ON DEPLOYMENT AND SECURITY IN MOBILE WIRELESS SENSOR NETWORKS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Sriram Chellappan, B.E., M.S., M.S.

* * * * *

The Ohio State University

2007

Dissertation Committee:

Dong Xuan, Adviser
Eylem Ekici
Ten H. Lai

Approved by

__________________________
Adviser
Graduate Program in Computer Science and Engineering
ABSTRACT

Wireless sensor networks have become increasingly pervasive with promises to fulfill many of our critical necessities today. One issue that has permeated sensor networks recently is mobility. Broadly, mobility in sensor networks can be categorized into two classes: Internal mobility and External mobility. Internal mobility is the class where sensors themselves can move from one location to another, while external mobility is the class where certain external agents (not sensors) move in the network. Both mobility classes have patent and significant impacts to sensor networks operation. However, being an emerging topic, a clear understanding of opportunities and challenges of sensor networks mobility is lacking today, and hence is an important need of the hour. In this dissertation, we make contributions in both classes of sensor networks mobility.

First, we study the issue of how sensors can use their mobility to enhance quality of network deployment. We define two representative mobility assisted sensor network deployment problems. In our mobility model, there are hard limitations in both sensors’ mobility pattern and distance. Such limitations are natural due to constraints on sensors’ form-factor and energy. We identify critical challenges arising in deployment under such hard mobility limitations. We then propose a suit of sensor mobility algorithms for our deployment problems, and demonstrate their performance using theoretical analysis and extensive simulations.
Second, we study the issue of external mobility in sensor networks from a security perspective. We identify a unique security threat in sensor networks called physical attacks. We define a representative model of physical attacks, wherein an external mobile agent (human being or robot) moves in the network detecting inherent physical/electronic sensor signals to localize sensors, and then physically destroys them. We formally model such attacks in sensor networks, demonstrate their destruction potential, identify variations, and finally propose countermeasure guidelines against them.

With the emergence of mobility in wireless sensor networks, coupled with its significances, we hope that our work in this dissertation can provide strong foundations and further motivations for researchers to explore this topic that promises to revolutionarize sensor networks research in the near future and beyond.
Dedicated To

Ramya

For Her Love, Support And Trust
ACKNOWLEDGMENTS

First and foremost, I would like to express my sincerest gratitude to my Adviser, Dr. Dong Xuan. The last five years of working with him has been tremendously rewarding for me across several spectrums. His overall vision and application to research has left a deep positive influence on me, which I hope to imbibe and practice for the rest of my life. Furthermore, his equal doses of support and criticisms during advising were always constructive, and has greatly helped channel my research in the right directions. I am truly grateful to him for all his help. I also wish to sincerely express my gratitude to my committee members, Dr. Eylem Ekici and Dr. Ten H. Lai for their overall guidance and inputs to me in the past year. I would also like to acknowledge the National Science Foundation (NSF) and Army Research Office (ARO) for funding part of our research.

At Ohio-State, I have been fortunate to be tutored by many magnificent teachers. These include Dr. Hitay Ozbay, Dr. Vadim Utkin, Dr. Ten H. Lai, Dr. Eylem Ekici, Dr. Arjan Durresi, Dr. Ken Supowit, Dr. Neelam Soundarajan, Dr. Tim Long, Dr. Tim Carlson, Dr. Anish Arora, Dr. Ming T. Liu and many others. All these teachers have had tremendously positive influences at various stages of my academics, and I am truly grateful to them.

This work would have never reached completion without help from my colleagues in our research group. Thanks to Xun Wang, Wenjun Gu, Xiaole Bai, Zhimin Yang
for numerous collaborations and discussions. Their intelligence, diligence and compassion were second to none, and has gone a great way in shaping my overall research productivity. I also wish to express my sincerest thanks to several other colleagues in computer science including Mukundan Sridharan and Kai-Wei Fan for being such wonderful friends.

I wish to also express my sincerest gratitude to the never say no staff of the Computer Science and Engineering Department at Ohio-State. Ewana, Tom, Catrena, Elizabeth, Carrie, Marty, Tamera have all been so kind and supportive to my needs at innumerable times. My stay in Computer Science would not have been half as comfortable without the terrific staff the department has.

On a more personal note, I express my ineffable gratitude to Steve Yao and Satoko Itani. Simply put, these are two persons whose influence on me permeates just about every sphere of my existence. I also wish to thank Vaidy, Gowtham, Andre and many other friends of mine at Ohio-State who have made my stay so enjoyable in the last several years.

My family has been greatly supportive of me in any decision that I have taken including my decision to pursue a Ph.D. degree. I would like to express my thanks and gratitude to my parents, my sister and my cousins for all their support, encouragement and numerous sacrifices during the last several years.

Finally, I would like to express my gratitude to my beloved wife Ramya. Ramya has been the most positive influence on me in the last couple of years. Her patience, empathy, love, support and trust can never be compensated. She is truly the light of my life. Small wonder that I dedicate this work to her.
VITA

April 7, 1978 ............................... Born - Palani, India

1999 ........................................Bachelor in Engineering
University of Madras

2002 ........................................Master of Science
The Ohio-State University

2006 ........................................Master of Science
The Ohio-State University

2002-present ..............................Graduate Research/ Teaching Asso-
ciate
The Ohio-State University

PUBLICATIONS

Research Publications


FIELDS OF STUDY

Major Field: Computer Science and Engineering
Studies in:

Computer Networking  Prof. Anish Arora
                    Prof. Eylem Ekici
                    Prof. Ten H. Lai
                    Prof. Ming T. Liu
                    Prof. Dong Xuan

Theory                Prof. Tim Carlson
                    Prof. Tim Long

Control Systems       Prof. Hitay Ozbay
                    Prof. Vadim Utkin
# TABLE OF CONTENTS

| Abstract | ii |
| Dedication | iv |
| Acknowledgments | v |
| Vita | vii |
| List of Figures | xiv |

## Chapters:

1. **INTRODUCTION** ................................................................. 1
   1.1 Mobility in wireless sensor networks ................................. 1
       1.1.1 Internal mobility in wireless sensor networks .............. 2
       1.1.2 External mobility in wireless sensor networks ............. 3
   1.2 Contributions of this dissertation and their significances .... 4
       1.2.1 Mobility algorithms design for wireless sensor networks deployment ........................................ 4
       1.2.2 Modeling mobile agent orchestrated physical attacks in wireless sensor networks ............... 8
   1.3 Organization of this dissertation ...................................... 10

2. **MOBILITY ALGORITHM DESIGN FOR 1–COVERAGE IN WIRELESS SENSOR NETWORKS** ............................. 11
   2.1 Motivations ............................................................... 11
   2.2 Mobility Model and Problem Definition ............................. 15
       2.2.1 The Flip-based Sensor Mobility model ....................... 15
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Implementations of limited mobility sensor models</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>A snapshot of the sensor network and a movement plan to maximize coverage (a), and the resulting deployment (b)</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>A snapshot of the sensor network and the optimal movement plan (a), and the resulting deployment (b)</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>The Virtual Graph with only regions 1 and 2 in it</td>
<td>24</td>
</tr>
<tr>
<td>2.4</td>
<td>The initial sensor network deployment (a) and the corresponding virtual graph at the start (b) in Case $R = d$</td>
<td>27</td>
</tr>
<tr>
<td>2.5</td>
<td>The initial sensor network deployment (a) and the corresponding virtual graph at the start (b) in Case $R &gt; d$</td>
<td>29</td>
</tr>
<tr>
<td>2.6</td>
<td>Sensitivity of $CI$ to $F$ under different $C$ in 150$ \times $150 and 300$ \times $300 network</td>
<td>37</td>
</tr>
<tr>
<td>2.7</td>
<td>Sensitivity of $FD$ to $F$ under different $C$ in 150$ \times $150 and 300$ \times $300 network</td>
<td>38</td>
</tr>
<tr>
<td>2.8</td>
<td>Sensitivity of $CI$ to $F$ under different $R$</td>
<td>40</td>
</tr>
<tr>
<td>2.9</td>
<td>Sensitivity of $CI$ to $F$ under different $\sigma$</td>
<td>40</td>
</tr>
<tr>
<td>2.10</td>
<td>Sensitivity of $CI$ to $A$ in 150$ \times $150 and 300$ \times $300 network</td>
<td>43</td>
</tr>
<tr>
<td>2.11</td>
<td>Sensitivity of $PN$ to $A$ in 150$ \times $150 and 300$ \times $300 network</td>
<td>43</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

A wireless sensor network typically consists of many tiny, battery-powered devices with built-in CPU and radio links that can sense particular characteristics in the vicinity of their environment and communicate with each other. Some well formed applications of sensor networks include battlefield monitoring, target/intruder tracking, volcanic monitoring, structural health monitoring etc. A significant amount of theory has already been invested in sensor networks, and simultaneously many prototype systems have been built and thoroughly tested. The unifying conclusions after all these efforts are the clear and patent viability of wireless sensor networks to fulfill many of our critical necessities in the near future and beyond.

1.1 Mobility in wireless sensor networks

With the tremendous growth of wireless sensor networks both in terms of technology and applications, one issue that is gaining rapid attention in this realm is mobility. We categorize mobility in sensor networks into two classes: Internal mobility and External mobility. Internal mobility is the class where sensors themselves are capable of moving from one location to another in the network with a particular purpose. A typical instance of this class is a sensor that is fitted on a mobile
platform (say a motor driven robot) and capable of making movement decisions by itself in order to fulfill certain mission objectives like repair coverage holes, physically move closer to events in the network for better sensing etc. On the other hand, external mobility is the class where certain external agents (not sensors) move in the network with a particular purpose. A typical instance of this class is a vehicle or a soldier that is moving in the sensor network and interacting with the sensors for one of several objectives including data gathering, sensor energy replenishment, sensor localization etc. The incorporation of mobility to sensor networks (both Internal and External mobility) has resulted in a wide spectrum of significances to sensor networks operations as discussed below \(^1\).

### 1.1.1 Internal mobility in wireless sensor networks

When sensors themselves have the ability to control their movements from one location to another, there are tremendous benefits to the network. Consider deployment. In many situations like battlefields, volcanic zones, forest fire zones etc., hand-placement of sensors in selected spots is not always feasible during deployment. In such cases, sensors are randomly deployed in the network (e.g., sprayed from a remote location like say an air-plane). Consequently, the sensor positions on the deployment field cannot be controlled, and it may happen that some parts of the network may not be covered (i.e., not within the sensing range of any sensor). Also, as time goes on, sensors in some part of a network may die (due to faults, energy losses etc.), which may compromise coverage/ connectivity. When sensors are mobile, they can relocate themselves automatically to correct such deployment shortcomings \([1],[2],[3],[4]\).

\(^1\)More detailed discussions on the works discussed in Sections 1.1.1 and 1.1.2 appear under Related Work in Chapter 5.
Other benefits of sensor mobility include the design of better intruder tracking systems presented in [5], where it is shown how sensors with mobility can help track intruders in a faster and more accurate manner than if the sensors were just static. In works like [6], [7], [8], sensor mobility is leveraged to enhance event detection quality and reliability by enabling sensors to move closer to events for better sensing as and when they dynamically occur in the network. In [9], security of the sensor network is improved by leveraging mobility, where sensors relocate themselves to selectively find better (more secure) neighbors for keys establishment, while simultaneously ensuring that coverage is not compromised.

1.1.2 External mobility in wireless sensor networks

When there are certain external agents moving in a sensor network, a host of opportunities lie in store. Typically, in all existing works, the mobile agent is assumed to be more powerful than the sensors themselves (which is highly reasonable). The stronger mobile agent will then assist the sensors in the network to overcome some of their inherent weaknesses during the network operation. In [10], [11], [12], [13], [14], an external mobile agent periodically moves in the network collecting information stored by the sensors. In other words, the mobile agent will itself act as a sink, or will act as a relay to the sink. The objective here is to leverage the resource rich mobile agent (that can reach closer to sensor nodes) for data delivery in order to minimize energy consumption, compared the case where sensors themselves have to transmit data via multiple hops to the sink. In [15], algorithms are designed to leverage the mobility of certain energy rich external mobile agents to deliver energy to static sensors as and when their energy is depleted due to network operations. In works
like [16], [17], [18], [19], an external mobile agent is used to localize many static sensors in the network. The motivation for this line of work is primarily because, being small in size sensors cannot be provisioned with GPS like devices for localization. Thus a GPS provisioned external mobile agent is used to move towards sensors and assist them in their localization.

1.2 Contributions of this dissertation and their significances

In this dissertation, we makes contributions in both classes of mobility in sensor networks. Our specific contribution in each class and their significances are listed below.

1.2.1 Mobility algorithms design for wireless sensor networks deployment

Contributions: Our first contribution is the design of high-performing sensor mobility algorithms for deploying clustered wireless sensor networks under hard sensor mobility limitations \(^{2}\) [20], [21], [22]. Specifically, 1). We first define two representative sensor mobility assisted deployment problems in clustered sensor networks for both 1-coverage and \(k\)-coverage (\(k \geq 1\)) per region (i.e., cluster) under hard limitations in the movement model and movement distance of each sensor. 2). We design novel weight-based methodologies to convert certain non-linear optimization objectives in our deployment problems (formally presented later) to linear objectives. 3). We then design centralized optimum and distributed heuristic sensor mobility algorithms for our deployment problems using ideas from minimum cost maximum flow approaches in graph theory. 4). We demonstrate the performances of our algorithms using

\(^{2}\)Note that for the rest of the rest of dissertation, the term sensor mobility always refers to internal mobility unless otherwise stated.
extensive analysis and simulations. Finally, we present detailed discussions on the generality of our methodologies and algorithms to address a wide spectrum of mobility assisted deployment problems in sensor networks, other than just the ones formally addressed in this dissertation.

Significances: The first significance of our work comes from the nature of our deployment problems being representative in many sensor network applications. The approach to cluster sensor networks has been widely adopted in practice [23], [24], [25] and [26]. Clustering a sensor network into regions (or clusters) enhances scalability, improves routing and power efficiency, provides better support to higher level functionality etc. From a coverage perspective, it is clearly important to cover every region in the network in order to be able to detect all events occurring there. For many sensor networks applications, 1–coverage suffices. However, the requirement of at-least one sensor per region requirement may not be always satisfied. For example, if sensors are randomly deployed (sprayed from a vehicle, airdropped etc.), the requirement is hard to be satisfied, as the location of the sensors after deployment cannot be guaranteed. Even if at initial deployment all regions have at-least one sensor, as time goes on, faults, failures, energy losses etc. can violate this requirement. In such situations, sensors can leverage mobility to current such violations. However, many sensor networks today are also being increasingly deployed in hostile zones like battlefields, international borders etc. For such missions, 1–coverage might not suffice, and the requirement will be that each region needs at-least \( \bar{k} \) sensors \( (\bar{k} \geq 1) \). The desired number of sensors \( (\bar{k}) \) per region can be contingent on one or more factors including sensing considerations, mission criticality, fault tolerance, resilience to attacks, lifetime etc. For instance, consider sensing. Due to physical limitations on
sensors and associated hardware, the sampling rate at which sensors can sense the environment may be limited (e.g., 100 kHz for acoustic sensors [27], and 4200 Hz for magnetometers [28]). As such in critical applications where the environment may have to be sensed quite frequently, or at all times of operation, multiple sensors per region have to be deployed to meet sensing objectives. Secondly, there could be obstacles in the regions or external factors (like heat, vibration etc.) that may affect sensing ranges during network operation. Such sensing dynamics can be compensated with multiple sensors per region. Furthermore, fault tolerance and resilience to attacks improve with multiple sensors; lifetime can be prolonged using role rotation among multiple sensors etc. Hence, in many mission-critical scenarios, multiple sensors per region is clearly necessary. As discussed before, when such deployment requirements are violated, sensor mobility can be leveraged to correct such violations.

The second significance of our work stems from the considerations of hard mobility limitations in sensors, which existing works do not. Specifically, in all existing works on sensor mobility algorithms design [1–9], it is assumed that if a sensor wishes to
move to a new location from its current location, it can do so without any restriction in its movement distance. However, this is a very restrictive assumption. Sensors are severely energy constrained, and available energy has to be shared for sensing, data processing, transmission etc. Since mobility also consumes energy, it is very likely that there is a limit on the overall movement distance capability of the sensors. This is especially true in environments when sensors have to work un-attended after deployment, and where recharging sensors is not always feasible (e.g., hostile zones, battlefields etc.). To validate this claim on mobility limitations, we briefly discuss four recent mobile sensor models implemented in practice, shown in Figures 1.1 (a) to (d). As part of the self-healing minefield program, DARPA has designed a class of mobile sensors called Intelligent Mobile Landmine Units (IMLM) with limited hop-by-hop mobility to detect and repair coverage breaches in battlefields [29] (Figure 1.1 (a)). The ability to move for these sensors comes from a fuel powered propulsion mechanism. Clearly, the mobility is constrained by the amount of available fuel. In the implementation in [29], the sensors can make up to 100 hops only. In the XYZ model in [30], a motion-enabled and power aware sensor node platform has been designed (Figure 1.1 (b)). A battery enabled miniature geared motor actuates sensor motion. In this design, the maximum movement distance of the sensor is 165 meters only. Other implementations of mobile sensor platforms include the Robomote [31] and Khepera models [32] (Figures 1.1 (c) and (d)) which are also motor driven, and whose overall movement distance is limited to about 360 meters and 3600 meters respectively. As such, while the internal mobility semantics may be different in each implementation, the fact is that the sensor’s overall maximum movement distance is clearly limited. Existing algorithms for sensor mobility [1–9] that do not consider
hard mobility limitations clearly have limited applicability under such restrictions (detailed discussions appear under Related Work in Chapter 5), hence demonstrating the significance of our work.

1.2.2 Modeling mobile agent orchestrated physical attacks in wireless sensor networks

**Contributions:** Our second contribution is the modeling of physical attacks in wireless sensor networks orchestrated by a malicious external mobile agent [33]. Specifically, 1). We first illustrate the threat of physical attacks in sensor networks. Physical attacks are those that physically destroy sensors or otherwise render them permanently inoperable in the network. 2). We then define a highly representative model of physical attacks called *Search-based Physical Attacks* in sensor networks, where an external malicious mobile agent (human being or robot) enters the sensor network deployment field to search for inherent physical or electronic sensor signals using appropriate signal detection equipment. By measuring certain signal properties, the attacker locates sensors and then physically reaches the sensors to destroy them. 3). We then define a novel metric called Accumulative Coverage $AC$ to measure impacts of physical attacks. This metric simultaneously captures both the lifetime and coverage, two of the most fundamental metric in sensor networks. 4). Using formal analysis and extensive simulations, we highlight the significance of physical attacks threat to sensor networks from the perspective of losses in Accumulative coverage. 5). We also present variations to our basic physical attack model, and also discuss important guidelines for countermeasures against the threat of physical attacks in sensor networks.
Significances: To the best of our knowledge, all existing works in the realm of external mobile agents interacting with sensor networks focus on only the benign aspects of this interaction. Ours is the first work to consider the downsides of such interactions from a security perspective. Specifically, we define a novel threat from the perspective of physical destruction of small sensors in sensor networks. Physical attacks are easy to launch with low effort and technology. A mobile agent with simple equipment to detect signals and physically destroy sensors suffices. The ease of attack execution coupled with the inherent vulnerability of the small sized sensors to physical destructions makes physical attacks inevitable threats to sensor networks. The end effects of physical attacks can be disastrous. The backbone of the network (the sensors themselves) can be destroyed. Destruction of sensor nodes can also result in the violation of established network paradigms. These include accuracy and latency of data/event delivery, lifetime, coverage and connectivity. As such, a wide spectrum of impacts may be caused due to physical attacks. When left unaddressed, physical attacks have the potential to compromise the entire sensor network mission. Furthermore, the attacker can use mobility as a leverage to significantly enhance the geographical scope of sensor destruction, while simultaneously making it hard to localize the attacker and its impacts. The significance of our work lies in the identification of the problem, formal modeling of an external mobile agent orchestrated physical attacks, demonstrating their destruction potential and variations, and proposing countermeasure guidelines for defending sensor networks against them.
1.3 Organization of this dissertation

The rest of the dissertation is organized as follows. In Chapter 2, we present our work on algorithms design for limited mobility sensor networks deployment for 1-coverage. In Chapter 3, we present our work on algorithms design for limited mobility sensor networks deployment for \( k \)-coverage \((k \geq 1)\). In Chapter 4, we present our work on modeling external mobile agent orchestrated physical attacks in sensor networks. In Chapter 5, we discuss important related works, and we conclude this dissertation with some final remarks in Chapter 6.
CHAPTER 2

MOBILITY ALGORITHM DESIGN FOR 1-COVERAGE IN WIRELESS SENSOR NETWORKS

In this chapter, we study the issue of deploying wireless sensor networks for 1-coverage under limited mobility constraints. Specifically, we define a representative sensor networks deployment problem, and design an optimum centralized algorithm for the deployment problem. Extensive analysis and simulations are also presented to validate the performance of the proposed algorithm.

2.1 Motivations

Sensor networks deployment in an important phase of sensor networks operation. A host of works has appeared in this realm in the recent past [3], [2], [34], [1], [35], [36], [4], [37], [38]. One of the important goals of sensor networks deployment is to ensure that the sensors meet critical network objectives that may include coverage, connectivity, load balancing etc. When a number of sensors are to be deployed, it is not practical to manually position sensors in desired locations. In many situations, the sensors are deployed from a remote site (like from an airplane) that makes it very hard to control deployment.
To address this issue, a class of work has recently appeared where mobility of sensors is taken advantage of to achieve desired deployment [3], [2], [1], [4]. Typically in such works, the sensors detect lack of desired deployment objectives. The sensors then estimate new locations, and move to the resulting locations. While the above works are quite novel in their approaches, the mobility of the sensors in their models is unlimited. Specifically, it is assumed that if a sensor wishes to move to a new location, it can do so without any restriction in its movement distance. However, as discussed in the sensor mobility instances before, this may not always be true. Under hard mobility constraints, existing works have limited applicability. For instance, in the well known virtual force approach for balancing sensor networks deployment [1], [4] and [3], sensors exert virtual forces among themselves during movements. Two sensors repel (or attract) each other if they are too close (or too far apart). By balancing virtual forces, sensors spread themselves in the field. However, under hard mobility constraints, two sensors may not be able to achieve force balance if the distance required to be traversed is too large. Secondly, the virtual force approach results in several back and forth movements during force balancing, which across many iterations will rapidly deplete sensor mobility capacity. For similar reasons, other works on mobility assisted deployment also have limited applicability under hard sensor mobility constraints (more discussions on the challenges of limited mobility appear in the next section).

In this chapter, we study sensor networks deployment for 1-coverage using sensors with limited mobilities. Such limitations span both the movement pattern and movement distance. In our model, sensors can flip (or hop) only once to a new location, and the flip distance is bounded (similar to the model in Figure 1.1 (a)
in Section 1.2.1). We call such sensors as flip-based sensors. A certain number of flip-based sensors are initially deployed in the sensor network that is clustered into multiple regions. The approach to cluster sensor networks has been widely adopted in practice [23], [24], [25] and [26]. Clustering a network into regions (or clusters) enhances scalability, improves routing and power efficiency, provides better support to higher level functionality etc. Ideally, it is desirable that each region in the network have at-least one sensor in it to ensure that all events in that region are covered (i.e., all events are within the sensing range of at-least one sensor). In many situations, it may happen that the initial deployment may not cover all regions in the network. For example, if sensors are randomly deployed (sprayed from a vehicle, airdropped etc.), full coverage may not be possible. Such regions that do have any sensor in them are holes. Note that even if at initial deployment all regions have at-least one sensors, as time goes on, faults, failures, energy losses etc. can violate this requirement. In such cases, the limited mobility sensors have to self-adjust their positions to correct such violations. Our problem is to determine an optimal movement (or flip) plan for the sensors in order to maximize the number of regions that is covered by at-least one sensor (or minimize the number of holes), and simultaneously minimize the total number of sensor movements (or flips).

**Our Contributions:** We propose a minimum-cost maximum-flow based algorithm to our deployment problem. Our approach is to translate the sensor network at initial deployment and sensor mobility model into a graph (called virtual graph). Regions in the sensor network are modeled as vertices, and possible sensor movement paths between regions are modeled as edges between corresponding vertices in the virtual graph. Capacities for the edges model the number of sensors that can flip
between regions. A cost value is also assigned to the edges to capture the number of flips between regions. Since the virtual graph models the sensor network, our problem of optimally moving sensors to holes, can be translated as one where we want to optimally determine flows to hole vertices in the virtual graph. The first objective of our problem, namely determining a movement plan to maximize coverage can be translated as determining the flow plan (a set of flows in the virtual graph) that corresponds to the maximum flow to hole vertices in the virtual graph, without violating edge capacities. Note that, there can be more than one flow plan that can maximize the flow in the virtual graph. Out of such flow plans, our second objective is to determine the plan that minimizes the overall cost, which corresponds to the minimizing the number of sensor flips. Intuitively, each flow in the flow plan (in the virtual graph) denotes a path for a sensor movement to a hole in the sensor network. As we discuss later, the maximum flow value in the virtual graph denotes the maximum number of holes into which a sensor can move without violating the mobility constraints, while the minimum cost denotes the corresponding minimum number of sensor movements (or flips).

In our algorithm, we translate the flow plan corresponding to the minimum cost maximum flow in the virtual graph into a movement plan for the sensors in the region. We subsequently prove the optimality of this movement plan. We also propose multiple approaches that sensors can adopt to execute our algorithm in practice. We then perform simulations to study the sensitivity of coverage and the number of flips to flip distance under different initial deployment distributions of sensors. We observe that increased flip distance achieves better coverage, and reduces the number of flips required per unit increase in coverage. However, such improvements
are constrained by initial deployment distributions of sensors, due to the limitations on sensor mobility.

2.2 Mobility Model and Problem Definition

2.2.1 The Flip-based Sensor Mobility model

In this chapter, we model sensor mobilities as a flip. That is, the motion of the sensor is in the form of a flip (or hop) from its current location to a new one when triggered by an appropriate signal. Such a movement can be realized in practice by propellers powered by fuels [29], coiled springs unwinding during flips, external agents launching sensors after deployed in the field etc. In our model, sensors can flip only once to a new location. This could be due to propeller dynamics, or the spring unable to recoil after a flip, or the external agent launching the sensor. The distance to which a sensor can flip is limited. The sensor can flip in a desired angle. Mechanisms in [29] can be used for orientation during flips. The limitation in sensor mobility comes from the bound on the maximum distance they can move, which again depends on available fuel quantity, degree of spring coil etc. We study two models of flip-based mobility. The first is a fixed distance mobility model, while the second is a variable distance mobility model.

We denote the maximum distance a sensor can flip to as $F$. In the first model, the distance to which a sensor can flip is fixed and is equal to $F$. We extend the above model further. Although the number of flips is still one, in many cases, depending on the triggering signals, fuel can be metered variably, or the spring can unwind only partially, or the external agent can variably adjust the flip distance during launching. In the second model, sensors can flip to distances between 0 and $F$. We denote $d$
as the basic unit of distance flipped. We assume that $F$ is an integral multiple of the basic unit $d$. Thus in the second model, sensors can flip once to distances $d$, $2d$, $3d$, $\ldots$ $nd$ from its current location, where $nd = F$. To differentiate the above two models, we introduce the notation $C$ to denote choice for flip distance. $C = 1$ denotes the first model, where the sensor has only one fixed choice for flip distance (the maximum distance $F$). $C = n$ denotes the second model, where the sensor has $n$ choices for the flip distance (between $d$ and maximum distance $F$). For the rest of the chapter, unless otherwise clearly specified, the term flip distance denotes the maximum flip distance $F$.

### 2.2.2 Problem Definition

The sensor network we study is a square field. It is divided into 2-dimensional regions, where each region is a square of size $R$. A certain number of flip-based sensors are deployed initially in the network. The initial deployment may have holes that are not covered by any sensor. In this context, our problem statement is; Given a sensor network of size $D$, a desired region size $R$, an initial deployment of $N$ flip-based sensors that can flip once to a maximum distance $F$, our goal is to determine an optimal movement plan for the sensors, in order to maximize the number of regions that is covered by at least one sensor, while simultaneously minimizing the total number of flips required. The input to our problem is the initial deployment (number of sensors per region) in the sensor network, and the mobility model of sensors. The output is the detailed movement plan of the sensors across the regions (which sensors should move, and where) that can achieve our desired objectives.
Assumptions: We make the following assumptions. The region size $R$ is contingent on the application, based on sensing/transmission ranges of sensors, and application demands. We assume that \( \min\{\frac{S_{sen}}{\sqrt{2}}, \frac{S_{tr}}{\sqrt{5}}\} \geq R \), where $S_{sen}$ and $S_{tr}$ are sensing and transmission ranges of the sensors respectively. This guarantees that if a sensor is present in a region, every point in the region is covered by the sensor, and the sensor can communicate with sensors in each of its four adjacent regions. In this chapter, we first assume that the desired region size $R$ is an integral multiple of the basic unit of flip distance, i.e., $R = m \times d$, where $m$ is an integer ($\geq 1$). We discuss the general case of $R$ subsequently in Section 2.5. We assume that each sensor knows its position. To do so, sensors can be provisioned with GPS devices, or approaches in [39] can be applied, where sensors are localized using sensors themselves as landmarks. In our model, the regions to which a sensor can flip, are those in its left, right, top and bottom directions. However, those regions need not be just the adjacent neighbors. They depend on the flip distance $F$. After discussing the above case, the general case, where a sensor can flip to regions in any arbitrary direction is discussed subsequently in Section 2.5. We also assume sensors are homogeneous in sensing and transmission ranges, and they are unaffected during network operation as in [1], [2], [4], [3]. We assume a free space radio propagation model, where there exists a clear line of sight path between two communicating sensors in the network. The base-station can reside anywhere as long as it is able to communicate with the sensors.

2.2.3 An Example of our Problem

We illustrate our problem further with an example. Figure 2.1 (a) shows an instance of initial deployment in the sensor network. The shaded circles denote sensors,
and the numbers denote the id of the corresponding region in the network. The
neighbors of any region are its immediate left, right, top and bottom regions. For
instance in Figure 2.1, the neighbors of region 6 are regions 2, 5, 7 and 10. In Figure
2.1 (a) after the initial deployment, regions 1, 6, 11, 12, 16 are not covered by any
sensor and are thus holes. Optimally covering such holes is the problem we address
in this chapter.

![Figure 2.1: A snapshot of the sensor network and a movement plan to maximize
coverage (a), and the resulting deployment (b)](image)

The above problem is not easy to solve. For instance, consider Figure 2.1 (a).
For ease of elucidation, let the desired region size $R = d$. Let the flip distance
$F = d$. One intuitive approach towards maximizing coverage is to let sensors from
source regions (more than one sensor) to flip to hole regions (no sensors) in their
neighborhood, using local information around them. In Figure 2.1 (a), region 7 has 3
sensors in it, while region 11, a neighbor of 7 is empty. Similarly, region 8 has a sensor
while its neighbor, region 12 is empty. If we allow neighbors to obtain local neighbor
information, then intuitively a sensor from region 7 will attempt to cover regions 11
and 16. This intuition is because region 7 (with extra sensors) is nearest to holes 11 and 16. Similarly, region 4 will try to cover region 12. The resulting sequence of flips, and the corresponding deployment are shown in Figures 2.1 (a) and (b) respectively.

![Figure 2.1: A snapshot of the sensor network and the optimal movement plan (a), and the resulting deployment (b)](image)

With this movement plan, region 16 is still uncovered, as shown in Figure 2.1 (b). This is because, while region 7 has extra sensors, there are no mobile sensors in regions 11 and 12 (that can flip further). Also, region 15 cannot provide a sensor, without making itself (or some other region) a hole. This means that all paths to region 16 are blocked in this movement plan, preventing region 16 from being covered. However, there exists an optimal plan that can cover all regions in this case as shown in Figure 2.2. For optimal deployment, the path of movements to cover region 16 incurs a chain of flips starting from region 5 towards region 16. In fact, for optimal coverage, this plan also requires the minimum number of flips (10 flips). The key challenges we have to overcome to solve our problem are 1) the trade-offs in simultaneously attempting to optimize both coverage and number of flips and 2) the constraints arising from limited
mobility, due to which a sensor from a far away region may need to flip towards a far away hole, and a chain of flips may need to progressively occur towards the particular hole for covering it. Determining such a movement plan for optimizing both coverage and number of flips is not trivial.

2.3 Our Optimal Algorithm

2.3.1 Design Rationale

In this chapter, we propose an algorithm where information on number of sensors per region for all regions is collected by the Base-station, and an optimal movement plan for the sensors is determined and forwarded to the sensors. We propose a minimum-cost maximum-flow based algorithm that is executed by the Base-station using the region information to determine the movement plan. The Base-station will then forward the movement plan (which sensors should move and where) to corresponding sensors in the network.

Each sensor in the network will first determine its position and the region it resides in. Sensors then forward their location information to the Base-station. The packets are forwarded towards the Base-station through other neighboring regions closer to the Base-station. To do so, protocols like [40], [41], [23] can be used, where the protocols route packets towards sinks in the network (Base-station in our case) using shortest paths. Another approach that does not require a centralized node is to let individual sensors collect region information, and execute our algorithm independently to determine the movement plan. We discuss the latter approach in Section 2.3.4.

Alternatively, a region-head can be elected in each region to collect and forward information on number of sensors in their region.
We now discuss how to translate our problem into a minimum-cost maximum-flow problem. Let us denote regions with at least two sensors as sources. Source Regions can provide sensors (like region 5 in Figure 2.1 (a)), or they can be on a path between another source and a hole. Let us denote regions with only one sensor as forwarders. Forwarder regions cannot provide sensors (otherwise, they become holes themselves), but they can be on a path between a source and a hole (like regions 9, 13, 14, 15 in Figure 2.1 (a)). Let us denote regions without any sensor as holes. Obviously holes can only accept sensors (regions 1, 6, 11, 12 and 16). The first objective of our problem is to maximize the number of holes that eventually have a sensor in them. Since there can be multiple sources and multiple forwarders, in the event of maximizing the number of holes that eventually contain a sensor, there can be many possible sequences of sensor movements. Out of such possible sequences, our second objective is to find the sequence that minimizes the number of sensor movements.

If we identify regions (sources, forwarders and holes) using vertices, and incorporate path relationships in the sensor network as edges (with appropriate constrained capacities) between the vertices, then from a graph-theoretic perspective, our problem is a version of the multi-commodity maximum flow problem, where the problem is to maximize flows from multiple sources to multiple sinks in a graph, while ensuring that the capacity constraints on the edges in the graph are not violated. While obtaining the optimal plan to maximize coverage, we also want to minimize the number of flips. That is, if we associate a cost with each flip, we wish to minimize the overall cost of flips while still maximizing coverage. This problem is then a version of the minimum-cost multi-commodity maximum-flow problem, where the objective is to find paths that minimize the overall cost while still maximizing the flow. Our
approach is to model the sensor network as an appropriate graph structure (called *virtual graph*) following the objectives discussed above, determine the minimum cost maximum flow plan in the virtual graph. Each flow in the flow plan (in the virtual graph) denotes a path for a sensor movement from a source to a hole in the sensor network. As we discuss later, the maximum flow value in the virtual graph denotes the maximum number of holes into which a sensor can move without violating the mobility constraints, while the minimum cost denotes the corresponding minimum number of sensor movements (or flips). We then translate the flow plan as flip sequences in the sensor network. For the rest of the chapter, if the context is clear, we will call our algorithm as minimum-cost maximum-flow algorithm.

### 2.3.2 Constructing the Virtual Graph $G_V$ from the initial deployment

We now discuss the construction of the virtual graph in detail. The inputs are the initial deployment (with $N$ sensors), the granularity of desired coverage (region size $R$), flip distance ($F$) and the number of sensors per region $i$ ($n_i$). We denote the number of regions in the network as $Q$. Let $G_S(V_S, E_S)$ be an undirected graph representing the sensor network. Each vertex $v \in V_S$ represents one region in the sensor network and each edge $e \in E_S$ represents path relationship between regions. $G_S$ purely represents the initial network structure (and does not reflect whether regions are sources, forwarders or holes), and as such is undirected. The virtual graph (denoted by $G_V(V_V, E_V)$) is constructed from $G_S$.

The key task in constructing the virtual graph ($G_V$) is to first determine its vertices (the set $V_V$) commensurate with the status of each region as a source, forwarder or hole. Then, we have to establish the edges (the set $E_V$), directions, capacities and
costs in $G_V$ between the vertices. For any region $i$ in the sensor network, we denote its reachable regions as those to which a sensor from region $i$ can flip to. Obviously, the reachable regions depend on the flip distance $F$. In $G_V$, edges are added between such reachable regions. The directions of edges between vertices are based on whether the corresponding regions are sources, forwarders or holes in the sensor network. The capacities of the edges depend on the number of sensors in the regions, while the cost is used to quantify the number of sensor flips between regions in the sensor network. We denote $C(p, q)$ as the capacity, and $Cost(p, q)$ as the cost of the edge between vertices $p$ and $q$ in $G_V$ respectively. The final objective is to ensure that the minimum-cost maximum-flow plan in $G_V$ can be translated into an optimal movement (flip) plan for sensors in the network. In the following, we first discuss how to construct the virtual graph for a simple, yet representative basic case. We then discuss how to construct the virtual graph for general case.

**Constructing the virtual graph for the case** $R = d$: In this case, the region size $R$ is equal to the basic unit of flip distance $d$. To explain the virtual graph construction process clear, we first describe it for the case where the flip distance $F = d$, and $C = 1$. That is, the flip distance is the basic unit $d$ and the sensor has only one choice for flip distance. We discuss the case where $F > d$ and $C = 1$, and $F > d$ and $C = n$ (multiple choices) subsequently. The case where $R > d$ is also discussed subsequently.

**a) Construction when $F = d$ and $C = 1$**: In the virtual graph, each region (of size $R$) is represented by 3 vertices. For each region $i$, we create a vertex for it in $G_V$ called base vertex, denoted as $v^b_i$. The base vertex $v^b_i$ of region $i$ keeps track on the number of sensors that are in region $i$. For each region, we need to keep track
of the number of sensors from other regions that have flipped to it, and the number of sensors that have flipped from this region to other regions. The former task is accomplished by creating an in vertex, and the latter is accomplished by creating an out vertex for each region. For each vertex $i$, its in vertex in the virtual graph is denoted as $v_{i}^{in}$ and its out vertex is denoted as $v_{i}^{out}$.

![Diagram](image)

Figure 2.3: The Virtual Graph with only regions 1 and 2 in it

Having established the vertices, we now discuss how edges (and their capacities) are added between vertices in $G_V$. Recall that each region that has $\geq 2$ sensors is considered a source, and each region that has $\geq 1$ sensor is considered a forwarder. We are interested in how to optimally push sensors from such regions. For such regions, an edge is added from the corresponding $v_{i}^{b}$ to $v_{i}^{in}$ with capacity $n_i - 1$. The interpretation of this is that when attempting to determine the flow from the base vertex ($v_{i}^{b}$), at least one sensor will remain in the corresponding region $i$. Then an edge with capacity $n_i$ is added from the same $v_{i}^{in}$ to $v_{i}^{out}$. This ensures that it is possible for up to $n_i$ sensors in this region to flip from it. Recall the example in Figure 2.1 (a). Region 2 is a source. The $G_V$ construction corresponding to this
region is shown in Figure 2.3, where there is an edge with capacity \( n_2 - 1 = 1 \) from vertex \( v^b_2 \) to \( v^{in}_2 \), and an edge of capacity \( n_i = 2 \) from \( v^{in}_2 \) to \( v^{out}_2 \). Other source and forwarder regions are treated similarly in \( G_V \).

Each region that has 0 sensors is considered a hole. We are interested in how to optimally absorb sensors in such regions. For holes, an edge is added from the corresponding \( v^{in}_i \) to base vertex \( v^b_i \) with edge capacity equal to 1. This is to allow a maximum of one sensor into the base vertex \( v^b_i \) of hole region \( i \). If a sensor flips to this hole, the hole is then covered, and no other sensor needs to flip to this region. Then an edge with capacity 0 is added from the same \( v^{in}_i \) to \( v^{out}_i \). This is because a sensor that moves into a hole will be not able to flip further. Recall again from the example in Figure 2.1 (a). Region 1 is a hole. In Figure 2.3, there is an edge with capacity 1 from vertex \( v^{in}_1 \) vertex to \( v^b_1 \), and edge of capacity 0 from \( v^{in}_i \) to the \( v^{out}_i \). Other holes are treated similarly in \( G_V \). We now have,

\[
\forall v^{in}_i \text{ and } v^{out}_i \in V_V, \quad C \left( v^{in}_i, v^{out}_i \right) = n_i \quad (2.1)
\]

\[
\forall v^b_i \text{ and } v^{in}_i \in V_V \mid n_i > 0, \quad C \left( v^b_i, v^{in}_i \right) = n_i - 1 \quad (2.2)
\]

\[
\forall v^{in}_i \text{ and } v^b_i \in V_V \mid n_i = 0, \quad C \left( v^{in}_i, v^b_i \right) = 1. \quad (2.3)
\]

The final step is to incorporate the reachable relationship that holds in the original deployment field into the virtual graph. Recall that for any region \( i \) in the sensor network, its reachable regions as those regions to which a sensor from region \( i \) can flip to, which is determined by the flip distance \( (F) \). We have to incorporate this in the virtual graph. To do so, an edge of infinite capacity (denoted by \( \text{inf} \)) is added from \( v^{out}_i \) to \( v^{in}_j \), and another edge of infinite capacity is added from \( v^{out}_j \) to \( v^{in}_i \) if regions

\[\text{In practice an edge with capacity 0 need not be \textit{specifically} added. We do so to retain the symmetricity in the virtual graph construction.}\]
i and j are reachable from each other. This is to allow any number of flips between reachable regions, if there are sensors in them. In Figure 2.3, regions 1 and 2 are reachable from each other since $R = d$ and $F = d$. Thus, edges with infinite capacity are added from the $v_1^{out}$ to $v_2^{in}$, and from $v_2^{out}$ to $v_1^{in}$. Formally, for all regions $i$ and $j$ that are reachable from each other in the sensor network, we have

$$C(v_1^{out}, v_2^{in}) = C(v_2^{out}, v_1^{in}) = \infty.$$  \hfill (2.4)

Having discussed the capacity among edges, we now incorporate costs for each flow in $G_V$. If a flip occurs from some region $i$ to some region $j$ in the sensor network, we consider that a cost of one has incurred. From equation (2.4), we can see that the flips between reachable regions (say $i$ and $j$) in the sensor network is translated in $G_V$ by an edge from $v_1^{out}$ to $v_2^{in}$, and from $v_2^{out}$ to $v_1^{in}$. In order to capture the number of flips between these regions, we add a cost value to these corresponding edges in $G_V$, with cost value equal to 1. Let us denote $Cost(i, j)$ as the cost for a flip between vertices $i$ and $j$. Formally, for all regions $i$ and $j$ that are reachable from each other in the sensor network, we thus have

$$Cost(v_1^{out}, v_2^{in}) = Cost(v_2^{out}, v_1^{in}) = 1.$$  \hfill (2.5)

The $Cost$ for other edges in $G_V$ is 0. This is because, from the view of the sensor network, the edges apart from those between $out$ and $in$ vertices across regions, are internal to a region. They cannot be counted towards sensor flips (which only occurs across regions). Instances of such edges are those from $v_1^{in}$ to $v_1^{out}$, from $v_1^{in}$ to $v_1^{out}$ in Figure 2.3. An instance of original deployment and the corresponding virtual graph at the start are shown in Figures 2.4 (a) and (b) respectively. In Figure 2.4 (a) the
numbers denote the id of the corresponding region. We do not show the Cost values in the virtual graph in Figure 2.4 (b).

![Diagram](image)

Figure 2.4: The initial sensor network deployment (a) and the corresponding virtual graph at the start (b) in Case $R = d$

**b) Construction when $F > d$ and $C = 1$:** In the above, we discussed $G_V$ construction for the case where $R = d$, $F = d$ and $C = 1$. When $F > d$ and $C = 1$, only the reachability relationship changes. Specifically, immediate neighboring regions are not reachable anymore from each other like in the case where $F = d$. When $F > d$, regions beyond immediate neighboring regions become reachable (depending on $F$).

In $G_V$, edges of infinite capacity are added from $v_i^{\text{out}}$ to $v_j^{\text{in}}$, and from $v_j^{\text{out}}$ to $v_i^{\text{in}}$ if regions $i$ and $j$ have are reachable from each other. For example, if $F = 2d$ and $C = 1$ in Figure 2.4, the reachable regions for region 1 are regions 3 and 9 only. Thus edges are created from $v_1^{\text{out}}$ to $v_3^{\text{in}}$ and $v_9^{\text{in}}$, and from $v_3^{\text{out}}$ and $v_9^{\text{out}}$ to $v_1^{\text{in}}$. Other edges in
are modified similarly. The edge capacities \( C(i, j) \) and costs \( Cost(i, j) \) can be obtained following from discussions above in constructing the virtual graph for the case \( R = d \).

c) **Construction when \( F > d \) and \( C = n \):** In this case again, only the reachability relationship changes. Recall that if \( F > d \) and \( C = n \), the distance of flip can be \( d, 2d, 3d, \ldots nd \), where \( F = nd \). For example if \( F = 2d \) and \( C = 2 \) in Figure 2.4, then the reachable regions of region 1 are regions 2, 3, 5 and 9. Thus, apart from existing edges, edges are also added from \( v_1^{out} \) to \( v_3^{in} \) and \( v_5^{in} \), and from \( v_3^{out} \) and \( v_9^{out} \) to \( v_1^{in} \). Other edges, capacities and costs are modified similarly.

**Constructing the virtual graph for the case \( R > d \):** In this case, the region size \( R > d \). We first describe the virtual graph construction for where the \( R \) is an integral multiple of \( d \), i.e., \( R = x \times d \), where \( x \) is an integer \((\geq 1)\) \(^5\), \( F = d \), and \( C = 1 \). For instance, if \( x = 2 \), then the requirement is to maximize number of regions (of size \( 2d \)) with at least one sensor. Note that if \( R = x \times d \), then there are \( x^2 \) sub-regions in each region. This is shown in Figure 2.5 (a), where the region \((R = 2d)\) is the area contained within dark borders. We denote each area within the shaded lines as sub-regions. Each sub-region has a size \( d \). The id of the regions is the number in bold at the center of the region. For ease of understanding, we keep the id of the sub-regions in Figure 2.5. To explain construction process better, we say that a region \( i \) represents its sub-regions. In Figure 2.5, region 1 is a representative of sub-regions 1, 2, 5 and 6.

In the virtual graph, each region (of size \( R \)), and whose id is \( i \), has a vertex \( v_i^h \). For each region we are interested in how many sensors from other sub-regions have

\(^5\)We discuss the general case of \( R \) in Section 2.5.
flipped to it. Despite covering multiple sub-regions, we are interested in coverage of the region in itself (and not the sub-regions). Thus, we still need only one in vertex \(v_{i}^{in}\) for each region. However, each region has multiple sub-regions, and sensors in them can be pushed out. Thus the number of out vertices per region is equal to the number of sub-regions as shown in Figure 2.5 (b). Note that, sensors do not need to make internal flips to other sub-regions within a region, as there is no improvement in region coverage. Hence, there are no edges created for internal flips within a region.

We now discuss how edges are added between vertices in \(G_{V}\). Each region \(i\) that has \(\geq 1\) (or \(= 1\)) sensors, is a source region (or a forwarder region), and an edge is added from \(v_{i}^{b}\) to \(v_{i}^{in}\) with edge capacity equal \(\sum_{j=1}^{x^2} n_j - 1\) in the virtual graph, where \(x^2\) is the number of sub-regions in each region, and \(n_j\) is the number of sensors in sub-region \(j\). For example in Figure 2.5, region 1 is a source, and there is an edge with capacity \(\sum_{j=1}^{4} n_j - 1 = 3\) from vertex \(v_{1}^{b}\) to \(v_{1}^{in}\). This ensures that, when
determining the flow from this region, at least one sensor remains. Then, edges with capacity $n_k$ are added from this $v_{i}^{in}$ to each $v_{i}^{out \ k}$ as shown in Figure 2.5, where $v_{i}^{out \ k}$ is the out vertex corresponding to sub-region $k$ in region $i$. For each hole $j$, we add an edge with capacity 1 from $v_{j}^{in}$ vertex to $v_{j}^{b}$, an edge of 0 capacity from $v_{j}^{in}$ to each $v_{j}^{out \ k}$ in the virtual graph. Finally, to incorporate the reachability relationship between regions, an edge of infinite capacity is added from $v_{i}^{out \ m}$ to $v_{j}^{in}$, and an edge of infinite capacity is added from $v_{j}^{out \ p}$ to $v_{i}^{in}$ if regions $i$ and $j$ are reachable from each other and sub-region $m$ is reachable from region $j$ and sub-region $p$ is reachable from region $i$ as shown in Figure 2.5. The cost, that captures number of flips between regions is 1 between the corresponding edges above. Other edges that are between regions also have cost value 1. For each region $i$, the edges from $v_{i}^{b}$ to $v_{i}^{in}$, and from $v_{i}^{in}$ to $v_{i}^{out \ k}$ (for all $k$) do not count towards sensor flips and have 0 cost (similar to the preceding case). We can obtain equations for capacities and costs following from discussions above in constructing the virtual graph for the case $R = d$. An instance of original deployment and the corresponding virtual graph at the start are shown in Figures 2.5 (a) and (b) respectively. We do not show the cost values in the virtual graph in Figure 2.5 (b).

The extensions to construct $G_V$ when $F > d$ and $C = 1$, and when $F > d$ and $C = n$ for the case where $R > d$ are similar to those proposed for $F > d$ and $C = n$, and $F > d$ and $C = 1$ respectively for the case where $R = d$. Due to space limitations, we do not describe the construction for these cases.
2.3.3 Determining the optimal movement plan from the Virtual Graph

Recall that the base vertex \( (v^b_i) \) keeps track of the number of sensors in region \( i \) in \( G_V \). Also, the edges going into the base vertices of holes have capacity one to allow a maximum of one sensor into the holes. Consequently, our problem can be translated as determining flows from the base vertices of source regions to as many base vertices of holes as possible in \( G_V \), with minimum overall cost. Let us now discuss why this is true. From the construction rules of \( G_V \), we can see that for each feasible sensor movement sequence in the sensor network between a source region and a hole, there is a feasible path for a flow in \( G_V \) between base vertices of the corresponding regions and vice-versa. For example, consider a feasible sensor movement sequence from a source region (say \( i \)) to a hole (say \( j \)) through forwarder regions \( k, l, \ldots, m \) and \( n \). We denote the path as a tuple of the form \( < i, k, l, \ldots, m, n, j > \). The feasibility of this path means that each of regions \( i, k, l, \ldots, m, n \) have at least one sensor in them, and region \( i \) has at least two sensors. From the construction of \( G_V \), the capacities \( C(v^b_i, v^m_i), C(v^m_i, v^o_i), C(v^o_i, v^k_i), C(v^m_k, v^o_k), C(v^o_k, v^m_l), \ldots, C(v^m_m, v^o_n), C(v^o_n, v^j_n) \), and \( C(v^o_j, v^b_j) \) are all \( \geq 1 \) (where \( C(i, j) \) was defined above for the case of constructing the virtual graph when \( R = d \). Thus a flow from \( v^b_i \) to \( v^b_j \) is feasible in \( G_V \). The cost of this flow in is the summation of cost of all edges in the path. Recall that the cost of edges from \( out \) to \( in \) of reachable regions is one, and all other edges costs are zero. Consequently, the cost of the above flow in \( G_V \) is the number of times a flow occurs between \( out \) and \( in \) vertices of successive reachable regions in the path. Clearly, this is the number of regions traversed in the sensor network.

\[ \text{A similar argument can show that for every feasible flow in } G_V, \text{ a corresponding sensor movement sequence is feasible in the sensor network.} \]
network (i.e., the number of flips). At this point it is clear that if we can determine a flow plan (the actual flow among the edges) in $G_V$ that can maximize the flow from the base vertices of source regions to as many base vertices of holes with minimum overall cost, we can translate the flow plan as a movement plan for sensors (across the regions) in the sensor network that maximizes coverage and minimizes the number of sensor flips.

Determining the minimum cost maximum flow plan in a graph is a two step process. First, the maximum flow value from sources to sinks in the graph is determined (for which there are many existing algorithms). Second, the minimum cost flow plan (for this maximum flow) in the graph is determined (for which also there are many existing algorithms). In our implementations, we first determine the maximum flow value in $G_V$ from all base vertices of source regions to base vertices of hole regions using the Edmonds-Karp algorithm [42]. We then determine the minimum cost flow plan (for the above maximum flow) in $G_V$ using the method in [43], which is an implementation of the algorithm in [44]. For more details on the algorithms, readers can refer to [42], [43] and [44]. The corresponding flow plan is a set of flows in all edges in $G_V$ corresponding to the minimum cost maximum flow in $G_V$.

Let $W^V$ denote the flow plan (a set of flows) corresponding to the minimum cost maximum flow in $G_V$, where the amount of each flow is one. Each flow in $W^V$ is actually a path from the base vertex of a source to the base vertex of a hole. The flow value is one for each flow, since only one sensor eventually moves to a hole (from our problem definition). Consequently, the value of the maximum flow is the number of such flows (with flow value one), which in turn is the maximum number of holes that can be covered eventually with one sensor. Each flow $w^V_{i,j} \in W^V$ is a flow from
the base vertex of a source \( v^b_i \) to the base vertex of a hole \( v^b_j \) in \( G_V \), and is of the form \( \langle v^b_i, v^{\text{in}}_i, v^{\text{out}}_i, v^{\text{in}}_k, v^{\text{out}}_k, \ldots, v^{\text{out}}_m, v^{\text{in}}_n, v^{\text{out}}_j, v^b_j \rangle \), which denotes that the path of the flow is from \( v^b_i \) to \( v^{\text{in}}_i \), from \( v^{\text{in}}_i \) to \( v^{\text{out}}_i \), ..., from \( v^{\text{in}}_j \) to \( v^b_j \). From the construction of \( G_V \), for each such \( w^V_{i,j} \) (\( \in W^V \)), a corresponding movement sequence in the sensor network \( w^S_{i,j} \) can be determined, and is of the form \( \langle r_i, r_k, r_l \ldots r_m, r_n, r_j \rangle \), where \( r_i, r_k, r_l \ldots r_m, r_n, r_j \) correspond to regions \( i, k, l, \ldots, m, n, j \) in the sensor network respectively. Physically, this means that one sensor should flip from regions \( i \) to \( k \), \( k \) to \( l \), \( \ldots \) \( m, n \) to \( n \) and \( n \) to \( j \). The sensor flip (or movement) plan \( W^S \) (set of all \( w^S_{i,j} \)) is the output of our solution.

Once the Base-station determines the flip plan, it will forward instructions to the sensors (that need to flip). For each sensor, the Base-station can forward instructions on the reverse direction of the original path of communication between the sensor and the Base-station. The forwarded packet contains the destination of the sensor and the intended region the sensor needs to flip to. Since sensors know the regions they reside in, they can determine the direction of the intended region (i.e., left, right, top or bottom region). We assume that sensors are equipped with steering mechanisms (similar to the one in [29]) that allow sensors to orient themselves in an appropriate direction prior to their flip. Theorem 1 shows that the flip plan obtained by our algorithm optimizes both coverage and the number of flips.

**Theorem 1.** Let \( W^V_{\text{opt}} \) be the minimum-cost maximum-flow plan in \( G_V \). Its corresponding flip plan \( W^S_{\text{opt}} \) will maximize coverage and minimize the number of flips.

**Proof.** We first prove that our algorithm optimizes coverage. We prove by contradiction. Consider a flip plan \( W^S_{\text{opt}} \) in the sensor network corresponding to a flow plan
$W^V_{opt}$ in $G_V$ obtained by our solution. Let $W^S_{opt}$ be non-optimal in terms of coverage. This means there is a better flip plan, $W^S_x$ that can cover at least one extra region in the sensor network. Clearly, a corresponding flow plan $W^V_x$ can be found in $G_V$. The amount of flow in $W^V_x$ is larger than the maximum flow in $W^V_{opt}$. This is a contradiction.

We now prove that our algorithm optimizes the number of flips. We prove by contradiction. Consider a flip plan $W^S_{opt}$ in the sensor network corresponding to a flow plan $W^V_{opt}$ in $G_V$ obtained by our solution. Let $W^S_{opt}$ be non-optimal in terms of the number of flips. This means there is a better plan, $W^S_x$ that can reduce at least one flip in the sensor network. Clearly, a corresponding flow plan $W^V_x$ can be found in $G_V$. The number of flips in $W^V_x$ is less than that in $W^V_{opt}$. This is a contradiction.

We now discuss the time complexity of our solution. There are three phases in our algorithm while determining the optimal movement plan. The first is the virtual graph construction, the second is determining the maximum flow, and the third is the execution of the minimum-cost flow algorithm. Denoting $|V|$ and $|E|$ as the number of vertices and edges in the virtual graph respectively, we have $|V| = O((\lceil \frac{D}{R} \rceil)^2 (\lceil \frac{R}{d} \rceil)^2)$ and $|E| = O((\lceil \frac{E}{d} \rceil)(\lceil \frac{R}{d} \rceil)(\lceil \frac{L}{R} \rceil)^2)$, where $\lceil \frac{P}{R} \rceil^2$ denotes the number of regions, $\lceil \frac{R}{d} \rceil^2$ denotes the number of sub-regions and $\lceil \frac{E}{d} \rceil$ denotes the number of reachable regions for each region. The time complexity of the virtual graph construction is $O(|V| + |E|)$. The time complexity for determining the maximum flow using the implementation in [42] is $O(|V||E|^2)$, and time complexity for determining the minimum cost flow using the implementation in [44] is $O(|V|^2|E|\log|V|)$. As such, the resulting time complexity of our algorithm is $O(max \ (|V||E|^2, |V|^2|E|\log|V|))$. We wish to emphasize here that the above implementations are not necessarily the fastest. For a detailed survey of
other works on the maximum flow and minimum cost problems, please refer to [45] and [46].

2.3.4 Alternate Approaches to Execute our Algorithm

Our algorithm requires information on the number of sensors in each region in the network. In the above, we proposed to let a centralized Base station to collect this information and execute our solution. We now discuss distributed approaches to execute our solution. In the first approach, sensors in the network once again share information about the number of sensors in their regions. In the extreme case, a sensor in each region can execute our algorithm independently with this information. An alternate distributed approach is to divide the network into multiple domains. In this approach, we let each domain to obtain region information only in their domain and not exchange it with other domains. A special sensor in each domain can then execute our algorithm independently only with this information (without global synchronization with other domains), and determine a movement plan for sensors in its domain. We call this the domain-based approach. However, this approach can only achieve local optima in each domain and cannot guarantee global optima in the sensor network. We will study performance of this approach further using simulations in Section 2.4.

2.4 Performance Evaluations

In the above, we proved the optimality of our algorithm in terms of coverage and number of flips. We now study the sensitivity of coverage and the number of flips to flip distance under different choices (for flip distance), initial deployment scenarios and
coverage requirements. We also study performance and overhead when our algorithm is executed using the domain-based approach discussed in Section 2.3.4.

2.4.1 Performance Metrics and Evaluation Environment

Let the total number of regions in the network be $Q$. We denote $Q_i$ as the number of regions with at least one sensor at initial deployment, and denote $Q_o$ as the number of regions with at least one sensor after the movement plan determined by our algorithm is executed. Our first metric is the Coverage Improvement $CI$. Since, we already proved the optimality of final deployment, we want to study here the improvements in coverage as a result of executing our algorithm compared to initial deployment. Formally, $CI = Q_o - Q_i$. We define the Flip Demand as the number of flips required per region increase in coverage. The Flip Demand quantifies the efficiency of flips in improving coverage. Denoting $J$ as the optimal number of flips as determined by our algorithm, we have $FD = \frac{J}{Q_o - Q_i}$. In order to compare the overhead of our optimal and domain-based approach, we define the metric packet number $PN$ incurred by the algorithms. This metric is defined subsequently when we actually compare the two algorithms.

We conduct the following simulations on two network sizes, $300 \times 300$ units and $150 \times 150$ units. The region sizes are $R = 10$ and $R = 20$ units. The basic unit of flip distance $d = 10$ units. We vary the flip distance $F$ from 10 units to 40 units. The choices are $C = 1$ and $C = n$. Recall that if $F$ is say, 40 and $C = 1$, the flip distance for the sensors is fixed as 40 units. When $C = n$, we can have flip distances between 0 and $F$ in discrete steps of the basic unit of flip distance $d$ (= 10 units). Thus if $F = 40 \ (4d)$, we have $C = 4$, and in this case, sensors can flip to distances 10, 20, 30,
40 units. The number of sensors deployed is equal to the number of regions. All data reported here were collected across 10 iterations, and averaged. Our implementations of the maximum flow algorithm is the Edmonds-Karp algorithm [42], and minimum cost flow algorithm is the one in [44]. We conduct our simulations using MATLAB. We use a topology generator for 2D-Normal distribution. The $X$ and $Y$ co-ordinates are independent of each other (i.e., $\sigma_x = \sigma_y$). We use $\sigma = \frac{1}{\sigma^2}$ to denote the degree of concentration of deployment in the center of the network field. Thus larger values for $\sigma$ implies more concentrated deployment in the center of the field. When $\sigma = 0$, the deployment is uniform.

2.4.2 Performance Results

a). Sensitivity of $CI$ and $FD$ to $F$ under different $C$: Figures 2.6 (a) and (b) show the sensitivity of $CI$ to flip distance $F$ under different choices $C$ in two different network sizes ($150 \times 150$ and $300 \times 300$), where $R = 10$, $\sigma = 1$ and
$d = 10$. The number of regions in the networks are $15 \times 15 = 225$ and $30 \times 30 = 900$ respectively. We observe that in both figures, for a given value of $F$, $C = n$ has larger $CI$ compared to $C = 1$, except for $F = d = 10$, when both $C = n$ and $C = 1$ have the same performance. This is because, when $F = 10$, we have $n = 1$, and so $C = n$ is the same as $C = 1$. However, when $F > 10$, for a given $F$ there are more flip choices that our algorithm can exploit when $C = n$, hence increasing $CI$. The second observation we make is that when $C = n$, as $F$ increases $CI$ increases in both figures. This is also because when $C = n$, an increase in $F$ means more choices to exploit. However, the trend is different when $C = 1$. In general, a large flip distance (even when $C = 1$) may appear to be perform better as more far away holes can be filled if $F$ increases. However, when $C = 1$, such improvements depend on the size of the network. Beyond a certain point (depending on the network size), an increase in $F$ becomes counter-productive. This is due to two reasons. First many sensors near the

Figure 2.7: Sensitivity of $FD$ to $F$ under different $C$ in $150 \times 150$ and $300 \times 300$ network
borders of the sensor network have flips that cannot be exploited when $F$ is too large. Secondly, the chances of sequential flips (i.e., sensor $x$ flipping to sensor $y$’s region, sensor $y$ flipping to sensor $z$’s region and so on) to cover holes are reduced when $F$ is too large. The value of $F$, where this shift takes place in the $150 \times 150$ network in Figure 2.6 (a) is $F = 40$, where $CI$ decreases. Such a shift is not observed for the $300 \times 300$ network in Figure 2.6 (b), as the network size is quite large.

Figures 2.7 (a) and (b) show the sensitivity of flip demand $FD$ to flip distance $F$ under different choices $C$ in two different network sizes. We wish to emphasize here that the number of flips does not linearly increase with coverage. Consequently, to enable a fairer comparison, we compare $FD$ across different cases when the final coverage is the same. For both network sizes, we set $R = 10$ and $\sigma = 0$, where the final deployment covers all regions. Since the initial distribution is the same, the coverage improvement is the same. The comparison becomes more meaningful. From Figures 2.7 (a) and (b), we see that for a given value of $F$, $C = n$ has a lower $FD$ than $C = 1$, except for $F = d = 10$, when both $C = n$ and $C = 1$ have the same performance. This observation is consistent with our earlier observations on $CI$. The second observation we make is that as $F$ increases, $FD$ decreases irrespective of $C$ in both figures. When $F$ is small, in order to achieve optimality, there may be multiple flips from sensors farther away from a hole (although the number of flips is still optimum). As $F$ increases, it is likely that far away sensors can flip to such holes directly, minimizing the number of flips. By comparing Figures 2.7 (a) and (b), we observe that $FD$ is more for the $300 \times 300$ network compared to the $150 \times 150$ network. For a larger network with more regions, more flips have to be made. Consequently $FD$ is larger.
Figure 2.8: Sensitivity of $CI$ to $F$ under different $R$

Figure 2.9: Sensitivity of $CI$ to $F$ under different $\sigma$
b). Sensitivity of CI to F under different R and σ: Figure 2.8 shows how the flip distance F impacts coverage improvement (CI), under different region sizes (R). Here σ = 1 and C = n. In order to study the sensitivity of CI to region size fairly, the number of regions for different regions sizes should be the same. In Figure 2.8, for 150 × 150 and 300 × 300 network we set the region sizes as R = 10 and R = 20 respectively. The number of regions in both cases is 15 × 15 = 225. We observe that when flip distance (F) increases, CI is consistently better irrespective of R. When F increases, our algorithm can exploit more choices (C = n) and CI increases. The second observation is that, as R increases, CI decreases. This is because, when R is small, neighboring regions are closer to each other (in terms of distance between the centers of the regions). For the same F, our algorithm is more likely to find sensors from other regions that can flip to fill holes. However, when R is large for the same F, the sensors that can flip from one region to another have to be relatively close to the borders of the regions. Thus, the number of sensors that can be found to flip are less. Naturally CI (which captures improvement) decreases when R is large. Thus, performance improvement due to increases in flip distance is constrained by the desired region size.

Figure 2.9 shows how the flip distance F impacts CI under different distributions in initial deployment. The network size is 150 × 150, R = 10, and C = n. We vary σ from 0 (uniform distribution) to 4 (highly concentrated at the center of the field). The first observation we make here is that increases in flip distance (F) increases CI. However, the degree of increase in CI is impacted by σ. When σ = 0 (uniform), CI is almost the same for all values of F. This is because, in our simulations, close to
full coverage is achieved when $\sigma = 0$. Since the initial deployment is the same for all cases, the improvement is the same.

We now study the trade-off between increasing $F$ and $\sigma$ in terms of which parameter has a more dominating effect on $CI$. In Figure 2.9, when $F = 10$, $\sigma$ dominates over $F$. We can see that as $\sigma$ increases (bias increases), $CI$ decreases. The increase in bias cannot be compensated using sensors with flip distance of only 10 units. However, when $F$ increases, our algorithm can exploit more choices ($C = n$). Thus $F$ dominates when it increases. However, the degree of domination still depends on $\sigma$.

When $F = 20$, $\sigma = 1$ performs better than $\sigma = 0$. This is because, the increase in bias can be compensated better when $F = 20$ (than was the case, when $F = 10$). Thus, $CI$ increases. However, increasing $\sigma$ beyond this point makes the bias dominate and consequently $CI$ decreases. When $F > 20$, the increase in $F$ consistently dominates the increase in bias (although the degree of domination is different), showing that performance improvement due to increases in flip distance is limited by initial deployment distribution.

c). Sensitivity of $CI$ and $PN$ to domain size: Recall from Section 2.3.4 that an alternate approach to execute our algorithm is to divide the entire network into smaller domains and determine a movement plan in each domain independently. We study this approach under two network sizes: $150 \times 150$ and $300 \times 300$. We set $R = 10$, $F = 20$, $C = n$ and $\sigma = 1$ for both cases. Thus, the number of regions are 225 and 900 respectively. We divide the networks into multiple domains. We denote the domain size $A$ as the number of regions along one dimension in each domain.\footnote{In both network sizes, for optimal solutions, the domain size should be 15 and 30 respectively.}
Figure 2.10: Sensitivity of CI to A in 150 × 150 and 300 × 300 network

Figure 2.11: Sensitivity of PN to A in 150 × 150 and 300 × 300 network
We introduce a new metric here called packet number per region $PN$. Denoting $P$ as the total number of packets (or messages) sent, and $Q$ as the number of regions, we have $PN = \frac{P}{Q}$. The packet number quantifies the overhead incurred by the approaches. The packet number is calculated based on a simple protocol. After initial deployment, an elected region-head in each region sends a packet to Base-station (located in the center of the network) with information on the number of sensors in its region. The packets are forwarded along shortest paths through other regions. After the Base-station receives all packets and determines a movement plan, it sends a packet to each region in the reverse path, informing sensors of their movement plan. A similar protocol is assumed for the domain-based approach, where the region-head in each domain will forward region information to a special sensor in the domain, which executes the algorithm and forwards a movement plan to each region in the domain. Note that, there can be other versions of the above protocols, like direct relaying of messages, row-wise (or column wise) message delivery etc.

For the $150 \times 150$ network, we study $CI$ and $PN$ under three domain sizes namely, 15, 5 and 3. For the $300 \times 300$ network, we study $CI$ and $PN$ under four domain sizes namely, 30, 15, 10 and 5. From figures 2.10 (a) and (b), we can see that $CI$ decreases as $A$ decreases. When individual domains execute our algorithm independently, the domains can only achieve locally optimum solutions. However, we observe that when the domain size is half of the network size (i.e., $A = 15$ in Figure 2.10 (b)), $CI$ is close to the globally optimum case. This is because of the exploitation of the bias in initial deployment. Since sensors are initially deployed one time targeted at the center of the network, the sensors are uniformly balanced in all directions surrounding the center. In this case, if we choose the domain size as half the network size (resulting
in four domains), we are optimizing deployment independently in the four directions from the center of the network. Since the sensors are uniformly balanced in all the four directions, the ensuing $CI$ is not far from optimal (when $A = 15$ in Figure 2.10 (b)). Figures 2.11 (a) and (b) show the packet numbers for the optimal algorithm and the domain-based approach. We can see that $PN$ decreases as $A$ decreases, demonstrating the savings in overhead in the domain-based approach.

2.5 Discussions

2.5.1 Extensions to construct the Virtual Graph for any Region size $R$

In the above, we discussed the construction of the virtual graph when $R$ was an integral multiple of $d$. In the following, we discuss the construction for any general value of $R$. Without loss of generality, let $R = sd+xd$, where $s$ is an integer ($\geq 0$) and $x$ is a real number ($1.0 \geq x \geq 0$). In preceding cases, $x$ was either 1.0 ($R = (s + 1)d$) or 0 ($R = sd$). Let us now consider the case when ($1.0 > x > 0$). For two adjacent regions (say regions $i$ and $j$), we cannot determine whether a sensor in region $i$ can flip to region $j$ in case $1.0 > x > 0$. To circumvent this problem, we leverage the concept of sub-regions. For each region of size $R$, we add a certain number of sub-regions of same size that meets the following condition; the size of each sub-region should be a factor of $d$, and a factor of $R$. In this situation, we can correctly determine if a sensor in a particular sub-region can or cannot flip to another region. This can be done by traversing an integral number of sub-regions depending on $d$ and the size of sub-regions. Our above solution is optimal if $x$ is a terminating decimal. If $x$ is non-terminating (e.g. $x = \frac{1}{3}, \frac{2}{3}$ etc.), we can choose an approximate sub-region size,
such that the number of sub-regions is an integral multiple of $R$. The smaller the size of the sub-region, smaller is the error from optimality in this case.

2.5.2 Deployment under hostile zones/ failures in Sensor Networks

In some cases, there may be certain hostile zones in the network (lakes, fires, etc.) that can destroy sensors. To avoid sensor flips to such zones, we only have to modify the edges and their capacities to such hostile zones in the virtual graph. The resulting solution is optimal.

In some cases, there can be faults/ failures in the sensors and their communication. For example, if a sensor makes an erroneous movement to a region other than the intended region, there will be an extra hole in the network. Or if a packet from a region does not reach the Base-station, the region will be incorrectly treated as a hole, which may result in extra sensors in that region. As such, our algorithm can tolerate a degree of faults/ failures in the network at a cost of optimality. A rigorous study of deployment under faults/ failures will be part of future work.

2.5.3 Arbitrary Sensor Movement Directions and Network Partitions

Our algorithm can be extended to handle situations where a sensor can flip to regions in arbitrary directions apart from left, right, top and bottom directions. The reachability relationship between regions changes under arbitrary flip directions. In $G_V$, we have to add edges from each region to all newly reachable regions from it, corresponding to arbitrary directions of sensor flips.
In some situations the network may be partitioned, and we many need to repair them. In the approach proposed by Wu and Wang [2], empty holes are filled by placing a *seed* from a non-empty region to a hole. The algorithms to place seeds are tuned to meet load balancing objectives. We can apply the algorithms in [2] to repair partitions in our case. Once seeds are placed, our proposed algorithm can be executed. The key issue is the optimality of final coverage and the number of flips, given the mobility limitations of sensors. Developing optimal algorithms for partition recovery problem using flip-based sensors is a part of our on-going work.

2.6 Summary

In this chapter, we studied sensor network deployment for 1-coverage using flip-based sensors. We proposed a minimum-cost maximum-flow based algorithm to optimize coverage and the number of sensor flips. We also proposed multiple approaches to execute our algorithm in practice. Our performance data demonstrated that while increased flip-distances achieves better coverage improvement, and reduces the number of flips required per region increase in coverage, such improvements are constrained by initial deployment distributions of sensors, due to the limitations on sensor mobility.
CHAPTER 3

MOBILITY ALGORITHMS DESIGN FOR $\bar{k}$-COVERAGE IN WIRELESS SENSOR NETWORKS

In this chapter, we study the issue of deploying wireless sensor networks for $\bar{k}$-coverage ($\bar{k} \geq 1$) using limited mobility flip-based sensors. Specifically, we define a representative sensor networks deployment problem for $\bar{k}$-coverage, and design both optimum centralized, and distributed heuristic algorithms for the deployment problem. Extensive analysis and simulations are also presented to validate the performance of the proposed algorithms.

3.1 Motivations

In this chapter, we extend our sensor networks deployment work in Chapter 2 across several respects. Specifically, in this chapter, we address a general sensor networks deployment problem in clustered sensor networks with flip sensors (similar to the model in Figure 1.1 (a) in Section 1.2.1). The deployment objective is for each region to have a certain number of sensors (denoted by $\bar{k}$) that is application decided. We also consider a more relaxed mobility model, where each sensor can hop (or flip) multiple times (although the number of flips and distance per flip is bounded). In this scenario, our problem statement is; Given a deployment of flip sensors in the network,
the objective is to determine a sequence of sensor movements in order to minimize
the variance in the number of sensors from $\bar{k}$ among all regions in the network, and
simultaneously minimize the total number of sensor movements.

In many sensor networks today, multiple sensors per region are clearly important. The desired number of sensors ($\bar{k}$) per region can be contingent on one or more factors including sensing, fault tolerance, resilience to attacks, lifetime etc. For instance, due to physical limitations on sensors and associated hardware, the sampling rate at which sensors can sense the environment may be limited (e.g., 100 kHz for acoustic sensors [27], and 4200 Hz for magnetometers [28]). Thus in applications where the environment needs to be sensed quite frequently, or at all times of operation (e.g., intruder tracking, military surveillance etc.), multiple sensors per region have to be deployed to meet sensing objectives. Secondly, there may be obstacles in the regions or external factors (like heat, vibration etc.) that affect sensing ranges during network operation. Such sensing dynamics can be compensated with multiple sensors per region. Furthermore, fault tolerance, resilience to attacks improve with multiple sensors; lifetime can be prolonged using role rotation among multiple sensors etc. Hence, multiple sensors per region provide many benefits to sensor networks. However, the desired $\bar{k}$ sensors per region requirement may not be always satisfied. For example, if sensors are randomly deployed (sprayed from a vehicle, airdropped etc.), the requirement is hard to satisfy. Even if at initial deployment all regions have $\bar{k}$ sensors, as time goes on, faults, failures, energy losses etc. can violate this requirement. In such cases, the limited mobility sensors have to self-adjust their positions to correct such violations.
**Our Contributions:** In this chapter, we design a set of movement algorithms for our deployment problem that can be executed by limited mobility flip-based sensors. Our contributions are:

*A methodology for translating our non-linear optimization problem:* Our first contribution is a weight-based methodology that translates our non-linear variance objective into a linear one. We propose a weight assignment rule for regions depending on \( \bar{k} \), so that when sensors move, larger weights regions are given higher priority to balance sensor movements among all regions. We then define a new linear objective function called *Score* that captures weighted sensor movements, and prove that maximizing the *Score* minimizes the deployment *Variance* and vice versa. The number of sensor movements is minimized by treating each movement as a cost, and minimizing overall costs during *Score* maximization.

*The Optimal maximum flow based centralized algorithm:* Our first algorithm is the Optimal Maximum Flow based (OMF) centralized algorithm. Here, the sensor network at initial deployment is translated into a graph \((G_V)\). Vertices in \(G_V\) represent regions, and are assigned appropriate weights based on the above methodology. Edges represent movement ability between regions, and are assigned corresponding capacities and costs. We first show how the *minimum cost maximum weighted flow plan* in \(G_V\) maximizes the *Score* with minimum cost. We then show how this flow plan can be translated as a sensor movement plan that minimizes deployment *Variance* and sensor movements in the network. Note that the *maximum weighted* flow problem is similar to the maximum flow problem except that each target (vertex) has a weight, and the objective is to maximize the summation of the flow amount to each
target multiplied with the target weight. The maximum flow problem is its special
case, where the weight of each target is one.

_The Simple Peak-Pit based distributed algorithm:_ We then propose a local, light-
weight and purely distributed Simple Peak-Pit based (SPP) algorithm that is executed
by sensors themselves. Regions needing sensors send local requests containing _weights_
based on number of sensors needed. Surplus regions that receive the requests will
serve them in a descending order of weights, along with minimizing sensor movements
in serving them. As discussed subsequently, path feasibility (to guarantee unbroken
chain of movements) is ensured in our algorithms before sensors make real movements.

_Theoretical analysis and Performance evaluations:_ We conduct a detailed theo-
retical analysis and performance evaluations of our algorithms. We formally prove
the optimality of our _OMF_ algorithm in minimizing variance and number of sensor
movements, and derive its complexity. We then conduct extensive simulations to
evaluate the performance of our algorithms. For comprehensiveness, we also sim-
ulate the well known Virtual Force algorithm [1] (discussed briefly in Section 2.1).
In general the _OMF_ algorithm (being optimal), achieves best variance and sensor
movement minimization. However, under certain scenarios (small \( \bar{k} \), uniform initial
deployment), performance of the _SPP_ algorithm is close to the _OMF_ algorithm. We
also study communication overhead in our algorithms. We observe that the overhead
in the _SPP_ algorithm is generally lower. However, when initial deployment is highly
concentrated, the overhead in the _OMF_ algorithm is quite close to the _SPP_ algorithm,
while being smaller in some cases. Finally, we observe that all our algorithms have
better performance than the virtual force algorithm, with less overhead. As pointed
before in Section 2.1, this is due to many back and forth sensor movements in the virtual force algorithm resulting in rapid expiration of sensor mobility capacity.

We wish to point out that, our contributions in this chapter are significantly different from that in Chapter 2. In Chapter 2, our problem is maximizing number of regions in the network with at least one sensor, where the sensors were can hop only once to a fixed distance. The problem we address in this chapter is minimizing variance in number of sensors among all regions from an arbitrary integer \( \tilde{k} \). Clearly, variance is a non-linear objective. We hence need newer methodologies and algorithms to solve this problem. Also, we have a more general limited mobility model in this chapter, where a sensor can flip multiple times. As we discuss subsequently, we also present a distributed heuristic sensor mobility algorithm in this chapter for our problem, unlike in Chapter 2.

### 3.2 The Sensor Network Deployment Problem

#### 3.2.1 Problem Definition

Our sensor network is a square field of size \( Q \). It is clustered into 2-dimensional square regions, where each region is of size \( R \). The number of regions is denoted as \( S \) \((S = (\frac{Q}{R})^2)\). We denote the number of sensors in region \( i \) at time of initial deployment as \( n_i \). The deployment objective is for each region to have a certain number of sensors, denoted by \( \bar{k} \). At the time of initial deployment, not all regions will have \( \bar{k} \) sensors. The sensors deployed are limitedly mobile (similar to the model in Figure 1.1 (a) in Section 1.2.1). If a sensor moves from one region to any of its adjacent neighboring regions, we consider that as one hop (or flip) made by the sensor. We denote \( H \) as the maximum number of such hops a sensor is capable of. In this context, our
The problem statement is; Given a sensor network with $S$ regions each of size $R$, an initial deployment of $N$ limited mobility flip-based sensors, we want to determine a sequence of sensor movements so that 1) at the conclusion of movements, the variance in the number of sensors from $k$ among all the regions in the network with less than $k$ sensors is minimized, and 2) the overall number of hops of the limited mobility sensors is also minimized. Denoting $k_i$ as the number of sensors in a region $i$ at the conclusion of sensor movements, the variance $Var$ is,

$$Var = \frac{1}{S} \sum_{i=1}^{S} (\bar{k} - \min(k_i, \bar{k}))^2.$$  \hspace{1cm} (3.1)

Denoting $h_i$ as the number of hops made by sensor $i$ (where, $h_i \leq H$), and denoting $N$ as the number of sensors initially deployed, the overall number of sensors movement hops is,

$$M = \sum_{i=1}^{N} h_i.$$  \hspace{1cm} (3.2)

Our problem is to simultaneously minimize two objectives, namely $Var$ (a non-linear function) and $M$.

Problem Features: Our problem is general, since we place no restriction on $\bar{k}$. If $\bar{k} = 1$, then the requirement is one sensor per region. To enhance reliability, $\bar{k}$ can be set larger than 1. Also, it is not necessary that $\bar{k}$ is same for all regions. In non-uniform environments, some regions may need more sensors than others, meaning $\bar{k}$ is different for different regions. The Variance definition still holds, except that $\bar{k}$ in equation (3.1) becomes $\bar{k}_i$ for region $i$. Our problem is also not contingent on the
number of mobile sensors. It holds even when only a part of sensors in the network are mobile \(^8\).

An important feature of our problem is that we do not minimize the variance in number of sensors among all regions from \(\bar{k}\). We minimize it among only the regions that have less than \(\bar{k}\) sensors at final deployment, which is captured by the term \(\min(k_i, \bar{k})\) in equation (3.1). In many cases, sensors are over-deployed. When the deployment objective is only \(\bar{k}\) sensors per region, the nature of our problem will not let extra sensors move, when the requirement of at least \(\bar{k}\) sensors among all regions has been met. This is to preserve the mobility of sensors in such cases. Eventually, when some sensors fail (due to faults, power losses etc), the deficiency in \(\bar{k}\) requirement can be met by the spare sensors whose limited mobility was initially preserved, effectively complementing the motivations for over-deployment.

Assumptions: We make the following assumptions. We assume that \(\min\left\{\frac{S_{sen}}{\sqrt{2}}, \frac{S_{tr}}{\sqrt{5}}\right\} \geq R\), where \(R\) is the region size, and \(S_{sen}\) and \(S_{tr}\) are a sensor’s sensing and transmission ranges respectively. If each region has \(\bar{k}\) sensors at final deployment, then \(\frac{S_{sen}}{\sqrt{2}} \geq R\) means every point in each region is covered by \(\bar{k}\) sensors, and \(\frac{S_{tr}}{\sqrt{5}} \geq R\) means a sensor in any region can communicate with \(\bar{k}\) sensors in each of its four adjacent regions.

We assume that each sensor knows which region it resides in. To do so, sensors can be provisioned with GPS devices, or methods in [39] can be used, where sensor location are determined using sensors themselves as landmarks. For simplicity, we first assume that the regions to which a sensor can move to, are regions in its adjacent left, right, top and bottom directions only (denoted as neighboring regions). After

\(^8\)In the following, we discuss solutions for the basic problem first. Extensions to other problems are discussed later.
discussing this case, the general case where a sensor can move in any arbitrary direction is discussed next. Also, we first assume that the network is not partitioned. The issue of partitions is discussed later. We also assume sensors are homogeneous in sensing and transmission ranges, and they are unaffected during network operation as in [1], [2], [4], [3]. We assume a free space radio propagation model, where there exists a clear line of sight path between two communicating sensors in the network. The base-station can reside anywhere as long as it is able to communicate with the sensors.

3.2.2 An Example of our Problem and Challenges

We illustrate our problem further with an example. Consider an instance of initial deployment in the network, shown in Figure 3.1 (a). The number inside circles denotes the number of sensors in that region. The number in the upper left corner denotes the corresponding region ID. Let maximum number of hops $H = 1$, and $\bar{k} = 2$. There are 32 sensors initially deployed. At time of initial deployment, regions 2, 3, 9, 10, 11, 14, 15 have less than $\bar{k}$ sensors. An intuitive way to minimize the variance from $\bar{k}$ is to let neighboring regions locally synchronize for movement. Using local information exchanges, it is likely that the sensors move according to the sequence shown in Figure 3.1 (a). The arrows indicate direction of movement, and the numbers beside arrows indicate number of sensors moved from that region.

Let us denote regions that have at least one sensor at initial deployment as source regions (or sources), and denote regions that do not have any sensor at initial deployment as holes. Region 4 is a source and moves sensors to region 3, since region 3 is close to it, and needs sensors. With local information exchange, region 7 will not
move sensors to region 3, rather it will move sensors to region 11, after synchronizing with regions 4 and 6. Similarly, since region 13 has four sensors, and since regions 9 and 14 do not have any sensor, a sensor moves from region 13 to fill regions 9 and 14. But since region 5 has two sensors, and it receives two sensors from region 1, two sensors move from region 5 to region 9. Other regions also follow the same intuition and synchronization to move sensors. The final deployment is shown in Figure 3.1 (b). Note that regions 14, 15 and 16 have only one sensor. In fact with this movement plan, minimum variance (equal to 0) cannot be achieved. Consider region 14. The only way region 14 can get a sensor is from regions 13, 10 or 15. However, regions 10 and 15 initially did not have any sensor. Thus, no sensor can move to region 14 via regions 10 and 15 since $H = 1$. Similarly, no sensor can move to region 13 via region 9. Besides, region 13 has no extra sensor now. Consequently all paths to region 14 are blocked in this movement plan. A pertinent question to raise at this point is; whether there exists an optimal movement plan that can make the variance 0. If so, what is the plan, or more importantly, what are the challenges need to be addressed in this movement plan. We discuss both issues below.
Figure 3.2: An instance of initial deployment and an optimal movement plan (a), and the resulting deployment (b).

There are two key challenges to our problem. The first challenge is due to our objective of simultaneously minimizing variance and the number of sensor movement hops. Consider the movement plan in Figure 3.1 (a). Region 6 that has six sensors in it, wishes to fill regions 2 and 10. The intuition is because both regions are empty and region 6 is close to them. But this plan, that attempts to minimize hops, cannot minimize variance. There is thus a conflict that may be present in minimizing variance, and the number of hops using local information. For optimum deployment, region 6 should move sensors to regions 10 and 15 (in Figure 3.2 (a)). The path to region 15 may appear long, but it is the one that makes the global variance 0, shown in Figures 3.2 (a) and (b).

The second challenge arises due to limited mobility. Under mobility limitations, if a sensor in one region wishes to move to some far away region, then depending on $H$, there must be mobile sensors in one or more intermediate regions (like a chain) in the corresponding path (if $H = 1$, then all intermediate regions in the path need

---

9We reiterate here the challenges due to limited mobility discussed before in Section 2.2.3 for the sake of completeness.
to have a mobile sensor). If there is no mobile sensor after a sensor has traveled $H$ hops to a particular region, no sensor can move beyond that region, resulting in blocked paths. For instance in Figure 3.1 (b), although region 1 still has extra mobile sensors, all paths from region 1 to region 14 are blocked. The challenge is in determining such optimal chains for sensor movements. Such a path may traverse many intermediate regions as shown in Figure 3.2 (a), where the path from region 6 to region 15 traverses five regions, and a chain of movements is possible in each intermediate region. Determining such a chain of movements for optimal variance and sensor movement hops is not trivial. If sensors make purely local decisions, then optimality cannot be achieved. Also, it is preferable for sensors to make a movement plan (which sensors should move, and where) prior to their movement, in order to avoid erroneous movements, and compensating such errors later on.

3.3 Methodology of Our Algorithms

Our sensor network deployment problem has two objectives; 1) minimizing variance and 2) minimizing overall number of movement hops of the limited mobility sensors. In the following, we discuss our methodology to achieve both objectives. Consider any two regions $i$ and $j$ in a sensor network. Let the number of sensors in region $i$ be less than that of region $j$, both of which are less than $k$. If one sensor is available to move to one of these two regions, the contribution to global variance minimization in the network is larger if the sensor moves to region $i$ than if it moves to region $j$. Our methodology to capture this notion of priority is by weight assignment to regions. When sensors move, larger weight regions are given priority.
compared to smaller weight regions with the objective of global variance minimization. In the above example region $i$ will have larger weight than region $j$ to prioritize sensor movements to region $i$. We discuss our methodology in further detail below.

The overall variance is minimized (equal to 0), when each region has at least $k$ sensors. Thus, for each region $i$ in the sensor network, we first create $\tilde{k}$ virtual sinks (or simply sinks) in order to allocate a position (virtually) for each of the $\tilde{k}$ sensors that are needed in each region. Let each sink in region $i$ be denoted by $s^1_i, s^2_i, s^3_i, \ldots, s^{\tilde{k}}_i$. For each sink $s^1_i, s^2_i, s^3_i, \ldots, s^{\tilde{k}}_i$, we assign weights to them denoted by $w^1_i, w^2_i, w^3_i, \ldots, w^{\tilde{k}}_i$ respectively to prioritize movements towards larger weight sinks. The weights are,

$$w^j_i = 2 \cdot j - 1 \quad (1 \leq j \leq \tilde{k}).$$

(3.3)

Note that sink $s^m_i$ has more weight than $s^n_i$, if $m > n$. Also, $w^m_i = w^m_j$ for any two regions $i$ and $j$.

After sensors move towards sinks (according to their weights), some sinks will have sensors, while some do not. In order to capture the presence of a sensor in each sink among the multiple regions (after sensors move), we define the following function.

$$\phi^j_i = \begin{cases} 1, & \text{if sink } s^j_i \text{ has a sensor,} \\ 0, & \text{otherwise.} \end{cases}$$

(3.4)

There is a constraint for the function $\phi$. If $\phi^j_i = 1$, then $\phi^m_i = 1$ for all $m > j$. We are in effect saying here that if sink $s^j_i$ in region $i$ has a sensor, then each sink $s^m_i$ in region $i$ with larger weights (i.e., $m > j$) should have a sensor. The function $\phi$ captures whether a sink contains a sensor. We define a new metric here called $Score$ as follows,

$$Score = \frac{1}{S} \sum_{i=1}^{S} \sum_{j=1}^{\tilde{k}} \phi^j_i \times w^j_i.$$
The Score function is the summation of weights of those sinks (for all regions) that contain a sensor in them. Clearly, the Score is larger when there are more sinks containing a sensor. The Score function also considers the weight of a sink. As such, in the event that a sensor can move to more than one sink, the Score is larger, when the sensor moves to the sink with the largest weight. Therefore during sensor movements, when we attempt to maximize the Score, we are in effect ensuring that as many sinks as possible contain a sensor, while also ensuring that larger weight sinks always have higher priority compared to smaller weight sinks. We now have the following Theorem.

**Theorem 2.** A sequence of sensor movements that maximizes Score will minimize the variance $\text{Var}$ and vice versa.

**Proof.** Consider two arbitrary sequences of sensor movements $F$ and $G$, with functions $\{f_i^j\}$ and $\{g_i^j\}$ respectively. Assume there are $m_i$ and $n_i$ sinks in region $i$ that have a sensor at the end of sequences $F$ and $G$ respectively. Recalling the constraint of $\phi_i^j$ in (4), we have,

\[
 f_i^j = \begin{cases} 
 1, & j > \bar{k} - m_i, \\
 0, & j \leq \bar{k} - m_i,
\end{cases} \quad (3.6)
\]

\[
 g_i^j = \begin{cases} 
 1, & j > \bar{k} - n_i, \\
 0, & j \leq \bar{k} - n_i.
\end{cases} \quad (3.7)
\]

The gain of Score for sequence $F$ compared with Score for sequence $G$ ($\text{Score}(F) - \text{Score}(G)$) is

\[
 = \frac{1}{S} \sum_{i=1}^{S} \sum_{j=1}^{\bar{k}} (f_i^j \cdot w_i^j) - \frac{1}{S} \sum_{i=1}^{S} \sum_{j=1}^{\bar{k}} (g_i^j \cdot w_i^j) = \frac{1}{S} \sum_{i=1}^{S} ((m_i - n_i) \cdot (2\bar{k} - m_i - n_i)) \quad (3.8)
\]

The loss of variance $\text{Var}$ of $F$ compared with that of $G$ ($\text{Var}(G) - \text{Var}(F)$) is

\[
 = \frac{1}{S} \sum_{i=1}^{S} (\bar{k} - n_i)^2 - \frac{1}{S} \sum_{i=1}^{S} (\bar{k} - m_i)^2 = \frac{1}{S} \sum_{i=1}^{S} ((m_i - n_i) \cdot (2\bar{k} - m_i - n_i)) \quad (3.9)
\]
We can see that the amount of gain in \textit{Score} for \textit{F} is the same as the amount of loss in \textit{Var}. Thus, the sequence of sensor movements that maximizes \textit{Score} simultaneously minimizes \textit{Var}, and vice versa.

From the above theorem, we can see that our original non-linear variance objective can be translated to a linear objective. In this chapter, we propose three algorithms for our deployment problem, following the above methodology. In our algorithms, we create sinks for each region depending on the number of sensors needed. Each sink has a weight associated with it, such that when sensors move, sinks with larger weights have higher priority compared to sinks with smaller weights. The goal of our algorithms is to maximize \textit{Score}, which according to Theorem 2 minimizes the variance \textit{Var}.

The second objective of our problem is minimizing total number of sensor movement hops. We achieve this goal by treating sensor movement hops as costs, and minimizing such costs in our algorithms. When there are multiple sinks in other regions with same weights, our algorithms will ensure that sensors move to sinks in those regions that are closer in terms of distance to be traversed. Clearly, larger weight sinks are still given priority compared to smaller weight sinks. However, with such movements, the resulting number of overall sensor movement hops is minimized, along with maximizing \textit{Score}. If a sensor in a region does not need to move to another region we treat the sensor as \textit{virtually} moving to a sink in the same region. Such a movement incurs 0 cost.
3.4 The Optimal Maximum Flow Based (OMF) Centralized Algorithm

Our first algorithm is the Optimal Maximum Flow based (OMF) centralized algorithm. In the OMF algorithm, the sensor network at initial deployment is translated as a graph structure. The algorithm then determines the minimum cost maximum weighted flow in the graph. The corresponding flow plan in the graph is translated as a movement plan for the sensors in the network. In the following, we describe our OMF algorithm from the perspective of a Base-station executing the algorithm. An alternate approach to execute the OMF algorithm is presented in Section 3.6.1.

3.4.1 Steps in Algorithm Execution

Algorithm 1 Pseudocode of the OMF algorithm

1: Collect the information on the number of sensors in each region in the sensor network.
2: Construct a graph $G_V(V, E)$ using the above region information, desired number of sensors per region $k$ and the sensor mobility capacity $H$. $G_V$ models the sensor network at initial deployment time.
3: Determine the minimum cost maximum weighted flow from source regions to weighted sinks in $G_V$.
4: Determine a movement plan for the sensors in the sensor network based on the above flow plan in $G_V$.
5: Forward the movement plan to sensors in the network.

Algorithm 1 shows the sequence of steps in the OMF algorithm. In Step 1, each sensor in the network identifies which region it resides in. Sensors then forward information on the number of sensors in their region towards the Base-station. For routing packets towards Base-station, protocols like [40], [41], [23], [47] can be used, where the protocols route packets towards intended destinations in the network (Base-station in our case) using shortest paths. The Base-station thus obtains information on the number of sensors in all regions in Step 1. As pointed out before, for determining which region a sensor resides in, sensors can be provisioned with GPS devices or
methods proposed in [39] can be used where location of sensors is determined by using sensors themselves as landmarks. Also, we assume the network is connected without partitions. The issue of partitions is discussed later.

In Step 2, the Base-station constructs a virtual graph \( G_V \), whose vertices and edges model the regions and sensor movement ability between regions respectively at initial deployment. In Step 3, the Base-station determines the maximum weighted flow to the sinks in \( G_V \) (that maximizes equation 3.5) with minimum cost. In Step 4, the flow plan in the \( G_V \) corresponding to the minimum cost maximum weighted flow is translated as a movement plan for the sensors. In Step 5, the Base-station forwards the movement plan (which sensors should move and where) to the sensors in the network. We subsequently prove that this movement plan minimizes the variance, and overall number of sensor movement hops in the sensor network. Each of Steps 2, 3, 4 and 5 in our OMF algorithm is discussed in detail below.

3.4.2 Constructing the Virtual Graph \( G_V \)

We now discuss Step 2, that involves the construction of the virtual graph denoted by \( G_V(V_V, E_V) \). Before we discuss \( G_V \), we introduce the notation of reachability between regions. For any region \( i \) in the sensor network, we denote its reachable regions as those regions to which a sensor from region \( i \) can move to. Obviously, the reachable regions depend on the maximum movement hops \( H \). We first assume that the regions to which a sensor can move to, are regions in its adjacent left, right, top and bottom directions only. Thus, if \( H = 1 \) in Figure 3.1, then the reachable regions for region 1 are regions 2 and 5. If \( H = 2 \), the reachable regions are regions 2, 3, 5, 6 and 9.
The construction of $G_V$ involves 1) The establishing of vertices and edges for each region in the sensor network and creation of sinks for each region, 2) The establishing of reachability relationship between the regions, 3) Adding weights to sinks following our discussions in Section 3.3 and 4) Adding costs to edges to capture sensor movements across regions. The objective of this construction is to ensure that $G_V$ models the sensor network, identifies sources, sinks, and reachability relationship among regions. Figure 3.3 (a) shows an instance on initial deployment for a $2 \times 2$ network with 4 regions and 12 sensors, and where $\bar{k} = 3$ and $H = 1$. Its corresponding virtual graph $G_V$ is shown in Figure 3.3 (b). The numbers inside the circles in Figure 3.3 (a) denotes the number of sensors in the corresponding region in the sensor network.

In the following, we describe the virtual graph construction process in detail. Let us first describe the establishment of vertices and edges assignment in $G_V$ for one arbitrary region in the sensor network. Without loss of generality, consider region $i$ with initially $n_i$ sensors. For this region, we create a vertex called as the base vertex of region $i$ (denoted by $v_{ib}^i$) in $G_V$. We create vertex $v_{out}^i$ to keep track of the number of sensors that can move out from region $i$. We then create $\bar{k}$ sink vertices for region $i$ (due to deployment requirement of $\bar{k}$ sensors per region). The sink vertices for region $i$ are denoted by $vs_{1i}, vs_{2i}, vs_{3i}, \ldots, vs_{\bar{k}i}$. We also create vertex $v_{in}^i$ as a proxy for the $\bar{k}$ sink vertices.

The next step is adding edges between vertices for this region. An edge of capacity $n_i$ is added from $v_{ib}^i$ to $v_{out}^i$. This means that up to $n_i$ sensors can move from region $i$. Since $v_{in}^i$ is a proxy for the sink vertices, the capacity from $v_{out}^i$ to $v_{in}^i$ is also $n_i$. From $v_{in}^i$, an edge is added to each of the vertices $vs_{1i}, vs_{2i}, vs_{3i}, \ldots, vs_{\bar{k}i}$ with capacity 1. Since the deployment requirement is $\bar{k}$ sensors per region, we allow up to one sensor
to move to each sink (for $k$ such sinks). All other regions are treated similarly in $G_V$. For example, for region 1 in $G_V$ in Figure 3.3 (b), we create six vertices corresponding to the base vertex ($v_1^b$), in vertex ($v_1^{in}$), out vertex ($v_1^{out}$) and $k = 3$ sink vertices ($vs_1^1$, $vs_1^2$ and $vs_1^3$). Edges between the vertices, and their capacities for region 1 are also shown. All other regions are treated similarly.

Figure 3.3: An instance of the initial network deployment (a) and the corresponding virtual graph $G_V$ (b).

The second step is establishing reachability relationship among the regions into $G_V$. Let us consider two arbitrary regions $i$ and $j$ that are reachable from each other. In $G_V$, edges are added from $v_i^{out}$ to $v_j^{in}$, with edge capacity $n_i$, which is the number of sensors in region $i$. This is to allow up to $n_i$ sensors to move from region $i$ to region $j$. Correspondingly, edges are added from $v_j^{out}$ to $v_i^{in}$, with capacity $n_j$. For example in Figure 3.3 (b), there is an edge from $v_1^{out}$ to $v_2^{in}$ with capacity $n_1 = 4$, and an edge
from $v_2^{out}$ to $v_1^{in}$ with capacity $n_2 = 2$ since regions 1 and 2 are reachable from each other.

The next steps are weight assignment to sinks, and cost assignment to edges. Consider region $i$ again. For sinks $vs_{1i}^1$, $vs_{2i}^2$, $vs_{3i}^3$, ..., $vs_{ki}^k$ in region $i$, we denote their weights as $w_{1i}^1$, $w_{2i}^2$, $w_{3i}^3$, ..., $w_{ki}^k$. Following from the discussions in Section 3.3, the values for the weights are 1, 3, 5, ..., $2\tilde{k} - 1$ respectively. Since $\tilde{k} = 3$ in the example in Figure 3.3, we have weights 1, 3 and 5 for the sinks (shown along side the sink vertices). Note that, $w_{ni}^m$ is larger than $w_{ni}^j$, if $m > j$. We now discuss costs for edges between regions in $G_V$ in order to capture number of sensor movements. If a sensor moves from its region to its adjacent region, then it denotes one hop made by the sensor. Let us consider two regions $i$ and $j$ in the sensor network that are reachable from each other. Let the distance between them in terms of number of hops be $d_{i,j}$. That is, $d_{i,j}$ denotes the minimum number of hops required for a sensor in region $i$ to move to region $j$ (or vice versa). For instance, in Figure 3.1 (a), $d_{1,3} = d_{3,1} = 1$. Obviously $d_{i,j} \leq H$, if regions $i$ and $j$ are reachable from each other. To incorporate this in $G_V$, between any two reachable regions $i$ and $j$, the costs of edges from $v_i^{out}$ to $v_j^{in}$, and the costs of edges from $v_j^{out}$ to $v_i^{in}$ are assigned as $d_{i,j}$. Apart from the above, the only remaining edges in $G_V$ are the ones from $v_i^b$ to $v_i^{out}$, from $v_i^{out}$ to $v_i^{in}$, and from $v_i^{in}$ to $vs_{i1}^1$, $vs_{i2}^2$, $vs_{i3}^3$, ..., $vs_{ki}^k$ (for all regions $i$). These edges denote internal movements within a region, and the cost for these edges is set as 0. The costs of edges in $G_V$ are not shown in Figure 3.3.

At this point, Step 2 of our OMF algorithm is completed. The Base-station has constructed $G_V$ that models the sensor network at initial deployment. Before proceeding to Step 3, we define a flow plan $Z$ in $G_V$ and a metric $W$. $Z$ is the sequence
of flows (in $G_V$) that meets the following condition; $W = \sum_{i=1}^{S} \sum_{j=1}^{K} (f_i^j \cdot w_i^j)$ is maximized, where $f_i^j$ is the subflow to sink $v s_i^j$ in flow plan $Z$. We call $Z$ as a maximum weighted flow plan in $G_V$. If the cost of $Z$ is minimized, $Z$ is called as a \textit{minimum cost maximum weighted flow plan}. With sinks in $G_V$ having weights associated with them, a maximum weighted flow plan must maximize the number of sink vertices that receive a flow, and prioritize flows to larger weight sinks first compared to smaller weight sinks in $G_V$. Since the capacity of the edge from $v_i^m$ to $v s_i^j$ in $G_V$ is 1, $f_i^j$ meets the constraint of function $\phi$ defined in (3.4). Since $G_V$ is a translation of the sensor network, the flow plan $Z$ in $G_V$ can be translated as a corresponding movement plan for sensors in the sensor network (exactly how this is done is discussed in Section 3.4.4). From the definition of $\text{Score}$ in (3.5) and $W$ above, the corresponding sensor movement plan maximizes the $\text{Score}$ with minimum cost, which in turn minimizes $Var$ (from Theorem 2) with minimum cost. To summarize, with the construction of $G_V$ in place, the variance/movement minimization problem now becomes one, where the weighted flow to sinks in $G_V$ is to be maximized with minimum cost.

\subsection*{3.4.3 Computing the Minimum Cost Maximum Weighted Flow in the Virtual Graph $G_V$}

We now proceed to Step 3 in the \textit{OMF} algorithm, where determine the minimum cost maximum weighted flow in $G_V$. In the following, we present our algorithm to determine the minimum cost maximum weighted flow in $G_V$. Our approach to maximize the weighted flow is to translate larger weight sink vertices, as lower cost sink edges. Thus, prioritizing flows to large weight sinks now becomes prioritizing flows through lower cost edges. This is the crux of our algorithm described below.
Algorithm 2 Pseudocode for computing the minimum cost maximum weighted flow in $G_V$

1: Input: $G_V(V_V, E_V)$, $H$, $S$ and $k$
2: Output: Graph $G^m_V(\bar{V}, \bar{E})$ and Minimum Cost Maximum Weighted Flow Plan $Z$ in $G_V$
3: $|E| = \text{No. of Edges in} \ G_V$, $|V| = \text{No. of Vertices in} \ G_V$
4: $\bar{V} = |V_V| + S \times (k + 1)$
5: $\bar{V} = |V_V| + 2$
6: Add vertices $S^v_{\text{source}}$ and $S^v_{\text{sink}}$ to $G_V$ to create graph $G^m_V$
7: for each region $i$ do
8:   for $j$ from 1 to $k$ do
9:      Add edge from sink $v_{s_i}^j$ to $S^v_{\text{sink}}$
10:     Assign corresponding edge capacity as 1
11:     Assign corresponding edge cost as $-(2j - 1) \times H \times |E|$
12:   end for
13: Add edge from $S^v_{\text{source}}$ to $v^b_{i}$
14: Assign corresponding edge capacity as $\infty$
15: Assign corresponding edge cost as 0
16: end for
17: Determine the maximum flow value $|\bar{Z}|$ from $S^v_{\text{source}}$ to $S^v_{\text{sink}}$ in $G^m_V$
18: Determine the minimum cost flow plan $Z$ (for flow value $|\bar{Z}|$) from $S^v_{\text{source}}$ to $S^v_{\text{sink}}$ in $G^m_V$

Algorithm 2 is the pseudocode to determine the minimum cost maximum weighted flow in the virtual graph $G_V$. The input is $G_V(V_V, E_V)$, $H$, number of regions $S$ and $k$. We first create a new graph from $G_V$ called $G^m_V(\bar{V}, \bar{E})$ as follows. We first create two new vertices called $\text{Super Source}$ and $\text{Super Sink}$, denoted by $S^v_{\text{source}}$ and $S^v_{\text{sink}}$ respectively. Edges are added from each sink vertex to $S^v_{\text{sink}}$, with capacity 1 to allow only one sensor to move from each sink towards $S^v_{\text{sink}}$. The cost of the edges from sinks $v_{s_i}^1$, $v_{s_i}^2$, $v_{s_i}^3$, ..., $v_{s_i}^k$ to $S^v_{\text{sink}}$ (for all regions $i$) are set as $-H \times |\bar{E}|$, $-3 \times H \times |\bar{E}|$, $-5 \times H \times |\bar{E}|$, ..., $-(2k - 1) \times H \times |\bar{E}|$ respectively, where $|\bar{E}|$ is defined in Algorithm 2. Finally, edges are added from $S^v_{\text{source}}$ to all base vertices (i.e., $v^b_i$ for all $i$), with capacity $\infty$ to allow any amount of flow from $S^v_{\text{source}}$. The costs for these edges are set as 0, since the flow through such edges are not actual sensor movements.

At this point (Step 16 in Algorithm 2), $G^m_V$ has been constructed. Determining the flow plan to maximize weighted flow to sinks with minimum cost in $G^m_V$ is a two-step process (Steps 17 and 18 in Algorithm 2). The Base-station will first determine the

\[10\] The interpretation of $|\bar{E}|$ is discussed subsequently.
maximum flow value ($|\vec{Z}|$) from $S_{source}^v$ to $S_{sink}^v$ in $G_V^m$. The maximum flow value $|\vec{Z}|$ indicates the maximum number of sinks that can get a sensor in $G_V^m$. However, this only indicates the maximum number of sinks. The determination of the maximum flow value does not consider the fact that sinks have different weights and larger weight sinks need to be accorded higher priority. Our objective however, is to determine the flow plan $Z$ (the actual flow among the edges) in $G_V^m$ such that weighted flow to sinks is maximized with minimum cost. We do this in Step 18 by determining the minimum cost flow plan $Z$ (for maximum flow value $|\vec{Z}|$) in $G_V^m$, discussed further below.

We know that when executing the minimum cost flow algorithm on any graph, flow is prioritized through edges with lower cost. By setting the edge costs from sinks to $S_{sink}^v$ as the negative of weights of the corresponding sink, we will achieve our objective of prioritizing flow to sinks with larger weights in determining the minimum cost flow to $S_{sink}^v$. There is one issue we have to resolve during cost assignment. Recall that sensor movements between reachable regions are considered as costs in $G_V^m$. Clearly, these costs will affect the minimum cost flow plan when determining flows to sinks with minimum cost in $G_V^m$. To prevent this from happening, the costs from sinks to $S_{sink}^v$ is assigned as the negative of the sink weights multiplied by a large constant (namely, $H \times |\vec{E}|$). This constant is large enough to ensure that the flow plan ($Z$) to maximize weighted flow in $G_V^m$ is not affected by the costs between reachable regions, while still minimizing costs between reachable regions (that denote sensor movements). Before discussing how to translate this flow plan $Z$ into a sensor movement plan, we state the following theorem showing the relationship between $G_V^m$ and $G_V$.  

Theorem 3. The flow plan corresponding to the minimum cost maximum flow in $G^m_V$ is the flow plan corresponding to the minimum cost maximum weighted flow in $G_V$.

Proof. We first prove that the flow plan corresponding to the minimum cost maximum flow in $G^m_V$ is the flow plan corresponding to the maximum weighted flow in $G_V$. We will prove this by contradiction. Let the minimum cost maximum flow plan in $G^m_V$ be $Z$. Suppose $Z$ does not yield the maximum weighted flow in $G_V$. This means there exists a flow plan $Y$ that has a higher weighted flow than that of $Z$. Let us denote the weighted flow values of $Z$ and $Y$ to sinks in $G_V$ by $W_Z$ and $W_Y$ respectively. We then have $W_Y - W_Z \geq 1$. Denoting $Cost_Z$ and $Cost_Y$ as the cost values of $Z$ and $Y$ in $G^m_V$ respectively, we have,

\begin{align}
Cost_Z &= -W_Z \cdot |\overline{E}| \cdot H + Cost'_Z \\
Cost_Y &= -W_Y \cdot |\overline{E}| \cdot H + Cost'_Y
\end{align}

in which $Cost'_Z$ and $Cost'_Y$ denote the sum of the edge costs from $v^\text{out}_i$ to $v^\text{in}_j$ for all regions $i$ and $j$ in $Z$ and $Y$ respectively. Since $Z$ applies minimum cost flow algorithm, we have $Cost_Z < Cost_Y$. However, we can also obtain,

\begin{align*}
Cost_Z &= -W_Z \cdot |\overline{E}| \cdot H + Cost'_Z \geq -W_Z \cdot |\overline{E}| \cdot H \geq -W_Y \cdot |\overline{E}| \cdot H + |\overline{E}| \cdot H \\
&> -W_Y \cdot |\overline{E}| \cdot H + Cost'_Y = Cost_Y,
\end{align*}

which is a contradiction. Therefore, flow plan $Z$ yields the maximum weighted flow in $G_V$. Since $Z$ is the plan after executing the minimum cost algorithm in $G^m_V$, the costs of flow among edges between reachable regions is minimized in $G^m_V$. $G_V$ is made of exactly the same edges (edges between reachable regions). Therefore, flow plan $Z$ corresponds to the minimum cost maximum weighted flow in $G_V$. \qed
3.4.4 Determining the optimal movement plan from the virtual graph $G_V$

Once the minimum cost maximum weighted flow to each sink in $G_V$ (and the corresponding flow plan in all edges in $G_V$) is obtained, we proceed to Step 4 in Algorithm 1. In Step 4, we translate the flow plan from Step 3 into actual sensor movements as follows. Let $Z^V$ denote the flow plan (a set of flows) corresponding to the minimum cost maximum weighted flow algorithm in $G_V$, where the capacity of each flow is 1. Each flow $z^V(v_i^b, vs_j^x) \in Z^V$ is a flow from $v_i^b$ to $vs_j^x$ in $G_V$. The flow $z^V(v_i^b, vs_j^x)$ is of the form $\langle v_i^b, v_i^{out}, v_j^{in}, vs_j^x, i \rangle$. Thus, for the flow plan $Z^V$, we can map it to a corresponding movement plan $Z^S$ (set of movement sequences for sensors) in the sensor network. That is for each $z^V(v_i^b, vs_j^x) (\in Z^V)$ of the form $\langle v_i^b, v_i^{out}, v_j^{in}, vs_j^x, i \rangle$, the corresponding $z^S(i, j) (\in Z^S)$ is of the form $\langle i, j \rangle$. Physically, this means that one sensor should move from region $i$ to region $j$. The sensor movement plan $Z^S$ (consisting of the set of all such $z^S$, obtained from $z^V$) is our output. This movement plan that indicates which sensors should move and where to, is forwarded by the Base-station to the sensors in the network.

3.4.5 Optimality of the OMF Algorithm

Before discussing optimality, we first introduce the concept of feasible flows and movement sequences. We call a flow $z^V(v_i^b, vs_j^x)$ of the form $\langle v_i^b, v_i^{out}, v_j^{in}, vs_j^x, i \rangle$ feasible in $G_V$ if there exists positive edge capacities from vertices $v_i^b$ to $v_i^{out}$, $v_i^{out}$ to $v_j^{in}$, $v_j^{in}$ to $vs_j^x$. We call a movement sequence $z^S(i, j)$ of the form $\langle i, j \rangle$ feasible in the sensor network if there is at least one mobile sensor in region $i$ that can move to region $j$. We have the following lemma for a flow in $G_V$ and a sensor movement sequence.
Lemma 1. A flow $z^V(v_i^b, vs_j^x)$ in $G_V$ is feasible if and only if the corresponding movement sequence $z^S(i, j)$ is feasible in the sensor network.

Proof. We first prove if $z^S(i, j)$ is feasible, then $z^V(v_i^b, vs_j^x)$ is feasible. If $z^S(i, j)$ is feasible, then there is at least one mobile sensor in region $i$, and regions $i$ and $j$ are reachable from each other. That is, the capacities of the edges from $v_i^b$ to $v_i^{out}$, and from $v_i^{out}$ to $v_j^{in}$ are $\geq 1$, and there exists an edge from $v_j^{in}$ to $vs_j^x$, whose capacity is 1 (from Section 3.4.2). Thus, $z^V(v_i^b, vs_j^x)$ is feasible.

We now prove if $z^V(v_i^b, vs_j^x)$ is feasible, then $z^S(i, j)$ is feasible. If $z^V(v_i^b, vs_j^x) = (v_i^b, v_i^{out}, v_j^{in}, vs_j^x)$ is feasible, then the capacities of the edges from $v_i^b$ to $v_i^{out}$, from $v_i^{out}$ to $v_j^{in}$ and from $v_j^{in}$ to $vs_j^x$ are all $\geq 1$. This implies that there is a sensor in region $i$, and regions $i$ and $j$ are reachable from each other. So a sensor can move from region $i$ to region $j$. Thus $z^S(i, j)$ is feasible. \hfill \square

We obtain the following corollary from Lemma 1.

Corollary 1. For a feasible flow plan $\bar{Z}^V$ (set of all $z^V$) in $G_V$, a corresponding feasible sensor movement sequence plan $\bar{Z}^S$ (set of all $z^S$) can be found in the sensor network and vice versa.

Proof. We first prove that for a feasible flow plan $\bar{Z}^V$ in $G_V$, a corresponding sensor movement sequence plan can be found in the sensor network. Consider an arbitrary feasible flow $z^V$ in $\bar{Z}^V$. By Lemma 1, a corresponding feasible sensor movement sequence $z^S$ in the sensor network can be found for the flow $z^V$ in $G_V$. The set of such sensor movement sequences is $\bar{Z}^S$ in the sensor network.

We now prove that for a feasible sensor movement sequence plan $\bar{Z}^S$ in the sensor network, a corresponding flow plan can be found in $G_V$. Consider an arbitrary feasible
sensor movement sequence $z^S$ in $\tilde{Z}^S$. By Lemma 1, a corresponding feasible flow $z^V$ in $G_V$ can be found for the sensor movement sequence $z^S$ in the sensor network. The set of such flows is $\tilde{Z}^V$ in $G_V$.

The following Theorem shows that the movement plan obtained by our OMF algorithm optimizes both variance and the number of sensor movement hops.

**Theorem 4.** Let $Z^V_{opt}$ be the minimum cost maximum weighted flow plan in $G_V$. Its corresponding movement plan $Z^S_{opt}$ will minimize variance and the number of sensor movement hops in the sensor network.

**Proof.** We first prove that our OMF algorithm is optimal in terms of minimizing variance. We prove by contradiction. Consider a sensor movement plan $Z^S_{opt}$ that corresponds to a flow plan $Z^V_{opt}$ determined by executing the minimum cost maximum weighted flow algorithm on $G_V$. Let this movement plan be non-optimal in terms of variance. This implies there is a better movement plan, $Z^S_x$ that can further minimize variance in the sensor network. By Corollary 1, a corresponding flow plan $Z^V_x$ can be found in $G_V$. The amount of weighted flow in this plan is larger than the weighted flow achieved using plan $Z^V_{opt}$, which is a contradiction. Hence $Z^S_{opt}$ is the optimal movement plan for sensors that minimizes variance.

We now prove that our OMF algorithm is optimal in terms of minimizing number of sensor movement hops. We prove by contradiction. Consider a sensor movement plan $Z^S_{opt}$ that corresponds to a flow plan $Z^V_{opt}$ determined by executing the minimum cost maximum weighted flow algorithm on $G_V$. Let this movement plan be non-optimal in terms of number of sensor movement hops. This implies that there is a
better plan, $Z^S_x$ that can reduce at least one movement in the sensor network. By Corollary 1, a corresponding flow plan $Z^V_x$ can be found in $G_V$. The number of movement hops (or overall cost) in this plan is less than that achieved using $Z^V_{opt}$, which is a contradiction. Hence $Z^S_{opt}$ is the optimal movement plan that minimizes number of sensor movement hops.

We now discuss time complexity of the OMF algorithm. There are three phases in our algorithm in determining the optimal movement plan. The first is construction of $G_V(V_V, E_V)$ and $G_{mV}(\tilde{V}, \tilde{E})$, the second is determining the maximum flow in $G_{mV}^m$, and the third is determining the minimum cost flow in $G_{mV}^m$. The time complexity is dominated by determining the maximum flow and minimum cost flow in $G_{mV}^m$. Our implementations of the maximum flow algorithm is the Edmonds-Karp algorithm [42], and minimum cost flow algorithm is the one in [43]. The resulting time complexity is $O(max (|\tilde{V}|^2, |\tilde{V}|^2|\tilde{E}|log|\tilde{V}|))$. Here $|\tilde{V}|$ and $|\tilde{E}|$ denote the number of vertices and edges in $G_{mV}^m$, and are given by, $|\tilde{V}| = O(\tilde{k}(\lceil \frac{Q}{R} \rceil)^2))$, and $|\tilde{E}| = O(\tilde{k}H^2(\lceil \frac{Q}{R} \rceil^2))$, in which $Q$ is the sensor network size and $R$ is the region size.

3.5 The Simple Peak-Pit based (SPP) Distributed Algorithm

3.5.1 Algorithm Rationale

In the above, we presented a centralized and optimal OMF algorithm to our deployment using our weight-based methodology. We now present the Simple Peak-Pit (SPP) based algorithm to our deployment problem, that is local, light-weight and purely distributed. In the SPP algorithm, regions request sensors from adjacent regions, with weights attached to each request. As before, requests with larger weights are given higher priority when compared to requests with smaller weights, while
simultaneously preferring shorter movement hops to satisfy requests. We first discuss some important notations used in the algorithm description. Regions in the network are classified into three types: \textit{pits}, \textit{peaks} and \textit{forwarders}. A \textit{pit} is a region whose number of sensors is less than \(k\) and not more than any of its neighboring regions. A \textit{peak} is a region whose number of sensors is larger than any of its neighboring regions. All other regions are \textit{forwarders}. We define an \textit{over-}\(\tilde{k}\) \textit{forwarder} as a \textit{forwarder} with more than \(\tilde{k}\) sensors, and denote the \textit{richest} neighbor of a region as a neighbor with the largest number of sensors.

In the SPP algorithm, a \textit{pit} \(i\) will request \(\tilde{k} - n_i\) sensors in its request (\textit{REQ}). The \textit{pit} \(i\) will assign different weights to each of the \(\tilde{k} - n_i\) requested sensors as, \(w_i^j = 1, 3, 5, \ldots, 2j - 1\), where \(1 \leq j \leq (\tilde{k} - n_i)\) (as before). Here, we let only \textit{pits} send \textit{REQ}s, so that \textit{non-pit} regions will not compete with \textit{pits} during requests to ensure that more deficient regions will be given priority. A \textit{REQ} generated will be forwarded towards progressively richest neighbors to increase likelihood of \textit{REQ}s arriving at \textit{over-}\(\tilde{k}\) \textit{forwarders} or \textit{peaks} on shorter paths. Recipients receiving \textit{REQ}s will sort all the requested sensors in the \textit{REQ}s by \textit{weights} and serve those with larger \textit{weights} first. Ties are broken by fulfilling requests with shorter paths first. We call the neighbors chosen for the next hop as \textit{tried} neighbors.

### 3.5.2 Steps in Algorithm Execution

Algorithm 3 shows the pseudocode of our SPP algorithm. It is executed by each region \(i\) independently and is event driven. Using inter-region communications, a region leader will be elected for co-ordination. Each leader obtains the number of sensors in its region, and its four adjacent neighboring regions. The region leader
of each pit will send an REQ to its richest neighbor, requesting number of sensors needed. If multiple richest neighbors exist, ties are broken randomly. If some regions have no sensors, they can be assisted by neighboring leaders in sending our requests.

We discuss this issue in further detail later.

Algorithm 3 Pseudocode of the SPP algorithm run by region \(i\)

```
1: Region leader selection
2: while TRUE do
3:    switch type of event
4:        case region \(i\) becomes a pit:
5:            send REQ to richest untried neighbor;
6:        case receive REQ:
7:            put REQ into Queue(i);
8:            if region \(i\) is an over-k forwarder, then
9:                select REQs in Queue(i) to serve by weights and path lengths;
10:               send ACKs, move sensors and/or forward REQs accordingly;
11:            else if region \(i\) is a peak, then
12:                select REQs in Queue(i) to serve by weights and path lengths;
13:                send ACKs, move sensors and/or send FAILs accordingly;
14:            else
15:                forward REQ to richest neighbor;
16:            case receive ACK for pit \(j\):
17:                forward ACK to \(j\) if \(i \neq j\);
18:            case receive FAIL for pit \(j\):
19:                resend REQ to richest untried neighbor;
20:            case detect hole neighboring region \(j\):
21:                if region \(i\) can provide sensor, then
22:                    move a sensor to \(j\) after random delay;
23:        end switch
24:    end while
```

Due to limited mobility, when requests are sent out, it is important that path feasibility should be maintained during the selection of next hop forwarder. This means there should exist at least one mobile sensor on any continuous \(H\) hop segment of the path a REQ traverses. Otherwise, mobile sensors on the other side of the segment will not be able to move back to the requesting pit due to limited mobility. In case there is not enough mobile sensors in a certain segment with \(H\) hops on the path, the requested number of sensors in the REQ message should be adjusted since
we can never move enough sensors back on the path. All the intermediate *forwarders* will reserve enough number of mobile sensors to guarantee the feasibility of the path.

When *REQs* are forwarded to *over-\( \bar{k} \)* *forwarders* or *peaks*, some of them may or may not get served. Considering that the *REQ* with largest *weight* requested sensor may not always come first, the *over-\( \bar{k} \)* *forwarder* or *peak* will put the *REQs* into its queue and serve them in periodic intervals of time. When serving multiple requested sensors with the same *weights*, those with shorter paths will be served first. An *over-\( \bar{k} \)* *forwarder* will send *ACKs* back to the *pits* whose *REQs* contains sensors that will be served, and forward the *REQs* if not all sensors can be served. Those forwarded *REQs* will be updated if part of the requested sensors are served eventually. A *peak* will send *ACKs* back to the *pits* whose *REQs* contains sensors that will be served, and send *FAILs* back to *pits* if not all requests can be served

11. Sensors will start moving after *ACKs* are sent, following the reserved paths of the corresponding *REQs*. After receiving the *ACK* and mobile sensor(s), each *pit* will inform its neighbors its new sensor number, and *REQs* are generated if need be.

After a *pit* or *forwarder* receives a *FAIL*, it will release the reserved path and resend *REQs* to its *richest untried* neighbor and so on. The algorithm terminates when each *pit* has either obtained at-least \( \bar{k} \) sensors, or expiration of movement capability of the sensors, or if a certain number of requests have been tried without success by requesting sensors

12. It may happen some regions have no sensors in them (i.e., *holes*) after initial deployment. A hole can be filled by one of its neighbors with extra mobile sensors,

---

11We do not let recipients of requests choose shortest return paths as such paths may be *blocked* due to mobility limitations.

12The number of unsuccessful requests per sensor is application decided.
after coordination by other non-empty neighbors. In case a hole cannot be filled
directly due to none of its neighbors being able to provide an extra sensor, one of its
neighbors can become its proxy region via the same mechanism discussed above. In
the extreme case when all of a hole’s neighbors are empty, the hole may be filled by
sensors, or have a proxy region leader later when some of its neighbors get sensors
during the $SPP$ algorithm execution.

3.6 Discussions

In the above, we presented an optimal centralized $OMF$, and a distributed $SPP$
algorithm for our deployment problem. We now discuss execution of our algorithms,
balancing sensor deployments and movements, extensions to non-uniform scenarios,
the issues of arbitrary sensor movement directions and network partitions.

3.6.1 Executing our Algorithms

The algorithms we proposed above can be executed in more than one way. We
first discuss a semi-distributed version of the the $OMF$ algorithm, called the $Domain-
based OMF$ ($D-OMF$) algorithm. Here the sensor network is divided into multiple
domains, and each domain contains multiple regions. We let each domain obtain
region information (number of sensors) only in their domain. The movement plan for
variance minimization in each domain is independently determined with this informa-
tion (without exchanging information with other domains) using the $OMF$ algorithm.
The Base-station can do this for each domain, or a special sensor in each domain can
do so. Note that the $D-OMF$ algorithm being semi-distributed has lower messaging
and computational complexity than $OMF$ algorithm. But, optimality is compromised
since the $D$-$OMF$ algorithm achieves local optima in each domain and cannot guarantee global optima. Note that this trade-off depends on the domain size, uniformity of initial deployment and sensor mobility capacity. Conducting an analytical comparison of performance of $D$-$OMF$ and $OMF$ algorithms is too difficult if not impossible. We study this using extensive simulations in Section 3.7. Furthermore, we point out that our proposed algorithms can also be combinedly executed. A simple instance is one where the $OMF$ algorithm is executed first to optimize deployment, and at later stages the lightweight distributed algorithms can be executed to repair deployment under faults, failures etc.

3.6.2 Balancing Sensor Deployment and Movements

a). Balancing Sensor Deployment: In some applications, the deployment goal is to ensure that the number of sensors among all regions is balanced [2]. Denoting $k_{ave}$ as the average number of sensors per region, the problem now is to minimize the global variance in the number of sensors among all regions from $k_{ave}$. We now discuss how to modify $G_{V}$ in the $OMF$ algorithm for this problem. The key issue here is how many sinks per region need to be created in $G_{V}$. Intuitively, it may appear that $k_{ave}$ sinks per region need to be created. However, due to limited mobility, it may happen that not all regions will eventually have $k_{ave}$ sensors. Also, for global balancing, we should not constrain the possible movement choices for the sensors, which means we cannot set $k = k_{ave}$, since this constrains the number of sensors that can move into a region strictly by $k_{ave}$. We thus need $k$ to be a larger number. The following Lemma shows that the resulting deployment plan with some arbitrary constant $k_{arb}$
will minimize the global variance in terms of number of sensors among all regions from $\bar{k}_{ave}$.

**Lemma 2.** Let $k_{arb}$ be any arbitrary constant. The final deployment plan that minimizes the global variance in the number of sensors among all regions from $k_{arb}$ will minimize the global variance in the number of sensors among all regions from $\bar{k}_{ave}$.

**Proof.** At final deployment, let regions $r_1, r_2, r_3, \ldots, r_S$ contain $k_1, k_2, k_3, \ldots, k_S$ sensors respectively. The numbers $k_1, k_2, k_3, \ldots, k_S$, correspond to the deployment that minimizes variance from $k_{arb}$. Clearly, the number of sensors in the network, $N = \sum_{i=1}^{S} k_i$. The variance from $k_{arb}$ is given by $V_{arb} = \frac{1}{S} \sum_{i=1}^{S} (k_i - k_{arb})^2$. By our assumption, this is minimum. Therefore, $k_1^2 + k_2^2 + k_3^2 + \ldots + k_S^2 - 2k_{arb}(k_1 + k_2 + k_3 + \ldots + k_S) + Sk_{arb}^2$ is minimum. That is, $k_1^2 + k_2^2 + k_3^2 + \ldots + k_S^2 - 2k_{arb}(N) + Sk_{arb}^2$ is minimum. That is, $k_1^2 + k_2^2 + k_3^2 + \ldots + k_S^2$ is minimum, since $S$, $N$ and $k_{arb}$ are constants. This implies that $k_1^2 + k_2^2 + k_3^2 + \ldots + k_S^2 - 2\bar{k}_{ave}(k_1 + k_2 + k_3 + \ldots + k_S) + S\bar{k}_{ave}^2$ is also minimum, since $\bar{k}_{ave}$ is a constant. That is, variance from $\bar{k}_{ave}$ is also minimum. \qed

With the above Lemma, we create $n_{max}$ sinks for each region in $G_V$, where $n_{max}$ (a constant) is the number of sensors in the region with the maximum number of sensors at initial deployment. All other construction rules for $G_V$ remain the same. The following theorem shows the optimality of our solution to the problem of balancing sensor deployment.

**Theorem 5.** The deployment plan after executing the OMF algorithm with $n_{max}$ sinks for each region in $G_V$ will minimize the global variance in the number of sensors among all regions from $\bar{k}_{ave}$. 

80
Proof. We first prove that executing the OMF algorithm with \( n_{\text{max}} \) sinks will minimize the variance in the number of sensors among all regions from \( n_{\text{max}} \). We prove this by contradiction. Let \( Z_{\text{opt}} \) be the movement plan after executing the OMF algorithm with \( n_{\text{max}} \) that is optimum. Assume region \( i \) has more than \( n_{\text{max}} \) sensors after execution of OMF algorithm. That is, \( k_i > n_{\text{max}} \). Since \( n_i \leq n_{\text{max}} \), at least one of the \( k_i \) sensors in region \( i \) comes from another region. Let this region be denoted by \( j \) (\( i \neq j \)). Let us denote the sensor moving from region \( i \) to region \( j \) as \( s_1 \). If \( k_j < n_{\text{max}} \), then we can construct a new movement plan, where sensor \( s_1 \) stays in region \( j \). Clearly this plan reduces the variance compared to \( Z_{\text{opt}} \), which is a contradiction. Similarly, if \( k_j = n_{\text{max}} \), then we can construct a new movement plan, where sensor \( s_1 \) stays in region \( j \). Clearly this plan reduces the number of movements, while still maintaining the same variance compared to \( Z_{\text{opt}} \), which is a contradiction. Therefore \( k_j > n_{\text{max}} \). Now, since region \( j \) initially had \( \leq n_{\text{max}} \) sensors, and since it now has more than \( n_{\text{max}} \) sensors, it must be the case that a sensor \( s_2 \) has moved from some region \( m \) to region \( j \). Using the above same argument, we can show that this is a contradiction. Therefore, \( n_{\text{max}} \) sinks suffices for all sensors to move to some sink during variance minimization, which means the global variance from \( n_{\text{max}} \) is minimized in the corresponding movement plan (since we have \( n_{\text{max}} \) sinks). Clearly, \( n_{\text{max}} \) is a constant. By Lemma 2, a deployment plan that minimizes variance in number of sensors from \( n_{\text{max}} \) among all regions, also minimizes the variance in number of sensors from \( \bar{k}_{\text{ave}} \) among all regions. Consequently, the resulting sensor deployment is globally balanced.

\( \Box \)
b). Balancing Sensor Movements: In some cases, apart from the objectives of minimizing deployment variance and minimizing movement hops, it may be necessary to balance the remaining movement capability of the sensors in the network (especially if sensors are expected to move later). Balancing the remaining mobility among sensors will translate to balancing remaining energy after final deployment. We propose a new cost assignment rule to $G^m$ that ensures that apart from minimizing variance and movement hops at final deployment, the remaining movement distance among all sensors is also balanced in the OMF algorithm. Between any two reachable regions $i$ and $j$ in $G^m$, the new cost of the edge from $v^i_{in}$ to $v^j_{out}$ is $d_{i,j} \times |E| \times H^2 + d_{i,j}^2$. The cost of edges from sink $m$ in region $i$, $v^m_{sink} \times S^v_{sink}$ for all $m$ is assigned as $-(2m - 1) \times |E| \times (|E| \times H^3 + H^2)$. Other construction rules for $G^m$ remain unchanged.

**Theorem 6.** The movement plan after executing the OMF algorithm with the modified $G^m$ will minimize the deployment variance, number of movement hops, and variance in terms of remaining movement distance among all sensors in the network (i.e., remaining movement distance of all sensors is balanced).

**Proof.** Basically, the cost between reachable regions includes $\alpha \times d_{i,j} + d_{i,j}^2$. The large value of $\alpha$ will ensure that $\sum d_{i,j}$ and $\sum d_{i,j}^2$ are simultaneously minimized, which means overall movement distance, and the variance in movement distance among all sensors is minimized. The costs of edges from $v^j_m$ to $S^v_{sink}$ is assigned such that weighted flow is still maximized, with the above cost assignment rule. The proof as such is very similar to the proof of Theorem 3. \qed
For the SPP algorithm, only now $\bar{k}$ can be set as an arbitrarily large number for balancing sensor deployment. For balancing sensor movements, each region can move the sensor with the most remaining mobility capability when requests are served to other regions.

3.6.3 Extensions to Non-Uniform Scenarios

a). Non-Uniformity at Sensor Side: We have so far assumed that all sensors are homogeneous in their mobility capacity. In many scenarios, due to deployment costs, faults in sensors etc., it may happen that only a subset of the deployed sensors is mobile. Our solutions can be extended in such scenarios. The weight assignment rule is still the same. In the OMF algorithm, we have to modify reachability information in $G_V$. For example, say only one sensor in region 2 in Figure 3.3 is mobile. Then the edges from region 2 to its reachable regions 1 and 4 (i.e., from $v^\text{out}_2$ to $v^\text{in}_1$, and from $v^\text{out}_2$ to $v^\text{in}_4$) each have capacity one. This allows upto only one sensor to move out from region 2. Other construction rules remain the same. The resulting solution is still optimal. The SPP algorithm needs no changes. Only, fewer paths will be feasible now due to not all sensors being mobile.

b). Non-Uniformity at Deployment Area Side: In our discussions above, we focused on uniform deployment areas, where $\bar{k}$ is the same for all regions. However, in many situations the deployment area can be non-uniform. Examples are certain sensitive zones that need to be sensed to a higher degree, which means $\bar{k}$ is more in such zones that others; certain hostile zones like lakes, fires etc. that can destroy sensors, which means $\bar{k} = 0$ for such zones etc. For addressing such requirements, our weight assignment rule is still the same as in equation (3.3). However, the number of
sinks created per region and their corresponding weights will be different depending on
the desired $\bar{k}$ per region in the OMF algorithm. In the SPP algorithm, the number
of requests generated and their weights are modified accordingly. The rest of our
solutions is still the same.

3.6.4 Arbitrary Sensor Movement Directions and Network
Partitions

In Section 3.2, we assumed that sensors can move only to regions in its adjacent
left, right, top and bottom directions only. We now discuss the case of arbitrary
sensor movement directions. For OMF and D-OMF algorithms, only virtual graph
$(G_V)$ construction changes. In $G_V$, we now have to add new edges (with corresponding
costs and capacities) from a region to all newly reachable regions corresponding to
arbitrary movement directions. In SPP algorithm, there are now more neighbor
choices to forward a request, and extra feasible paths can be reserved while sensors
move to satisfy requests.

In Section 3.2, we assumed that the sensor network is not partitioned. In some
situations it may happen that sensors in one part of the network may not be able
to communicate with sensors in another part. In such cases, we have to repair such
partitions, while still being constrained by mobility distance. In the approach pro-
posed by Wu and Wang [2], empty holes are filled by placing a seed from a non-empty
region to a hole. We can apply the algorithms in [2] to repair partitions in our case.
However, we are still constrained by the mobility in sensors. Addressing the issue of
repairing network partitions optimally using limited mobile sensors is a part of our
on-going work.
3.7 Performance Evaluations

In this section, we report our experimental data to study the performance of our OMF, D-OMF and SPP algorithms under various sensor and network parameters. We also simulate the well known VORonoi-based Virtual Force (VOR) algorithm proposed in [1] and compare its performance with our algorithms.

3.7.1 Performance Metrics and Evaluation Environment

a). Performance Metrics: We have three major performance metrics in this chapter. The first is the Variance Improvement (denoted by VI) at final deployment after sensors have finished movements. It is defined as \( VI = \left( \frac{Var_{in}}{Var_{out}} \right) \times 100 \), where, \( Var_{in} \) is the variance at initial deployment and \( Var_{out} \) is the variance at final deployment. Our second metric is the number of sensor Movement Hops per percent variance improvement (denoted by MH). It is defined as \( MH = \frac{M}{VI} \), where \( M \) denotes the total number of sensor movement hops. The reason we define \( MH \) as a ratio is because, it is more fair to compare number of hops per improvement in variance, than just the number of hops.

Our third metric is the messaging overhead incurred by our algorithms, which is defined as the Packet Number per region (denoted by PN). Denoting \( P \) as the total number of packets (or messages) sent, and denoting \( S \) as the number of regions, we have \( PN = \frac{P}{S} \). Physically speaking, VI captures the improvement in deployment as a result of our algorithms, while MH and PN reflect the overhead in terms of sensor movement hops and messaging overhead. The packet number for our OMF algorithm is calculated based on a simple protocol. After initial deployment, an elected region-head in each region sends a packet to Base-station (located in the center of the
network) with information on the number of sensors in its region. The packets are
forwarded along shortest paths through other regions towards the Base-station. After
the Base-station receives all packets and determines a movement plan, it sends one
packet to each region in the reverse path, informing regions of its movement plan.
A similar protocol is assumed for the $D$-$OMF$ algorithm, where the regions in each
domain will forward packets to a special sensor in the domain, which executes the
algorithm and forwards a movement plan to each region in the domain. Note that,
there can be other versions of the above protocols, like direct relaying of messages,
row-wise (or column wise) message delivery etc.

**b). Evaluation Environment:** We denote the number of regions in the
network as $n \times n$ (represented in the figures as simply $n$). Our default value is $8 \times 8$.
The default desired number of sensors per region is $\bar{k} = 3$ and maximum number
of hops a sensor can move is $H = 3$. By default, the number of sensors initially
deployed is $n \times n \times \bar{k}$, and all sensors in the network are mobile by default. For the
$D$-$OMF$ algorithm, we choose the domain size $A$ as $A = \frac{n^2}{2}$. Our implementations of
the maximum flow algorithm is the Edmonds-Karp algorithm [42], and minimum cost
flow algorithm is the one in [43]. In the $SPP$ algorithm, a peak and over-$\bar{k}$ forwarder
will batch up the coming $REQ$s in a time period to serve. In our simulation, the time
period is given by $t_u \times n$, in which $t_u$ is the message transmission delay between two
neighboring regions. For comparisons, we also simulate the VORonoi-based virtual
force ($VOR$) sensor movement algorithm [1], the basic idea of which was discussed in
Section 2.1. The termination condition for the $SPP$ algorithm is when each pit has
either obtained at-least $\bar{k}$ sensors, or expiration of sensor movement capability, or if a
certain number of requests have been tried without success by requesting sensors. By
default, the number of requests per sensor without success is set as 3. For the VOR algorithm, the termination condition was each region obtaining at-least \( k \) sensors or expiration of sensor movement capability.

We conduct our simulations on a custom simulator. For initial deployment, our simulator uses a topology generator for 2D-Normal distribution [48]. A 2D-Normal distribution involves two random variables, \( x \) and \( y \) with mean values \( \mu_x \) and \( \mu_y \). The mean values corresponding to each variable can be written as a vector \( \mathbf{u} = (\mu_x, \mu_y)^T \). Each variable will have a variance \( \sigma_x \) and \( \sigma_y \). However, it may happen that the variables are related to each other, in which case there will be covariances \( \sigma_{xy} \) and \( \sigma_{yx} \) with \( \sigma_{xy} = \sigma_{yx} \), all of which can be incorporated into a variance-covariance matrix: \( \mathbf{v} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{pmatrix} \). The 2D-Normal distribution is then given by \( P(\mathbf{z}) = \frac{1}{2\pi\sqrt{|\mathbf{v}|}} e^{-\frac{1}{2}(\mathbf{z}-\mathbf{u})^T\mathbf{v}^{-1}(\mathbf{z}-\mathbf{u})} \) where \( |\mathbf{v}| \) is the determinant of \( \mathbf{v} \). In our simulations, we set \( \sigma_{xy} = \sigma_{yx} = 0 \), which means the location at \( x \) and \( y \) axis are independent when sensors are deployed. We let \( \sigma^2 = 1/\sigma_x^2 = 1/\sigma_y^2 \). Hence, when \( \sigma \) increases, sensors will be more concentrated at the center, and when \( \sigma \) tends to 0 sensors are more uniformly distributed. For our simulations, by default, \( \sigma = 4 \). All data reported here were collected across 10 iterations, and averaged \(^{13}\).

### 3.7.2 Performance Results

**a). Performance comparison of all algorithms:** We first study the sensitivity of \( VI \), \( MH \) and \( PN \) to mobility capacity \( H \) for the \( OMF \), \( D-OMF \), \( SPP \) and \( VOR \) algorithms. We assume that the sensors are initially deployed as a one time step targeted towards the center of the network. All other settings are default. From Figure 3.4, we observe that the \( OMF \) algorithm (being optimal) performs best in terms of \( VI \).

\(^{13}\)The standard deviations for all data sets are shown in the corresponding figures.
We observe that the $D$-$OMF$ algorithm performs quite close to the $OMF$ algorithm in all cases, while the performance of the $SPP$ and $VOR$ algorithms are quite good for smaller values of $H$. Among all algorithms, the virtual force ($VOR$) algorithm has the poorest performance. Since sensors in the $VOR$ algorithm attempt to achieve local force balance between themselves, they incur several back and forth movements that rapidly depletes overall sensor movement capability resulting in overall poor variance improvement. Figure 3.4 also shows the effects of limited mobility on $VI$. When $H$ increases, $VI$ increases in all algorithms as increased movement ability in general helps to move sensors to needy regions farther away. The improvement stays constant when $H \geq 4$ for the $OMF$ algorithm, and for $H \geq 3$ in the other algorithms. This demonstrates that mobility beyond a certain point does not bring in further benefits. Note that when $H \geq 4$, the $OMF$ algorithm achieves the upper bound of 100% $VI$, because movement choices can be optimally exploited by the $OMF$ algorithm with larger $H$, unlike other algorithms.

Figures 3.4 and 3.5 show that when $VI$ increases, $MH$ also increases for the $OMF$, $D$-$OMF$ and $SPP$ algorithms. In our algorithms, each sensor movement typically results in a more balanced deployment for the network. Also, it could happen that for improvement in variance, many potentially longer paths are found in our algorithms. Consequently, when $VI$ increases $MH$ also increases in our algorithms, while staying a constant when $VI$ stops increasing. On the other hand, we observe that the $MH$ for the $VOR$ algorithm is initially a higher value, then decreases and stays constant. When $H$ is small, $VI$ is quite low for the $VOR$ algorithm, due to more stringent mobility limitations, which makes $MH$ higher for smaller $H$. As $VI$ improves further, $MH$ decreases for the $VOR$ algorithm in Figure 3.5. Since $VI$ stays constant beyond
Figure 3.4: Sensitivity of $VI$ to $H$ for all four algorithms

Figure 3.5: Sensitivity of $MH$ to $H$ for all four algorithms

Figure 3.6: Sensitivity of $PN$ to $H$ for all four algorithms
$H > 2$, $MH$ also stays constant. Note that when $H$ increases, the $MH$ in the $D$-$OMF$, $SPP$ and VOR algorithms are quite close to the optimal $OMF$ algorithm (while even being smaller in some cases). As pointed before, this is because of the improved $VI$ that can be achieved by the $OMF$ algorithm, by exploiting several additional movement choices compared to the other algorithms, which increases the number of movements, and hence $MH$ in the $OMF$ algorithm.

In Figure 3.6, we can see that $PN$ in the $OMF$ and $D$-$OMF$ algorithms are constant, since packet number does not depend on $H$ for these algorithms. The messaging overhead in the VOR algorithm is the maximum because of many local message exchanges caused by several back and forth sensor movements. The $PN$ in $SPP$ algorithm takes a middle ground, since only deficient regions send out requests and that too towards richer regions only, while responses always aim to take shorter movement paths. In both $SPP$ and VOR algorithms, $PN$ increases with $H$ due to more movement choices when $H$ increases.

Figures 3.7, 3.8 and 3.9 show the sensitivity of our performance metrics to $\sigma$ for our algorithms. In Figure 3.7, $VI$ decreases when $\sigma$ increases for the $OMF$ and $SPP$ algorithms. Since larger $\sigma$ implies more concentrated initial deployment, it is harder for regions near the boundary to find sensors under mobility constraints, which decreases $VI$ for the $OMF$ and $SPP$ algorithms. We also see that $VI$ of the $D$-$OMF$ and $SPP$ algorithm become closer to that of the $OMF$ algorithm as $\sigma$ increases. This is because, the amount of mobility choices that the $OMF$ algorithm can exploit is not significantly more than that of the other algorithms, when deployment is highly concentrated.
In Figure 3.8, we see that $MH$ increases as $\sigma$ increases for our algorithms. This is mainly because of the reduction in $VI$ with increasing $\sigma$. Note here that $MH$ is lower for the $SPP$ algorithm when $\sigma$ is small. This is because, when deployment is more uniform (smaller $\sigma$), more pits can find enough over-$\bar{k}$ forwards or peaks nearby, which causes a reduction in overall sensor movements. In Figure 3.9, we see that the $PN$ for the $OMF$ and $D-OMF$ algorithms decreases with $\sigma$, since the number of sensors farther away from the center of the network decreases with increased $\sigma$. On the other hand, $PN$ increases with $\sigma$ in the case of the $SPP$ algorithm, since the increase in $\sigma$ means that the bias increases, resulting in more requests and responses.

We also observe that when $\sigma$ is less, the $PN$ in the $SPP$ algorithm is lower than that of the $OMF$ and $D-OMF$ algorithms. This is because, more pits can find enough over $\bar{k}$ forwards or peaks in the $SPP$ algorithm when the deployment is more uniform, further highlighting the fact that the distributed $SPP$ algorithm achieves less overhead under favorable deployment conditions.

We now study the sensitivity of our performance metrics to $\bar{k}$ for our algorithms. In order to compare the sensitivity to $\bar{k}$ fairly, the number of sensors initially deployed is fixed as $8 \times 8 \times 3 = 192$ for all cases (all other settings are default). From Figures 3.10 and 3.11, we can see that an increase in $\bar{k}$ causes a decrease in $VI$ and an increase in $MH$ in our algorithms. When $\bar{k}$ increases, the objective becomes harder, which causes this trend. We can also see that the $D-OMF$ algorithm performs quite close to the $OMF$ algorithm in all cases. An interesting observation here is that, when $\bar{k}$ is small, the performance of the $SPP$ algorithm in all metrics compares quite favorably with the other algorithms. This is because, when the deployment objective is relatively mild (less $\bar{k}$), local requests and responses suffices for good performance. Once again,
Figure 3.7: Sensitivity of $VI$ to $\sigma$

Figure 3.8: Sensitivity of $MH$ to $\sigma$

Figure 3.9: Sensitivity of $PN$ to $\sigma$
the $PN$ for the $OMF$ and $D-OMF$ algorithms in Figure 3.12 is independent of $\bar{k}$ and hence is constant. The $PN$ of the $SPP$ algorithm is similar to the other algorithms for less $\bar{k}$, and increases with increasing $\bar{k}$ since more requests and responses are generated when $\bar{k}$ increases.

In Figures 3.13, 3.14 and 3.15, we can see that as $n$ increases, $VI$ decreases and both $MH$ and $PN$ increase for our algorithms. A larger $n$ implies a larger network, which makes more regions near the boundary of the network unable to get sensors, and thus $VI$ decreases. Also, sensors need to travel longer distances, which increases $MH$ and $PN$.

b). Performance when only a subset of sensors are mobile: Our default case above consisted of all sensors in the network as capable of being limitedly mobile. We now demonstrate the sensitivity of performance ($VI$) of our algorithms under different sensor mobility capacity ($H$) when only a subset of sensors is mobile. The value of $H$ is set as 3. All other settings are default. In Figure 3.16, the term $P_r$ on the X-axis denotes the percentage of sensors that are mobile. For instance if $P_r = 0.2$, then only 20% of the sensors are mobile. We observe that while $VI$ improves with increasing $P_r$, there is a threshold beyond which increase in $VI$ is negligible in all algorithms. For the case when $H = 3$ in Figure 3.16, the threshold is around 60%. The threshold in fact depends on $H$ and decreases as $H$ increases and vice versa $^{14}$. This demonstrates that, not all sensors in the network need to be mobile. Depending on the mobility capacity $H$, there is a threshold beyond which deployment quality cannot be enhanced significantly with more mobile sensors.

$^{14}$We do not report data for other values of $H$ due to space limitations.
Figure 3.10: Sensitivity of $VI$ to $k$

Figure 3.11: Sensitivity of $MH$ to $\bar{k}$

Figure 3.12: Sensitivity of $PN$ to $\bar{k}$
Figure 3.13: Sensitivity of $VI$ to $n$

Figure 3.14: Sensitivity of $MH$ to $n$

Figure 3.15: Sensitivity of $PN$ to $n$
Figure 3.16: Sensitivity of $VI$ to $Pr$

Figure 3.17: Sensitivity of convergence time to $H$ in the $SPP$ and $VOR$ algorithm
c). Convergence time of the *SPP* and *VOR* algorithms: In Figure 3.17, we study the sensitivity of convergence time of the *SPP* and *VOR* algorithms to $H$. All other settings are default. For the *SPP* algorithm, the convergence time (in terms of rounds) is obtained as follows. We denote one unit time as the time taken by a sensor to perform local computations and send a packet to a sensor in a neighboring region. The number of rounds is the total number of unit times spent to complete execution of the *SPP* algorithm in the network. For the *VOR* algorithm, it is simply the number of rounds it takes for the algorithm to terminate similar to the definition in [1]. From Figure 3.17, we see the convergence time increases with $H$ for both algorithms, since more movement choices are available with increasing $H$. We also observe that the convergence time begins to saturate with increasing $H$, demonstrating that beyond a certain point, increase in mobility does not help deployment much. Note that the number of rounds in the *VOR* algorithm is much lower than the *SPP* algorithm. However, this should not be construed as better performance by the *VOR* algorithm. Rather, it is due to the faster depletion of sensor mobility capacity, and the lower $VI$ in the *VOR* algorithm compared to the *SPP* algorithm. The sensitivity of number of rounds to $\tilde{k}$, $\sigma$ and $n$ follow expected trends and are not reported here (due to space limitations).

### 3.8 Summary

In this chapter, we defined a general sensor networks deployment problem for $\bar{k}$- coverage ($\bar{k} \geq 1$) under limited mobility flip-based sensors, and proposed a set of sensor movement algorithms for it. We proposed a weight-based methodology that translates our non-linear variance objective into a linear one. Based on this
methodology, we designed a centralized optimum algorithm and a distributed heuristic algorithm for the deployment problem. We demonstrated the performance of our algorithms using extensive analysis and simulations.
CHAPTER 4

MODELING MOBILE AGENT-ORCHESTRATED PHYSICAL ATTACKS IN WIRELESS SENSOR NETWORKS

In this chapter, we study the issue of physical attacks in sensor networks being orchestrated by an external mobile agent. We define a representative model of physical attacks and a metric to quantify attack impacts. Extensive simulations are conducted to demonstrate the destructive impacts of physical attacks in sensor networks. We also discuss variations of physical attacks and propose countermeasure guidelines to defend sensor networks against them.

4.1 Motivations

A critical component of on-going research in wireless sensor networks is the security. Research in this area has contributed a host of potential attacks in sensor networks and effective defenses against such attacks [49–56]. It is widely accepted that viability of sensor network applications in the future is closely contingent on the security of the networks.

The small form factor of sensors, coupled with the unattended and distributed nature of their deployment expose sensor networks to a special class of attacks that could result in the physical destruction of sensors. We denote Physical Attacks as
those that result in the physical destruction of sensors, thereby rendering them per-
manently nonoperational. The significance of studying physical attacks comes from
the following factors. Physical attacks are inevitable threats in sensor networks. Phys-
ical attacks are relatively simple to launch and fatal in destruction. In the simplest
case, the attacker can just drive a vehicle in the sensor field or hurl grenades/bombs in
the field and destroy the sensors. A smarter attacker can detect and destroy sensors
with stealth by moving across the sensor network. In any case, the end result of phys-
ical attacks can be quite fatal. The backbone of the network (the sensors themselves)
is destroyed. Destruction of sensors may also result in the violation of the important
network paradigms. A wide spectrum of impacts may result due to physical attacks
and when left unaddressed, physical attacks have the potential to render the entire
sensor network mission useless.

Our Contributions: In this chapter, we consider a representative model of phys-
ical attacks orchestrated by a malicious mobile agent. We call such attacks as search-
based physical attacks, wherein a malicious mobile attacker moves in the deployment
field searching for sensors, and then reaches the sensors to physically destroy them.
The attacker’s searching process is executed by using its mobility and by means of
detecting electronic, magnetic, heat signals emitted by the sensors. Once sensors are
identified, the attacker localizes the sensors by means of the signals received, and then
physically reaches the sensors to destroy them. We consider both flat and hierarchical
sensor networks. The attacker in our model uses a weighted random selection based
approach to discriminate multiple target choices (normal sensors and cluster-heads)
during the destruction phase. The discrimination is due to the attacker’s objective
of maximizing performance degradation by *efficiently* identifying and then destroying sensors. We point out that this attack model is opposed to a rather blind or brute force destruction of sensors in the field (using bombs, grenades, tanks etc) that are easy to be detected. Also, such brute-force attacks may cause casualties to the deployment field, which the attacker might want to preserve (airports, oil fields, battlefields etc. of interest to the attacker). The salient features of search-based physical attacks come from the ability to stealthily search for and then destroy sensors. This improves the efficiency of the attack process, as the attacker can also identify and destroy only important sensors (cluster-heads). In such a manner the search-based attacker can cause maximum destruction to the sensor network while causing minimum casualties to the deployment field. Furthermore, the attacker can use mobility as a leverage to significantly enhance the geographical scope of sensor destruction, while simultaneously making it hard to localize the attacker and its impacts.

We first define a representative model of search-based physical attacks in sensor networks. We then conduct a detailed performance analysis of impacts of search-based physical attacks in sensor networks, under varying attacker and sensor network parameters. Our performance metric in this chapter is the *Accumulative Coverage* of the network. Accumulative Coverage captures both the lifetime and coverage and as such is an effective metric to measure performance. Our performance data clearly show that search-based physical attacks dramatically reduces accumulative coverage, further highlighting the importance of our work. We observe that the attacker parameters, namely attacker’s mobility, signal detection ranges, accuracy; and the sensor network parameters such as the frequencies of sensors’ beacons and frequency
of cluster-head rotation have significant impacts on attack effectiveness. We also observe that using a hierarchical sensor network has a mixed impact on maintaining $AC$ under attacks; on one hand, the compromised cluster-heads (in hierarchical networks) cause decrease in $AC$ significantly; on the other hand however, the existence of cluster-heads with cluster-head rotation can mislead the attacker and make its movements less efficient. We also observe that the impacts of the attacker and sensor network parameters to attack effectiveness interact with each other to affect $AC$.

4.2 Modeling Mobile Agent-Orchestrated Physical Attacks

In this section, we will discuss the search-based physical attack model in detail. The objective of the attacker is to detect and physically destroy sensors in the network to compromise network performance (We define our performance metric subsequently). Modeling search-based physical attacks involves modeling the sensor network properties (from the perspective of signals emitted by the sensors), and modeling the attacker capacity (from the perspective of mobility, signal detection and sensor destruction) in compromising performance. In the following, we first classify signals emitted by sensors that can be detected by the attacker. We then discuss attacker capacities from the perspective of mobility, signal detection and sensor destruction methods. Following this will be our detailed search-based physical attack model.

4.2.1 Sensor Network Signals

We classify sensor signals that can be detected by the attacker into two types, namely $Passive$ signals and $Active$ signals. Passive signals include heat, vibration,
magnetic signals etc., that are part of the physical characteristics of the sensors. 
Active signals on the other hand include communication messages, beacons, query messages etc., that are part of normal communications between sensors in the network. The attacker can detect both passive and active signals to identify sensor locations. However, the distance within which an active signal can be detected is larger than the distance for detecting a passive signal because the active signal can propagate larger distance. We assume destroyed and sleeping sensors do not emit active nor passive signals.

In this chapter, we model search-based physical attacks in flat and hierarchical networks. When the sensor network is flat, the detectability of active signals and passive signals (by the attacker) are same for all sensors. In a hierarchical network, we model the upper tier sensors (denoted as cluster-heads) as having higher active signal transmission strengths compared to the lower tier normal sensors. Generally speaking, this is due to cluster-heads having to communicate with other cluster-heads, base-station etc., that may be far away from a cluster-head. That is, the attacker can detect active signals emitted by cluster-heads from a larger distance, compared to the distance within which it can detect an active signal emitted by a normal sensor. The detectability of passive signals (due to physical characteristics) is still the same for cluster-heads and normal sensors. The sensors rotate among themselves to periodically elect cluster-heads. We denote the frequency of cluster-head rotation among sensors as $\mu$, which is the reciprocal of the interval between cluster-head elections. The size of the cluster (the average number of sensors in a

\[ \text{We assume that the attacker cannot visually identify sensors.} \]
cluster) is denoted as $C$. In this chapter, we assume the attacker does not have the capability to reach or destroy the base station.

4.2.2 Attacker Capacities

We now discuss the capacities of the search-based attacker in our model.

*Signal Detection and Sensor Isolation:* The features characterizing this action include, target of search, method of search and capacity of search. In our model, the target can be normal sensors or cluster-heads. While the attacker can detect multiple sensors, the *target* denotes the particular sensor that the attacker currently chooses to destroy. The searching method used by the attacker is by detecting signals emitted by sensors. The search capacities of the attacker are the distance within which signals can be detected (also called detection range) and detection accuracy. We discuss further below.

The first parameter to model signal detection capacity is the range of detection of Passive and Active signals. Both types of signals are quite different in terms of their detectability by the attacker. Passive signals are detectable from only small distances. We denote $R_{ps}$ as the maximum distance within which the attacker can detect a passive signal (for both normal sensors and cluster-heads). We denote $R_{as}^{s}$ as the maximum distance within which the attacker can detect an active signal emitted by a normal sensor, and $R_{as}^{h}$ as the maximum distance within which the attacker can detect an active signal emitted by a cluster-head. Since cluster-heads send out higher strength active signals, we have $R_{as}^{s} < R_{as}^{h}$. Since, passive signals are detectable from smaller ranges compared to active signals, we also have $R_{ps} < R_{as}^{s} < R_{as}^{h}$. We emphasize here that the attacker may or may not have the ability to distinguish
between normal sensors and cluster-heads. We discuss both scenarios when we present our attack model subsequently.

The second parameter to model signal detection capacity is the detection accuracy. Once a signal from a sensor (say $S_i$) is detected, the attacker will attempt to locate the sensor ($S_i$). To do this, the attacker needs to estimate the distance ($d_i$) from its current position to the sensor, and the orientation or the arrival angle of the detected signal [57, 58]. Approaches discussed in [57, 58] can be used for this purpose. The attacker can only isolate the sensor’s location within a certain area because its estimation of the sensor’s location is not accurate. Defining $r_i$, as the maximum distance between the sensor’s actual location and its location estimated by the attacker. The area isolated is a circle with radius $r_i$ and the center of it is the estimated location of the detected sensor. In order to determine $r_i$, the attacker will make use of the detection error, $\theta$, when it detects the orientation of the received signal [57, 58] and the estimated distance of the signal source (a sensor) as following,

$$r_i = d_i \times \theta.$$  \hspace{1cm} (4.1)

We can see the accuracy of the attacker in determining the sensors location is inversely proportional to $\theta$. That is, if $\theta$ is small, the accuracy is higher ($r_i$ is small). Similarly, a large $\theta$ means larger $r_i$, which consequently means less accurate isolation. Hence, we use $\theta$ to measure the detection accuracy of the attacker. We call the area ($= \pi r_i^2$) as the *sweeping area* for sensor $S_i$. The attacker will proceed to sweep this area if it wants to destroy this sensor.

*Attacker Mobility:* The search-based physical attacker is mobile. Basically, mobility is a leverage for the attacker to enhance the geographical scope of sensor destruction, while simultaneously making it hard to localize the attacker and its impacts. In
the absence of a target, the attacker walk can be random (while performing searching). We denote $v_{mv}$ as the average movement speed of the attacker. If there is only one target, naturally the attacker moves towards it. In our model, if the attacker detects multiple signals, it will store the estimated locations of the signal sources and the corresponding sweeping areas in memory. The memory available to the attacker is large enough in our model to keep all identified locations. When there are multiple targets, the attacker needs to schedule its movement to reach targets efficiently. In our model, this behaviors is captured using the notion of weights (discussed in Section 4.2.3) assigned to each detected sensor by the attacker. We also assume that the attacker has the ability to detect boundaries of the sensor network, and can hence stay within the network area as long as it is attacking the network.

Sensor Destruction Methods: The method used by the attacker to physically destroy sensors can be by means of applying physical force, radiation or it can be by means of other hardware/circuit tampering techniques. Recall that the attacker identifies a sweeping area for destruction. Denoting $v_{sw}$ as the sweeping speed, i.e. the size of area that the attacker can sweep per second, and $B_i$ as the sweeping area for sensor $S_i$, we have the time taken for sweep ($t_{sw}^i$) of this sweeping area as,

$$t_{sw}^i = \frac{B_i}{v_{sw}}$$  \hspace{1cm} (4.2)

In our model, the attacker has to physically reach sensors to destroy them. The attacker does not have the capacity to launch remote destruction of sensors.

4.2.3 Description of the Physical Attack Model

a). Attacker Objective: The objective of the attacker is to identify and destroy sensors with the intention of compromising sensor network performance. We define
a novel performance metric in this chapter, namely Accumulative Coverage ($AC$). $AC$ is defined as the integration of the network coverage over the effective lifetime of the sensor network. Network coverage is defined as the percentage of the sensor field that is in the sensing range of at least one active sensor\textsuperscript{16}, and effective lifetime is the time period until when the sensor network becomes nonfunctional because the coverage falls below a certain threshold $\alpha$ (that is system desired). Denoting $\text{coverage}(t)$ as the network coverage at time $t$, and $EL$, as the effective lifetime, we have,

$$AC = \int_{t=0}^{EL} \text{coverage}(t)dt.$$  

(4.3)

We believe that $AC$ is an effective metric to measure the performance of a sensor network in many situations since it effectively combines both coverage and lifetime, two of the most important performance metrics in sensor networks. Effective lifetime is defined as the maximum time period during which the coverage is above a certain threshold and thus considers both coverage and lifetime. However, it is not representative enough for situations where for the same effective lifetime, a sensor network with a higher coverage can provide more accurate information than one with a lower coverage. Our metric in this chapter, $AC$ considers both coverage and lifetime, and is hence representative of real-life situations. The degradation of $AC$ measures the effectiveness of the search-based physical attacks because its objective is to compromise sensor network performance.

b). The Attack Model: Algorithm 4 describes our search-based physical attack model. The attacker’s behavior in the network is primarily event driven. That is, the attacker responds to different events in the network, depending on the\textsuperscript{16}We consider 1-coverage in this chapter.
Algorithm 4 Search-based physical attack model

1: Initialization: Target ← Φ; Mem ← Φ;
2: while the attacker is executing attacks do
3: switch type of event
4: case detected a sensor S:
5: Target ← S;
6: Calculate S.weight; Add S to Mem; Target ← S;
7: Target.location ← S.location;
8: Target ≠ Φ:
9: add S to Mem; Update S_i.weight ∀ S_i ∈ Mem;
10: Randomly choose S' ∈ Mem and the chance to choose S_i
11: Target ← S'; Target.location ← S'.location;
12: case reached Target.location:
13: Destroy Target;
14: case finished the destruction of Target:
15: Remove Target from Mem;
16: Update S_i.weight ∀ S_i ∈ Mem;
17: Randomly choose S' ∈ Mem and the chance to choose S_i
18: is proportional to S_i.weight;
19: Target ← S'; Target.location ← S'.location;
20: default:
21: Whenever Target ≠ Φ, walk towards Target.location,
22: otherwise walk randomly;
23: endswitch
24: end while

event occurred. Broadly, there are three types of events in Algorithm 4. The first is signal detection and choice of a sensor as target. The second is reaching the target and destroying it. The third event is choosing a new target after destruction of the current target. The events are overlapped in time and need not be sequential. An instance is the attacker can detect a signal, while it is moving towards a target.

At the start in Algorithm 4, the attacker does not have any detected sensor to destroy. Here, the attacker performs a random straight line walk (with an average speed \( v_{mv} \)) in the network field and keeps detecting signals. Once the attacker detects one or more signals (Event: detecting a sensor), the attacker detected one or more sensors. While the attacker has only one detected sensor, the attacker will move towards this target until it reaches the target (Event: reaching Target location), and destroy the target. However, when multiple sensors are detected (Event: detecting a
sensor), or when a new sensor is detected during moving to current target, or when the attacker has multiple sensor locations in memory and it has just finished destroying a target (Event: \textit{finish the destruction of target}), the attacker has to efficiently \textit{choose} the next target.

The issue here is how the attacker should respond when it has multiple detected sensors (normal sensors and cluster-heads), detected by means of active or passive signals, with possibly different times taken to destroy each, in order for the attacker to achieve its objective of compromising \( AC \) as much as possible. In this chapter, we model the attacker as using a random weighted selection approach to choose one particular sensor as target among multiple detected sensors. We model this using the notion of \textit{weights} assignment for each detected sensor. In this model, the attacker assigns a weight for each detected sensor depending on the time taken to reach and destroy it and its importance (discussed further next) in compromising \( AC \). Once weights are assigned, the attacker randomly chooses a sensor as the target, where the probability of choosing a particular sensor as the target is proportional to the weight of that sensor. That is, larger the weight is for a sensor, higher is the probability that the attacker chooses that sensor as the target.

There can be other approaches to choose targets. One approach is to always choose the sensor with the largest weight as target. Another approach is for the attacker to construct best paths for destroying multiple sensors it has detected. While the above approaches seem to be intuitively reasonable, they are not the best choices for the attacker. Purely choosing targets by decreasing order of weights, may not be best in the long run. In the second approach, the attacker may detect new sensors during destruction, which it may not be able to use efficiently.
c). **Weight Determination:** The determination of weight for each detected sensor, is a critical component in attacker success. Determination of the weight for a particular detected sensor is related to the following; current distance from the attacker to the detected sensor, whether the sensor is a normal sensor or a cluster-head, accuracy of isolation of the sensor.

We propose two approaches for determining the weights. The first strategy models attackers that aim to maximize performance degradation by destroying sensors that take less time to destroy (irrespective of whether the attacker’s can or cannot distinguish between cluster-heads and normal sensors). An intuitive notion of weight is related to the time taken to reach and destroy a sensor. Let us denote $d_i$ as the estimated distance between the current location of the attacker and the location of a detected sensor $S_i$\(^{17}\). Recall that $B_i$ is the sweeping area for sensor $S_i$. Let $t_i$ denote the total time taken to reach and destroy sensor $S_i$. This includes the time to move to sensor $S_i$ and the time to sweep the sweeping area for sensor $S_i$. We then have,

$$t_i = \frac{d_i}{v_{mv}} + \frac{B_i}{v_{sw}},$$

where $v_{mv}$ and $v_{sw}$ denote the attacker’s movement speed, and sweeping speed respectively (defined in Section 4.2.2).

We then have the weight $w_i$ for sensor $S_i$ as,

$$w_i \propto \frac{1}{t_i}. \quad (4.5)$$

The second weight approach models the attacker giving importance to cluster-heads if it has the ability to distinguish between cluster-heads and normal sensors.

\(^{17}\)The distance $d_i$ estimated by the attacker for a sensor $S_i$ will change as the attacker keeps moving.
That is if the target is a cluster-head, the weight for the cluster-head is given by,
\[ w_i \propto \frac{H}{t_i}, \]  \hspace{1cm} (4.6)
where \( H \) denotes the importance of a cluster-head from the attackers perspective. If the attacker has some knowledge of the number of children sensors per cluster-head, then \( H \) can be set depending on this value. However, such knowledge is not easy for the attacker to obtain. \( H \) can be set as a very large value, this means that the attacker gives high priority to cluster-heads compared to normal sensors.

We emphasize here that calculating weights for sensors in memory is not a one time operation. According to the definition of weight, whenever the attacker moves, the distances to detected sensors are changed, hence the weights of them are changed and need to be updated/recalculated. The frequency with which weights are updated has trade-offs between computational overhead and accuracy of weights. However, the attacker only needs most updated weights of un-destroyed detected sensors when it needs to choose or change target. Thus, weights are only need to be updated if one or more of the following events happen; the attacker destroys a target; new sensors are detected; new signals are received for an already detected sensor. The occurrence of any of the above events not only means that the attacker needs to choose or change target, it also means that the number of candidates of target changes or the sweeping area of an already detected sensor should be updated. In our model, the attacker updates weights following equation 4.4, 4.5 and 4.6 when any of above events happens. We now discuss how to update weighs, when new signals are received for a sensor already detected and isolated.

From equation 4.4, one can observe that the time (\( t_i \)) to reach and destroy a sensor \( S_i \) is proportional to the sweeping area size. The determination of weight \( w_i \) is
inversely proportional to the time \( t_i \). We propose an approach by which the attacker can smartly reduce the sweeping area size and save time, when multiple signals are detected for sensor \( S_i \). For each signal received, the area isolated by the attacker for sensor \( S_i \) changes. Thus, the attacker has multiple sweeping areas for the same sensor. The attacker can save time for destroying this sensor by sweeping only the small area that overlaps with all identified sweeping areas (identified per each signal received) for this sensor. That is, the overlapped area is the new sweeping area for this sensor. The novelty in this approach is that the attacker has minimized the time required to sweep the area in order to destroy this sensor. This will enhance the potency of the attacker, further compromising performance.

### 4.3 Performance Evaluations

In this section, we report our performance evaluations of the impacts of search-based physical attacks on sensor networks. Our performance metric here is the Accumulative Coverage (\( AC \)), and the search-based attack model is the one described in Section 4.2. We will study the sensitivity of both attacker features, and sensor network features to \( AC \) under search-based physical attacks.

#### 4.3.1 Evaluation Environment

Our sensor network is a field of size 500 m \( \times \) 500 m. In the field, 1000 sensors are randomly uniformly deployed. Sensors emit active signals with a rate denoted as \( f \). The network can be flat or can be hierarchical and divided into clusters. Each cluster has a cluster-head to which all its children sensors send data. Sensors rotate among themselves to periodically elect new cluster-heads. The attacker initially performs a random straight line walk in the network searching for sensors.
Unless otherwise stated, following are the values of specific sensor network and attacker parameters used in the simulations. Active signal frequency, \( f = \frac{1}{30 \text{ seconds}} \); cluster head rotation frequency, \( \mu = \frac{1}{500 \text{ seconds}} \); cluster size (average number of sensors in a cluster), \( C = 10 \); passive signal detection range, \( R_{ps} = 1 \text{ meter} \); active signal detection range of cluster-heads, \( R_{as}^{h} = 50 \text{ meters} \); active signal detection range of non-cluster-head sensors, \( R_{as}^{s} = 20 \text{ meters} \); detection error of the signal arrival angle, \( \theta = 1/10 \text{ radian} \); attacker moving speed, \( v_{mv} = 0.5 \text{ meter/second} \); attacker sweeping speed, \( v_{sw} = 0.25 \text{ meters}^2/\text{second} \); minimal coverage requirement of the sensor network, \( \alpha = 50\% \) (\( \alpha \) is used in determining effective lifetime and \( AC \)). Each point of data in the following figures is the average of results from simulations on 5 different random network topologies.

We consider two attack scenarios in our evaluations. The scenario \( PA/H \) denotes one where the attacker gives priority to cluster-heads, in which case \( H \) can be a very large number (in equation 4.6). The scenario \( PA/NH \) denotes one where the attacker does not give priority to cluster-heads or the attacker cannot distinguish cluster-head, in which \( H = 1 \) in weight calculation.

4.3.2 Performance Results

We first report data to study the impacts of search-based physical attacks on sensor network performance under varying attack features. We then report data to study the impacts of search-based physical attacks on \( AC \) under varying sensor network features.

a). Sensitivity of \( AC \) to Attacks under different Attacker Capacities (Attacker Movement Speed, Detection Range and Accuracy): Figure 4.1
(a) shows the impacts of attacks on $AC$ with varying attacker movement speed ($v_{mv}$) under different detection accuracy ($\theta$). We report data for both $PA/H$ and $PA/NH$ scenarios here. Irrespective of $\theta$ and whether cluster-heads are given priority, we can see that as $v_{mv}$ increases, $AC$ decreases. The decrease in $AC$ is very sharp in the left portion of the curve, which represents the transition from no attack to slight increases in attack speed. Beyond a certain point, further increases in $v_{mv}$ does not affect $AC$ significantly. The reason is that when $v_{mv}$ is small, any change in speed distinguishes a static attacker from a mobile attacker considerably. The consequence is a large difference on the attack effectiveness, which causes the steep fall in $AC$.

This clearly demonstrates the impacts of mobility for physical attackers in terms of destroying sensors. However, when $v_{mv}$ is large, most of the sensors are destroyed in a short time and the sensor network becomes sparse quickly. Thus, there is less room for the attacker to cause sensor destructions, and the fall in $AC$ is less pronounced with further increases in movement speed.

The second observation is that $AC$ is sensitive to $\theta$, and $AC$ decreases when $\theta$ decreases. When $\theta$ decreases, the isolation error decreases based on equation 4.1, and the sweeping area is smaller. Consequently, the attacker can detect sensors more accurately and destroy them faster, naturally composing $AC$ better.

The third phenomenon we can study from Figure. 4.1 (a), is the sensitive of $AC$ to whether cluster-heads are given priority ($PA/H$) or not ($PA/NH$). We observe that when $\theta$ is large ($\theta = 1/5$), $PA/NH$ compromises $AC$ more compared with $PA/H$. However, when $\theta$ is very small ($\theta = 1/20$), $PA/H$ compromises $AC$ better than $PA/NH$. When $\theta$ is large, the isolation of sensors and cluster-heads is not accurate and the resulting sweeping areas are large. Since the signal detection range
of cluster-heads is larger than that of normal sensors \( (R_{as}^h > R_{as}^s) \), the cluster-heads can be detected far away. Thus, the sweeping areas of cluster-heads are much larger that those of normal sensors (from equation 4.1). Consequently, giving priority to cluster-heads \( (PA/H) \) cost significant amount of time for the attacker (from equation 4.4) to destroy a cluster-head. Thus, when \( \theta \) is large, giving priority to cluster-heads \( (PA/H) \) is not beneficial for the attacker. On the other hand, when \( \theta \) is small, the isolation of sensors and cluster-heads is very accurate and the corresponding sweeping areas are small. Under this situation, \( PA/H \) is beneficial for the attacker, since it can destroy cluster-heads faster and can compromise \( AC \) better.

![Figure 4.1](image_url)

**Figure 4.1**: Sensitivity of \( AC \) to attacker capacities, \( v_{mv}, \theta, R_{as}^s \) and \( R_{ps} \).

Figure 4.1 (b) shows the sensitivity of \( AC \) to attackers signal detection ranges \( (R_{ps} \) and \( R_{as}^s) \) under both \( PA/NH \) and \( PA/H \) scenarios. Irrespective of \( R_{as} \) and \( PA/NH \) or \( PA/H \) scenarios, we can see that as \( R_{as}^s \) increases, \( AC \) decreases.

The second observation we make from Figure 4.1 (b) is the relationship between \( R_{as}^s, PA/NH \) and \( PA/H \) scenarios on \( AC \). We observe that, when \( R_{as}^s \) is small,
PA/H is more effective in compromising AC, and when $R_{as}^s$ is large, PA/NH is more effective. This is because when $R_{as}^s$ is small, the attacker can detect very few nearby normal sensors. In this case, attacking cluster-heads in the local area is more effective to compromise AC. Consequently, PA/H is better for the attacker. When $R_{as}^s$ is large, the attacker can detect many normal sensors within a large range. In this case, giving priority to detected normal sensors (which are many in number) is more effective. Thus, when $R_{as}^s$ is large, PA/NH compromises AC better. The sensitivity of AC to $R_{as}^h$ with different $R_{ps}$ follows a similar pattern and we do not report corresponding performance results.

b). Sensitivity of AC to Attacks under different Sensor Network Parameters (Active Signal Frequency, Cluster-head Rotation Frequency and Cluster Size): Figure 4.2 (a) shows the impact of attack on AC when cluster-head rotation frequency ($\mu$) changes, with different cluster sizes ($C$) under both PA/NH and PA/H scenarios. Intuitively speaking, more frequent cluster-head rotation is better under search-based attacks, since destroyed cluster-heads can be replaced faster. Thus, when $\mu$ increases, AC increases as we observe in Figure 4.2 (a). The second observation here is the relation between the choices of PA/NH and PA/H scenarios with different $\mu$. When $\mu$ is small (slower rotation), giving priority to cluster-heads is better, and reduces AC more significantly compared to PA/NH. When $\mu$ is larger, cluster-heads are replaced rapidly. In this case, giving priority to cluster-heads is less beneficial for the attacker, hence PA/NH compromises AC more in this case.

The third observation we make is that, when cluster size ($C$) is small, AC is generally larger. This is because, a small cluster size means, less impacts on AC when
the cluster-head is destroyed. From this figure, we can also see the performance trade-offs between flat and hierarchical sensor networks. First, when $\mu$ is extreme small (little or no cluster-head replacement), a destroyed cluster-head causes significant loss in $AC$. Thus, a flat network is better to maintain $AC$ under attacks when $\mu$ is very small. Second, when $\mu$ is large enough, hierarchical sensor network performs better. The reason is that, when $\mu$ is large enough, although the attacker spends significant amounts of time to reach the cluster-heads and destroy them, the cluster-head will be replaced soon. On the other hand, the cluster-head can attract the attacker to spend significant amounts of time in moving to them and destroying them (since cluster-head signals propagate farther than normal sensors and result in larger moving distance and larger sweeping areas according to equation 4.1).

Figure 4.2: Sensitivity of $AC$ to sensor network parameters, $\mu$, $f$ and $C$.

Figure 4.2 (b), shows the sensitivity of $AC$ to the active signal frequency ($f$), under varying cluster sizes ($C$), under both $PA/NH$ and $PA/H$ scenarios. When $f$ is larger, the attacker can detect more sensors, and consequently $AC$ goes down
rapidly. The second observation is that $PA/NH$ compromises $AC$ better than or similar to $PA/H$ when $f$ is small. The opposite effect can be observed when $f$ is large. This is because, when $f$ is large, more sensors are detected, including cluster-heads and normal sensors. In this case, it is better for the attacker to give priority to many normal sensors near it than giving priority to cluster-heads far from it. Giving priority to cluster-heads can initially decrease $AC$ but the fall of $AC$ is only temporary due to the cluster-head rotation (replacement of cluster-heads). Destroying more normal sensors reduces long term $AC$ even if cluster-head rotations happen later to replace the destroyed cluster-heads. That is why when $f$ is large, $PA/NH$ makes the attack more effective.

The third observation we make from Figure 4.2 (b) is, when $f$ is extreme small, a large cluster size ($C$) results in less decrease of $AC$. When $f$ is large, a larger cluster size reduces $AC$ more significantly compared with a smaller cluster size. The reason is because, when $f$ is extreme small, very few normal sensors or cluster-heads will be detected through active signals. In this case, a large cluster size means less number of clusters, which reduces the chance that a cluster-head is detected and destroyed. Consequently, $AC$ is larger. However, when $f$ is large, more number of cluster-heads will be detected irrespective of the cluster size. In this case, larger cluster size ($C$) increases the impact of destroyed cluster-heads on $AC$, thus decreases $AC$.

4.4 Variations of Physical Attacks

In this chapter, we presented a representative model of an external mobile agent orchestrated physical attacks in wireless sensor networks, and analyzed its impacts. However, the physical attacks model presented here need not be the only one. In [59],
we studied the issue of blind physical attacks in sensor networks. Typical blind attacks include physical attacks in the form of bombs or grenades destroying contiguous portions of sensors, tanks or vehicles driven around in the field, etc. Note that blind physical attacks are a particular instance of search-based physical attacks, wherein the searching phase is just executed by roughly identifying the sensor network field. Following this, the attacker attacks the sensor field blindly using a brute-force approach, and nodes that happen to be in the attacked areas are destroyed. However, while blind physical attacks are potent in destroying sensors, they may not always be practical for attackers to launch such attacks. A fundamental downside of brute-force attacks is the easy detection of such attacks. Secondly, blind physical attacks will cause casualties to the deployment field. In many cases, it may be of interest for the attacker to preserve the deployment field. Such fields can include airports, oil fields, and battlefields etc. that are of interest to the attacker. In such cases, the attacker will prefer to launch search-based attacks in the deployment field by using mobility to identify and destroy only the sensors, without being detected or compromising the deployment fields. The attacker will be successful in physically destroying sensors and the network mission in a stealthy manner, while still preserving the area. Furthermore, the attacker can use mobility as a leverage to significantly enhance the geographical scope of sensor destruction, while simultaneously making it hard to localize the attacker and its impacts. Other situations, where search-based attacks are preferred include ones where the attacker has the ability to identify critical sensors like cluster-heads, data-aggregators etc. Destroying such sensors can cause more damage than just a blind destruction of sensors. In [60], we model the combination of
search-based and blind physical attacks. We call such attacks as *Policy driven physical attacks*, where a mobile attacker searcher for sensors, but physically destroy them using different destruction methods based on balancing trade-offs between maximizing sensor destructions and low deployment field casualty. We also conduct a detailed performance analysis to study the impacts of policy driven physical attacks on sensor networks, under varying attacker capacities in [60]. While such variations (and some others) are likely methods to physically destroy sensors, we believe that the mobile orchestrated *search-based* physical attack model detailed in this dissertation is the most representative one.

### 4.5 Countermeasure Guidelines for Defending against Physical Attacks

With the destruction potential of physical attacks patent from the above discussions, we now present certain countermeasure guidelines for defending sensor networks against physical attacks. We first highlight the critical challenges in defending sensor networks against physical attacks, followed by countermeasure guidelines.

**Major challenges defending against physical attacks:** Defending against physical attacks is challenging. There is a clear tug of war between the attacker and the sensors. In this war, the attacker will attempt to destroy sensors and compromise the mission, while the sensors will try to escape detection and also sustain their mission. The first challenge in defending against physical attacks is due to the attacker being very likely more powerful than the sensors. Consider the searching phase of the attacker. The attacker is likely a human being (or a mobile robot) with signal detection equipment to locate sensors. While sensors may also be able to detect the attacker, the sensing range of the attacker with its equipment is very
likely larger than the small sized sensors. Consequently, the attacker will be able to obtain sensor positions earlier than sensors can detect the attacker and trigger corresponding defense mechanisms. This makes the aforementioned war asymmetric, and it automatically favors the attacker. The fact that the attacker is mobile makes the difference in capacity more pronounced, as the attacker can expand its sensor detection scope throughout the network with mobility. Another consequence of mobility is that the sensors will have difficulties in localizing the attacker in the network, which will further complicate defense. During the sensor destruction phase, the attacker can use simple mechanisms like heat, radiation, circuit tamper devices etc. While tamper resistant packaging does help, it will not be enough for defending against physical attacks due to the small size of sensors and cost factors.

The second challenge arises due to the nature of sensor networks being mission-oriented. The sensors should sustain the mission by performing sensing, processing, communication tasks according to the mission. Let us consider that sensors can also detect the attacker (by means of magnetic signals from the attackers antenna, vibration due to attackers motion etc.). A simple and straightforward defense strategy against physical attacks here is for sensors to detect the attacker and propagate attacker information further to other sensors. Sensors within an area that receive attacker information simply shut down. A sensor is shut down (neither senses nor transmits) when it turns itself off completely. Thus, attackers will not be able to detect sensor signals, and the sensors are prevented from destruction. The conflict here is that the conservativeness of the above strategy compromises coverage and connectivity in a portion of the network. The challenge is to design defense mechanisms that effectively resolve the above conflicts.
Guidelines for defending against physical attacks: In the following, we present certain guidelines for defending sensor networks against physical attacks that aim to resolve the aforementioned conflicts. We classify our countermeasure guidelines into three phases: Pre-attack phase, Attack phase, and Post-attack phase.

- Countermeasure guidelines in pre-attack phase: The pre-attack phase is where the network deployer can implement countermeasures in anticipation of physical attacks in order to mitigate consequences should an attack commence. In this phase, one natural guideline is to over-provision sensors at deployment time to compensate for sensor destructions under attacks. The deployer can let the extra sensors sleep initially, and periodically wakeup to check for the live status of their neighbors. Depending on the 'alive' status of their neighbors, such sensors can stay alive or go back to sleep. However, note that the extra sensors to be over-provisioned are not free. The deployer has to know how many extra sensors are needed? The number depends on the intensity of the potential attacks and the run time/ post attack defense strategies used. How to optimally deploy the extra nodes is important here and is contingent on the attacks therein. Another technique that could be employed is the use of decoying to actively confuse the attackers. A set of decoy sensors can be placed in optimal geographical locations to confuse attackers by sending signals that confuse an attacker. However, it needs to be mentioned that decoys will cost more nodes and energy. The research issues are how to optimally place the decoys and the approach to confuse attackers, and to study of the trade offs involved.
Countermeasure guidelines in attack phase: This phase is one where there is a physical attacker present in the sensor network that is orchestrating attacks. One effective solution against sensor destruction is to prevent their localization by physical attackers. Recall that while conducting attacks, the attacker will attempt to localize sensors by associating multiple signals from the sensors. However, such association can be nullified if sensors encrypt their messages. This is because, when the attacker can still detect raw sensor signals, the actual message content and sensor identities are unknown to the attacker, and signals associations to sensor ids are not possible any more. When sensors cannot be localized, the attacker cannot reach them and destroy. Another solution we proposed in [61] is to use certain nodes as sacrificial nodes during on-going physical attacks. A sacrificial node is one which detects the attacker and propagates this information to many other sensors to save them from destruction at the risk of itself being detected and physically destroyed by the attacker. The existence of sacrificial nodes compensates the weakness of the sensors’ ability to detect and be prepared for the attacker (due to attacker’s mobility) by extending the area in which sensors are aware of the proximity of the attacker. The core principle of the sacrificial node-assisted defense protocol is to trade short term local coverage for long term global coverage through the sacrificial node-assisted attack notification and states switching of sensors. While this approach is novel, and is shown to significantly prolong Accumulative coverage, issues like how to optimally choose sacrificial nodes, their number, their role (cluster-heads vs. normal sensors), rotation among multiple sacrificial nodes are open issues. Other possible countermeasures under physical attacks are prediction of attacker speed.
and future movement patterns (using approaches designed for target tracking in sensor networks [25,62–64]) to notify potentially vulnerable sensors to let them sleep and escape detection.

- **Countermeasure guidelines in post-attack phase:** This phase is one after the attacker has exited the sensor network. Obviously, after sensor destructions have taken place, repair and recovery are very important for the network to sustain operations. First one must be able to accurately isolate boundaries of sensor destructations. Existing approaches in boundary detection in sensor networks can be leveraged for this purpose [65–68]. Another effective approach to detect disruptions in the network is by checking for invariants that the network must maintain at all times. In [69], such invariants are designed for wireless sensor networks to detect and self-heal corrupted network topologies under perturbations. Such well designed network invariants taking into account physical attack oriented perturbations can be an effective post-attack countermeasure. Note that in special situations, redeploying sensors in such zones to compensate for physical attacks may be a feasible option. However this may not be easy always. The issues to study are seamlessly integrating new sensors with existing ones while causing minimum service disruption. In some cases, a few mobile sensors can be deployed, that can attempt to co-ordinate the joining process, repair holes etc. For such mobile sensors, our algorithms in this dissertation to repair coverage holes are natural choices for network repair.
4.6 Summary

In this chapter we addressed the issue of modeling an external mobile agent orchestrated search based physical attacks in sensor networks. We identified critical features of search-based physical attacks and modeled a representative instance of search-based physical attacks. We studied performance impacts based on a novel metric that we defined, namely the Accumulative Coverage ($AC$). The accumulative coverage effectively captures both coverage and lifetime. Our performance data clearly showed that search-based physical attacks dramatically reduces accumulative coverage ($AC$), further highlighting the importance of our work. Furthermore, we also discussed several variations of physical attacks in sensor networks, and also proposed countermeasure guidelines against them.
CHAPTER 5

RELATED WORK

In this chapter, we discuss important works related to this dissertation. We classify this section into four subsections: works on internal mobility assisted sensor networks deployment; works on external mobile agents assisted sensor networks operation; works on sensor networks security; and finally the differences of this dissertation from these existing works.

5.1 Internal mobility assisted sensor networks deployment

Deployment is one of the most fundamental issues in wireless sensor networks, and has been heavily explored. Traditional works in this area has focused on theoretical aspects of both deterministic [70–74] deployment and random deployments [36, 37, 62, 75–82], of sensors to achieve desired coverage, connectivity, lifetime etc. More recently though, the issue of mobility assisted sensor networks deployment has gained significant attention in the community [1–9, 83–87], wherein sensor mobility is leveraged to attain one or more desired deployment objectives. In the following, we discuss important works in the area of mobility assisted sensor networks deployment.

a). Virtual Force movement algorithm: In [1], [4] and [3], the authors design virtual force based algorithms for improving 1–coverage in the network after an initial
random deployment of sensors. In these algorithms, sensors exert virtual forces, where closer sensors repel each other and farther sensor attract. In this manner, the sensors spread themselves uniformly in the network after several iterations. Heuristics are proposed to minimize unnecessary movements. In [9], the virtual force approach is extended to consider coverage and secure connectivity. Each sensor has a certain number of keys initially predistributed, and two sensors can securely communicate if they are in the transmission range of each other and share a common key. To enhance both coverage and secure connectivity, weights assigned to the virtual forces depending on keys shared. The weights ensure that sensors sharing keys do not move too far apart, while those that don’t share keys move farther. In this manner coverage is enhanced among secure neighbors, and so are chances of other sensors finding secure neighbors.

b). Scan-based movement algorithm: In [2], the authors consider the problem of balancing number of sensors per region in a clustered sensor network, after an initial random deployment. In their approach, sensors scan the regions row-wise first with information on number of sensors per region. A row-wise sensor movement follows to balance number of sensors in each row. A subsequent column-wise scan and movements balances number of sensors in both rows and columns. It is proved that the number of movements in this algorithm is at-most twice the number of optimal movements.

c). Bidding movement algorithm: In [7], the authors consider a heterogeneous sensor network, where the goal is to enable cooperation between mobile and static sensors to enhance network coverage. In the algorithm, mobile sensors are treated as servers to heal coverage holes. Each mobile sensor has a certain base price for
serving one hole in the sensing field. The price is related to the size of any new hole generated by their movement (a higher price means a larger loss in coverage if that sensor moves). Static sensors will detect the coverage holes locally, estimate their sizes as bids, and bid the mobile sensor with a base price lower than their bids. Mobile sensors choose the highest bids and move to heal those coverage holes.

**d). Voronoi based movement algorithm:** In [88] and [6], algorithms are designed, where sensors move towards events as and when they are generated. The objective is to enhance event coverage without compromising existing coverage (i.e., create new coverage holes). After deployment, sensors discover locations of their neighbors. When events are generated, each sensor determines the predicted motion of neighbors in its vicinity to determine their final positions and its own, based on constructing voronoi polygons for itself and its neighbors. Sensors move if they can get closer to the event, and if the area that it has previously covered will still be covered by one or more neighbors.

**e). Randomized movement algorithm:** In [5], a mobility model is proposed where each sensor independently moves by choosing a random direction and speed, for a particular interval. After each interval, a new direction and speed are randomly chosen, and the process repeats for the duration of the mission. In this model, coverage of any particular point is not fixed, but alternates over time. The authors analytically demonstrate that while the fraction of area covered at any instant remains unchanged, the total area covered during a time interval significantly improves in this mobility model. Using this result, the authors derive the distribution of detection time for static and mobile intruders in the network, as a function of initial sensor distribution, their sensing ranges and movement speeds.
5.2 External mobility assisted sensor networks operation

A host of research work has recently appeared wherein the larger capacity of certain external mobile agents are leveraged to overcome the weaknesses of static sensors in sensor networks. In works like [10], [11], [12], [13], [14], the authors address the trade-offs in energy savings between using a mobile sink in the network vs. sensors relaying information via multi-hop to a sink. Basically, in large scale sensor networks, sensors may have to transmit information via several hops to reach a sink. Clearly, energy is consumed during each hop which reduces the lifetime of the network. In the above works, the authors propose to leverage certain external mobile agents (e.g., a vehicle or a soldier) to move closer to sensors and collect stored data. In this manner multi-hop transmission among sensors is avoided. In all the works in [10], [11], [12], [13], [14], the authors derive movement schedules for one or more mobile agents, along with routing algorithms that exploit agent mobility for minimizing energy consumption at sensor side and to enhance sensor network lifetime. In [15], the authors propose a method to exploit robotic mobility by having energy producers be mobile robots. These robotic agents try to keep themselves recharged by moving to locations with abundant energy supply. Once charged, they migrate to the service areas in the network for delivering energy to the energy depleted (static) sensor nodes. The authors characterize the problem of uneven energy consumption and provide a simple analytical framework to evaluate energy consumption of the scheme. Experiments are also conducted to evaluate the feasibility of the energy replenishment scheme proposed.

In works like [16], [17], [18], [19], mobile agents with GPS devices attempt to localize sensors that do not know their positions in the network. In [17], a mobile beacon
aware of its position continuously broadcasts its coordinates. A sensor measures RSSI of the signal when it receives one from the mobile and estimates its likely range. A Bayesian inference method is used to converge to real position after successive measurements of the mobile’s signal. In [18], a method is proposed where the mobile measures distances between sensor pairs until these distance constraints form a globally rigid structure that guarantees a unique localization. In [16], a mobile beacon advertises its position periodically, and sensors keep track of three positions from the mobile after filtering out the rest. The three positions correspond to when the sensor first detects the mobile (when the mobile enters its range), when it has last detected the mobile (when the mobile just leaves its range), and when the sensors re-detects the mobile (when the mobile again enters its range). The localization algorithm is based on the idea that a perpendicular bisector of a chord passes through the center of the circle. In [19], the authors propose an approach based on extended kalman filters to associate RSSI signal strengths advertised from a mobile beacon into position estimates for sensor localization.

5.3 Sensor networks security

Sensor networks security is a topic that has received significant attention recently [52, 56, 89–135]. There are a host of challenges in securing sensor networks. First, since many sensor networks are prone to deployed in hostile zones that may be unattended, the chances of them being targets of attacks are much higher. Second, sensors being small in size are easy to be compromised. The fact that sensors are severely energy constrained also precludes the possibility of complicated defense approaches that will entail too much computational/ communication overhead. In the
following, we present discussions on certain well known and practical security threats in sensor networks.

Eavesdropping attacks are an issue that is well studied in sensor networks. A host of symmetric key cryptographic approaches have been proposed for protecting information disclosure to eavesdropping adversaries [89–112, 136]. Fundamentally, all these approaches are based on probabilistic key sharing among the nodes of a random graph and simple protocols for shared key discovery, path-key establishment, key revocation, re-keying, and adding nodes to the network. This is done by means of selective distribution and revocation of keys to sensor nodes, as well as node re-keying without substantial computation and communication requirements. Another type of attacks salient to sensor networks are location based attacks, where attackers attempt to discover sensor locations/ fake the location estimates etc. In [122, 123], the issue of how to prevent traffic analysis attacks to protect the location of the base-station is discussed. In works like [117–121], the issue of how to securely localize sensors in the presence of malicious localization attacks are proposed. Orthogonally, in [124–130], mechanisms are presented by which false location claims of sensors can be discovered. In works like [131–133], approaches are proposed to detect malicious ensure nodes that claim identities other than its own. In [52], a comprehensive list of routing attacks and their defenses are presented. In [56, 134, 135], approaches are presented to detect and defend against jamming based denial of service attacks in sensor networks. We wish to point out that security of wireless sensor networks is a vast topic. The above attacks are only representative ones in sensor networks. For a more comprehensive survey on sensor networks security, please refer to [137].
5.4 Differences of this dissertation from existing works

a). Differences of our work from exiting works on mobility assisted sensor networks deployment: In [1], [4], [9] and [3], the goal is uniform 1—coverage of the network. This means that every point in the network is covered by at least one sensor. The approach in [1], [4] and [3] is balancing sensor virtual forces. Two sensors may repel or attract each other based on the distance between them. At each iteration, sensors move to achieve a better force balance, and sensors stop moving when a force equilibrium is reached. However, under hard mobility constraints, two sensors may not be able to achieve force balance if the distance required to be traversed is too large. Secondly, the virtual force approach will result in several back and forth sensor movements during force balancing, which across many iterations will rapidly deplete mobility capacity of sensors, that cannot be tolerated by sensors with hard mobility limitations. Another difference is that all of the above works focus on one-coverage of the sensor network, while in this dissertation, we design algorithms for both 1—coverage and $\bar{k}$—coverage (arbitrary $\bar{k}$). In the scan-based mobility algorithm in [2], a major drawback is that the ratio of number of hops in the proposed algorithm and the optimal case is bounded by a factor of 2. Limited mobility sensors cannot tolerate so many unwanted moves. Also, the problem in [2] is a special case of our general variance minimization problem in this dissertation. Other sensor mobility algorithms like [7], [88], [6], [5] and [8] also have limited practical applicability, since they do not consider mobility limitations in sensors.

Fundamentally, in all the above works on mobility assisted sensor networks deployment, sensors move based on local measurements and communications. Erroneous movements made by the sensors are eventually corrected over time, since the
sensor movement distance is assumed to be unlimited. In our mobility model, sensor movement distance itself is a hard constraint (which is always true in practice). This means that, we are fundamentally constrained by the movement choices available to us (compared to unlimited mobility) during deployment. The consequence is the increased importance that each sensor movement needs to be accorded. Sensors therefore cannot just make local decisions and move. We need to determine a movement plan for the sensors prior to their movement, which is the core of our solution in this dissertation. We wish to point out that recently (in September 2007), another work on limited sensor mobility assisted deployment for clustered sensor networks (of dimension $L \times L$) has appeared [138]. In [138], the authors consider the problem of deriving bounds for the amount of mobility needed to achieve $\bar{k}$-coverage. The authors formally prove that an all mobile sensor network can provide $\bar{k}$-coverage over a sensor field with a constant sensor density of $O(\bar{k})$, where the maximum distance a mobile sensor moves is $O\left(\frac{1}{\sqrt{\bar{k}}} \log^{\frac{3}{4}}(\bar{k}L)\right)$. The authors then propose a hybrid network where with $O\left(\frac{1}{\sqrt{\bar{k}}}\right)$ mobile sensors. For this network, the authors prove that $\bar{k}$-coverage is achievable with a constant sensor density of $O(\bar{k})$, where each mobile sensor’s movement is bounded by $O(\log^{\frac{3}{4}}L)$. Finally, the authors propose a sensor movement algorithm that is inspired by our flow-based algorithm in this dissertation to achieve $\bar{k}$-coverage. However, the mobility algorithm proposed in [138] focuses on purely load balancing sensors in the network, which is a special case of our deployment problem. Furthermore, unlike our algorithms, the algorithm in its presented form in [138] cannot work the case where multiple regions may need multiple sensors (i.e., different $\bar{k}$ for each region). Nevertheless, the mobility bounds derived in [138] is a significant contribution, and along with our work in this dissertation, they can
be a foundation for further research in the area of limited mobility sensor networks deployment.

b). Differences of our work from exiting works on external mobility and security in sensor networks: To the best of our knowledge, all existing works on external mobile agents in sensor networks focus on only the benign aspects of such agents. Ours is the first work in this regard to focus on the security downsides of such external mobile agents, and formally model and analyze the resulting impacts. Furthermore, the issue of physical attacks is a novel threat that is salient and highly representative to sensor networks. Physical attacks are clearly different from attacks discussed above such as eavesdropping, identity fabrication, jamming, localization attacks etc. In the case of physical attacks, the attacker (human being or a robot) physically resides and moves in the sensor network. This provides opportunities for the attacker to react to defense mechanisms executed by sensors, unlike other attacks. Secondly, physical attacks destroy sensors permanently which is not the case in other attacks. Note here that physical attacks need not be stand alone. Smart attackers can combine physical attacks with other attacks like jamming, eavesdropping attacks etc., furthering the damage. Another difference is the mobility assisted searching phase in physical attacks, which is not present in other attacks. Mobility can be an effective leverage for the attacker to significantly enhance the geographical scope of sensor destruction, while simultaneously making it hard to localize the attacker and its impacts. Across some respects, the end effects of physical attacks are similar to fail-stop fault models [139,140], where the sensor is simply dead. However in physical attacks, the node destructions are orchestrated by an attacker, which means that the faults (i.e., dead sensors) are not independent or isolated. Rather, they have
geographical coherence, based on execution of the searching phase (which in turn depends on attacker’s mobility patterns), further demonstrating the uniqueness of mobile agent orchestrated physical attacks in sensor networks.
CHAPTER 6

FINAL REMARKS

In this dissertation, we made contributions in the area of mobility in wireless sensor networks. We first categorized sensor networks mobility into two classes: Internal mobility and External mobility. We then made contributions to both classes of mobility from the perspective of sensor networks deployment and security.

First, we studied the issue of how sensors capable of limited mobility can relocate themselves to attain desired deployment. Towards this extent, we defined two mobility assisted sensor network deployment problems for both 1-coverage and $\bar{k}$-coverage ($\bar{k} \geq 1$). Our problems are highly representative in many sensor networks deployment scenarios. We then identified critical challenges arising in deployment under hard limitations on sensor mobility. We then proposed novel methodologies, centralized optimal and distributed heuristic algorithms for our deployment problems. Using formal analysis and extensive simulations, we demonstrated the performance of our proposed algorithms. To the best of our knowledge, ours is the first work to study the issue of sensor networks deployment under hard constraints on sensor mobility (both in movement patterns and distance). As we demonstrated, such limitations are natural as a result of sensor’s being severely constrained in their form factor and energy, and existing mobility algorithms that do not consider such constraints have
limited applicability in practice. Another significance of our work is the generality of our methodologies and algorithms, in the sense that they are applicable to solve a wide spectrum of other mobility assisted deployment problems in sensor networks, other than just the ones formally addressed in this dissertation.

Second, we studied the issue of an external mobile agent orchestrating physical attacks in sensor networks. We defined a representative model of physical attacks, wherein an external malicious mobile agent (human being or robot) enters the sensor network field to detect inherent physical or electronic sensor signals using appropriate signal detection equipment. By measuring certain signal properties, the attacker locates sensors and then reaches the sensors to physically destroy them. We formally modeled such attacks in sensor networks, demonstrated their destructive potential for sensor networks security, identified their variations, and finally proposed guidelines for countermeasures against such attacks. To the best of our knowledge, ours is the first work to highlight security impacts under the presence of external mobile agents and to address the topic of physical attacks, both of which we believe will be critical components of sensor networks security in the future.

With the emergence of mobility in wireless sensor networks, coupled with its patent significances, we hope that our work in dissertation can provide strong foundations and further motivations for researchers to explore this topic. Some critical open issues in this realm are understanding impacts of internal mobility to issues like channel contention, connectivity/topology corruption, security etc. Also in some applications of today, it may happen that even if the sensors are static and cannot make (or control) movement decisions, they can still be relocated from one location to another by third-party forces beyond the control of sensors themselves. Typical
instances of such types of internal mobility include sensors fitted on mobile patients to monitor their heart-rates, sensors fitted on animals to monitor their behaviors in habitats etc. Clearly, the sensors here will be physically relocated during the mission even though they have no ability to control their mobility. Understanding such types of internal mobility with respect to developing mobility models, understanding mobility impacts to protocols design (e.g., neighbor discovery and data delivery) are interesting open issues. From the perspective of external mobility, further investigations into both their opportunities (for periodic sensor network health monitoring, rapid response to network events/dynamics etc.) and modeling and defenses of other security threats will also be an important topic of future research.
BIBLIOGRAPHY


International Parallel and Distributed Processing Symposium (IPDPS), April 2004.


