Iterative Majorization based Localization for Wireless Sensor Networks

A Thesis Submitted

in Partial Fulfilment of the Requirements

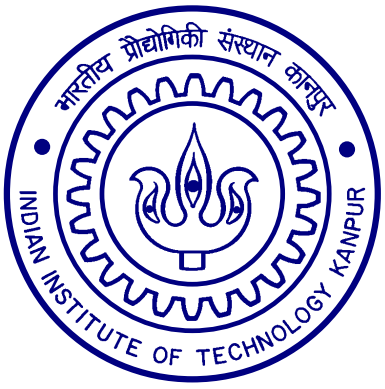
for the Degree of

Master of Technology

by

Vijayender Reddy K.

(Roll No. Y5827221)



to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

June, 2010

Certificate

This is to certify that the work contained in the thesis titled "Iterative Majorization based Localization for Wireless Sensor Networks" by K. Vijayender Reddy (Y5827221) has been carried out under my supervision and this work has not been submitted elsewhere for any degree.

(Dr. Joseph John)

Professor,

Department of Electrical Engineering,

Indian Institute of Technology Kanpur

May 2010

To Almighty.

Abstract

Location information is an important addition to the sensor data in a Wireless Sensor Network. This information is used in routing protocols, storage algorithms etc. In-network localization techniques that solve for the coordinates of its constituent nodes are contemporary topics of research. We have studied localization techniques with primary focus on multidimensional scaling. Multidimensional scaling techniques are less popular in Wireless Sensor Networks, because of the common notion that they are computationally intensive. Simplex algorithm, simulated annealing and iterative majorization are the relevant multidimensional scaling techniques that have been implemented and simulated, in this work, to ascertain their performance in the context of Wireless Sensor Networks. Based on our study we chose iterative majorization to implement localization. We have used certain results of MDS, which simplified the necessary computations. To bridge the gap between research and on field implementations experiments were carried out and the obtained data was used to model our simulations. In this work received signal strength intensity is used to estimate the distance between nodes. IRIS motes were used in our experiments. We found significant ground-bounce effect on the path loss of radio signal strength. In this work localization was implemented and discussed using plain 'iterative majorization' and a slightly modified 'distributed iterative majorization'.

Acknowledgements

I am very much indebted to my Thesis Supervisor, Dr. Joseph John, for dedicating his valuable time and advice. I've benefited a lot from his constant encouragement and the discussions I had with him. I express my deep sense of gratitude to him for providing freedom to explore, and patiently rectifying the mistakes that were committed in the process.

I am thankful to my colleagues, friends and wing-mates for their excellent company and support during my work. I would like to mention a special thanks to Ketan Sharma, who had provided company and assisted me in my experiments.

Finally, I express my gratitude to my parents for their support and encouragement.

Vijayender Reddy K.

Contents

List of Figures

Chapter 1

Introduction

The past two decades of information revolution has created a growing demand for automated live information feeds and remote accessibility of all sorts of devices. The proliferation and easy availability of the embedded electronics and radio transceivers have resulted in development of relatively cheap, low-power, multifunctional sensor nodes that are small in size and can communicate over wireless medium. Wireless Sensor Networks (WSN) are notable for their ease of setup and scalability. In addition to these, the combined processing power of all the networks has paved way to research of novel techniques of sensor data acquisition. Instead of sending raw data to the sink, the sensor nodes now have the capability to carry out simple instructions and collaboratively process the data. These features have accrued a wide range of audience and applications for WSNs. Some of the prominent areas of application are environment, health, military and home.

Apart from easy deployment; self-configuring ability and scalability are important factors that shall increase the adoption of WSNs. We are still in the process of finding a ubiquitous solution for self-configuration of sensor networks. Availability of relevant location information of the nodes is one particular aspect, that can boost self-configuration. In many scenarios sensor data has no value without the location information.

The process of finding location information of sensor nodes is termed as localiza-

tion. In the recent years, a considerable amount of research has been done for finding efficient and accurate localization solutions. Unfortunately most of the research has been either analytical study or limited to simulations. Methods suggested suffered from either intensive memory/processing requirements, accuracy, partial implementations or complexity. One could receive the location information from GPS devices, or other external means. In fact the accuracy of these methods is better than the in-network localization. But, as one of the primary motives of Wireless Sensor Networks is to provide cheap ubiquitous solutions, that can be easily adapted. Hence in-network localization is widely researched.

Localization is useful for:

- Routing protocols: Protocols can exploit the location information and provide robustness and reduce latency.
- Query systems: Most of the environmental properties have a predictable gradient. Availability of location information can be exploited along with this property to provide faster and low power consuming (in-network) query processing systems.
- Compression of data: Location information can provide assistance to compression algorithms of data aggregation, which conserve energy by reducing communication link usage.
- Tracking and logistics: Though the accuracy of in-network localization techniques is low, they still are an option for cheap and simple solution for tracking and logistics.

1.1 Objective of thesis

Many techniques have been provided for localization in WSNs. They can be broadly classified to lateration based and multidimensional scaling (MDS) techniques. Lateration techniques were the initial research in this field. They were known to suffer

from lack of adaptability to errors and complexity in implementation. Though MDS was a well established field, it has been less researched as it was considered to be computationally intensive.

The objective of this thesis is to compare relevant MDS techniques in the context of localization for WSNs. Based on the study of MDS techniques, a solution for localization in WSNs is to be implemented for a Wireless Sensor Network.

1.2 Thesis organisation

Chapter 2 provides a review of the Wireless Sensor Networks. Related hardware and software are discussed. Applications of localization and related work in this field are discussed.

Chapter 3 introduces the basics of localization for WSNs. Relevant MDS techniques are introduced, and are compared. Through the studies in this chapter it is found that iterative majorization outperforms the other studied techniques.

Chapter 4 provides details of work done for range estimation and implementation of iterative majorization for a Wireless Sensor Network.

Chapter 5 further simplifies the iterative majorization method and provides a distributed algorithm for localization. Details of performance of the distributed algorithm are discussed therein.

Chapter 6 concludes the thesis and gives suggestions for future work in this area.

Chapter 2

Review of Wireless Sensor Networks

Wireless Sensor Networks consist of spatially distributed autonomous sensor nodes, that collaboratively perform tasks like monitoring physical and environmental conditions. Each of these sensor nodes is referred as a "mote", and is capable of performing some processing. Typically these devices are battery driven, and are used for unattended monitoring for a duration that might range from a few months to a couple of years.

Motes are cost efficient economical devices which can be used in large numbers. The power of Wireless Sensor Networks is in the ability to deploy large number of tiny nodes, that work cooperatively. In the scenarios involving deploying devices in large numbers, self configuring ability is important. These networks are expected to work without manual management or configuration. In many scenarios the protocols are designed to facilitate scattering the motes randomly in an ad hoc manner[2, 3].

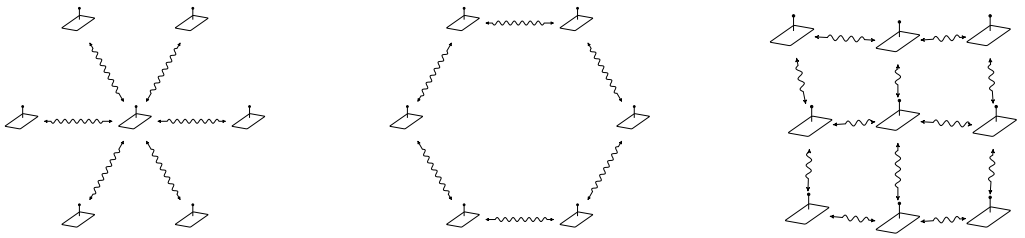


Figure 2.1: Schematics of star, ring and mesh grid protocols

Unlike traditional wireless devices, motes do not communicate to a central high power node. Typically communication is only among the peers. Through these peer-to-peer communications they form a mesh network, where information can be passed between far away nodes through multiple hops. There are a plethora of routing protocols for communicating between the nodes. They vary from the traditional star, ring to mesh grid protocols (See Fig. 2.1). The protocols of ad hoc networks use one of the following topologies: flat network topology [4], hierarchical network topology [5] and hierarchical cluster topologies [6, 7, 2].

Mobile Ad Hoc Networks (MANETS) are close relatives of WSNs as they deal with similar network structure. MANETs as such are used in situations where the nodes/terminals are much more powerful, such as PDA, laptops etc. WSNs have to scale to much larger numbers (perhaps hundreds or thousands) of entities than current ad hoc networks for MANETs. One particular problem that WSNs share with MANETs is self reorganisation of networks without much human intervention. Some of the concepts regarding self reorganising networks are discussed in [8]. Ad Hoc On-Demand Distance Vector (AODV) routing protocol is a popular protocol used by mobile nodes in ad hoc networks [9].

There is extensive research in the development of new algorithms for data aggregation [10], ad-hoc routing [11, 9], and distributed signal processing in the context of Wireless Sensor Networks.

2.1 Components of a Wireless Sensor Network

2.1.1 Hardware overview

This section discusses the hardware components and the composition of a single node of a Wireless Sensor Network, a 'mote'. A functional diagram of a mote is provided in the Fig. 2.2. Components used in a mote are typically cost of the shelf equipments to facilitate low price availability of the technology. Motes have to meet the requirements that come from the specific applications they are deployed for,

for e.g., they might have to be cheap, have small form factor, adequate memory resources, energy requirements, appropriate sensors on-board etc.

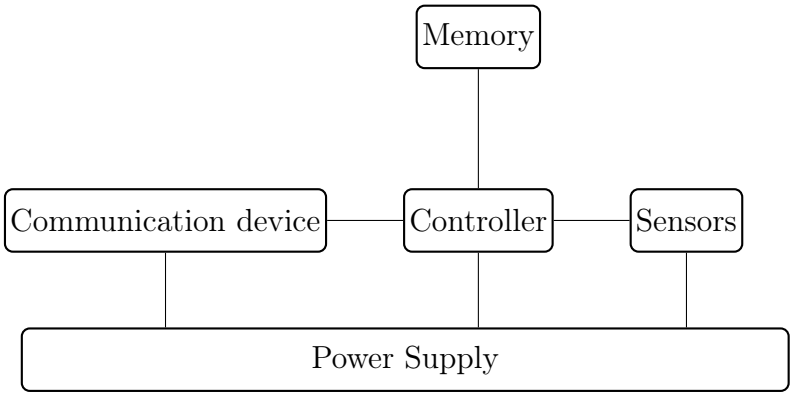


Figure 2.2: Overview of mote hardware components

Controller

The controller is the core of a mote. It collects data from the sensors, processes the data, decides when and where to send it, and receives data from the other nodes. It might have to do time critical signal processing. A significant part of the network protocol and routing information are done by the controller.

Keeping in mind the cost and ease of construction, microcontrollers, which are widely used in embedded systems are the common choice for Controllers. A few popular microcontrollers are listed below:

Intel StrongARM is in WSN terms a fairly high-end processor and mostly used when intensive computation is required as in signal processing scenarios.

Texas MSP 430 (16bit RISC) is a class of nodes specifically designed for embedded applications provided by Texas Instruments.

ATmega 128L (8-bit) is another popular choice of microcontrollers for WSNs provided by Atmel. The Crossbow Technology Inc. provides a range of motes derived from MICA series (MICA2, MICAZ, IRIS etc.) which use ATmega 128L microcontroller. MICA is a platform Wireless Sensor Networks developed by University of California, Berkley.

Memory

RAM for program execution and ROM (more exactly EEPROM), for storage of program code are typically provided by the microcontrollers. However external serial flash devices, e.g., AT25Dxxx, are used to retain data over power losses. The energy consumed for a single write to flash device is significant hence they are used efficiently and primarily for data logging purposes. Also the long read and write delays of flash devices prevent them from being used as a secondary RAM device.

Communication device

Communication device is used to exchange data between individual nodes. Wireless communication consumes more energy than wired communication, for similar distances. Hence, often, the buses on the microcontroller are multiplexed for a number of nearby sensors whenever feasible.

For a practical RF-based system, the carrier frequency has to be carefully chosen. Except for Ultrawideband technologies most of today's RF-based systems work at frequencies below 6 GHz[12]. Keeping in mind the low cost of the sensor nodes, these devices have to operate in ISM bands which require no licensing. Thus the popular choice of frequency band for WSNs is the band specified by IEEE 802.15.4. [13]. IEEE 802.15.4 operates on one of three possible unlicensed frequency bands:

- 868.0-868.6 MHz: Europe, allows one communication channel.
- 902-928 MHz: North America, up to thirty.
- 2400-2483.5 MHz: worldwide use, up to sixteen channels.

Transceiver is the device that performs the radio communication in a mote. A few of the popular IEEE 802.15.4 compliant transceivers that are available in the market are CC1000, CC2420, RF230.

The tasks and characteristics of a transceiver are [12]:

- Service to upper layers (most notably to the Medium Access Control layer): The service either packet oriented or byte level interface as requested by the controller. It also provides an interface for configuring the PHY parameters.
- Power consumption and energy efficiency: The communication unit must provide idle (Listening for packets) and sleep (Low power with significant parts turned off) power states, which can be utilised to conserve energy.
- Modulation: Modulation of the data which is further put into the medium by means of an antenna.
- Gain control and channel selection: Interface to vary the signal strength of outbound messages and to select one of the available channels for both inbound and outbound messages are provided.
- Carrier sense and RSSI measurements.
- Some of the chips also provide CRC checking for the received packets.

Sensors

A few sensors commonly used in Wireless Sensor Networks for environmental sensing or humidity, temperature, ambient light etc. The above examples fall under the category of passive omnidirectional sensors. Typically sensors are interfaced to the controller using the ADCs of microcontroller. Some sensors that have advanced configuration controls are connected to the microcontroller through bus protocols like SPI, USART or I2C.

Power supply

Traditional batteries are used for the power supply in most of the commercial motes. The life of a pair of AAA batteries (of 1000mAh each) in a mote ranges from 6 months to a few years, based on the frequency of usage. Sending data over the wireless medium requires a considerable amount of energy compared to the idle

state energy consumption, hence judicious use of the communication link must be made. Compression and decompression might require lower amount of energy than the amount required to transfer the uncompressed stream of data. Methods avoiding use of battery are discussed in [14], which derive energy from e.g., solar cells. These techniques can make the sensors near perpetual devices. As per their estimation these devices could run unattended for about 45 years.

2.1.2 Software

Each mote runs an instance of an Operating System (OS) specially designed for the task it is deployed. The traditional tasks of an OS are controlling and protecting the access to resources and manging their allocation to different users as well as the support for concurrent execution of several processes and communication between these processes[15]. Only some of these tasks are implemented in an OS for sensor nodes.

Support for concurrent execution is crucial for motes, which have to perform time certain critical duties. Most of the programs in a traditional OS are designed to work sequentially, and the processes are switched from time to time. But the microcontrollers used for sensor nodes, do not have the required resources to support a full-blown operating system, or an efficient process scheduling and context switching mechanism. Moreover, in sequential logic, the system will have to frequently switch to processing of incoming packets. In the absence of concurrent execution, a traditional sequential model would run into the risk of missing data while a packet is processed or missing a packet when sensor information is processed, especially if either of the operations are time consuming.

The two techniques that address concurrency are process-based concurrency and event-based programming. In an event-driven system the main program logic is divided into small tasks and events. It must be noted that similar to this event-driven logic, the WSNs operate on a reactive basis. Changes in environment create events, and events create other events.

TinyOS

TinyOS is a free and open source component-based operating system and a platform targeting Wireless Sensor Networks. In the current work, TinyOS-2 has been used. It has been chosen as it is most actively researched and developed in the academia, among the operating systems for WSNs. Its main advantages are its speed and its specific structure that serves the purpose of the sensor networks. Same piece of code can be deployed onto various hardware, in fact on different microcontrollers, with little or no modifications. TinyOS is built ensuring that extending support to new systems and architectures will be an easy task. By virtue of this key feature most hardware of Wireless Sensor Network is supported in TinyOS.

The operating system TinyOS[16] along with the programming language nesC(network embedded system C) [17] address the challenges of implementing an event-driven low latency operating system for sensor networks. The event-driven model might be alien to most programmers and requires getting used to, which might consume time. But it does provides considerable advantages [18]. It was observed that on a same hardware, using TinyOS, performance improved by a factor of 8, instruction/data memory by 2 and 30, respectively, and power consumption was reduced by a factor of 12, when compared to its general purpose counterpart eCOS[19].

TinyOS supports modularity and even-based programming by the concept of interfaces, modules and components. The code related to each piece of hardware, and each functionality are divided into components. Components communicate by way of commands and events. For example to get the temperature from a temperature sensor, following are the steps:

- The main program logic component calls a command `temperature.read()`
- The component `temperature` issues the necessary commands to speak with the sensor hardware.
- Until the hardware responds, the main program can perform other events and tasks such as receiving available packets.

- When the hardware responds an event `temperature.readDone(error_t result, val_t val)` defined in the component of main program, is called.

Such an approach is termed as **split-phase programming**

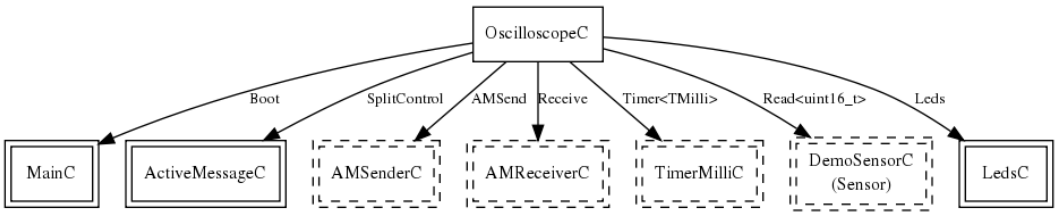


Figure 2.3: Interconnect of the various components in OscilloscopeAppC

Fig. 2.3 shows the interconnect of the various components in a sample application. Only the higher level components are shown. Each driver that talks with the hardware is further divided into components belonging to three tiers of abstraction [20].

Commands and events are used for triggering duties. In particular commands must not block or wait for an indeterminate amount of time. The actual computational work is done in tasks. Let us consider, a computation involves a large number of iterations, which might require considerable amount of time to run to a conclusion. These iterations are done using tasks. Typically not more than few iterations are to be accomplished in a single task. It is a good practise to schedule the task again and again until the iteration runs into a conclusion, rather than waiting for all the iterations to yield in a single task. This allows for running other time critical duties in between tasks, that have been triggered by events in the meanwhile.

2.2 Application of localization for Wireless Sensor Networks

Wireless Sensor Networks can enable information gathering, information processing and monitoring of environments for a variety of applications. There are different kinds of applications in which sensor networks perform a wide range of activities.

Of them certain applications require that sensor networks collectively form an ad hoc distributed processing network. In these applications the networks must self-configure and self-organise. Such unattended sensor networks are becoming increasingly popular in a large number of applications [21, 22, 23, 24].

Many signal processing tasks in a sensor networks assume the availability of location information (For e.g. see [24]). Location information ascertains the space time relation of the sensor data, which can be further used in meaningful aggregation of data. One could develop querying systems similar to TinyDB[25] (used for query processing in a network built with TinyOS) and query based on both spatial and temporal information of the monitored environment.

In most situations, sensor information is useful only along with spatial information. Accurate information of sensor location is often not available. Sensor nodes are either carefully placed by hand or scattered. In the former case location information can be assumed or obtained easily, since the nodes are static. Often in the former case, nodes are used in small numbers. But in the later case, which is often the case when people need large number of sensors, manually locating each sensor node is tedious and impossible in many situations (e.g., airborne scattering). One could employ each sensor node with a GPS to obtain location but this adds to the expense, power consumption and form factor, let alone the fact that GPS devices might fail under a roof(e.g., cellar, tunnel etc.). Both these techniques viz., manual setting of location information and GPS location setting are not applicable to localization of large scale Wireless Sensor Networks. Thus there is a growing interest in developing reliable self-localization techniques for sensor networks.

Self-localization in Wireless Sensor Network is an active area of current research. Location information is used to find spatial origin of the sensor readings which is used in tracking [26, 27]. This information is also used in routing protocols based on geographical distribution [28, 29]. Location information is also used in techniques of storage of sensor data [30, 31].

2.2.1 Related work

Many localization algorithms have been proposed to provide location information in Wireless Sensor Networks. These techniques can be broadly classified into two kinds[32]: range-free schemes and range-based schemes. Range-free schemes do not require any assisting technology or equipment to measure the distance between nodes, they just apply the communication among nodes to localise unknown nodes. The representative schemes are centroid [33], Approximate Point in Triangle(APIT) [34] and DV-Hop [35]. But they provide only coarse locations. In our work, we describe range estimation techniques and range-based localization schemes. A few of the range-free localization techniques can be considered a subset of range-based techniques. In these techniques distance between each pair of connectible nodes is set to a predefined value, and this data is further processed to derive location information[36]. Range free localization schemes often presume a certain degree of sparsity of the network.

Range based techniques process the distance information, among the nodes, to provide us with possible coordinates of the nodes. There are a variety of techniques for obtaining the distance information. Common techniques for distance or angle estimation include Time of Arrival (ToA), Time Difference of Arrival (TDoA), Angle of Arrival (AoA), and Received Signal Strength indicator (RSSI)[37]. Each of these are metrics, available when two nodes communicate, from which one can estimate the distance in between. These techniques have their own caveats and advantages. Some provide accuracy, while some have longer range, and some have special hardware requirements.

Range based techniques can be again broadly classified to beacon based and beacon-less techniques. In the beacon based techniques, a few nodes are powerful enough to send strong beacons. Techniques that employ beacons have been documented in [33, 38]. In beacon-less techniques, nodes estimate their inter-nodal distance by discovering their neighbours and interacting with them.

Processing the inter-nodal distance to coordinates is done by a variety of tech-

niques. Initial research in localization for Wireless Sensor Networks involved using multilateration [32, 39, 33, 40], triangulation[41]. The other set of popular localization techniques fall under Multidimensional Scaling (MDS). MDS is a set of techniques which have developed long before the advent WSNs. Their primary goal was to provide visual maps of data based on the similarity, dissimilarity or distance information among the points. There haven't been any notable implementations or on-field trials in localization using MDS techniques. Some of the important research using MDS in WSNs' are [36, 42, 43]. MDS is a name given to a large group of techniques. Most of the techniques either suffer from requirement of centralised computation or intensive computation requirements. Hence most of the research of MDS in WSNs have remained as simulation works far from implementation. Distributed weighted-multidimensional scaling (dwMDS)[42] is the most impressive and complete technique among the MDS techniques and elegant enough to work for a real network.

Chapter 3

Node Localization in Sensor Networks

3.1 Basics of localization

Localization is the process of finding the location information of nodes of a sensor network. This location information can be found using services like GPS, or using self-localization techniques. Self-localization techniques solve for the location of the coordinates using information available within the sensor network. An important set of self-localization techniques translate the set of distances between the nodes, to coordinates.

3.1.1 Methods of acquiring distances

The inter-nodal distances can be obtained from signal strength, time of arrival, time difference of arrival etc. The following are the link level metrics available which can be used to estimate the distance between two nodes.

Received Signal Strength Indicator (RSSI): RSSI is a measurement of the power present in a received radio signal. In the Wireless Sensor Networks most of the transceivers compute this metric while receiving messages and is stored in a register. This metric can be read by the software in mote, when

required.

Time-of-Arrival (ToA): ToA is the travel time of a radio signal from a single transmitter to a remote receiver.

Time difference of Arrival (TDoA): TDoA is the difference of arrival time of signal transmitted from multiple synchronised transmitters.

Among the above methods, time based ToA and TDoA provide most accurate estimates of the distances between the nodes. But these methods require a high frequency clock and precision measurements, if they intend to use times from propagation of EM waves. The other alternative is using ultrasonic sounders and corresponding receivers. These provide better accuracy with low resolution clocks, but require additional hardware and are prone to suffer due to environmental noise. Also these devices have much lesser range than an EM wave.

Apart from the distance based methods, there is triangulation which uses angle-of-arrival (AoA) from the direction of propagation of radio signals to find the co-ordinates. It has the drawback of requiring more than one receivers at a single node to find the angle of arrival. Thus this method is not in usage. RSSI metric is readily available in almost all hardware of Wireless Sensor Networks. Hence, in spite of their poor accuracy, compared to time based metrics, they are widely used for range estimation. Thus RSSI is ubiquitous for localization in low cost sensor network solutions.

3.1.2 Methods of localization

Number of methods are available for translating the inter-nodal distances to co-ordinates. The popular traditional methods are triangulation, tri-lateration and multilateration which solve for the coordinates by finding intersections and solving appropriate equations [32]. The performance of these algorithms is deteriorated by range estimation errors and inaccurate distance measures, which are common in a wireless sensor networks. Attempts were made to improve the approach by itera-

tively computing. However the method added large number of communication costs, and still could not generate good position estimation in some cases [44].

Multidimensional Scaling (MDS)

MDS is a set of statistical techniques often used in information visualisation for exploring similarities or dissimilarities in data. It has its origin in psychometrics. It was initially proposed to help understand people's judgements of the similarity of members of a set of objects. As of now MDS is used in diverse fields such as sociology, physics, political science [45].

Techniques of MDS provide a visual map of a set of objects, on providing the similarity,dissimilarity or the distances between the objects. Classical MDS was initially introduced by Torgerson [46]. He provided mathematical calculation which translated a set of distances between objects to their coordinates in n-dimensions. Classical MDS requires one of the following extensive computational procedures: Singular Value Decomposition or Eigen Decomposition. This led to development of computationally simpler and efficient algorithms like Newton-Raphson, majorization, simulated annealing, which work on the principle of finding the solution that minimises a given loss function [47].

Given a set of n points which are to be localised, the following are a few notations

1. Matrix of pair-wise distances among the n points

$$\Delta = \begin{pmatrix} 0 & \delta_{12}^2 & \dots & \delta_{1n}^2 \\ \delta_{21}^2 & 0 & \dots & \delta_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1}^2 & \delta_{n2}^2 & \dots & 0 \end{pmatrix}$$

where δ_{ij} is the distance between i^{th} and j^{th} points.

2. $n \times 2$ matrix of estimated coordinates of the n points.

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{pmatrix}$$

3. Matrix of pair-wise distances among the n estimated coordinates

$$D = \begin{pmatrix} 0 & d_{12}^2 & \dots & d_{1n}^2 \\ d_{21}^2 & 0 & \dots & d_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1}^2 & d_{n2}^2 & \dots & 0 \end{pmatrix}$$

where d_{ij} is the distance between the coordinates of i^{th} and j^{th} points.

Loss function: Loss function (also referred as stress function) expresses the quality of approximation of a given configuration $\vec{X}$.

For a given configuration $\vec{X}$ the approximation error in representing nodes i and j is given by $e_{ij} \stackrel{\text{def}}{=} \left| d_{ij}(\vec{X}) - \delta_{ij} \right|$.

The most basic form of stress is Raw Stress. It is given by:

$$\sigma_r(\vec{X}) \stackrel{\text{def}}{=} \sum_{i<j} e_{ij}^2 = \sum_{i<j} \left(d_{ij}(\vec{X}) - \delta_{ij} \right)^2 \tag{3.1}$$

Normalised stress: For the same accuracy, raw stress varies with number of nodes. The following is the equation of normalised stress, which has no units, and can be used to compare the accuracy of different solutions with different number of nodes.

$$\sigma_n(\vec{X}) \stackrel{\text{def}}{=} \sum_{i<j} e_{ij}^2 = \frac{\sum_{i<j} \left(d_{ij}(\vec{X}) - \delta_{ij} \right)^2}{\sum_{i<j} \delta_{ij}^2} \tag{3.2}$$

Loss function based techniques require minimisation algorithms. Configuration

$\vec{X}$ that provides the minimum loss is the optimal configuration. These techniques start by assuming an initial value of X, and iteratively process on it, until an optimum minimum loss close to 0 is obtained.

3.2 Overview of algorithms studied

Of the available minimisation techniques for minimising the loss function, we shall study Simplex algorithm [48], simulated annealing [48] and iterative majorization [50].

3.2.1 Simplex algorithm

Most of the minimisation algorithms require the computation of derivatives. Nelder and Mead [51] developed downhill simplex algorithm for minimisation, which can be used to minimise a function without the evaluation of derivatives. It is used in scenarios where computation of derivatives might be computationally intensive or resource inefficient, for example the loss function.

A simplex is the geometrical figure consisting, in N dimensions, of N +1 points (or vertexes) and all their interconnecting line segments, polygonal faces, etc. In two dimensions, a simplex is a triangle. In three dimensions it is a tetrahedron.

A huge simplex is defined. In each iteration of the algorithm, the function to be minimised is evaluated at each vertex of the simplex. The downhill simplex method now takes a series of steps, most steps just moving the point of the simplex where the function is largest ("highest point") through the opposite face of the simplex to a lower point. These steps are called reflections. Based on certain other such heuristics on the values of function at the vertexes, the simplex is shifted, shrunk, stretched or expanded. At the end of the iterations the minimum is contained in the simplex. The details of the algorithm can be found in [48].

3.2.2 Simulated annealing

Simulated annealing (SA) is a generic probabilistic meta-heuristic for the global optimisation problem, for finding a good approximation to the global minimum of a given function in a large search space.

The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one.

In this method, each point of the search space is analogous to a state of some physical system, and the function f to be minimised is analogous to the internal energy of the system in that state. The goal is to bring the system, from an arbitrary initial state, to a state with the minimum possible energy.

Procedure for minimising a function f: This method iteratively solves for the minimum.

- Start with a high temperature and decrease the temperature.
- At each value of temperature certain number of iterations of the following is carried out
  1. A random step $X_1 = X + \Delta X$ is made, where ΔX is a random step with size taken proportional to temperature.
  2. find $f(X_1)$
  3. if $f(X_1) < f(X)$: Accept the new configuration X_1.
  4. if $f(X_1) > f(X)$: Accept the new configuration by comparing $p(\delta E) = \exp\left(-\frac{\delta E}{kT}\right)$ with a uniform random variable, where $\delta E = f(X_1) - f(X)$.

Initially the function f takes big steps proportional to the temperature (step 1),

searching for a minimum. It accepts configurations, if they are of lower energy than current state (step 3). The method allows for a few upwards climbs of function $f(X)$ (step 4), to avoid local minimums if any.

3.2.3 Iterative majorization

Iterative majorization is an elegant algorithm for computing an MDS solution. Unlike the previous methods which are generic minimisation (which can be used on any particular function without much modification), a given majorization technique can be used only to minimise the function for which it has been designed. Rather randomly searching for the minimum, this methods exploits the analytical properties of the function to be minimised.

The central idea of majorization method is to replace iteratively the original complicated function $f(x)$ by an auxiliary function $g(x, z)$, where z is some fixed value. The function g has to meet the following set of requirements for it to be accepted as a majorizing function.

- $g(x, z)$ should be simpler to minimise compared to $f(x)$.
- $f(x) \leq g(x, z)$
- $f(z) = g(z, z)$

On finding such a majorizing function, $g(x)$, for a given $f(x)$, the following iterative procedure will result in a minimum of the function $f(x)$:

- Let $x^{(i)}$ be the known minimum of function in i^{th} iteration.
- $x^{(i)}$ is considered as the new fixed point z, and
- $x^{(i+1)} = \text{minimum}\,(g(x, z))$ is computed. It must be observed that $x^{(i+1)}$ satisfies $f(x^{(i+1)}) \leq f(x^{(i)})$.

One of the main features of IM is that it generates a monotonically non-increasing sequence of function values. Thus each iteration guarantees a better estimation of the minimum.

The loss function given by raw stress (equation 3.2) can also be rearranged to

$$\sigma_r(\vec{(}X)) = \eta_\delta^2 + \eta^2(\vec{X}) - 2\rho(\vec{(}X)) \tag{3.3}$$

where $\eta_\delta^2 = \sum_{i<j} \delta_{ij}^2$, $\eta^2(\vec{X}) = \sum_{i<j} d_{ij}^2(\vec{X})$ and $\rho(\vec{X}) = \sum_{i<j} \delta_{ij} d_{ij}(\vec{X})$.

The majorizing function for the loss function given by equation 3.4, is

$$\tau(X, Z) = \eta_\delta^2 + \text{trace}(X'VZ) - 2 \times \text{trace}(X'B(Z)Z) \tag{3.4}$$

which has the minimum

$$X^u = V^+ B(Z)Z \tag{3.5}$$

where V^+ is the Moore-Penrose pseudoinverse [52] of V. Pseudoinverse is the generalisation of inverse of a matrix.

$B(Z)$ is given by $n \times n$ matrix $[b_{ij}]$:

$$\begin{aligned} b_{ij} &= \begin{cases} -\frac{\delta_{ij}}{d_{ij}(Z)} & \text{for } i \neq j \text{ and } d_{ij}(Z) \neq 0 \\ 0 & \text{for } i \neq j \text{ and } d_{ij}(Z) = 0 \end{cases} \\ b_{ii} &= -\sum_{j=1, j\neq i}^{n} b_{ij} \end{aligned}$$

The above method requires computation of Moore-Penrose inverse, which requires considerable amount of computation. Substituting all the weights by 1 simplifies X^u and leaves us with the simple equation

$$X^u = n^{-1} B(Z)Z \tag{3.6}$$

Hence to localise a matrix of distances $\Delta_{n\times n}$ and provide a map X the following are the steps

1. Initialise $X_{n\times 2}$ with random values

2. Update X with X^u where

$$X^u = n^{-1}B(X)X \tag{3.7}$$

3. Calculate the normal stress σ_n (equation 3.1). If $\sigma_n >$ threshold jump to step 2, else we have arrived at a minimum.

The above proofs of majorizing function and its minimum, are available in [50].

3.3 Comparison of different MDS techniques

To study the performance of the above said MDS techniques and to find the technique that provides the best and fast results, a test framework has been made in C language. Simplex algorithm and simulated annealing had certain configuration variables, viz. max-step size, temperature etc., that can be varied. This simulation has been carried out to understand the effect of these variables on the performance of the techniques, and to find the set of variables that provide best performance results. GNU Scientific Library (GSL)[53] was used for implementing the minimisation techniques. C has been chosen, as porting the algorithms to nesC (to run on hardware) from C is an easy task.

The running time of each technique are logged for test cases of varying number of nodes and also for different parameters of each technique studied. To find out the exact timing information `getrusage()` available in `<sys/resource.h>` has been used which provides the most accurate estimate of system usage by a given process, excluding the time spent on all other processes, that might have run in parallel in the computer. The usage statistics found by using `getrusage()` do not have any absolute significance. For the same amount of usage they may vary on different machines. However these metrics can be compared if all the simulation runs have been carried out on the same computer. Considering this precaution, all the simulations have been carried out on a single computer. System usage is directly proportional

to the time consumed by the technique for a given test case.

Each test case of the simulation is as follows:

- Generate a random map.
- Find the pair-wise distance between all pairs of nodes.
- Compute wsn-like distances (explained in the next section) based on the pair-wise distances.
- Input these wsn-like pair-wise distances, along with the configuration variables to the test-suite.
- The test-suite runs the MDS technique and provides us with the following output which are recorded:
    1. System usage
    2. Final loss to which the algorithm converged. If the final loss is less than 0.1, we can safely assume that the technique has converged to a solution.

It must be noted that, in the simulations carried out, all nodes were considered connectible. i.e., distance between every pair of nodes is available.

3.3.1 WSN-like distances

In order for the simulation to present us with the performance of the various techniques in real scenarios, the input distances to the MDS techniques should closely model the distances inferred from signal strength between motes in a real setting. We shall call distance inferred by two motes based up on the signal strength between them as *wsn-like* distance. Patwari [54] provided a model for the signal strength path loss. According to [54], the signal strength $P_d(\text{dBm})$ of a received signal at a distance of d from mote is given by

$$\begin{aligned} P_d(\text{dBm}) &\sim \mathcal{N}(\bar{P}(\text{dBm}), \sigma^2) \\ \bar{P}_d(\text{dBm}) &= P_0(\text{dBm}) - 10n_p \log_{10}(d/d_0) \end{aligned} \tag{3.8}$$

where $\mathcal{N}$ is normal distribution, σ and n_p are to be determined by conducting experiments. P_0 is the signal strength at a known distance of d_0. Typically n_p is 2 in free space. Patwari has carried out experiments and obtained RSSI for various distances. The results of these experimental data have been provide at [1]. We have used this data to find wsn-like distances discussed above.

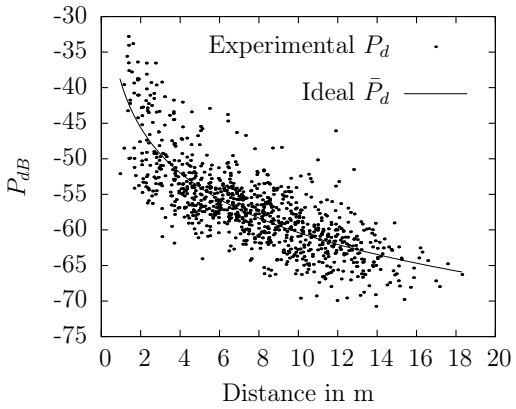


Figure 3.1: Readings provided by Patwari [1]

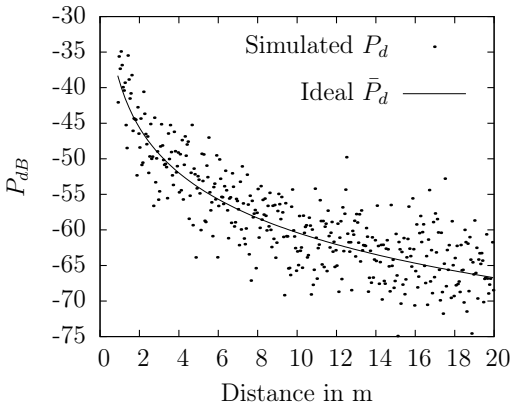


Figure 3.2: Simulated readings

Distance d (inferred using signal strength) between two motes δ distance apart can be derived from equation 3.8. d is given by

$$d = \delta * 10^{\frac{\mathcal{N}(0,\sigma)}{n_p}} \tag{3.9}$$

By fitting the data with equation 3.9, we found that data provided in [1] has $\sigma \simeq 4$ and $n_p \simeq 2.11$. We used these values for the simulations.

d from equation 3.9 is the wsn-like distance between two motes δ apart. Using this we generated WSN-like distances between nodes and compared with the experimental data. Fig. 3.1, shows us experimental signal strengths. Fig. 3.2 shows us the signal strengths generated using wsn-like distances. These are the distances that have been used to compare the MDS techniques studied in our work.

3.3.2 Performance analysis of simplex algorithm

Simplex algorithm has been discussed in section 3.2.1. This technique has been assessed by running the simulation for a number of test cases with different configuration variables. The following are the configuration variables of simplex algorithm:

- Maximum step size, per iteration
- Number of nodes.

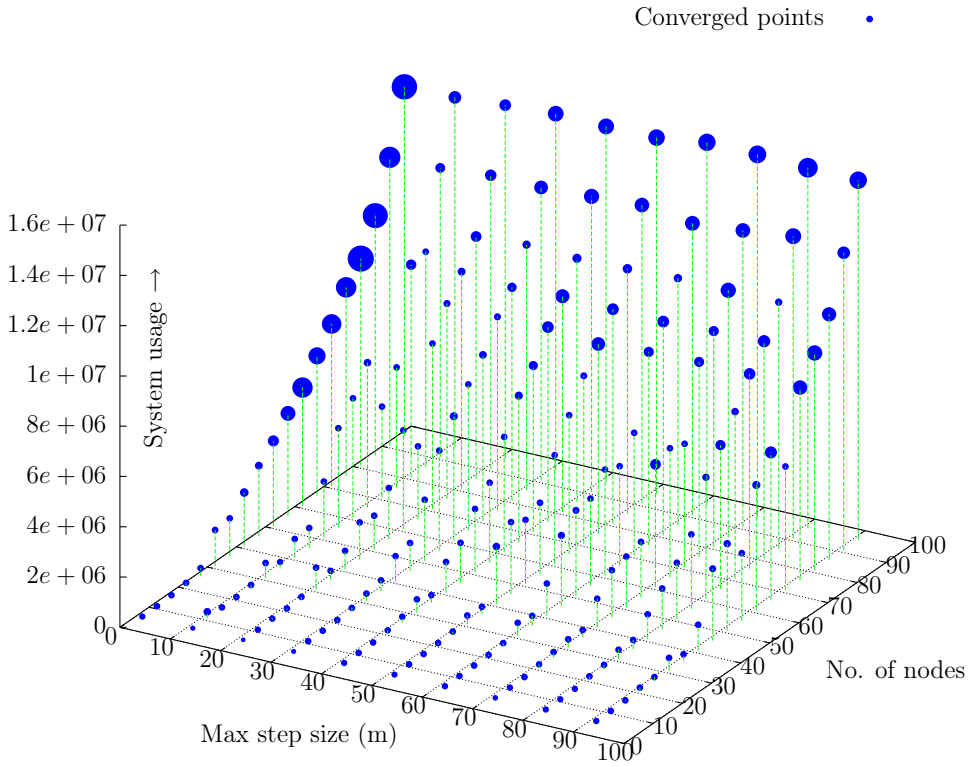


Figure 3.3: Performance analysis of simplex algorithm using wsn-like distances

The results of the simulation are shown in Fig. 3.3. It must be noted that, in the figure, each point corresponds to a simulation run, with the *size of point* proportional to the final loss attained at the end of all iterations.

From the simulation results we can infer that the technique performs poorly for number of nodes greater than 40. Also, any of the max-step size above or equal to

10 can be used. Not much performance variation is found on changing the max-step size.

3.3.3 Performance analysis of simulated annealing

Implementation details of simulated annealing have been provided in section 3.2.2. The configuration variables of this technique are:

- number of nodes
- max-step size
- damping constant
- minimum temperature

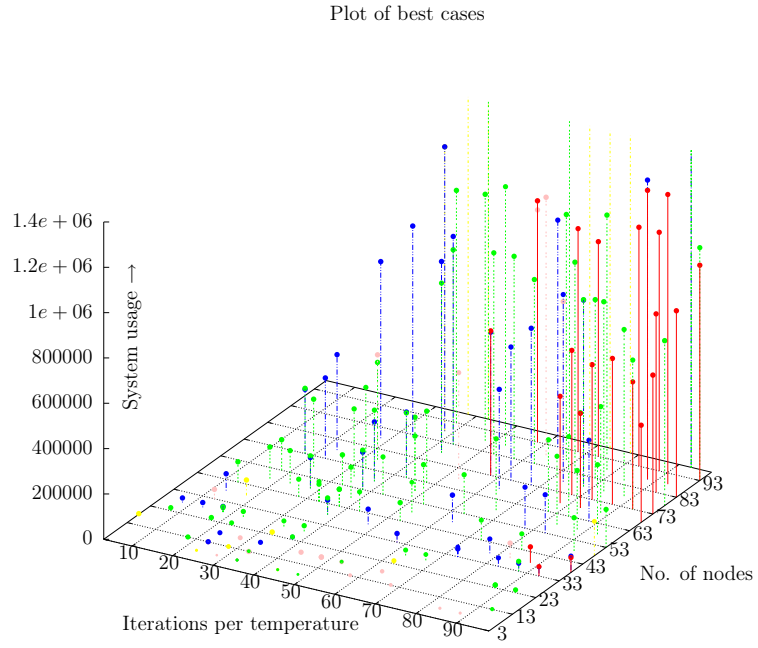


Figure 3.4: System usage plot for simulated annealing

Fig. 3.4 and Fig. 3.5 show only the best converged solutions, which shall be used to understand the configuration variables that provide better results. The corresponding configuration variables where step-sizes are marked on the points in

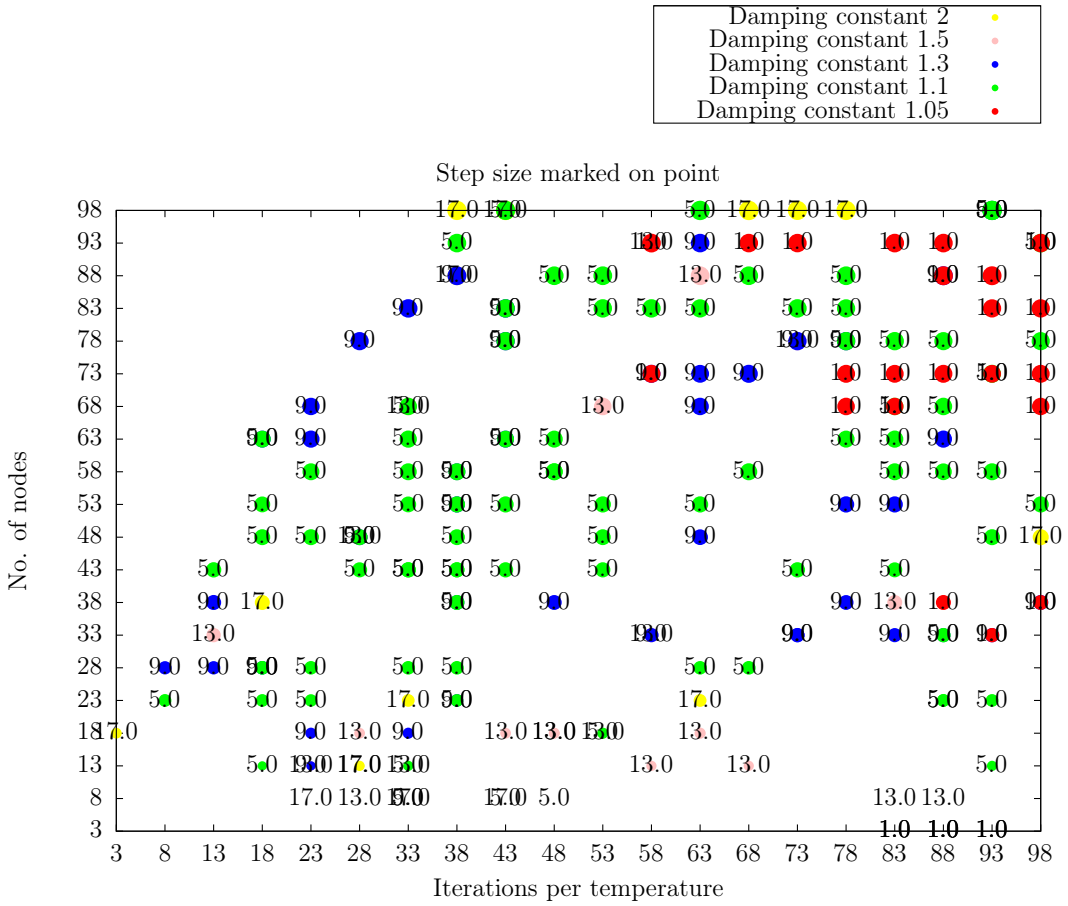


Figure 3.5: Performance analysis of simulated annealing

Fig. 3.5. Fig. 3.4 provides us a 3D view, to help visualise the system usage. It must be noted that these plots contain only those points which have converged. From the plots, we can infer the following:

- A max-step size of 5-9 converges in most of the cases.
- For a damping constant greater than 1.1 performance deteriorates.
- 'Iterations per temperature' must be varied proportional to the number of nodes, for better performance.

We can observe that this technique out performs simplex algorithm.

3.3.4 Performance analysis of Iterative Majorization

There are no configuration variables in Iterative Majorization. Implementation details have been explained in Section 3.2.3.

Fig. 3.6 shows the performance of IM against varying number of nodes. It has been noted in many a literature, that at times, IM might fail by converging to a local minimum[42]. However in our study we found that, out of 666 trials, 662 test cases converged within 100 iterations. 3 test cases took 100-200 iterations to converge. There was only one test case, which could only converge to a loss 0.101 (which is fairly accurate), even after 1000 iterations. Hence in this work we are ruling out the cases where IM might fail to converge.

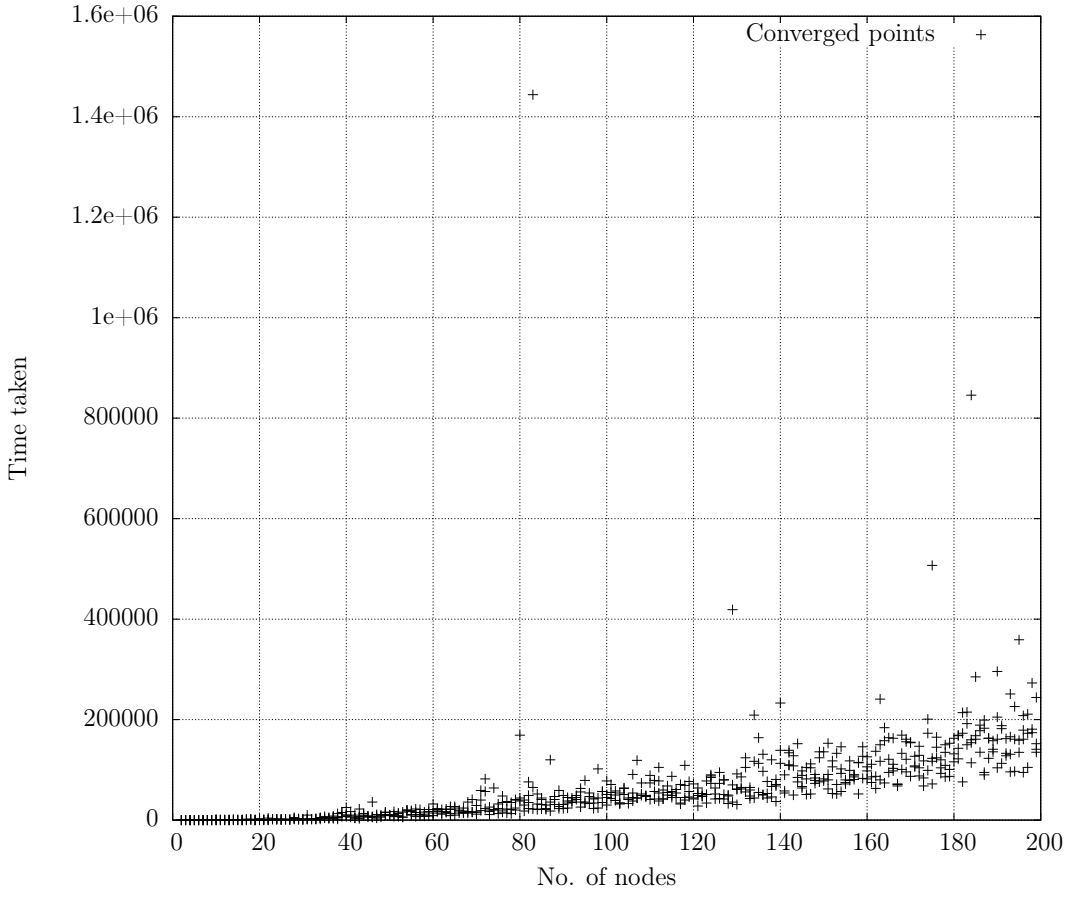


Figure 3.6: Performance analysis of iterative majorization

3.3.5 Comparison

Fig. 3.7 compares the best case solutions (system usage using the configuration variables that give the best and fastest solution for a given number of nodes) for simplex algorithm and simulated annealing. However all solutions of iterative majorization are shown in the plot. We can see that iterative majorization is the clear winner compared to the other two MDS techniques. Hence iterative majorization has been chosen for implementing localization in TinyOS in this work.

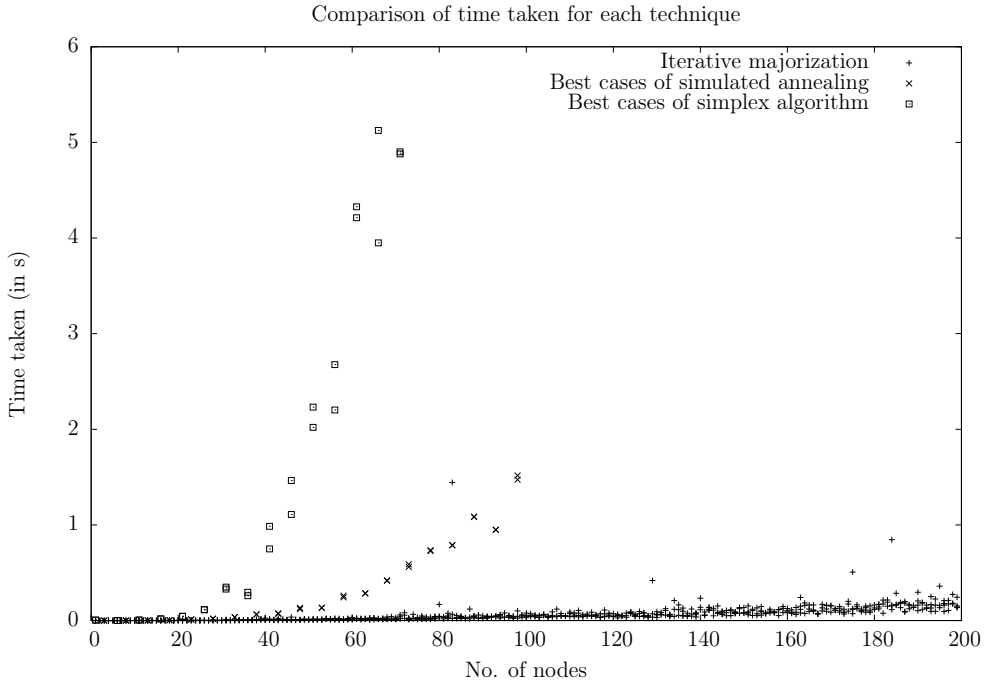


Figure 3.7: Comparison of MDS techniques

Simplex algorithm and simulated annealing were simulated only up to 100 nodes, as the simulation times have increased to a few seconds on computer. The time required to carry a computation of similar order on motes would be inordinate. However the time required for iterative majorization had remained low. Hence simulations have been carried out up to 200 nodes, which can be safely assumed to be the upper limit of number of connectible nodes.

Fig. 3.8 shows us the rate of convergence of each of the techniques on a particular test case involving 44 nodes. Note that the x-axis is logscale. We can see the speed with which each algorithm minimises the loss function.

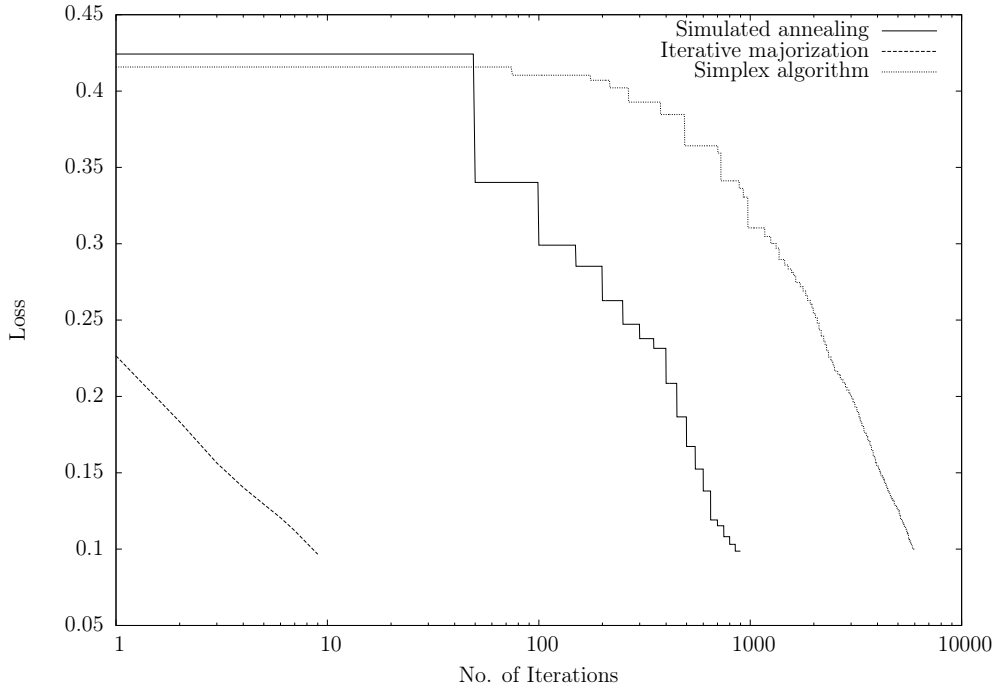


Figure 3.8: Comparison of speed of convergence of MDS techniques for a test case of 44 nodes

Chapter 4

Implementation in motes

In this work, TinyOS has been used to develop and test applications for Wireless Sensor Networks. We have seen that MDS techniques translate the set of inter-nodal distances to corresponding coordinates of the nodes. The following section explores the study of inter-nodal distances in commercial motes.

4.1 Obtaining inter-nodal distances

The distance between a pair of nodes is estimated using certain metrics we have seen in Section 3.1.1. In this work we have studied RSSI, ToA, LQI and ED to understand their variations with distance. Previous research using these metrics have been mostly done on test-bed platforms [1] which have better accuracy and time resolution. Their results help us understand the theoretical significance of these metrics. But they fail to provide us with an idea of performance of the metrics in commercially viable motes, which are in popular use. Hence we have studied metrics and their relation with distances.

4.1.1 Experimental setup

Our experimental setup consists of IRIS motes [55], which have RF230 radio transceiver [56]. These transceivers provide the following metrics after the receipt

of a radio message:

RSSI RSSI register is updated every 2 µs as long as it is receiving a message.

Energy Detection (ED) ED is another metric that reflects the energy in received signal. It is computed by averaging the RSSI metric, over 8 IEEE 802.15.4 symbols (128 µs). The relation between signal strength and ED of IRIS motes, as provided by the datsheet, is

$$\frac{\text{ED}}{84} + \frac{\text{signal\_strength} - 7}{-91} = 1 \tag{4.1}$$

Link Quality Indication (LQI) LQI determines the link quality of the radio link. LQI values are integers from 0 to 255 and can be associated with packet error rate (PER).

ToA ToA reflects the time taken for a message to travel between two motes. This metric is not directly provided by the hardware. Techniques involving usage of the message timestamps have to be used for find ToA.

To obtain ToA metric between motes `m1` and `m2`, clocks on both the motes should be synchronised. Synchronisation of clocks itself is a much researched topic, as it is used in many other scenarios e.g., MAC and routing protocols.

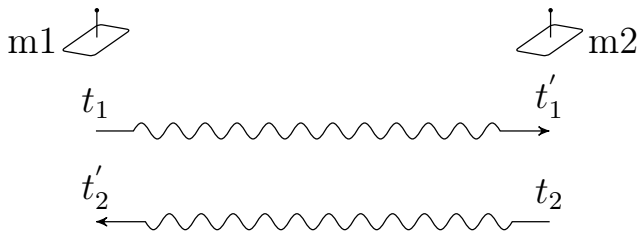


Figure 4.1: Finding ToA without synchronised clocks

In our experiments the following method is used which doesn't require synchronised clocks. Consider Fig. 4.1.

- Transmit a message from `m1`. record the time of transmission t_1 (according to clock of `m1`).

- On receipt of message on `m2`, record the time $t_1^{'}$ (according to clock of `m2`).
- Without much processing delays transmit back message from `m2` to `m1`, record time t_2 (according to clock of `m2`).
- On receipt of message on `m1`, record the time $t_2^{'}$ (according to clock of `m1`).
- ToA is now given by $\frac{(t_1-t_2^{'})-(t_1^{'}-t_2)}{2}$.

Considering the speed of propagation of a radio signal, the following is the chart of minimum possible resolution of distances computed from ToA corresponding to clock resolutions.

Clock frequency	Minimum distance resolution
1 KHz	299.79 km
1 MHz	299.79 m
4 MHz	74.948 m
8 MHz	37.47 m
16 MHz	18.73 m

IRIS motes work at 8MHz, and henceforth only resolutions greater than 37.47m are possible. However in reference [54] high frequency equipment, whose clocks had resolution of nano seconds, have been used and henceforth resolution in the range of a few meters had been possible.

To measure these metrics and study their variation with distance, we have varied distance between two motes, in steps, from 0m to 50m. At each different distance, a code is run which acquires the metrics and stores them in the flash memory. These readings were later transferred to the computer with the help of Base Station mote. While acquiring the metrics, both the motes have been placed at a height of 1m from the ground.

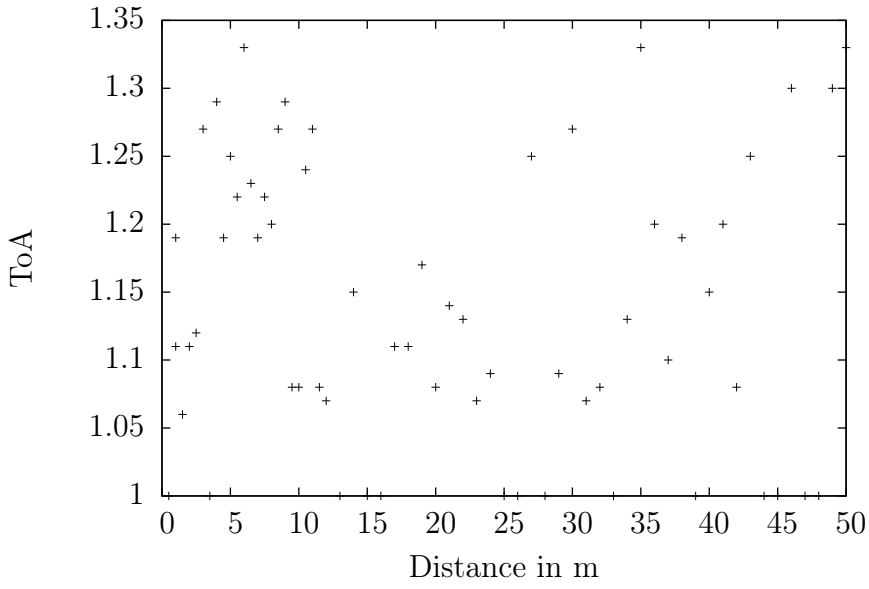


Figure 4.2: ToA vs. distance in open space

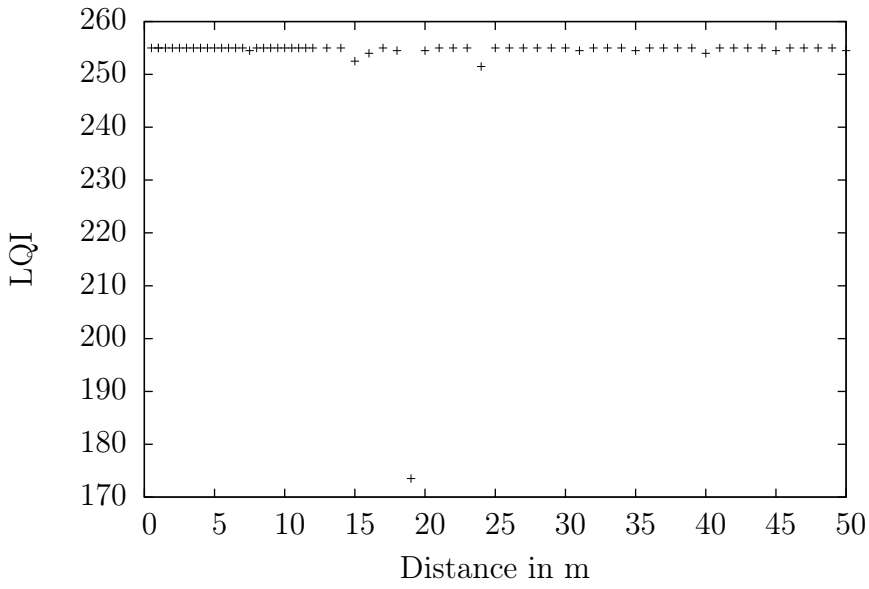


Figure 4.3: LQI vs. distance in open space

Fig. 4.2 shows the variation of ToA with distance. As expected, we are unable to infer any information from ToA, because of the clock resolution of motes.

Fig. 4.3 shows the variation of LQI with distance. LQI is computed from packet error rate (PER). We find that LQI doesn't vary with distance, in a clear line of sight test. Hence it is an insufficient metric to estimate inter-nodal distance..

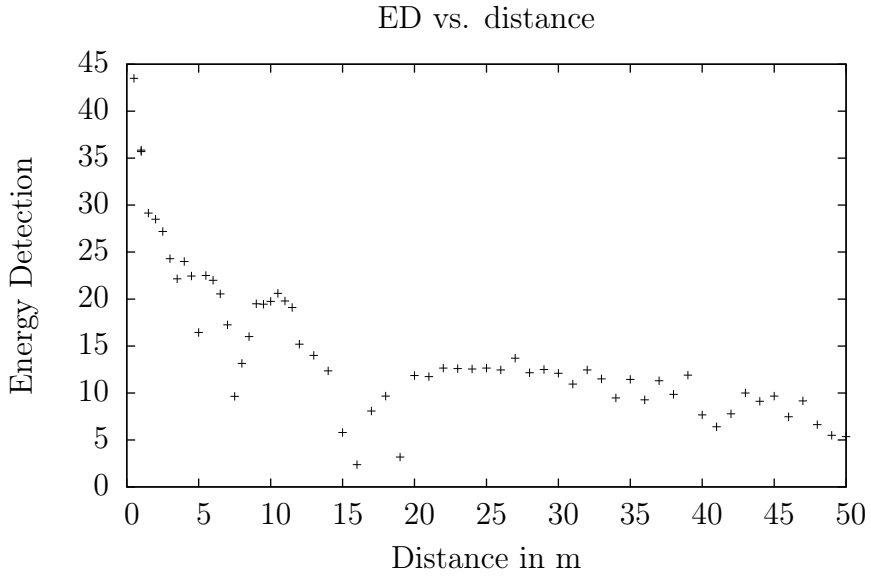


Figure 4.4: ED vs. distance in open space

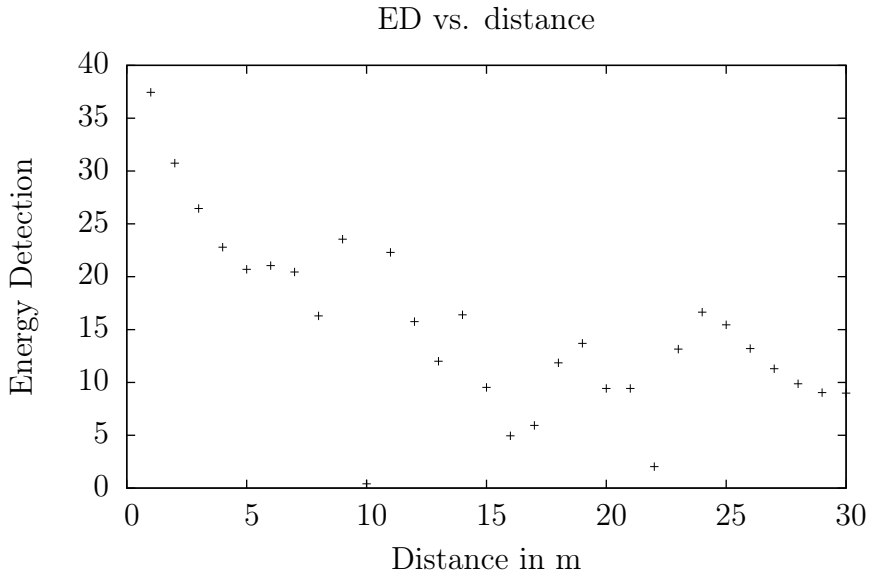


Figure 4.5: ED vs. distance over the corridor roof

Fig. 4.4 and Fig. 4.5 show the variation of ED in an open space and over a corridor, connecting Lecture Hall Complex and Faculty Building at IIT Kanpur. In both the experiments, clear line of sight between motes has been maintained. Corridor had adjacent trees, but the readings show no significant effects of adjacent objects.

4.1.2 RSSI readings

Out of the available metrics on IRIS motes, RSSI follow a pattern. Hence ED (average of RSSI) is the best metric to estimate inter-nodal distances. However we find that signal strength doesn't vary as predicted by 3.8. The following section provides us with a model to understand the ED variations in Fig. 4.4 and Fig. 4.5.

Ground-bounce path loss

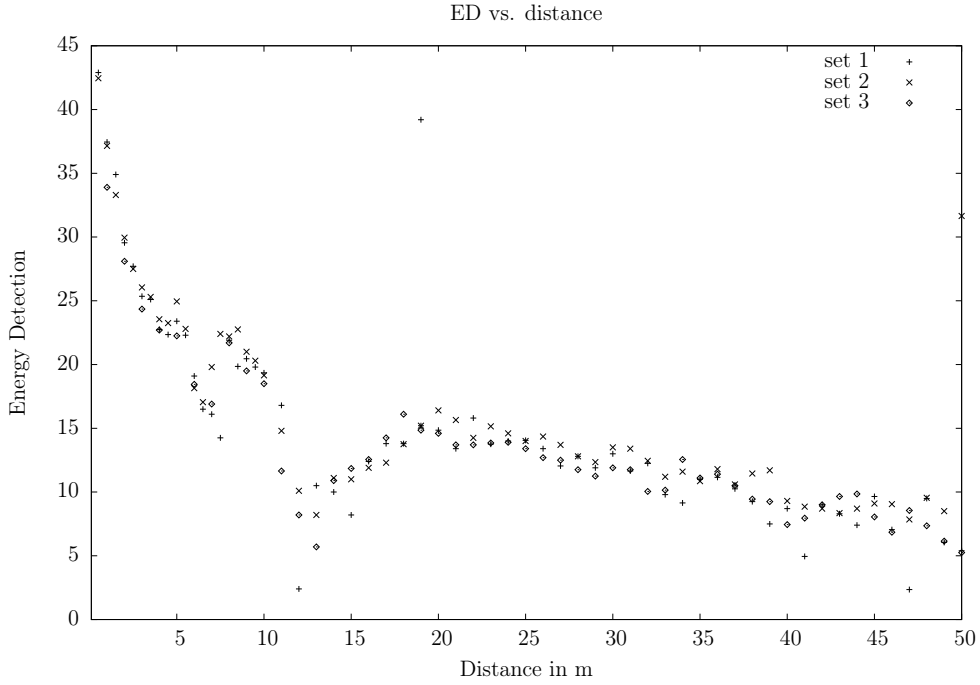


Figure 4.6: ED vs. distance in open space for 3 different sets showing the ground bounce effect

Fig. 4.6 shows three different readings sets, taken at completely different locations. These plots show us a path-loss that is different from simple log-distance model of equation 3.8. The undulations seen are not environmental noise. They have taken the same pattern in all the acquired sets of readings. These undulations are caused because of interference between the line-of-sight radio wave, and the radio wave reflected by the ground.

The antennae used in IRIS motes, are omnidirectional. So a signal of almost equal intensity reflects at the earth surface, and reaches the receiver. A model to find the

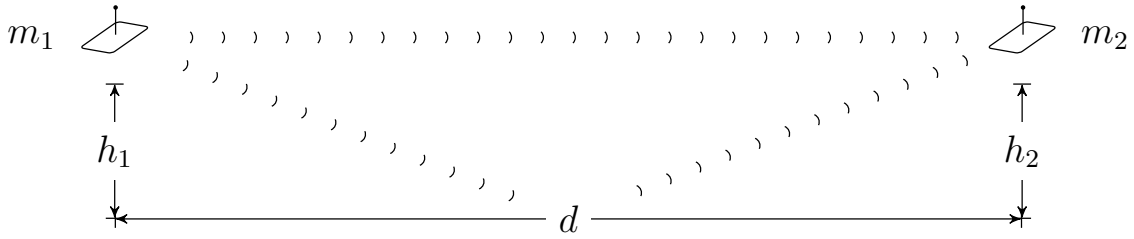


Figure 4.7: Illustration of ground-bounce between two motes

resultant signal intensity after the interference with reflected signal is provided in [57]. The following equation 4.2 models the path loss after ground-bounce where two motes are placed at heights h_1 and h_2 as shown in Fig. 4.7:

$$\begin{aligned} \triangle d &\approx 2h_2h_1/d \\ P_d &\propto \left(\frac{\lambda}{4\pi d^2}\right)^2 \sin^2\left(\frac{\pi \triangle d}{\lambda}\right) \end{aligned} \tag{4.2}$$

In our experiment both h_1, h_2 are set to 1m. λ for the radio channel of 2.4GHz is 12.49cm.

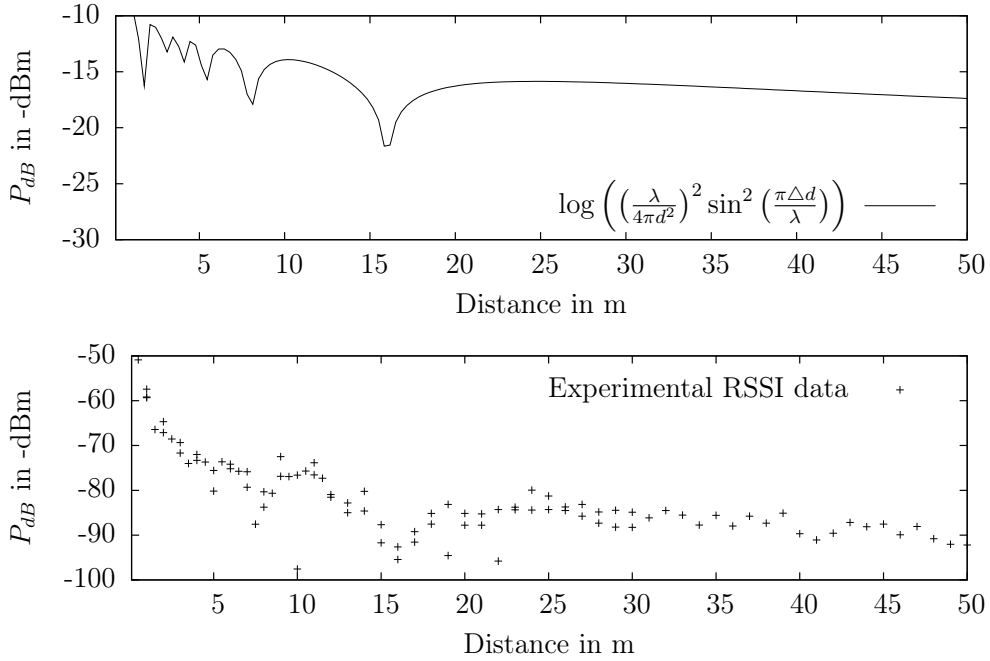


Figure 4.8: Verifying ground-bounce path loss model

Fig. 4.8 verifies the ground bounce path-loss model, which is commonly ignored in discussions of WSN localizations. Most of the simulations of WSNs use direct

line-of-sight free space path-loss as input to the simulations.

Estimating distance from ED

The mote can receive only integral values of ED varying from 0 to 84. In practice the value of ED never crossed ≈ 54, for mote to mote communication.

To estimate the inter-nodal distance from the ED value found after the receipt of a message from a mote, we have created a table of ED vs. distance. The mote looks up the table and finds the distance corresponding to a given ED. To enumerate the table we have fitted the experimentally found ED vs. distance data onto the following fifth order monotonically decreasing function:

$$\text{ED} = -\left(1.3 \times 10^{-6}\right) d^5 + 0.00021\, d^4 - 0.013\, d^3 + 0.37\, d^2 - 5.1\, d + 40 \qquad (4.3)$$

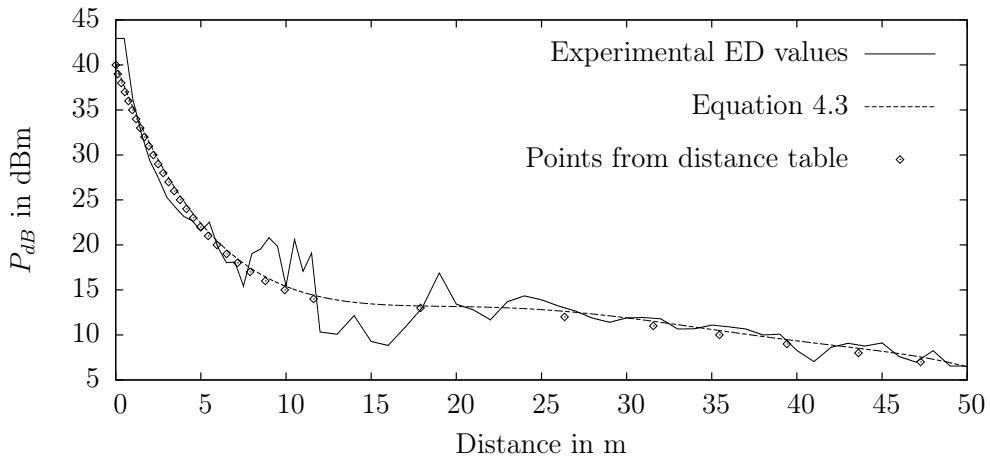


Figure 4.9: Points of distance table for estimation of distance from ED

The 5^{th} order equation 4.3 and the reconstruction table are plotted in Fig. 4.9. The error in estimating distances on using this model are shown in Fig. 4.10. This error is inevitable, because ED vs. distance is non monotonic. There are more than one possible distances for a given ED. Observe in Fig. 4.9 that a section from 12m to 18m, has estimation errors as high as 25m. Such large errors could be catastrophic for localization techniques. In this work we provide no solution for correcting this error. But we speculate that if the nodes are mobile, Kalman filtering could decrease

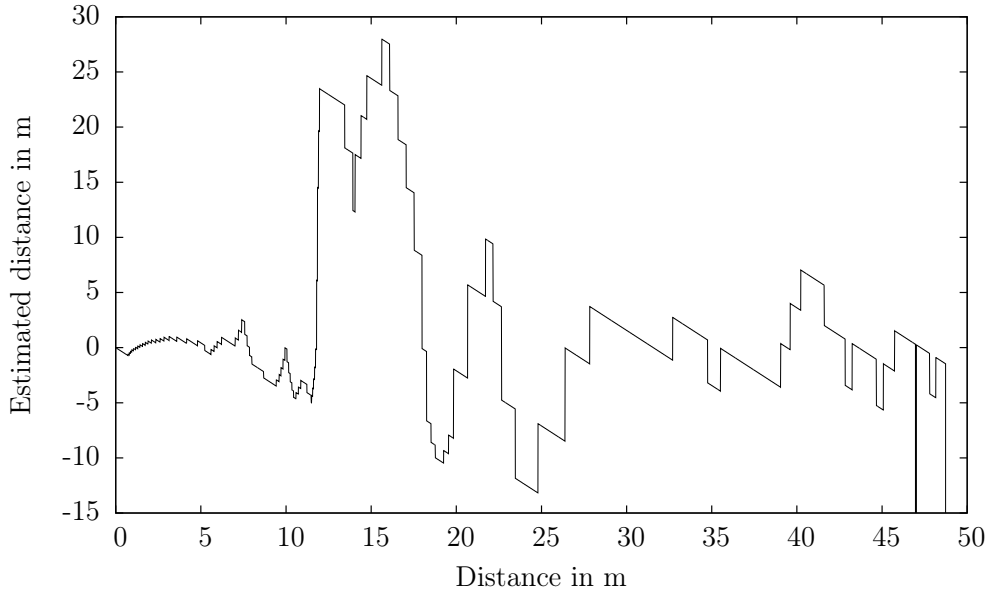


Figure 4.10: Errors in estimating distance from ED using distance table

the errors.

4.2 Asynchronous neighbour discovery

The first step of localization of a given network of nodes is to identify the neighbouring nodes. Since this is the first step after the motes are turned on, we assume no particular protocol of the nodes. We use a bare-bones version of neighbour discovery, primarily based on random amount of back-off.

Detection of the neighbours is done by each mote broadcasting its presence. Since this is the first step of communication after setting up the network, there is no particular order in which motes should take turns to broadcast their presence.

In this version we assume that the entire network is connectible. This is required as the IM can't deal with holes in the D matrix. A group leader is chosen for the entire network, which will do the main processing of localization.

The steps involved in choosing the group leader and discovering the neighbours are as follows:

1. Each mote sets a timer that fires off after a random interval. (say 0-16ms)

2. On firing off of the timer a `discovery` packet is broadcast by the mote.

3. On receipt of every `discovery` packet, the timer if hasn't fired yet, is restarted and set to a different random value. Also the `node_id`, address, and the ED values are stored in a table.

4. The mote in which the timer fires off first, becomes the group leader.

5. If there are no more `discovery` packets in the medium, for a `TIMEOUT`, say for a 120ms period, the neighbourhood discovery is considered complete.

4.3 Iterative majorization

A small library that provides some of the functionality of GSL was created. This enabled usage of the library code mentioned in Section 3.3, which was built using GSL.

After the completion of neighbourhood discovery, the group leader will enter the state of `syndication`. In this state, it will query and collect the neighbourhood information (`node_id` and ED values) from each mote present in its neighbourhood table.

Once the `syndication` is complete, the group leader allocates the necessary memory to carry out IM. The ED values are translated to distance estimates and stored in a matrix D. A task `im_iterate()` is posted which runs one iteration of the IM technique. At the end of the task, it posts itself recursively, until `im_test()` returns `True`.

Function `im_test()` is satisfied, if the difference in losses of the map X before and after iteration is lesser than a threshold.

Fig. 4.11 shows us a flow chart of the implementation of iterative majorization. It must be noted that, in this algorithm the amount of memory required at the group leader mote is $\mathcal{O}(n^2)$.

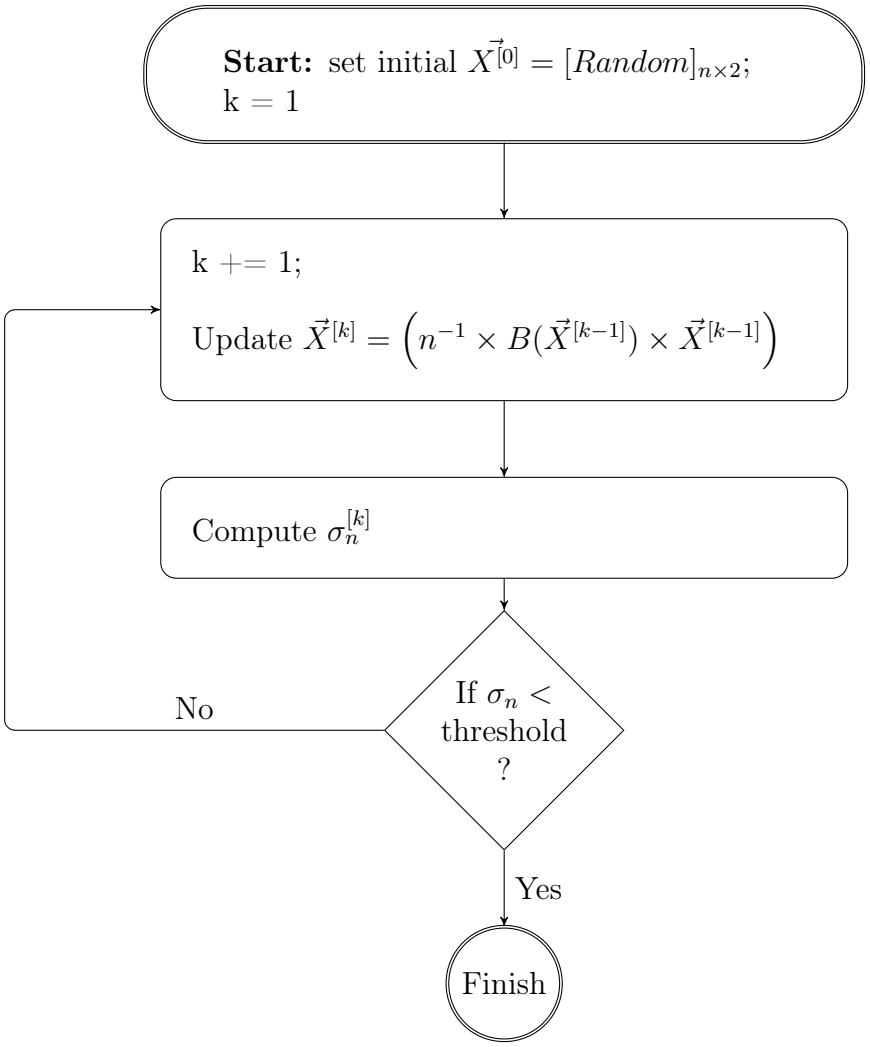


Figure 4.11: Flow chart of iterative majorization

4.4 Testing

All the code for iterative majorization has been written in nesC. Most of the code has been written using tasks which run for a small duration, so as to let the motes be responsive for sensor data aggregation and other processing tasks. We tested the code, by arranging 4 motes in easily recognisable patterns like 'L', square, triangle etc., and obtained satisfactory results. We had a physical limitation of 5 motes, in order to test the code of IM. For robust testing of the algorithm we had to test it on $\approx$ 10 motes. Hence we have used the TOSSIM (TOS simulator)[58] to see the performance and verify the working of the code. TOSSIM is the TinyOS

simulator. It simulates entire TinyOS applications. It works by replacing a few of the components with simulation implementations. It is built to seamlessly simulate 1000s of nodes simultaneously.

To run a simulation, TOSSIM requires a helper code that steps through the simulation, monitors variables and injects packets into the system. The helper code has to initialise the motes that are to be simulated and add the radio links between the motes. TOSSIM also accepts the signal strength and a noise model for each of the links. The helper code can be programmed in either C++ or Python. Python has been chosen in this work.

We have developed a Python helper code, which accepts a set of coordinates corresponding to the nodes that are to be simulated. A radio link is established between every two motes which are within a range of 50m. IRIS motes have a range of $\approx$ 100m, but since the experimental measurements were limited to 50m, we are using 50m as the limit of radio connection. In practise one can decrease the transmit power of the motes so that the maximum range of the motes is limited to 50m. While establishing the radio links, the signal strength between a pair of motes, d distance apart, is set to the value obtained from the experiment for d. In the readings we obtained, for clear line of sight (Fig. 4.8), the noise was little compared to the deviations caused in path-loss due to ground-bounce effect. Hence no noise was added to the radio links.

The simulation computes a matrix Δ of inter-nodal distances.

$$\Delta = \begin{pmatrix} 0 & \delta_{12}^2 & \dots & \delta_{1n}^2 \\ \delta_{21}^2 & 0 & \dots & \delta_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1}^2 & \delta_{n2}^2 & \dots & 0 \end{pmatrix}$$

These distance are converted to a matrix of signal strengths of radio links and fed into the simulator. After the neighbour discover, the group leader estimate a matrix $\hat{\Delta}$ from the signal strengths.

$$\hat{\Delta} = \begin{pmatrix} 0 & \hat{\delta}_{12}^2 & \dots & \hat{\delta}_{1n}^2 \\ \hat{\delta}_{21}^2 & 0 & \dots & \hat{\delta}_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\delta}_{n1}^2 & \hat{\delta}_{n2}^2 & \dots & 0 \end{pmatrix}$$

The relation between $\hat{D}$ and D have been shown in Fig. 4.10.

The results of a few test cases are show in Fig. 4.12, along with the final loss to which they converged. It must be noted that a mote is aware only of $\hat{\Delta}$, because of which the loss reported by mote is $\sigma_n(\vec{X}, \hat{\Delta})$ (equation 4.4). However $\sigma(\vec{X}, \Delta)$ shows us how close to the original coordinates the results have approached. Hence both the losses ($\sigma(\vec{X}, \hat{\Delta})$ as $\hat{\sigma}$, $\sigma(\vec{X}, \Delta)$ as σ) are reported.

$$\sigma(\vec{X}, \Delta) \stackrel{\text{def}}{=} \frac{\sum_{i<j} e_{ij}^2 = \sum_{i<j} \left(d_{ij}(\vec{X}) - \delta_{ij} \right)^2}{\sum_{i<j} \delta_{ij}^2} \tag{4.4}$$

Fig. 4.13 shows the simulation results for the test cases which were used in Fig. 4.12, but using ideal distances. Wsn-like distances, which model the effect of ground-bounce path loss, significantly reduced the accuracy. To attain better results, it is necessary to research measures which can counter the effects of known and predictable phenomenon like ground-bounce.

No. of points, $\sigma_n(\vec{X}, \Delta)$, $\sigma_n(\vec{X}, \hat{\Delta})$ mentioned at the top of each image. Both axes of all the plots are distance in m. The plots show vectors from actual coordinates to estimated coordinates obtained by the MDS technique.

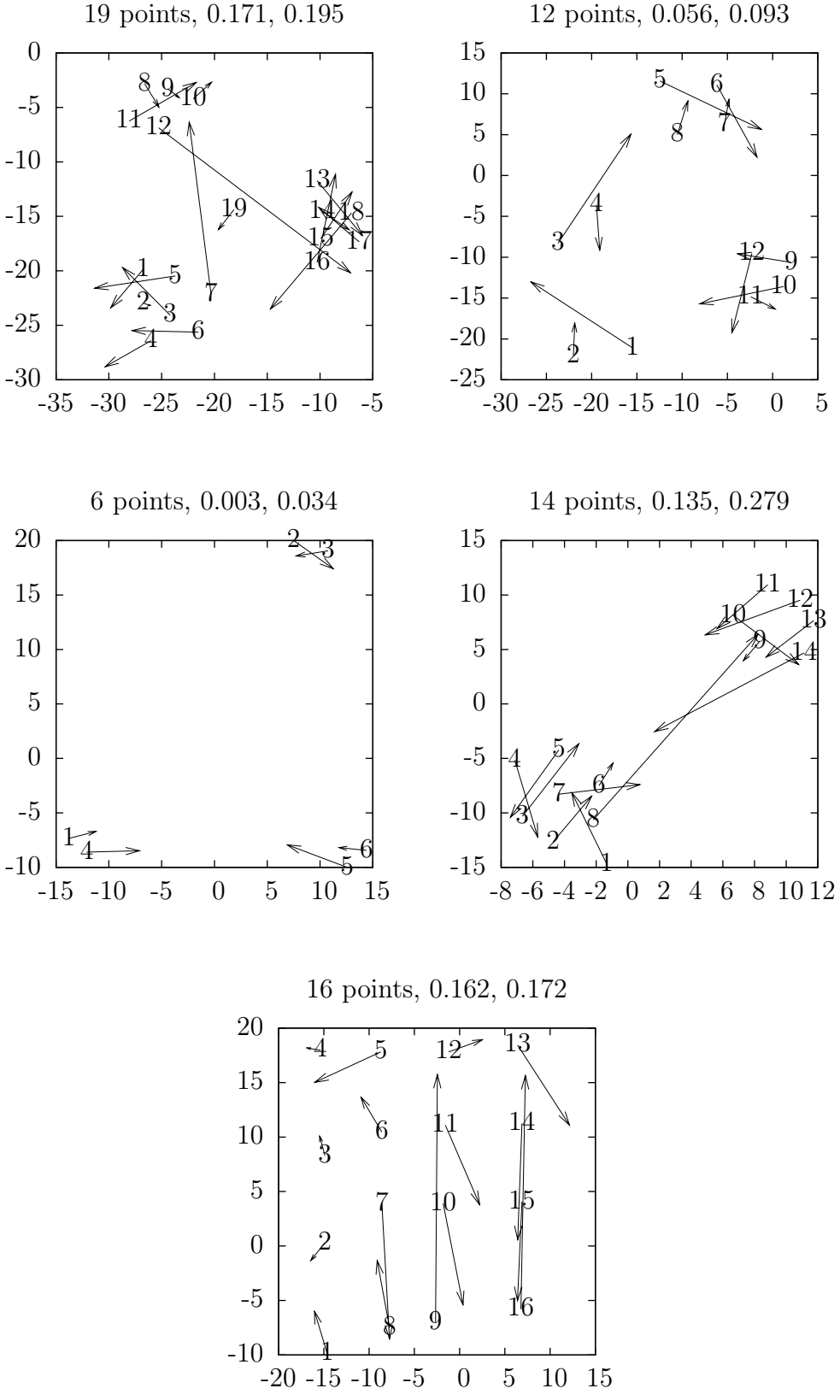


Figure 4.12: Results of a few test cases on simulation with IM using wsn-like distances

No. of points, $\sigma_n(\vec{X}, \Delta)$, $\sigma_n(\vec{X}, \hat{\Delta})$ mentioned at the top of each image. Both axes of all the plots are distance in m. The plots show vectors from actual coordinates to estimated coordinates obtained by the MDS technique. Note that most of the vectors are hidden under the coordinate labels, as the results found are accurate.

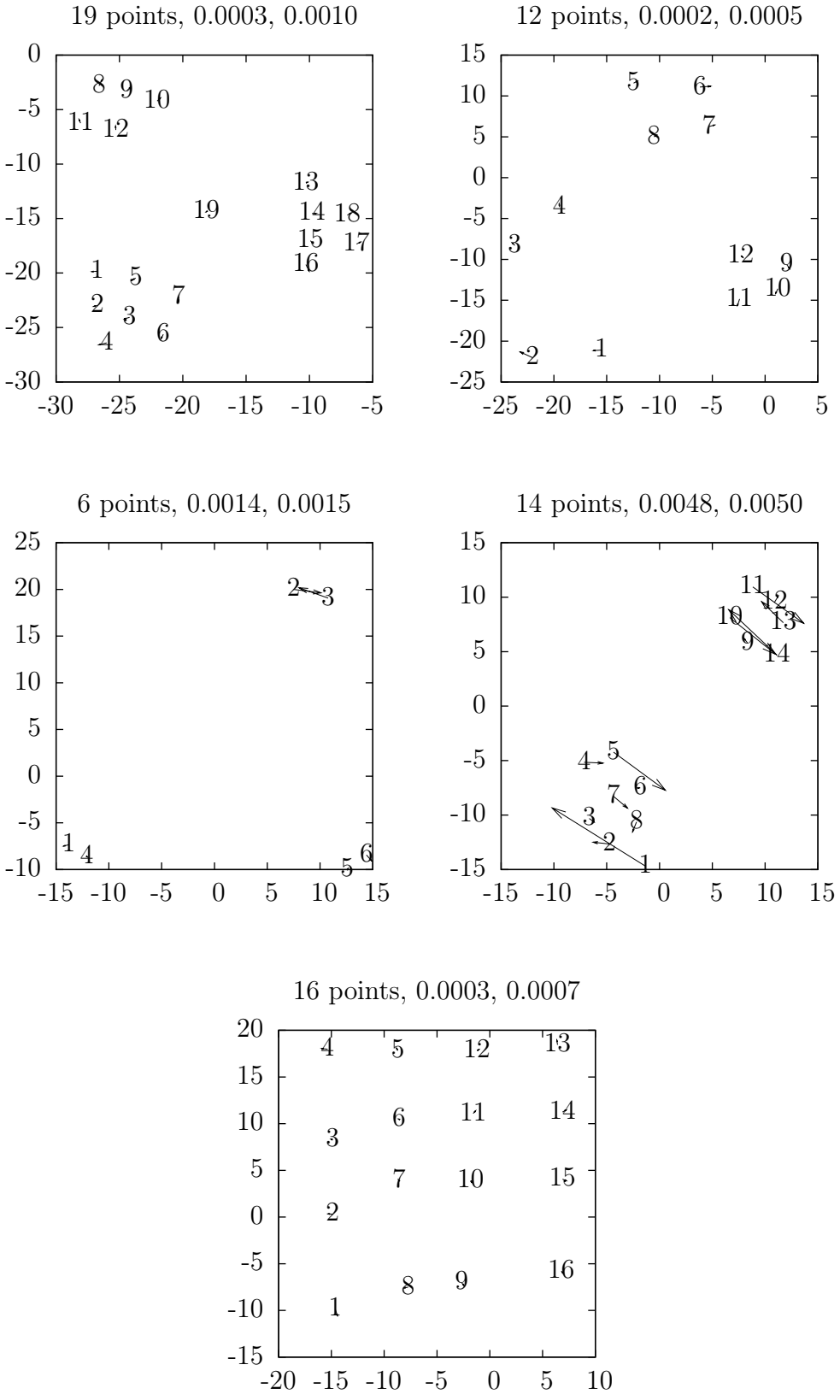


Figure 4.13: Results of a few test cases on simulation with IM using ideal distances

The loss at the end of each iteration for a test case of 9 motes using TOSSIM is shown in Fig. 4.14. We have observed that wsn-like distances take longer than ideal distances to converge. Wsn-like distances are different from ideal distances in the sense that, these distances do not absolutely satisfy as the distance matrix of any coordinate system. This is due to the errors in range estimation. However the ideal distances used can be fitted into a coordinate system, with a small error.

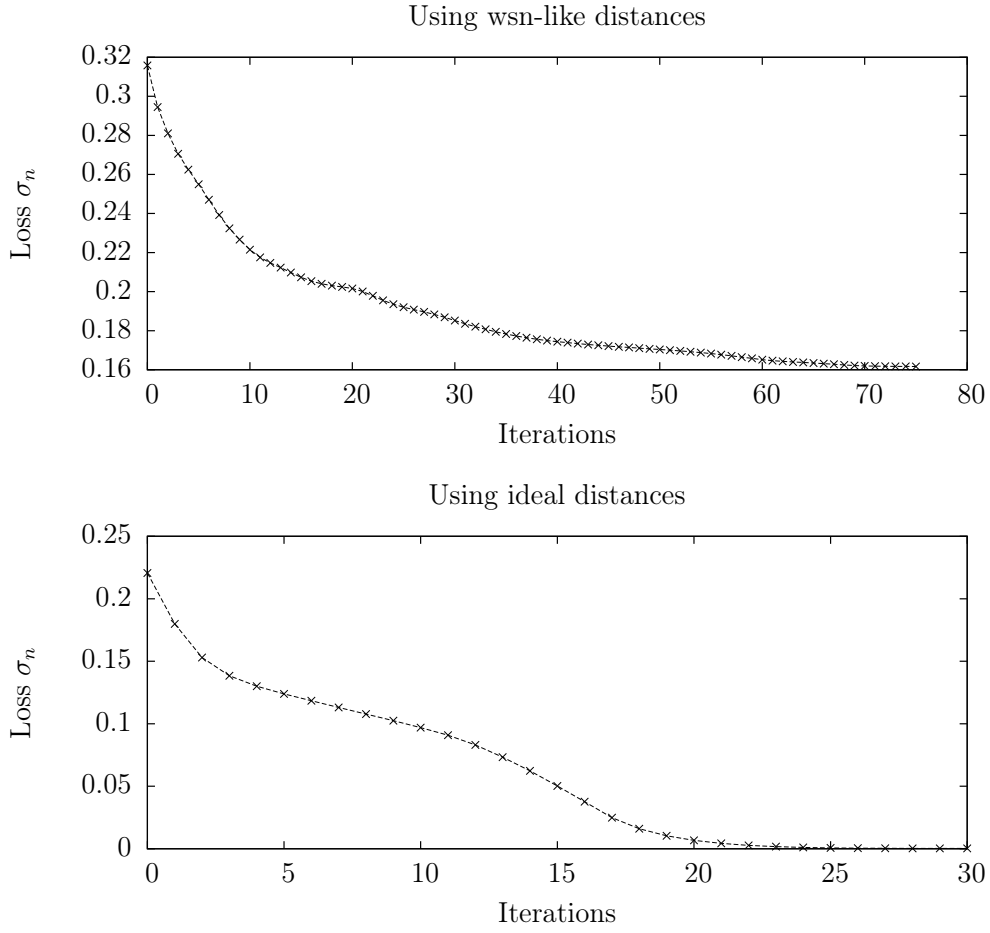


Figure 4.14: Loss after each iteration for a test case of 9 motes

Chapter 5

Distributed localization

The majorization technique implemented in the previous chapter suffers from a few drawbacks. Given a network of motes to localise, each mote should be connectible to every other mote in the network. It is difficult to establish this condition. Consider the case where numerous, say 200, nodes are placed in a small area such that every two nodes are connectible. The previous technique will require 4K bytes of memory to store the D matrix. The memory requirement is $\mathcal{O}(n^2)$, which will soon exhaust the memory on embedded systems.

One workaround is to split the network into small connectible regions. These small localised regions can be stitched together to a final global map. To stitch two maps, the maps must be overlapping with a minimum of 3 common nodes (present in both maps). One has to find the relative shift, rotation and flip(reflection) of both the maps with the help of the common nodes and adjust the local maps. This is true only when the normalised stress of the maps is low. The number of common nodes required in the case of noisy readings will be significantly greater than 3. The degree of overlapping, for correct stitching, is dependent on the accuracy of locating the nodes.

Consider a map $X_{n\times 2}$ which is to be localised from a given distance matrix D. We have seen that each iteration of IM involves improving the estimate X to X^u. Consider (x_i, y_i), the coordinates of i^{th} node. We simplified equation 3.7 and the

corresponding coordinates of X^u are

$$(x_i^u, y_i^u) = \left(\frac{\sum_{j \neq i} \left(- x_j \frac{d_{ij}}{\hat{d}_{ij}} \right) + x_i \sum_{j \neq i} \left(\frac{d_{ij}}{\hat{d}_{ij}} \right)}{n}, \frac{\sum_{j \neq i} \left(- y_j \frac{d_{ij}}{\hat{d}_{ij}} \right) + y_i \sum_{j \neq i} \left(\frac{d_{ij}}{\hat{d}_{ij}} \right)}{n} \right) \tag{5.1}$$

Equation 5.1 can be reorganised to

$$\begin{aligned} (x_i^u, y_i^u) &= \left(\frac{x_i}{n} \left(1 + \sum_{j \neq i} \left(\frac{d_{ij}}{\hat{d}_{ij}} \right) \right), \frac{y_i}{n} \left(1 + \sum_{j \neq i} \left(\frac{d_{ij}}{\hat{d}_{ij}} \right) \right) \right) \\ &\quad - \left(\frac{\sum_{j \neq i} \left(x_j \frac{d_{ij}}{\hat{d}_{ij}} \right) + x_i}{n}, \frac{\sum_{j \neq i} \left(y_j \frac{d_{ij}}{\hat{d}_{ij}} \right) + y_i}{n} \right) \end{aligned} \tag{5.2}$$

The second term of equation 5.1, pulls the centroid of the map to the origin. This can be easily visualised by considering the case in which matrix D is generated by taking inter-point distance of map X. In this case estimated distance $\hat{D}$ and computed distance D are the same. Thus we have $d_{ij}/\hat{d}_{ij} = 1$. Substituting this equation 5.2 reduces to

$$(x_i^u, y_i^u) = (x_i, y_i) - \left(\frac{\sum_j x_j}{n}, \frac{\sum_j y_j}{n} \right) \tag{5.3}$$

We see from equation 5.3 that, the iterations will change the map, for as long as $\left(\frac{\sum_j x_j}{n}, \frac{\sum_j y_j}{n} \right) \neq 0$ (i.e., centroid of map not at origin).

This could be problematic for distributed localization as one has to use separate coordinate systems for each region. To avoid using separate coordinate systems, we have modified the equation 5.1 by adding a third compensating term, so as to stop the equation from moving the centroid of map to the origin. The modified equation is

$$\begin{aligned} (x_i^u, y_i^u) &= \left(\frac{x_i}{n} \left(1 + \sum_{j \neq i} \left(\frac{d_{ij}}{\hat{d}_{ij}} \right) \right), \frac{y_i}{n} \left(1 + \sum_{j \neq i} \left(\frac{d_{ij}}{\hat{d}_{ij}} \right) \right) \right) \\ &\quad - \left(\frac{\sum_{j \neq i} \left(x_j \frac{d_{ij}}{\hat{d}_{ij}} \right) + x_i}{n}, \frac{\sum_{j \neq i} \left(y_j \frac{d_{ij}}{\hat{d}_{ij}} \right) + y_i}{n} \right) \\ &\quad + \left(\frac{\sum_j x_j}{n}, \frac{\sum_j y_j}{n} \right) \end{aligned} \tag{5.4}$$

which can be rearranged to

$$(x_i^u, y_i^u) = \left(\frac{\sum_{j \neq i} \left(x_j (1 - \frac{d_{ij}}{\hat{d}_{ij}}) \right) + x_i \left(1 - \sum_{j \neq i} \left(1 - \frac{d_{ij}}{\hat{d}_{ij}} \right) \right)}{n}, \frac{\sum_{j \neq i} \left(y_j (1 - \frac{d_{ij}}{\hat{d}_{ij}}) \right) + y_i \left(1 - \sum_{j \neq i} \left(1 - \frac{d_{ij}}{\hat{d}_{ij}} \right) \right)}{n} \right) \tag{5.5}$$

The new distributed algorithm scheme is using equation 5.5. Initially each node assumes a random coordinate. Each node, computes its new position using equation 5.5 and announces the new coordinate to its neighbours. Each neighbour updates their local distance table on receiving the announcement. This process is repeated until there are no more significant updates in the coordinates. Perfect synchronisation of collection of data from all the nodes will require a tedious protocol. The scheme we propose is:

- At a fixed regular interval, $T1$, each node computes its new location using equation 5.5 with data from its *distance table* (Maintained by each mote).
- The above computed coordinate is broadcasted in the medium, after a random back-off period(which must be lesser than $T1$) .The back-off is introduced as a preemptive measure, to avoid collisions. n used in equation 5.5 is set to the number of entries in the *distance table*, rather than number of nodes in network.
- Each node within range registers this new coordinate and updates the *distance table*, provided that no collision of packets has occurred. These new coordinates are used in the next iteration for the computation of coordinates.
- This process is repeated for as long as computation provides new coordinates different from the old ones.

This computation is done by every node in the network. We can now relax the criteria of all nodes being connectible, as we are no more using the D matrix (which required the network to be connectible) in the computation of coordinates. This scheme can be visualised as each node computing its coordinates using data in a

particular sub-region. The network has as many sub-regions as the nodes which vary smoothly. Sub-regions of two adjacent nodes, have very few different nodes. Hence the stitching work is done implicitly by the equation 5.5.

Fig. 5.2 shows the scheme of final algorithm implemented, for distributed localization. Fig. 5.1 defines the blocks used in the final algorithm.

Initialise iteration

```
x' = 0;
y' = 0;

factor = 0;
size = 0;

backoff_timer.start(
    Random.rand() % T2);
compute_timer.start(T1);
/* T2 < T1 */
```

'Back-off' timer fired

```
msg {
    x;
    y;
};

msg.x = x;
msg.y = y;

broadcast(msg);
```

On receipt of msg

```
δ = estimated distance from
      received message's ED

d = distance between (x, y)
      and (msg.x, msg.y)

x' += (1 - d/δ) * (msg.x);
y' += (1 - d/δ) * (msg.y);

factor += (1 - d/δ);
size += 1;
```

'Compute' timer fired

```
factor /= size;

x' = (x'/size +
    (1 - factor) * x);
y' = (y'/size +
    (1 - factor) * y);

/* (x',y') ⇐ new coordinates */
```

test convergence

```
swap (x, y) with (x', y')
if (|x - x'| < threshold) and
    (|y - y'| < threshold) {
      return TRUE;
      IM finished;
} else {
      return FALSE;
      /* call ''initialise
              iteration'' */
}
```

Figure 5.1: Blocks used in Fig. 5.2

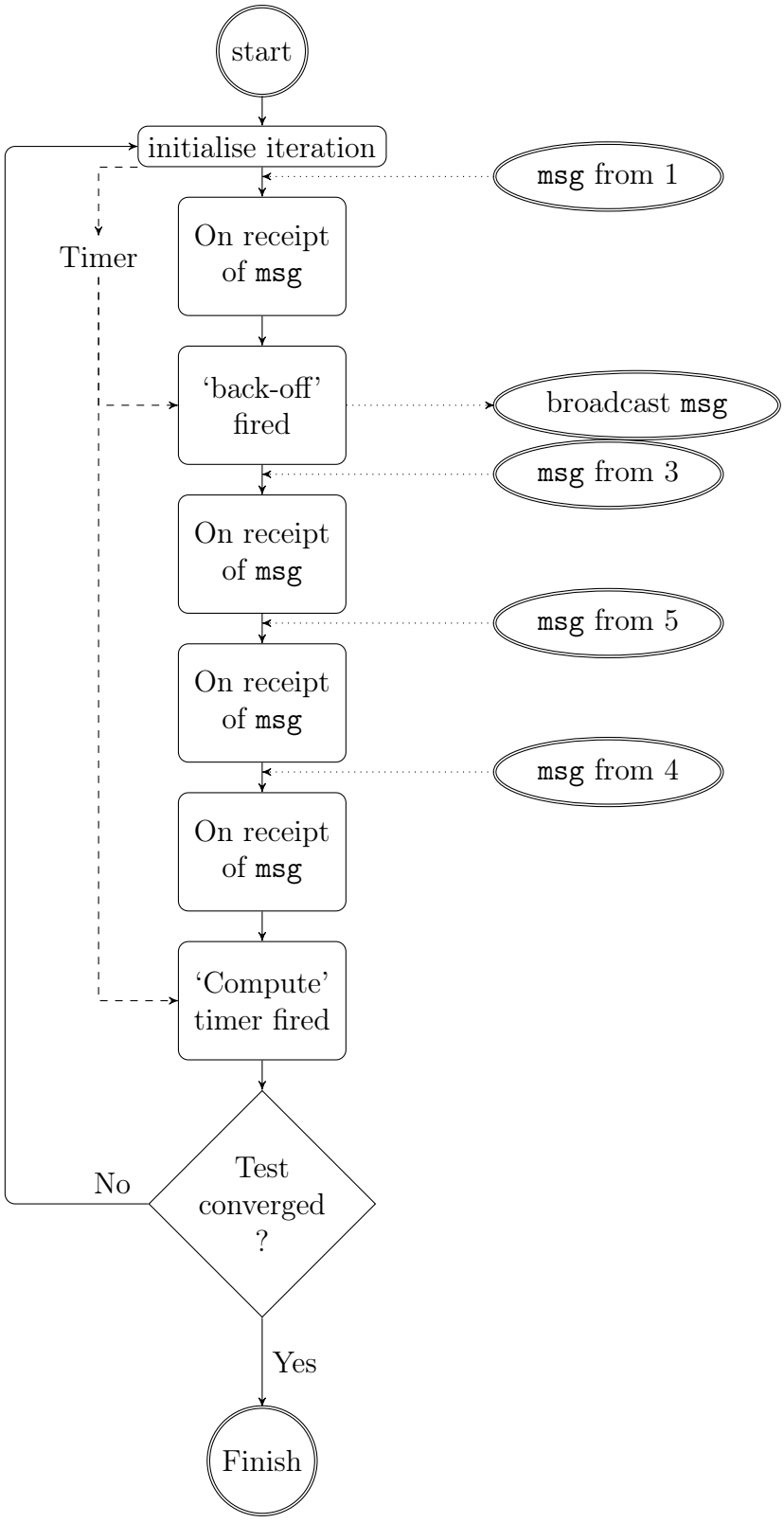


Figure 5.2: Flow chart of distributed iterative majorization. Details of blocks used are provided in Fig. 5.1

No. of points, $\sigma_n(\vec{X}, \Delta)$, iterations consumed mentioned at the top of each image. Both axes of all the plots are distance in m. The plots show vectors from actual coordinate to estimated coordinate obtained by the MDS technique. Note that most of the vectors in plots of experiments using ideal distances are hidden under the coordinate labels, as the results found are accurate.

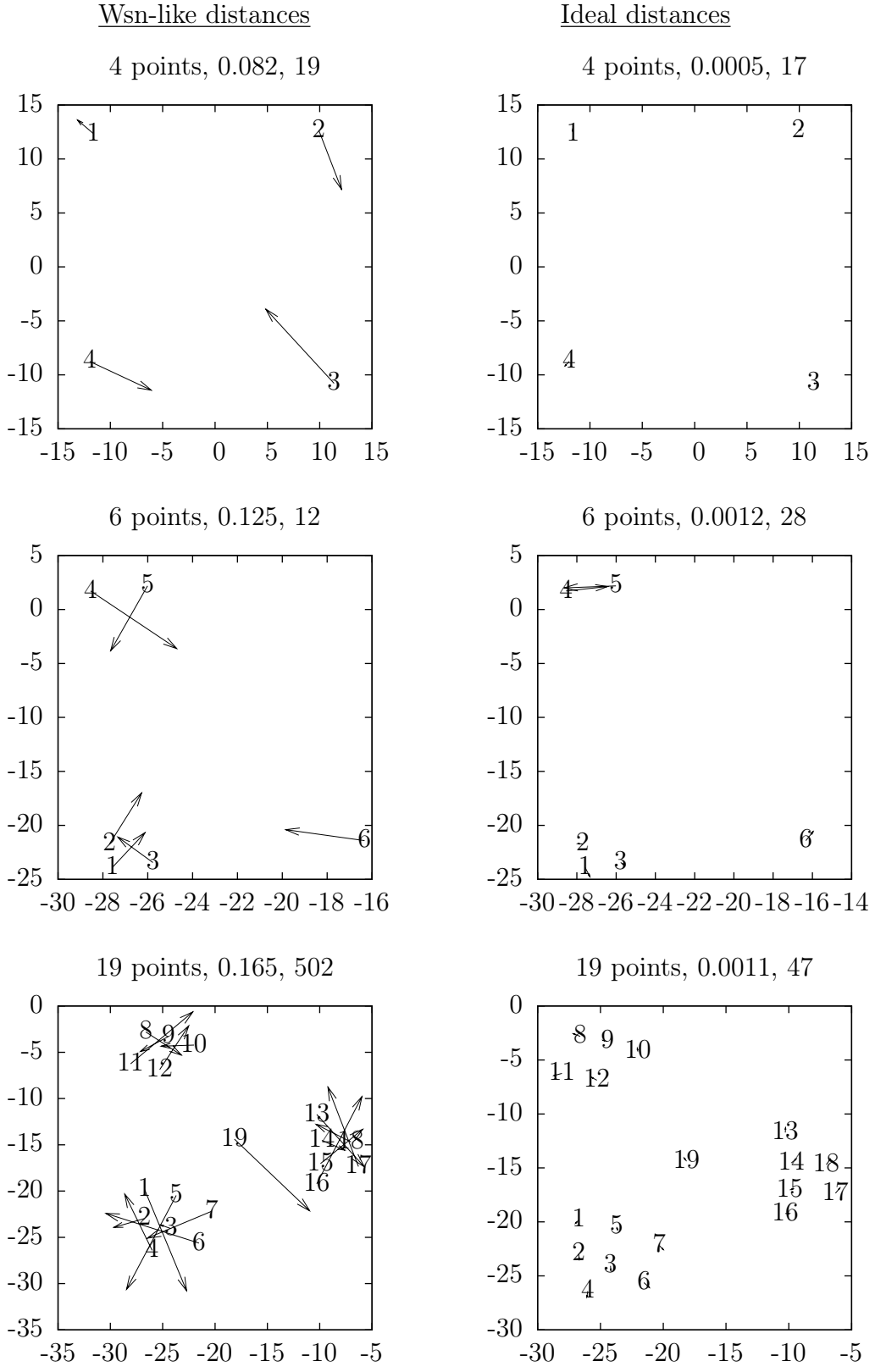


Figure 5.3: Results of a few test-cases on simulation with IM using ideal distances

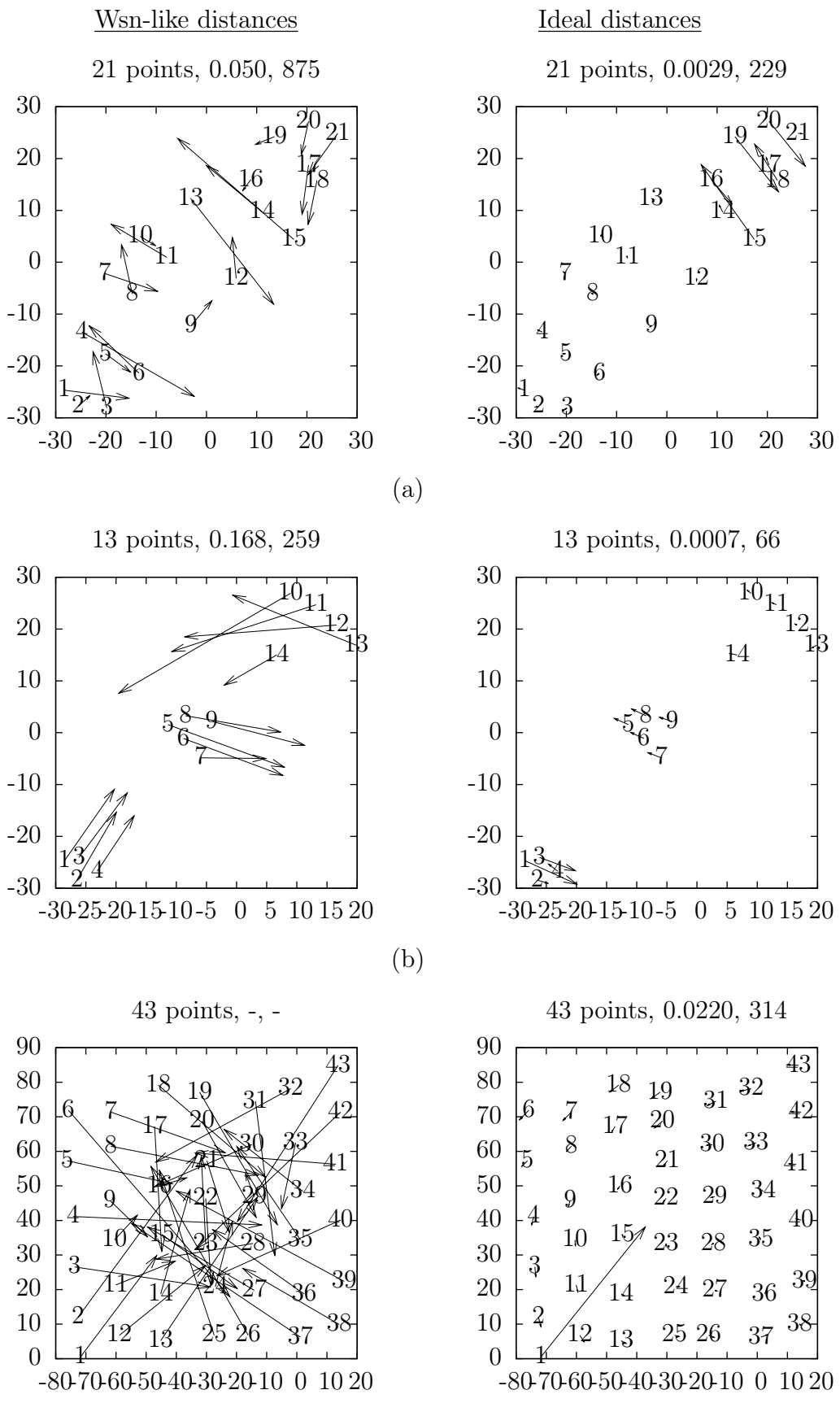


Figure 5.4: Continued from 5.3

Fig. 5.3 and Fig. 5.4 show the input and output of a few test cases and also the iterations required and final loss. test cases, have been simulated with ideal distances and wsn-like distances. Note that the communication range is set to 50m. Test cases in Fig. 5.4 have nodes with more than 50m inter-nodal distance. Hence these maps are not completely connectible. We can observe that test cases have achieved a good accuracy using ideal distances. These same test cases, using wsn-like distances, achieved a decent accuracy in Fig. 5.3 where the maps are completely connectible. But wsn-like distances deteriorated performance in the Fig. 5.4 where nodes are not completely connectible. In fact in test case (c), which had very low degree of connectivity, had worst results, and showed absolutely no convergence of the loss computed.

This algorithm uses $\mathcal{O}(1)$ memory on each node. The memory (both code and variables) required on each mote will be a few 100 bytes. Hence we consider this algorithm efficient in terms of memory usage.

However as the degree of connectivity decreased, the time/iterations required for the localization to converge increased. We speculate that the speed can be increased, if we initialise the network, with approximate coordinates rather than random values. These approximate coordinates can be found using range free techniques. Alternatively one can localise locally in rigid sub regions and explicitly stitch the maps to generate a global map.

Chapter 6

Conclusions and suggestions for further work

The main aim of the present work was to study different MDS techniques and provide a solution for in-network localization. Simplex algorithm, simulated annealing and iterative majorization were implemented and simulated using wsn-like distances. The results show that iterative majorization outperforms the other two techniques. From the speed of convergence, it can be safely assumed that iterative majorization is one of the fastest among the available techniques.

IRIS motes were chosen for implementing and testing localization. Various metrics available at the time of radio communication have been compared. RSSI was the only metric that provided useful information for range estimation. We observed significant ground-bounce effect in the signal attenuation pattern. This is a fact that is largely unnoticed in WSN literature for localization. Localization was implemented and tested using TinyOS and TOSSIM. Ground-bounce significantly decreased the localization accuracy.

The initial version of localization using iterative majorization, had a few problems. It required that all nodes in network should be connectible and was aggressive in memory usage. A slightly modified version of iterative majorization was devised which addressed connectibility and memory issues. This version used distributed

computation. The condition of completely connectible network has been relaxed. Memory usage was reduced from $\mathcal{O}(n^2)$ on one mote to $\mathcal{O}(1)$ on n motes. However the time required to solve large maps of low connectivity will be significantly large. This method proved disastrous using wsn-like distances when the connectivity among nodes is low.

6.1 Suggestions for further work

In our work, we haven't studied in detail the effect of degree of connectivity on the performance of distributed algorithm. Such a study could provide insights on the robustness of the algorithm. We have seen errors in localization that are introduced by ground bounce effect of path loss. Research can be done to provide range estimation techniques that compensate for known distortions of path loss. In the case of mobile nodes, Kalman filtering can decrease these errors to some extent.

Good accuracy is tough to provide using in-network localization due to physical limitations. Hence the other important direction of work is to improve techniques so as to provide the confidence level of a given result for each particular node. This information can be helpful in many scenarios.

We have seen that iterative majorization is computationally simple and efficient compared to most other localization techniques. In our work we visualise the network in 2 dimensions, based on their signal strengths. It is possible that a higher dimensional analysis, can provide valuable information that can be further used by routing techniques in finding energy efficient paths for transport of packets.

References

[1] N. Patwari, "Wireless sensor network localization measurement repository." http://www.eecs.umich.edu/ hero/localize.

[2] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, p. 10, 2000.

[3] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, no. 5, pp. 16–27, 2000.

[4] D. Blough, M. Leoncini, G. Resta, and P. Santi, "The k-neigh protocol for symmetric topology control in ad hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, p. 152, ACM, 2003.

[5] M. Kochhal, L. Schwiebert, and S. Gupta, "Role-based hierarchical self organization for wireless ad hoc sensor networks," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, p. 107, ACM, 2003.

[6] A. Amis and R. Prakash, "Load-balancing clusters in wireless ad hoc networks," in *Proceedings of 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, pp. 25–32, 2000.

[7] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, and C. MIT, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on wireless communications*, vol. 1, no. 4, pp. 660–670, 2002.

[8] L. Subramanian and R. Katz, "An architecture for building self-configurable systems," in *First Annual Workshop on Mobile and Ad Hoc Networking and Computing*, pp. 63–73, 2000.

[9] M. Ad, E. Royer, C. Perkins, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," *Internet Draft*, 2000.

[10] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks." http://www.usenix.org/events/osdi02/tech/full_papers/madden/madden_html/paperhtml.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on bile computing and networking*, pp. 56–67, ACM, 2000.

[12] H. Karl and A. Willig, *Protocols and architectures for wireless sensor networks*. Wiley-Interscience, 2007.

[13] P. IEEE802, "15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," *LAN/MAN Standards Committee of the IEEE Computer Society*, 2003.

[14] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.

[15] A. Tanenbaum and A. Woodhull, *Operating systems: design and implementation*. Prentice Hall India, 1997.

[16] "TinyOS." http://www.tinyos.net/.

[17] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, p. 11, ACM, 2003.

[18] S. Li, R. Sutton, and J. Rabaey, "Low power operating system for heterogeneous wireless communication systems," in *Compilers and operating systems for low power*, pp. 1–16, Kluwer Academic Publishers, 2003.

[19] "ecos." http://ecos.sourceware.org/.

[20] H. Vlado, D. Gay, *et al.*, "Hardware abstraction architecture in TinyOS." http://www.tinyos.net/tinyos-2.1.0/doc/html/tep2.html.

[21] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *IEEE International Conference on Acoustics Speech and Signal Processing*, vol. 4, Citeseer, 2001.

[22] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[23] N. Srour, "Unattended ground sensors a prospective for operational needs and requirements," *ARL Report Prepared for NATO*, 1999.

[24] S. Kumar, F. Zhao, and D. Shepherd, "Collaborative signal and information processing in microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 13–14, 2002.

[25] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, p. 173, 2005.

[26] S. Oh and S. Sastry, "Tracking on a graph," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 195–202, 2005.

[27] P. Sridhar, A. Madni, and M. Jamshidi, "Intelligent Object-Tracking using Sensor Networks," in *IEEE Sensors Applications Symposium, 2007. SAS'07*, pp. 1–5, 2007.

[28] X. Hong, K. Xu, M. Gerla, and C. Los Angeles, "Scalable routing protocols for mobile ad hoc networks," *IEEE network*, vol. 16, no. 4, pp. 11–21, 2002.

[29] B. Karp and H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 243–254, ACM, 2000.

[30] J. Xu, X. Tang, and W. Lee, "A new storage scheme for approximate location queries in object-tracking sensor networks," *IEEE transactions on parallel and distributed systems*, vol. 19, no. 2, pp. 262–275, 2008.

[31] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hash table for data-centric storage," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, p. 87, ACM, 2002.

[32] S. Zhang, G. Li, W. Wei, and B. Yang, "A Novel Iterative Multilateral Localization Algorithm for Wireless Sensor Networks," *Journal of Networks*, vol. 5, no. 1, p. 112, 2010.

[33] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

[34] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proceedings of the 9th*

annual international conference on Mobile computing and networking, pp. 81–95, ACM, 2003.

[35] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," *Telecommunication Systems*, vol. 22, no. 1, pp. 267–280, 2003.

[36] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from mere connectivity," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pp. 201–212, ACM, 2003.

[37] N. Patwari, J. Ash, S. Kyperountas, A. Hero Iii, R. Moses, and N. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal processing magazine*, vol. 22, no. 4, pp. 54–69, 2005.

[38] S. Roumeliotis and G. Bekey, "Synergetic localization for groups of mobile robots," in *Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 4, 2000.

[39] A. Savvides, C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc wireless sensor networks," 2001.

[40] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable coordination for wireless sensor networks: self-configuring localization systems," in *Proceedings of the 6th International Symposiumon Communication Theory andApplications*, Citeseer, 2001.

[41] C. Savarese, J. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *USENIX technical annual conference*, vol. 2, Monterey, CA, 2002.

[42] J. Costa, N. Patwari, and A. Hero III, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 1, p. 64, 2006.

[43] J. Costa, N. Patwari, and A. Hero III, "Achieving high-accuracy distributed localization in sensor networks," in *Proceedings on IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 2005.

[44] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, 2004.

[45] W. Forrest, *Encyclopedia of Statistical Sciences*, vol. 5. John Wiley & Sons Inc., 1985. Relevant section available at http://forrest.psych.unc.edu/teaching/p208a/mds/mds.html.

[46] W. Torgerson, "Multidimensional scaling: I. Theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[47] B. Wojciech, "Multidimensional scaling." http://www.pavis.org/essay/multidimensional_scali

[48] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes in C*. Cambridge university press Cambridge, 1992.

[49] "Gnu scientific library (gsl)." http://www.gnu.org/software/gsl.

[50] I. Borg and P. Groenen, *Modern multidimensional scaling: Theory and applications*, ch. 8. Springer Verlag, 1997.

[51] J. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, p. 308, 1965.

[52] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, pp. 406–413, Cambridge University Press, 2008.

[53] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi, *GNU scientific library*. Citeseer, 2002.

[54] N. Patwari, A. Hero III, M. Perkins, N. Correal, and R. O'dea, "Relative location estimation in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2137–2148, 2003.

[55] C. Technology, "Iris." http://www.xbow.com/Products/productdetails.aspx?sid=264.

[56] Atmel, "Rf230." http://www.atmel.com/dyn/products/product_card.asp?part_id=3941.

[57] MAXIM, "Path loss in remote keyless entry systems." http://www.maxim-ic.com/app-notes/index.mvp/id/3945, 2006.

[58] "Tossim." http://docs.tinyos.net/index.php/TOSSIM#Introduction.

Figure 1: IRIS mote

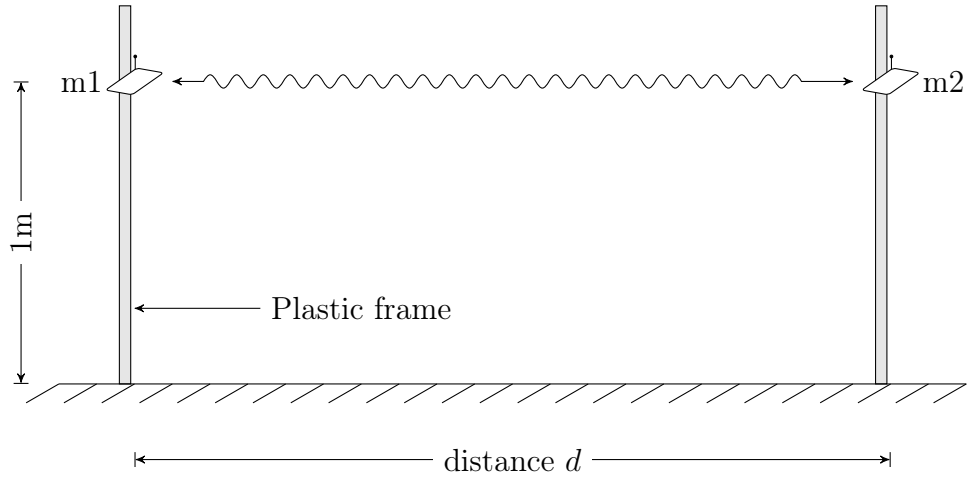


Figure 2: Experimental setup to measure the various metrics for range estimation