DISTRIBUTED OPTIMIZATION ON A WIRELESS SENSOR NETWORK TESTBED

BY

NEERAJ VENKATESAN

THESIS
Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:
Professor Venugopal V. Veeravalli
The focus of this thesis is to implement various distributed optimization algorithms on a physical wireless sensor network. Distributed optimization refers to optimization of some global function which is not completely known to any single node in a communication network. The global function is some combination of local functions that are available at each node. Therefore the objective is for all nodes to achieve consensus on the global optimum given only local information and communication with neighbors.

Algorithms from the literature that address this problem in different settings are introduced, focusing on an incremental subgradient-based algorithm and a broadcast, gossip-based algorithm. These algorithms are applied to localize a light source. This localization problem is formulated as a distributed optimization problem in which the global optimum is the true location of the source, and the local information is comprised of light intensity measurements at each node. Simulation results and results from physical implementations on the testbed are presented for the two different approaches. A modified version of the broadcast algorithm is also presented, and is shown to be superior to the unaltered algorithm in certain settings via simulation and testbed results.
ACKNOWLEDGMENTS

I would like to thank my adviser, Prof. Venugopal Veeravalli, for all of the guidance he provided for this thesis. I would like to thank my parents and my brother for their love and support throughout all of my endeavors. Finally, I would like to thank the friends I have made during my time here in Illinois for all of the fun times we shared through graduate school.
# TABLE OF CONTENTS

**CHAPTER 1 INTRODUCTION** .................................................. 1  
1.1 Context ................................................................. 1  
1.2 Overview .............................................................. 3  
1.3 Notation ............................................................... 3  

**CHAPTER 2 DISTRIBUTED OPTIMIZATION FORMULATION** ........... 4  
2.1 General Problem ...................................................... 4  
2.2 Localization Problem ................................................ 6  

**CHAPTER 3 INCREMENTAL OPTIMIZATION ALGORITHM** .......... 14  
3.1 Algorithm Description .............................................. 14  
3.2 Simulation Results .................................................. 16  
3.3 Testbed Results ...................................................... 19  

**CHAPTER 4 BROADCAST OPTIMIZATION ALGORITHM** ............ 24  
4.1 Algorithm Description .............................................. 24  
4.2 Simulation Results .................................................. 25  
4.3 Testbed Results ...................................................... 28  

**CHAPTER 5 CENSORED BROADCAST ALGORITHM** ................. 33  
5.1 Algorithm Description .............................................. 33  
5.2 Simulation Results .................................................. 35  
5.3 Testbed Results ...................................................... 38  

**CHAPTER 6 CONCLUSION** ............................................... 43  
6.1 Future Directions .................................................... 43  

**REFERENCES** .............................................................. 44
CHAPTER 1

INTRODUCTION

1.1 Context

Wireless sensor networks (WSNs) have become ubiquitous for applications that involve monitoring or data collection. They are comprised of low-cost nodes, or motes, equipped with a variety of sensors that are connected via a wireless interface. Despite the fact that each node in the system may not be particularly computationally powerful, several of them can be used in this manner to cooperatively solve a problem or perform a task quickly and efficiently. Recently WSNs have been used for applications such as monitoring water pollution [1], air pollution [2], [3], and monitoring the health or failure of large-scale structures or machines[4]. Such devices have thus been shown to be ideal for these types of applications where the problem or task in question is to be solved in a distributed manner.

One popular class of distributed algorithms that lends itself naturally to WSNs is that of distributed optimization. In standard optimization problems, the function over which the optimization is taking place, or global function, is known \textit{a priori}. In distributed optimization, the global function is split into several local functions, each of which is known to only a single node or agent in the system. The goal, therefore, is to use this limited local information at each node, and, via inter-node communication, arrive at the optimum for the global function. The nodes do not send their local functions or any other local information to the other nodes, otherwise it would become infeasible with large networks or complicated local functions. Ideally, we would want the nodes to arrive at the same value for their estimated optimum, and so consensus is a secondary objective as well. There have been various approaches to this problem in several different settings, but this thesis will primarily focus on gradient or subgradient-based methods.
One early proposed method for distributed optimization in WSNs is the incremental algorithm proposed in [5]. This algorithm was a decentralized adaptation of the incremental sub-gradient method presented in [6]. This algorithm requires reliable communication among the nodes and it proceeds by iteratively passing the current estimate of the optimum through all of the nodes. The nodes each update the estimate with their local gradient information. This also implicitly requires that the communication network contains a cycle. In [7], it is shown that this algorithm still converges to a value close to the true optimum when the gradients are corrupted by noise. It is important to note that in this algorithm, only one node is updating the estimate of the optimum at any given time.

The incremental algorithm can be contrasted with the gossip-based approaches of [8], [9], and [10]. In these algorithms, there is only a connectivity requirement on the communication graph, but there is no implicit requirement of the existence of a cycle that passes through all of the nodes as there is with the incremental algorithm. Nodes exchange information only with their immediate neighbors in all of these algorithms. However, this thesis focuses only on the broadcast-based approach of [10], to contrast with the incremental algorithm. This is because there is no clear fully decentralized, asynchronous implementation of either the distributed dual averaging algorithm of [9] or the gossip-based subgradient algorithm of [8] for a WSN. This is because unlike the incremental algorithm, multiple nodes are updating at any given time. In a WSN, this will lead to interference among the different nodes without some kind of multiplexing, either in time, frequency or space. This requires some kind of central coordination for resource allocation. Furthermore, any message sent by one node to a neighbor must be met immediately by a response from that neighbor. Due to the lossy nature of the wireless channel, this may not always be a practical assumption for a WSN. While there has been some investigation into adapting the dual averaging method to be agnostic to communication delays [11], it is not resilient to message loss. The broadcast algorithm from [10] does not require any such coordination, and is reasonably agnostic to message loss and noisy gradients. It can, therefore, be implemented with minimal overhead on a WSN.
1.2 Overview

The focus of this thesis is to implement the aforementioned algorithms from the literature that address the distributed optimization problem and implement them on an actual WSN testbed. The algorithms will be used to localize a light source, which is served by a desk lamp. These nodes must find and agree on the location of the lamp without any central coordination or computation. Therefore, the implemented algorithms are fully decentralized, in that there is no data fusion center and no coordination for resource allocation. This differs from implementations such as the one presented in [12] which relays all sensor information to a fusion center for processing. In this thesis, we first give formal descriptions of the general distributed optimization and the specific formulation of the distributed localization problem on the testbed. The incremental algorithm and the asynchronous broadcast algorithm are formally introduced and implementation concerns and results are discussed. Finally, two modifications to the broadcast algorithm are introduced with a discussion on their impact on convergence speed and stability.

1.3 Notation

In this section we introduce some of the notation that will be used in the following chapters of this thesis.

- Scalars are written in lowercase (e.g. $x$, $y$).
- Vectors are written in boldface lowercase (e.g. $\mathbf{x}$, $\theta$).
- $\nabla f(\mathbf{a})$ indicates the gradient of the function $f(\mathbf{x})$ evaluated at $\mathbf{x} = \mathbf{a}$.
- Sets are written in calligraphic font (e.g. $\mathcal{X}$).
CHAPTER 2

DISTRIBUTED OPTIMIZATION FORMULATION

In this chapter we will first formally introduce the general distributed optimization problem. We will then discuss the specific formulation for the localization problem. This will require the development of a model for the light intensity observations obtained from the sensor motes or nodes. A brief introduction to the sensor motes and physical testbed will be provided to serve as motivation for the choice of model and localization problem formulation.

2.1 General Problem

As stated earlier, distributed optimization is the problem of finding an optimum to a global function, which is not known locally to any node in the system. Rather, each node has access to a local function, which is a piece of the global function. The simplest, and most often used formulation of this problem is when the global function is the sum of the local functions, which is shown in Equation (2.1).

\[
\begin{align*}
\text{minimize} \quad & F(x) = \sum_{i=1}^{N} f_i(x) \\
\text{subject to} \quad & x \in \mathcal{X}
\end{align*}
\] (2.1)

Here \( F(x) \) is the global function and \( f_i(x) \) represents the local function available only to node \( i \). \( \mathcal{X} \) indicates a constraint set on \( x \), the parameter over which \( F \) is being optimized. This problem formulation lends itself well to the distributed setting as both the global function and its gradient are the sum of the local functions and local gradients respectively.
Without some kind of similarly separable structure, it may not possible to solve the problem using a distributed algorithm that uses only local information; meaning that the local functions and their gradients are kept private to each node. The algorithms developed in [5], [7], and [10] which will be used throughout this thesis all address this particular formulation of the distributed optimization problem. It should be noted that these algorithms do not require the existence of the actual local gradients, \( \nabla f_i \), but will converge as long as a subgradient of \( f_i \) exists for all \( i \) and for all \( x \in \mathcal{X} \), along with some other regularity conditions on the \( f_i \)'s and \( \mathcal{X} \). The function \( \nabla f_i(x) \) is defined to be a subgradient of \( f_i \) if Equation (2.3) holds.

\[
\nabla f_i(x)^T(y - x) \leq f_i(y) - f_i(x) \\
\forall x, y \in \mathcal{X}
\]

It is useful that full differentiability of the \( f_i \)'s is not a requirement to solve the general problem in a distributed fashion. However, this property will not be necessary for the localization problem formulation as the \( f_i \)'s will all be differentiable, so \( \nabla f_i \) will be the true gradient of \( f_i \). Another point to note is that if the \( \nabla f_i \)'s are noisy, they become random variables and the problem becomes a stochastic optimization problem. This is relevant to the localization problem as the light intensity measurements from the sensors will be noisy. Since these noisy measurements are used to compute the local gradients, the localization problem will be stochastic in nature.

For any algorithm we use to solve this distributed optimization problem, whenever a local gradient computation is required, we can consider two options for the computation. The first option is to simply estimate the true local gradient, assuming that it is the expectation of the noisy gradient as follows.
\[ \nabla f_i(x) = E(\nabla \tilde{f}_i(x)) \]  \hspace{1cm} (2.4)

\[ \nabla f_i(x) \approx \sum_{k=1}^{K} \nabla f^k_i(x) \]  \hspace{1cm} (2.5)

Here \( \nabla \tilde{f}_i(x) \) indicates the \( k \)th sample of the noisy gradient, and \( E(\cdot) \) indicates the expectation. This is simply estimating the expectation of the noisy gradient as a running average of samples of the noisy gradient. This estimate is guaranteed to converge to the true gradient as \( K \) increases to infinity by the law of large numbers. This seems like a reasonable solution to the problem of noisy gradients, but is inadequate for the following reasons. If the gradients are very noisy, meaning that the signal-to-noise ratio (SNR) is low, \( K \) might need to be very large to obtain a good estimate of the true gradient. If the algorithm is to be implemented on a WSN, whenever a local gradient computation is required, a sensor node will have to wait to poll the sensor \( K \) times before proceeding. Sensors often have some maximum polling frequency and so this may slow down any algorithm considerably. As the value of \( K \) may need to be adjusted as the noise level changes, this also introduces the additional problem of estimating the noise level in the system.

The second, much simpler option is to simply use the sample of the noisy gradient, \( \nabla \tilde{f}_i(x) \), whenever a local gradient computation is required. Despite the fact that the gradient that is actually used by any node at any single time step is likely to be incorrect due to the corruption by noise, it is shown in both [7] and [10] for their associated algorithms that this approach will still allow convergence to the true optimum. This is a powerful result as the nodes do not need to perform any additional estimation procedure which could slow down the optimization algorithm. This approach will be utilized in all of the WSN implementations shown in this thesis.

2.2 Localization Problem

The optimization problem formulation that will be used to localize the light source in the WSN testbed will now be presented. The idea behind this is to develop an observational model as to how the sensor measurements vary as
the source is moved to various locations relative to the sensor node location, and to use this model to determine the true location of the source. Given this model, we can pose the localization problem as a least-squares problem in which the goal is to minimize the sum of squared-errors between the model and local observations across all of the nodes [5], [13]. This is formally stated as follows.

\[ \hat{\theta} = \arg \min_{\theta} \sum_{i=1}^{N} E((k_i - g(x_i, \theta))^2) \]

subject to \( \theta \in \mathcal{X} \) \hspace{1cm} (2.6)

Here \( \hat{\theta} \) represents the estimate of the source location, \( x_i \) is the location of node \( i \), \( k_i \) is the noisy sensor measurement at node \( i \), and \( g(x_i, \theta) \) is the observational model for \( k_i \) given the node location and a source location. The expectation in the minimization problem is required, as the sensor measurements are noisy. For the purposes of localization, both \( x_i \) and \( \theta \) could be two or three dimensional, depending on which space we are localizing over. For the implementations presented in this thesis, however, we will only be considering localization over a two-dimensional space. Our formulation relies heavily on the development of the observational model, \( g(x_i, \theta) \). The sensor measurements are proportional to the received intensity of light at the sensor node, and so we might expect the model function to be of an inverse-square form if we assume that the source is an isotropic point source. In particular,

\[ g(x_i, \theta) = \frac{A}{r_i^2 + B} + Z_i \] \hspace{1cm} (2.7)

\[ r_i = \|\theta - x_i\|_2 \] \hspace{1cm} (2.8)

Here \( A \) and \( B \) are constant model parameters that attempt to capture the specific type of light source that is present, and \( Z_i \) is some form of additive noise. This form of observational model is similar to the one proposed in [5] for acoustic localization, which makes sense as both light and sound intensity decay in a manner that is inversely proportional to the distance to the source. The parameter \( B \) is necessary to ensure that the function has some fixed
maximum when \( r_i = 0 \). This is to ensure that the function and its gradient are bounded and to more closely model actual sensors.

### 2.2.1 Sensor Motes

In order to discuss how the model parameters are developed, we provide a brief introduction to the actual sensor motes that are used in the testbed throughout this thesis. The testbed will consist of several low-power TelosB sensor motes developed by Berkeley [14]. Figure 2.1 shows one such sensor mote.

![A TelosB Sensor Mote](image)

Each mote has two integrated light sensors, one temperature sensor, and one humidity sensor, but has the capability to include additional sensors. There are two Hamamatsu photodiodes that serve as the light sensors; one having a greater sensitivity in the visible light range and the other having a greater sensitivity in the infrared range. In this thesis, only the visible light sensor will be utilized. The sensor measurements are proportional to the photodiode current, which in turn is proportional to the received intensity of light. The motes are powered by a single 8 MHz, 16-bit MSP430 micro-
controller and communication is provided by a CC2240 radio with the IEEE 802.15.4 standard. The motes run a real-time, open-source operating-system (OS) known as Contiki OS [15]. This OS allows for simple programming of the motes, and provides useful libraries that facilitate sensor management and communication. One important point to note is that the microcontroller has no floating-point unit, and so all local computations must be in fixed-point format. This presents a key constraint on the model and optimization problem that is used, as we would like all computations to be performed on the WSN itself without a fusion center. Due to this fixed-point processor and the fact that the maximum addressable data size in Contiki OS is 64 bits, we have a precision limitation. We can effectively choose where to place the decimal point within the 64 bits, but this will constrain the maximum value that the system can utilize. Thus, if a particular optimization formulation or model function requires very high precision, while still having large values present in any computation, it will be unsuitable for implementation on the testbed. The least-squares formulation in Equation (2.6) coupled with the inverse-square model in Equation (2.7) was chosen over other potential formulations, as it performed well within this precision constraint. More complex models may be slower in implementation as we might need to use iterative methods to perform integer computations such as square roots. Such methods would also suffer greater penalties from precision errors.

2.2.2 Model Parameters

In order to determine the model parameters, data from the sensor motes is collected, while they are positioned at various distances from the light source. The role of the light source is served by a desk lamp. Some error will be incurred in the model by representing this light source as a symmetric, isotropic point source, but as we will show, we can still get some reasonably accurate localization by using this distributed optimization method. Data is collected for two different arrangements of the source and sensors, shown in Figures 2.2 and 2.3. One configuration is linear, where the sensors are all placed 10 cm apart and the other is a 3 x 3 square grid with a grid spacing of 30 cm. The localization will be performed on the grid, so it is important to include data collected from this grid topology with the lamp placed in a
random point in the interior of the grid to train the model parameters.

In the grid configuration, the lamp is placed 13 cm from the left edge of the grid, and 17 cm from the bottom edge of the grid. These measurements are relative to the center of the bulb. In both configurations and for all implementations in this thesis, the height of the lamp is kept fixed at 15 cm, or 13 cm relative to the sensor height. Changing the height of the lamp would require re-training of the model parameters. Furthermore, the height itself is irrelevant as the localization is only performed in a two-dimensional space. It should be noted that all data is collected with the room lights turned off. In Figures 2.2 and 2.3, the room lights are on to better show the arrangement of the sensors. However, with the room lights on, the sensor measurements are too noisy to discern the effect of the desk lamp. When
turned on, each sensor mote is programmed to repeatedly broadcast its sensor value. One sensor mote is plugged into a computer to collect these sensor measurements and record them. From each sensor mote in both configurations, 100 measurements are collected with the lamp turned off, and 100 measurements are collected with the lamp turned on. The measurements collected with the lamp turned off serve to estimate \( E(Z_i) \) in the model for each node \( i \). This is subtracted from the corresponding average of the measurements with the lamp turned on. A least-squares regression was performed on this mean-subtracted sensor data versus the distance to the lamp to determine the model parameters \( A \) and \( B \). An additional constraint that \( \frac{A}{B} = 1000 \) is added to ensure that the maximum sensor reading never exceeds 1000, which was observed to be a reasonable upper bound for the particular lamp that was used. The result of that regression is shown in Figure 2.4. Each blue asterisk indicates a particular sensor measurement and distance pair while the red dashed line indicates the estimated \( g(x_i, \theta) \) function.

The determined model parameters, rounded to an integer \( B \), were \( A = 48000 \) and \( B = 48 \). It can be seen in Figure 2.4 that the model is fairly good when the distance to the source is greater than 15 cm, but breaks down at closer distances. This means that the localization will likely fail when the light is close to being directly over any sensor mote in the grid. However, it will be shown that the localization is still fairly accurate for positions on the interior of the grid. As the model parameters were determined with the lamp height fixed to 13 cm relative to the sensor plane, when performing the localization in a two-dimensional space, we can absorb the effect of the lamp
height into the $B$ parameter and the following final problem formulation is obtained.

\[
\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} f_i(x_i, \theta)
\]

\[
\theta \in \mathcal{X}
\]

\[
f_i(x_i, \theta) = E((k_i - g(x_i, \theta))^2)
\]

\[
g(x_i, \theta) = \frac{48000}{r_i^2 + 217} + Z_i
\]

\[
r_i = \|\theta - x_i\|_2
\]

One final note about this particular formulation is that $F(\theta)$ is not convex as written. It will have many local maxima and minima. To observe this directly, we can simulate a system with noise-free observations and plot the global function. In this simulation, the observations are generated exactly according to the model and are noise-free, and so the location that minimizes the global function will be the exact location of the source. The nodes are placed in a 3 x 3 grid arrangement with a 30 cm spacing and the source is placed at the coordinate (13 cm, 17 cm) relative to the bottom-left corner of the grid. This is the same arrangement of sensor nodes and the light source that was used to train the model parameters. Since the global function is a sum of squared-error functions, it is non-negative and so we would expect the global minimum to be achieved at the point (13, 17) with a global function value of zero.

Figure 2.5 shows the global function with a three-dimensional visualization and the global minimum is marked. Figure 2.6 displays the same function but with a color map to more easily see the behavior of the global function. As expected, the global minimum is zero and is achieved at the true location of the source, since the observations were generated exactly according to the model function. More importantly, however, we can see that there are several local maxima at the locations of the nodes. Despite the non-convexity of the problem, we can still achieve a reasonably accurate localization given a good initialization. We will also show how the broadcast approach is somewhat
less sensitive to this initialization than the incremental approach. Now that the localization problem has been appropriately formulated, the implementations and results of the chosen distributed optimization algorithms will be discussed in the following chapters.
CHAPTER 3

INCREMENTAL OPTIMIZATION ALGORITHM

In this chapter, the incremental algorithm from [7] and [5] is formally introduced. This algorithm is implemented to solve the localization problem as formulated in Chapter 2. Simulation and testbed results are presented to motivate the use of and to contrast with the broadcast algorithm shown in Chapter 4.

3.1 Algorithm Description

The cyclic incremental algorithm from [5], [7], and [16] is a distributed optimization algorithm intended to solve problems of the form of Equation (2.1). This algorithm is so named because it requires a cycle, specifically a Hamiltonian cycle, to exist within the network of nodes in the system. At any given point in time, there is only one estimate of the optimum present at exactly one node within the network. This node updates the current estimate using its local information and sends the new estimate to its downstream neighbor, which then performs a local update and passes the new estimate to its downstream neighbor. One iteration of this algorithm is completed when the estimate has traversed the entire cycle once; meaning that every node in the system has updated the estimate once using its local information. This update relation is shown in Equation (3.1) for a network of \( N \) nodes.

\[
\begin{align*}
  y_1 & = x_j \\
  y_{i+1} & = P_x[y_i - \alpha_j(\nabla f_i(y_i))] \\
  x_{j+1} & = y_{N+1}
\end{align*}
\]  

(3.1)

Here, the system is currently in iteration \( j \) of the incremental algorithm.
At the beginning of the iteration, the first node in the cycle sets its iterate equal to the current estimate of the optimum, $x_j$. Each node $i$ in the cycle receives an iterate from its upstream neighbor $i - 1$, performs a gradient update, and sends the iterate to node $i + 1$. In the gradient update step, $P_X$ indicates a projection onto the constraint set and $\alpha_j$ is the step size for iteration $j$. After all nodes have performed a gradient update, the new estimate for iteration $j + 1$ is obtained from the output of the $N^{th}$ gradient update.

There are a few important facets of this algorithm that must now be considered. Since there is only ever one iterate in the system, this algorithm can be implemented asynchronously, as long as message reliability is guaranteed in the system. Whenever a message is sent, it must be eventually received by the recipient. Without message reliability, the algorithm cannot make progress as the iterate would eventually stop at some node, potentially before the algorithm converges. This will pose a limitation when implementing the algorithm on a WSN, as the testbed results will show. The local gradient, $\nabla f_i(y_i)$, can be a sample of a noisy gradient, $\tilde{\nabla} f_i(y_i)$, while still allowing convergence to the optimum [7], as discussed in Chapter 2. This will be of particular importance in the testbed implementation as the noisy sensor measurements are used to compute the gradient. Though convergence to the global optimum is guaranteed only for a convex global function, convergence to a local optimum is still guaranteed in the event that the global function is non-convex [16]. This will result in a very high sensitivity to initialization as the simulation and testbed results will show. One final point to note is that the step size, $\alpha_j$, is allowed to vary with the iteration. The convergence of this algorithm is well defined for both constant and diminishing step sizes [16], though only fixed step sizes will be used in the implementations in this thesis. This is due to the issue of highly limited precision on the sensor motes that was discussed in Chapter 2. As the motes cannot resolve very small values, any step size that diminishes with the iteration number will quickly fall below the sensor mote precision before convergence is achieved. Furthermore, in [16], it is noted that whenever quantization errors are present, better performance is achieved when a fixed step size is used. Since limited machine precision does induce quantization errors, using fixed step sizes for the testbed implementation is a reasonable approach.
3.2 Simulation Results

In this section, simulation results for the localization problem as formulated in Equation (2.9) are presented. All simulations are performed in MATLAB R2014b. The sensor nodes are arranged in a 3 x 3 grid with a 30 cm grid spacing. The simulated light source is placed at the coordinate (47 cm, 41 cm) relative to the bottom left sensor node, a different interior point than what was used to train the model parameters.

![Figure 3.1: Incremental Algorithm Network Topology](image)

Figure 3.1: Incremental Algorithm Network Topology

Figure 3.1 shows the network topology that is used in the sensor node grid. The circles indicate the sensor node positions and the lines indicate the presence of a communication link. The “x” indicates the location of the simulated light source. The link from the center node to the bottom-left node does not technically exist in a grid topology, but can be simulated by relaying the messages through an intermediate node that does not perform any updates on the relayed message. The bottom-left node begins the algorithm and it proceeds counterclockwise throughout the network cycle. In the simulation, a constant step size of $\frac{2}{512}$ is used and sensor observations are generated according to the model in Equation (2.9). Gaussian noise is added to the sensor observations at an SNR of 20 dB. The constraint set $\mathcal{X}$ is the square whose corner points are (-30, -30) and (90, 90). The initialization point for the first node in the cycle is placed the center of the grid (30, 30). Figures 3.2 and 3.3 show the convergence of the $x$- and $y$-coordinates of the estimate of the source location for one trial of the incremental algorithm simulation. The red dashed lines indicate the true values for the location.
of the source. Nine updates in the “update count” indicates one round of the incremental algorithm as there are nine nodes present in the cycle. Figure 3.4 plots the evolution of the estimate, overlaid on the sensor topology. We can see that the estimate converges close to the true location of the source within approximately 80 local gradient updates by the nodes, though it somewhat oscillates thereafter.

![Figure 3.2: Simulation Estimate of x-Coordinate of the Source](image1)

![Figure 3.3: Simulation Estimate of y-Coordinate of the Source](image2)

This simulation was repeated 1000 times to obtain statistics on the convergence of the algorithm. Here the number of updates to converge to the true location of the source is defined as the index of the first update that moves the estimate to within 0.5 cm of the true location. Following this definition, the average number of local gradient updates required to converge
A histogram of the 1000 simulations is shown in Figure 3.5. This seems to be a promising result as the algorithm appears to be fairly accurate. This breaks down, however, with a poorer choice of the initialization point, as shown in Figure 3.6, where evolution of the estimate is overlaid on the sensor topology. The location estimate of the source was initialized to the point (0, 0) in this simulation, and we can see that the estimate is essentially caught in the lower-left cell of the grid and never leaves. This shows that the incremental algorithm is very sensitive to the initialization for the non-convex formulation, and with a poor choice of initial point, the algorithm will not correctly identify the location of the source.
3.3 Testbed Results

In this section, the implementation of the incremental algorithm studied in Section 3.2 using the same TelosB wireless sensor motes [14] that were discussed in Chapter 2. The motes are arranged in a 3 x 3 grid setup as in Figure 3.1. The lamp is placed such that the center of the bulb is directly over the same (47, 41) point relative to the bottom-left sensor node as in the simulations. This physical arrangement is shown in Figure 3.7. The lower-left node will begin the algorithm with the point (30,30) for an initial estimate for the source location. In order to collect data from the sensor nodes, there is one node plugged into a desktop computer. Whenever any of the sensor nodes in the grid perform a gradient update, they send one message to the desktop node for data collection and then they send the same message to the downstream sensor node. No processing happens at the desktop, and so the system is still fully decentralized.

To ensure some degree of message reliability without synchronization between the nodes in the testbed, a simple retransmission and acknowledgment system is used. When a node attempts to send its estimate to its downstream neighbor, it waits for one second to receive an acknowledgment message from the downstream neighbor. If no acknowledgment is received, it retransmits its estimate and waits again. To prevent multiple location estimates from cycling throughout the network, nodes also pass an iteration counter along with the current location estimate. This iteration counter is incremented whenever a node performs a gradient update. In this way, if a node receives
a message from its upstream neighbor which has a lower iteration counter than its own, the node can simply ignore it. To prevent nodes from causing too much interference by repeated retransmissions, the maximum number of retransmissions is limited to five. This does mean that there is a small, non-zero probability that the algorithm will stop making progress due to message failure, but this setup proved to be reliable enough to reach convergence before stopping. The system precision is set to 9 bits, meaning that the smallest resolvable value is $\frac{1}{512}$. A constant fixed step size of $\frac{2}{512}$ is used; the same value used in the simulations in Section 3.2. This step size is chosen, as any larger step size causes too much oscillation in the location estimate and the convergence is poor. Sensor measurements are obtained from the photodiodes on the TelosB motes, and so they do not exactly follow the observational model given in Equation (2.9). Prior to each gradient update, the node that is updating polls the light sensor for a measurement. This measurement is the sample of the noisy measurement, and is used to compute the sample of the noisy gradient. Each node has access to an estimate of $E(Z_i)$ from Equation (2.7). This value is subtracted from the sensor measurement, $k_i$, in each update step prior to the computation of the local gradient. The constraint set $\mathcal{X}$ is once again the square whose corner points are (-30, -30) and (90, 90). Figures 3.8 and 3.9 show the convergence of one trial of the incremental algorithm implementation on the testbed. The red dashed lines indicate the approximate point of convergence for the estimate of the x- and y-coordinates of the source. Figure 3.10 shows the evolution of the source location estimate, overlaid on the sensor topology. Only one trial
is shown for the testbed implementation since it is much more time intensive to implement and run these algorithms on a physical system as opposed to running simulations.

![Figure 3.8: Testbed Estimate of x-Coordinate of the Source](image)

![Figure 3.9: Testbed Estimate of y-Coordinate of the Source](image)

We can see that the algorithm converges within approximately 100 local gradient updates, or just over 11 cycles through the network, and oscillates thereafter. The estimate of the source location converges approximately to the point \((52, 40)\) which represents a 5.09 cm error from the true source location. In the simulation, the algorithm converged almost exactly to the true source location, because the observations were generated exactly by the model function given in Equation (2.9). In the testbed implementation, as the sensor motes poll the photodiode sensor for physical measurements, some
error in the model itself is induced. This error arises from the fact that the desk lamp is modeled as an isotropic point source of light, despite being a complex, possibly asymmetric, three-dimensional source. Furthermore, it is also possible that the measurement SNR is less than the 20 dB figure that was used in the simulations. Despite this simplification, we still get a localization that is reasonably close to the true location of the source. However, there are some noted disadvantages to this algorithm. It was observed that this localization implementation on the testbed typically took at least three minutes to converge. As the gradient computations are not particularly intensive, it suffices to say that the algorithm convergence speed suffers when ensuring the message reliability of the system. The retransmission and acknowledgment system may take several seconds in order to send just one
message to a downstream neighbor. Furthermore as Figure 3.11 shows, the
testbed implementation also exhibits the same sensitivity to initialization
for the non-convex formulation as the simulation in Section 3.2. Here the
initial value is set to the point (0,0) and it can be seen that the estimate
converges to some local optimum outside the grid. This poses a significant
problem as the starter node has no information with which to determine the
relative quality of an initialization point. Thus, to improve the performance
of the distributed localization on the testbed, an algorithm that is agnostic
to message loss and less sensitive to initialization is required. A gossip-based
approach might satisfy these requirements and obtain superior performance
to the incremental algorithm for the localization problem, as we show in
Chapter 4.
CHAPTER 4

BROADCAST OPTIMIZATION ALGORITHM

In this chapter, the broadcast optimization algorithm is formally introduced. This algorithm is implemented to solve the localization problem formulated in Chapter 2. Simulation and testbed results are shown to contrast the performance of this algorithm with the incremental algorithm from Chapter 3.

4.1 Algorithm Description

The broadcast distributed optimization algorithm from [10] also solves problems of the form Equation (2.1) but differs greatly from the incremental algorithm. The algorithm follows the structure of the broadcast consensus algorithm presented in [17], with added gradient updates. In this algorithm, each node maintains its own estimate of the optimum, rather than passing a single estimate between the nodes. Furthermore, links between nodes are allowed to independently fail with some probability, while still allowing convergence to the optimum. This is particularly useful for a WSN, as nodes do not have to expend power and time performing retransmissions to ensure message reliability. Each node maintains a local clock, which is typically modeled by a Poisson random process. The specific rates of the local clocks do not matter, but all of the clocks are assumed to have at the same rate.

At each tick of the local clock, node \(i\) broadcasts its current estimate of the optimum, \(x_i\). Out of the set of neighbors of node \(i\), \(\mathcal{N}\{i\}\), only some of them will receive the message as links are allowed to fail. If some node \(j \in \mathcal{N}\{i\}\) receives this message, it averages its local estimate with that of node \(i\) and performs a gradient update. This update relation is shown in Equation (4.1).
\[ x_j \leftarrow \beta x_i + (1 - \beta)x_j \]
\[ x_j \leftarrow P_{x_i}[x_j - \alpha_j(\nabla f_j(x_j))] \quad (4.1) \]

Note that the step size \( \alpha \) need not be coordinated among the nodes. Each node can select its own step size and the algorithm will still converge to the optimum. Nonetheless, consistent step sizes will be used in both the simulation and testbed results presented in this thesis. The parameter \( \beta \in (0, 1) \) is a mixing parameter and this can be used to control the convergence speed of the algorithm. For the implementations in this thesis, a \( \beta \) of 0.5 is chosen as it consistently provided the best results for the localization problem. Similarly to the incremental algorithm, the convergence of the broadcast algorithm is well defined for both fixed and diminishing step sizes. For the same reasons outlined in Chapter 3, only fixed step sizes will be used in this thesis. This algorithm is also robust to noise in the gradient computations and so the local gradients, \( \nabla f_i(x_i) \), are still allowed to be a sample of a noisy gradient, \( \nabla \tilde{f}_i(x_i) \).

### 4.2 Simulation Results

In this section, simulation results for the localization problem as described in Equation (2.9) are presented. The arrangement of the sensor nodes is the same 3 x 3 grid with 30 cm spacing as discussed in the earlier chapters in this thesis. The simulated light source is once again placed at the point (47 cm, 41 cm) relative to the bottom-left node.

Figure 4.1 shows the network topology of the grid. As in Chapter 3, each circle indicates the position of a sensor node, and each edge indicates that the adjacent nodes can communicate with one another. The red “x” indicates the true location of the source. A constant step size of \( \frac{8}{512} \) is used in the simulations as it is the largest step size, for a precision of 9 bits, that still converges well. Despite that fact that limited precision is not simulated, we would like to mirror the simulation as closely as possible when implementing the algorithm on the testbed. This is another reason why a step size of \( \frac{8}{512} \) is chosen for the simulations. In the simulations, each link in the topology
is allowed to fail with probability 0.2 and sensor measurements are subject to additive Gaussian noise at an SNR of 20 dB. The constraint set $\mathcal{X}$ is the square whose corner points are (-30, -30) and (90, 90). Unlike the incremental algorithm, we need not make any specific initialization to ensure the convergence of the algorithm. Each node’s local estimate is simply initialized to its own location. This information is known \textit{a priori}, as it is required to compute the local gradients. Thus, no node in the system needs any information about the relative quality of an initialization point. Each node has a local clock, modeled by a rate 1 Poisson process.

Since each node has its own local estimate, the performance of this algorithm is best examined by looking at the average location estimate in the network and the node disagreements. Figures 4.2 and 4.3 show the convergence of the x- and y-coordinates of the average estimate of the source location for one trial of the broadcast algorithm simulation. The red dashed line indicates the true coordinate of the source. We can see that convergence in both coordinates is achieved after approximately 150 total broadcast updates are sent. Figure 4.4 shows the maximum disagreement between any two nodes in the network after each broadcast update is sent. The maximum disagreement does not quite reach zero, but this is a moot point as after the average estimate converges, we can simply use an average consensus algorithm, similar to those presented in [17] and [18], to ensure that all nodes converge to the same average point. Figure 4.5 plots the path of the average estimate in the network, overlaid on the sensor topology to illustrate the success of the localization.
This simulation is repeated 1000 times to gain an understanding of the convergence statistics of the broadcast algorithm in this context. Once again, the number of updates to converge to the true location of the source is defined as the first broadcast update that moves the estimate to within 0.5 cm of the true location. Using this definition, the average number of broadcasts necessary to achieve convergence over the 1000 trials is 291.02. The histogram of the 1000 simulations is shown in Figure 4.6. It would seem as though far more message transmissions are required in this algorithm as compared to the incremental algorithm. However, the simulations in Chapter 3 did not account for message loss as these simulations do. We will see in the testbed implementation of the broadcast algorithm that this provides a significant advantage when implemented on a physical testbed. Furthermore,
this algorithm allows the sensor nodes to use their own location as an initialization point, and so it seems to be somewhat less sensitive to initialization than the incremental algorithm when used to optimize over the non-convex formulation.

4.3 Testbed Results

In this section we present results for the testbed implementation of the simulations from Section 4.2 using the same TelosB sensor motes [14] that were discussed in the earlier chapters in this thesis. The same 3 x 3 grid configuration from Figure 4.1 and the lamp is positioned such that the center of
the bulb is over the (47, 41) point relative to the bottom-left sensor node as shown in Figure 3.7. All nodes initialize their local estimates to their own location to the grid, which is part of the private information that each node has access to. When a sensor mote performs a gradient update, it polls the light sensor for a new sample, which is used to compute the local gradient. As discussed in Chapter 3, each sensor node has access to an estimate of \( E(Z_i) \) from Equation (2.7) that is subtracted from the sensor measurement, prior to the computation of the local gradient. This gradient is, therefore, the sample of a noisy gradient. Each sensor mote in the grid is driven by a local clock that ticks every four seconds. This broadcast rate is chosen as any faster clock rate seems to create too much interference and likely drives the message loss probability too high to converge reliably in a reasonable amount of time. The clocks are not synchronized but are driven by a precise crystal oscillator [14], and so they can be assumed to run at the same rate, but with different offsets. There is no retransmission of messages, as this algorithm is agnostic to message loss. The system precision is set to 9 bits, thereby fixing the smallest resolvable value to \( \frac{1}{512} \). A constant fixed step size of \( \frac{8}{512} \) is used; the same value used in the simulations in Section 4.2. The constraint set \( \mathcal{X} \) is again the square whose corner points are (-30, -30) and (90, 90). In order to collect the data, one sensor mote is plugged into a desktop computer. At each tick of the local clock, each sensor mote in the grid sends a copy of the broadcast message to the desktop mote for data collection prior to broadcasting its estimate to the nodes within the grid. The communication topology in Figure 4.1 must be artificially imposed as the nodes in the grid are all within

Figure 4.6: Simulation Convergence Histogram
each others’ transmission range. This is to gain an understanding of the performance of the algorithm in a realistic, physical testbed. The algorithm will converge faster, given a fully connected network topology [10], but this is not a reasonable assumption for large WSNs. Thus to provide more meaningful results, we artificially impose the topology in Figure 4.1, by forcing nodes in the grid to ignore any messages that were not sent by their immediate neighbors in the grid. As discussed in Section 4.2, it is most instructive to examine the average source location estimate amongst the nodes, and the maximum disagreement between nodes in the grid after each broadcast is sent. Figures 4.7 and 4.8 show the convergence of one trial of the broadcast algorithm implementation on the testbed. The red dashed lines indicate the approximate convergence value. Figure 4.9 shows the maximum disagreement between any two nodes in the network after each broadcast update is sent. Similarly to the simulations from Section 4.2, the maximum node disagreement does not converge exactly to zero, but we can always stop updating the gradient post-convergence and simply perform averaging with each broadcast as per [17] to ensure that the neighbor disagreement converges to zero. Figure 4.10 shows the evolution of the source location estimate, overlaid on the sensor topology.

![Graph of Testbed Average Estimate of x-Coordinate of the Source](image)

Figure 4.7: Testbed Average Estimate of x-Coordinate of the Source

We can see that the algorithm converges after approximately 100 broadcast messages are sent. Since each of the nine nodes in the grid broadcasts every four seconds, this test took approximately 45 seconds to converge with roughly 11 broadcasts per node. This already represents a significant improvement over the cyclic incremental algorithm from Chapter 3, which took
on the order of several minutes to converge. Although the convergence is much faster, the localization is not appreciably more accurate than the incremental algorithm. The estimate of the source location converged approximately to the point (52, 40.5) which still represents a 5.02 cm error from the true source location. This error is most visible in Figure 4.10. This error is not present in the simulations in Section 4.2 as the observations were generated directly from the model in Equation (2.7) It is likely that the errors in the model, and the assumption of an isotropic point source of light, places a hard limit on the achievable accuracy of the localization. Nonetheless, we obtain a much faster convergence while using the broadcast approach as opposed to the incremental approach, which is attributable to a number of factors. We are able to use a larger step size, \( \frac{8}{512} \), for the broadcast algorithm as opposed
to $\frac{2}{512}$ for the incremental algorithm. The use of any larger step size with the incremental algorithm caused the estimate not to converge. Comparing Figures 3.8 and 3.9 to Figures 4.7 and 4.8, we can clearly see that there is far less oscillation post-convergence in the broadcast algorithm, despite using a larger step size. We also do not have to expend time retransmitting the same message to ensure message reliability, as the broadcast algorithm is agnostic to message loss. This is a major advantage as WSNs use wireless communication by definition and the channel is all but guaranteed to be lossy. Another advantage the broadcast algorithm has over the incremental algorithm in this non-convex context is that nodes do not need to select a special initialization point, but can simply initialize their estimate to their own location. In the incremental algorithm, without initializing the estimate to the center of the grid, the algorithm does not localize the light source correctly. This property of the broadcast algorithm for the non-convex localization problem may not necessarily generalize to alternative topologies, or even larger grid sizes, but the modification to this algorithm that is discussed in Chapter 5 will allow each sensor node to initialize to its own location, and the location estimate will converge with an arbitrary grid size.

Now that the results for the incremental and broadcast optimization algorithms have been discussed, a modified version of the broadcast algorithm that further improves the localization performance in will be introduced in Chapter 5.
CHAPTER 5

CENSORED BROADCAST ALGORITHM

In this chapter, a modified version of the broadcast optimization algorithm from [10] is presented. This new algorithm is shown to converge much faster than the unaltered broadcast algorithm for the localization problem. It is specifically intended for least-squares problems, such as the localization formulation presented in Chapter 2.

5.1 Algorithm Description

The motivation for this censored broadcast algorithm comes from the intuition that perhaps only a few of the $N$ nodes are actually needed to localize the source. In the noise-free case, if the observations are determined exactly by an inverse-square formula, the localization problem is essentially an attempt to find the unique intersection of $N$ spheres of different radii centered at the sensor node locations [13]. From a geometrical standpoint, only four nodes would be needed to locate the source in a three-dimensional space, and only three nodes would be needed to locate the source in a two-dimensional space in that scenario. Another source of intuition is that perhaps not every node has equally reliable measurements. The measurements from the farthest nodes in the testbed when the incremental and broadcast algorithms were implemented proved to be very small, and could easily be corrupted by noise. Using these measurements might not contribute in any meaningful manner to the localization accuracy. This idea is supported by Figure 5.1, which shows the empirical SNR ($\mu/\sigma$) for 100 measurements collected from the testbed linear arrangement from Figure 2.2. We can see that the measurement SNR is approximately inversely proportional to the distance from the light source. Thus, measurements from farther nodes are inherently less reliable, and we could potentially disregard them in the optimization. We will show in the
following sections that disregarding these unreliable measurements does not impose any penalty on the localization accuracy.

In order to remove observations from the nodes that are too far from the source, we use a simple threshold on the average sensor measurement. If a sensor node’s average measurement is above the threshold, it places itself in the set of participating nodes. This threshold should be chosen such that for an interior point near the center of the grid, only the four closest nodes participate. Since this partition should be isotropically spatial in nature and the topology is a grid, the network of participating nodes will remain connected, which is a requirement for convergence of the broadcast algorithm [10]. It should be noted that an SNR-based threshold would likely apply in a more general least-squares setting, but this simple threshold was used to ensure the connectivity of the set of participating nodes. In this algorithm, nodes also keep track of how many hops they are from a participating node, $h_i$. At each tick of the local clock, node $i$ broadcasts $h_i$ in addition to the Boolean variable $b_i$, which indicates its participation in the optimization, and its local estimate of the optimum, $x_i$. Initially, $h_i$ is set to zero if the node is participating, and some number much larger than the diameter of the network if the node is not participating. Nodes that are participating ignore any broadcasts from nodes that are not participating. This restricts the optimization to only the set of participating nodes. Thus there are three types of exchanges that are possible in this framework: a broadcast from a participating node $i$ to another participating node $j$, a broadcast from a participating node $i$ to a non-participating node $j$, and a broadcast from a
non-participating node $i$ to a non-participating node $j$. In the first case, as both nodes are participating, we run the typical broadcast optimization update rule given in Equation (4.1). In the second case, the following update rule is performed.

\[
    h_j = 1 \\
    x_j \leftarrow x_i
\]  

(5.1)

Non-participating nodes do not perform any gradient updates, and they simply take any estimate from a participating node to be their own. In the final interaction between two non-participating nodes, if $h_j > h_i$, the following update is performed.

\[
    h_j = h_i + 1 \\
    x_j \leftarrow x_i
\]  

(5.2)

Between two nodes that are an equal number of hops from the set of participating bodes, no update is performed. Thus nodes that are further away from the set of participating nodes simply take the estimates they are given. This allows the converged estimate between the set of participating nodes to propagate throughout the network.

5.2 Simulation Results

In this section, simulation results for the same localization problem from Equation (2.9) are presented. The same $3 \times 3$ grid with 30 cm spacing is used for the sensor node topology. The simulated light source is placed at the point (47 cm, 41 cm) relative to the bottom-left node as it was for the previous chapters in this thesis. The set of update rules from Section 5.1 effectively enforces the topology shown in Figure 5.2 after a few broadcasts have been sent. The arrows indicate a directionality in the links between nodes, as participating nodes simply ignore broadcasts from non-participating nodes, and non-participating nodes ignore broadcasts from non-participating nodes.
who have higher hop counter values. Note that the lower-left node will accept messages from both neighbors.

![Figure 5.2: Censored Broadcast Effective Topology](image)

The red “x” marks the location of a source, and the red circles indicate the set of participating nodes. Those four nodes will eventually converge in their estimate of the source location, and their estimates are simply copied by the group of outer nodes. In order to serve as a proper comparison to the simulation results from Chapter 4, we use the same simulation parameters: a step size of $\frac{8}{512}$, a link failure probability of 0.2, sensor measurements that are subject to additive Gaussian noise at an SNR of 20 dB, and the same square constraint set $\mathcal{X}$ whose corner points are (-30, -30) and (90, 90). As before, we initialize each node’s local estimate of the source location to its own location. Each local clock is all modeled by a rate 1 Poisson process. The observations are generated exactly by the model given in Equation (2.7).

Since we are running a modified version of the broadcast algorithm, the performance of the algorithm is best defined by the average estimate of the source location amongst the nodes in the network and the maximum neighbor disagreement after each broadcast. Figures 5.3, 5.4 and 5.5 show the results of one trial of this simulation. The red dashed lines once again indicate the true location of the source. We see that convergence in both coordinates and the neighbor disagreement is achieved in less than 50 broadcasts. Figure 5.6 plots the path of the average estimate in the network, overlaid on the sensor topology. We once again see that convergence to the true source location is achieved. Figure 5.7 shows the histogram of 1000 trials of this particular simulation. The number of broadcasts required to converge for
each trial is once again defined as the first broadcast update that moves the average estimate in the network to within 0.5 cm of the true location. Using this definition, the average number of broadcasts needed to converge over the 1000 simulations is 164.49. This is approximately half of the result for the same simulation for the unaltered broadcast algorithm from Chapter 4 and represents a significant improvement. The main reason for this improvement is that fewer nodes are actually running the broadcast algorithm over a much smaller topology. Furthermore, since the set of participating nodes initialize their estimates to their own location, and participating nodes are by definition close to the source, the average estimate amongst the nodes begins closer to the true location of the source than in the unaltered algorithm. The interesting point however, is that no penalty is paid in terms of
the localization accuracy. This property will continue to hold in the testbed results presented in Section 5.3.

5.3 Testbed Results

In this section we implement the censored broadcast algorithm on the same testbed of TelosB wireless sensor motes [14] that have been used to demonstrate the previous algorithms in this thesis. The testbed and lamp are arranged in the same manner as shown in Figure 3.7 such that the center of the lamp bulb is directly over the (47, 41) point used in the simulations. The sensor nodes initialize their local estimates to their own locations. Prior
to the start of the algorithm, each node collects 100 measurements from the light sensor to estimate of $E(Z_i)$ from Equation (2.7), and to compare the average measurement to the threshold to determine if it will participate in the optimization. Participating nodes then set their hop counters to zero, and non-participating nodes initialize their hop counters to 100. As discussed in Chapters 3 and 4, the estimate of $E(Z_i)$ is subtracted from the sensor measurement, prior to the computation of the sample of the noisy local gradient during each update. The same step size, $\frac{8}{512}$, and constraint set from the simulations in Section 5.2 are used. Just as in Chapter 4, the sensor motes each have a local clock that ticks every 4 seconds. The same procedure involving a sensor mote plugged into a desktop that was used in Chapter 4 is used here for data collection. And as explained in Chapter 4, the sensor motes artificially impose the grid communication topology by ignoring messages that are not sent by their neighbors. Figures 5.8 and 5.9 show the convergence of one trial of the censored broadcast algorithm on the testbed. The red dashed lines indicate the approximate level of convergence for both coordinates. Figure 5.10 shows the maximum disagreement between source location estimates across any two nodes in the testbed after each broadcast update. Figure 5.11 shows the evolution of the source location estimate, overlaid on the sensor topology as in the results in the previous chapters in this thesis.

As we can see, convergence is achieved after 50 broadcasts. Since there are nine nodes broadcasting every four seconds, the localization takes approximately 22 seconds to converge. This is only half of the time the broadcast
algorithm from Chapter 4 takes to localize, and so it is in agreement with the simulation results from Section 5.2. From Figure 5.11 we see that the point of convergence is not the true location of the source. The approximate point of convergence is (52, 39.5) which represents a 5.22 cm error from the true source location. As discussed in the earlier chapters of this thesis, this error is due to limitations of the observational model. The simulations all converged exactly to the true source location as the observations were generated exactly from the model. However, the important point to take away from these results is that we do not suffer any significant penalty in localization accuracy between the broadcast and censored broadcast algorithms, but we gain a 50% improvement in convergence speed. For a much larger sensor network, it is likely that the speed improvement would be even greater.
the initialization procedure used in the censored broadcast algorithm extends very naturally to larger grid sizes. Since only the closest nodes initialize their estimates to their own locations, by definition, the initial average estimate among the participating nodes will be close to the true source location. For the unaltered algorithm, the initial average estimate will always be at the center of the grid, and so it is unclear if the convergence will extend to larger grid sizes. Thus, in this non-convex localization context, the censored broadcast algorithm is superior to the unaltered algorithm. It must be noted that this censoring can only apply in distributed optimization problems where not all of the agents need to participate in order to converge to the optimum. Thus, it is best used in these kinds of least-squares problems. Nonetheless, despite the fact that all three algorithms implemented in this thesis achieved
virtually the same localization accuracy, the censored broadcast algorithm
achieves by far the fastest convergence. Furthermore, the incremental algo-

rithm may not even converge properly unless a “good” initialization point is
chosen. This is a disadvantage that the censored broadcast algorithm does
not possess.
CHAPTER 6

CONCLUSION

This thesis explored the problem of distributed source localization using a least-squares optimization framework. Two algorithms from the literature that address the distributed optimization problem, the cyclic incremental algorithm [5], [7], [16] and the broadcast optimization algorithm [10] were introduced. These algorithms were implemented on an actual wireless sensor network testbed consisting of TelosB sensor motes [14]. It was found that the broadcast algorithm was far superior to the incremental algorithm when implemented on a physical wireless sensor network, due to the algorithm being agnostic to message loss and, to some extent, initialization.

A modified version of the broadcast algorithm was introduced and implemented. This censored broadcast algorithm further improved the localization speed over the unaltered algorithm. The key intuitions behind this algorithm were that not all of the sensors were needed to localize the source, and not all sensors had equal quality of information.

6.1 Future Directions

The work presented in this thesis has a number of future extensions. The censored broadcast algorithm greatly improved the convergence speed of the algorithm, without sacrificing the localization accuracy. One possible avenue for further investigation is the application of the algorithm in a general least-squares context. Furthermore, the specific trade-off between exclusion and convergence speed warrants further inspection. This work focused exclusively on the localization of a single light source, and it would be of interest to consider localization of multiple sources and perhaps different kinds of sources such as heat or odor sources. It may also be of some use to investigate the use of such a framework to track a time varying or moving source [12].
REFERENCES


