Distributed Optimization For Location Refinement in Ad-hoc Sensor Networks

Sarfraz Nawaz   Chun Tung Chou   Sanjay Jha

University of New South Wales, Australia
{mnawaz,ctchou,sanjay}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-0820
September 2008

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

A number of range based sensor network localization systems form a rough layout of the network and then starting from this rough layout gradually refine the location coordinates of sensor nodes. We model this coordinate refinement as an unconstrained non-linear optimization problem and show that current heuristic based approaches that require empirical tunning of parameters cannot guarantee convergence in ad-hoc network deployments. We then present a completely distributed algorithm for location refinement and show that this problem can be solved by iteratively performing aggregate sum computations of certain locally computed values over the entire sensor network. Our proposed algorithm does not require any empirical tunning and thus can work with ad-hoc network deployments. We show through simulations and real experimentation that our algorithm exhibits faster convergence as compared to current empirically tunned approaches with similar overhead.

1 Introduction

In recent years, wireless sensor networks have emerged as an important class of complex distributed systems providing new opportunities and challenges. Wireless sensor networks allow us to instrument our world in novel ways providing detailed insight that had not been possible before. Since these networks provide an interface to the physical world, it is necessary for each sensor node to learn its location in the physical space. The availability of location information at individual nodes allows the network to provide higher layer services like event reporting, geographic routing, in-network processing etc.

A number of sensor network localization systems assume that each sensor node is capable of measuring distance to some of its neighboring nodes and use these measured inter-node distances to determine the sensor node coordinates up to a global translation, rotation and reflection. The specific problem addressed by these algorithms can be described as, *Given a set of vertices and a set of weighted edges, determine the euclidean coordinates of vertices up to global translation, rotation and reflection.* This problem has received considerable attention in other fields like graph drawing, distance geometry and dimensionality reduction. Sensor network localization systems addressing this problem heavily borrow ideas from these fields but the most important aspect that sets these localization algorithms apart is their decentralized or distributed nature. Generally these algorithms function in two phases. In the first phase, a rough initial layout of the network is created and in the second phase the node location coordinates are iteratively refined to closely conform to the measured inter-node distances. Although a number of very elegant algorithms have been proposed for the initial layout formation in the first phase, distributed coordinate refinement of the second phase has not received considerable attention. This paper deals with this distributed coordinate refinement problem. Following are the primary contributions of the work presented in this paper.

- We model coordinate refinement as an unconstrained non-linear optimization problem and show that current approaches require empirical tunning of certain parameters for convergence. Without the empirical adjustments, these approaches cannot guarantee convergence.
- We present a completely distributed algorithm for coordinate refinement that does not require empirical adjustments.
- We show that our algorithm transforms the optimization problem into aggregate sum computation which is a well studied problem in peer to peer networks and recently in wireless sensor networks.
- We show through simulation and real experimentation that our algorithm exhibits faster convergence as compared to current approaches even if these have been empirically tunned.

The rest of this paper is organized as follows: In Section (2), we describe some of the background work to put our work in proper context. In Section (3), we model the coordinate refinement as an unconstrained non-linear optimization problem. In Section (4), we present the analysis of current approaches and point out the shortcomings of these approaches. In Section (5), we describe the optimization algorithm that we propose to use for distributed location refinement. In Section (6), we outline our distributed location refinement approach. In Section (7) and (8), we present the simulation and experimental results that validate our analysis. Finally Section (9) summarizes the work presented in this paper.

2 Related Work

The work presented in this paper is targeted toward a class of reference free localization algorithms that determine a rough initial layout of the network from measured inter-node distances and then refine these initial position estimates to achieve more accurate node coordinates. Priyantha et al. [14] use logical network distances among the nodes to form an approximate coordinate system and then use a distributed mesh relaxation (MR) approach to refine these coordinates. Gotsman and Koren [10] use distributed spectral graph drawing (SGD) to form a rough layout of the network. They propose to use stress majorization [11], an optimization technique from multidimensional scaling (MDS), to improve this initial layout. Broxton et al. [6] also use spectral graph drawing followed by mesh relaxation for their Pushpin Computing sensor network. Biswas et al. [3] propose to use semi-definite programming (SDP) for a rough estimate of node positions followed by a distributed mechanism for coordinate refinement. Shang et al. [18] use multidimensional scaling (MDS), a data analysis technique for high dimensional data, to form an estimate of network layout and then propose to refine this layout using a centralized approach. Rao et al. [15] also use logical network distances to determine the node coordinates and refine these coordinates using a distributed approach in which each node adjusts its coordinates by determining the centroid of all of its neighbor locations. Roa et al. [15] have shown that the coordinates obtained in this manner are not the true node coordinates but can be used with geographic routing algorithms.

The coordinate refinement phase used by Priyantha et al. [14] and Broxton et al. [6] referred as mesh relaxation (MR) models the network as a physical system of point masses connected together with springs. These springs exhibit tension and try to compress or expand to their rest lengths pulling the masses to correct locations. However, this physical model requires empirical adjustments for certain parameters without which convergence cannot be guaranteed. These empirical adjustments make this approach unsuitable for ad-hoc deployments of sensor networks. The stress majorization technique used by Gotsman and Koren [10] for coordinate refinement involves solving n linear equations for each of the coordinate dimensions in each iteration. Although, it is possible to accomplish this using distributed computation, this makes the stress majorization approach very expensive and slow. They use a heuristic based approach to speed up convergence. Biswas et al. [3] use a distributed gradient descent optimization approach which also requires empirical tunning and suffers from slow convergence. Shang et al. [18] also model their coordinate refinement phase as unconstrained non-linear optimization. However, a centralized algorithm is used for solving this optimization problem and no distributed approach is presented. Gotsman and Koren [10] have shown that the process of adjusting node coordinates according to the centroid of neighboring nodes used by Rao et al. [15] is in fact a distributed implementation of the spectral graph drawing and thus must be followed by other refinement approaches like majorization to estimate the true node coordinates.

Our work aims to address all of the above mentioned issues with the location refinement approaches proposed so far. We model the location refinement process as an unconstrained non-linear optimization problem and present a completely distributed algorithm for its solution. Our proposed approach does not require empirical adjustments, exhibits faster convergence as compared to previous approaches and is suitable for resource constrained sensor networks.

3 Location Refinement Problem

Let us represent a sensor network with a graph $G(V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices representing sensor nodes and E is a set of edges representing the measured distances among some node pairs i.e. for each measured distance l_{ij}, there is weighted edge $\langle i, j\rangle \in E$. Let us suppose that a primary localization algorithm like SGD, MDS or SDP etc. is used to generate an initial rough layout of G in d dimensions with $d = 2$ or $d = 3$. Given the initial estimated coordinates of each node i as $p_i = (x_i, y_i, z_i)$ and the measured distances l_{ij} among some node pairs, the coordinates can be refined by solving the unconstrained non-linear optimization problem given by Eq. (3.1).

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{\langle i,j\rangle \in E} (d_{ij} - l_{ij})^2, \mathbf{x} \in \Re^{dn} \tag{3.1}$$

where $f : \Re^{dn} \to \Re$, $d_{ij} = \|p_i - p_j\|$ and $\mathbf{x}$ is a vector that contains the coordinates of sensor nodes. Our aim is to find a vector $\mathbf{x}^*$ that minimizes the above function f in a distributed manner.

According to the classical steepest descent method, a local minimum $\mathbf{x}^*$ of Eq. (3.1) can be found by using the iteration,

$$\mathbf{x}(k+1) = \mathbf{x}(k) - \alpha(k) \nabla f(\mathbf{x}(k)) \tag{3.2}$$

where $\alpha(k)$ is called step size and $\alpha(k) > 0$. It is determined from a one dimensional minimization problem as

$$\min_{\alpha(k)} f(\mathbf{x}(k) - \alpha(k) \nabla f(\mathbf{x}(k))) \tag{3.3}$$

It is generally not possible to perform this one dimensional minimization in a distributed manner. Therefore, a constant step size γ is used in Eq. (3.2) if distributed calculations are required. This leads to the following algorithm which we will refer as fixed gradient descent.

$$\mathbf{x}(k+1) = \mathbf{x}(k) - \gamma \nabla f(\mathbf{x}(k)) \tag{3.4}$$

If this fixed gradient descent method is used, each node i can refine its x-coordinate as follows

$$x_i(k+1) = x_i(k) + 2\gamma \sum_{j \in S_i} (x_j(k) - x_i(k)) \left(1 - \frac{l_{ij}}{d_{ij}(k)}\right) \tag{3.5}$$

where S_i is a set of neighbors of node i. The y and z coordinates are updated in a similar manner. From Eq. (3.5), we can see that when using this refinement approach each sensor node i has to communicate only with its adjacent neighbors $j \in S_i$. However, the major issue with this approach is the choice of step size γ. What value of γ should we use in Eq. (3.5) to ensure that this refinement approach converges to a solution $\mathbf{x}^*$ of Eq. (3.1)? We will show in the next section that it is not possible to estimate an optimal value of step size γ that can guarantee convergence if the network topology is not already known.

Biswas et al. [3] use this method for location refinement and present their results for an empirically chosen value of γ. Gotsman and Koren [10] also use a variation of this method. In fact the physical system of masses and springs used by Priyantha et al. [14] and Broxton et al. [6] is also the same fixed gradient descent approach. Both of these works report difficulties with empirical tunning required for this method.

4 Issues With Fixed Step Size

Let us assume that $f : \Re^n \to \Re$ is a continuously differentiable objective function such that

$$f(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \Re^n \tag{4.1}$$

Let us also assume that the gradient ∇f of this function is Lipschitz continuous i.e. there exits a constant $L_f > 0$ such that

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L_f \|\mathbf{x} - \mathbf{y}\|_2, \forall \mathbf{x}, \mathbf{y} \in \Re^n \tag{4.2}$$

If the gradient ∇f satisfies the *Lipschitz Continuity*, then according to the following Lemma found in many optimization text books (for example see Bertsekas and Tsitsiklis [2])

$$f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + \mathbf{y}^T \nabla f(\mathbf{x}) + \frac{L_f}{2} \|\mathbf{y}\|_2^2, \forall \mathbf{x}, \mathbf{y} \in \Re^n \tag{4.3}$$

For a single iteration of the fixed descent algorithm with step size $\gamma > 0$, we can re-write this as

$$f(\mathbf{x}(k+1)) \leq f(\mathbf{x}(k)) - \beta \|\nabla f(\mathbf{x}(k))\|_2^2 \tag{4.4}$$

where

$$\beta = \gamma \left(1 - \frac{L_f \gamma}{2}\right) \tag{4.5}$$

For all $k \geq 0$, we have an inequality similar to (4.4). Adding all of these inequalities and combining with (4.1), we get

$$0 \leq f(\mathbf{x}(k+1)) \leq f(\mathbf{x}(0)) - \beta \sum_{\tau=0}^{k} \|\nabla f(\mathbf{x}(\tau))\|_2^2 \tag{4.6}$$

For the fixed descent algorithm to converge, the inequality (4.6) must be satisfied which is only possible with $\beta > 0$. Using this constraint on β and that $\gamma > 0$, we obtain

$$0 < \gamma < \frac{2}{L_f} \tag{4.7}$$

This shows that the fixed gradient descent algorithm with step size γ can only converge if γ satisfies the inequality (4.7). The value of the Lipschitz constant L_f

depends on the objective function f which in turn depends on the underlying network topology. Thus it is not possible to estimate an optimal value of γ that can guarantee convergence in an ad-hoc network deployment. A simplistic solution to this problem is to use a very small value of γ but this slows down convergence significantly. A slow converging algorithm increases the number of transmitted packets and consequently the energy consumption of each node in the network.

Gotsman and Koren [10] propose to use a different step size per sensor to overcome this issue. They use $\gamma_i = \frac{1}{|S_i|}$ in Eq. (3.5) where $|S_i|$ is the number of neighbors of node i. In optimization theory, such an algorithm is known as scaled gradient algorithm. By following a similar analysis as above, we have shown (see Appendix A for detailed proof) that this scaled gradient algorithm converges only if the condition given in Eq. (4.8) is satisfied where γ_{max} is the largest step size over the entire network.

$$0 < \gamma_{max} < \frac{2}{L_f} \tag{4.8}$$

$$\gamma_{max} = \max\{\gamma_i\}, \; i = 1, \ldots, n \tag{4.9}$$

Thus this heuristic of Gotsman and Koren [10] also suffers from the same drawback. In this section, we saw that with the current approaches of coordinate refinement, it is difficult to estimate the value of step size. In the next section, we present an optimization algorithm by Barzilai and Borwein [1] that does not have this shortcoming.

5 Barzilai Borwein Method

Barzilai and Borwein [1] proposed an optimization method (the BB method) for solving large scale unconstrained minimization problems. They proposed to use a different strategy for choosing step size as compared to classical steepest descent and showed that this new choice of step size required less computational work and greatly improved convergence. This method is summarized below.

If f is an objective cost function such that $f : \Re^n \rightarrow \Re$ then the solution $\mathbf{x}^*$ of the following minimization problem

$$\min f(\mathbf{x}), \mathbf{x} \in \Re^n \tag{5.1}$$

can be found by using the following iteration

$$\mathbf{x}(k+1) = \mathbf{x}(k) - \alpha_{BB}(k)\,\mathbf{g}(k) \tag{5.2}$$

where $\mathbf{g}(k) = \nabla f(\mathbf{x}(k))$ and step size $\alpha_{BB}(k)$ is given as

$$\alpha_{BB}(k) = \frac{\Delta\mathbf{x}^T \Delta\mathbf{g}}{\Delta\mathbf{g}^T \Delta\mathbf{g}} \tag{5.3}$$

or

$$\alpha_{BB}(k) = \frac{\Delta\mathbf{x}^T \Delta\mathbf{x}}{\Delta\mathbf{x}^T \Delta\mathbf{g}} \tag{5.4}$$

with

$$\Delta\mathbf{x} = \mathbf{x}(k) - \mathbf{x}(k-1)$$

$$\Delta\mathbf{g} = \mathbf{g}(k) - \mathbf{g}(k-1)$$

Each iteration of the BB method requires a gradient $\nabla f(\mathbf{x}_k)$ evaluation and only $O(n)$ floating point operations for determining the step size. It is also interesting to note that the step size estimation does not require any function $f(\mathbf{x})$ evaluations or one dimensional line searches. We will later see that this property of the BB method lends it to distributed calculations.

Barzilai and Borwein [1] showed convergence of this method for two dimensional quadratic functions. Raydan [16] extended this to general quadratic functions of any number of variables and Dai and Liao [8] established faster convergence of this method as compared to steepest descent for quadratic functions. Raydan [17] proposed a globalization strategy for BB method for use with non-quadratic problems. This work also showed that it is possible to use the original unmodified BB method for non-quadratic problems if the deviation of $f(\mathbf{x})$ from a quadratic function is small which is generally the case if the method is started close to the local minimum. In our implementation we use the unmodified BB method because in this particular application a good starting point close to the local minimum is provided by the primary localization mechanism (e.g. by Spectral graph drawing, MDS, SDP etc) in the form of the rough layout of the network. Although, the BB method has been used to solve large scale centralized minimization problems, to the best of our knowledge, this is the first application of this method to a distributed problem setting.

6 Distributed Location Refinement

In this section, we outline our distributed location refinement algorithm. We assume that the first phase of a distributed localization mechanism (e.g. SGD, MDS, SDP etc) has been executed in a network of n static sensor nodes forming a rough layout of the network and providing each sensor node i with its rough estimated coordinates $p_i = (x_i, y_i, z_i)$. These estimated coordinates can be improved by making them conform to the set E of measured inter-node distances l_{ij}. This can be achieved by solving the unconstrained non-linear optimization problem of $3n$ variables given in Eq. (3.1) as

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{\langle i,j\rangle \in E} (d_{ij} - l_{ij})^2, \mathbf{x} \in \Re^{3n} \tag{6.1}$$

The objective function f can be minimized in a distributed manner by using the gradient method of Barzilai and Borwein [1]. When using this method each node i can refine its x-coordinate by using the following iteration

$$x_i(k+1) = x_i(k) + 2\alpha_{BB}(k) \sum_{j \in S_i} (x_j(k) - x_i(k)) \left(1 - \frac{l_{ij}}{d_{ij}(k)}\right) \tag{6.2}$$

where S_i is a set of neighbors of node i. Here we see that this iteration requires only adjacent node communication between node i and its neighbors $j \in S_i$ if the current step size $\alpha_{BB}(k)$ is known. The y and z coordinates are updated in a similar manner. However, for each iteration k, the step size $\alpha_{BB}(k)$ must be determined. For the function f defined in Eq. (6.1), this step size is given as

$$\alpha_{BB}(k) = \frac{\sum_{i=1}^{n}\left(\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2\right)}{\sum_{i=1}^{n}\left(\Delta x_i \Delta g_i^x + \Delta y_i \Delta g_i^y + \Delta z_i \Delta g_i^z\right)} \tag{6.3}$$

where

$$\begin{aligned}
\Delta x_i &= x_i(k) - x_i(k-1) && (6.4)\\
\Delta g_i^x &= g_i^x(k) - g_i^x(k-1) && (6.5)\\
g_i^x(k) &= 2\sum_{j\in S_i}\left(x_i(k) - x_j(k)\right)\left(1 - \frac{l_{ij}}{d_{ij}(k)}\right) && (6.6)\\
d_{ij}(k) &= \|p_i(k) - p_j(k)\| && (6.7)
\end{aligned}$$

The expressions for y and z terms in Eq. (6.3) are similar to Eq. (6.4 - 6.6) for x and are omitted to save space over here. Let

$$\begin{aligned}
a_i &= \left(\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2\right) && (6.8)\\
b_i &= \left(\Delta x_i \Delta g_i^x + \Delta y_i \Delta g_i^y + \Delta z_i \Delta g_i^z\right) && (6.9)
\end{aligned}$$

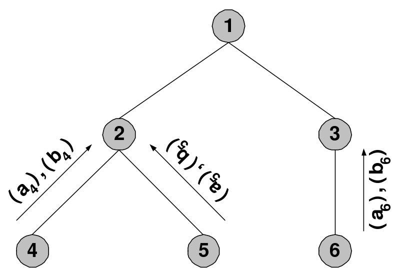
With Eq. (6.8) and Eq. (6.9) we can re-write the step size α_{BB} from Eq. (6.3) as

$$\alpha_{BB}(k) = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \tag{6.10}$$

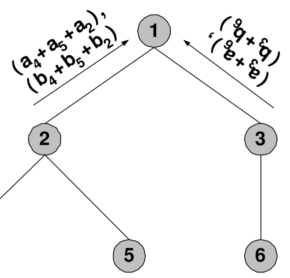
From Eq. (6.4 - 6.9), we can see that each node i can calculate its own a_i and b_i values after exchanging current estimated coordinates $p_i = (x_i, y_i, z_i)$ with adjacent neighbors $j \in S_i$. Once each node i has calculated its own a_i and b_i values, the problem of finding the step size α_{BB} reduces to the aggregate sum calculation problem which can be formally described as,

Given a network of n nodes in which each node i holds a value v_i, find the aggregate sum $s=\sum_{i=1}^{n} v_i$ of these values in a decentralized manner.

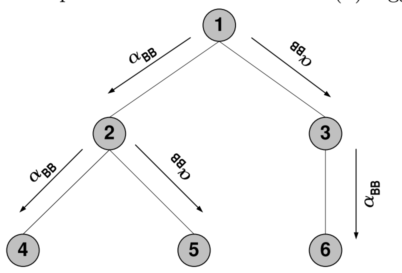
This is a well studied problem in peer-to-peer networks and recently it has also received some attention in wireless sensor networks. The algorithms available for this problem can be divided into two categories, tree based approaches and gossip based algorithms. Tree based approaches like those proposed by Madden et al. [13], Zhao et al. [19] and Boulis et al. [4] are deterministic algorithms that compute the exact value of the aggregate. These algorithms provide an effective and energy efficient aggregation technique. However, these tree based algorithms are generally susceptible to node failures and topology changes. But these issues can be addressed by using tree maintenance approaches like those by Madden et al. [13]. On the other hand gossip based algorithms like those by Boyd et al. [5], Chen et al. [7] and Kempe et al. [12] are distributed localized algorithms that do not require any routing infrastructure like a tree and are thus extremely resilient to link and node failures. These algorithms can be used to compute aggregate functions like average, maximum, minimum etc. of node values within a given error range. Now we describe how both of these approaches (tree based and gossip based) can be used for computing the BB step size.



(a) Aggregate computation



(b) Aggregate computation



(c) Step size

Figure 6.1: Computing the step size α_{BB} using a tree based aggregation approach

6.1 Tree Based Step Size Computation

A tree based aggregation algorithm works by creating a spanning tree rooted at a *root* node. This tree acts as a routing tree for partial aggregate data. Any one of the network nodes can be used as a root node. This root node can be chosen either randomly or through a distributed leader election algorithm like the one proposed by Dulman et al. [9]. Once the root node has been elected, it starts the tree creation process by broadcasting a message containing its own ID and its *level* set to zero. Any node that has not set its level and hears this message, assigns its own level to be the level in the message plus one. It also records the ID of the sender node as its *parent* node. This node then broadcasts the message with its own node ID and level. This process continues until all of the nodes in the network have been assigned a parent node and a level value. At this point each node registers with its direct parent node. This allows each node to learn how many child nodes it has. All of the nodes with zero child nodes can recognize themselves as leaf nodes of the resulting tree. This tree setup phase is required only in the beginning of our algorithm. Once the spanning tree has been setup, it can be used for all the k iterations of our algorithm.

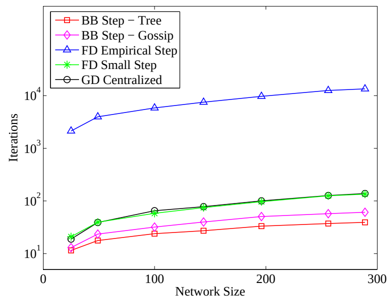
After the spanning tree has been set up, the network starts the step size calculation process. This process starts at the leaf nodes of the spanning tree. These leaf nodes insert their a and b values in a single packet and forward it to their parent nodes. Each parent node waits to receive the packets from all of its child nodes. Each parent node then sums up all of the a and b values received from its child nodes, adds its own a and b values to the respective sums and then forwards these to its parent node. This partial summation of a and b values continues at each level of the tree and eventually the desired aggregate sums reach the root node. The root node then computes the current step size α_{BB} by dividing aggregate sum of a values by the aggregate sum of b values. This step size α_{BB} is then flooded down the tree to all of the network nodes. This entire process is illustrated with a simple example in fig. (6.1) where a small network of six nodes has formed a spanning tree rooted at node 1. The leaf nodes 4, 5 and 6 forward their a and b values to their parent nodes. At node 2 the sums $(a_4 + a_5 + a_2)$ and $(b_4 + b_5 + b_2)$ are computed and forwarded to node 1. Similarly node 3 computes partial sums $(a_6 + a_3)$ and $(b_6 + b_3)$ that are forwarded to node 1. Node 1 adds up the respective partial sums, adds a_1 and b_1 to these partial sums and computes the step size α_{BB} with a simple division. This step size value is then flooded down to all of the nodes. On receiving the step size, each node updates its coordinates by using Eq. (6.2). This completes one iteration of the algorithm. In the next iteration a and b values are computed again by all the nodes and the already setup up tree is used to compute the next step size. This process continues until the optimization procedure converges to the local minimum of objective function f. In Section (6.3), we outline the process for detecting the convergence to the local minimum and terminating the distributed optimization process.

6.2 Gossip Based Step Size Computation

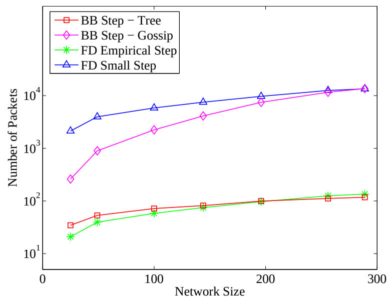
In this section, we describe how a gossip based algorithm can be used for computing the BB step size in each iteration of our location refinement approach. We propose to use Distributed Random Grouping (DRG) by Chen et al. [7] because it takes advantage of the broadcast nature of wireless transmissions in wireless sensor networks in contrast to other algorithms that were primarily intended for peer to peer networks. DRG computes the average $\frac{1}{n}\sum_{i=1}^{n} v_i$ of all of the node values v_i as opposed to the sum aggregate required by our algorithm. However, the average A of a_i values and B of b_i values can be used to compute the BB step size as

$$\begin{aligned}\frac{A}{B} &= \frac{\frac{1}{n}\sum_{i=1}^{n} a_i}{\frac{1}{n}\sum_{i=1}^{n} b_i} \\ &= \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \\ &= \alpha_{BB} \qquad (6.11)\end{aligned}$$

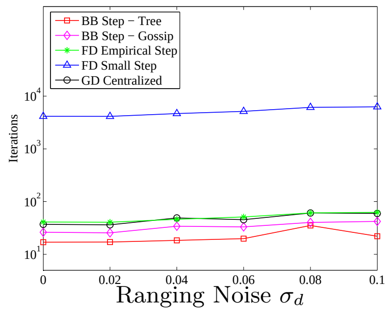
Thus, in each iteration of our refinement algorithm, we can use the average aggregates computed by DRG algorithm to compute the step size α_{BB}. Here we briefly describe the DRG algorithm in the context of our application. Each node running this algorithm can be in one of the three states; idle, group leader or group member. A node in idle state becomes a group leader with probablity p_g or remains idle with $1 - p_g$. A group leader broadcasts its ID and waits for its neighboring idle nodes to join the group. An idle node that receives this announcement sends its a and b values to the group leader and changes its state to a group member. A group member does not respond to any other group join requests. The group leader computes the group average A_g of received a values and B_g of received b values and broadcasts these group averages. Each group member that receives this broadcast overwrites its a and b values with



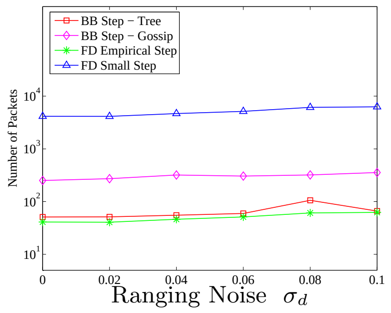
(a) Algorithm Iterations



(b) Transmitted Packets



(c) Algorithm Iterations



(d) Transmitted Packets

Figure 6.2: Comparison of our location refinement algorithm with current approaches using Spectral Graph Drawing (SGD) as the starting point for all of the algorithms.

A_g and B_g respectively. If this process is continuously repeated, the a and b values of each node converge to global average A and B respectively. Chen et al. [7] has shown that the upper bound for convergence of this process depends on grouping probability p_g, underlying network topology, the variance of initial values and the desired accuracy of the aggregate average. They have also shown that fastest convergence of DRG is achieved when $p_g = \frac{1}{\chi}$ where χ is the average number of two hop neighbors. However, a distributed stop mechanism for DRG is not discussed. In our implementation of DRG, we have used a simple stop mechanism that allows each node to determine if the aggregate average has converged or not. Each node monitors its aggregate values and if these aggregates do not change significantly in a number of successive rounds, it stops the DRG algorithm and computes the step size using Eq. (6.11). After the step size has been computed, each node can update its coordinates using Eq. (6.2). This completes one iteration of the location refinement algorithm. In the next iteration, a and b values are computed again and the DRG algorithm is used to compute the new step size. This process continues until the location refinement algorithm converges to the local minimum of objective function f. In the next section, we outline a procedure for detecting this convergence and terminating the location refinement process.

6.3 Termination

At the local minimum of the objective function f, the gradient $g = \nabla f(\mathbf{x})$ of the objective function becomes zero. Therefore, convergence to the local minimum can be detected by checking if $\|\nabla f(\mathbf{x})\| = 0$. Here we describe a distributed process for performing this check. For the objective function f defined in Eq. (6.1), we have

$$\|\nabla f(\mathbf{x})\|^2 = \sum_{i=1}^{n} \left\{ (g_i^x)^2 + (g_i^y)^2 + (g_i^z)^2 \right\} \tag{6.12}$$

Let us define c_i for each node i as

$$c_i = (g_i^x)^2 + (g_i^y)^2 + (g_i^z)^2 \tag{6.13}$$

then we have

$$\|\nabla f(\mathbf{x})\|^2 = \sum_{i=1}^{n} c_i \tag{6.14}$$

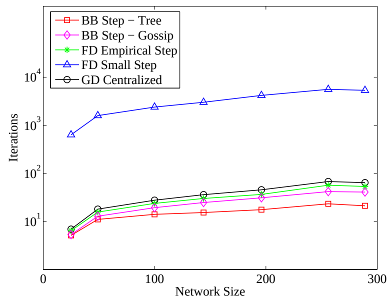
Eq. (6.14) shows that the squared norm of the gradient of objective function f can be computed by finding the aggregate sum of c_i values over the entire network. Thus the problem of testing the convergence of the algorithm also reduces to the aggregate sum problem. This aggregate sum $\sum_{i=1}^{n} c_i$ can be computed in the same manner as $\sum_{i=1}^{n} a_i$ and $\sum_{i=1}^{n} b_i$.

With the tree based approach, in each iteration of the algorithm, each node sums up the c values received from all of its child nodes along with the a and b values and forwards all three partial sums to its parent node in a single packet. Thus a single *wave* of packets moving toward the root node is used to compute all of the three aggregate sums. When these aggregate sums arrive at the root node, the root node can test if the algorithm has arrived at local minimum. If at the root node $\|\nabla f(\mathbf{x})\|^2 = \sum_{i=1}^{n} c_i \approx 0$, all the nodes down the spanning tree are informed to terminate the algorithm. Otherwise a new value of the step size α_{BB} is computed at the root node and flooded down the spanning tree. The algorithm continues until convergence is achieved.

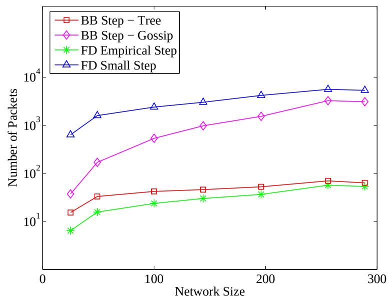
When using the gossip based approach, in each iteration of our algorithm, all of the nodes gossip about a, b and c values with neighboring nodes by transmitting all of the three values in the same packet. Thus when the gossip algorithm converges, each node has the global averages $A = \frac{1}{n}\sum_{i=1}^{n} a_i$, $B = \frac{1}{n}\sum_{i=1}^{n} b_i$ and $C = \frac{1}{n}\sum_{i=1}^{n} c_i$. Now each node can check if the location refinement algorithm has converged by checking if the computed aggregate $C = \frac{1}{n}\sum_{i=1}^{n} c_i \approx 0$. Otherwise a new value of the step size α_{BB} is computed from A and B using Eq. (6.11) and the location refinement algorithm continues until convergence is achieved.

6.4 Computational Cost

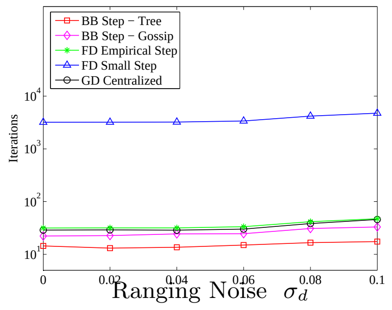
It is generally not possible to analytically predict the exact number of iterations of the distributed location refinement algorithm because the number of iterations required for convergence depend on the spectrum (distribution of eigenvalues) of the Hessian of the objective function f. We can however analyze the cost of



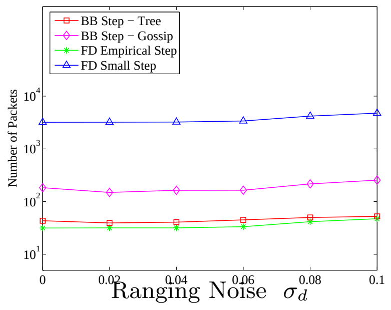
(a) Algorithm Iterations



(b) Transmitted Packets



(c) Algorithm Iterations

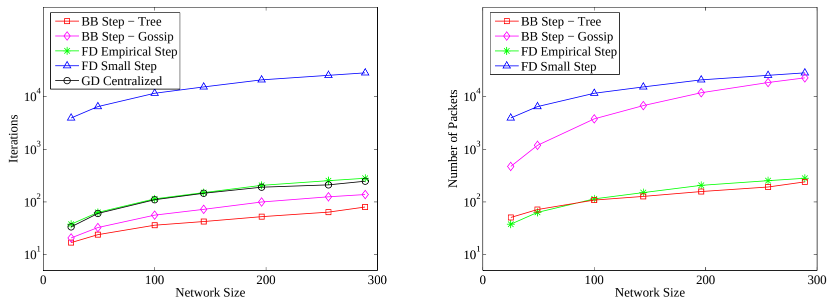


(d) Transmitted Packets

Figure 6.3: Comparison of our location refinement algorithm with current approaches using Multidimensional Scaling (MDS) as the starting point for all of the algorithms.

each iteration of our algorithm. In each iteration of the algorithm, each node i calculates components g_i of the gradient which are then used to compute the b_i and c_i values. The a_i values are also computed from the difference of coordinates from the last two iterations. From Eq. (6.4-6.6), we can see that these computations depend on the number of dimensions d in which the node i has been localized and $|S_i|$ the number of neighbors of node i. Thus the computational complexity for each node running the location refinement algorithm is $O\left(d + |S_i|\right)$ per iteration.

While executing the algorithm, each node has to store the coordinates $p_i\left(k-1\right)$ and gradient components $g_i\left(k-1\right)$ from the previous iteration. Therefore, the space complexity of the algorithm for each node is $O\left(d\right)$. In each iteration, each node i broadcasts its current coordinates $p_i\left(k\right)$ and calculates its a_i, b_i and c_i values from the received coordinates $p_j\left(k\right), j \in S_i$ from its neighboring nodes. The message complexity of computing the step size from these values depends on the type of aggregation approach being used. In the tree based approach, a_i, b_i and c_i values at each node are combined with those received from child nodes and sent to the parent node. On receiving the current step size, node i broadcasts it to its child nodes. Thus, with a tree based approach, message complexity of the algorithm is $O\left(1\right)$ per iteration since each node has to transmit a fixed number of packets in each iteration. In the gossip based approach,



(a) Algorithm Iterations (b) Transmitted Packets

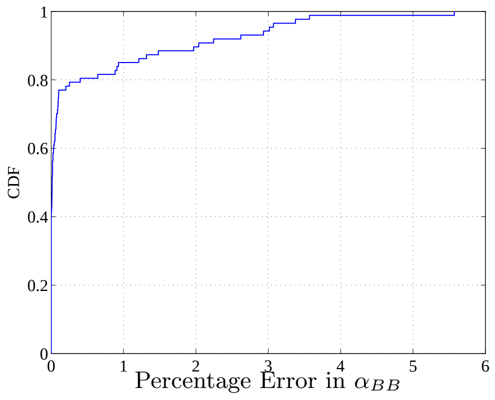
Figure 7.1: Comparison of our location refinement algorithm with current approaches using Fold Free Layout (FFL) as the starting point for all of the algorithms.

each node sends its values to a group leader and receives the computed group averages in successive rounds. With this gossip based approach, the message complexity of each iteration of our refinement algorithm depends on the number of rounds of DRG gossip algorithm for converging to the aggregate function. Chen et al. [7] has shown that it is given by $O\left(\frac{1}{\beta}\log\left(\frac{\phi}{\epsilon}\right)\right)$ where β depends on grouping probability p_g and network topology, ϕ is the grand variance of initial values and ϵ is the error tolerance of the aggregate average.

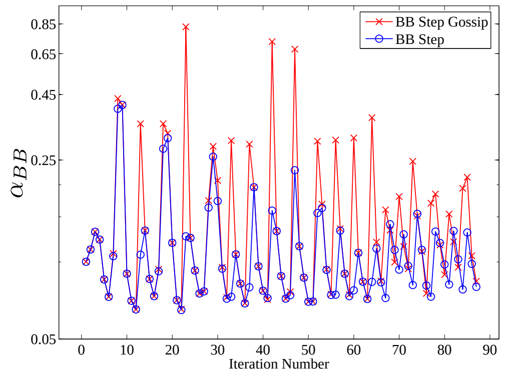
7 Simulation Results

In this section, we present simulation results that compare our algorithm with current location refinement approaches and show that our algorithm exhibits faster convergence as compared to current algorithms while incurring similar overhead. We compare the performance of refinement algorithms under a range of network parameters like network size, node density and the distance measurement error. The network topologies used in these simulations were generated by placing nodes in a square shaped region in a noisy grid manner. The placement error was modeled as Guassian noise. Thus a random value drawn from a normal distribution $N(0, \sigma_p)$ with $\sigma_p = 2$ units was added to each grid point. Each node in the network had a maximum ranging distance of $r = 5$ units. In all of the generated topologies the average number of node neighbors was around six.

We implemented four different variants of the gradient descent approach for comparison against our algorithm. As we reviewed in Section 2, currently available approaches use a fixed gradient descent algorithm where an *empirically* chosen value of fixed step size γ is used in Eq. (3.5) to refine node coordinates. For our set of topologies, we also experimented with a range of γ values and selected the empirical value $\gamma_e = 0.1$ which provided the fastest convergence for the fixed gradient descent algorithm. We will refer to this variant as fixed descent with empirical step size. An alternative to empirical tunning is to use a very small value of γ. Thus for the second variant we used a small step size $\gamma_s = 0.001$ in Eq. (3.5). We will refer to this variant as fixed descent with small step size.



(a) Error CDF of gossip α_{BB}



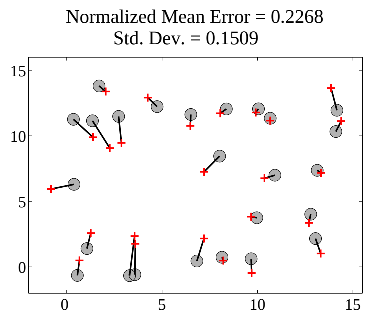
(b) Comparison of true and gossip α_{BB}

Figure 7.2: Comparison of true and gossip computed α_{BB} values

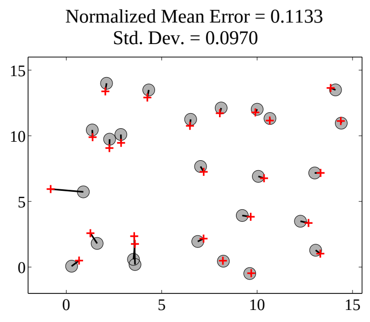
In the third variant, we used a different step size for each sensor node which was derived from the number of node neighbors i.e. $\gamma_i = \frac{1}{|S_i|}$. We will refer to this variant as fixed descent with neighbor step size. We also implemented a centralized gradient descent algorithm that derives an exact step size in each iteration using a centralized line search of Eq. (3.3). This centralized algorithm represents a bound on the fixed descent method since no amount of empirical tunning of the fixed descent method can allow it to perform significantly better than the centralized gradient descent algorithm. In our simulations, we use distributed multidimensional scaling (MDS) by Shang et al. [18], spectral graph drawing (SGD) by Gotsman and Koren [10] and the fold free layout (FFL) by Priyantha et al. [14] to generate a rough layout of the network. These rough network layouts are then used as starting points for all the location refinement algorithms.

The results for SGD starting layouts are presented in Fig. (6.2). Fig. (6.2a) shows the effect of increasing the size of the network while maintaining constant node density on the performance of different algorithms. Each data point in the figure is an average of 10 runs of simulation on 10 different similar sized randomly generated topologies. It shows that the number of iterations required for convergence by each algorithm gradually increase with the network size. The performance of fixed descent method with empirical step is identical to the centralized gradient descent method. It shows that our empirically tunned value of step size $\gamma_e = 0.1$ is indeed a good choice for this set of topologies when using the fixed descent method. Using the small value of step size γ_s for the fixed descent method results in considerably large number of iterations (30 times more than the fixed descent with empirical step) of the algorithm. The fixed descent method with neighbor step size γ_i failed to converge for all of the topologies in these simulations and is therefore not included in these plots. The figure shows that our Barzilai and Borwein based approach requires smaller number of iterations for convergence as compared to various fixed descent algorithms. It also shows that using the gossip based approach to calculate the step size α_{BB} results in slightly larger number of iterations as compared to the case when a tree based algorithm is used to compute the step size α_{BB}. This is due to the fact that when the magnitude of the step size being computed is very small, the gossip based approach results in slightly different step size values for different nodes across the network for a given error tolerance ϵ. Although, this deteriorates the performance of BB method, it still exhibits faster convergence as compared to fixed descent method with empirical step size γ_e. Fig. (7.2a) shows the CDF plot of the percentage error between the true step size α_{BB} and the mean of the step size values over the entire network computed using gossiping for the entire run of our refinement algorithm on one of the network topologies. It shows that almost 80% of step sizes have no errors and the remaining 20% of step sizes have less than 6% error. Fig. (7.2b) shows the plot of true value of step size α_{BB} as blue circles and the mean value of the step size over the entire network computed using DRG gossip as red crosses. This demonstrates that our location refinement approach can not only tolerate small errors introduced in step size α_{BB} when using gossiping but still maintian faster convergence as compared to fixed descent algorithms.

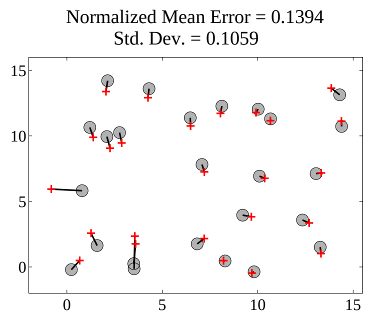
The radio transceiver is the largest energy consumer in a wireless sensor node. Therefore, it is very important to characterize the radio usage of sensor network algorithms to gain an insight into network lifetime. We now compare the number of packets transmitted by a sensor node when refining location coordinates using different refinement algorithms. When using the fixed descent approach, each node transmits only one packet in each iteration of the algorithm for sharing the current estimated coordinates with its neighbors. However, in our algorithm each node has to transmit additional packets in each iteration to compute the step size α_{BB}. Fig. (6.2b) shows the total number of packets transmitted by each node in the network when executing different refinement algorithms. We point out that only distributed algorithms that exhibited convergence have been compared in this figure. This excludes the centralized gradient descent and the fixed descent method with neighbor step size. As expected, the fixed descent method with small value of step size requires the largest number of packet transmissions for convergence. Fig. (6.2b) shows that our algorithm which does not require any empirical adjustments, exhibits similar overhead as the empirically tunned fixed descent method when an aggregation tree is used to compute the step size α_{BB} in our algorithm. On the other hand, relatively larger number of packets are transmitted when DRG gossip is used to calculate the step size. However, the gossip based refinement algorithm still results in smaller number of packet transmissions as compared to fixed descent with very



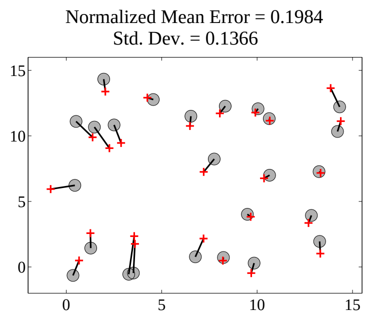
(a) SGD starting layout



(b) Our algorithm



(c) FD with empirical step size



(d) FD with small step size

Figure 7.3: Improved sensor node coordinates after executing 15 iterations of different algorithms.

Table 7.1: Number of iterations required for convergence of different algorithms with increasing node density.

Average Neighbors	**BB Tree**	**BB Gossip**	**Neighbor Step**	**Empirical Step**	**Small Step**
	α_{BB}	α_{BB}	$\gamma_i = \frac{1}{\lvert S_i \rvert}$	$\gamma_e = 0.1$	$\gamma_e = 0.001$
7	50	95	**X**	164	16479
14	29	37	**X**	**X**	3840
23	18	32	38	**X**	1846
33	20	28	31	**X**	1219

small step size. Here, we would like to mention the inherent trade off between the tree based and gossip based step size computation. The tree based algorithm offers an energy efficient and low overhead approach but does not offer good robustness against node and link failures. On the other hand, the gossip based approach is extremely resilient to failures but exhibits slightly higher overhead. The network user must choose the appropriate step size computation approach based on the application requirements and the hostility of the environment in which the sensor network has to be deployed.

We now measure the sensitivity of various refinement algorithms to the distance measurement noise. We restrict ourselves to the convergence speed of different algorithms under varying measurement noise as the effects of ranging noise on localization error are already documented in current literature. We model the measurement noise as Gaussian noise and add a random value drawn from a normal distribution $N_d\left(0, \sigma_d l_{ij}\right)$ to each measured distance l_{ij}. For these simulations we used a small topology of 25 sensor nodes. Fig. (6.2c) and (6.2d) show the performance of different algorithms for a range of σ_d values. Each point on the curves is an average of 100 runs of simulation with randomly generated ranging noise. These results show that each algorithm requires larger number of iterations for convergence as the measurement noise is increased. Increasing the measurement noise deteriorates the starting layout generated by SGD and thus each algorithm has to perform more work to get to the local minimum. In our simulations, all algorithms converge to the same local minimum (except fixed descent with neighbor step which failed to converge) and thus exhibit same localization errors. As shown in the figure, our approach requires least number of iterations for convergence and exhibits similar overhead as the fixed descent with empirical step value with tree based step size computation.

We perform the same simulation experiments using the starting layouts generated by distributed MDS and FFL. The results for these simulations are presented in fig. (6.3) and fig. (7.1) respectively. These results are quite similar to those presented in fig. (6.2) for SGD layouts. The performance of fixed descent with empirical step is almost identical to that of centralized gradient descent and the descent method with small step size shows very slow convergence. The descent method with neighbor step size fails to converge and our algorithm exhibits faster convergence with both the tree based and gossip based step size calculations.

We also compare different algorithms with the increasing density of sensor

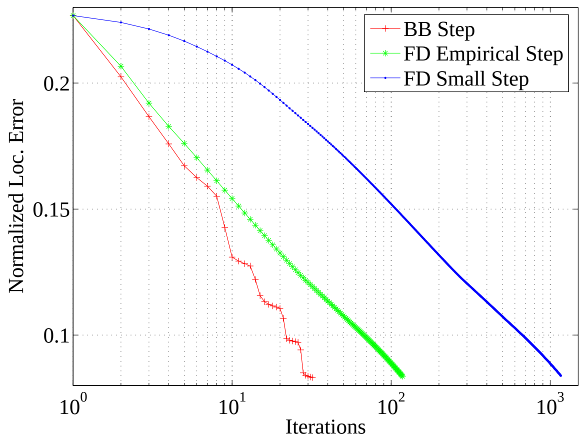


Figure 7.4: Localization error of different refinement algorithms.

nodes. For this simulation we use a 256 node noisy grid topology where the network density is varied by gradually increasing the ranging distance of each node in the network. Table (7.1) shows the number of iterations taken by each algorithm for convergence to local minimum for different node densities. In the table, a **X** indicate that the algorithm failed to converge. The fourth column of the table shows that the fixed descent method with empirically chosen value of step size γ_e fails to converge when the characteristics of the network topology change. Thus an empirically chosen value of the step size for certain type of topologies cannot guarantee convergence with different network topologies. This was also predicted by our analysis in Section (4). Fixed descent method with neighbor step size γ_i only converges for very dense configurations where the average number of node neighbors is more than 20. Our algorithm not only converges under all configurations but also exhibits faster convergence as compared to all of variants of fixed descent method.

Now we use an example topology to illustrate the performance of different algorithms. We use spectral graph drawing (SGD) to create a rough starting layout of the network which is shown in fig. (7.3a). Fig. (7.3) shows a snapshot of the estimated node coordinates after an equal number of iterations (15 iterations) of different algorithms starting from the initial layout have been executed. In the figure, grey circles indicate estimated coordinates and red crosses indicate true node locations. Fig. (7.3b) shows that after a fixed number of iterations, our algorithm results in smaller localization errors as compared to the coordinates generated by fixed descent method with empirical and small step sizes shown in fig. (7.3c) and fig. (7.3d) respectively. This shows that our algorithm is able to perform more improvement in node coordinates per iteration as compared to the fixed descent method. This is also illustrated in fig. (7.4) where the normalized mean localization error of the estimated coordinates of different algorithms is plotted against the iteration number. Fig. (7.4) also presents a very interesting aspect of these location refinement algorithms which is the inherent energy accuracy trade off of these algorithms. It shows that if an application requires very accurate node coordinates, the sensor network would have to execute more iterations of the location refinement algorithm thus

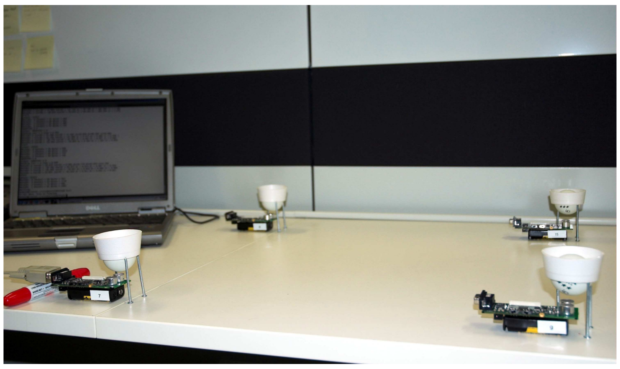


Figure 8.1: Coordinate Refinement Experiment

spending more energy. On the other hand less accurate location coordinates require fewer iterations of the algorithm and thus conserve the energy. Thus these algorithms present an *energy accuracy knob* which can be used to tune the algorithm according to the application requirements.

8 Experiment

We implemented our location refinement algorithm and the fixed descent approach in `TinyOS` and conducted a series of experiments on a real network of MIT Cricket motes to validate our simulation results. In this section, we present the results from these experiments.

For these experiments, we used a small single-hop network of four Cricket motes placed at the four corners of a $55cm \times 55cm$ square on an office table as shown in Fig. (8.1). One of the motes was connected to a laptop computer through the serial interface to collect different network statistics. The maximum ranging distance of each Cricket mote was deliberately limited to $60cm$ to form a sparse network in which each mote could measure distance to just two of its adjacent neigbhor nodes. This setup was only adopted for ease of experimentation with in a small office space and the results presented here are representative of the performance of these algorithms in larger networks.

Each of our experiments consists of three distinct phases; ranging, coordinate estimation and finally coordinate refinement. In the first phase, each Cricket mote periodically transmits radio beacons accompanied by ultrasound signals which are used by other motes to estimate distance to the transmitting mote. Each mote continuously filters the distance measurements using a Kalman filter and stores the most up to date distances to its neighbors in a neighbor table. In the second phase, filtered distance measurements from each mote are collected at the laptop and `MDS-MapP` algorithm of Shang et al. [18] is used to determine the initial coordinates of each Cricket mote. These initial coordinates are then transmitted to respective motes. We must point out that although we used a centralized implementation of `MDS-MapP`, it can be executed in the network in a distributed fashion. We chose to use a centralized implementation to save the limited code and program memory available on Cricket hardware. However, this choice of centralized coordinate estimation does not affect the location refine-

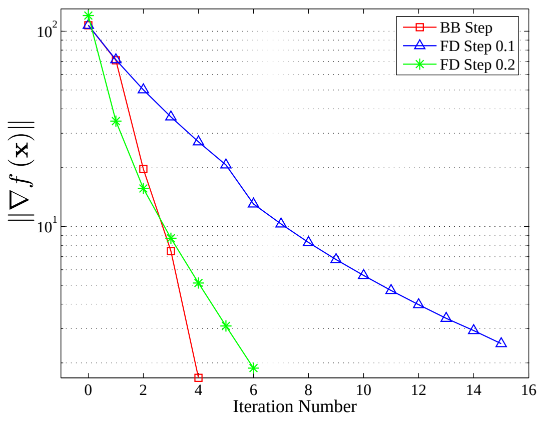


Figure 8.2: Coordinate Refinement Experiment

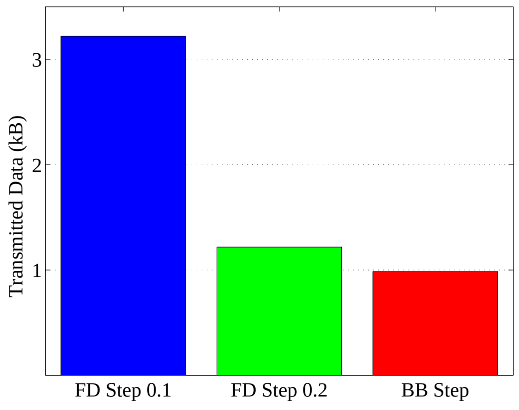


Figure 8.3: Algorithm Overhead

ment algorithm which is executed in the third phase. For coordinate refinement, we used our BB based approach and the fixed descent algorithm with two different step sizes $\gamma = 0.1$ and $\gamma = 0.2$. Since it is not possible to choose an optimal value of the step size, we chose 0.1 as an arbitrary value for the step size. The second step size $\gamma = 0.2$ was chosen after observing the behaviour of fixed descent algorithm with $\gamma = 0.1$. Thus $\gamma = 0.2$ can be regarded as the empirically adjusted step size for the fixed descent approach. We used the same termination criteria of $\|\nabla f\| < 3$ for both of the algorithms.

Fig. (8.2) plots the norm-2 of the gradient $\|\nabla f\|$ versus the iteration number for all of the three experiments. It shows that in this real network, our algorithm exhibits faster convergence as compared to the fixed descent algorithm which has the additional difficulty of choosing an appropriate step size. Our algorithm, on the other hand, does not require the user to choose a step size. Fig. (8.2) shows that our algorithm achieved convergence in just 4 iterations as opposed to 15 and 6 iterations for fixed descent algorithm with step sizes $\gamma = 0.1$ and $\gamma = 0.2$ respectively. Fig. (8.3) shows the average amount of data transmitted by each mote when running different refinement algorithms. It shows that when using our BB based refinement approach, each node in this small network transmits just around 1kB of overhead data as compared to more than 3kB when the fixed descent algorithm is used with an arbitrary value of 0.1 as step size.

9 Conclusion

In this paper, we showed that current heuristic based approaches of coordinate refinement cannot guarantee convergence for ad-hoc network topologies. We modeled coordinate refinement as unconstrained nonlinear optimization problem and then presented a completely distributed and heuristic free algorithm for solving this problem. We showed through simulation and real experimentation that our proposed algorithm exhibits faster convergence as compared to heuristic based empirically tunned algorithms.

A Scaled Gradient Descent Convergence

Let us assume that we wish to solve the following coordinate refinement optimization problem,

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{\langle i,j\rangle \in E} (d_{ij} - l_{ij})^2 \tag{A.1}$$

If we use the heuristic proposed by Gotsman and Koren [10], each node i can refine its coordinates by using the following iteration

$$x_i(k+1) = x_i(k) + 2\gamma_i \sum_{j \in S_i} (x_j(k) - x_i(k)) \left(1 - \frac{l_{ij}}{d_{ij}(k)}\right) \tag{A.2}$$

Each node i uses a different step size $\gamma_i = \frac{1}{|S_i|}$ where $|S_i|$ is the number of neighbors of node i. The y and z coordinates are update in a similar manner. We can rewrite this iterative process for the entire network as

$$\mathbf{x}(k+1) = \mathbf{x}(k) - \mathbf{D}\nabla f(\mathbf{x}(k)) \tag{A.3}$$

where $\mathbf{x}$ is a vector of node coordinates and $\mathbf{D}$ is a diagonal matrix called the scaling matrix. It is given as,

$$\mathbf{D} = \begin{bmatrix} \gamma_1 & 0 & \dots & 0 \\ 0 & \gamma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \gamma_n \end{bmatrix} \tag{A.4}$$

Let us also assume that the gradient ∇f of the function given in (A.1) is Lipschitz continuous i.e. there exits a constant $L_f > 0$ such that

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L_f \|\mathbf{x} - \mathbf{y}\|_2, \forall \mathbf{x}, \mathbf{y} \in \Re^n \tag{A.5}$$

If the gradient ∇f satisfies the *Lipschitz Continuity*, then we have the following Lemma

$$f(\mathbf{x}+\mathbf{y}) \leq f(\mathbf{x}) + \mathbf{y}^T \nabla f(\mathbf{x}) + \frac{L_f}{2} \|\mathbf{y}\|_2^2, \forall \mathbf{x}, \mathbf{y} \in \Re^n \tag{A.6}$$

For a single iteration of the scaled gradient descent algorithm, we can re-write this as

$$\begin{aligned} f(\mathbf{x}(k+1)) \leq\ & f(\mathbf{x}(k)) - \nabla f(\mathbf{x}(k))^T M \nabla f(\mathbf{x}(k)) \\ & + \frac{L_f}{2} \|D \nabla f(\mathbf{x}(k))\|_2^2 \end{aligned} \tag{A.7}$$

If λ_{max} is the largest eigenvalue of $\mathbf{D}$,

$$\nabla f(\mathbf{x}(k))^T M \nabla f(\mathbf{x}(k)) \leq \lambda_{max} \|\nabla f(\mathbf{x}(k))\|_2^2 \tag{A.8}$$

$$\|D \nabla f(\mathbf{x}(k))\|_2 \leq \lambda_{max} \|\nabla f(\mathbf{x}(k))\|_2 \tag{A.9}$$

Using (A.8) and (A.9), we can rewrite (A.7) as

$$f(\mathbf{x}(k+1)) \leq f(\mathbf{x}(k)) - \eta \|\nabla f(\mathbf{x}(k))\|_2^2 \tag{A.10}$$

where

$$\eta = \lambda_{max}\left(1 - \frac{\lambda_{max} L_f}{2}\right) \tag{A.11}$$

For all $k \geq 0$, we have an inequality similar to (A.10). By adding all of these inequalities, we get

$$0 \leq f(\mathbf{x}(k+1)) \leq f(\mathbf{x}(0)) - \eta \sum_{\tau=0}^{k} \|\nabla f(\mathbf{x}(\tau))\|_2^2 \tag{A.12}$$

For the scaled gradient descent algorithm to converge, the inequality (A.12) must be satisfied which is only possible with $\eta > 0$. Combining this with the fact that

$$\lambda_{max} = \gamma_{max} \tag{A.13}$$

$$\gamma_{max} = \max\{\gamma_i\} \quad i = 1, 2, \ldots, n \tag{A.14}$$

we have the following bounds on the step size

$$0 < \gamma_{max} < \frac{2}{L_f} \tag{A.15}$$

This shows that the heuristic approach proposed by Gotsman and Koren [10] can only converge if the largest step size over the entire network γ_{max} satisfies the bounds given in Eq. (A.15) which may not always be the case in an ad-hoc network deployment. Because in this approach,

$$\gamma_i = \frac{1}{|S_i|} \tag{A.16}$$

another way of looking at Eq. (A.15) is

$$\min |S_i| > \frac{L_f}{2} \tag{A.17}$$

which shows that this algorithm by Gotsman and Koren [10] algorithm will only converge if the minimum number of node neighbors in the network is greater than $\frac{L_f}{2}$ which again may not be possible in an ad-hoc network deployment.

Bibliography

[1] Jonathan Barzilai and Jonathan M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, 1989.

[3] Pratik Biswas, Tzu-Chen Lian, Ta-Chung Wang, and Yinyu Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks*, 2(2):188–220, 2006.

[4] Athanassios Boulis, Saurabh Ganeriwal, and Mani B. Srivastava. Aggregation in sensor networks: an energy-accuracy trade-off. *Ad Hoc Networks*, 1(2-3):317–331, 2003.

[5] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: design, analysis and applications. In *INFOCOM'05: Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1653–1664, 13-17 March 2005.

[6] Michael Broxton, Joshua Lifton, and Joseph A. Paradiso. Localization on the pushpin computing sensor network using spectral graph drawing and mesh relaxation. *SIGMOBILE Mobile Computing & Communication Review*, 10(1):1–12, 2006.

[7] Jen-Yeu Chen, Gopal Pandurangan, and Dongyan Xu. Robust computation of aggregates in wireless sensor networks: distributed randomized algorithms and analysis. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 348–355, April 2005.

[8] Yu-Hong Dai and Li-Zhi Liao. R-linear convergence of the barzilai and borwein gradient method. *IMA Journal of Numerical Analysis*, 22(1):1–10, 2002.

[9] S. Dulman, P. Havinga, and J. Hurink. Wave leader election for wireless sensor networks. In *Proceedings of the 3rd International Symposium on Mobile Multimedia Systems & Applications*, pages 43–50, 2002.

[10] Craig Gotsman and Yehuda Koren. Distributed graph layout for sensor networks. In *Graph Drawing*, pages 273–284, 2004.

[11] P. Groenen. *The majorization approach to multidimensional scaling: some problems and extensions.* DSWO Press, 1993.

[12] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 482, Washington, DC, USA, 2003. IEEE Computer Society.

[13] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[14] Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, and Seth Teller. Anchor-free distributed localization in sensor networks. Technical Report 892, MIT, Laboratory for Computer Science, April 2003.

[15] Ananth Rao, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108, New York, NY, USA, 2003. ACM Press.

[16] Marcos Raydan. On the Barzilai and Borwein choice of steplength for the gradient method. *IMA Journal of Numerical Analysis*, 13(3):321–326, 1993.

[17] Marcos Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7(1):26–33, 1997.

[18] Yi Shang, Wheeler Rumi, Ying Zhang, and Markus Fromherz. Localization from connectivity in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(11):961–974, 2004.

[19] Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 139–148, May 2003.