the local neighborhood information without the complexity of a global optimization problem. Once the node is localized, its location information can be used to further localize other nodes. This gives an iterative localization algorithm shown as Algorithm 3, which is similar to the iterative trilateration protocol (ITP) proposed in [18], however with different conditions.

Algorithm 3 Iterative Localization Using Interferometric Ranging

```
Require: every node knows its 1-hop neighbor
  for all Localized nodes do
    broadcast its location and 1-hop neighbor set
  end for
  for all Unlocalized nodes $i$ do
    $S_{senders} \leftarrow \phi$
    Receive broadcasts and construct local connectivity map
    Find nodes $s_1$, $s_2$ and $r$ such that $e_{s_1,i}$, $e_{s_2,i}$, $e_{s_1,r}$, and $e_{s_2,r}$ exist in the local map
    if $(s_1, s_2) \notin S_{senders}$ and $(s_2, s_1) \notin S_{senders}$ then
      add $(s_1, s_2)$ to $S_{senders}$
    end if
    if $S_{senders}$ contains at least 3 pairs and at least 4 distinct senders then
      negotiate to obtain q-ranges using each pair as senders or receivers
      determine its location
      broadcast its location and 1-hop neighbor set
    end if
  end for
```

Algorithm 3 requires all nodes to have their 1-hop connectivity information, which can be collected by observing ”Hello” messages from the neighbors. Before starting the localization process, a small number of localized nodes (anchors) need to be deployed. Due to the condition listed in Lemma 3.4.1 and 3.4.2, those anchors need to be close to each other such that the nearby unlocalized nodes can be localized. The anchors then broadcast their locations and 1-hop connectivity information. When an unlocalized node hears the broadcast, it builds a local connectivity map, which is needed to validate the condition required by Lemma 3.4.1 and 3.4.2. In particular, when there are at least three distinct pairs of potential interferometric senders (s_1, s_2), each of which shares a common receiver r, then the node can be localized. When the node detects that the localization condition is satisfied, it contacts the potential sender pairs and the corresponding receiver to schedule an interferometric reading. When at least three readings are obtained, the node computes its location, and then broadcasts it to its neighbors so that the newly discovered location can be used to localize the neighbors in the next round. The localization algorithm continues until all nodes are localized or the next round does not produce any newly localized node.

Simplicity is the real advantage of the above iterative algorithm. Instead of trying to solve for a global solution for all unlocalized nodes in the network at once, the algorithm computes the locations based on the local information only and then progressively builds up a global solution. The algorithm is also distributed in nature and can be implemented directly on each sensor. However, since the conditions dictated by Lemmas 3.4.1 and 3.4.2 are *not* sufficient conditions for a node to be localized, the algorithm does *not* guarantee to localize a node even though the node *could* be theoretically localized. For a randomly deployed sensor network, the ratio of the localizable nodes to the population using the iterative algorithm is a function of the network density. Fortunately, our simulations show that this ratio is reasonably high when compared to the localization ratio of the iterative trilateration protocol (ITP) in [18] that uses direct RSSI ranging.

To compute nodes' locations after the sufficient number of q-ranges are obtained, we run a simple simulated annealing algorithm [41]. The simulated annealing approach is taken because i) in reality the system is often over-determined by multiple q-ranges which contain errors, and ii) the location function to be optimized is non-linear with multiple local minimums. Since the localization is performed using local q-ranges only, we are able to drastically limit the search space to be the area within the range of all components involved in the transmission. Simulations show that the simulated annealing algorithm converges quickly to the correct solution.

3.5 Simulation Results

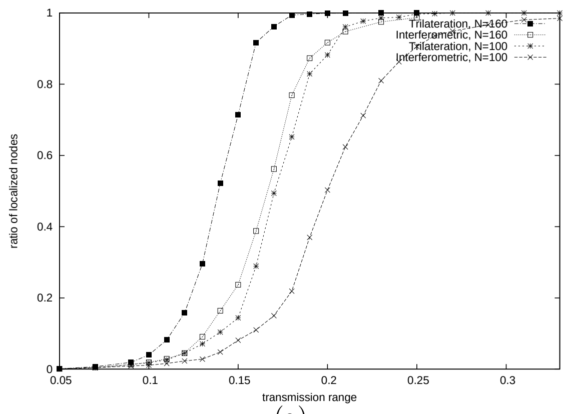
To evaluate the behavior of the iterative localization algorithm using interferometric ranging, we have conducted a number of simulations in various of settings. Our simulation environment consists of a network of n sensors randomly deployed in a square area of 1 unit square. The sensors are assumed to be homogeneous, i.e., they all have the same transmission range. To start the localization process, we deploy four anchors to the center of the unit square (the anchors have the same transmission range as the sensors). The results obtained are the average of 30 independent runs.

3.5.1 Coverage and Rounds

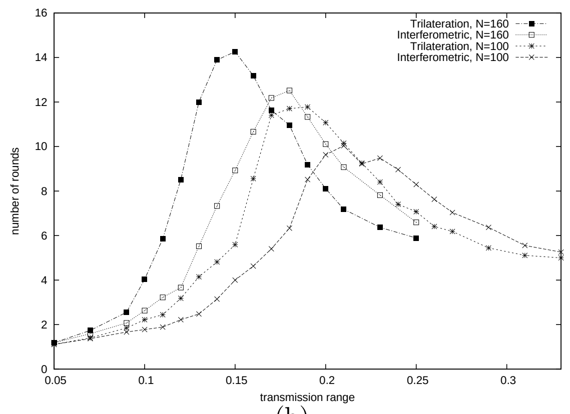
We first look at the localization coverage of the iterative algorithm and the number of rounds executed by the algorithm. We compare the result against that of the iterative trilateration protocol (ITP) proposed in [18]. The node localization condition of the ITP is that an unlocalized node needs to be the neighbor of at least three localized nodes. The localization condition stated in Lemma 3.4.1 and 3.4.2 is stricter since it requires three independent q-ranges. At the minimum, an unlocalized node needs to neighbor with four localized nodes to satisfy such a condition [2]. Thus, it is expected that for the same network the localization coverage produced by the iterative algorithm for interferometric ranging is lower than that of the trilateration.

As validated in Figure 3.4(a), for a fixed transmission range, the number of nodes localized using the interferometric condition is indeed smaller than that of the trilateration condition. However, the difference is not particular great and can be overcome by increasing the network density. As shown in Figure 3.4(a), by increasing the node density (by increasing the population to $n = 160$), the interferometric localization condition can reach a coverage similar to what trilateration reaches at $n = 100$. In terms of density, the network of $n = 100$ has a degree of 12 when the transmission range is set to 0.2, which results in 90% coverage under the trilateration condition. The equivalent coverage can be obtained under the interferometric condition by increasing the density to 19 degrees

[2]Three localized nodes, along with the unlocalized node, would generate a maximum of two independent q-ranges instead of three.



(a)



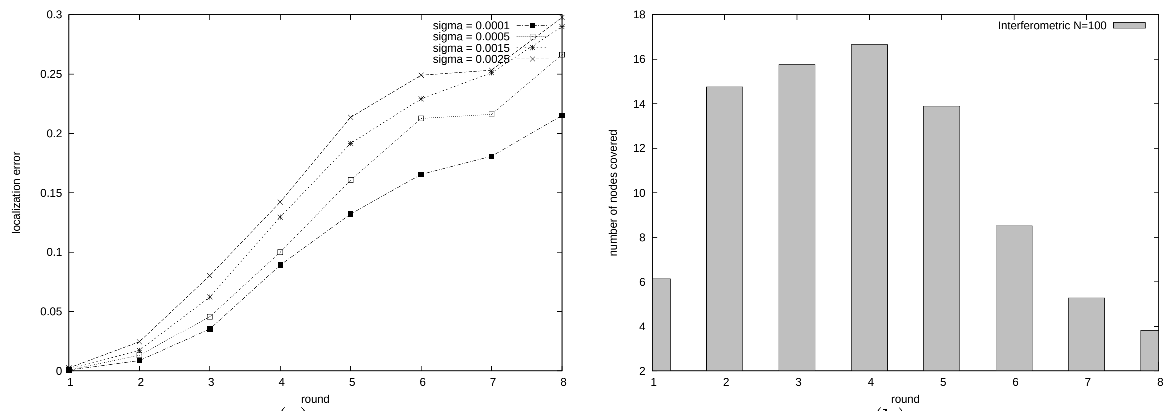
(b)

Figure 3.4. Coverage and rounds of the iterative localization method. The figure shows (a) localization coverage and (b) the number of rounds required.

when using $n = 160$ with the same transmission range. The number of rounds required to complete the iterative algorithm is shown in Figure 3.4(b).

3.5.2 Localization Error

The iterative localization algorithm allows us to study the error propagation behavior of interferometric ranging. In particular, we are interested in how the error from the interferometric measurement affects the localization error and how the error is aggregated and propagated through the network. Figure 3.5(a) shows the average localization error at each round for a network of 100 nodes using the iterative algorithm (transmission range set to 0.25 units). A Gaussian noise of $N(0, \sigma)$ is added to the interferometric measurement. The standard deviations are derived by the actual hardware devices in [55]. Figure 3.5(a) indicates a linear increase of the localization error of the nodes localized at each additional round. Thus, it is more desirable that most nodes are localized with limited rounds. The number of nodes localized at each round is shown in Figure 3.5(b). In our simulation scenario, most nodes are localized within 5 rounds. Nevertheless, the simulation indicates that the error propagation behavior poses a significant constraint



(a) (b)

Figure 3.5. Localization error of the iterative localization method. The figure shows (a) localization error of the nodes localized at each round and (b) the number of nodes localized at each round.

on how effectively the interferometric measurement can be applied to the localization problem. In order to achieve more precise localization, the effect of error propagation has to be controlled.

3.6 Summary

Interferometric ranging has been recently proposed as a viable measurement type to solve the localization problem in sensor networks. However, in addition to constraints imposed on the hardware devices, interferometric ranging also imposes new challenges to the algorithmic design of localization. In this chapter, we formally proved that localization using interferometric ranging is an NP-Complete problem. Compared to heuristics on direct RSSI or TOA ranging (both of which problems are also NP-Complete) it can be argued that heuristics on localization using interferometric ranging are even more difficult (however this added difficulty is polynomial). Whereas each direct ranging measurement is a function of two locations, the interferometric measurement is a function of four. Thus, localization using interferometric ranging requires a considerably larger set of measurements.

Previous interferometric localization algorithms try to use optimization to obtain a global solution either by using a genetic algorithm approach [55] or by reducing the search space with additional RSSI readings [67]. The difficulty of the problem still limits their solutions to smaller networks (16 nodes in [55] and 25 nodes in [67]). The iterative algorithm proposed in this paper allows networks of larger size to be localized using interferometric ranging. However, our simulations indicate that error propagation can be a potentially significant problem. In order to localize large networks using interferometric ranging from a small set of anchors, future localization algorithms need to find a way to effectively limit the error propagation.

CHAPTER 4

LINK LONGEVITY ESTIMATION

4.1 Problem Definition

In mobile ad hoc networks, links are established on the fly as mobile nodes are moving in and out of the transmission ranges of each other. This node mobility not only invalidates previous localization results as demonstrated in Chapter 2, it also infers a constantly changing network connectivity graph. Due to the distributed nature of ad hoc networks, a route between two arbitrary nodes is likely a combination of multiple links over several intermediate nodes. The selection of the sequence of links between nodes is the task of the ad hoc routing protocol. Due to link failures caused by node movement, routes can be disrupted while in service. Loss of links can invalidate routing entries, which in turn can cause undesired latencies in packet delivery. Availability of a good estimation of link longevity between neighboring nodes could permit the selection of a more stable route, thus enabling a better enforcement of quality of service (QoS) provisioning contracts.

In this chapter, we consider the problem of predicting the expecting link uptime between two mobile nodes. We refer to this as the link longevity problem. This problem can be seen as a special case of localization among two neighboring nodes. Instead of estimating the absolute node locations in this case, we want to estimate the *relative* mobility between the nodes. In particular, the link longevity does not depend on the absolute mobility but the relative mobility between two nodes. For the link longevity problem, we propose three link longevity estimators that could be embedded into mobile nodes. In our approach, link longevity estimates do not require knowledge of the mobility patterns/models of nodes. The foundation of all three designs lies in extended Kalman filtering, where a linear process model is implemented to represent the state transition, while a non-linear measurement model is included to account for the received signal strength indication measurements. We provide the mathematical background for our estimators and their input/output vectors, and show various performance metrics using extensive simulations. We conclude that our filters provide good estimates for the remaining up-time of wireless links.

4.2 Related Works

There are a number of previous works addressing the link longevity problem based on probability models [57, 37]. Their contribution lies in defining node mobility models and performing a mathematical analysis on these models to quantify statistical properties of the longevity of links. In [57], the link longevity is measured as the probability of the link remaining available for a time t. The probabilistic model is then used as the basis to form and maintain clusters within ad hoc networks to maximize the cluster stability. The probability calculation assumes that all nodes move according to an epoch based movement pattern. For each node, the movement history is divided into a sequence of epochs. Within each epoch, the node moves at a randomly selected (but constant) direction and velocity. The pure probability based model is extended in [36] by incorporating a measurement model. First, the link longevity, in terms of remaining time (t) in which the link remains available, is measured under the assumption that the nodes maintain their constant velocities. A probability model is then applied to calculate the probability of the link availability at time t by considering the cases of temporal velocities.

A non-probabilistic solution is proposed in [79], in which the link longevity estimation is used to measure the stability of the entire route so that a handoff can be triggered in anticipation. The longevity estimation relies on a Global Positioning System (GPS) receiver at each node to provide the location and velocity data. The remaining connection time between two nodes is calculated from the GPS data assuming that the nodes maintain their headings and speeds. If GPS data is not available other measurement-based models could be used. In [54], the location, velocity and acceleration of a mobile node are estimated by measuring the received signal strength indication (RSSI) from multiple base stations in a cellular network. The measured power levels are fed into a Kalman filter. Since base station locations are assumed to be well-known in a cellular network, mobile nodes can use them as reference points.

RSSI measurements can also be used to estimate the location of mobile nodes in ad hoc networks. In [58], the authors propose a method of propagating the location data from nodes that are GPS-equipped (beacons). Other non-GPS nodes (regular nodes) can then deduce their distances to the GPS nodes by measuring the RSSI from neighboring nodes. The actual location can then be calculated using trilateration methods from the distance information. The method is further improved in [60] by assuming non-GPS nodes are equipped with devices that measure the incoming signal directions. The directional information allows the receivers to obtain the angle of arrival (AoA) of the signal thus allowing more accurate location estimates.

In this chapter we present a measurement-based approach to the link longevity problem. Unlike the probability-based solutions that rely on a particular mobility model [57, 36], we propose an estimator (to be embedded into the mobile nodes) that operates regardless of the mobility model. The estimator's basis is a Kalman filter [40] used to estimate the relative location of two nodes based solely on simple signal properties like RSSI. The information obtained from the filter is then used to derive the expected time the link will remain available. A major advantage of Kalman filters is that they can quickly and efficiently compute estimates. Therefore, they are particularly suited for ad hoc networks due to the potentially limited computing power of mobile nodes. Our solution is similar to the estimator in [54], but it is designed to work in the distributed ad hoc environment, where all nodes are mobile and the relative location needs to be determined. Furthermore, unlike the measurement-based location estimators [58, 60], our solution is geared towards estimating the link longevity in time t instead of the exact node locations. This means that a node will not only need to determine the locations of other nodes but also a change in their movement patterns.

The remainder of the chapter is organized as follows. Section 4.3 describes three different link longevity estimator designs including detailed process models used by the Kalman filters. Section 4.4 presents the simulation results and compares the three estimators under different movement and noise conditions. Section 4.5 concludes this chapter.

4.3 Kalman Estimator Designs

This section outlines three different designs of an ad hoc link longevity estimator. Each of the designs is based on extended Kalman filters [75]. In all three designs, a linear process model is implemented to represent the state transition. The process model assumes that nodes maintain their current direction and velocity between each state update. Corrections to the errors in the process model are made via the measurement model of the filter. The need for the extended Kalman filter arises due to the measurement model's inherent non-linearity (radio signal power levels are not linear with the propagation distance). Each filter design is unique: they rely on input from different types of sensors and/or keep their state information in different variables in the process model.

4.3.1 Design I

Our first filter design assumes that both the incoming signal strength and its direction are observable at mobile nodes. Using a signal propagation model, the distance and direction to a transmitting node can be estimated from the received signal strength (RSSI sensor) and the direction of the received signal (direction sensor). Given the measurements the estimator is able to track the relative location and velocity in both x and y directions. Therefore, the state variable of the filter at time t is (x_t, y_t, v_x, v_y), where x_t and y_t represent the relative displacement at time t, while v_x and v_y represent the relative velocity. By further assuming that nodes move at a constant velocity, a new state can be derived from the previous state using a linear process update function $x_{t+1} = x_t + v_x\Delta t$ and $y_{t+1} = y_t + v_y\Delta t$, where Δt is the observation interval. The state transition matrix is therefore

$$\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Given a pre-determined transmission range, the expected time of link termination can be calculated from the state variables.

Hardware-wise, sensors that measure RSSI are widely available for mobile devices. Indeed most off-the-shelf technologies implicitly provide such information (e.g., most Wi/Fi cards provide RSSI). However, sensors that measure the signal direction require much more sophistication to antenna design (which cannot be easily justified for location estimation only). Nevertheless, our first model provides a simple yet precise estimator design to which subsequent designs can be referenced and compared.

4.3.2 Design II

Our second design relaxes some of the previous assumptions by requiring only the availability of an RSSI sensor, thus only relying on easy-to-measure properties. In this case, it is not possible for the receiver to estimate the relative position and velocity of the sender. Yet, the relative distance to the sender and the rate of the distance change over time Δt can be estimated. Figure 4.1 explains how the rate of distance change is related to the location and velocity of the nodes. Let us consider two nodes, n_0 and n_1 that are moving with absolute velocity (v_{x_0}, v_{y_0}) and (v_{x_1}, v_{y_1}) in some Descartes coordinate system. Let D_t be the distance between n_0 and n_1 at time t, and R_t be the rate of the distance change at t. Here, we define R_t to be positive if the nodes are moving away from each other and negative otherwise. In general, let θ_x and θ_y be the angles from the displacement line (*away* from the other node) moving counter-clockwise to the x and y components of the absolute velocity. R_t can be calculated as the sum of the portions from all four velocity segments as $R_t = v_{x_0}cos(\theta_{x_0}) + v_{y_0}cos(\theta_{y_0}) + v_{x_1}cos(\theta_{x_1}) + v_{y_1}cos(\theta_{y_1})$.

For the Kalman filter design, we denote the state variables to be (D_t, R_t). Note that in reality R_t is not constant over time (even at a constant velocity) since both nodes move simultaneously. Nevertheless, the process model maintains that R is a constant, resulting in the following process update function: $D_{t+1} = D_t + R_t\Delta t$. The state transition matrix is therefore

$$\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

To assess the error that this assumption incurs, it is important to note that as the change to R_t decreases D_t increases. Because a link longevity estimate is more useful when the nodes are further away (i.e., when D_t is large), this assumption, though incorrect, should have minimal impact on the overall results (as it was verified by our simulation studies).

4.3.3 Design III

The third design - similarly to the second - requires only the availability of RSSI sensors at nodes. However, in this case we derive the new distance estimates based on a history of previous estimates. Since our underlying assumption is that the nodes move

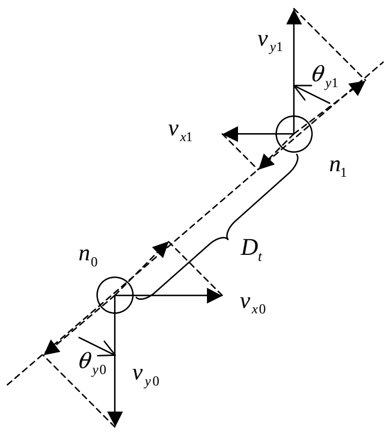


Figure 4.1. Calculating R_t from velocity segments.

at constant speeds and headings, it is possible to derive the new distance estimate based on previous estimates. For the filter based on k previous states, where $k > 1$, the state space consists of the following variables: $(D^2_{t-k}$, …, D^2_{t-1}, D^2_t , $S^2)$. The variables D_{t-k}, …, D_{t-1}, D_t are the distance estimates at time $t-k, \ldots, t-1,$ and t, respectively, while S is the *relative speed* between the two nodes. Given the relative velocity as v_x and v_y, S is simply $\sqrt{v_x^2 + v_y^2}$. The new state is derived under the assumption that v_x and v_y are constant.

The new state is calculated from the previous state as follows. A new estimate $\hat{D}^2_{t+1}$ can be calculated from each historical reading $D^2_{t-i} (1 \leq i \leq k)$ as well as the latest reading D^2_t. Since relative velocities are assumed to be constant, we can envision the receiver as stationary while the sender is moving at S to a fixed direction. From Figure 4.2, let A be the stationary location of the receiver, and B, C and D be the relative location of the sender at time $t-i$, t and $t+1$ respectively. Thus, $D_{t-i} = AB$, $D_t = AC$, and $\hat{D}_{t+1} = AD$. BC and CD can be derived from the assumption that S is constant and that the filter runs with a period of Δt. By solving the triangulation in Figure 4.2, a new estimate of $\hat{D}^2_{t+1}$ can be found as

$$\hat{D}^2_{t+1} = -\frac{D^2_{t-i}}{i} + \frac{i+1}{i} D^2_t + (i+i) S^2 \Delta t^2$$

Note that the above estimate $\hat{D}^2_{t+1}$ is obtained from D^2_t and a single previous reading of D^2_{t-i}. We can then repeat the above calculation for all $D^2_{t-i} (1 \leq i \leq k)$ and average a total number of k estimates. The new distance estimate D^2_{t+1} is therefore the following:

$$D^2_{t+1} = \sum_{i=0}^{k-1} \frac{-D^2_{t-k+i}}{k(k-i)} + \frac{D^2_t}{k} \cdot \sum_{i=0}^{k-1} \frac{k+1-i}{k-i} + \frac{S^2 \Delta t^2}{k} \cdot \sum_{i=0}^{k-1} (k+1-i)$$

The above equation translates to following state transition matrix:

$$\begin{bmatrix} 0 & 1 & 0 & \ldots\ldots & 0 \\ \vdots & 0 & 1 & \ldots\ldots & \vdots \\ \vdots & \vdots & & & \vdots \\ T_{k-1} & \ldots\ldots & T_0 & V & W \\ 0 & \ldots\ldots & & 0 & 1 \end{bmatrix}$$

Here, $T_i = \frac{-1}{k(k-i)}$, $V = \frac{1}{k} \cdot \sum_{i=0}^{k-1} \frac{k+1-i}{k-i}$, and $W = \frac{\Delta t^2}{k} \cdot \sum_{i=0}^{k-1} (k+1-i)$.

By averaging the estimates from all k previous values, we expect the estimator to be less prone to abnormal sensor readings. Meanwhile, it will take longer to adapt to the change of the movement pattern. Variable k provides this design with an additional parameter (besides the Kalman filter variances) to adjust the filter to the environment. As verified by simulations, the third design is expected to show better performance with a larger k when the node movement pattern is more predictable and the sensor reading is noisier.

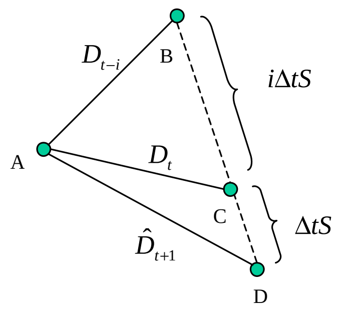


Figure 4.2. Calculating D_{t+1}^2 based on previous calculation of D_t^2 and D_{t-i}^2.

4.4 Simulation Results

To evaluate the three link longevity estimators, we have implemented them in our own C++ based discrete event simulation engine. All our simulations involve two mobile nodes moving within a 2000m side-length square. The node movement model is based on the epoch model used in [57] with the following properties:

1. The entire movement path of the node is defined by a sequence of "epochs," i.e., $(e_1, e_2, \cdots, e_n)$.
2. The duration of each epoch is I.I.D. exponentially distributed with a mean of $1/\lambda$.
3. Within each epoch, the node moves at a constant velocity.
4. At the end of each epoch, nodes randomly select a new velocity vector. The direction of the movement is I.I.D. randomly chosen from a uniform distribution between 0 and 2π. The absolute value of the velocity is I.I.D. normally distributed with a mean μ of and a variance of σ^2.

Note that since immobile nodes do not cause any difficulties for our model, we eliminate the idle time between epochs from the original model, and thus the nodes are always on the move. For our simulations, we use $\mu = 10m/s$ and $\sigma^2 = 10$.

When a node hits the border of the square, its bounces back at the same angle. Of the two mobile nodes simulated, one is designated as the sender and the other as the receiver. The sender continuously transmits signals, and the receiver continuously monitors the incoming signal. The signal propagation model is given by $p = c \cdot d^{-2}$, in which the power of the received signal p is inversely proportional to the second power of the distance d. Here, c is an arbitrary constant. When the received signal power p is below a threshold p_{min}, it is considered too weak to be captured by the receiver thus the link breaks. For our simulations, we select $c = 10^6$ and $p_{min} = 1$. Note that the c and p_{min} selection does not affect the overall simulation results, as long as the same values are used in the observation model of the filters. In fact, the same can be said about all other signal propagation models - all we require is a model that represents the receiving power as a function of distance. Given our signal model, the threshold $p_{min} = 1$ translates to a transmission range of 1000m when noise is not considered. To estimate the time when the link will be cut the receiver processes the sensor data every 0.1 seconds. The sensor data is then feed into all three estimators simultaneously to obtain their estimations for comparison.

Noise is incorporated based on the noise model in [62]. The model considers the fact that radio signals hardly ever propagate in line-of-sight paths. Instead, they tend to bounce off from nearby structures along the way due to multipath fading and far field scattering. The actual distance of signal propagation at time t is given by $d_t = d'_t + m_t$, where d'_t is the geometric distance between the two nodes and m_t the extra distance covered due to signal reflection. m_t is recursively defined as

$$m_t = m_{t-1} + P_0 N(0, {\sigma_0}^2) + P_1 N(0, {\sigma_1}^2),$$

where P_0 and P_1 are the probability of the whether or not the signal bounces off a different structure, and N is a Gaussian distribution with a zero mean. Since the distance can change more drastically when the signal bounces off a different (than before) structure,

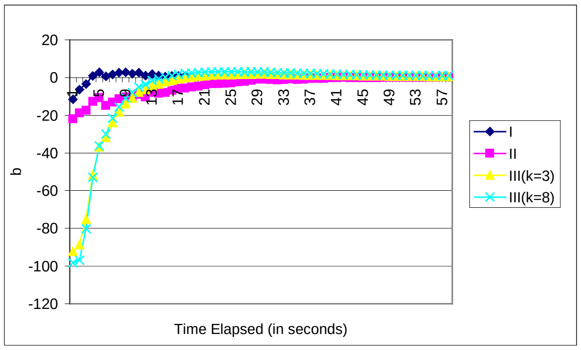


Figure 4.3. Convergence of estimation error.

it can be assumed that ${\sigma_0}^2 >> {\sigma_1}^2$. Furthermore, m_t should be non-negative for all t. For our simulations, we use $P_0 = 0.1$ and $P_1 = 0.9$.

4.4.1 Estimator Convergence

Figure 4.3 shows how the link longevity estimator of the three different designs converges in a typical scenario when the sender and receiver are pulling away from each other at constant velocities. The figure shows the error of the estimations as the filters analyze the incoming signals. Clearly, the first design converges the quickest due to its extra signal direction sensor. Of the estimators relying solely on the RSSI sensor, the estimator based on the third design takes longer to converge than the estimator of the second design but it provides better estimates (due to the availability of previous estimates). Of the two different versions of the third design, the one that relies more on historical estimates ($k = 8$) converges a little quicker than the other which relies on less available previous estimates ($k = 3$).

4.4.2 Effect of Node Movement

Figure 4.4 and 4.5 show the effect of node mobility on the estimators by varying the mean of the epoch duration ($1/\lambda$). For this simulation, we keep the variances of the movement model constant at ${\sigma_0}^2 = 50$ and ${\sigma_1}^2 = 5$. We let the sender and receiver run for a duration of 10^6 seconds of simulation time. For each simulation run, we obtain the results by varying $1/\lambda$ from 20 to 200. As the mean $(1/\lambda)$ epoch length increases, the node movement becomes more predictable, and thus the link longevity becomes easier to predict. To qualify the results, we define an estimation to be *acceptable* if it is within the range of $+/-10$ seconds when the link actually breaks. Furthermore, we denote $T_{success}$ to be the time *before* the actual link breakage when the different estimators converge to the acceptable range. There are also cases where the estimators never manage to produce acceptable estimations before link breakages. As such, we let $P_{success}$ be the probability that an acceptable estimation can be obtained in average. Essentially, $T_{success}$ indicates how *good* the estimations are, and $P_{success}$ indicates how *fast* they are obtained.

Figure 4.4 and 4.5 show the effect of node movement on $T_{success}$ and $P_{success}$. The figures indicate that the extra direction sensor in the first design greatly contributes to its superior performance. The second is not far behind from the first in terms of $P_{success}$. The two versions of the third design do not have a great performance in terms of $P_{success}$. However, they outperform the first design in $T_{success}$, indicating that in general the estimators based on the third design take longer to adjust to the movement updates. The figures also indicate that the more sensor data is processed the better the precision of the estimate will be. Furthermore, the gap in terms of $P_{success}$ between $k = 8$ and $k = 3$ increases slightly indicating that a larger k is more appropriate when node movement is more predictable.

4.4.3 Effect of Signal Noise

The effect of sensor noise on $T_{success}$ and $P_{success}$ is captured in Figures 4.6 and 4.7. For these simulations we have set the mean of the epoch duration to $1/\lambda = 50$. We then

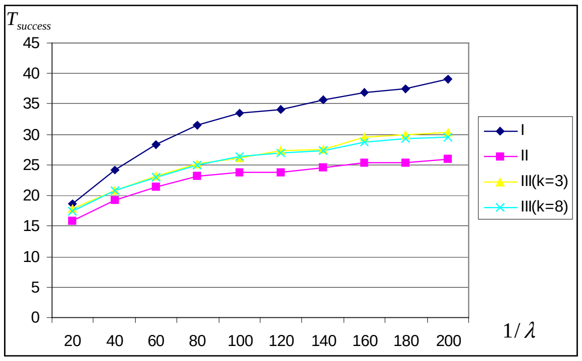


Figure 4.4. The effect of node movement on $T_{success}$.

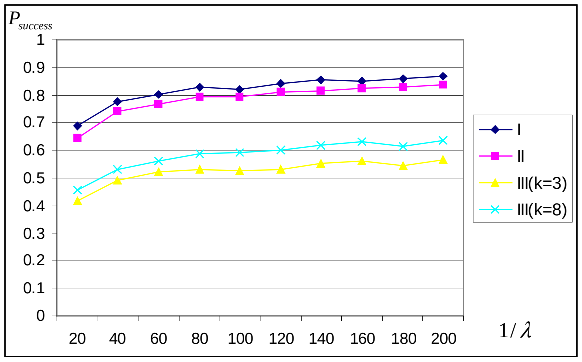


Figure 4.5. The effect of node movement on $P_{success}$.

vary the variance ${\sigma_0}^2$ of the noise model from 10 to 90. The other variance ${\sigma_1}^2$ is set to one tenth of the current ${\sigma_0}^2$ value.

Figure 4.6 shows that the quality of the estimations varies little as the noise increases. Meanwhile, Figure 4.7 indicates that it takes longer for the estimators to converge to the acceptable range with increasing noise. Since all of the estimator designs are based on Kalman filters, it is not surprising that they are rather resilient to noise, even though our noise model is not Gaussian. However, noise does have an effect on the estimators in that it takes longer to obtain acceptable estimations in a noisier environment. Comparing the two cases of the third design (where $k = 3$ and $k = 8$) we can observe that a larger k is better suited for a noisier environment.

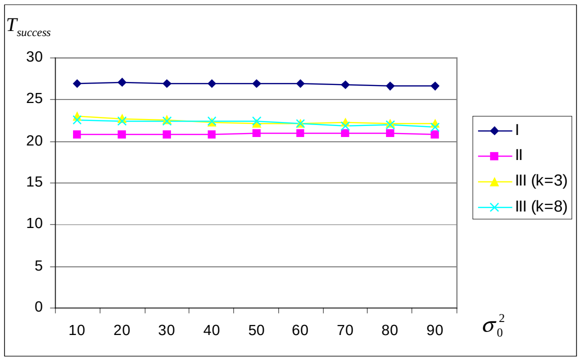


Figure 4.6. The effect of noise on $T_{success}$.

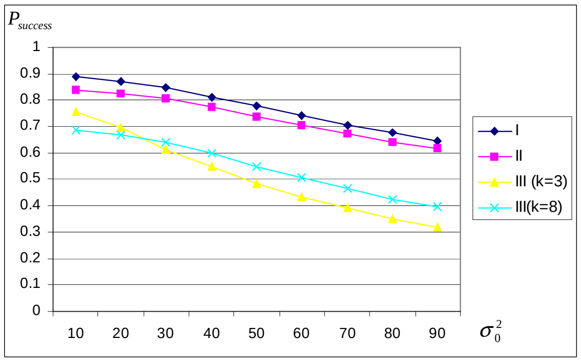


Figure 4.7. The effect of noise on $P_{success}$.

4.5 Summary

This chapter has presented three different estimators that predict the link longevity in ad hoc mobile networks. These extended Kalman filter based estimators obtain their estimates by tracking node movement using RSSI measurements. Since Kalman filters are known to be light-weighted, these estimators are especially suitable for mobile nodes with strict resource constraints. Our simulations demonstrate that all estimators are capable of producing useable estimations, even though their performance is subject to the underlying predictability of node movement and sensor noise. The simulations also indicate that estimators fed with a radio signal direction sensor provide only slightly better estimates than estimators based solely on RSSI readings.

CHAPTER 5

DYNAMIC BEACON DEPLOYMENT FOR LOCALIZATION

In Chapter 2, the theoretical analysis of the general ad hoc localization problem (GAHLP) based on graph theory illustrated the difficulty of the localization problem when the network is sparse. In particular, when a ranging measurement is used, the network connectivity induced graph needs to be redundantly rigid and triconnected in order for the network to be 100% localized [29, 4]. For a sparse network such criteria might not be easily satisfied especially considering sensor networks are commonly deployed in random fashion. Thus, a question emerges asking what is the optimal way of deploying additional regular nodes and/or beacons such that the entire network can be localized. We refer to this problem as network deployment problem (NDP). In practice, it is often impossible to strategically deploy sensor nodes one-by-one with localizability in mind. For instance, sensor networks for military applications could be mass deployed from airplanes. In such case, one of the methods to obtain location information is by positioning a number of beacons *after* the sensors have been deployed. Since beacons are often more expensive, there is a strong economical incentive to minimize their numbers.

5.1 Problem Definition

In this chapter, we consider the beacon deployment problem according to the following model. We assume that a certain number of unlocalized sensors already reside somewhere in a deployment area. The beacon deployment (BD) problem deals with the question of optimally deploying beacons to localize those nodes. The beacons are assumed to be deployed one at a time in an *online* fashion. After a beacon is deployed, we assume that it can identify the sensor nodes that it covers. This model can be realized in a number of ways. For instance, a mobile robot equipped with a GPS receiver can be sent to cover the deployment area. A computer onboard the robot can run an online algorithm to solve the beacon deployment problem. When this algorithm decides that a beacon should be deployed at a certain location, it will instruct the robot to go to that location and broadcast its coordinates there. Then, the algorithm decides on the next beacon location and moves there, repeating the above process. Since all beacon locations are served by a single mobile robot, the beacon deployment algorithm has the complete history of the previously deployed beacon locations as well as information on the number of nodes covered by each beacon to help it make the next decision. Alternatively, we can use a data dissemination method such as HEAP [10], where the beacon deployment algorithm is hosted on a centralized node at a fixed location. The result of each beacon deployment is disseminated through a virtual tree imposed on the sensor network, until it reaches the centralized node at the root of the tree.

In particular, we initially consider a simplified version of the beacon deployment (BD) problem in which we deploy beacons to the initial network such that all nodes in the network can be localized *directly* from the beacons. By *directly*, we mean that the nodes are to be localized from the beacons 1-hop away. No multihop information is used in the localization. The goal is to minimize the number of beacons while keeping the network localized. Later on, we will relax the 1-hop requirement to allow localized sensor nodes to serve as virtual beacons to help localizing other nodes.

5.2 Related Works

Localization has been studied in various contexts. In this section, we concentrate on the problem of localization by deploying beacons.

Based on the network localizability analysis described in Chapter 2, an incremental node deployment strategy is proposed in [25]. With ranging measurement as the edge weight, the algorithm first finds redundantly rigid and triconnected subgraphs. Each of the subgraphs found can then be localized by deploying three beacons. The problem of this algorithm lies in that it is not really designed as a distributed algorithm. In particular, the method to identify the subgraphs is strictly centralized.

An adaptive beacon placement algorithm is proposed in [9], where beacons are deployed sequentially based on empirically data of the perceived localization error. The perceived error is obtained by observations among neighboring beacons. When beacons are densely deployed over an area, the perceived localization error among them should reflect the error characteristics of the terrain or environment, which can then be applied to estimate the localization error of the actual sensor nodes. In [9] two beacon deployment algorithms are proposed, MAX and GRID, both of which deploy the beacons to locations where the estimated error is perceived to be at its maximum. The two algorithms differ in the size of the area they consider. Further work in [10] provides a framework to realize the adaptive algorithm in the real world by proposing a distributed algorithm to disseminate the perceived localization error into a centralized location.

The authors of [35] describe a beacon deployment strategy with a different objective: to minimize the number of deployed beacons (cameras) to localize mobile objects. They ask the question: “given the workspace and an error threshold, what is the minimum number, and placement of cameras so that the error in localization is less than the threshold at every point in the workspace? [35]”. Two cameras, by the means of angle measurement, are needed to localize nodes. The goal is to minimize the number of cameras so that the overall error is below a threshold. While this method is not designed to localize sensors, it introduces the notion of explicit error thresholds as part of the design.

5.3 Our Contribution

In this chapter, we first show some computational complexity results of the beacon deployment (BD). We then propose a number of approximation methods to solve the problem. The approximation algorithms are presented in both offline and online versions. The offline version solves a simplified (although still a theoretically hard) problem by assuming that sensor node locations are known. The offline version is studied because it serves as the performance guideline to the online version. The online version deals with the realistic scenario of progressively deploying beacons to localize sensors of unknown locations. Finally, using simulations, we show that the results of the proposed offline and online algorithms, in terms of the total number of beacons deployed, is reasonably close to the optimum especially at higher sensor densities. More precisely, our contributions are:

1. We show that the general optimal beacon deployment problem without geometric constraints is NP-Complete (when the number of sensor nodes is the factor).
2. We propose an approximation algorithm based on the set cover problem to solve the beacon deployment problem that requires each sensor node to be covered by at least k beacons. We show that this algorithm has a logarithmic approximation ratio to the optimal solution.
3. We extend the approximation algorithm to explicitly deal with issues that impact localization accuracy such as availability of beacons, measurement errors, and relative position of beacons. Our method minimizes the number of beacons (i.e., minimizes equipment cost) under the accuracy constraint explicitly specified as a threshold to the Cramer Rao Bounds (CRB).

Our work on the online version of the deployment method is closest to that of [9], both of which select the beacon location to minimize the localization uncertainty. The primary difference is that while the work in [9] measures the uncertainty using empirical data, we choose to use the more theoretically sound concept of error bounds. Furthermore, the algorithms in [9] use connectivity measurements (i.e., Centroid method) only. Our methods work with ranging measurements based on RSSI or ToA, and can be easily extended to support AoA.

5.4 Complexity of BD

The *offline* version of the BD problem asks for the minimum number of beacons needed to localize a given network. The network is "given" in the sense that we know all node locations. The offline version seems pointless since by definition all nodes have already been localized. But we study this problem so that we have a base for comparison (an optimal performer) for the online algorithms.

The condition for a node being localizable depends on the measurement type. Consider the trivial case where all nodes have ranging and signal direction measurement capacity; here one measurement from any beacon (i.e., one neighboring beacon) is sufficient to localize a node. If nodes have capabilities for only signal direction measurements, two beacon neighbors are required. Three beacon neighbors are required if only ranging measurements are available. In general, let the number of beacons required for a particular measurement type be k; we denote BD(k) as the version of beacon deployment problem where sensor nodes require measurements from k beacons, i.e., each sensor should have at least k beacon-neighbors for the sensor network to be localizable. In the following discussions we are going to assume that the deployment space for beacons is defined by a grid, i.e., we discretize the location space beacons can occupy from a continuous search space, so that we now deal with a combinatoric optimization problem. We denote V_{sensor} as the set of sensors, and V_{grid} as the set of all grid locations.

BD(k) is clearly NP, since when given a solution candidate (a certificate), a polynomial time algorithm can validate this solution in $O(k \cdot |V_{sensor}| \cdot |V_{grid}|)$. BD($k$), in fact, is a special case of the multiset covering (MSC) problem with special geometric constraints. We define MSC(k) as the problem of finding the minimum number of sets that would cover each element at least k times. The MSC is a general case of the set covering (SC) problem, which is known to be NP-Complete [22]. Here, each vertex $v_{grid} \in V_{grid}$ can be viewed as a set containing the vertices in V_{sensor} such that there is an edge between v_{grid} and $v_{sensor} \in V_{sensor}$. Thus, the trivial case when $k = 1$ is just the set covering problem. The following theorem shows that the problem remains NP-Complete when $k > 1$ (intuitively, $k > 1$ is a harder problem).

Theorem 5.4.1. *The offline version of the MSC(k) problem is NP-Complete.*

Proof. We superpose a square (n by n) grid over the deployment area of the sensors; we assume that beacons can only reside at grid points. We use V_{sensor} to denote the set of sensor nodes in the grid. Note that sensors can reside anywhere in the deployment square and do not have to reside at the grid points. We then construct an augmented network $\hat{G} = (\{V_{sensor}, V_{grid}\}, \hat{E})$, where V_{grid} contains *all* possible deployment locations of beacons. The augmented edge set $\hat{E}$ contains all the edges formed between V_{grid} and V_{sensor}. An edge exists in $\hat{E}$ if two nodes are within the transmission range. The solution to MSC is the minimum subset of V_{grid} such that V_{sensor} can be localized.

To show MSC(k) is NP-Complete when $k \geq 1$, we reduce it from the NP-Complete vertex covering (VC) problem [22]. Given an instance of the vertex covering problem on a graph $G_{VC} = (V_{VC}, E_{VC})$, we create an instance of MSC(k) as follows:

1. For every vertex in V_{VC}, create a vertex in V_{grid} of the MSC(k) instance.
2. For every edge in E_{VC} of the vertex covering instance, create a vertex in V_{sensor} of the MSC(k) instance, and connect this vertex to the two corresponding vertices in V_{grid} to which this edge is adjacent in the original vertex covering instance.

3. Create an additional $k-1$ vertices in V_{grid} and fully connect them to every vertex in V_{sensor}. We denote these additional vertices as $\{v'_1, v'_2, \ldots, v'_{k-1}\}$.

Figure 5.1 illustrates such reduction. This reduction can be done in polynomial $O(k \cdot |V_{VC}| \cdot |E_{VC}|)$ time. We need to show that a solution to the VC instance implies a solution to the reduced MSC(k) instance, and vice versa. This can be validated by the claim that for any non-trivial case where $|V_{sensor}| >> k$, the solution to the reduced MSC(k) instance must contain all the additional vertices $\{v'_1, v'_2, \ldots, v'_{k-1}\}$. Each vertex in $V_{sensor} = \{e_1, e_2, ...\}$ connects to two vertices in $\{v_1, v_2, ...\} \subseteq V_{grid}$. For a vertex in V_{sensor} to be covered by k vertices in , at least $k-2$ connections need to come from the additional vertices of $\{v'_1, v'_2, \ldots, v'_{k-1}\}$. If there are exactly $k-2$ vertices in $\{v'_1, v'_2, \ldots, v'_{k-1}\}$ used, *all* of the vertices in $\{v_1, v_2, ...\}$ will have to be used to give k covers. Since we are considering the nontrivial case where $|V_{sensor}| >> k$, it is always better to use *all* of the $\{v'_1, v'_2, \ldots, v'_{k-1}\}$ vertices to give $k-1$ covers and find the remaining 1 cover from $\{v_1, v_2, ...\}$. Thus, a solution to the original vertex cover problem where the number of vertex cover is s would result in a solution of $s+k-1$ to the reduced MSC(k). Conversely, a solution of s' to the reduced MSC(k) would result in a solution of $s'-k+1$ to the vertex cover problem. Since vertex cover problem is NP-Complete, so is MSC(k) .

□

Note that the above complexity result is given for the multiset covering (MSC) problem, which does not consider the geometric constraints. The beacon deployment (BD) problem is a special case of MSC in which the cover sets have to follow constraints on the euclidean distance. When $k=1$, the BD problem is also referred as the geometric disk covering problem (GDC), which is known to be NP-Complete [39]. Thus, although no yet proven, it is expected that BD(k) is also NP-Complete when $k>1$.

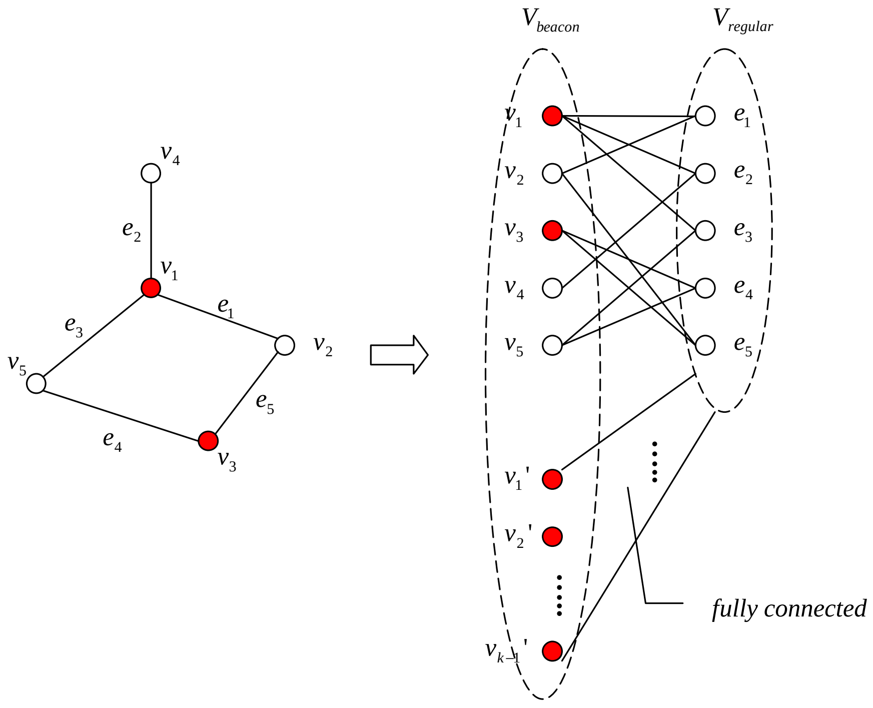


Figure 5.1. Reduction from vertex covering to MSC(k).

5.5 Offline Approximation of BD(k)

For the set covering (SC) problem, there is a well-known greedy algorithm with approximation ratio to the optimal solution of $O(\ln(|X|))$, where $|X|$ is the size of the superset [17]. At each step, the greedy algorithm selects a set that covers the maximum number of uncovered members in the superset X, until the entire superset has been covered. The greedy algorithm performs close to the lower bound of the approximation ratio, as it has been shown in [19] that the unweighted set cover problem cannot be approximated within a factor $(1-\epsilon)\ln(|X|)$, for any $\epsilon > 0$.

Based on the greedy algorithm for set covering, we can give a pseudo code for the BD(k) offline greedy algorithm as shown in Algorithm 4. At each step, the algorithm selects a beacon location, g_{max}, from the available grid locations that covers the most sensor nodes that have not been k-covered:

$$g_{max} \leftarrow \arg\max_{g \in U} |\{s : p_s < k, s \dashv g\}|$$

The algorithm stops once all sensor nodes have been k-covered and returns the set of beacon locations selected. In terms of complexity, the outer loop at line 7 runs at most $O(min(k \cdot |V_{sensor}|, |V_{grid}|))$ times, and the loop statements between line 8 and 13 can be implemented to run in $O(|V_{sensor}| \cdot |V_{grid}|)$. Thus, the overall runtime complexity is $O(min(k \cdot |V_{sensor}|, |V_{grid}|) \cdot |V_{sensor}| \cdot |V_{grid}|)$

Even through BD(k) is a generalization of the set covering problem where $k > 1$, the same logarithmic approximation ratio can be obtained, as stated by the following theorem.

Algorithm 4 Greedy Offline Approximation of BD(k)

```
1: $V_{sensor}$: set of all sensors
2: $V_{grid}$: set of all grid locations
3: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
4: initialize $p_s \leftarrow 0$ for all $s \in V_{sensor}$
5: $U \leftarrow V_{grid}$
6: $C \leftarrow \emptyset$
7: **while** $\exists s \in V_{sensor} : p_e < k$ **do**
8:     select $g_{max} \leftarrow \arg\max_{g \in U} |\{s : p_s < k, s \dashv g\}|$
9:     **for all** $s : p_s < k, s \dashv g_{max}$ **do**
10:        $p_s \leftarrow p_s + 1$
11:    **end for**
12:    $U \leftarrow U - \{g_{max}\}$
13:    $C \leftarrow C \bigcup \{g_{max}\}$
14: **end while**
15: **return** $C$
```

Theorem 5.5.1. *The greedy offline approximation algorithm of BD(k) has an approximation ratio of* $\ln(|V_{sensor}|) + 1$.

Proof. Let the cost of each selection of $g \in V_{grid}$ be 1. We distribute that cost into the $s \in V_{sensor}$ that g covers at each step. Let c_s^j be the distributed cost when s is covered for the jth time. In particular, at the jth time, if s is covered by g, then

$$c_s^j = \frac{1}{r_g}$$

where r_g is the reward in selecting g at this step. Based on the algorithm, the reward r_g is calculated by counting the number of vertices $s : p_s < k$ that it covers.

Since all vertices $s \in V_{sensor}$ are covered k times, the total cost of the greedy algorithm $|C|$ is therefore

$$\begin{aligned} |C| &= \sum_{s \in V_{sensor}} \sum_{j=0}^{k-1} c_s^j \\ &\leq k \cdot \sum_{s \in V_{sensor}} c_s^{k-1} \end{aligned}$$

The inequality holds here because the greedy choice always chooses the selection with the maximum reward and thus ensures a non-decreasing order of the cost c_s^j for a particular s as j increases. The maximum cost of covering s always occurs when it is covered at the kth, the final, time (when $j = k - 1$).

The optimal algorithm C^* also needs to cover each s at least k times. Thus,

$$|C| \leq \sum_{g^* \in C^*} \sum_{s \dashv g^*} c_s^{k-1} \tag{5.1}$$

where $s \dashv g^*$ means that s is covered by g^*.

Let u_i be the size of the set $\{s : s \dashv g^*, p_s < k-1\}$ after $g_1, g_2, \ldots, g_i$ has been selected by the greedy algorithm. Thus, u_0 contains all s covered by g^* since $p_s = 0$ initially. As the greedy algorithm makes its selections to cover more vertices covered by g^* , p_s increases. Thus u_i decreases until it becomes 0 at step $i = l$. Expressing the cost in terms of u_i, we have

$$\sum_{s \dashv g^*} c_s^{k-1} = \sum_{i=1}^{l} (u_{i-1} - u_i) \cdot \frac{1}{r_{g_i}}$$

where $u_{i-1} - u_i$ is the number of nodes $s \dashv g^*$ that are covered by g_i for the kth time, and $\frac{1}{r_{g_i}}$ is the distributed cost of the selection g_i. Because of the greedy choice, g_i is chosen at step i because it gives the maximum reward. Meanwhile, the reward of selecting g^* at this step is exactly u_{i-1}. Thus, $u_{i-1} \leq r_{g_i}$. Consequently, we have

$$\begin{aligned}
\sum_{s \dashv g^*} c_s^{k-1} &\leq \sum_{i=1}^{l} (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}} \\
&\leq \sum_{i=1}^{l} (H(u_{i-1}) - H(u_i)) \\
&\leq H(u_0) - H(u_l) \\
&\leq H(u_0) - H(0) \\
&\leq H(u_0) \\
&\leq H(|\{s : s \dashv g^*\}|) \qquad (5.2)
\end{aligned}$$

where $H(n)$ is the harmonic number of the integer n.

Combining inequality of 5.1 and 5.2, we have

$$\begin{aligned} |C| &\leq \sum_{g^* \in C^*} \sum_{s \dashv g^*} c_s^{k-1} \\ &\leq \sum_{g^* \in C^*} H(|\{s : s \dashv g^*\}|) \\ &\leq |C^*| \cdot H(\max(|\{s : s \dashv g^*\}|)) \end{aligned}$$

Since the maximum number of sensor nodes s any grid location g can cover is $|V_{sensor}|$, we have

$$\begin{aligned} |C| &\leq |C^*| \cdot H(|V_{sensor}|) \\ &\leq |C^*| \cdot (\ln(|V_{sensor}|) - 1) \end{aligned}$$

which gives us an approximation ratio of $\ln(|V_{sensor}|) - 1$. $\square$

Figure 5.2(a) depicts a simple scenario illustrating the operation of the offline greedy algorithm. The scenario consists of two sensor nodes a and b, shown as circles, residing in a deployment area of a 5x5 grid. The task is to deploy the minimum number of beacons, shown as squares, so that each sensor node connects to at least three beacons ($k = 3$). The coverage radius of the beacons is 3 (i.e., a beacon at $(0, 0)$ will cover sensors at $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 0)$, $(1, 1)$, $(1, 2)$, $(2, 0)$, $(2, 1)$, $(2, 2)$ and $(3, 0)$). As seen in Figure 5.2(a), a sequence of three beacons are deployed by the offline greedy algorithm. Initially, the node potentials are $p_a = 0$, $p_b = 0$, and the set U contains all 25 grid points. The first beacon is selected at grid $(2, 1)$ as a greedy choice because the beacon would cover both sensor nodes. Evidently, there is more than one grid point that covers both, but we simply choose one of them. After the first beacon location is selected, the potentials of both sensor nodes increase to 1 ($p_a = 1$ and $p_b = 1$). The selected location $(2, 1)$ is

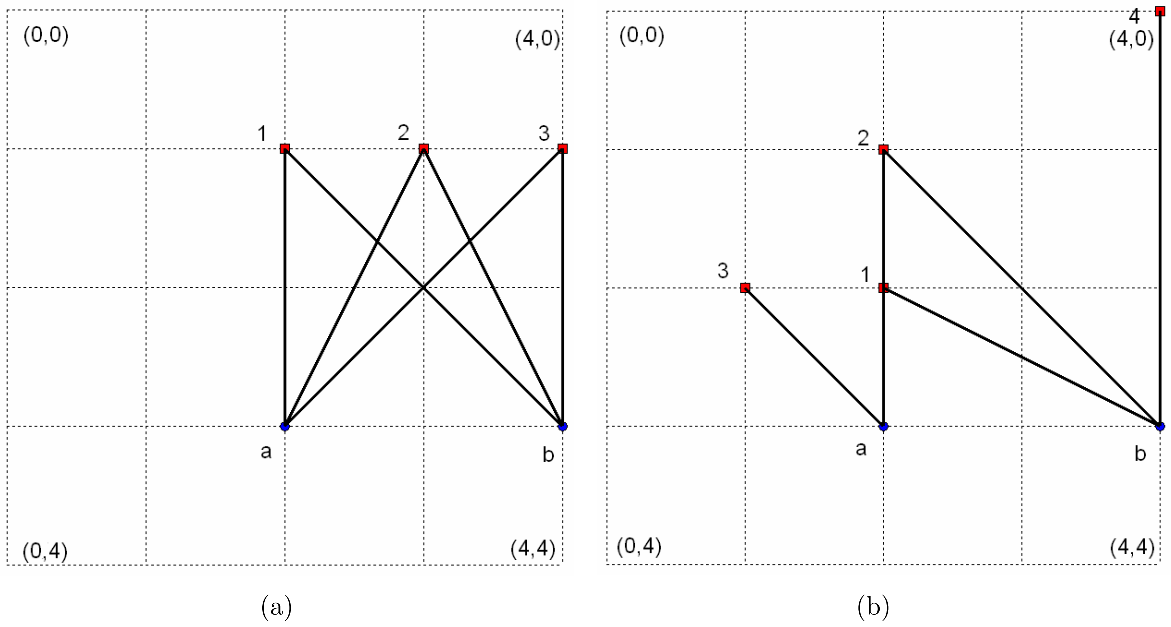


Figure 5.2. Example of the greedy approximation of BD(k). The figure shows two cases: (a) offline and (b) online.

removed from the set U and added to the cover set C. Since both potentials (p_a and p_b) are not k yet, a second beacon is inserted and its location is greedily selected from the remaining grid points in U. The algorithm continues until both potentials reach k (when both sensor nodes are k covered).

5.6 Online Approximation of BD(k)

The offline algorithm of solving BD(k) assumes that the locations of the sensor nodes are known, which essentially defeats the purpose of the beacon deployment. However, as shown even this simplified problem is NP-Complete when trying to minimize the number of beacons. In this section, we remove the node position knowledge assumption. We consider this problem to be an *online* problem in the sense that we have to select the next beacon location based on the feedback (i.e., how many sensor nodes have been covered) of the previous beacon locations. (In the offline version the feedback result is known *before* the beacon is deployed.) Our basic assumption of the online version is that sensor nodes are uniformly distributed over the entire deployment area. We also assume knowledge on the population of the sensor nodes. Based on these assumptions, we can design a greedy online deployment algorithm, Algorithm 5, that selects beacon locations by maximizing the coverage probability.

The online greedy algorithm maintains $p_{s,g}$ as the probability of the sensor node s residing at grid location g. Assuming that sensor nodes are uniformly distributed, $p_{s,g}$ is initialized to $1/|V_{grid}|$, where $|V_{grid}|$ is the size of the potential beacon locations. To select the next beacon's location, the algorithm first sums up all the neighborhood probabilities $p_{s,g}$ for each grid point; a greedy choice is then made to pick the grid point with the maximum overall $p_{s,g}$ value:

$$g_{max} \leftarrow \arg\max_{g \in U} \sum_{g' \dashv g} \sum_{s} p_{s,g'}$$

In reality, the greedy choice might not be the best one, since it is unforeseeable how many sensor nodes it will cover until the beacon is deployed. However, as more beacons are deployed and feedback is received, the probability distribution $p_{s,g}$ will be refined to reflect the actual sensor location. For each of the sensor nodes that are already covered k times, the $p_{s,g}$ value is set to 0, indicating that the k-cover condition of this node is satisfied. For all other sensor nodes, the procedure *AdjustProbability* (Algorithm 6) is called to adjust the $p_{s,g}$ values based on the latest beacon choice. If a sensor node is not within the range of the newest beacon, the algorithm raises the probability of the grid points that are within the range of the beacon compared to those that are not. If a sensor node is within the range of the latest beacon, the reverse is done. The purpose of *AdjustProbability* is to reduce the scope of the possible location for each sensor node and thus help the algorithm to make better choices in subsequent iterations. The algorithm stops when all the probabilities ($p_{s,g}$) become zero, i.e., when all sensor nodes are k covered. Figure 5.2(b) depicts a possible outcome of the online greedy algorithm given the same scenario of localizing two sensor nodes in a 5 by 5 grid and $k = 3$. The shown instance of the online algorithm took four beacons (one more than the offline version) to complete the k-cover.

Algorithm 5 Greedy Online Approximation of BD(k)

```
1: $V_{sensor}$: set of all sensors
2: $V_{grid}$: set of all grid locations
3: $p_{s,g}$: the probability of the sensor $s$ residing close the grid location $g$
4: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
5: initialize $p_{s,g} \leftarrow 1/|V_{grid}|$ for all $s \in V_{sensor}$ and $g \in V_{grid}$
6: $U \leftarrow V_{grid}$
7: $C \leftarrow \emptyset$
8: while $\exists s \in V_{sensor}, g \in V_{grid} : p_{s,g} > 0$ do
9:    select $g_{max} \leftarrow \arg\max_{g \in U} \sum_{g' \dashv g} \sum_{s} p_{s,g'}$
10:   for all $s \in V_{sensor}$ do
11:      if $s$ is covered by $k$ beacons then
12:         $p_{s,g} \leftarrow 0$ for all $g$
13:      else
14:         $AdjustProbability(p, s, g_{max})$
15:      end if
16:   end for
17:   $U \leftarrow U - \{g_{max}\}$
18:   $C \leftarrow C \bigcup \{g_{max}\}$
19: end while
20: return $C$
```

The initialization phase of Algorithm 5 has a runtime of $O(|V_{sensor}| \cdot |V_{grid}|)$. The condition of the outer loop at line 8 can be validated in $O(|V_{sensor}| \cdot |V_{grid}|)$. Within the outer loop, the greedy selection step at line 9 has a runtime of $O(|V_{sensor}| \cdot |V_{grid}|)$, and the *for* loop at line 10 also has a runtime of $O(|V_{sensor}| \cdot |V_{grid}|)$ (the runtime of the function *AdjustProbability* is $O(|V_{grid}|)$. Thus, the total run time of each iteration of the outer loop is $O(|V_{sensor}| \cdot |V_{grid}|)$. In the worst case, the algorithm will cover all grid points, in

Algorithm 6 *AdjustProbability*(p, s, g)

```
1: $sum \leftarrow 0$
2: if $s \dashv g$ then
3:     for all $g' : g' \not\dashv g$ do
4:         $sum \leftarrow sum + p_{s,g'}$
5:         $p_{s,g'} \leftarrow 0$
6:     end for
7:     $I \leftarrow$ set of all $g' : g' \dashv g, p_{s,g'} > 0$
8:     for all $g' \in I$ do
9:         $p_{s,g'} \leftarrow p_{s,g'} + sum/|I|$
10:    end for
11: else
12:    for all $g' : g' \dashv g$ do
13:        $sum \leftarrow sum + p_{s,g'}$
14:        $p_{s,g'} \leftarrow 0$
15:    end for
16:    $I \leftarrow$ set of all $g' : g' \not\dashv g, p_{s,g'} > 0$
17:    for all $g' \in I$ do
18:        $p_{s,g'} \leftarrow p_{s,g'} + sum/|I|$
19:    end for
20: end if
```

which case the outer loop will execute in $O(|V_{grid}|)$ time. Overall, the worse case runtime complexity of the algorithm is $O(|V_{sensor}| \cdot |V_{grid}|^2)$.

One note on the actual implementation: although the online greedy algorithm assumes that the sensor nodes are uniformly distributed, it is equally effective if the spatial distribution of sensors is known before the deployment process starts. One can simply adjust the initial probability distribution to match the perceived location distribution. If it is expected that the sensor nodes are more likely to concentrate on a smaller area in the deployment region then the probability distribution should be adjusted to give higher probability to the locations within this area. A more accurate initial probability distribution will enable the online algorithm to make better decisions.

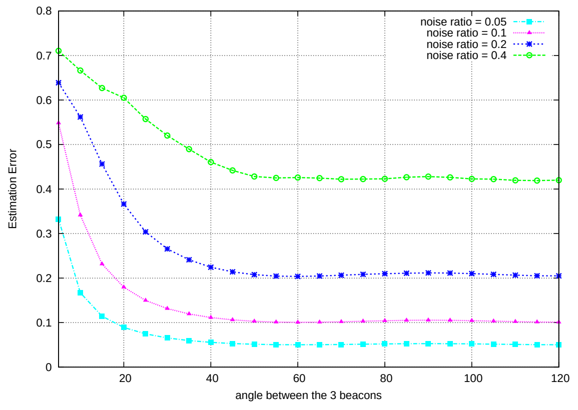
5.7 Measurement Quality

Our prior analysis makes the simple assumption that for a particular measurement model (i.e., ranging, angle, or both), k measurements are required to localize the node. Each measurement is treated equally. This assumption is valid only when two conditions are satisfied: i) the measurements are not collinear and ii) the measurements are perfect without any noise. While the first condition can be validated during the localization process after the beacons are deployed, it is much more difficult to satisfy the second condition. After all, measurement noise is not the exception but the reality. In reality, however, noisy measurements imply that not all measurements should be treated equally. Certain measurement might contain more useful information than others. Take localization using ranging for instance, the following two factors impact the localization accuracy due to noisy measurements:

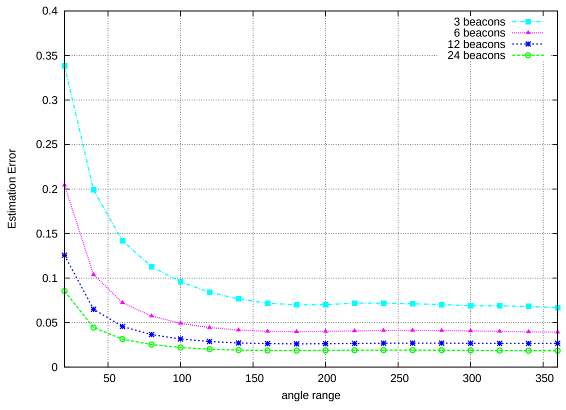
1. Distance between the sender and the receiver. With ranging using RSSI, the amount of noise increases as the distance increases.
2. Angle between the measurements. Intuitively, we would prefer the ranging measurements to come from more "spread out" beacons.

We have performed some experiments to better understand the impact of the second factor. In these experiments, we intend to localize a node by measuring the noisy ranging readings from beacons located at a unit circle (eliminating the effect of beacon distance). By varying the noise ratio and the angle between each measurement at each run, we collect the error of the estimated location after the trilateration. All our results have a 95% confidence that the error between the mean and the sample average is less than 5% of the mean.

In Figure 5.3(a), results are shown when exactly three beacons are used with a fixed pair-wise angle (estimation error is measured on the unit of the unit disk graph). For instance, if the angle is set to be 20°, each of the three ranging readings are 20°



(a)



(b)

Figure 5.3. Impact of the beacon location on ranging based localization. The figure shows the cases when (a) setting the pair-wise angle and (b) limiting the angle range.

apart. The experiment shows that increasing the pair-wise angle greatly helps reducing the localization error until the angle exceeds about 60° after which the error stabilizes. Thus, for this particular scenario, localization error can be minimized if a pair-wise angle greater than 60° can be used.

The experiment shown in Figure 5.3(b) sets an upper limit on the angle range allowed during localization. For instance, if the angle limit is set to 20°, all readings have to come from beacons with pairwise angles between 0° and 20°. Here, we fix the noise ratio to 0.05. The figure shows that the localization error decreases as the angle limit increases until about 180° after which the error stabilizes. For the same angle limit, increasing the number of beacons also decreases the error. However, in general the number of beacons needs to be doubled in order to reduce the error by half. Furthermore, it would be more economical to spread out the beacons. For example, to obtain a localization error less than 0.1 unit in our simulations, only three beacons are needed if the angle range is greater than 100°, whereas six beacons are needed if the range is limited to 40°.

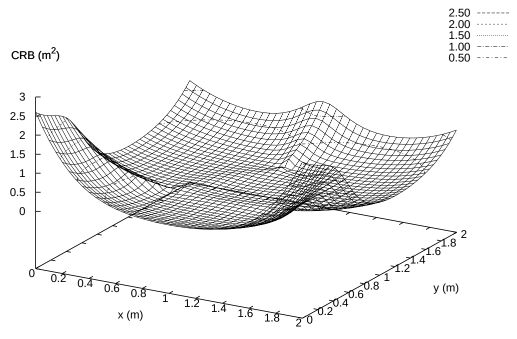
5.8 Applying Cramer Rao Bounds (CRB)

Chapter 2 introduced the concept of Cramer Rao Bounds. In the context of localization, the Cramer Rao Bound (CRB) [38, 65, 66] is often used to mathematically qualify the lower bound on the localization error. However, the CRB is not specifically designed for the localization problem. It is in fact a more general tool to bound the covariance of any unbiased estimator given any measurement data. For the localization problem, the CRB is a function of the following: i) the relative locations of the sensor nodes and the beacons, ii) the measurement model determined by the measurement type, and iii) the noise model characterizing the terrain and the environment. Previous works have derived the CRB formulas for a number of measurement types including RSSI ranging [66], TOA ranging [66], AoA [65], and a hybrid of RSSI and TOA [13]. Note that the actual localization algorithm being used has no impact on the CRB. Thus, the CRB is essentially a bound determined solely by the particular localization scenario. It gives an indication of how difficult a particular localization scenario is and what is the best *any* localization algorithm can do given the scenario.

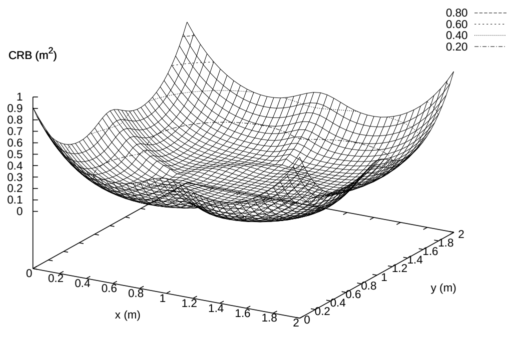
Figure 5.4 shows the Cramer Rao Bound (CRB) of a single sensor node based on the RSSI ranging measurement. Here, we use the signal propagation model and noise model given in [66]. In particular, we assume that the range measurement noise is Gaussian with a constant variance introduced by shadowing. The received signal strength from a beacon location i to a sensor node j that are $d_{i,j}$m apart is therefore

$$N(P_0 - 10n_p log_{10}(d_{i,j}/d_0), \sigma_{dB}^2)$$

where P_0 is the received signal strength at a reference distance d_0, and we use $d_0 = 1$m. n_p is the path loss exponent that is environment-dependent, and σ_{dB}^2 is the constant variance introduced by the shadowing. As in [66], we choose $n_p = 1$ and $\sigma_{dB}^2 = 1.7$.



(a)



(b)

Figure 5.4. The Cramer Rao Bound (CRB) of RSSI measurements. The figure shows two cases: (a) 3 beacons and (b) 4 beacons.

In Figure 5.4 the CRB was measured within a 2 by 2 unit square, with the beacons placed at an inner circle of radius 0.5 unit centered at $(1, 1)$. In the case of three beacons (Figure 5.4(a)), the beacons are placed with equal angles at $(1+\cos(0)/2, 1+\sin(0)/2)$, $(1+\cos(2/3\pi)/2, 1+\sin(2/3\pi)/2)$, and $(1+\cos(4/3\pi)/2, 1+\sin(4/3\pi)/2)$. In the case of four beacons (Figure 5.4(b)), they are placed at the four corners: $(3/2, 1)$, $(1, 3/2)$, $(1/2, 1)$, and $(1, 1/2)$. In either case, it is clear from Figure 5.4 that locations within the beacon deployment radius have lower CRB. When the location exceeds the radius, the CRB progressively becomes greater, which is consistent with the intuition that it is more difficult to localize the nodes outside the convex hull formed by the beacons. Furthermore, compared the case of three beacons with that of four, it is clear that increasing the number of beacons reduces the overall CRB.

5.9 CRB-Based Approximation of BD

We raised the issue that the localization accuracy is impacted by the beacon location and the measurement noise. Depending on the particular beacon location, the actual localization results can vary greatly due to the measurement noise. Thus, in practice it might not be sufficient to only require that each sensor node is covered by k beacons ($k = 3$ for 2-dimensional ranging localization), since the localization accuracy varies depending on how those k beacons are placed. In this section, we address the accuracy issue in the BD problem. In particular, we modify the offline and online approximation algorithms presented earlier to use the CRB as the stoppage and greedy selection criteria.

Algorithm 7 shows the pseudo code of the CRB modified version of the greedy offline algorithm. The algorithm will continue to deploy beacons until the CRBs of all sensor nodes are reduced below a given threshold crb_T. For each sensor node s, we track the location of beacons that cover s. The CRB of s can then be calculated from the beacon locations and stored in c_s. Let function $CRB(s)$ return the CRB of the node s with the already deployed beacons. We also define a function called $CRB^P(s, g)$ that returns the potential of the new CRB with a new beacon g added. (The CRBs of initially unlocalized sensor nodes are set to infinity.) At each iteration, a greedy choice, g_{max}, is made to pick the next beacon location that will cause the maximum reduction to the CRB of the sensor nodes, i.e., maximizing $CRB^P(s, g) - CRB(s)$:

$$g_{max} \leftarrow \arg\max_{g \in U} \sum_{s:s \dashv g} (CRB^P(s, g) - CRB(g))$$

The pseudo code of the modified CRB based online solution is shown in Algorithm 8. For the online version, since the exact location of each sensor node s is unknown, we present it as a probability distribution over the entire deployment area, where $p_{s,g}$ is the probability of node s residing close to the grid location g. Furthermore, the CRB calculation needs to based on a perceived instead of the exact sensor location. Thus, if a sensor s is perceived to be located at g', $CRB_{g'}(s)$ returns the CRB of s at g'. The greedy beacon selection is then based on maximizing the sum of $p_{s,g'} \cdot (CRB^P_{g'}(s, g) - CRB_{g'}(s))$ for all locations g' within the range of the beacon at location g. In other words, the

Algorithm 7 Greedy Offline Approximation of BD(crb_T)

```
1: $V_{sensor}$: set of all sensors
2: $V_{grid}$: set of all grid locations
3: $CRB(s)$: CRB of the sensor $s$
4: $CRB^P(s, g)$: Potential CRB of the sensor $s$ when adding a beacon broadcast at $g$
5: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
6: initialize $c_s \leftarrow \inf$ for all $s \in V_{sensor}$
7: $U \leftarrow V_{grid}$
8: $C \leftarrow \emptyset$
9: while $\exists s \in V_{sensor} : CRB(s) > crb_T$ do
10:    select $g_{max} \leftarrow \arg\max_{g \in U} \sum_{s:s \dashv g} (CRB^P(s, g) - CRB(g))$
11:    for all $s : s \dashv g$ do
12:       re-calculate $CRB(s)$
13:    end for
14:    $U \leftarrow U - \{g_{max}\}$
15:    $C \leftarrow C \bigcup \{g_{max}\}$
16: end while
17: return $C$
```

greedy location choice is the one that maximizes the overall CRB reduction of all sensor nodes considering their probabilities of residing within the beacon range:

$$g_{max} \leftarrow \arg\max_{g \in U} \sum_{g' \dashv g} \sum_{s} p_{s,g'} \cdot (CRB^P_{g'}(s, g) - CRB_{g'}(s))$$

Figure 5.5 shows results of the offline and online greedy algorithms of BD(crb_T) running on a simple scenario of two sensor nodes (with the same RSSI parameters as in [66]). The CRB threshold is set to $crb_T = 1$. The sample run for the offline version selected three beacons while the online version picked four. Compared to results in Figure 5.2, which were obtained using k-cover ($k = 3$) as the objective, it is clear that the CRB requirement provides a better spread for the beacon locations. In the case of the offline version, the three beacons selected in Figure 5.2(a) constitute a spread of 45° angle for both sensor nodes, while the beacons in Figure 5.5(a) constitute a spread of 90° for node a and 63° for node b. Wider angle spread is also observed in the online

Algorithm 8 Greedy Online Approximation of BD(crb_T)

```
1: $V_{sensor}$: set of all sensors
2: $V_{grid}$: set of all grid locations
3: $CRB_{g'}(s)$: CRB of the sensor $s$ assuming it's located at grid location $g'$
4: $CRB^P_{g'}(s, g)$: Potential CRB of the sensor $s$, assumed to be located at $g'$, when adding
   a beacon broadcast at $g$
5: $p_{s,g}$: the probability of the sensor $s$ residing close the grid location $g$
6: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
7: initialize $p_{s,g} \leftarrow 1/|V_{grid}|$ for all $s \in V_{sensor}$ and $g \in V_{grid}$
8: $U \leftarrow V_{grid}$
9: $C \leftarrow \emptyset$
10: **while** $\exists s \in V_{sensor}, g \in V_{grid} : p_{s,g} > 0, CRB_g(s) > crb_T$ **do**
11:    select $g_{max} \leftarrow \arg\max_{g \in U} \sum_{g' \dashv g} \sum_s p_{s,g'} \cdot (CRB^P_{g'}(s, g) - CRB_{g'}(s))$
12:    **for all** $s$ **do**
13:       **if** $s \dashv g_{max}$ **then**
14:          re-calculate $CRB_{g'}(s)$ for all $g' \dashv g_{max}, p_{s,g'} > 0$
15:       **end if**
16:       $AdjustProbability(p, s, g_{max})$
17:    **end for**
18:    $U \leftarrow U - \{g_{max}\}$
19:    $C \leftarrow C \bigcup \{g_{max}\}$
20: **end while**
21: **return** $C$
```

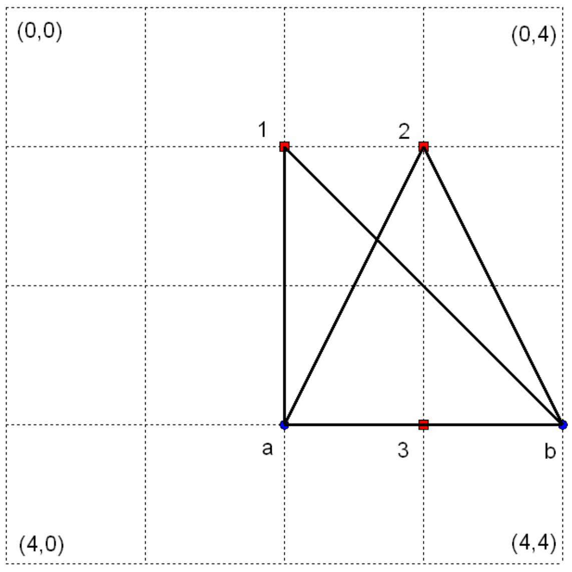
version. The better spread of the beacon locations dictated by the CRB requirement would ultimately result in more accurate localization.

In practice, the threshold crb_T is a system configurable variable, which should be adjusted according to the desired localization accuracy. As expected, a lower crb_T would result in better localization accuracy at the expense of more deployed beacons. The choice of crb_T also affects the runtime complexity. In the worst case, when crb_T is close to zero, the outer loops of both offline and online versions would continue executing until all potential beacon locations are used, i.e., $O(|V_{grid}|)$. For the offline version in Algorithm 7, line 10 can be implemented in $O(|V_{sensor}| \cdot |V_{grid}|)$. The inner loop at line 11 can also be implemented in $O(|V_{sensor}| \cdot |V_{grid}|)$. Thus, the worst case runtime complexity of the offline version is $O(|V_{sensor}| \cdot |V_{grid}|^2)$. The worst case runtime complexity of the online version (Algorithm 8) can be similarly derived to be $O(|V_{sensor}| \cdot |V_{grid}|^3)$.

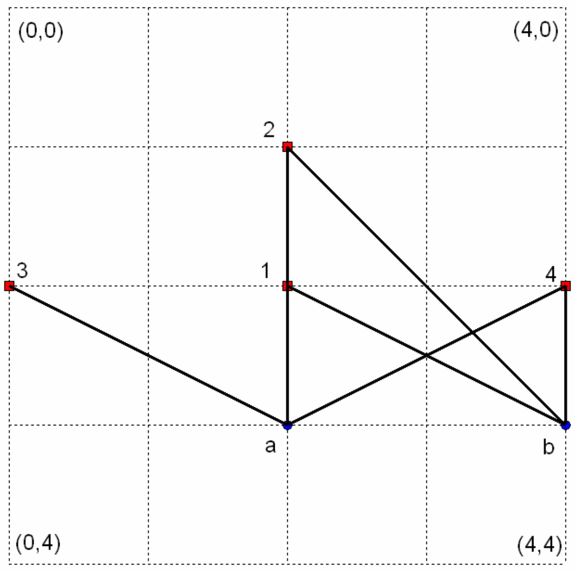
The BD(crb_T) approximation methods select the beacon locations based on the perceived CRB reduction, so there remains the question on how to obtain accurate CRB measurements. The CRB calculation depends on the range or angle measurement model, whose perceived error characteristics are explicitly included into the calculation. However, the error characteristics (such as shadowing and multi-path fading in RSSI) are generally environment-specific. The CRB measurements will suffer in the case when it is not possible to estimate those error characteristics ahead of time, or when the error characteristics vary greatly within the deployment area due to vastly heterogeneous terrain. In such cases, the empirical method introduced in [9] could be used, which allows the beacons to estimate the error characteristics by observing each other. Based on the generally valid assumption that environment-specific characteristics tend to be similar around the same neighborhood, the estimated error characteristics between the beacons should be similar to those between beacons and sensor nodes.

5.10 Indirect BD Approximation: BD-E(k) and BD-E(crb_T)

The BD approximation methods introduced earlier require all sensors to be localized by the neighboring beacons *directly*. By abiding such requirement, we are able to deduce the complexity and approximation bound of this difficult problem. However, in practice it is often desirable to relax such requirement and allow sensor nodes to localize *indirectly* by observing other sensor nodes. In many cases, a sensor node does not need direct measurement from the beacons. Instead, the measurements from the sensor nodes will constitute a certain rigidity requirement that helps to localize the node. While enforcing



(a)



(b)

Figure 5.5. Example of the greedy approximation of BD(crb_T). The figure shows two cases: (a) offline and (b) online.

a complete rigid network [18, 25, 29] is beyond the scope of this work, we would extend our BD approximation methods to allow such indirect localization.

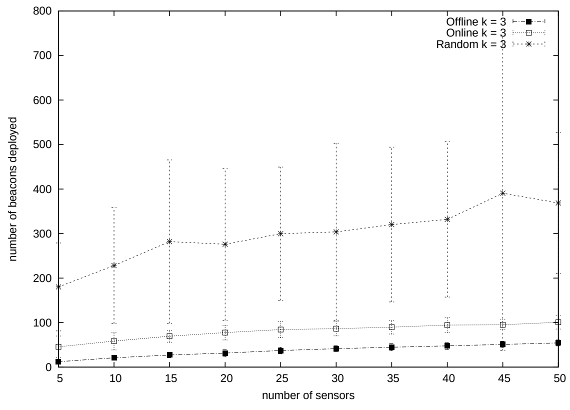
In particular, we make a simple modification to the proposed BD approximation methods to treat any localized sensors as pseudo-beacons, and thus their estimated locations can then serve as new beacon locations for other neighboring sensors. We call those methods BD-E(k) and BD-E(crb_T) (E for Extended) accordingly. Intuitively, compared to BD(k) and BD(crb_T), BD-E(k) and BD-E(crb_T) would reduce the number of beacons required to localize the network when the sensors are densely populated. However, the localization error can increase since the error of the localized nodes can now propagate to other nodes.

5.11 Simulation Results

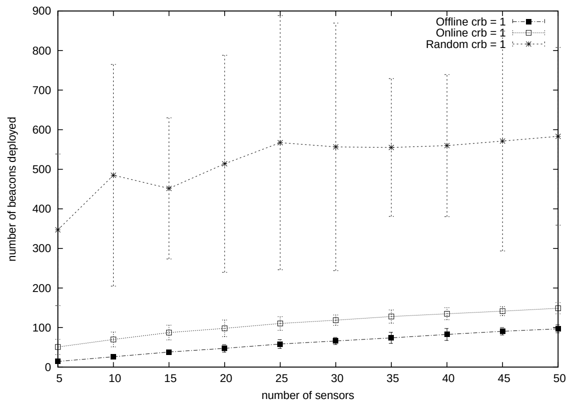
We evaluate the performance of the various BD approximation methods using computer simulations. The simulation environment consists of a number of sensor nodes uniformly distributed over a deployment area of a 50 by 50 grid. For every deployment scenario, we run each of the approximation methods and collect the number of beacons needed to localize all sensor nodes. The beacons are assumed to have a transmission range of 5 grid radius. For BD(k), we assume a 2-D localization which requires $k = 3$. In the case of BD(crb_T), we assume the range is obtained via RSSI readings with a constant Gaussian noise added. We use the same signal propagation and noise model as in [66], with a reference distance $d_0 = 1m$, a path loss exponent $n_p = 1$, and a noise variance $\sigma_{dB}^2 = 1.7$. In addition to BD(k) and BD(crb_T), we also include the results of a simple random deployment algorithm, in which the beacon locations are i.i.d. uniform randomly selected until the stoppage criteria ($k = 3$ or $CRB < crb_T$) is satisfied. The simulation results presented in the figures are the average of 30 different runs with the 95% confidence interval shown as the vertical error bars.

5.11.1 Network Size

Figure 5.6 compares the total number of beacons deployed for each of the approximation methods while varying the number of sensor nodes residing in the deployment area. Figure 5.6(a) shows the results of using $k = 3$ as the stop condition, and Figure 5.6(b) uses CRB threshold of $crb_T = 1$. As expected, in both the k and the crb_T cases, the online approximation methods deploy more beacons than the offline versions. However, the difference does not seem to increase as the number of sensor nodes increases. This indicates that the online version would be preferable with a denser deployment area, since the relative difference between the online and offline version becomes smaller as the number of sensor nodes increases.



(a)



(b)

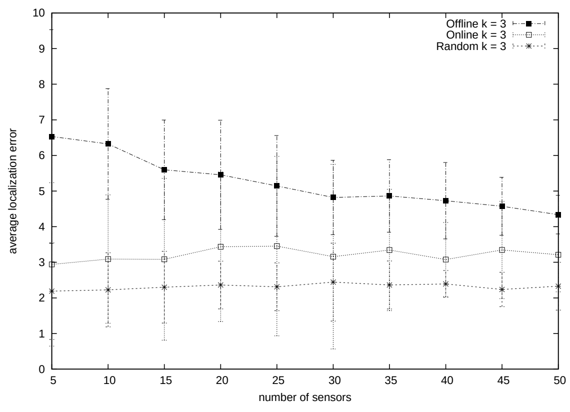
Figure 5.6. Number of beacons deployed using (a) BD(k) and (b) BD(crb_T).

5.11.2 Localization Error

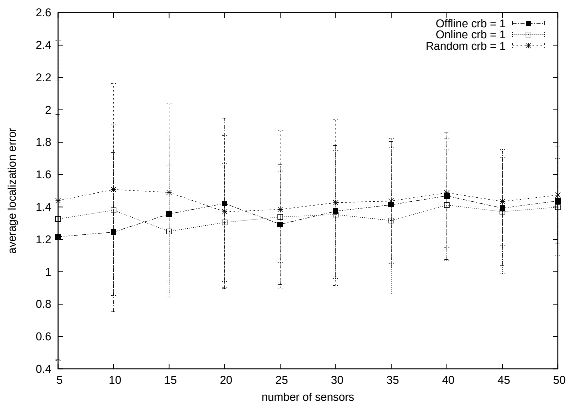
Different BD approximation methods produce different beacon locations, which ultimately affect the localization accuracy. The actual localization error of each method is shown in Figure 5.7. In the k-cover version (Figure 5.7(a)), the offline algorithm produces the greatest amount of the error, and the random algorithm produces the least amount of error. This is a direct result of the number of beacons deployed, since the k-cover version does not consider how the relative beacon location impacts the localization error. The random algorithm deploys the largest number of beacons, thus producing the least amount of localization error. In the crb_T version (Figure 5.7(b)), however, our algorithm explicitly considers the localization error as a part of the deployment strategy.

5.11.3 CRB Threshold crb_T

To demonstrate the effect of the threshold crb_T in beacon deployment, we use a scenario of 20 sensor nodes in the deployment area while varying crb_T from 0.1 to 1.2. The number of beacons deployed by the random algorithm, and offline and online version of the BD(crb_T) approximation methods are shown in Figure 5.8(a). Note that as the crb_T



(a)



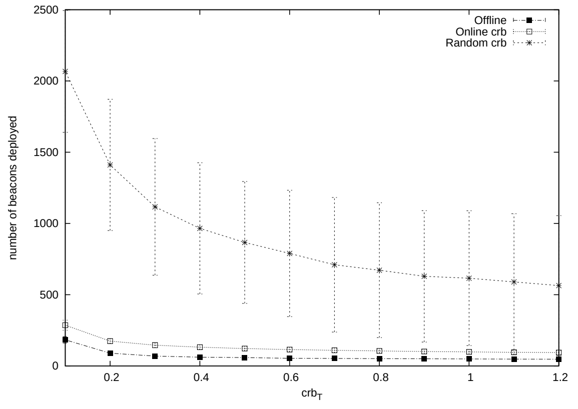
(b)

Figure 5.7. Localization error using (a) BD(k) and (b) BD(crb_T).

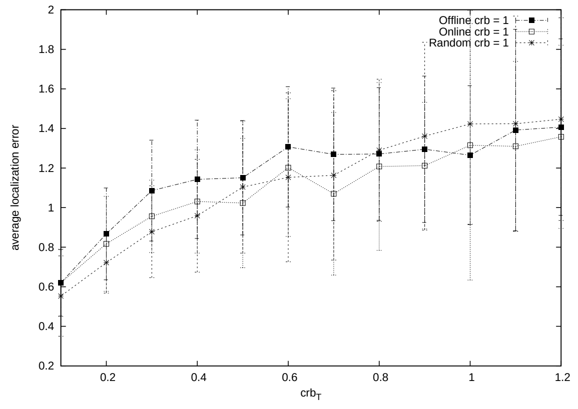
threshold decreases, the increase in the beacon population surpasses linearity. As shown in Figure 5.8(b), the actual localization error decreases as the crb_T lowered. Furthermore, Figure 5.8(b) shows that the localization error stays relatively constant for all three deployment algorithms, even though the random algorithm deploys a significantly higher number of beacons than online and offline BD(crb_T). This result demonstrates that the selected threshold crb_T effectively controls the localization error. *Thus, by selecting the appropriate crb_T, we can balance the trade-off between the localization accuracy and the overhead associated with the number of beacons deployed.*

5.11.4 Indirect Localization

Figure 5.9 shows the number of beacons deployed by the indirect localization methods BD-E(k) and BD-E(crb_T) compared to the direct methods. As expected, the indirect methods reduce the number of deployed beacons as those localized sensors now serve as pseudo-beacons. As the sensors become more densely populated, more nodes can share their localization information with their neighbors. Thus, more beacons can be reduced with increasing the number of sensors as the figure indicates.

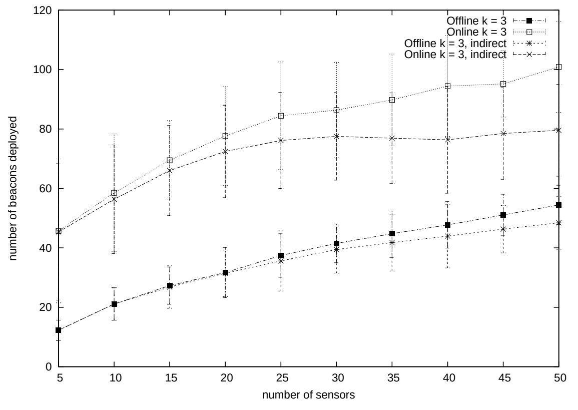


(a)

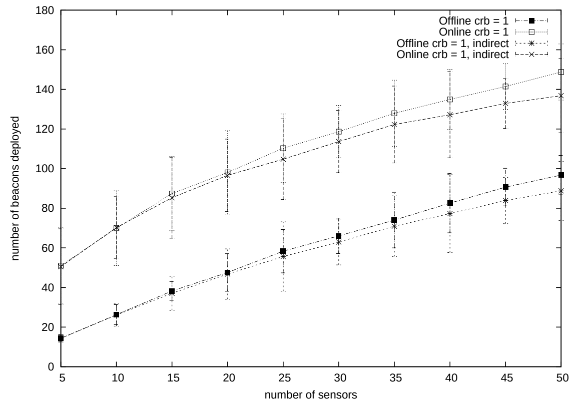


(b) Average localization error

Figure 5.8. Results of varying the threshold crb_T in BD(crb_T). The figure shows (a) number of beacons deployed and (b) average localization error.



(a)



(b)

Figure 5.9. Number of beacons deployed with various BD-E approximation methods. The figure compares (a) BD-E(k) vs. BD(k) and (b) BD-E(crb_T) vs. BD(crb_T).

5.12 Summary

In this chapter, we have studied the beacon deployment problem for localizing sensors (i.e., sensor nodes). The objective is to deploy the minimum number of beacons to localize all sensor nodes. We have shown that the multiset cover problem (MSC), a superset of the beacon deployment problem, is NP-Complete. We then proposed a number of approximation algorithms to solve the beacon deployment problem. An offline version of the approximation algorithm greedily picks the beacon location that covers the largest number of uncovered sensor nodes and is used as a basis for result comparison. The online version maintains a probability distribution of the estimated node locations, and selects a location for the beacon by maximizing the potential of reducing the overall variances of the location distributions. We further describe a variation of the proposed algorithms that uses Cramer Rao Bounds (CRB) as the evaluation criteria, which incorporates the localization accuracy into the deployment problem.

While our simulations only consider the ranging using RSSI as the measurement, the proposed framework will work for any other measurement types such as ToA and AoA. Essentially, the k-cover and crb_T requirements are independent of the measurement types except for the model from which k and CRB are calculated. While in this work the sensor nodes are localized via the beacons directly, we have also shown the advantage of collaboration among sensor nodes. Using this collaboration results in smaller number of beacons required, as sensor nodes would need to hear less of the beacons directly.

CHAPTER 6

LOCALIZATION USING A MOBILE BEACON

6.1 Problem Definition

The localization problem can also be solved using a single mobile beacon that travels through the deployment area. Given a network graph $G = (V_{regular}, E)$, the objective of a mobile beacon-assisted localization algorithm (MBALA) is to find the location of the nodes $V_{regular}$ by a sequence of measurements provided by a mobile beacon.

The mobile beacon-assisted approach is especially suited to localize wireless sensor networks (WSN). WSNs are special cases of MANETs, and the general localization algorithms designed for MANETs work in WSNs as well. Thus, if we model the localization problem in WSNs as the general ad hoc localization problem (GAHLP) defined in Chapter 2, then all algorithms that solves GAHLP will work for WSNs. However, many characteristics unique to WSNs make it practically infeasible to apply MANET localization algorithms; some of the WSN specific challenges include:

1. *Limited computing capacity.* Sensor networks consist of nodes with very limited computing capacity. In addition, sensors are likely to have restricted and limited energy sources further restricting the total amount the computation that can be performed by them. This limited local computing capacity of sensor networks precludes any localization algorithms that require extensive computation at the sensor nodes.
2. *Limited sensory capacity.* Studies have shown that more accurate localization can be performed if nodes are capable of measuring distances and angles to their neighbors. However for such operations additional hardware might be required (e.g.,

to obtain the distance readings from neighbors, nodes have to be equipped with sensors that measure RSSI or ToA). Angle of arrival (AoA) sensors are currently technologically infeasible as they mostly rely on programmable directional antennas. Availability of these sensory capacities just for the localization purpose may increase the complexity and the cost of sensor nodes. Above and beyond, it may be a strong and restricting assumption that all sensor nodes have the same sensory capacity; thus, it is desirable to devise localization algorithms with less reliance on a particular sensory capacity.

3. *Stricter security requirements.* One of the driving applications of sensor networks is to monitor hostile battlefields for enemy activities. In such adversary environments, it is essential for localization algorithms to perform their tasks securely. For localization algorithms that rely on sensor collaboration, it is important that the results are not drastically skewed when a small number of sensors are compromised. Unfortunately, most existing localization algorithms do not consider such security requirement.

4. *Deployment requirement.* WSN nodes are generally deployed in an ad hoc manner; when sensors are "spread" over a battlefield, it is often difficult to exercise any control over the resulting topology. Thus, in the context of localization it is difficult to obtain an ideal beacon placement strategy. For instance, it is well known that most algorithms perform better when beacons are placed around the edge of the network. However, such beacon deployment is often difficult to achieve. Network mobility is another factor; unlike MANETs, sensor networks are generally viewed as stationary once deployed (although some researchers envision mobile WSNs). Thus, it is often sufficient for the localization algorithms to ignore the mobility requirement of MANETs.

Those characteristics unique to WSNs make the mobile beacon-assisted localization more attractive. In this method, a mobile beacon travels through the deployment area while broadcasting its location along the way. Sensors localize themselves by monitoring information coming from the beacon. Compared to the traditional localization methods that rely on stationary beacons and collaborating nodes, localization using a mobile beacon has the following advantages:

1. *Less reliance on the network connectivity.* For a network to be uniquely localized, the necessary condition states that the network needs to be tri-connected and redundantly rigid [29, 4]. However, such condition is difficult to satisfy in a sparse sensor network, which could render it unlocalizable without deploying additional nodes. The mobile beacon-assisted localization method has no such constraints and it is suitable to localize sparse networks.
2. *Overcome obstacles.* Obstacles residing on the deployment field such as buildings, hills and trees could cause significant measurement error of ranging and angle. When sensors are deployed without prior knowledge of the obstacles, the inter-sensor measurement can be error-prone. Since the mobile beacon actually travels through the deployment area, it can potentially "see" those obstacles and avoid them when providing the measurement.

The incremental beacon deployment methods presented in the previous chapter can be modeled as mobile beacon-assisted localization. For each of the new beacon location determined by the deployed method, one can envision that the location is actually served by a mobile beacon. The mobile beacon moves there and then localizes based on the measurement at that location. Thus, the mobile beacon essentially acts like a sequence of virtual beacons from which the regular nodes can be localized. Compared to deploying physical beacons, using a mobile beacon as virtual beacons may be a less expensive alternative hardware-wise.

Other than serving as the mobile beacon, a robot can also perform other tasks necessary for the operation of sensor networks. For instance, the robot can be used to reconfigure or recalibrate sensors, synchronize the clocks, aggregate the collected data from the sensors, deploy new sensors, and disable existing ones. Thus, in sensor networks that already incorporate mobile robots as part of the design, enabling localization through mobile beacons can be a cost-effective way of achieving sensor network localization.

6.2 Previous Works

The concept of localization using a mobile beacon has been implemented in a real world scenario [16], in which an autonomous CSIRO helicopter, equipped with a GPS receiver, was used as the mobile beacon to localize 54 Mica Motes deployed in an outdoor environment. A simple constraint-based Centroid localization method was used in [16]. The estimated sensor locations were applied to compute the shortest paths to be used for navigation purposes.

A straight-forward technique using the above method is described in [78], where sensors are required to receive at least three communications with the same RSSI reading from the beacon. Given that the same RSSI readings imply similar distances to the beacon locations, the physical sensor location can be derived using simple geometric functions. Computation-wise, this method is simple making it suitable for resource-limited sensors. However, it requires the beacon to directly pass by the ranging area of the sensor. In addition, in most cases, the beacon has to pass by the sensor twice because the sampling positions of the beacon when the three RSSI readings are taken should not be on the same line. This method also assumes that errors are insignificant in the RSSI to distance translation.

Instead of computing the location directly, a probabilistic approach may be taken; here sensor location is viewed as a probability distribution over the deployment area.

In [73], sensors measure a series of RSSI readings from the mobile beacons and localize themselves by a sequential update process to the probability distributions of their locations. Each sensor starts with a uniform distribution covering the entire deployment area. As the beacon passes through, the distribution is updated to fit the received RSSI readings (using a propagation model). The method is further improved in [68] by adding the negative information (that is, the information that the beacon is out of range) as well as RSSI readings from the neighbors. These probabilistic methods provide much improved location estimates, but have the drawback of being complex. For a deployment grid of n by n units, the time and space complexity is $O(n^2)$. As the sensors at present time have very limited resources it is difficult to implement these methods directly for the large deployment. Indeed, the experimental results shown in [73] are performed on pocket PCs, which are much more powerful than common sensors.

A similar method of localizing the networks using a mobile beacon was presented in [21]. Instead of the actual probability distribution, the possible sensor locations are represented there with a bounding box. As the beacon passes by, the area contained by the bounding box is progressively reduced as positive and negative information are processed. The bounding box method drastically simplifies the probability computation, making it possible to implement this method on sensors. However, such large simplification has its side-effects in that it sacrifices the preciseness of the distribution for its simplicity as it is not possible to describe multiple possible locations with a single box. There is also an additional problem when noise from the ranging sensors is considered. This method may work well when ranging error is minimal, however when erratic errors are present (which is inevitable when using RSSI ranging), there might be situations where no bounding box exists to satisfy all readings.

In [64], the authors proposed a localization algorithm based on an extended Kalman filter. The localization is performed by storing the location states of every sensor as a vector within a mobile robot and updating this state vector based on the RSSI measurements obtained as the robot moves within the deployment area. The authors implemented the method using Mica2 motes as the sensors and a Lego MindStorm robot as the mobile beacon. Our work is similar to that of [64], but we use a particle filtering method instead. Furthermore, we view the localization problem of each sensor as a local optimization problem, in which a solution is provided when the sensor hears at least three non-collinear beacon broadcasts. The model used in [64], however, views the problem as a global optimization.

6.3 Localization Using a Randomly Moving Beacon

Recognizing various drawbacks of previous mobile beacon-assisted localization methods, we propose a hybrid approach that uses a single mobile beacon to localize nodes in WSNs. We propose a probabilistic framework where sensory readings are processed using a Monte Carlo filtering technique to compute the probability distributions as densities of particle samples. In our method, the localization is performed entirely at the mobile beacon instead of the sensors. During localization, the sensors are only responsible for relaying the sensory data and localization results. For a sensor deployment over a large area, unmanned vehicles or airplanes can serve as mobile beacons. Localization can be performed by a computer onboard or at the base station after the mobile beacon relaying the sensory data back via a secure channel. Our method also supports sensors of different sensory capacity such as ranging, angle or connectivity only, and allows them to co-exist in the same network. Our Monte Carlo filtering technique allows those different types of sensors to collaborate during localization.

By sampling a random process with a random variable and maintaining the outcomes (samples) of the random variable, the probability density of the random process can be approximated. In our case the random variable is two dimensional and takes a value according to the probability of a node's physical location. For instance, when a node is not yet localized, the best estimate about its location's random variable is that it is uniformly distributed and thus the location distribution can be approximated as a set of randomly placed samples over the entire deployment area. When a node receives a transmission from the mobile beacon, it will obtain a distance reading d from the beacon as well as the beacon's location; the random variable's underlying distribution now is modified to a uniform distribution over a circle around the beacon and thus samples will be placed on a ring around the beacon with radius d. When a second distance reading from the beacon (residing at a different location) is overheard, the random variable (optimally) becomes a discrete random variable with two equal probabilities at the intersection points of two rings (the sample set will concentrate on the intersection points of two rings). The exact location can be found when the third broadcast is overheard, assuming all three beacon locations are not collinear.

Overall, there are advantages and disadvantages of using sample points to represent probability densities. While preciseness of actual distributions is somewhat lost by approximating them with sample points, the real advantage of using sample points is that we free ourselves from the complexity of individual distributions (as all distributions would have to be described by mathematical formulas even in the presence of errors). To represent more complex distributions, or to represent distributions more precisely, only the population of the sample set has to be increased. Compared to the bounding box method in [21], sample sets give much better resolution and can deal with distributions with several significant "modes". (For instance, the bounding box cannot precisely describe the actual distribution of the two intersecting rings as described above.) Without the loss of generality in the following discussion we will assume that all sensors and the mobile beacon have the same transmission range. (Our model is easily extended to cases where this is not true; however for the sake of a simple discussion we will keep the previous assumption.) A pair of sensors or a sensor and the beacon can overhear each other if they are in each other's range, i.e., when they are *neighbors*.

Algorithm 9 presents a pseudo-code for our localization algorithm. Sensors are responsible for collecting the range sensory data from neighbors, i.e., this data contains at least one-hop connectivity information. The same data will also include the range or AoA readings from the neighbors if such sensors are available. Such data can be easily collected by observing "Hello" messages from neighbors. The sensors will await the arrival of the beacon, and transmit the collected one-hop sensory data to the beacon. All localization processing is done at the beacon and the result is transmitted back when a sensor is successfully localized.

Algorithm 9 Localization procedure executed at each sensor

```
form a observation set $R_v = (...r_w...)$, where $r_w$ is an observation from neighbor $w$
send $R_v$ to beacon upon request
receive localization result $(id, x, y, var)$
if $id = v$ then
  $(x, y)$ is the location of the sensor, and $var$ is the variance
else
  forward $(id, x, y, var)$ to the neighbor $id$
end if
```

The localization algorithm at the mobile beacon is listed as Algorithm 10. Location distributions for sensors are stored as sample sets at the mobile beacon. We assume that the movement of the beacon is defined by a bacon movement model. For each new location of the beacon, there are two types of observations that are useful to localize the sensor: i) positive observations in which a sensor can hear the beacon; ii) negative observations where a sensor is out of range. The beacon updates the localization distributions for both positive and negative observations. To reduce the amount of processing, we define a critical region for each sensor v, so that the location distribution will only be updated when the beacon arrives at the critical region. For connectivity-only observations, it is more useful to update the distribution as the beacon just enters or is about to leave the sensor range. Range-based observations are useful when the movement trajectory is not collinear. AoA-based observations do not have many restrictions as long as the beacon does not pass directly over the sensor. Based on the above observations, we define the critical region as a predefined outer ring of the maximum transmission range of the beacon. The location distribution will be updated when a sensor just "arrives" in the critical region or when the beacon changes its moving direction.

The beacon calls the UpdateFilter procedure to update the location distribution of sensor v based on the sensor's latest observation data R_v as well as a reference distribution X_{beacon}. The reference distribution X_{beacon} is simply the current location of the beacon. After the location distribution of sensor v is updated, the beacon will also update the location distribution of v's neighbors based on the one-hop observation data between the pair of neighbors. Here, we use X_v as the reference distribution. This later step helps to increase the localization performance as it allows the sensors to be localized even when the beacon does not pass by them directly. Our simulations show that using the one-hop observation data reduces the average estimation error up to 50% when the beacon can only spend limited amount of time over the deployment area.

The beacon decides whether a sensor is localized based on the variance of the location distribution. If the variance is reduced below a pre-define threshold T_{var} (as defined by the user or the application), the sensor is considered localized. The mode of the particle density distribution is used as the estimated location, which is transmitted back to the sensor. The variance of its particle distribution can also be sent back to serve as an indicator of the estimation quality, with smaller variances indicating better estimates. Note that the threshold T_{var} could vary depending on the application that utilizes the location information. It has been previously noted that precise locations are not essential for some applications [81]. In those cases, a larger threshold can be chosen so that better localization coverage can be obtained.

Algorithm 10 Localization procedure executed at the mobile beacon

```
$\hat{R} \leftarrow$ storage of all observation sets
$X_v \leftarrow$ sample set of sensor $v$, initialized uniformly
$X_{beacon} \leftarrow$ sample set of the beacon reflecting the current location
move over the deployment area based on a movement model
for all sensor $v$ in $\hat{R}$ such that there is new negative observation $N_v$ do
    $X_v \leftarrow$ FilterUpdate($X_v$, $X_{beacon}$, $N_v$)
end for
if the current location is in the critical area of sensor $v$ OR beacon just changed direction then
    request $R_v$ from $v$
    update $R_v$ in $\hat{R}$
    $X_v \leftarrow$ FilterUpdate($X_v$, $X_{beacon}$, $R_v$)
    if $var(X_v) < T_{var}$ then
        send localization result $(v, x, y, var)$ to $v$, where $(x, y)$ is the expected location
    end if
    for all $v$'s neighbor $w$ do
        $X_w \leftarrow$ FilterUpdate($X_w$, $X_v$, $R_w$)
        if $var(X_w) < T_{var}$ then
            send localization result $(w, x, y, var)$ to $v$, where $(x, y)$ is the expected location
        end if
    end for
end if
```

6.3.1 System Model

To apply the classic Monte Carlo sampling-based Bayesian filtering approach described in the previous section, we need to provide a system model and a measurement model. For the system model, we simply assume that at any point in time the node moves with a random velocity drawn from a Normal distribution with a mean of $0m/s$ and a fixed standard deviation σ. No information about the environment is included in this model, and as a consequence, the filter permits the estimates to move along arbitrary paths. Although we know that in the system sensor nodes do not move, we still have to assign a movement model to enable particles to move closer to the actual location. Thus, our system model is simply $p(s_t|s_{t-1}) = N(0, \sigma)$, where N denotes the Normal distribution.

In summary, using the above system model, at every iteration each particle $s_{t-1}^{(i)}$ within the sample set X_v will be randomly moved from its original location based on the Normal distribution, resulting into particle $\tilde{s}_t^{(i)}$. The time complexity of this step is $O(N)$, where $N = |X|$ is the size of the sample set; the space complexity is also $O(N)$.

6.3.2 Measurement Model

The measurement model is used to modify the initial particle $\tilde{s}_t^{(i)}$ to fit into the latest observation. Consequently, the measurement model depends on the observation data R_v as well as the reference distribution X_w. Here, we consider the following four types of observation data: i) connectivity only, ii) exact ranging, iii) bounded ranging and iv) AoA. Regardless of the observation data, the original particle $\tilde{s}_t^{(i)}$ is updated by comparing its location to the observation data using each particle in the distribution X_w as its reference. More weights are awarded to the particles that are more consistent with the observation. For every particle in X_v and X_w, the procedure AssignWeight is called to award the weight according to the observation data, R_v, as follows:

1. *Connectivity Only.* There are two types of connectivity only observations: positive and negative. We assume that all sensors are able to make connectivity observation at the very minimum. When R_v is a positive reading, it means nodes v and w are in the range of each other. Therefore, AssignWeight returns 1 if the distance between the two samples $d(\tilde{s}_t^{(i)}, s_t^{(j)}) \leq d_{max}$, where d_{max} is the maximum range possible. Otherwise, AssignWeight returns 0. Conversely, when R_v is negative, it means that

the two nodes are out of range; thus AssignWeight returns 0 if $d(\tilde{s}_t^{(i)}, s_t^{(j)}) \leq d_{max}$, and 1 otherwise.

2. *Exact Ranging.* When relatively reliable ranging data can be obtained using methods like ToA, the ranging observation can be used directly to update the weights. Let the distance reading be $\hat{d}$. We compare the observed distance $\hat{d}$ with the sample distance $d(\tilde{s}_t^{(i)}, s_t^{(j)})$. In this case, we use the following evaluation equation

$$AssignWeight = \begin{cases} 1 - (\frac{|d(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{d}|}{d_{max}})^2, & |d(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{d}| \leq d_{max} \\ 0, & \text{otherwise} \end{cases}$$

   Compared to the earlier case where only connectivity data is available, the above equation will award more weight to the particles more consistent with the observed distance $\hat{d}$. Also, the weight increases quadratically as the distance difference decreases. We note that more complex methods could be used; however our simulations indicate that this simple method works as well as more complicated models.

3. *Bounded Ranging.* Ranging readings using methods such as RSSI are known to be unreliable due to multipath fading and scattering. In reality, the received signal strength would be usually weaker than what is expected based on the signal propagation model in the ideal environment. The distance estimate derived from such RSSI reading would be usually larger than the actual distance. In such cases, using the exact distance reading could adversely influence the location distribution. Thus, we use the bounded ranging observation, in which we merely regard the actual distance to be bounded by the observed distance. Let the observed distance reading be $\hat{d}$. For the bounded ranging observation, AssignWeight returns 1 when $d(\tilde{s}_t^{(i)}, s_t^{(j)}) \leq \hat{d}$, and 0 otherwise.

4. *AoA*. The AoA observation is similar to the exact ranging. Let $\hat{a}$ be to observed AoA reading, and $a(\tilde{s}_t^{(i)}, s_t^{(j)})$ be the AoA between the two particles. We use the following equation

$$AssignWeight = \begin{cases} 1 - (\frac{|a(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{a}|}{\pi})^2, & |d(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{d}| \leq d_{max} \\ 0, & \text{otherwise} \end{cases}$$

   The above equation will only award weight to a particle if its calculated distance to the reference particle is in range. More weight is awarded to the particles that are more consistent with the AoA reading.

When the weights of all particles in X_v have been assigned, they are normalized so that the sum of all weights becomes one. Then, particles are re-sampled with the probability governed by the weight distribution. The new particle is generated using a Normal distribution centered at $\tilde{s}_t^{(i)}$ and a standard deviation being the average of the standard deviation of the previous distributions X_v and X_w. The average is used here because our measurement model is influenced by both the original distribution X_v and the reference distribution X_w. As a common technique to avoid local minima, we also spread 5% of the particles randomly.

Overall, the time complexity of the measurement step is $O(N^2)$, and the space complexity is $O(N)$.

Algorithm 11 shows the FilterUpdate procedure including both the system model and measurement model. Notice that we actually skip the sample update steps if the variance of the reference distribution X_w is greater than the variance of the current sample distribution. This can be justified by the observation that it is unnecessary to update the current location samples when the reference samples are worse. Only when the reference samples are sounder than the current location samples should we run the update. This serves two purposes: i) reducing the amount of unnecessary processing; and ii) preventing the distribution to diverge when the neighbors adversely influence each other.

Algorithm 11 FilterUpdate(X_v, X_w, R_v)

```
if $var(X_v) < var(X_w)$ then
  return $X_v$
end if
$X_v \leftarrow$ sample set for node $v$ to be updated
$X_w \leftarrow$ reference sample set for node $w$ from which $R_v$ is observed
$R_v \leftarrow$ observation data
for all particle $(s_{t-1}^{(i)}, w_{t-1}^{(i)}) \in X_v$ do
  generate $\tilde{s}_t^{(i)} \leftarrow s_{t-1}^{(i)} \cdot N(0, \sigma)$
  $\tilde{w}_t^{(i)} \leftarrow 0$
  for all particle $(s_t^{(j)}, w_t^{(j)}) \in X_w$ do

    $\tilde{w}_t^{(i)} \leftarrow \tilde{w}_t^{(i)} +$ AssignWeight($\tilde{s}_t^{(i)}$, $s_t^{(j)}$, $R_v$)
  end for
end for
Normalize weights and resample $s_t^{(i)}$ based on the weights
Randomly generate p% particles
return new sample $X_v := \{(s_t^{(i)}, w_t^{(i)}) | i \in [1, N], w_t^{(i)} = 1/N\}$
```

6.3.3 Security Concerns

Compared to pure distributed localization methods such as APS, our method can be regarded as "semi-centralized." However, unlike typical centralized algorithms which require the presence of all observation data, our method works more like an online algorithm, in which all computation is done at the beacon. This model makes it easier to secure the localization process simply by securing the beacon itself. Let us assume that the beacon cannot be compromised, but the individual sensors can. A compromised sensor could attack the localization process in two ways: i) reporting false observations to the beacon and ii) interrupting the localization results. The report of false observations would affect the localization of the compromised sensor itself and more importantly those of the neighboring sensors one-hop away. However, given time, the beacon would eventually move into the range of the neighbors and thus bypass the compromised sensor. Since the Monte Carlo sampling method is designed to work under uncertain observations, the false observations would eventually be filtered out as more direct observations are made. To help preventing the second type of attack, we require all messages sent by the beacon that contain the localization results to be encrypted with the receiver's unique key (or public key) and signed with the beacon's signature. This way, the compromised sensor will not be able to falsify the localization results. Although the decryption process can be expensive considering the limited capacity of the sensors, it needs to be performed one time only. A compromised sensor may still elect not to forward the localization results, however such attack can be countered by sending the localization results via multiple paths.

6.4 Simulation Results

We have built a custom C++ discrete event simulation to evaluate the performance of our method. While our particle filter framework has no such restriction, we model all nodes in our simulation to have an identical transmission power. Thus, we can effectively control the network density (average degree) by varying the transmission range. When a node is located within the transmission range of another node, we assume that it is capable of receiving signal from the sender. We use a sensor network consisting of 49 nodes randomly deployed in a 1000m by 1000m square. A single mobile beacon flies through the deployment area. An epoch-based model controls the beacon movement. Starting from a random initial location, the beacon randomly chooses a velocity based on Normal distribution with a means of 20m/s and a standard deviation of 20m/s. The beacon proceeds to move at the chosen velocity for an epoch of time that is exponentially distributed with an expected value of 20s. At the end of the epoch, the beacon chooses a new velocity and time. When the beacon reaches the boundary of the deployment area, it bounces back at the same incidence angle (much like a ping pong ball). While this model is much more realistic than a random motion model, we acknowledge that it still lacks many features of real beacon movement. For instance, airplanes or vehicles tend to make a smooth directional changes instead of a sudden turns.

Our simulation considers the following network parameters: the network connectivity is d, the signal to noise ratio is σ_{noise}, the number of particles used is N. For each new network scenario, we record the average estimation error and coverage, based on availability of four different types of location sensory data: i) connectivity only, ii) exact ranging, iii) bounded ranging and iv) angle (AoA). We also consider the scenario where mixed types of sensory data co-exist in the same network. The final results shown in the figures are the average of 50 independent runs of different network topologies (exceeding 95% confidence with 5% relative error). We consider a sensor localized when its sample variance is less than a threshold T_{var}. Here we use $T_{var} = (d_{max}/2)^2$, that is, the standard deviation of the sample is less than half of the sensor range; this coverage criterion is rather arbitrary, as we can adjust to either a looser or a stricter threshold based on the actual application requirement. We also note, that the estimation error shown in the figures is the average of *all* nodes, including those not being localized.

6.4.1 Base Scenario

First, our simulations concentrate on a basic scenario so that we have a benchmark as a basis for comparison. For the basic scenario, we pick the average degree to be $c = 3.63$, the noise ratio to be $\sigma_{noise} = 0$ and the particle population to be $N = 200$. Figure 6.1 shows one such network. Networks of such low connectivity are highly likely

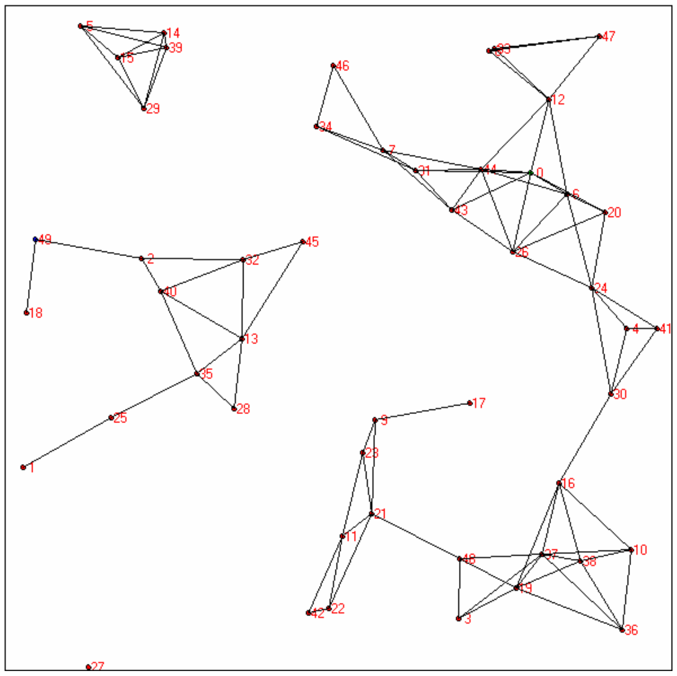


Figure 6.1. Base scenario of a network of connectivity = 3.63. In the figure, two sensors are connected if a link exists in between.

to be disconnected. Previous research indicates that it can be very difficult to localize sparse networks using collaborative localization methods such as APS [14]. However, we will show that it is still possible to obtain reasonably good localization results using a mobile beacon. Figure 6.2 shows the estimation error of every sensors in the same network of Figure 6.1 after the beacon has traveled over the deployment area for 500 seconds; we assume that AoA sensory data is used. The lines drawn from the nodes point in the figure to the estimated locations.

The average estimation error and the coverage for each of the four types of availability of sensory data are shown in Figure 6.3. As the beacon spends more time over the deployment area, the estimation error decreases while the coverage increases, indicating that more nodes are being localized. After 200 seconds, the average estimation error is reduced below the set threshold. Since the basic scenario does not include any noise,

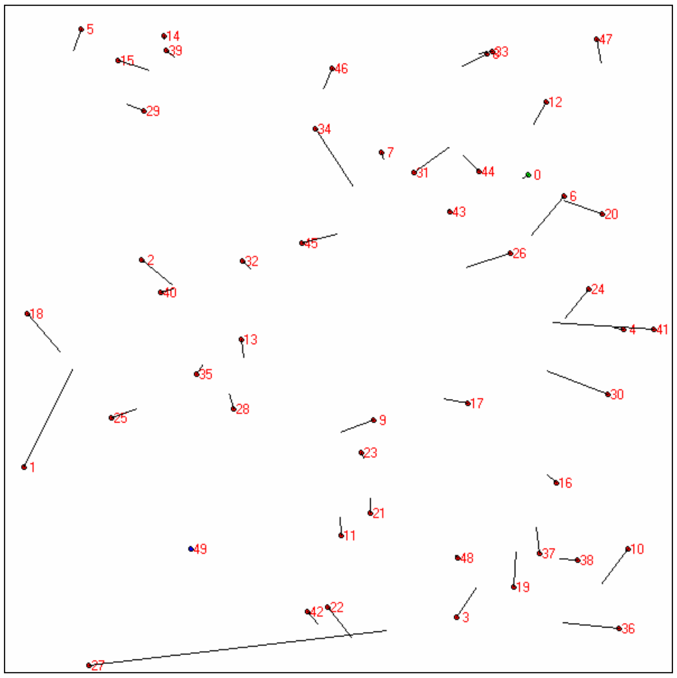
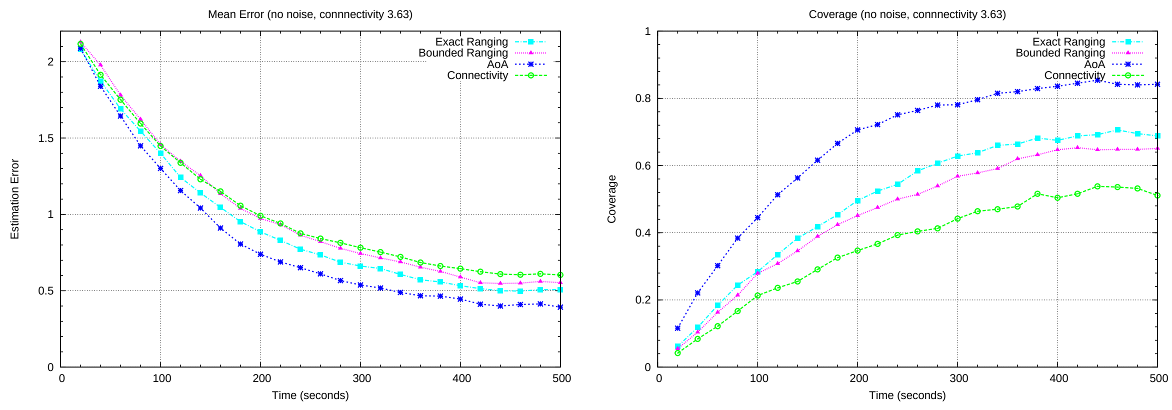


Figure 6.2. Estimation error at the end of a 500 second run. In the figure, the lines point to the estimated locations.

sensory data based on exact ranging and AoA are accurate and thus we expect these results to be better. Best results are obtained using the AoA data. AoA data has an advantage over exact ranging because it takes only two (versus three) known locations to localize the sensor. Also note that the improvement of the estimation results slows down with time progressing. This is expected because of the (semi-) random beacon movement. As time advances it becomes less and less likely for the beacon to reach a previously unvisited area. Furthermore, the sensors that are the most difficult to localize reside at the boundary of the network, where they are less likely to be visited by the beacon.

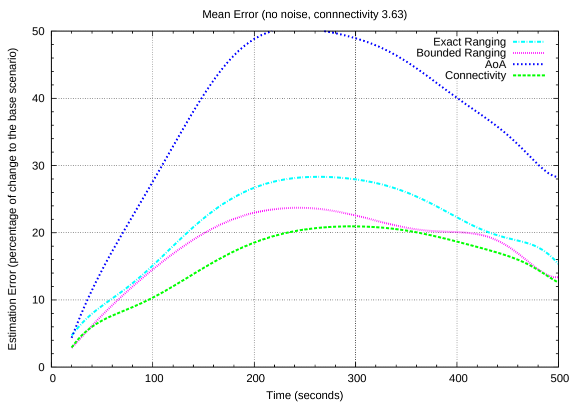


(a) (b)

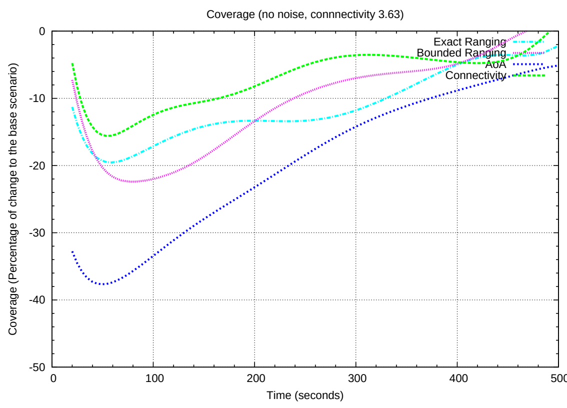
Figure 6.3. Results on the base scenario. The figure shows (a) estimation error and (b) coverage.

6.4.2 One-Hop Neighbor Observations

As stated in Algorithm 10, the localization process does not only consider direct readings from the beacon to the sensor, it also considers one-hop neighbor observations among sensors. Figure 6.4 shows results when one-hop neighbor observations are *not* used. In this case only the sensors that could directly overhear the beacon can be localized, which has been the traditional method of localization using a mobile beacon as proposed in [73]. However, as shown in Figure 6.4, in such case the localization results are inferior compared to our basic case as the estimation error increases and the coverage decreases. In particular, the difference of the estimation error peaks between 200 and 300 seconds, where up to 50% performance increase can be obtained by adding the one-hop neighbor observations. The advantage of using one-hop neighbor observations becomes less significant as time advances because the beacon will cover more and more previously not visited areas. Nevertheless, this experiment demonstrates the advantage of using one-hop neighbor observations especially when the beacon can only spend limited time over the deployment area.



(a)



(b)

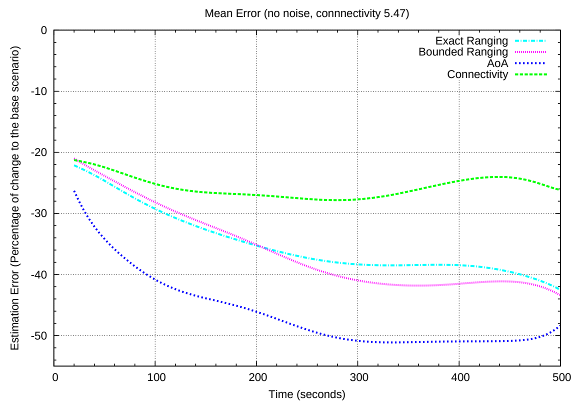
Figure 6.4. Results when 1-hop neighbor observations are *not* used. The figure shows (a) estimation error (percentage of change to the base scenario) and (b) coverage (percentage of change to the base scenario).

6.4.3 Connectivity

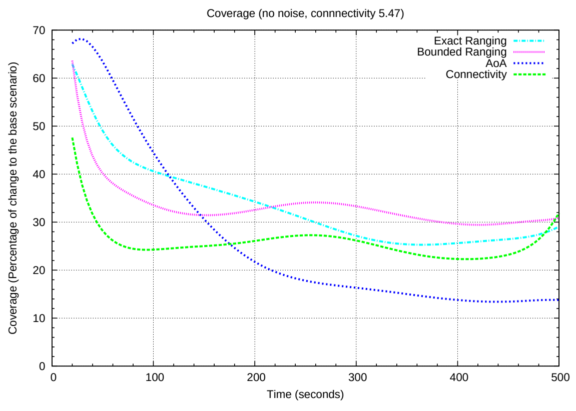
Figure 6.5 contains the results when the average degree is increased from 3.63 to 5.47. We can observe that the change within the first 100 seconds is rather large; this can be credited to the fact that fewer sensors are localized during that time, which skews the results heavily. The change stabilizes as time advances and more sensors are localized. At the end of the 500 seconds simulation, we observe a decrease of estimation error by 25% to 50% and an increase of coverage by 15% to 30% when the connectivity increases to 5.47. This is to be expected as most localization methods work better on denser networks.

6.4.4 Noise

All of our previous experiments assumed an ideal scenario where location sensory data are not influenced by noise. Here we will consider a noisy environment with the random noise added to measurements. In the case of ranging, when the actual distance is D, the measured distance will be $D + D \cdot Uniform(0, \sigma_{noise})$. In this case, we consider



(a)

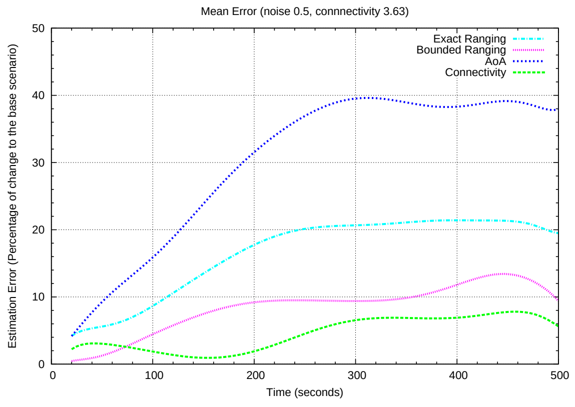


(b)

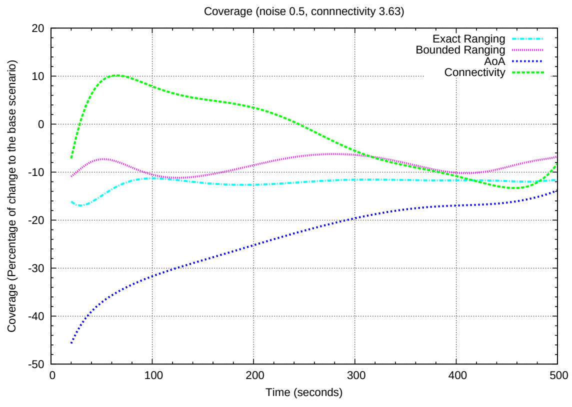
Figure 6.5. Results of a denser network. The figure shows (a) estimation error (percentage of change to the base scenario) and (b) coverage (percentage of change to the base scenario).

a situation where noise always *adds* to the actual distance (as described in the previous sections). The noise to AoA, on the other hand, is two-sided as it could either be *add*ed to or be *subtract*ed from the actual angle. Thus, if the actual angle is A, the measured angle will be $A + A \cdot Uniform(-arcsin(\sigma_{noise}), arcsin(\sigma_{noise}))$ (see section 2.6.6). This makes the noise ratio comparable to ranging and AoA in that for the same noise ratio σ_{noise}, the amount of noise for AoA is about twice as much as that of ranging.

Figure 6.6 shows results when the noise ratio σ_{noise} is set to 0.5 (i.e., 50% noise). In all cases, the estimate's error increases and the coverage decreases when compared to the basic scenario. The increase is less significant for the bounded ranging and connectivity type scenarios because they rely less on the exact sensory readings. But exact ranging and AoA suffer up to 40% performance loss. As expected from the noise model, AoA degrades about twice as much as ranging.



(a)



(b)

Figure 6.6. Results when noise ratio $\sigma_{noise} = 0.5$. The figure shows (a) estimation error (percentage of change to the base scenario) and (b) coverage (percentage of change to the base scenario).

6.4.5 Mixed Sensory Data

Since the same Monte Carlo localization framework can be applied to different sensory data, we also simulate the scenario where nodes are heterogeneous in their location sensory capabilities. We randomly assign a certain percentage of sensors to one of three location sensor types: exact ranging, AoA or connectivity-only; Figure 6.7 shows the results. It is interesting to note that adding AoA sensors helps to increase the localization performance. In particular, similar performance can be achieved using 1/3 of AoA, 1/3 of ranging and 1/3 of connectivity-only when compared to the scenario where all sensors have ranging capacity.

6.5 Summary

In this chapter, we proposed a localization method for wireless sensor networks using a probabilistic technique employing Monte Carlo sampling. We considered a scenario where a single mobile beacon passes through the deployment area. Location of nodes is computed on the beacon instead of the sensors. Possible sensor locations are represented

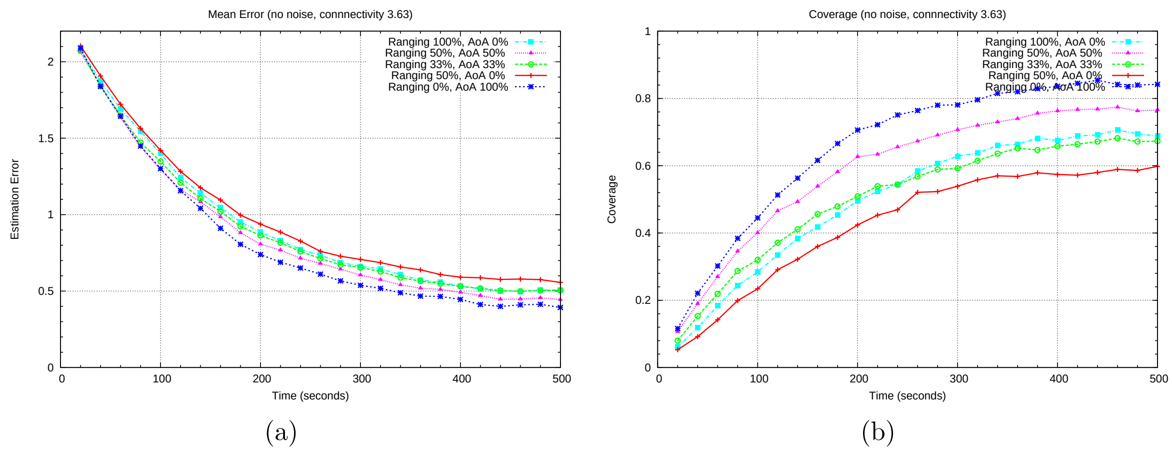


Figure 6.7. Results of using mixed sensory data. The figure shows (a) estimation error and (b) coverage.

by particle samples for each node, which are updated as the beacon passes through the network. The only tasks related to localization performed directly on the sensors are monitoring the sensory data from the neighbors and relaying localization results. This arrangement relieves the sensors of the heavy processing needed for accurate localization. At the same time, the use of sample sets allows us to maintain more complex location distributions without sacrificing the resolution. Furthermore, since all computation is done at the beacon, the localization process can be more easily secured. Our method also allows sensors of different sensory capacities, such as ranging and AoA, to exist in the same network. Simulations show that our method is able to obtain localization error of less than 50% of the transmission range and over 80% coverage on very sparse networks of degrees less than 4. Better results can be obtained by increasing the network connectivity.

CHAPTER 7

PATH PLANNING OF MOBILE BEACONS FOR LOCALIZATION

7.1 Problem Definition

In Chapter 6, we investigated the localization problem using a mobile beacon. We concentrated on the design of the actual localization algorithm executed as the mobile beacon under the simple assumption that the mobile beacon moves based on a random waypoint model. However, it is natural to question whether the random model can be improved. In this chapter, we study the path planning problem for localization. In particular, we ask the question of what should be considered as a better path to be taken by the mobile beacon with sensor localization as our primary objective. We further distinguish the path planning problem into static path planning and dynamic path planning. For the static paths, the mobile beacon follows a pre-determined path to perform localization. For the dynamic path planning, the mobile beacon is allowed to adjust its path during localization.

Note that the path planning problem we are considering here is related but ultimately different than the path planning and localization problem in robotics. The problem in the field of robotics deals with the issue of localizing and navigating robots in an unknown environment based on a certain type of signal behavior (from a per-calibrated RSSI map or pre-localized sensors) [84, 48, 23]. The problem we are considering here deals with localizing the sensors using a GPS-equipped mobile robot.

7.2 Static Path Planning

In this section, we study the problem of static path planning for localization *prior* to the beacon deployment. For this problem, we assume that, although we do not know the exact sensor locations prior to the beacon deployment, the sensors are uniformly distributed over a predefined deployment area. The objective of our static path planning is to design a path to guide the mobile beacon such that i) a higher percentage of the sensors can be localized (i.e., better coverage), ii) the localization error is minimized (i.e., better accuracy), and iii) the travel path length of the mobile beacon is shortened. We consider such path planning as *static* since it is done prior to the localization and cannot be modified during the localization. As such, the static path planning problem is invariant to the actual localization algorithm.

Our main contributions in this section include two new static path types, CIRCLES and S-CURVES, that are designed with localization accuracy and coverage in mind. We again base our static paths on the concept of Cramer Rao Bounds (CRB), an unbiased lower bound of localization accuracy, as the evaluator of path types. We then compare our path types to previously proposed ones using the CRB analysis. Our results show a clear advantage of our path types over others in terms of coverage and accuracy without increasing the path length.

7.2.1 Related Works

There has been limited amount of previous work on the path planning problem for sensor network localization. The problem was first brought up by [73] where the authors were primarily interested in the localization algorithm design of mobile beacons. As a side note, the authors acknowledged the difficulty of selecting an optimal trajectory for the beacon in that the trajectory needs to provide at least three non-collinear beacon broadcast locations for each sensor. However, the authors did not provide any specific path planning method.

While the authors of [78] did not explicitly study the path planning problem, they implicitly considered this issue during their localization algorithm design. In particular, they provided an algorithm running on the sensor nodes that tracks the series of beacon broadcasts to only use the non-collinear ones to localize. They did not introduce a path planning algorithm, i.e., the mobile beacon is assumed to move randomly, thus their method provides a passive way of extracting usable coordinates from the random path.

In [44], the authors studied three different types of static paths: SCAN, DOUBLE SCAN, and HILBERT in relationship to localization. The results show that HILBERT has the best localization performance as its larger number of direction changes effectively reduces the collinearity during localization. Their results are obtained using simulations of randomly deployed sensor networks using a Monte Carlo localization (MCL) method [30] as the localization algorithm.

In this section, we build on the work by [44] to design additional static paths that further reduce the collinearity. Instead of evaluating paths based on a particular localization algorithm, we use the more general Cramer Rao Bounds (CRB) as the metric to compare different paths. The CRB gives a theoretic lower bound on the best localization error achievable by *any* localization algorithm. Thus, by using CRB as the metric, we are able to provide a more fair comparison by eliminating any bias introduced in favor of a particular localization algorithm. Under the same test case, a path type that achieves the better overall CRB should be considered to be better.

7.2.2 Cramer-Rao Bounds (CRB)

We use the Cramer-Rao Bounds (CRB) as the evaluator of static path types. The CRB is an effective measure to qualify the localization inaccuracy attributed to the measurement types and noises[65]. The CRB is a lower bound on the covariance of any unbiased location estimator that uses noisy measurements such as RSSI, ToA, or AoA. Thus, the CRB provides a lower bound on the estimation accuracy of a given network scenario regardless of the localization algorithm. In other words, with CRB we have a way to tell the best *any* localization algorithm can do given a particular network, measurement type and measurement noise scenario. The CRBs of individual measurement types such as RSSI, ToA and AoA under most common noise models (mostly Gaussian) are discussed in more detail in [65]. Refer to Chapter 2 (Section 2.3)for a more detailed coverage on the CRB.

7.2.3 Static Paths

In this section, we define four different static path types and compare their localization performance using the CRB analysis. As in [44], we adopt a deployment area consisting of a 480m by 480m square. To calculate the CRB, we assume a Gaussian range measurement noise with a constant variance introduced by shadowing. The received signal strength from a beacon location i to a sensor node j that are $d_{i,j}$m apart is therefore

$$N(P_0 - 10n_p log_{10}(d_{b,s}/d_0), \sigma_{dB}^2) \tag{7.1}$$

where P_0 is the received signal strength at a reference distance d_0. Here, we use $d_0 = 1$m. n_p is the environment-dependent path loss exponent, and σ_{dB}^2 is the constant variance introduced by the shadowing. As in [66], we choose $n_p = 1$ and $\sigma_{dB}^2 = 1.7$.

We test four types of static paths: SCAN, HILBERT, CIRCLES and S-CURVES. Of the four types, SCAN and HILBERT were originally proposed in [44] (we include them for comparison purposes). We omit the third path type proposed in [44], DOUBLE

SCAN, here as it has been shown in [44] that it does not provide a significant advantage over SCAN or HILBERT.

For sensor network localization using a mobile beacon, the CRB values of the sensors are influenced by the following factors:

1. *Path Length.* A longer path means that the mobile beacon would have more opportunity to broadcast its location. Thus, we expect a better overall localization (i.e., lower CRB) with a longer path.
2. *Broadcast Interval.* A shorter broadcast interval means that the mobile beacon would broadcast its location more frequently. Thus, we expect a better overall localization with a shorter broadcast interval.
3. *Broadcast Range.* A larger broadcast range (transmission radius) of the mobile beacon would allow each broadcast to cover more sensors. Thus, we expect a better overall localization with a larger broadcast range.

The four types of static paths considered are shown in Figure 7.1. To provide a valid comparison among them, we fix the broadcast range (45m). We set the broadcast interval to 10m (i.e., there is one broadcast every 10 meters traveled). Therefore, the actual number of broadcast locations is a function of the path length. Each of the four different path types has a different path length. The solid line in Figure 7.1 shows the actual path, and the rectangles along the path show the broadcast locations (the distance between two rectangles is the broadcast interval). The small dots represent the locations from where a valid localization can be performed based on the RSSI range readings. Because of geometric constraints, a valid localization can be performed at a location only when at least three non-collinear broadcasts are heard.

7.2.3.1 SCAN

Of the four path types, SCAN, introduced in [44], is the most straight-forward in which the mobile beacon simply sweeps the deployment area in straight lines from left to right. More formally, SCAN divides the square deployment area into n by n sub-squares ($n = 8$ in our case) and connects their centers using straight lines as in Figure 7.1(a). The *path resolution*, R, of SCAN is defined as the side length of each sub-square ($R = 60$m in our case). The drawback of SCAN is that straight lines introduce collinearity, and because of this there are many locations where the beacon broadcasts heard are collinear. Since the broadcast range is set to 45m and two nearby vertical lines in Figure 7.1(a) are two resolutions apart (120m), the area near the vertical lines cannot be localized because all beacon broadcasts come from the same vertical line and thus are collinear. To reduce collinearity, we would have to reduce the resolution to match the broadcast range, which would substantially increase the path length.

7.2.3.2 HILBERT

To reduce the collinearity without significantly increasing the path length, HILBERT is proposed in [44], which makes the mobile beacon to take more turns. Same as SCAN, HILBERT divides the 2-dimensional space into n by n sub-squares ($n = 8$ in our case), but connects the centers of the sub-squares using n line segments as shown in Figure 7.1(b). The resolution of HILBERT is defined as the side length of each sub-square ($R = 60$m in our case). While the path length of HILBERT is $n \cdot R$ longer than that of SCAN at the same resolution, it contains shorter line segments, which reduces collinearity. We can see in Figure 7.1(b) that a significantly greater area can be localized with HILBERT compared to SCAN.

7.2.3.3 CIRCLES

Since straight lines invariably introduce collinearity, we would like to reduce the number of straight lines on the beacon path. Thus, we design CIRCLES which consists of a sequence of concentric circles centered within the deployment area. We define the resolution, R, of CIRCLES as half of the radius of the innermost circle, and we sequentially increase the radius by R at each outer circle ($R = 60$m in our case). Note that we could have defined a spiral like path that looks like CIRCLES, but we decided to study CIRCLES because its resolution parameter is comparable to other path types.

As shown in Figure 7.1(c), since CIRCLES does not introduce collinearity explicitly, all areas within the circles can be localized. However, since the deployment area is a square, CIRCLES would not cover the four corners effectively without adding larger circles, which would increase the path length. Furthermore, CIRCLES has an inherent scalability issue. When the deployment area increases, CIRCLES would require the beacon path to contain larger circles. As the circles become larger, the amount of non-collinearity is reduced, which in turn reduces the localization accuracy. Figure 7.2 illustrates such scalability issue in terms of CRB. Here, we measure the CRB of CIRCLES at various y locations when fixing $x = 240$m (i.e., splitting the deployment area in the middle). We observe that the CRB is at the minimum at the inner circle (around 240m), but it increases gradually to the outer circles (approaching 0m to the left and 480m to the right).

7.2.3.4 S-CURVES

S-CURVES is based on SCAN, which progressively scans the deployment area from left to right. However, at each scan, S-CURVES takes an 'S' curve instead of going in a straight line. More formally, dividing the deployment square into n by n sub-squares

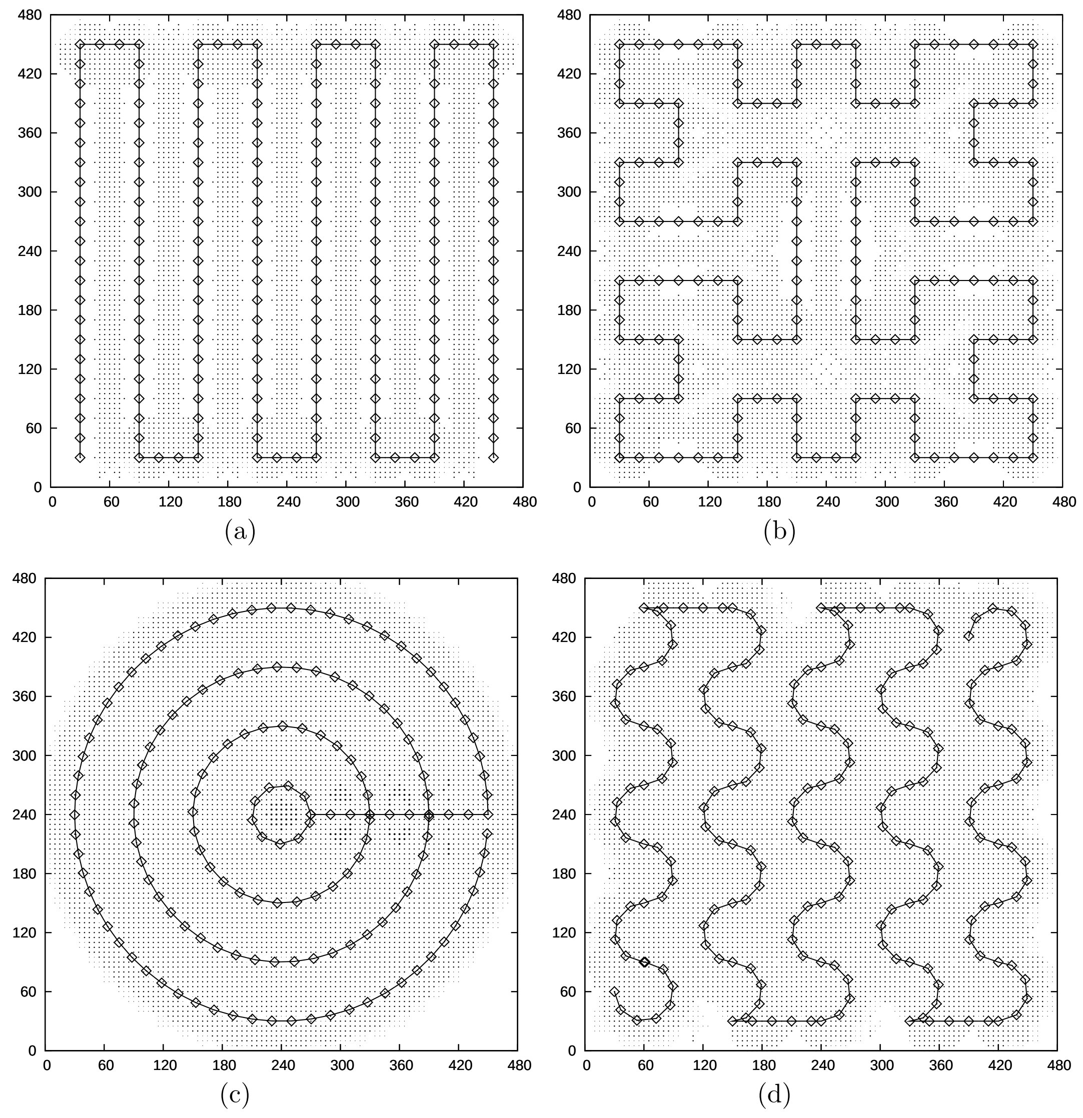


Figure 7.1. Static path types. The figure shows four path types: (a) SCAN (path length = 3780.00m), (b) HILBERT (path length = 3840.00m), (c) CIRCLES (path length = 3195.93m), and (d) S-CURVES (path length = 3752.92m).

($n = 8$), and let the resolution of S-CURVES be R ($R = 60$m). Then, each vertical S curve consists of $n - 1$ half squares of radius $\frac{R}{2}$, and there are a total of $\lfloor 2(n-1)/3 \rfloor + 1$ S curves from left to right. The S curves are connected with short straight lines like in SCAN.

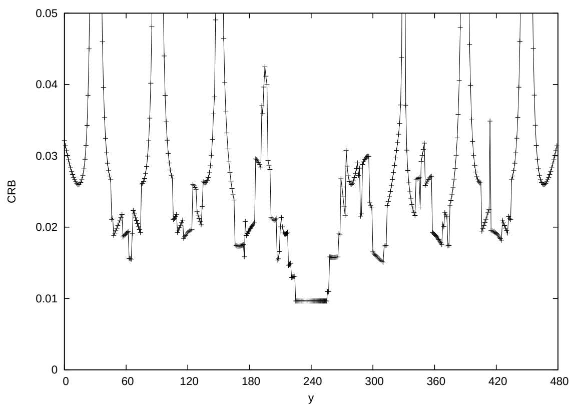


Figure 7.2. The CRB of CIRCLES at $x = 240$.

7.2.4 Evaluation

We compare the four path types based on three metrics: i) total path length, ii) the localization coverage, and iii) the localization accuracy. Consider the path length first. For each of the four path types, the total path length, L, is a function of R and n:

$$\begin{aligned} L_{SCAN} &= (n^2-1)R \\ L_{HILBERT} &= n^2R \\ L_{CIRCLES} &= \frac{n^2\pi R}{4} + (\frac{n}{2}-1)R \\ L_{S-CURVES} &= \frac{(n-1)\pi R}{2} \cdot (\lfloor 2(n-1)/3 \rfloor + 1) \\ &\quad +(n-2)R + \frac{R\pi}{2} \end{aligned}$$

As seen in our test scenario, CIRCLES has the shortest path length. The other three path types have similar path length, with S-CURVES being slightly shorter than the other two.

Now consider the localization coverage and accuracy. Here, we compare the four path types using Cramer Rao Bounds (CRB). For each path type, we calculate the CRB at various locations of the entire deployment area. We assume a 1-hop propagation of RSSI readings, and thus the CRB at each location is a strict function of the RSSI readings from broadcast locations 1-hop away. For those locations that *cannot* be localized because of the unavailability of three non-collinear broadcast locations, the CRB will be infinity. For other locations that *can* be localized, the CRB gives a tight lower bound of the localization error that can be possibly achieved at the particular location. Thus, the CRB analysis gives an estimate of both localization coverage and localization accuracy.

To perform the CRB analysis, we divide the 480m by 480m deployment area into a system of 1m by 1m grids. We then calculate the CRB at every grid location and construct a histogram of the CRB ranges produced by each of the four static path types. To study the relationship between the transmission range and the path resolution, we vary the beacon transmission range from 30m to 75m, while fixing the path resolution at 60m. Notice that varying the transmission range while fixing the path resolution can be seen as equivalent to varying the path resolution while fixing the transmission range.

Figure 7.3 shows the results of our CRB analysis. For each of the transmission ranges, we construct a histogram consisting of seven categories based on the CRB ranges: $[0, 0.01)$, $[0.01, 0.02)$, $[0.02, 0.03)$, $[0.03, 0.04)$, $[0.04, 0.05)$, $[0.05, 0.5)$, and $[0.5, \inf)$, represented as patterned boxes from bottom to top within the histogram bar. Each category contains the percentage of the grid locations whose CRB falls into its corresponding range. The percentage of a category is reflected by the size of the corresponding patterned box. Using the histogram, we are able to compare the static path types in terms of localization coverage as well as localization accuracy. A histogram with a large number of grid locations in the $[0.5, \inf)$ category (i.e., at the top of the histogram bar) indicates a poor coverage because the CRB values within this category are large indicating that those grid locations are difficult to localize. Conversely, a large number of grids in the first several categories (i.e., at the bottom of the histogram bar) indicates better localization accuracy.

Figure 7.3(a) clearly shows the superiority of CIRCLES and S-CURVES over the other two methods when the path resolution is higher than the transmission range. In such cases, many grid locations can only hear consecutive beacon broadcasts from a single path segment. Since a majority of the path segments of SCAN and HILBERT are straight lines, collinearity becomes a major obstacle. On the other hand, CIRCLES and S-CURVES have a minimal number of straight line segments, and collinearity is not a significant problem during localization. The results indicate the paths taken by CIRCLES and S-CURVES cover larger effective ground in terms of localization than SCAN and HILBERT. Another way to look at this is that it would take a longer path (for instance, using a smaller resolution) for SCAN and HILBERT to provide the equivalent coverage.

By increasing the transmission range (Figure 7.3(b) through 7.3(d)), the amount of collinearity is reduced, and thus we observe performance improvement in SCAN and HILBERT. In those cases, S-CURVES still performs as well as SCAN and HILBERT. However, CIRCLES has the worst performance in Figure 7.3(d). This is primarily due to the fact that we use a square as our test deployment area, and thus leave the four corners uncovered by CIRCLES. As such, we also note that CIRCLES has significantly shorter path length than the other three path types. Thus, CIRCLES is not best suited for the square or rectangle deployment area. But we can expect that it would work much better when the deployment area resembles a circle.

7.3 Dynamic Path Planning

We have introduced static path planning for localization in the previous section. Static paths work well when the sensors are assumed to be uniformly deployed. However, in the cases where such assumption is not valid, static paths might not be the best

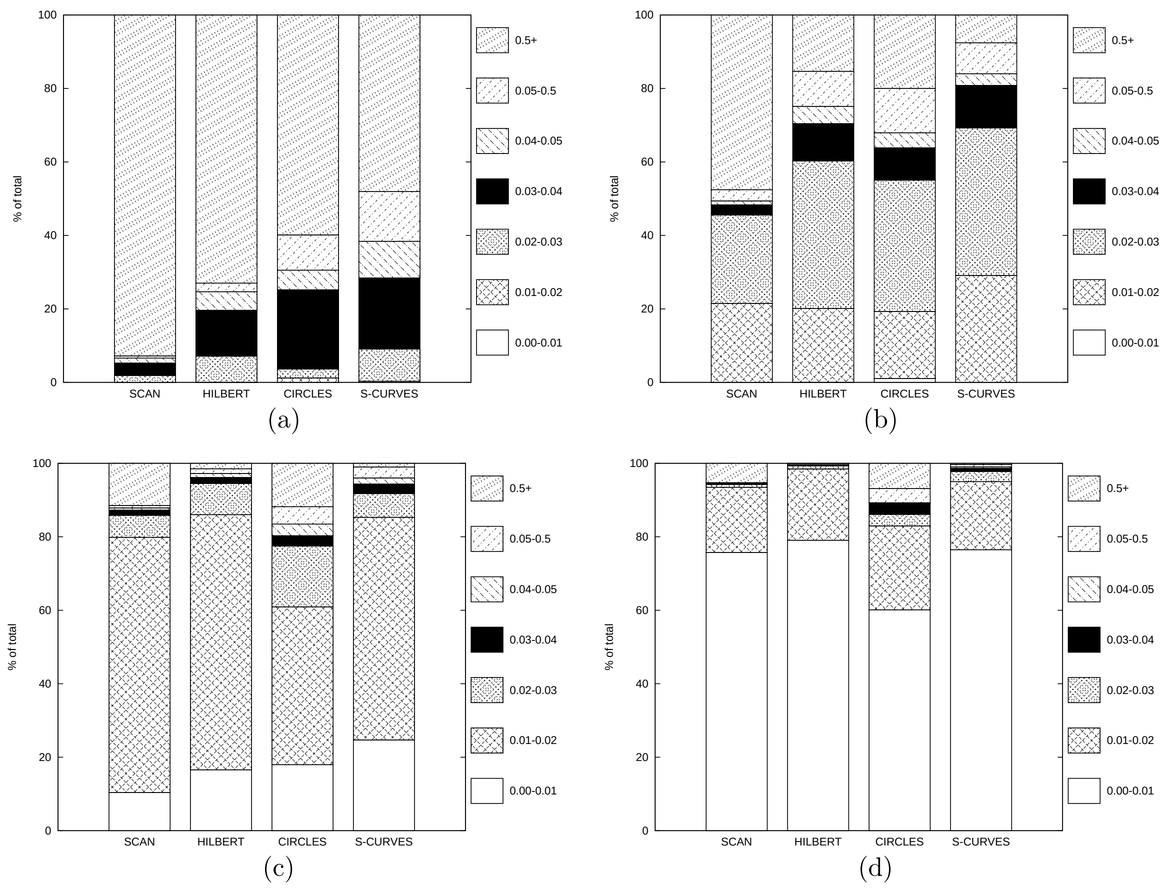


Figure 7.3. CRB ranges of the four static path types. The figure shows the CRB ranges when the transmission range is (a) 30m, (b) 45m, (c) 60m, and (d) 75m.

solution, since the mobile beacon would attempt to cover the entire deployment area uniformly, including those parts where sensors reside. Thus, there is a strong incentive to dynamically adjust the path *during* the localization procedure (*dynamic* path planning).

As in the static version, we assume that the mobile beacon broadcasts its location at a certain broadcast interval. However, instead of following a pre-determined static path, we allow the beacon to adjust its path at each broadcast interval. We also assume that the total number of sensors to be localized is known, and there exists a bi-directional communication path between the sensors and the beacon. When the sensor is close enough to hear the beacon broadcast, it is able to send an acknowledgment that contains its unique sensor ID back to the beacon. Based on such a model, the dynamic path planning problem becomes very similar to the beacon deployment problem discussed in Chapter 5 in that we will have to make an online decision on the next step based on the feedback from the previous steps. In both beacon deployment and dynamic path planning, the primary objective is to localize all the sensors. The secondary objective differs in that the beacon deployment problem tries to minimize the number of deployed beacons but the dynamic path planning problem tries to minimize the path length.

Due to the similarity between the two, we formulate the dynamic path planning problem into the model used to solve the beacon deployment problem in Chapter 6. In particular, we overlay a n-by-n virtual grid system onto the deployment area, which we assume to be a square. The mobile beacon will calculate the probability of a sensor near each grid location and then use the grid system as a guide to its dynamic path. Chapter 6 has shown that a greedy approach is quite effective in selecting the next beacon location. We use the same greedy algorithm to solve the path planning problem. However, in the case of path planning, we have to incorporate the distance to the destination into our greedy selection.

7.3.1 Greedy k-cover Path Planning: PP(k)

As in the beacon deployment problem, we first assume that it is sufficient to localize a sensor when it has heard k ($k = 3$ for ToA or RSSI ranging, and $k = 2$ for AoA) beacon broadcasts. Thus, the mobile beacon has to provide k-cover to every sensor. At each broadcast interval, the beacon has to decide on the path to take. Intuitively, the beacon should take the path to the location that would result in the largest number of beacons to be covered (i.e., maximum payoff). But unlike the beacon deployment problem, where each potential beacon location has the same cost, the cost of potential location for the path planning problem is proportional to the distance to the location. In other words, we sometimes want the mobile beacon to give up the location with the maximum payoff for a location with less payoff but with a shorter distance.

In this section, we present both the offline and online version of the greedy k-cover path planning algorithm. The offline version assumes that the sensor locations are known ahead of time. The online version has no prior knowledge of the exact sensor locations. Instead, it is given a perceived location distribution (in most cases, a uniform distribution) of where the sensors might be. The online algorithm adjusts this location distribution based on the feedback acknowledgment received by covered sensors as the beacon travels. We present the offline version here so that it can serve as the basis for comparison to the online version.

Algorithm 12 shows the offline k-cover version of the greedy algorithm. The mobile beacon starts at a corner $(0, 0)$ of the deployment square. The algorithm keeps the number of covers for each sensor s as c_s. At each broadcast location b, it performs a greedy selection of the grid location g_{max} such that g_{max} covers the maximum number of uncovered sensors while weighted against the distance needed to get there from the current location, i.e.,

$$g_{max} \leftarrow \underset{g}{\operatorname{argmax}} \frac{|\{s : c_s < k, s \dashv g\}|}{d_{g,b}}$$

Once the target grid location is selected, the beacon moves to that direction until the next broadcast interval is reached. If there are sensors being covered at this new location, c_s is updated accordingly. The process continues until all sensors are at least k-covered, i.e., $c_s >= k$ for all s.

Note that while the greedy selection g_{max} gives a direction for the beacon to travel from the present location b, it does not mean that the beacon will follow a straight line from b to g_{max}. As the beacon moves to the direction of g_{max}, it would likely cover

Algorithm 12 Greedy Offline Approximation of PP(k)

```
1: $V_{sensor}$: set of all sensors
2: $V_{grid}$: set of all grid locations
3: $d_{i,j}$: distance between $i$ and $j$
4: $b$: current beacon location, initialized to $(0,0)$
5: $t_{interval}$: broadcast interval time
6: $c_s$: the number of covers of sensor $s$
7: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
8: initialize $c_s \leftarrow 0$ for all $s \in V_{sensor}$
9: while $\exists s \in V_{sensor} : c_s < k$ do
10:    select $g_{max} \leftarrow \arg\max_{g \in V_{grid}} \frac{|\{s : c_s < k, s \dashv g\}|}{d_{g,b}}$
11:    move beacon to the direction of $g_{max}$ for $t_{interval}$ long, and update $b$
12:    for all $s : c_s < k, s \dashv b$ do
13:       $c_s \leftarrow c_s + 1$
14:    end for
15: end while
```

additional sensors along the way and have different distance measures d. Thus, at each broadcast interval, the beacon can select a different greedy location and adjust its course.

The offline algorithm knows the exact sensor locations when planning the path. Thus, the greedy selection is quite obvious. Now consider the online version, in which the sensor locations are initially unknown. Instead, for each sensor s, we maintain a probability distribution, $p_{s,g}$, of its likely location using the virtual grid system. We assume that we are unaware of the sensor location before deploying the mobile beacon, and thus the location distribution is uniform over all grid locations. As the mobile beacon moves across the deployment area and hears more sensors along the way, we update the location distribution accordingly.

Algorithm 13 shows the online version of the greedy path planning algorithm. The algorithm maintains $p_{s,g}$ as the probability of sensor s being located close to grid location g. $p_{s,g}$ is set to zero when s is successfully k-covered. Otherwise, $p_{s,g}$ is updated, with the procedure $AdjustProbability(p, s, b)$ (Algorithm 14), at each broadcast interval based on the feedback acknowledgment the beacon receives at the present beacon location b. Note that the procedure $AdjustProbability(p, s, b)$ adjusts the location distribution p based on both positive information (i.e., acknowledgment from the sensor s is heard at grid location g) and negative information (i.e., acknowledgment is *not* heard).

The online algorithm selects the greedy choice based the probability distribution p and the distance d. In particular, the algorithm evaluates each grid location g by summing up the probabilities of it covering sensors s. The greedy selection, g_{max}, is the one out of all grid locations that has the maximum of such sums:

$$g_{max} \leftarrow \arg \max_{g \in V_{grid}} \frac{\sum_{g' \dashv g} \sum_{s} p_{s,g'}}{d_{g,b}}$$

As such, the online greedy choice is a just a generalization of the offline version in which the exact sensor locations are not unknown. If the sensor locations are known, the online greedy choice becomes equivalent to that of the offline algorithm.

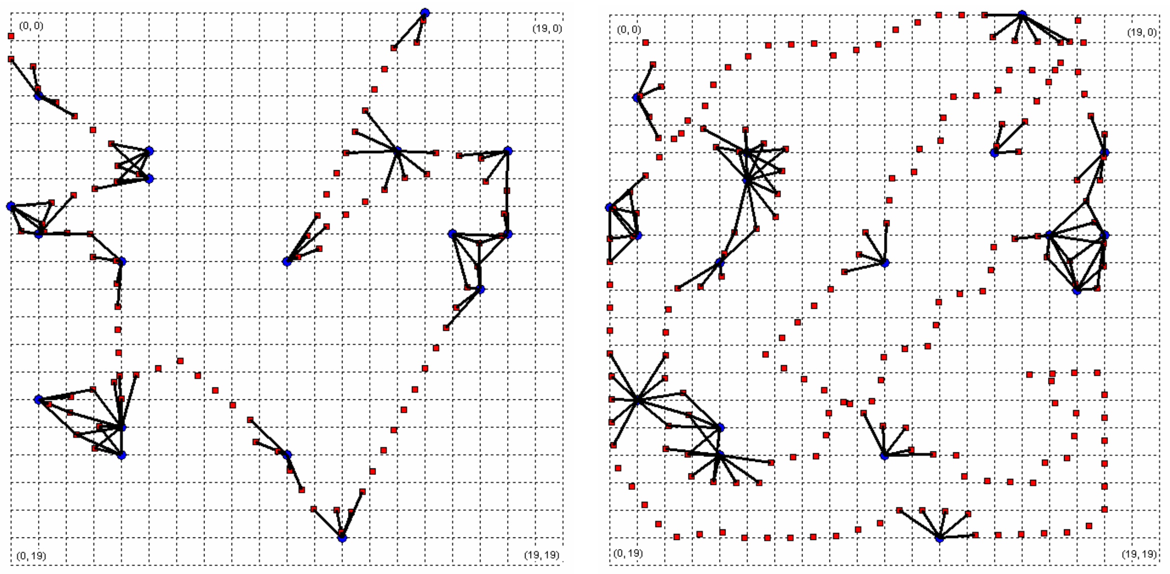
Figure 7.4 shows the results of the offline and online greedy k-cover paths of a sensor network of 20 nodes. Here, we overlay a 20-by-20 grid system and set the transmission range to be 2 times the grid side length. The larger circles in the figure indicate the sensor locations, and the smaller squares plot the trail of the mobile beacon with locations of the squares indicating the beacon broadcast locations. As expected, the offline version generates much shorter paths than the online version, since the online version has to probe for the sensors along the way.

7.3.2 Greedy CRB-Based Path Planning: PP(crb_T)

While the k-cover version of the path planning algorithm satisfies the basic requirement of localization such as $k = 3$ covers are required to localize from range observations, it does not distinguish between the effectiveness of each observation. In reality, observations differ considerably in the localization accuracy they produce. In the extreme case,

Algorithm 13 Greedy Online Approximation of PP(k)

```
1: $V_{sensor}$: set of all sensors
2: $V_{grid}$: set of all grid locations
3: $d_{i,j}$: distance between $i$ and $j$
4: $b$: current beacon location, initialized to $(0,0)$
5: $t_{interval}$: broadcast interval time
6: $p_{s,g}$: the probability of the sensor $s$ residing close the grid location $g$
7: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
8: initialize $p_{s,g} \leftarrow 1/|V_{grid}|$ for all $s \in V_{sensor}$ and $g \in V_{grid}$
9: **while** $\exists s \in V_{sensor}, g \in V_{grid} : p_{s,g} > 0$ **do**
10:    select $g_{max} \leftarrow \arg\max_{g \in V_{grid}} \frac{\sum_{g' \dashv g} \sum_{s} p_{s,g'}}{d_{g,b}}$
11:    move beacon to the direction of $g_{max}$ for $t_{interval}$ long, and update $b$
12:    broadcast beacon location and listen for acknowledgments
13:    **for all** $s \in V_{sensor}$ **do**
14:       **if** $s$ is covered by $k$ beacons **then**
15:          $p_{s,g} \leftarrow 0$ for all $g$
16:       **else**
17:          $AdjustProbability(p, s, b)$
18:       **end if**
19:    **end for**
20: **end while**
```



(a) (b)

Figure 7.4. Example of the greedy approximation of PP(k). The figure shows two cases: (a) offline and (b) online.

Algorithm 14 $AdjustProbability(p, s, b)$

```
1: $sum \leftarrow 0$
2: if $s \dashv b$ then
3:     for all $g : b \not\dashv g$ do
4:         $sum \leftarrow sum + p_{s,g}$
5:         $p_{s,g} \leftarrow 0$
6:     end for
7:     $I \leftarrow$ set of all $g : g \dashv b, p_{s,g} > 0$
8:     for all $g \in I$ do
9:         $p_{s,g} \leftarrow p_{s,g} + sum/|I|$
10:    end for
11: else
12:    for all $g : b \dashv g$ do
13:        $sum \leftarrow sum + p_{s,g}$
14:        $p_{s,g} \leftarrow 0$
15:    end for
16:    $I \leftarrow$ set of all $g$ that are *not* in range of $b$ and $p_{s,g} > 0$
17:    for all $g \in I$ do
18:        $p_{s,g} \leftarrow p_{s,g} + sum/|I|$
19:    end for
20: end if
```

the three ranging observations might be collinear, which would render the sensor unlocalizable. Chapter 6 introduced the concept of using Cramer Rao Bounds (CRB) as the evaluation criteria of selecting the next beacon location. In particular, instead of merely requiring k-cover, we require every sensor to be covered by enough beacon broadcasts so that the CRB of the location estimate reduces below a pre-defined threshold crb_T. Compared to k-cover, the CRB-based evaluation criteria explicitly considers the estimation accuracy during localization, and thus allows a better trade-off between the localization cost and the localization accuracy.

As with the k-cover version, we provide both the offline and the online algorithm of the path planning algorithm using crb_T. For the offline algorithm (Algorithm 15), since we know the exact sensor locations, the CRB of each sensor, $CRB(s)$, can be readily calculated from the relative beacon broadcast locations, the signal propagation model and the noise model. Before localization, we initialize $CRB(s)$ to infinity. As the sensors receive the beacon broadcasts while the beacon moves, the CRB is progressively reduced. The algorithm stops once the CRBs of all sensors have reduced below the threshold crb_T. At each broadcast interval, the mobile beacon picks the direction toward the grid location that would maximize the CRB reduction if the beacon broadcast were to originate from there. For any sensor s, $CRB(s)$ gives its current CRB value, whereas $CRB^P(s,g)$ denotes the potential CRB of sensor s when adding a beacon broadcast from grid location g. Thus, the greedy selection, g_{max}, is the grid location that maximizes $CRB^P(s,g) - CRB(s)$ for all sensors while considering the distance required to move there

$$g_{max} \leftarrow \arg \max_{g \in V_{grid}} \frac{\sum_{s:s \dashv g} CRB^P(s,g) - CRB(s)}{d_{g,d}}$$

Algorithm 15 Greedy Offline Approximation of PP(crb_T)

```
 1: $V_{sensor}$: set of all sensors
 2: $V_{grid}$: set of all grid locations
 3: $d_{i,j}$: distance between $i$ and $j$
 4: $b$: current beacon location, initialized to $(0,0)$
 5: $t_{interval}$: broadcast interval time
 6: $CRB(s)$: CRB of the sensor $s$
 7: $CRB^P(s,g)$: Potential CRB of the sensor $s$ when adding a beacon broadcast at $g$
 8: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
 9: initialize $CRB(s) \leftarrow$ inf for all $s \in V_{sensor}$
10: while $\exists s \in V_{sensor} : CRB(s) > crb_T$ do
11:     $g_{max} \leftarrow \arg\max_{g \in V_{grid}} \frac{\sum_{s:s \dashv g} CRB^P(s,g) - CRB(s)}{d_{g,d}}$
12:     move beacon to the direction of $g_{max}$ for $t_{interval}$ long, and update $b$
13:     for all $s : s \dashv b$ do
14:         re-calculate $CRB(s)$
15:     end for
16: end while
```

The online version (Algorithm 16) deals with the uncertainty of the sensor location with a probability distribution over the grid system. Let $p_{s,g}$ be the probability of sensor s residing close the grid location g, which is initialized uniformly to $1/|V_{grid}|$ for each sensor s at each grid location g. $p_{s,g}$ is adjusted based on the positive and negative feedback acknowledgment heard by the beacon at each broadcast interval using the same *AdjustProbability* procedure. The greedy selection of the grid location is also based on maximizing the CRB reduction. However, since the exact sensor location is not known, the CRB has to be calculated by assuming a particular sensor location. Here, let $CRB_{g'}(s)$ be the CRB of sensor s assuming it's located at grid location g', and $CRB^{P}_{g'}(s, g)$ be the potential CRB of sensor s, assumed to be located at g', when adding a beacon broadcast from g. The CRB reduction based on the assumed sensor location, $CRB^{P}_{g'}(s, g) - CRB_{g'}(s)$, is then factored by the probability, $p_{s,g'}$, that such an assumption is true. The grid location that would generate the maximum sum of the CRB reduction for all sensors is selected as the next grid location:

$$g_{max} \leftarrow \arg \max_{g \in V_{grid}} \frac{\sum_{g' \dashv g} \sum_{s} p_{s,g'} \cdot (CRB^{P}_{g'}(s, g) - CRB_{g'}(s))}{d_{g,b}}$$

7.3.3 Simulations

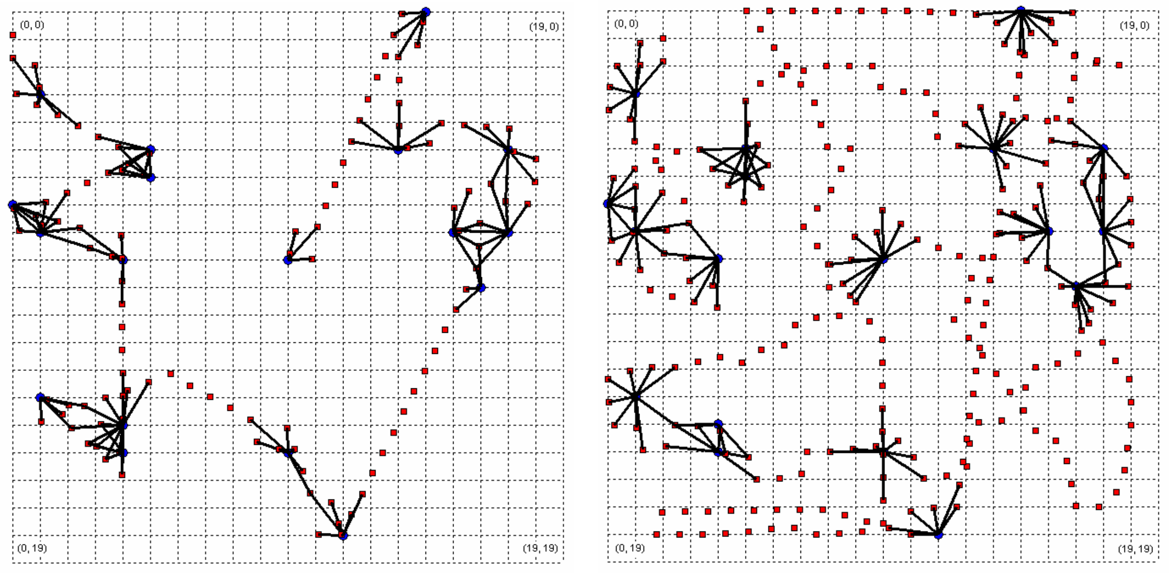
We have simulated the performance of the dynamic path planning methods in terms of overall path length and localization error. To compare dynamic paths with static paths, we have also implemented the S-CURVES static path type (Section 7.2.3.4), which we have shown using CRB analysis in Section 7.2.4 to be superior to other proposed static path types. The simulated deployment area is a 480m by 480m square. The transmission range of the beacon and the sensors is set to 45m, and the broadcast interval of the beacon is set to 20m per broadcast. Within the deployment area, we consider two types

Algorithm 16 Greedy Online Approximation of BD(crb_T)

```
1: $V_{sensor}$: set of all sensors
2: $V_{grid}$: set of all grid locations
3: $d_{i,j}$: distance between $i$ and $j$
4: $b$: current beacon location, initialized to $(0,0)$
5: $t_{interval}$: broadcast interval time
6: $CRB_g(s)$: CRB of the sensor $s$ assuming it's located at grid location $g$
7: $CRB^P_{g'}(s,g)$: Potential CRB of the sensor $s$, assumed to be located at $g'$, when adding
   a beacon broadcast at $g$
8: $p_{s,g}$: the probability of the sensor $s$ residing close the grid location $g$
9: operator $i \dashv j$: location $i$ can be covered by broadcast from location $j$
10: initialize $p_{s,g} \leftarrow 1/|V_{grid}|$ for all $s \in V_{sensor}$ and $g \in V_{grid}$
11: while $\exists s \in V_{sensor}, g \in V_{grid} : p_{s,g} > 0, CRB_g(s) > crb_T$ do
12:   select $g_{max} \leftarrow \arg\max_{g \in V_{grid}} \frac{\sum_{g' \dashv g} \sum_s p_{s,g'} \cdot (CRB^P_{g'}(s,g) - CRB_{g'}(s))}{d_{g,b}}$
13:   move beacon to the direction of $g_{max}$ for $t_{interval}$ long, and update $b$
14:   broadcast beacon location and listen for acknowledgments
15:   for all $s$ do
16:     $AdjustProbability(p, s, b)$
17:     if $s : s \dashv b$ then
18:       re-calculate $CRB_{g'}(s)$ for all $g' \dashv b, p_{s,g'} > 0$
19:     end if
20:   end for
21: end while
```

of sensor locations: uniform and clustered. The uniform type assumes that the sensors are deployed uniformly, whereas the clustered type assumes that the sensor are deployed to form a number of clusters. The latter can be viewed as a more realistic model in the situation where the sensors are dropped from an airplane. To measure the localization error, we assume that RSSI ranging is used. We use the same signal model as in Section 7.2.3. As in [66], we choose $n_p = 1$ and $\sigma^2_{dB} = 2.5$ (unless otherwise noted). The results are the average of 30 independent runs with the 95% confidence interval plotted as the vertical error bars.

We first obtain some preliminary results on effect of node density, measurement noise and grid size using uniform sensor networks.

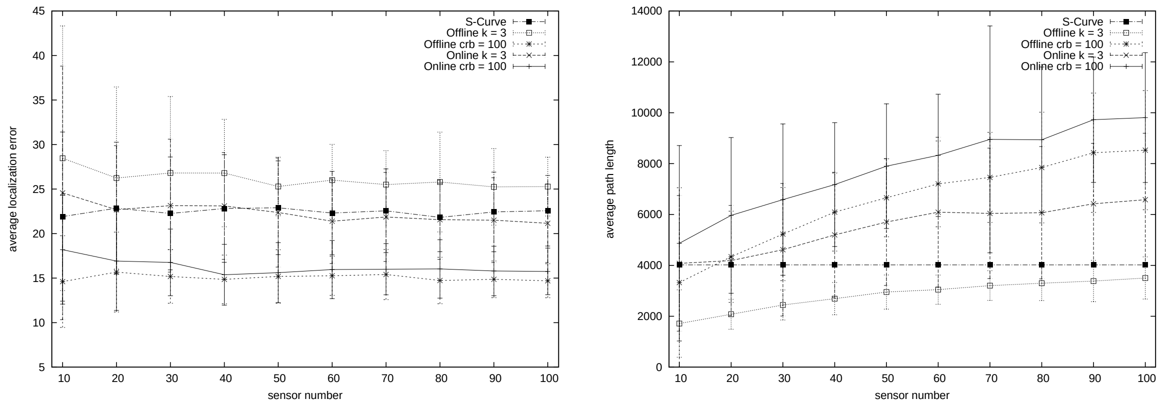


(a) (b)

Figure 7.5. Example of the greedy approximation of PP(crb_T). The figure shows two cases: (a) offline and (b) online.

7.3.3.1 Sensor Density

Intuitively, the static paths should work better when there are a large number of uniformly deployed sensors, since the mobile beacon has to cover virtually every part of the network anyway. This is validated by Figure 7.6. Here, we assume a transmission range of 45m and a grid size of 30x30. As we increase the sensor number from 10 to 100, Figure 7.6(a) shows that the average localization error stays relatively the same. However, the path length (Figure 7.6(b)) increases for the four dynamic path planning methods. Not surprisingly, the two online versions have longer path length than the corresponding offline versions. The CRB-based versions generate better localization results at the expense of longer path length than the k-covered versions.

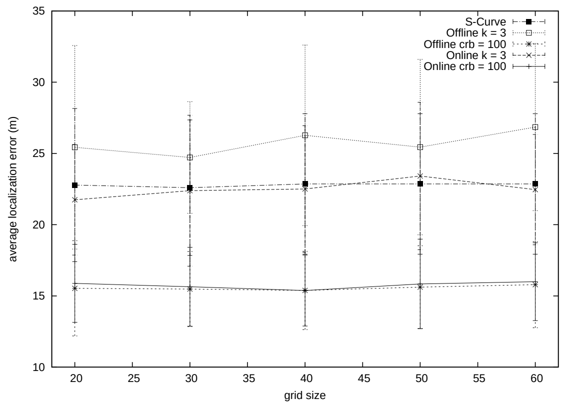


(a) (b)

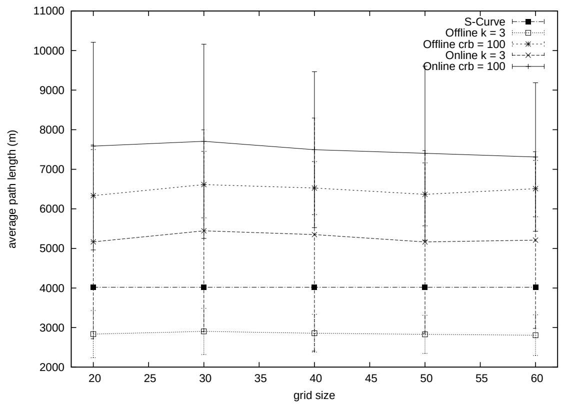
Figure 7.6. Effect of varying node density of uniformly deployed sensor network. The figure shows (a) localization error and (b) path length.

7.3.3.2 Grid Size

Since the dynamic path planning algorithms run on a virtual grid system, we are interested in how the grid size would affect the performance. In Figure 7.7, we measure the localization error and path length to localize 50 sensors while changing the grid size from 20-by-20 to 60-by-60. The results indicate that a relatively small grid system is sufficient for the path planning purpose, as the grid system serves only as a guideline to the dynamic paths. Since the actual broadcast interval is sufficiently small (at 20m per broadcast), the mobile beacon is able to produce a large enough number of broadcasts to satisfy the localization criteria regardless of the grid size. Since the grid size greatly impacts the computational complexity of the greedy selection, it is desirable to use a smaller grid size to speed up the computation. We thus use a 30-by-30 grid size in our following experiments.



(a)



(b)

Figure 7.7. Effect of varying grid size of uniformly deployed sensor networks. The figure shows (a) localization error and (b) path length.

7.3.3.3 Measurement Noise

The amount of the RSSI measurement noise directly impacts the localization accuracy. Furthermore, it can also impact the path length of the CRB-based dynamic path planning algorithms. More noise would cause higher CRB value; to reduce the CRB to the predefined threshold, additional beacon broadcast locations would be required, which in turn increases the path length. Figure 7.8 shows the results of the experiment of varying the noise parameter σ_{dB} from 1 to 3 based on the signal and noise model of Equation 7.1. The results indicate an increase in the localization error as σ_{dB} is increased. Among the five algorithms, the CRB-based algorithms have the smallest error increase, but they also cause longer path lengths. Comparably, the measurement noise has no impact on the path lengths of the k-cover algorithms, but their localization error increases more rapidly.

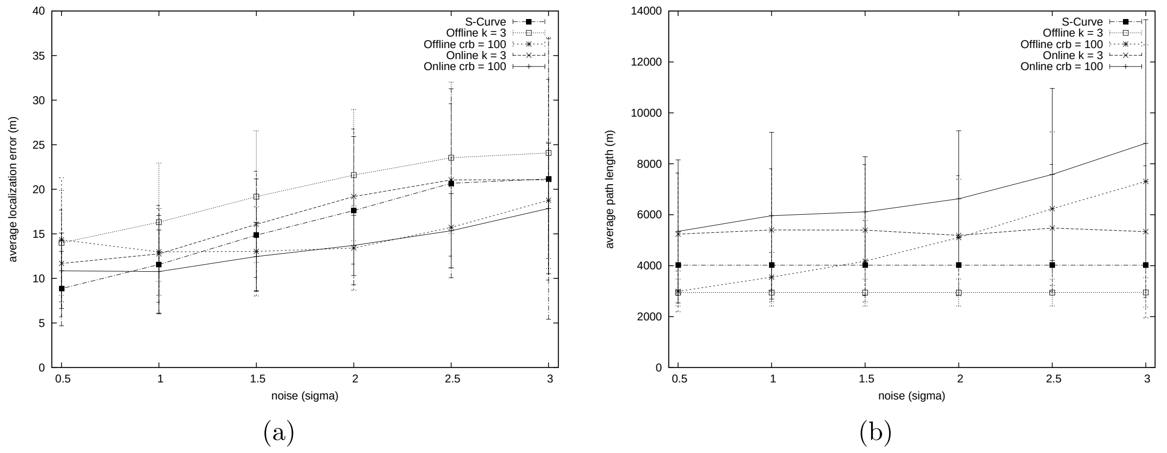


Figure 7.8. Effect of varying noise parameter for uniformly deployed sensor networks. The figure shows (a) localization error and (b) path length.

7.3.3.4 Clustered Sensor Networks

When an airplane drops a large number of sensors into a field, it is very unlikely that those sensors form a uniform distribution. More realistically, these sensor networks would exhibits a certain clustering behavior [28, 53]. To simulate the clusters formed by the sensors, we use the Neyman-Scott Poisson model, provided by the *spatstat* package implemented in CRAN [15], to generate clustered sensors. The Neyman-Scott Poisson model generates the sensors in two steps. First, it generates a Poisson distributed parent event with a certain intensity to represent the locations of the clusters. For each cluster location, it then independently produces a Poisson number of sensors. In the version provided by CRAN, the Neyman-Scott model needs three parameters: the parent intensity λ_c (i.e., the mean number of clusters), the maximum cluster radius r_{max} , and the child intensity λ_s (i.e., the mean number of sensors per cluster). In our experiments, we use two types of sensor networks based on the Neyman-Scott model. The first type uses $\lambda_c = 5$, $r_{max} = 96m$ and $\lambda_s = 25$ representing moderately clustered networks, and the second type uses $\lambda_c = 1$, $r_{max} = 144m$ and $\lambda_s = 75$ representing highly clustered

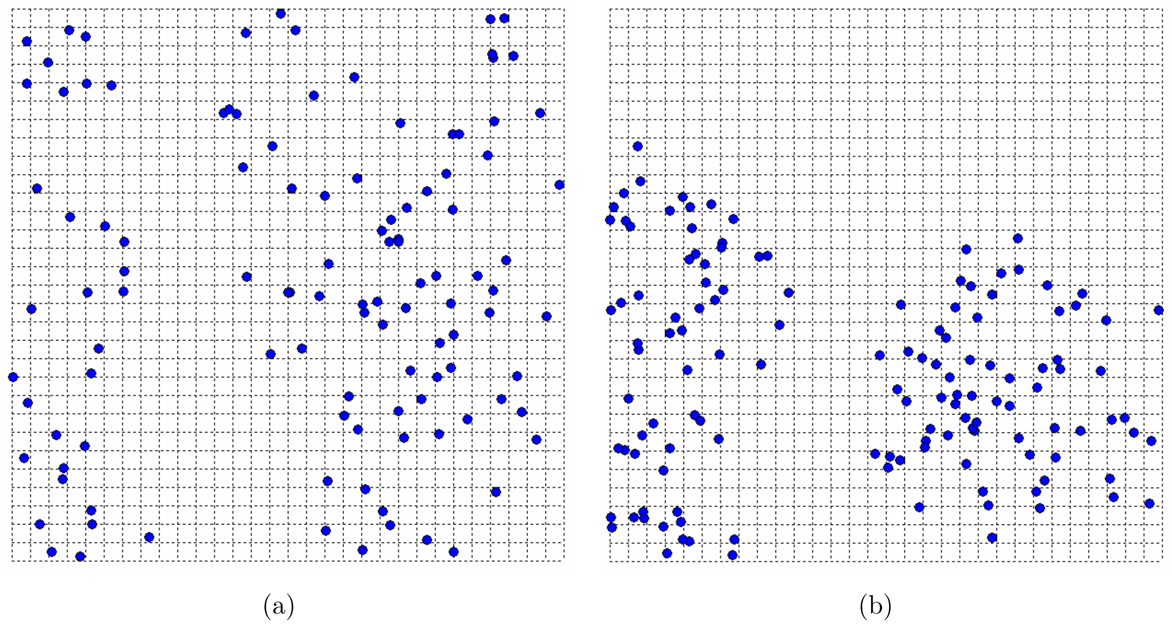
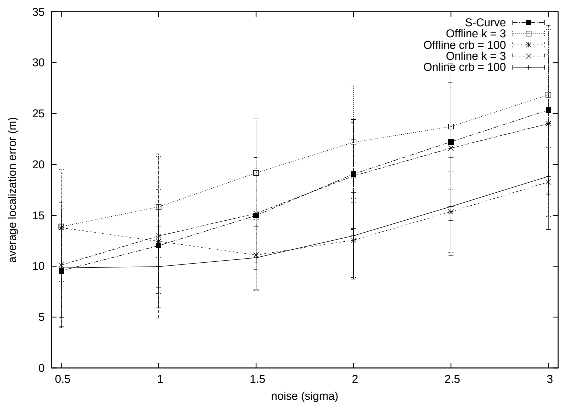


Figure 7.9. Two types of cluster based sensor networks. The figure shows (a) lightly clustered ($\lambda_c = 5$, $r_{max} = 96m$ and $\lambda_s = 25$) and (b) heavily clustered ($\lambda_c = 1$, $r_{max} = 144m$ and $\lambda_s = 75$).

networks. Figure 7.9 shows a sample sensor distribution of each type. Note that in our simulation we generate a different network for each run and average the results.

We repeated the experiment of varying the noise parameter σ_{dB} from 1 to 3 for the two types of clustered sensor networks, and the results are shown in Figure 7.10 and 7.11. While the characteristics of localization error stay relatively static, we observe that the path lengths are reduced compared to the case of that of uniform sensor networks. In particular, comparing Figure 7.10(b) to 7.11(b), the path lengths are significantly shorter when the sensors are more clustered. This indicates that dynamic path planning algorithms are better suited for the clustered scenarios, in which the algorithms can adjust the path to concentrate on the clustered areas. In the case of uniform sensor networks, the dynamic path planning algorithms do not have advantage over the static paths, since the entire deployment area has to be covered with equal importance.


(a)

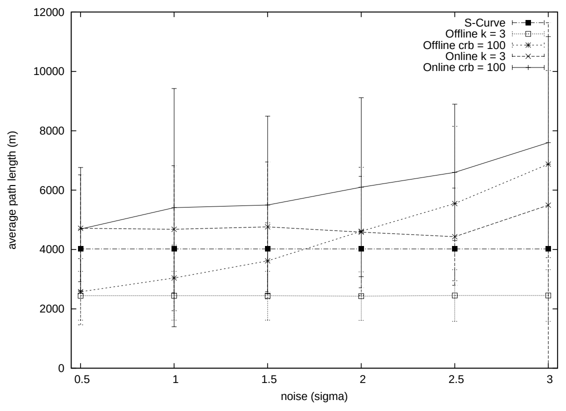
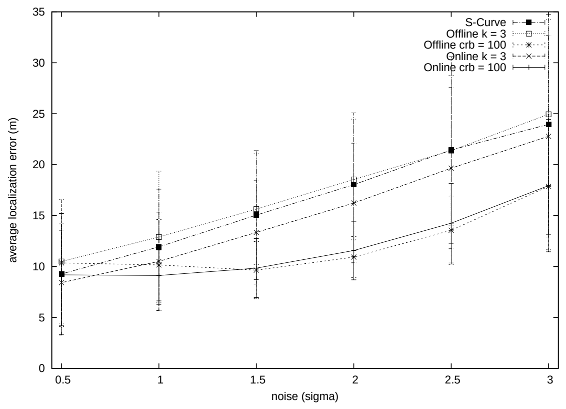

(b)

Figure 7.10. Effect of varying noise parameter for lightly cluster based sensor networks. The figure shows (a) localization error and (b) path length.


(a)

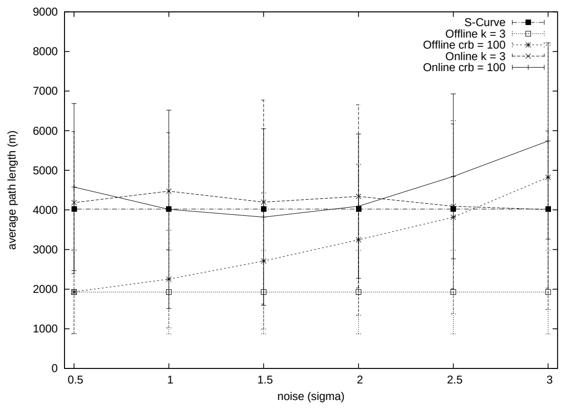

(b)

Figure 7.11. Effect of varying noise parameter for heavily cluster based sensor networks. The figure shows (a) localization error and (b) path length.

7.4 Summary

In this chapter, we studied the path planning problem for localization in sensor networks. The path planning problem asks for good paths, in terms of maximizing the localization accuracy while minimizing the path length, that will guide a mobile beacon through the deployment area. We proposed two types of paths: static paths and dynamic paths. In the case of static paths, we presented two new path types: CIRCLES and S-CURVES that are designed with better localization accuracy and coverage in mind. We have also proposed a new path evaluator using the unbiased Cramer Rao Bounds (CRB). Using the CRB analysis, we showed that the path types proposed by us handle the collinearity problem much better. When the path resolution is much larger than the transmission range, the collinearity becomes more significant. In such cases, our solutions significantly outperform previously proposed path types.

The static paths work well when sensors are assumed to be uniformly deployed. However, often in the real world sensors are deployed in clusters. Thus, it is desirable to design dynamic path planning algorithms capable of adjusting the path such that the beacon would spend more time in the area where the sensors are more heavily populated. Following the same approach introduced in Chapter 6, we used a greedy strategy to the dynamic path planning problem that provides either k-cover or CRB-based threshold to each sensor. We used simulations to show that the CRB-based method provides better localization accuracy at the expense of longer paths. We have also validated the claim that dynamic paths are more preferable in heavily clustered sensor networks.

CHAPTER 8

CONCLUSIONS

In this chapter, we provide a brief review of the topics and contributions covered by this work. We then list a number of open problem in the area of localization in mobile ad hoc networks.

8.1 Review

The scope of this work has been the localization problem in mobile ad hoc networks (MANET). The term of localization in the domain of MANET refers to the problem of finding the location of network nodes. In Chapter 1, we have established the need for localization by presenting an array of proposed MANET protocols that would not work without location information. However, localization is difficult because of its fundamental intractability based on the underlying graph theoretic model. To further complicate the problem, measurement data is not always available, and when it is indeed available it is often prone to noise. Localization techniques also differ considerably depending on the type of the measurement data, and thus a common framework is needed that works for multiple measurement types. Under this general scope of localization, we have studied a number of sub-problems including collaborative localization techniques (Chapter 2), localization with the new interferometric ranging techniques (Chapter 3), link longevity prediction problem (Chapter 4), beacon deployment problem (Chapter 5), localization using a mobile beacon (Chapter 6), and the path planning problem for mobile beacons (Chapter 7).

Unlike most existing localization techniques, which often treat the estimated locations as definite solutions, we have taken a probabilistic approach that explicitly considers the underlying location uncertainty. For instance, we used a particle filtering solution for the general collaborative localization in Chapter 2 and for the localization using a mobile beacon in Chapter 6, where the estimated location is not a singular value but a probabilistic distribution represented by particles. This approach allows us to integrate collaborative localization by simply exchanging particle distributions among the neighbors. The variance of the location distribution also gives a useful measure to the estimated accuracy, which can then be applied to satisfy different localization requirements. We have shown that the same probabilistic framework works for various measurement types such as RSSI and ToA ranging, AoA, and connectivity-only even in the case where different measurement types co-exist in the same network.

We have examined localization using interferometric ranging techniques in Chapter 3. Using computational complexity theories, we have shown that, like distance ranging, angling, and unit disk connectivity, localization using interferometric ranging is also NP-Complete. This demonstrates the fundamental intractability of the problem. Based on the results from our iterative localization algorithm, we have shown that it is also difficult to optimize in a collaborative manner due to error propagation. We studied the link longevity problem in Chapter 4, where we have presented three types of measurement based prediction methods using extended Kalman filters. While this problem does not deal with absolute localization, it does implicitly cover the node mobility issue applicable to routing (i.e., route maintenance).

We applied a similar probabilistic approach to the dynamic beacon deployment problem in Chapter 5 and the dynamic path planning problem in Chapter 7. We have shown that the beacon deployment problem is likely to be NP-Complete, and that there is a greedy offline approximation with a logarithmic approximation ratio. However, in order to apply the greedy approximation in an online fashion, we would have to consider the possible sensor locations, which we accomplished by using a probability distribution based on a grid system. We have explicitly considered the localization accuracy in our design. In particular, we have taken the theoretic Cramer Rao Bound (CRB), which had previously only been used as a passive evaluation tool, and incorporated it into our algorithm design. The CRB-based algorithms allow us to provide a better trade-off between accuracy and cost during localization to various applications.

8.2 Open Problems

Most results of this work are based on modeling and simulations. Although we have made every attempt to model our simulations as carefully as possible, they still cannot replace experiments of physical devices in real world scenarios. However, this problem is not unique to us. In fact, the majority of works in localization have been based on either theoretic models or simulations. The primary reason is the cost. To perform meaningful experiments for localization, one would normally need a large number (100+) of devices. Although mobile ad hoc network devices (for instance, sensors) are becoming cheaper by the day, it is still quite costly to implement them on physical devices. In addition, the sensing capacity of the current devices are usually limited to RSSI. There is no cheap hardware that implements AoA, ToA, or interferometric ranging, and thus most works using those measurement types are all based on simulations. In a sense, the advances in algorithmic work on the localization problem are currently outgrowing the advances in hardware. Future work needs to be done to significantly improve the hardware design to fill this gap.

Another issue related to the testing environment is that there is no common localization testbed. While NS2 [63] has a module for simulating mobile ad hoc networks, the module does not contain localization. It would be very helpful for researchers to implement an interface that allows a “plug-in” for future localization algorithms. This would not only give a common testbed for different localization schemes, it would also allow those location-depended algorithms (such as location-aided routing methods) to be implemented as other NS2 modules and compared based on the result of localization. The localization module should also implement the Cramer Rao Bounds(CRB) so that the theoretic error bound can be calculated for different localization scenario.

Because interferometric ranging is a relatively new type of measurement available to the localization problem, there are still many open problems in this area. Of the localization algorithms proposed for interferometric ranging, all but the iterative algorithm proposed in this work are centralized. There is a definite need to design distributed localization algorithms for interferometric ranging so that it can be implemented with reasonable efficiency and scalability. To reduce the number of beacons, the distributed algorithms should make use of multi-hop location information, which unfortunately is much more difficult for interferometric ranging because each measurement involves four nodes. Furthermore, our simulations have shown that the multi-hop error propagation has a big impact on interferometric ranging. Therefore, the control of the error propagation is another issue. There is also a need for a theoretic error bound (like CRB) for interferometric ranging. This however may be too difficult because more than two nodes are involved in each measurement.

REFERENCES

[1] A. A. Ahmed, H. Shi, and Y. Shang. "SHARP: A New Approach to Relative Localization in Wireless Sensor Networks," in Proc. of *25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2005)*, pp. 892-898, June 2005.

[2] J.N. Ash and L.C. Potter. "Sensor Network Localization via Received Signal Strength Measurements with Directional Antennas," in Proc. of *2004 Allerton Conf. Communication, Control, and Computing*, pp. 1861-1870, 2004.

[3] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur. "A Theory of Network Localization," to appear in *IEEE Transactions on Mobile Computing.*

[4] A. Berg and T. Jordan. "A Proof of Connelly's Conjecture on 3-connected generic Cycles," *Journal of Combinatorial Theory Series B*, vol. 88(1), pp. 77-97, May 2003.

[5] P. Biswas and Y. Ye. "Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization," in Proc. of *3rd International Symposium on Information Processing in Sensor Networks (IPSN 2004)*, pp. 46-54, Berkeley, California, USA, 2004.

[6] L. Blazevic, J.-Y. Le Boudec, and S. Giordano. "A Location Based Routing Method for Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 4(2), pp. 97-110, Mar-Apr 2005.

[7] H. Breu and D. G. Kirkpatrick. "Unit Disk Graph Recognition is NP-hard," *Computational Geometry. Theory and Applications*, vol. 9(1-2), pp. 3-24, 1998.

[8] J. Bruck, J. Gao, and A. Jiang. "Localization and Routing in Sensor Networks by Local Angle Information," in Proc. of *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 181-192, Urbana-Champaign, IL, USA, 2005.

[9] N. Bulusu, J. Heidemann, and D. Estrin. "Adaptive Beacon Placement," in Proc. of *ICDCS-21*, pp. 489-498, April 2001.

[10] N. Bulusu, J. Heidemann, V. Bychkovskiy, and D. Estrin. "Density-adaptive Beacon Placement Algorithms for Localization in Ad Hoc Wireless Networks," in Proc. of *IEEE INFOCOM'02*, New York, NY, 2002.

[11] N. Bulusu, J. Heidemann, and D. Estrin. "GPS-less Low Cost Outdoor Localization for Very Small Devices," *IEEE Personal Communications Magazine*, vol. 7/5, pp. 28-34, October 2000.

[12] S. Capkun, M. Hamdi, and J.-P. Hubaux. "GPS-Free Positioning in Mobile Ad-Hoc Networks," in Proc. of *34th Hawaii International Conference on System Sciences*, vol. 9, pp. 9008, 2001.

[13] A Catovic and Z Sahinoglu. "The Cramer-Rao Bounds of Hybrid TOA/RSS and TDOA/RSS Location Estimation Schemes," *IEEE Communications Letters*, vol. 8(10), Oct. 2004.

[14] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal. "Ad-Hoc Localization Using Ranging and Sectoring," in Proc. of *IEEE INFOCOM '04*, pp. 2662-2672, April 2004.

[15] *The Comprehensive R Archive Network*, Internet Resource. http://cran.r-project.org/.

[16] P. Corke, R. Peterson, and D. Rus. "Networked Robots: Flying Robot Navigation with a Sensor Net," in Proc of *Int. Symp. Robotics Research*, vol. 15, pp. 234-243, Nov 2003.

[17] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, MIT Press, 1990.

[18] T. Eren, D. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur. "Rigidity, Computation, and Randomization of Network

Localization," in Proc. of *IEEE INFOCOM'04*, vol. 4, pp. 2673-2684, Hong Kong, China, Apr. 2004.

[19] U. Feige. "A Threshold of $ln(n)$ for Approximating Set Cover," *Journal of the ACM*, vol. 45, pp. 634-652, 1998.

[20] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in Proc. of *National Conference on Artificial Intelligence*, Orlando, FL, July 1999.

[21] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem. "Distributed Online Localization in Sensor Networks Using a Moving Target," in Proc. of *3rd International Symposium on Information Processing in Sensor Networks (IPSN 2004)*, pp. 61-70, Berkeley, California, USA, 2004.

[22] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, 1979.

[23] A. Georgiev and P. K. Allen. "Localization Methods for a Mobile Robot in Urban Environments," *IEEE Transactions on Robotics*, vol. 20(5), pp. 851-864, October 2004.

[24] L. Girod and D. Estrin. "Robust Range Estimation using Acoustic and Multimodal Sensing," in Proc. of *IEEE IROS'01*, vol. 3, pp. 1312-1320, Maui, Hawaii, October 2001.

[25] D. Goldenburg, W. Krishnamurthy, A. Maness, Y. Yang, and A. Young. "Network Localization in Partially Localizable Networks," in Proc. of *IEEE INFOCOM'05*, vol. 1, pp. 313-326, 2005.

[26] N. Gordon. "Bayesian Methods for Tracking," Ph.D. thesis, University of London, 1993.

[27] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. "Range-Free Localization Schemes in Large Scale Sensor Networks," in Proc. of *ACM MOBICOM'03*, pp. 81-95, 2003.

[28] N. Heo and P.K. Varshney. “An Intelligent Deployment and Clustering Algorithm for a Distributed Mobile Sensor Network,” in Proc. of *IEEE International Conference on Systems, Man and Cybernetics 2003*, vol. 5, pp. 4576-4581, October 2003.

[29] B. Hendrickson. “Conditions for Unique Graph Realizations,” *SIAM Journal on Computing*, vol. 21(1), pp. 65-84, 1992.

[30] L. Hu and D. Evans. “Localization for Mobile Sensor Networks,” in Proc. of *ACM MOBICOM’04*, pp. 45-57, 2004.

[31] Y.-C. Hu, A, Perrig, and D. Johnson. “Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks,” in Proc. of *INFOCOM’03*, vol. 3, pp. 1976-1986, April 2003.

[32] R. Huang, G.V. Záruba, and M. Huber. “Link Longevity Kalman-Estimator for Ad Hoc Networks,” in Proc. of *IEEE Vehicular Technology Conference (VTC 2003)*, vol. 5, pp. 2819-2823, Orlando, FL, October 2003.

[33] R. Huang and G. V. Záruba. “Location Tracking in Mobile Ad Hoc Networks Using Particle Filters,” in Proc. of *ADHOC-NOW’05*, pp. 85-98, 2005.

[34] M. Isard and A. Blake. “Contour Tracking by Stochastic Propagation of Conditional Density,” in Proc. of *4th European Conference on Computer Vision*, vol. 1, pp. 343-356, UK, 1996.

[35] V. Isler. “Placement and Distributed Deployment of Sensor Teams for Triangulation Based Localization,” in Proc. of *IEEE ICRA’06*, pp. 3095-3100, May, 2006.

[36] S. Jiang, D. He, and J. Rao, “A Prediction-based Link Availability Estimation for Mobile Ad Hoc Networks,” in Proc. of *Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1745-1752, 2001.

[37] X. Jiang and T. Camp. “Review of Geocasting Protocols for a Mobile Ad Hoc Network,” in Proc. of the *Grace Hopper Celebration (GHC)*, 2002.

[38] D. Johnson. *Cramer-Rao Bound*, Internet article: http://cnx.org/content/m11266/latest.

[39] D. S. Johnson, "The NP-completeness Column: An Ongoing Guide," *J. Algorithms*, vol. 3(2), pp. 182-195, June, 1982.

[40] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME – Journal of Basic Engineering*, pp. 35-45, March 1960.

[41] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing," *Science*, vol 220(4598), pp. 671-680, May, 1983.

[42] Y. Ko and N. H. Vaidya. "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," *Wireless Networks*, vol. 6(4), pp. 307-321, July, 2000.

[43] Y. Ko and N. Vaidya. "Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms," in Proc. of *IEEE WMCSA'99*, pp. 101, 1999.

[44] D. Koutsonikolas, S. M. Das, and Y. C. Hu. "Path Planning of Mobile Landmarks for Localization in Wireless Sensor Networks," in Proc. of *IEEE Distributed Computing Systems Workshops*, pp. 86-86, July, 2006.

[45] F. Kuhn, T. Moscibroda, and R. Wattenhofer. "Unit Disk Graph Approximation," in Proc. of *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pp. 17-23, Philadelphia, PA, USA, 2004.

[46] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. "Geometric Ad-Hoc Routing: Of Theory and Practice," in Proc. of *ACM PODC'03*, pp. 63-72, 2003.

[47] B. Kusý, M. Maróti, G. Balogh, P. V olgyesi, J. Sallai, A. Nádas, A. Lédeczi, and L. Meertens. "Node Density Independent Localization," in Proc. of *5th International Conference on Information Processing in Sensor Networks (IPSN 2006)*, pp. 441-448, April 2006.

[48] D. Kurth. "Range-Only Robot Localization and SLAM with Radio," Master's Thesis, Tech. Report CMU-RI-TR-04-29, Robotics Institute, Carnegie Mellon University, May, 2004.

[49] K. Langendoen and N. Reijers. "Distributed Localization in Wireless Sensor Networks: a Quantitative Comparison," *Computer Networks*, vol 43(4), pp. 499-518, Nov. 2003.

[50] E.G. Larsson. "Cramer-Rao Bound Analysis of Distributed Positioning in Sensor Networks," *IEEE Signal Processing Letters*, vol. 11(3), pp. 334-337, Mar. 2004.

[51] W.-H. Liao, Y.-C. Tseng, and J.-P. Sheu. "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks," *Telecommunication Systems*, vol. 18(1), pp. 37-60, 2001.

[52] W.-H. Liao, Y.-C. Tseng, K.-L. Lo, and J.-P. Sheu. "Geogrid: A Geocasting Protocol for Mobile Ad Hoc Networks Based on Grid," *Journal of Internet Technology*, vol. 1(2), pp. 23-32, 2000.

[53] C. Liu, K. Wu, Y. Xia, and B. Sun. "Random Coverage with Guaranteed Connectivity: Joint Scheduling for Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17(6), pp. 562-575, June 2006.

[54] T.Liu, P. Bahl, and I. Chlamtac. "A Hierarchical Position-Prediction Algorithm for Efficient Management of Resources in Cellular Networks," in Proc. of *IEEE GLOBECOM'97*, vol. 2, pp. 982-986, Phoenix, AZ, November 1997.

[55] M. Maróti, B. Kusý, G. Balogh, P. V olgyesi, A. Nádas, K. Molnár, S. Dóra, and A. Lédeczi. "Radio Interferometric Geolocation," in Proc. of *ACM 3rd Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 1-12, Nov. 2005.

[56] M. Mauve, H. Fuler, J. Widmer, and T. Lang. "Position-Based Multicast Routing for Mobile Ad-Hoc Networks," *Technical Report TR-03-004, Department of Computer Science, University of Mannheim*, 2003.

[57] A. B. McDonald and T. Znati. "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks," *IEEE Journal on Selected Areas in Communication Special Issue on Wireless Ad-Hoc Networks*, vol. 17, no. 8, August 1999.

[58] D. Niculescu and B. Nath. "Ad Hoc Positioning System (APS)," in Proc. of *IEEE GLOBECOM'01*, vol. 5, pp. 2926-2931, San Antonio, 2001.

[59] D. Niculescu and B. Nath. "DV Based Positioning in Adhoc Networks," *Telecommunication Systems*, vol. 22, no. 1-4, pp. 267-280, January-April 2003.

[60] D. Niculescu and B. Nath. "Ad Hoc Positioning System (APS) Using AoA," in Proc. of *IEEE INFOCOM'03*, vol. 3, pp. 1734-1743, San Francisco, 2003.

[61] D. Niculescu and B. Nath. "VOR Base Stations for Indoor 802.11 Positioning," in Proc. of *10th Annual International Conference on Mobile Computing and Networking*, pp. 58-69, Philadelphia, PA, USA, 2004.

[62] P.J. Nordlund, F. Gunnarsson, and F. Gustafsson, "Particle Filters for Positioning in Wireless Networks," in Proc. of *XI. European Signal Processing Conference (EUSIPCO)*, 2001.

[63] *The Network Simulator - ns-2*, Internet Resource. http://www.isi.edu/nsnam/ns/.

[64] P. N. Pathirana, N. Bulusu, A. V. Savkin, and S. Jha. "Node Localization Using Mobile Robots in Delay-Tolerant Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 4, pp. 285-296, 2004.

[65] N. Patwari, A. Hero, J. Ash, R. Moses, S. Kyperountas, and N. Correal. "Locating the Nodes: Cooperative Geolocation of Wireless Sensors," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54-69, July 2005.

[66] N. Patwari, A. O. Hero III, M. Perkins, N. S. Correal, and Robert J. O'Dea. "Relative Location Estimation in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 51(8), pp. 2137-2148, August 2003.

[67] N. Patwari and A. O. Hero,"Indirect Radio Interferometric Localization via Pairwise Distances," in Proc. of *3rd IEEE Workshop on Embedded Networked Sensors (EmNets 2006)*, pp. 26-30, Boston, MA, May 30-31, 2006.

[68] R. Peng and M. L. Sichitiu. "Robust, Probabilistic, Constraint-Based Localization for Wireless Sensor Networks," in Proc. of *2nd Annual IEEE Communications Society*

Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005), pp. 541-550, Santa Clara, CA, Sep. 2005.

[69] N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller. "The Cricket Compass for Context-Aware Mobile Applications," in Proc. of *6th ACM MOBICOM*, pp. 1-14, Italy, 2001.

[70] C. Savarese, J. Rabay, and K. Langendoen. "Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks," in Proc. of *USENIX Technical Annual Conference*, pp. 317-327, Monterey, CA, June 2002.

[71] A. Savvides, H. Park, and M. B. Srivastava. "The n-Hop Multilateration Primitive for Node Localization Problems," *Mobile Networks and Applications*, vol. 8/4, pp. 443-451, Aug. 2003.

[72] J. Saxe. "Embeddability of Weighted Graphs in k-space is Strongly NP-hard," in Proc. of *17th Allerton Conference in Communications, Control and Computing*, pp. 480-489, 1979.

[73] M. L. Sichitiu and V. Ramadurai. "Localization of Wireless Sensor Networks with a Mobile Beacon," in Proc. of *1st IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*, pp. 174-183, Fort Lauderdale, FL, October 2004.

[74] F. Sivrikaya and B. Yener. "Time Synchronization in Sensor Networks: A survey," *IEEE Network*, vol. 18(4), pp. 45-50, Jul-Aug. 2004.

[75] H.W. Sorenson, "Least-Square Estimation: from Gauss to Kalman," *IEEE Spectrum*, vol. 7, pp. 63-68, July 1970.

[76] R. Stoleru and J. A. Stankovic. "Probability Grid: A Location Estimation Scheme for Wireless Sensor Networks," in Proc. of *1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pp. 430-438, Santa Clara, CA, Oct. 2004.

[77] I. Stojmenovic, A. P. Ruhil, and D. K. Lobiyal. “Voronoi Diagram and Convex Hull Based Geocasting and Routing in Wireless Networks,” *Wireless Communications and Mobile Computing*, vol. 6(2), pp. 247-258, March 2006.

[78] K.-F. Ssu, C.-H. Ou, and H.C. Jiau. “Localization with Mobile Anchor Points in Wireless Sensor Networks,” *IEEE Vehicular Technology Conference (VTC 2005)*, vol. 54(3), pp. 1187-1197, May 2005

[79] W. Su and M. Gerla, “IPv6 Flow Handoff in Ad Hoc Wireless Networks Using Mobility Prediction,” in Proc. of *IEEE GLOBECOM’99*, pp. 271-275, Rio De Janeiro, Brazil, 1999.

[80] S. Thrun. “Particle Filters in Robotics,” in Proc. of *18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pp. 511, San Francisco, CA, August 2002.

[81] S. Vural and E. Ekici. “Wave Addressing for Dense Sensor Networks,” in Proc. of *IEEE Wokshop on Sensor and Actor Network Protocols and Applications (SANPA)*, pp. 56-66, 2004.

[82] M. Waelchli, M. Scheidegger, and T. Braun. “Intensity-based Object Localization and Tracking with Wireless Sensors,” in Proc. of *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN’06)*, Uppsala, June 2006.

[83] Y. Xu, J. Heidemann, and D. Estrin. “Geography-Informed Energy Conservation for Adhoc Routing,” in Proc. of *ACM/IEEE MOBICOM’01*, pp. 70-84, 2001.

[84] G. V. Záruba, M. Huber, and F. A. Karmangar. “Monte Carlo Sampling Based In-Home Location Tracking with Minimal RF Infrastructure Requirements,” in Proc. of *IEEE GlOBECOM’04*, vol. 6, pp. 3624-3629, Dallas, TX, December 2004.

BIOGRAPHICAL STATEMENT

Rui Huang (黄锐) has obtained a B.S. degree in Computer Science from Texas Christian University, and a M.S. degree in Compute Science from the University of Texas at Arlington. He has been a member of the CReWMaN lab at the University of Texas at Arlington. His research interest includes sensor networks, mobile ad hoc networks, routing and localization.