12-12-2014

Cost-Effective Algorithms for Deployment and Sensing in Mobile Sensor Networks

Yuan Song
University of Connecticut - Storrs, andyyuansong@gmail.com

Cost-Effective Algorithms for Deployment and Sensing in Mobile Sensor Networks

Yuan Song, Ph.D.

University of Connecticut, 2014

ABSTRACT

In this dissertation, we propose cost-effective algorithms for deploying mobile sensors in traditional mobile sensor networks and scheduling mobile phone opportunistic sensing in mobile phone based sensor networks. Traditional sensor nodes must be deployed appropriately to successfully accomplish their sensing tasks. When the region of interest is unknown or hostile, manual deployment is infeasible. For such scenarios, how to employ sensor mobility to achieve cost-effective self-deployment is an interesting yet underexplored problem. While mobile phones are naturally sensor nodes, they differ from traditional sensor nodes in that they move along with their owners and only perform sensing tasks in an opportunistic manner. Therefore, an interesting question is how to cost effectively schedule sensing tasks on a group of mobile phones, taking account of the movements of these mobile phones.

In the absence of a prior knowledge of the region, deploying sensors evenly in the region, referred to as even self-deployment, is one of the best known strategies. In the first part of the dissertation, we propose distributed algorithms for energy-efficient even self-deployment in mobile sensor networks. Specifically, we first formulate a locational optimization problem that achieves even deployment while takes account of

energy consumption due to sensor movement, and then propose two iterative algorithms.

In the second part of the dissertation, we study self-deploying sensors to monitor a set of targets when the target locations are known beforehand (through surveillance). Our goal is to determine which target a sensor moves to so that the duration for which a target can be monitored is balanced among the targets. We start with the simplest scenario where all sensors have the same initial location and energy, and propose an optimal algorithm to solve it. For the general scenario, the problem is NP-hard. While it is a special case of Max-min fair allocation problem, existing solutions are computational intensive, and hence are unsuitable for resource-constrained sensors. We propose a greedy heuristic scheme and demonstrate that it achieves similar performance as the best known algorithm for Max-min fair allocation while requires much less running time.

In the third part of the dissertation, we study how to cost effectively schedule mobile phones to monitor a set of targets and upload the collected data. The goal of the problem is to obtain a cost-effective schedule of phone activities. We start with the offline problem, assuming the trajectories of the phones are known beforehand. We propose to overcome the limited cellular data plans of the phones by resorting to opportunistic communication among mobile phones. We then formulate and solve a minimum cost flow problem and a fair cost problem. After that, we investigate the online version of the problems under realistic assumptions where only the past trajectories of the phones are known. We develop two heuristic algorithms: one aims to minimize cost while the other aims to achieve cost fairness. Extensive simulation results show that the proposed algorithms perform well in terms of both minimizing total cost and achieving cost fairness.

Cost-Effective Algorithms for Deployment and Sensing in Mobile Sensor Networks

Yuan Song

M.E., Xidian University, China, 2008

B.E., Xidian University, China, 2005

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2014

APPROVAL PAGE

Doctor of Philosophy Dissertation

Cost-Effective Algorithms for Deployment and Sensing in Mobile Sensor Networks

Presented by

Yuan Song, B.E., M.E.

Major Advisor _____
Bing Wang

Associate Advisor _____
Alexander Russell

Associate Advisor _____
Zhijie Shi

University of Connecticut

2014

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Bing Wang, for her excellent guidance, caring, and patience throughout the challenges of my doctoral studies. She is my role model for great researcher and mentor. I have learned a great deal from her unique perspective on research, her sharp insight on almost any issue, her personal integrity and expectations of excellence. I would not survive my Ph.D without her constant support and encouragement.

I am very fortunate to have had Professor Alexander Russell, Professor Zhijie Shi serve on my committee. They have been the great sources of inspiration regarding my research. Their guidance and generous support had greatly expedited my progress on difficult research problems.

I am indebted to Professor Shalabh Gupta for his inspiring input on my research. His brilliant ideas have greatly broadened my research scope.

I would like to extend my gratitude to my colleagues, Xian Chen, Wei Zeng, Ruofan Jin and Yuexin Mao, to name a few, for not only their great help to my research but also excitement and happy moments they have brought to my graduate life.

I would also like to thank my parents. They were always supporting me and encouraging me with their best wishes.

My final thanks goes to my beloved wife, You Li for her unending support, and my beloved daughter, Chloe, whose love is worth it all.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Deployment and Coverage in Mobile Sensor Networks

The sensing capability of sensors provides a new way for people to collect data and facilitate the interaction between people and the physical world. Once deployed in an area, sensors take charge of monitoring the area. Wireless sensors networks, composed of a large number of sensors, have been widely applied in many sensing scenarios such as military surveillance, natural disaster recovery, medical health monitoring, and harsh environment exploration. For example, in military surveillance, sensors can be deployed to detect intrusion and collect data from hostile areas [26]. In health monitoring, a variety of system prototypes and commercial products on wearable sensors have been developed. These wearable sensors can be exploited to collect real-time feedback information about patient's health condition [55]. In en-

vironment exploration, sensors has been applied to monitor difficult-to-access areas such as environmental sensitive beaches and near-shore coastal oceans [33].

The deployment of sensors determines the quality of the surveillance that the sensor network can provide, e.g., how well a target area is monitored. Depending on the applications, the deployment goal of sensor networks may vary. Existing formulations of sensor deployment can be categorized into area coverage, point coverage and barrier coverage [18]. In *area coverage* problems [32, 71, 82], the goal of the deployment is to cover the whole target area while the *point coverage* problems [16, 72] focus on covering a set of known points in the target area. *Barrier coverage* problems [42, 63] differ from the other two types of coverage problems in that the goal is to minimize the probability of undetected penetration through the barrier (sensor network).

Sensors can generally be placed in a target area either deterministically or randomly [77]. Deterministic deployment is considered as a preferable way to deploy the sensors as it can always achieve the desired deployment goal. However, when deploying in unknown or hostile areas (e.g., remote harsh fields, disaster areas or toxic urban regions), deterministic deployment may not be viable due to operational constraints. In such applications, random distribution of nodes is the only feasible option as sensors are often randomly dropped or projected into the target area in such scenarios. It may not be realistic to expect sensors to be placed in a desired way. Furthermore, the sensors are generally considered fragile and failure-prone. It is possible that sensors may fail due to various reasons such as power depletion and hardware defects, which leads to network degradation or corruption. Mobile sensors become very useful in such cases. With the help of their locomotive abilities, they can cooperate by employing self-deployment to achieve the desired deployment goal or recover the network from unpredictable sensor failures [35, 71].

Energy consumption needs to be considered carefully in employing mobile sensors in the deployment and sensing applications. Mechanical movement of mobile sensors is one of the dominant sources of energy consumption [67]. Since most mobile sensors are battery powered, the energy consumption in movement may greatly reduce their lifetime in sensing tasks. Therefore, the movement paths and destinations of mobile sensors should be carefully planed so that the energy consumption in movement can be minimized.

1.1.2 Mobile Phone Sensing in Urban Area

For the sensing applications in the urban area, rapidly developing mobile phones have become a great alternative to traditional mobile sensors. With a growing set of cheap powerful embedded sensors, such as an accelerometer, digital compass, gyroscope, GPS, microphone, and camera, mobile phones are naturally mobile sensors. The richly equipped mobile phones can enable sensing applications in various domains such as environmental monitoring, social network, health-care, transportation, safety, etc [41,44]. For example, in [52], the authors have proposed a system called SoundSense, which collects sound events through mobile phones' microphones. It uses various learning techniques to classify the sound into different categories to recognize sound events in users' daily lives. In [74], the authors have proposed StreamShaper. It employs mobile phones to collect the environmental data, which provides a virtual sensor abstraction to applications.

There are two mobile phone sensing paradigms, *participatory sensing* and *opportunistic sensing* [44]. In participatory sensing, users actively engage in sensing activities by manually determining how, when, what, and where to sense. An advantage

of participatory sensing is that complex operations can be supported by leveraging the intelligence of the person who participates in the sensing tasks. In opportunistic sensing, sensing activities are fully automated without user involvement. The benefit of opportunistic sensing is that it lowers the burden placed on users, allowing overall participation of users to remain high even if the application is not that personally appealing.

Depending on the sensing scale, mobile phone sensing applications can be categorized into personal sensing, group sensing and community sensing [44]. A *personal sensing* application applies sensing to phone users. It typically collects data to track user's daily life. For example, the authors of [27] have proposed a smart phone system named HealthAware. It takes pictures of intake food of the user with phone's camera while records daily physical activities of the user with the embedded accelerometer sensor. It then presents the users how much daily activity is needed to burn the intake food to keep healthy. Mobile phone users participating in *group sensing* application generally share a common interest or goal. They share the sensing data with each other to facilitate the sensing process. MoVi system proposed in [11] is an example of group sensing application. It improves social event coverage by capturing video highlights of users' events. It detects the event by correlating the sensed data with the data from other phones in the same social group. Once an event is confirmed, the phone with the best view will be recruited to take the video of the event. The number of participating users in *community sensing* applicatiosn is usually large. The design goals of these applications are considered to benefit the general public. An example of such application is Ear-phone system [58]. In this system, the participating users collect noise data via mobile phones. A noise map of a city is created to assess noise pollution of the city.

Compared to sensing through tradition mobile sensors, mobile phone sensing provides many additional benefits. It incurs no deployment cost and provides sensing ability in any area where the mobile phone users travel to. Since mobile phones have matured as a computing platform, employing mobile phones in sensing tasks will enable a wide variety of sensing applications. As for the delay-sensitive sensing applications, an important advantage of employing mobile phone in sensing tasks is that they can offload the data right at the sensing site, either via WiFi hot spots or via widely available cellular networks. Consistent access to the Internet also provides the opportunity to allow centralized control over participating mobile phones. In designing an effective mobile phone sensing system, several important factors need to be considered carefully.

- *Uncertainty in moving trajectory* . Participating mobile phone users may allow opportunistic sensing only. In such case, users generally will not travel to sensing target actively. Their moving trajectories are uncontrollable and thus renders the sensing of the target unpredictable.

- *Limited 3G offloading capability.* It is quite common that the available data plans on mobile phones are limited, which limits their data offloading capabilities.

- *Operation costs.* All activities of mobile phones in sensing tasks may incur certain cost. The costs may either be energy consumptions of phone activities or the financial compensations the system has to pay to the participants for their phone activities.

1.2 State of the Art

1.2.1 Deployment and Coverage in Mobile Sensor Networks

Sensor deployment and coverage related topics have been an active research area. Besides the coverage problems described in Section 1.1.1, K-coverage problem where each point is covered by at least k sensors is studied in [43, 81]. The relationship between area coverage and network connectivity is investigated in [9, 72].

Deployment and coverage problems of mobile sensor networks have attracted a lot of attention. In [35, 82], virtual force based algorithms are used to repel nodes from each other and obstacles to maximize coverage area. In [71], the proposed algorithms apply Voronoi diagrams to detect coverage holes. Sensors move towards coverage holes to increase the coverage. A distributed control and coordination algorithm for sensor formation is proposed in [22]. It employs Voronoi diagram based utility function to compute the optimal sensor deployment step by step. In [12], a distributed grid-shaped deployment algorithm is proposed for deployment in unknown fields. In [47], the authors have proposed a optimal mobility strategies for sensors to detect intruder based on a game theoretic approach. In Chapter 2, we propose distributed algorithms for achieving energy efficient even self-deployment, which differs from many existing studies where the focus is the coverage.

Several studies [16,17,25,80] study how to schedule the sensors so that the lifetime of the sensor network is maximized. Assuming that a large number of sensors are randomly deployed in the region of interest, monitoring a set of targets with known locations, these studies schedule the wireless sensors to alternate between active and sleep modes to save energy consumption while maintain target coverage. In Chapter

3, we propose computation efficient algorithms for energy-balanced deployment to a priori known target locations. We explore how to use sensor mobility to move the sensors to the desired target locations (which can be predetermined by solving a coverage problem). Our solution can be used as a second step after solving any of the above static coverage problems when manual deployment of the sensors is not feasible.

1.2.2 Mobile Phone Sensing in Urban Area

Recent mobile sensing and computing applications have been increasingly through mobile phones [41, 44]. Besides the sensing applications described in Section 1.1.2, the authors in [57] have proposed METIS platform. It implements a sensing task distribution scheme that dynamically decides whether to perform sensing on the phone or in the infrastructure, considering the energy consumption, accuracy, and mobility patterns of the users. In [23], the authors have designed a citizen journalist application based on the proposed Platform for Remote Sensing using Smartphones (PRISM). It provides location-based triggers to alert mobile phone users, who are in the vicinity of a location of interest, to respond to the application so that the sensing data at the location can be collected.

Different operational costs of mobile phone sensing have been considered in the literature. In [65], the authors consider energy consumption of mobile phone sensing as the cost. They have developed an energy-efficient collaborative sensing model which schedules mobile phones to achieve area coverage. In [76], the authors consider the incentives to attract more user participation as the cost. They have proposed two system models: the platform-centric model where the platform provides a reward

shared by participating users, and the user-centric model where users have more control over the payment they will receive. The downlink limitation of 3G data plan of mobile phones has been addressed in a few recent studies. In [30], the authors have proposed to exploit opportunistic communications to facilitate information dissemination in the emerging Mobile Social Networks (MoSoNets) and thus reduce the amount of mobile data traffic. In [60], the authors have proposed to exploit delay tolerance and tries to download contents when users are close to Wi-Fi access points. Our work differs from these studies in that we address uplink limitation and consider uploading cost. In order to achieve system wide cost optimization, the mobile phone sensing and data offloading should be considered together. In Chapter 4, we design generic target monitoring system through mobile phone sensing and present cost-efficient practical algorithms for the system. To the best of our knowledge, we are the first to address both mobile phone sensing in target monitoring and 3G data plan limitation together to minimize the total cost.

1.3 Contributions of This Dissertation

The contributions of this dissertation are three-fold: (1) developing distributed algorithms for achieving even energy efficient self-deployment, (2) developing computation efficient algorithms for energy-balanced deployment to a priori known target locations, and (3) designing a generic target monitoring system through mobile phone sensing and presenting cost-efficient practical algorithms for the system.

1.3.1 Mobile Sensor Self-deployment without A Priori Knowledge of the Target Area

When deploying sensors in an area without a priori knowledge of the area, the optimal deployment cannot be pre-computed. The initial deployment of sensors may be uncontrollable as the sensors are generally air-dropped or projected into the target area. In order to achieve the desired deployment, mobilities of the mobile sensors are exploited to achieve the desired deployment goals. Existing works focus on moving sensors to achieve area coverage [35, 71]. The mobile sensors move to fix the coverage holes. However, the area coverage may not achieve the best sensing quality in the target area.

Since no a priori information of the target area is available, all points in the target area should be considered equally important in sensing tasks. Therefore, the sensing quality of the sensor network is determined by the average sensing quality on every point in the area. Due to the limitation of sensing hardware, higher sensing quality generally implies that a sensor needs to closer to a target object or area. Even deployment, in such scenario, is considered the best deployment as the average distance between any point in the area and its closest sensor is minimal compared to other deployments. In order to achieve even deployment in an energy efficient manner, mobile sensors need to collaborate with each other to carefully plan their moving paths.

In Chapter 2, we address the deployment problem in mobile sensor networks when a priori knowledge of the target area is not available. According to Gersho's conjecture [28], for a given area and a set of sensors, the sensors are evenly distributed when they form a Centroidal Voronoi Tessellation (CVT) [56]. Therefore even self-

deployment requires the sensors to form a CVT of the target area, which differs from many existing studies where the goal is to maximize the coverage of the area [32,71,82]. For hostile or unknown fields, a centralized solution that pre-computes the final CVT and sensor final destinations is often infeasible due to the difficulty of gathering global knowledge and the lack of a centralized entity. Distributed algorithms that require no global information, but rather rely on sensors cooperation to form a CVT, are more desirable. In the first part of the dissertation, we propose distributed algorithms for energy-efficient even self-deployment in mobile sensor networks. Specifically, we first formulate a locational optimization problem that achieves even deployment while taking account of energy consumption due to sensor movement, and then propose two iterative algorithms.

1.3.2 Mobile Sensor Self-deployment with A Priori Knowledge of the Target Area

When the a priori knowledge of the target area is available, there may be a set of locations in the target area that need to be covered by the sensors, or the set of locations forms the optimal deployment for the sensing application. In both cases, due to the unpredictability of initial locations of mobile sensors, a mobile sensor needs to decide which location it travel to after landing in the area. In this scenario, while having only one sensor at each desired location suffices to achieve the coverage goal, allowing multiple sensors to move to a desired location improves fault tolerance as the sensors at the same location serve as backups for each other. More importantly, multiple sensors at the same location prolongs the sensing time at the location through role rotation [78]. In this way, the duration a location can be covered, i.e., the sensing

lifetime at the location, is primarily determined by the sum of the remaining energy of the sensors at the location (the remaining energy of a sensor is its original energy subtracted by the amount of energy consumed to reach the location).

The sensing lifetime at all locations should be balanced. Not only because it increases the time that all locations are under surveillance, but also it may impact the network lifetime. As sensors at locations may form a network to deliver the sensing data out of the network, a single point of failure may lead to the breakdown of the whole network. Therefore, mobile sensors have to cooperate with each other to select destination locations so that each location is covered by at least one sensor while the sum of the remaining energy of the sensors at all locations are balanced.

In Chapter 3, we focus on the deployment problem in mobile sensor networks when the target locations are known beforehand (through surveillance). We start with the simplest scenario where all sensors have the same initial location and energy, and propose an optimal algorithm to solve it. For the general scenario, the problem is NP-hard. While it is a special case of Max-min fair allocation problem, existing solutions are computational intensive, and hence are unsuitable for resource-constrained sensors. We propose a greedy heuristic scheme and demonstrate that it achieves similar performance as the best known algorithm for Max-min fair allocation while requires much less running time.

1.3.3 Mobile Phone Sensing in Urban Area

In the urban area, the sensing tasks involving monitoring people's activities, such as street surveillance, become especially challenging as the number of targets may be everywhere. It may require to deploy a large number of sensors in the target area

to cover all those targets, which is cost prohibitive and thus may not be feasible. Employing mobile sensors to move back and forth among the targets is a possible solution. However, deploying mobile sensors in urban area is not practical due to safety issues. The prevalence of mobile phone sensing becomes an effective alternative. Mobile phones that are equipped with various sensors are naturally mobile sensors. They can enable sensing applications in various domains such as environmental monitoring, social network, health-care, transportation, safety, etc [44].

In Chapter 4, we study how to cost effectively schedule mobile phones to monitor a set of targets and upload the collected data. The goal of the problem is to obtain a cost-effective schedule of phone activities. We start with the offline problem, assuming the trajectories of the phones are known beforehand. We propose to overcome the limited cellular data plans of the phones by resorting to opportunistic communication among mobile phones. We then formulate and solve a minimum cost flow problem and a fair cost problem. After that, we investigate the online version of the problems under realistic assumptions where only the past trajectories of the phones are known. We develop two heuristic algorithms: one aims to minimize cost while the other aims to achieve cost fairness. Extensive simulation results show that the proposed algorithms perform well in terms of both minimizing total cost and achieving cost fairness.

1.4 Dissertation Roadmap

The remainder of this dissertation is organized as follows. In Chapter 2, we describe our work on energy-efficient even self-deployment in mobile sensor networks. We first

present the motivation and the challenge in achieving even deployment in Section 2.1. Then we introduce background information in Section 2.2. In Section 2.3, we formulate a locational optimization problem that achieves even deployment while taking into account sensor traveling distances. Sections 2.4 describes our main algorithm for the even deployment problem. Section 2.5 presents a method to deal with limited communication range of sensors in computation. Section 2.6 presents performance evaluation. Finally, we summarize our work in Section 2.7.

In Chapter 3, we present our work on energy-balanced deployment problem in mobile sensor networks. We first review the motivation for achieving energy-balanced deployment in wireless sensor networks in Section 3.1. Then we present the related work in Section 3.2. Section 3.3 describe the problem formulation. After that, we present optimal algorithm for homogeneous setting when sensors have the same starting location and energy in Section 3.4. Section 3.5 presents algorithms for the general scenarios where sensors have different starting locations and energy. We demonstrate the effectiveness of the algoirthms in section 3.6 and summarize our work in Section 3.7.

In Chapter 4, we investigate how to cost effectively schedule mobile phones to monitor a set of targets. We first present motivation and challenges in scheduling mobile phones to monitor targets. We then describe the state of the art related works in Section 4.2. The problem formulation is presented in Section 4.3. We present optimal algorithms under idealized assumptions in Section 4.4. We then present heuristic algorithms for practical scenarios in Section 4.5. We describe our simulation settings and results in Section 4.6 and summarize our work in Section 4.7.

Last, we conclude this dissertation and present future work in Chapter 5.

Chapter 2

Distributed Algorithms for Energy-efficient Even Self-deployment in Mobile Sensor Networks

2.1 Introduction

Sensor nodes must be deployed appropriately to successfully accomplish their sensing tasks. When the region of interest is unknown or hostile (e.g., remote harsh fields, disaster areas or toxic urban regions), manual deployment is infeasible. In such cases, employing sensor mobility to achieve self-deployment is a suitable approach [32,71,82]. Even self-deployment, i.e., deploying the sensors evenly in the region, is one of the best known strategies in the absence of a prior knowledge of the region. For instance, it provides an optimal deployment for barrier coverage problem [63]. It also implies good coverage in "spot–sensing" applications [20], where each sensor node makes a

measurement (e.g., temperature or humidity) at the precise location of the node.

According to Gersho's conjecture [28], for a given area and a set of sensors, the sensors are evenly distributed when they form a Centroidal Voronoi Tessellation (CVT) [56] (see more details on CVT in Section 2.2). Therefore, even self-deployment, which requires the sensors to form a CVT of the target area, differs from many existing studies where the goal is to maximize the coverage of the area [32, 71, 82]. For hostile or unknown fields, a centralized solution that pre-computes the final CVT and sensor final destinations is often infeasible due to the difficulty of gathering global knowledge and the lack of a centralized entity. Distributed algorithms that require no global information, but rather rely on sensors cooperation to form a CVT, are more desirable.

A widely used distributed algorithm to construct CVTs is Lloyd's method [50]. It is a simple, iterative, and distributed algorithm, derived from the locational optimization problem [5] that requires little a prior information on the region of interest. To the best of our knowledge, it is also the only distributed algorithm that has been applied to sensor networks for even deployment [22].

When the initial locations of all sensors and the boundaries of the area are available, each sensor can run Lloyd's method locally to compute the final CVT, and hence its final destination, and then move to the destination directly. This approach, however, may not always be feasible or desirable. First, a sensor may not know the initial locations of the other sensors. This is because sensors are often initially deployed by airdropping or being projected into the area, thereby making the initial sensor locations difficult to predict. Furthermore, due to the uncontrolled initial deployment, sensors may not form a connected network, making it difficult for a sensor to communicate its location to the other sensors. Secondly, the boundaries of the

area may not be known beforehand and the sensors may have to sense the boundaries while moving in the area. Thirdly, even if each sensor can pre-compute the final CVT, some mobile sensors may fail while moving to their final destinations, leading to an uneven deployment formed by the remaining sensors.

Due to the above reasons, in practice, a more robust and scalable way is to apply Lloyd's method iteratively while fully employing its distributed nature. Specifically, in each iteration, sensors update their neighbor information, sense unknown boundaries and compute their next destinations. In this way, sensors will form an even deployment eventually. Furthermore, sensor failures can be detected immediately and reflected in the following computations when sensors update the information from their neighbors.

Despite its scalability and robustness, Lloyd's method suffers from two critical issues when being used in mobile sensor networks. First, it relies on accurate Voronoi neighbor information (more details of Voronoi neighbors can be found in Section 2.2), which is often not available in mobile sensor networks since the sensors may be out of the communication ranges of their Voronoi neighbors. Secondly, iterative application of Lloyd's method is not energy efficient — it does not optimize sensor moving distances, and therefore the sensors may travel longer distances than necessary before reaching their desired destinations. Since mechanical movement of sensors is one of the dominant sources of energy consumption [67], they may waste a large amount of energy.

This chapter addresses the issue of energy-efficient mobile sensor deployment to evenly cover a region of interest. The proposed method improves the energy efficiency of the iterative Lloyd's method by defining two cost metrics of energy consumption. The first one is the traveling distance of the sensors and the other one is the number

of deployment steps, which roughly equals to the number of start/stop operations of each sensor. The latter has been shown to be another major energy consumption source during the deployment process [67]. The limited communication ranges of sensors are also considered.

The main contributions of the work are as follows.

- The locational optimization problem is reformulated by incorporating the traveling distances of the sensors. The new formulation can achieve even deployment while taking account of the energy consumption.

- A new algorithm called Lloyd-α is proposed that reduces the movement step sizes in Lloyd's method and saves traveling distance while maintaining the convergence property.

- A new distributed algorithm, called DEED (Distributed Energy-Efficient self-Deployment) algorithm, is proposed, that reduces the energy consumption by saving sensor traveling distances while maintaining a reasonable number of deployment steps.

- An intuitive method is proposed to deal with the incomplete Voronoi neighbor information due to limited communication ranges of sensors. Simulation results show that this method helps the convergence of both Lloyd's method (the original Lloyd's method and Lloyd-α) and DEED algorithm.

- The performance of DEED algorithm is evaluated using both analysis and simulations. Extensive simulation results indicate that, compared to Lloyd's method, it reduces the traveling distance by up to 54%, and reduces the energy consumption by up to 46%.

The rest of the chapter is organized as follows. Section 2.2 presents background information. Section 2.3 describes our new formulation of the locational optimization problem. Section 2.4 describes DEED and its theoretical analysis. Section 2.5 describes the method to deal with the incomplete neighbor information in Voronoi cell computation. Section 2.6 presents performance evaluation. Finally, Section 2.7 concludes the chapter and presents future work.

2.2 Background

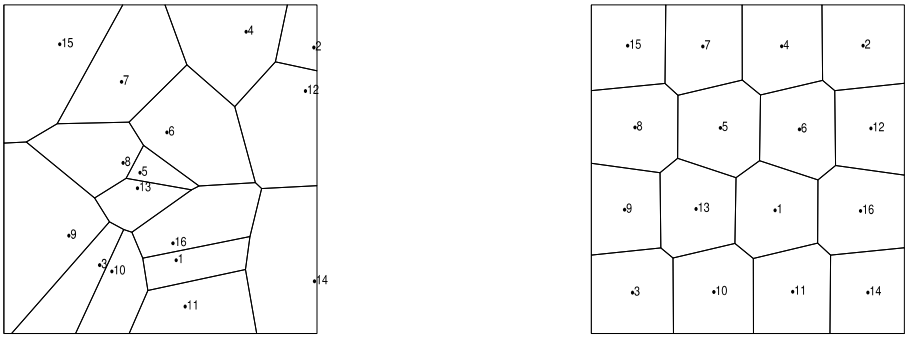
2.2.1 Voronoi Diagram and CVT

Given a region and a set of sensors, a Voronoi diagram divides the region into a set of Voronoi cells; each point in a cell is associated with its closest sensor. Specifically, consider a convex region, A, with a density function $\rho(x)$. Let s_i denote the location of sensor i. Let vector $\mathbf{s} = \{s_i\}_{i=1}^n$ denote the location of all the sensors. The Voronoi cells, $\mathcal{V}(\mathbf{s}) = \{\mathcal{V}_i\}_{i=1}^n$, generated by $\mathbf{s}$ are defined as

$$\mathcal{V}_i = \{x \in A \,|\, \|x - s_i\| \leq \|x - s_j\|, \forall j \neq i\},$$

where $\mathcal{V}_i$ is the Voronoi cell generated by the i^{th} sensor. All the sensors whose Voronoi cells are adjacent to $\mathcal{V}_i$ are called as the *Voronoi neighbors* of sensor i. Fig. 2.2.1(a) shows an example of a Voronoi tessellation generated by 16 sensors.

Note that due to limited communication ranges of sensors, the Voronoi neighbors of a sensor may not be its real neighbors defined by its communication range. For instance, in Fig. 2.2.1(a), sensor 6 has six Voronoi neighbors. Due to limited commu-



(a) Voronoi Cells (b) CVT

FIGURE 2.2.1: Illustration of Voronoi Cells and CVT with 16 sensors in a square area.

nication range, only three of them (sensors 7, 8, and 5) are within the communication range of sensor 6, while the other three (sensor 4, 12, 16) are out of the communication range of sensor 6.

A CVT is a Voronoi tessellation where each generator of its Voronoi cells coincides with the mass centroid of the Voronoi cell. Fig. 2.2.1(b) shows a CVT corresponding to the Voronoi tessellation in Fig. 2.2.1(a). It is obtained using Lloyd's method (see Section 2.2.3). Many CVTs may be derived from the initial deployment in Fig. 2.2.1(a); Fig. 2.2.1(b) only illustrates one of them.

2.2.2 Optimization Problem for Even Self-deployment

Consider the *CVT energy function* [49, 56] defined as

$$\mathcal{F}(\mathbf{s}) = \sum_{i=1}^{N} \int_{x \in \mathcal{V}_i} \|x - s_i\|^2 \, \rho(x) \, dx. \tag{2.2.1}$$

As shown in [7, 36, 56], the gradient of $\mathcal{F}(\mathbf{s})$ is

$$\nabla \mathcal{F}(s_i) = \frac{\partial F}{\partial s_i} = 2m_i(s_i - c_i), \tag{2.2.2}$$

where m_i and c_i are the mass and centroid of Voronoi cell $\mathcal{V}_i$, respectively, and

$$\begin{aligned} m_i &= \int_{x \in \mathcal{V}_i} \rho(x)\, dx, \\ c_i &= \frac{\int_{x \in \mathcal{V}_i} \rho(x) x\, dx}{\int_{x \in \mathcal{V}_i} \rho(x)\, dx}, \end{aligned}$$

where $\rho(x)$ denotes the density at point x. In our problem, $\rho(x) \equiv 1$.

One can observe from (2.2.2) that a CVT corresponds to a critical point of $\mathcal{F}(\mathbf{s})$. The study in [56] shows that the necessary condition for $\mathcal{F}(\mathbf{s})$ to be minimized is that $\mathbf{s}$ forms a CVT. Recall that a CVT corresponds to an even deployment according to Gersho's conjecture [28]. Hence the sensor locations that minimize the CVT energy function $\mathcal{F}$ form an even deployment. Therefore, the even-deployment problem can be formulated as an unconstrained minimization problem as

$$\mathbf{s}_r = \arg\min_{\mathbf{s}} \mathcal{F}(\mathbf{s}), \tag{2.2.3}$$

where $\mathbf{s}_r$ corresponds to a CVT of the target area. Note that for a given set of sensors and target area, there may exist multiple CVTs which correspond to the local minimizers and the global minimizer of the CVT energy function [56]. Any CVT can form an even deployment.

To solve the even self-deployment problem defined in (2.2.3), iterative algorithms are generally used. In these algorithms, the locations of sensors are iteratively up-

dated in the direction of the negative gradient of the CVT energy function, until the algorithm converges. If the algorithm is distributed, then the sensors compute the moving direction with the local information and move iteratively to minimize the CVT energy function. In particular, let $\mathbf{s_k}$ denote sensor positions in the k^{th} iteration. Let the movement step of N sensors be defined by the vector $\mathbf{p} = \{p_1, p_2, \ldots, p_N\}$. Then, the movement step $\mathbf{p}_k$ in the k^{th} iteration is computed as,

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \mathcal{F}(\mathbf{s_k} + \mathbf{p}), \tag{2.2.4}$$

and the sensor location vector is then updated as

$$\mathbf{s_{k+1}} = \mathbf{s_k} + \mathbf{p_k}. \tag{2.2.5}$$

Due to the iterative nature of the method, the sensors are required to be synchronized with each other in some fashion. The algorithm converges to a CVT when $\mathbf{p}_k = 0$.

2.2.3 Lloyd's Method

A widely used algorithm to construct a CVT is Lloyd's method [56]. In Lloyd's method, the sensors' locations are updated to the centroids of their Voronoi cells in each iteration. Subsequently, the Voronoi cells are computed again and the process is iterated until an approximate CVT of the target area is generated[1]. Since sensors can use distributed algorithms (e.g., those in [13]) to compute Voronoi cells and Voronoi

[1]Note that in [22] Lloyd's method can be an asynchronous algorithm in which sensors compute movement step and change destination on the go. However, this requires the sensors to update Voronoi cell continuously, which is not practical in sensor networks.

neighbors, the movement step can be computed distributedly based on the neighbor information. To apply it to mobile sensor deployment, we let each iteration consist of two phases: i) neighbor discovery phase and ii) movement phase. In the neighbor discovery phase, sensors exchange their location information with their neighbors. At the end of the neighbor discovery phase, sensors compute their Lloyd movement step, referred to as *Lloyd step*, as

$$p_i = c_i - s_i, \qquad i = 1, ..., N \tag{2.2.6}$$

Note that a sensor may not be within the communication ranges of its Voronoi neighbors. The method described in Section 2.5 can be used to deal with such cases.

2.3 Energy-Efficient Even Self-deployment

This section formulates a locational optimization problem that achieves even deployment while taking into account sensor traveling distances. Subsequently, it is shown that Lloyd's method provides an approximate solution to this problem.

2.3.1 Problem Statement

In order to save energy consumption, it is essential to reduce the traveling distances of sensors during self-deployment. In this regard, the iterative form in Eq. (2.2.4) is modified by adding a penalty function for the lengths of sensors' movement steps in each iteration. More specifically, the desired energy-efficient movement step in the

k^{th} iteration becomes

$$\mathbf{p_k} = \arg\min_{\mathbf{p}} \left(\mathcal{F}(\mathbf{s_k} + \mathbf{p}) + \frac{1}{2}\mathbf{p}^T \Gamma_k \mathbf{p} \right), \tag{2.3.1}$$

where the second term on the right hand side, $\frac{1}{2}\mathbf{p}^T\Gamma_k\mathbf{p}$, represents the penalty function and Γ_k is a diagonal matrix with positive elements.

The above method can be regarded as a proximal minimization algorithm [61]. When the algorithm converges, the sensor location vector $\mathbf{s}$ converges to the same minimizer as that of the energy function $\mathcal{F}(\mathbf{s})$, and hence forms a CVT. The larger the elements of Γ_k are, the smaller are the movements steps and more distance saving is expected. However, as we will see, larger elements in Γ_k also result in a larger number of deployment steps.

The introduction of the second order term $\mathbf{p}^T\Gamma_k\mathbf{p}$ in Eq. (2.3.1) requires that the iterative gradient method employs the second order information of $\mathcal{F}(\mathbf{s})$ to characterize the movement step $\mathbf{p_k}$. This leads to the application of Newton's method [54].[2] To solve for $\mathbf{p_k}$, $\mathcal{F}(\mathbf{s}_k + \mathbf{p})$ is approximated using Taylor's theorem as

$$\mathcal{F}(\mathbf{s}_k + \mathbf{p}) \approx \mathcal{F}(\mathbf{s}_k) + \mathbf{g}(\mathbf{s}_k)^T\mathbf{p} + \frac{1}{2}\mathbf{p}^T H(\mathbf{s}_k)\mathbf{p}, \tag{2.3.2}$$

where $H(\mathbf{s}_k)$ and $\mathbf{g}(\mathbf{s}_k)$ are the Hessian matrix and the gradient vector of $\mathcal{F}$ at $\mathbf{s}_k$, respectively. Substituting Eq. (2.3.2) into Eq. (2.3.1) yields

$$\mathbf{p_k} \approx \arg\min_{\mathbf{p}} \left(\mathcal{F}(\mathbf{s}_k) + \mathbf{g}(\mathbf{s}_k)^T\mathbf{p} + \frac{1}{2}\mathbf{p}^T(H(\mathbf{s}_k) + \Gamma_k)\mathbf{p} \right). \tag{2.3.3}$$

[2]Newton's method requires $\mathcal{F}$ to be at least C^2 [54]. A recent study [49] proved C^2 smoothness of $\mathcal{F}$ in any convex and most non-convex 2D domains.

Setting the derivative of the argument in the right hand side of Eq. (2.3.3) to zero, we obtain the energy-efficient movement step (EE-step) as

$$\mathbf{p_k} = -\left(H(\mathbf{s_k}) + \Gamma_k\right)^{-1} \mathbf{g}(\mathbf{s_k}). \tag{2.3.4}$$

Computing the EE-step via Eq. (2.3.4) requires the computation of the Hessian matrix and its inverse, which in general needs global information of sensor locations. As will be shown later, this computation can be done via cooperation among sensors in a distributed manner in mobile sensor networks.

2.3.2 Lloyd step vs EE-step

This subsection shows that the movement step in Lloyd's method is an approximation to the EE-step with a non-optimal choice of Γ_k. Furthermore, decreasing the step size of Lloyd's method leads to less traveling distance while maintaining the convergence property.

Note that if the Hessian matrix H is a diagonal matrix, $\mathbf{p_k}$ can be computed distributedly. Let H' denote a diagonal matrix that only contains the diagonal elements of H. Approximating H using H' yields

$$\mathbf{p_k} \approx -\left(H'(\mathbf{s_k}) + \Gamma_k\right)^{-1} \mathbf{g}(\mathbf{s_k}). \tag{2.3.5}$$

Following the results in [7,36], the i^{th} element in the diagonal part of H is $2m_i - \theta_i$, where m_i is the mass of the Voronoi cell $\mathcal{V}_i$ in the k^{th} iteration, and θ_i is a positive number (the exact form is described in [7,36]). Setting the i^{th} diagonal element of Γ_k

to θ_i, the movement step p_i of the i^{th} sensor in the k^{th} iteration becomes

$$p_i = -\left(2m_i - \theta_i + \theta_i\right)^{-1} \nabla \mathcal{F}(s_i), \quad i = 1, ..., N. \tag{2.3.6}$$

Substituting Eq. (2.2.2) into the above yields

$$p_i = -\left(2m_i\right)^{-1} 2m_i(s_i - c_i) = c_i - s_i, \quad i = 1, ..., N \tag{2.3.7}$$

which is equivalent to the movement step in Lloyd's method as shown in Eq. (2.2.6). The above implies that the movement step in Lloyd's method is an approximation of the EE-step — it is obtained by approximating the Hessian matrix by keeping only the diagonal elements and choosing a specific form of Γ_k (i.e., setting the i^{th} diagonal element of Γ_k to θ_i). This choice of Γ_k may not be optimal. In fact, we can expect to save more distance by increasing the diagonal elements of Γ_k. For example, if we increase the i^{th} diagonal elements in Γ_k to $(2m_i + \theta_i)$, then the movement step becomes

$$p_i = \frac{1}{2}(c_i - s_i), \quad i = 1, ..., N \tag{2.3.8}$$

Equivalently,

$$\mathbf{p_k} \approx \frac{1}{2}(\mathbf{c}_k - \mathbf{s}_k), \tag{2.3.9}$$

which is half of the movement step in Lloyd's method.

2.3.3 Lloyd-α Method

By choosing Γ_k appropriately, we can set the EE-step to an arbitrary fraction, α, of Lloyd step. This method is referred to as Lloyd-α method. The convergence of the

method is shown in Proposition 2.3.1.

Proposition 2.3.1. *Suppose the movement step of N sensors in Lloyd's method is given by a vector $\mathbf{p} = \{p_1, p_2, \ldots, p_N\}$. If the sensors take a movement step $\mathbf{p}' = \{p'_1, p'_2, ..., p'_N\}$ satisfying that $p'_i = \alpha_i p_i$, $\alpha_i \in (0,1)$, $\forall i \in [1, N]$ in all iterations, $\mathbf{k} = 1, 2, ...$, then the sequence of sensor positions $\mathbf{s_1}, \mathbf{s_2}, ...\mathbf{s_k}, ...$ converges to a CVT of the target area.*

Proof. Let $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ denote an arbitrary tessellation of N sensors in the area. We define

$$\mathcal{F}(\mathbf{s}, \mathcal{T}) = \sum_{i=1}^{N} \int_{x \in \mathcal{T}_i} \|x - s_i\|^2 \rho(x)\, dx, \tag{2.3.10}$$

When $\mathcal{T} = \mathcal{V}$, we obtain energy function (2.3.1). With the property of Voronoi tessellation, we have

$$\mathcal{F}(\mathbf{s}, \mathcal{V}) \leq \mathcal{F}(\mathbf{s}, \mathcal{T}), \tag{2.3.11}$$

with strict inequality if $\mathcal{T} \neq \mathcal{V}$. Let $\mathbf{c} = \{c_1, c_2, \ldots, c_N\}$ and $\mathbf{m} = \{m_1, m_2, \ldots, m_N\}$ denote the centroids and mass of $\mathcal{V}$ respectively. Applying the parallel axis theorem [22], we can rewrite $\mathcal{F}(\mathbf{s}, \mathcal{T})$ as

$$\mathcal{F}(\mathbf{s}, \mathcal{T}) = \sum_{i=1}^{N} \int_{x \in \mathcal{T}_i} \|x - c_i\|^2 \rho(x)\, dx + m_i \|s_i - c_i\|, \tag{2.3.12}$$

Let $\mathbf{s}' = \{s'_1, s'_2, \ldots, s'_N\}$ denote the new positions of sensors in the beginning of next iteration, i.e., $\mathbf{s}' = \mathbf{s} + \mathbf{p}'$. Since $p'_i = \alpha_i p_i = \alpha_i(c_i - s_i), \alpha_i \in (0,1)$, we have $\|s'_i - c_i\| = (1 - \alpha_i)\|s_i - c_i\| < \|s_i - c_i\|$. Together with equation (2.3.12), we have

$$\mathcal{F}(\mathbf{s}', \mathcal{T}) \leq \mathcal{F}(\mathbf{s}, \mathcal{T}), \tag{2.3.13}$$

with strict inequality if $\mathbf{s}$ is not a CVT. Let $\mathcal{V}'$ denote the Voronoi tessellation formed by $\mathbf{s}'$, from equations (2.3.11) and (2.3.13), we have

$$\mathcal{F}(\mathbf{s}', \mathcal{V}') \leq \mathcal{F}(\mathbf{s}', \mathcal{V}) \leq \mathcal{F}(\mathbf{s}, \mathcal{V}). \tag{2.3.14}$$

The above equation shows that new positions of sensors will decrease the energy function until sensors form a CVT, which concludes the proof. □

Simulation results show that compared to the original Lloyd's method, Lloyd-α method using partial step sizes saves traveling distance. However, it requires a larger number of deployment steps. For example, if half of the movement step of Lloyd's method is used, i.e., $\alpha = 0.5$, then the number of deployment steps is approximately doubled. Thus, the excessive energy consumed in start/stop operations may cancel out the energy saved in the traveling distance. This limits the application of this algorithm only to the scenarios in which the cost of start/stop operations is relatively low. The study in [49] shows that incorporating more information from the second order term of the CVT energy function, i.e., the Hessian matrix, can achieve significantly less iterations before convergence (corresponding to less deployment steps in our context). This motivates us to propose a new distributed algorithm, as described in Section 2.4, that carefully chooses Γ_k to reduce both the number of deployment steps and sensor traveling distances.

2.4 Distributed Energy Efficient self-Deployment (DEED) Algorithm

This section presents a new algorithm, named *Distributed Energy Efficient self-Deployment* (DEED). It is an iterative algorithm where in each iteration a sensor moves according to the EE-step. It has two main objectives: i) saving the traveling distances of the sensors and ii) reducing the number of deployment steps. To achieve these two objectives, Γ_k needs to be chosen carefully. In the following, we first describe how to choose Γ_k, and then describe how to compute EE-step in a distributed manner. In the end, we present the workflow of DEED algorithm.

2.4.1 Choice of Γ_k

Recall that the EE-step is a movement step that minimizes the CVT energy function $\mathcal{F}$ in Eq. (2.2.1) such that the EE-step lies in the direction of decreasing $\mathcal{F}$. This implies that the matrix $H(\mathbf{s_k}) + \Gamma_k$ in Eq. (2.3.4) needs to be positive definite [54]. Since the Hessian matrix H is sparse and is generally not positive definite, the elements of Γ_k should be large enough to make the matrix $H(\mathbf{s_k}) + \Gamma_k$ positive definite. Reference [54] suggests that Γ_k can take the form of $\epsilon_k \mathbf{I}$, where ϵ_k is a constant. Furthermore, if ϵ_k satisfies that $\epsilon_k = 0$ if the Hessian is positive definite and $\epsilon_k = -\lambda_{\min}(H) + \delta$ if the Hessian is positive indefinite, where $\lambda_{\min}(H)$ denotes the the minimum eigenvalue of H and $\delta > 0$ is a small positive constant, then $\Gamma_k = \epsilon_k \mathbf{I}$ is the matrix with the minimum Euclidean norm that makes $H(\mathbf{s_k}) + \Gamma_k$ positive definite [54]. The study of [49] demonstrates this choice of ϵ_k is effective in reducing the number of iterations. This choice of ϵ_k, however, may not be favorable in saving sensor traveling distances. Indeed in the above, ϵ_k takes small values so that $H(\mathbf{s_k}) + \Gamma_k$

is close to $H(\mathbf{s_k})$ in order to reduce the number of iterations (i.e., deployment steps in our context) [49] [54], while to save traveling distances, larger elements in Γ_k are preferable as described in Section 2.3.2. To achieve the dual objectives of reducing the number of deployment steps as well as saving sensor traveling distances, we set $\Gamma_k = \epsilon_k \mathbf{I}$ and choose the value of ϵ_k as follows.

Let a_{ij} denote the element on the i^{th} row and j^{th} column of the Hessian matrix $H(\mathbf{s_k})$ in the k^{th} iteration. Then the value of ϵ_k is chosen as

$$\epsilon_k = \max\{0, -\min_i\{a_{ii} - \sum_j |\, a_{ij}\,|\}\} + \delta, \tag{2.4.1}$$

where $\delta > 0$ is a positive constant. If H_+ is defined as

$$H_+ = H(\mathbf{s_k}) + \Gamma_k = H(\mathbf{s_k}) + \epsilon_k \mathbf{I}, \tag{2.4.2}$$

then it can be shown that the choice of ϵ_k in Eq. (2.4.1) makes the matrix H_+ positive definite as stated in the following lemma.

Lemma 2.4.1. *When setting ϵ_k as in Eq. (2.4.1), $\lambda_{\min}(H_+) \geq \delta$, where $\lambda_{\min}(H_+)$ denotes the minimum eigenvalue of H_+. In addition, the matrix H_+ is positive definite.*

Proof. We first prove the first statement. Recall Gerschgorin Theorem [39]. Let matrix $B \in \mathbb{R}^{n\times n}$ be symmetric with eigenvalues $\lambda_1, \ldots, \lambda_n$. Then

$$\min_{1\leq i\leq n} \lambda_i \geq \min_{1\leq i\leq n} \left\{ b_{ii} - \sum_{j=1, j\neq i}^{n} |\, b_{ij}\,| \right\},$$

where b_{ij} is the element of B on the ith row and jth column.

Let a_{ij} denotes the element of H on ith row and jth column. let $u_j = \sum_{j=1,j\neq i}^{n} \mid a_{ij} \mid$. Since $H_+ = H + \epsilon \mathbf{I}$, according to Gerschgorin Theorem, we have

$$\begin{aligned} \lambda_{\min}(H_+) &\geq \min_{1\leq i\leq n} \{\epsilon + a_{ii} - u_j\} \\ &= \left(\epsilon + \min_{1\leq i\leq n} \{a_{ii} - u_j\}\right), \end{aligned}$$

(i) if $\min_{1\leq i\leq n} \{a_{ii} - u_j\} < 0$, we have $\epsilon = -\min_{1\leq i\leq n} \{a_{ii} - u_j\} + \delta$ and

$$\lambda_{\min}(H_+) \geq \delta,$$

(ii) if $\min_{1\leq i\leq n} \{a_{ii} - u_j\} \geq 0$, we have $\epsilon = \delta$ and

$$\lambda_{\min}(H_+) \geq \min_{1\leq i\leq n} \{a_{ii} - u_j\} + \delta \geq \delta.$$

The second statement that H_+ is a positive definite matrix follows directly from above and the symmetry of H. □

Setting ϵ_k as in Eq. (2.4.1) has the following three advantages compared to the choice of ϵ_k in [49]. First, to save sensor traveling distance, the value of ϵ_k should not be too small. When H is positive indefinite, ϵ_k in Eq. (2.4.1) is always larger than $-\lambda_{\min}(H)$ (see the proof of Lemma 2.4.1), and hence is more desirable in saving sensor traveling distance. Secondly, the matrix H_+ and $\lambda_{\min}(H)$ are difficult to evaluate in a distributed manner. The widely adopted methods that compute the minimum modification to make the Hessian matrix positive definite (e.g., the modified Cholesky factorization [54, 64]) cannot be employed since they use centralized algorithms and have high complexities. In contrast, computing ϵ_k is easy to implement on distributed

sensors. It needs global consensus but requires much less overheads. Thirdly, as to be shown in Theorem 2.4.1, such choice of ϵ_k leads to the convergence of the Jacobi method which is the method used to distribute the computation.

Note that if a sensor is not within the communication ranges of its Voronoi neighbors or the network is disconnected, the Hessian matrix H and ϵ_k may not be computed correctly. The method described in Section 5 can be used to deal with such cases.

The local convergence property of applying the EE-step on a general function is shown by the following proposition.

Proposition 2.4.1. *Let the function $\mathcal{F}(\mathbf{s_k})$ satisfy the condition that its Hessian matrix H is locally Lipschitz continuous around an optimal point set $\mathbf{s}^*$. Furthermore, let the sequence $\{\mathbf{s_k}\}$ and all elements of H be bounded in a finite domain. Then if the starting point $\mathbf{s_0}$ is sufficiently close to the optimal set $\mathbf{s}^*$, then the sequence of iterative points generated by the solution of Eq. (2.3.4) and Eq. (2.4.1) converges to $\mathbf{s}^*$, and the rate of convergence is at least linear.*

Before proving Proposition 2.4.1, we first prove a lemma.

Lemma 2.4.2. *Define $\mathbf{s}^*$ as the optimal point of function $\mathcal{F}(\mathbf{s})$ where $\mathbf{g}(\mathbf{s}^*) = 0$. Consider any functions $\mathcal{F}(\mathbf{s})$ that is twice differentiable and whose Hessian matrix $H(\mathbf{s})$ is locally Lipschitz continuous with Lipschitz constant L around optimal point set $\mathbf{s}^*$ within region $\|\mathbf{s} - \mathbf{s}^*\| \leq r$. Do iterations as $\mathbf{s}_{k+1} = \mathbf{s_k} + \mathbf{p_k}$, where $\mathbf{p_k}$ is computed by $\mathbf{p_k} = -H_+^{-1}(\mathbf{s_k})\mathbf{g}(\mathbf{s_k})$ and $H_+ = H(\mathbf{s_k}) + \epsilon_k \mathbf{I}$. Then, if the starting point $\mathbf{s}_0$ satisfies the condition $\|\mathbf{s}_0 - \mathbf{s}^*\| \leq r$, the sequence of iterates $\mathbf{s_k}$ follows inequality below*

$$\|\mathbf{s}_{k+1} - \mathbf{s}^*\| \leq \frac{L}{2\delta}\left(\|\mathbf{s_k} - \mathbf{s}^*\|^2 + 2\epsilon_k \|\mathbf{s_k} - \mathbf{s}^*\|\right).$$

Proof. We denote $\mathbf{g}(\mathbf{s_k})$ and $\mathbf{g}(\mathbf{s}^*)$ as $\mathbf{g}_k$ and $\mathbf{g}^*$ respectively. From the definition of the iteration step and the optimality condition $\mathbf{g}^* = 0$, we have

$$\begin{aligned}
& \mathbf{s}_{k+1} - \mathbf{s}^* \\
= \; & \mathbf{s_k} + p_k - \mathbf{s}^* \\
= \; & \mathbf{s_k} - \mathbf{s}^* - \left(H(\mathbf{s_k}) + \epsilon_k \mathbf{I}\right)^{-1} \mathbf{g}_k \\
= \; & \left(H(\mathbf{s_k}) + \epsilon_k \mathbf{I}\right)^{-1} \\
& \left[\left(H(\mathbf{s_k}) + \epsilon_k \mathbf{I}\right)(\mathbf{s_k} - \mathbf{s}^*) - (\mathbf{g}_k - \mathbf{g}^*)\right], \qquad (2.4.3)
\end{aligned}$$

According to Taylor's theorem [54],

$$\mathbf{g}_k - \mathbf{g}^* = \int_0^1 H\left(\mathbf{s_k} + t(\mathbf{s}^* - \mathbf{s_k})\right)(\mathbf{s_k} - \mathbf{s}^*)dt, \qquad (2.4.4)$$

Then choosing $\mathbf{s}_0$ so that $\|\mathbf{s} - \mathbf{s}^*\| \leq r$, for $k \geq 0$ we have

$$\begin{aligned}
& \|(H(\mathbf{s_k}) + \epsilon_k \mathbf{I})(\mathbf{s_k} - \mathbf{s}^*) - (\mathbf{g}_k - \mathbf{g}^*)\| \\
= \; & \left\| \int_0^1 \left[H(\mathbf{s_k}) + \epsilon_k \mathbf{I} - H(\mathbf{s_k} + t(\mathbf{s}^* - \mathbf{s_k}))\right] \right. \\
& \left. (\mathbf{s_k} - \mathbf{s}^*)dt \right\| \\
\leq \; & \int_0^1 \left(\|H(\mathbf{s_k}) - H(\mathbf{s_k} + t(\mathbf{s}^* - \mathbf{s_k}))\| + \epsilon_k\right) \\
& \|\mathbf{s_k} - \mathbf{s}^*\| \, dt \\
\leq \; & \int_0^1 \left(Lt \|\mathbf{s_k} - \mathbf{s}^*\| + \epsilon_k\right) \|\mathbf{s_k} - \mathbf{s}^*\| \, dt \\
= \; & \frac{1}{2} L \|\mathbf{s_k} - \mathbf{s}^*\|^2 + \epsilon_k \|\mathbf{s_k} - \mathbf{s}^*\|, \qquad (2.4.5)
\end{aligned}$$

Let $\lambda_{\max}(H)$ denote maximum eigenvalue of H.

$$\begin{aligned}\left\|(H(\mathbf{s_k})+\epsilon_k\mathbf{I})^{-1}\right\| &= \lambda_{\max}\left((H(\mathbf{s_k})+\epsilon_k\mathbf{I})^{-1}\right)\\ &= \frac{1}{\lambda_{\min}\left(H(\mathbf{s_k})+\epsilon_k\mathbf{I}\right)}\end{aligned}$$

Since Lemma 2.4.1 tells us that $\lambda_{\min}(H_+) \geq \delta$, we have

$$\left\|(H(\mathbf{s_k})+\epsilon_k\mathbf{I})^{-1}\right\| \leq \frac{1}{\delta} \tag{2.4.6}$$

By substituting in (2.4.3) and (2.4.5), we obtain

$$\|\mathbf{s}_{k+1}-\mathbf{s}^*\| \leq \frac{L}{2\delta}\left(\|\mathbf{s_k}-\mathbf{s}^*\|^2 + 2\epsilon_k\|\mathbf{s_k}-\mathbf{s}^*\|\right). \tag{2.4.7}$$

□

We now prove Proposition 2.4.1.

Proof. Let L denote the local Lipschitz constant of H. Since all elements in H are bounded. Thus the value of ϵ computed by (2.4.1) should be upper bounded. We denote this upper bound as $\bar{U}_\epsilon$. Since the sequence $\{\mathbf{s_k}\}$ are all bounded in a finite domain, $\|\mathbf{s_k}-\mathbf{s}^*\|$ should also be upper bounded. We denote the upper bound as $\bar{U}_d$. Then, following from Lemma 2.4.2, we have,

$$\begin{aligned}\|\mathbf{s}_{k+1}-\mathbf{s}^*\| &\leq \frac{L}{2\delta}\left(\|\mathbf{s_k}-\mathbf{s}^*\|^2 + 2\epsilon\|\mathbf{s_k}-\mathbf{s}^*\|\right)\\ &\leq \widetilde{U}\|\mathbf{s_k}-\mathbf{s}^*\|,\end{aligned}$$

where $\widetilde{U} = \frac{L(\bar{U}_d+2\bar{U}_\epsilon)}{2\delta}$. Choosing $\mathbf{s}_0$ so that $\|\mathbf{s}_0-\mathbf{s}^*\| \leq 1/(2\widetilde{U})$, we can inductively

deduce that the sequence converges to $\mathbf{s}^*$, and the rate of convergence is linear. $\square$

It is seen from the proof of Proposition 2.4.1 that the local convergence rate depends on ϵ_k. A large ϵ_k will result in a linear rate of convergence which leads to a larger number of deployment steps, while a small ϵ_k results in nearly quadratic rate of convergence which leads to a smaller number of deployment steps.

2.4.2 Distributed Realization

This section describes how to compute the EE-steps distributively among sensors. From Eq. (2.3.4) and Eq. (2.4.2), it is observed that the EE-step, $\mathbf{p}_k$, is the solution to a system of linear equations

$$H_+\mathbf{p}_k = -\mathbf{g}(\mathbf{s}_k). \tag{2.4.8}$$

Therefore, we can use distributed iterative methods such as the Jacobi method [38,51] or the GaBP method [66] to obtain $\mathbf{p}_k$. In this chapter, the Jacobi method is chosen since it is simpler and more suitable for distributed sensor network applications. The Jacobi method decomposes H_+ into two matrices, a diagonal matrix $\mathcal{D}$ and a remainder matrix $\mathcal{R}$, i.e., $H_+ = \mathcal{D} + \mathcal{R}$. Then Eq. (2.4.8) is rewritten as

$$(\mathcal{D} + \mathcal{R})\mathbf{p}_k = -\mathbf{g}(\mathbf{s}_k).$$

Multiplying $\mathcal{D}^{-1}$ on both sides yields

$$(I + \mathcal{D}^{-1}\mathcal{R})\mathbf{p}_k = -\mathcal{D}^{-1}\mathbf{g}(\mathbf{s}_k).$$

The Jacobi method uses an iterative approach. More specifically, let $\mathbf{p}_k(t)$ denote the solution obtained at the t^{th} iteration. Then,

$$\mathbf{p}_k(t+1) = -\mathcal{D}^{-1}\left(\mathcal{R}\mathbf{p}_k(t) + \mathbf{g}(\mathbf{s}_k)\right). \tag{2.4.9}$$

Note that $\mathcal{D} = \mathcal{D}' + \epsilon_k \mathbf{I}$ where $\mathcal{D}'$ is the diagonal part of the Hessian matrix $H(\mathbf{s}_k)$. Thus, we get

$$\mathbf{p}_k(t+1) = -(\mathcal{D}' + \epsilon_k \mathbf{I})^{-1}\left(\mathcal{R}\mathbf{p}_k(t) + \mathbf{g}(\mathbf{s}_k)\right). \tag{2.4.10}$$

We observe two facts from the explicit formula of Hessian H [7, 36]. First, H is a sparse matrix in the sense that if two sensors are not Voronoi neighbors, then the elements corresponding to these two sensors are zeros. Second, if the location information of all the Voronoi neighbors is given, then each sensor can compute all the elements of its corresponding two rows[3] of H and thus $\mathcal{D}'$ and $\mathcal{R}$.

Therefore, if the Voronoi neighbors can share the information of their movement steps computed in iteration t via communication and since $\mathbf{g}(\mathbf{s}_k)$ can be computed distributively, the Jacobi iterative process in Eq. (2.4.10) can be carried out in a distributed manner, though it needs more communication overheads. The Jacobi iterative process converges to the solution of Eq. (2.4.8) when H_+ is strictly diagonally dominant [51]. The following lemma states that this condition holds if the choice of ϵ_k follows Eq. (2.4.1);

Lemma 2.4.3. *H_+ is a strictly diagonally dominant matrix if ϵ_k is computed as in Eq. (2.4.1).*

[3]Note that since sensors move in two-dimensional space, there are two rows corresponding to each sensor in the Hessian matrix.

Proof. According to equation (2.4.1),

- if H is strictly diagonally dominant itself, $\epsilon = \delta$ and $H_+ = H + \epsilon\mathbf{I}$ is still diagonally dominant since $\delta > 0$.

- if H is not strictly diagonally dominant, i.e., there is at least one row k in the H satisfying $a_{kk} - \sum_i \mid a_{ki} \mid \leq 0$, $-\min_i\{a_{ii} - \sum_j \mid a_{ij} \mid\} \geq 0$ and $\epsilon = -\min_i\{a_{ii} - \sum_j \mid a_{ij} \mid\} + \delta$. It is trivial to show that for any row k that satisfies $a_{kk} - \sum_i \mid a_{ki} \mid \leq 0$, $\epsilon = -\min_i\{a_{ii} - \sum_j \mid a_{ij} \mid\} + \delta > a_{kk} - \sum_i \mid a_{ki} \mid$ since $\delta > 0$. Hence, for those rows, we have $\epsilon + a_{kk} > \sum_i \mid a_{ki} \mid$ which makes H_+ strictly diagonally dominant.

□

Combining Lemma 2.4.3 and the results in [51], we have the following theorem regarding the convergence result of Jacobi method.

Theorem 2.4.1. *The Jacobi iterative process in Eq. (2.4.10) converges to the solution of Eq. (2.4.8) from any initial step vector and the approximate number of steps needed for convergence is*

$$-\frac{1}{\ln(\lambda_{\max}(\mathcal{D}^{-1}\mathcal{R}))}$$

where $\lambda_{\max}(\mathcal{D}^{-1}\mathcal{R})$ is the largest eigenvalue of the matrix $\mathcal{D}^{-1}\mathcal{R}$.

Choice of the Initial Value $\mathbf{p}(\mathbf{0})$

In order to reduce the number of iteration steps in the Jacobi method and thus the communication energy consumption of sensors, the initial value, $\mathbf{p}(\mathbf{0})$, needs to be chosen carefully to minimize the initial error. In our simulations in Section 2.6, the

initial value is set to be the movement step computed using the Lloyd's method according to Eq. (2.2.6). In this way, sensors can compute the initial steps distributively and the choice leads to a smaller number of iterations (~ 7) in our simulations.

Propagating ϵ_k Over the Network

Note that the Jacobi process requires ϵ_k to be known ahead. Since the max operation in computing ϵ_k in Eq. (2.4.1) needs global information, this requires extra packet exchanges to propagate ϵ_k over the network which leads to excessive energy consumption. To address this issue, observe that the max operation is over Hessian rows. Thus, a simple method is proposed in which ϵ_k propagates together with the Jacobi iterations. In this method, a sensor does not need to know the value of ϵ_k which is the global max before the Jacobi iteration. Instead, in the first iteration of the Jacobi process, it computes ϵ_k following Eq. (2.4.1) which is its local max, i.e., max value computed from its corresponding two rows. Then starting from the first iteration, it always inserts the current local max to the packet used in Jacobi process for message exchange. Other sensors will update their local max to the larger one received. The method requires the sensors to send out their corresponding diagonal elements in Hessian for their neighbors to update their steps with the updated local max. In this fashion, the global max will spread over the network together with the Jacobi packets without extra overhead. One should note that even when ϵ_k is not propagated over all the network, the way ϵ_k is computed and propagated does not violate the convergence property of Jacobi process shown in Theorem 2.4.1.

2.4.3 Workflow of DEED Algorithm

In DEED algorithm, sensors move in iterations. The workflow of one iteration of the DEED algorithm at each sensor is described in Algorithm 1. Each iteration starts with a neighbor discovery phase, followed by a Jacobi phase to perform Jacobi computations and finally a movement phase. The Jacobi phase is separated from neighbor discovery phase as the Hessian information can only be known after the neighbor discovery phase. The Jacobi phase is further divided into a predefined number of smaller "computation" sub-phases, each of which corresponds to an iteration in the Jacobi iterative process. In each Jacobi iteration, a new EE-step is computed using equation (2.4.10). Then in the following movement phase, sensors take the movement step as computed in the Jacobi phase. If a sensor fails to hear from one of its neighbors in any computation sub-phase, it simply takes a Lloyd movement step instead.

2.5 Computing Voronoi Cells with Limited Communication Range

The distributed Voronoi cell computation relies on the Voronoi neighbor information [13]. However, in mobile sensor networks, the communication ranges of sensors are limited. A sensor may not be within the communication ranges of its Voronoi neighbors. As a consequence, the distributed computation of Voronoi cells may not be feasible. Fig. 2.5.1(a) shows an example where S_1, S_2, S_3, S_4, S_5 and S_6 are six sensors. The computed Voronoi cell of S_1 should be the polygon $V_1V_2V_3V_4V_5$. However, since S_1 is located outside the communication range of S_2 and is not aware of the existence of S_2, it computes its Voronoi cell as the polygon $V_1V_2V_3V_6$. This incor-

Algorithm 1 One iteration of DEED algorithm at each sensor

▷*Discovery phase starts*

Discover neighbors and exchange locations with neighbors

▷*Discovery phase ends*

▷*Jacobi phase starts*

Compute the corresponding rows of the Hessian matrix and its Lloyd steps

for all *computation sub-phases* **do**

▷*Computation sub-phase starts*

Delay a random period and exchange following information with neighbors

- ϵ (local max)
- diagonal elements on its row
- Lloyd steps for the first computation sub-phase or computed EE-steps at the end of previous computation sub-phase

repeat Listen to the information from neighbors

if neighbor information received **then**

Save received EE-steps of the neighbor

if a larger ϵ is received **then**

Update previous computed EE-steps and received steps using received ϵ and diagonal elements

Apply received ϵ in future computation

end if

end if

until information of all neighbors received or the end of the computation sub-phase is reached

if fail to hear from any neighbor **then**

Skip following computation sub-phases and set EE-steps to Lloyd steps

end if

Compute new EE-steps based on received EE-steps of neighbors using equation (2.4.10)

▷*Computation sub-phase ends*

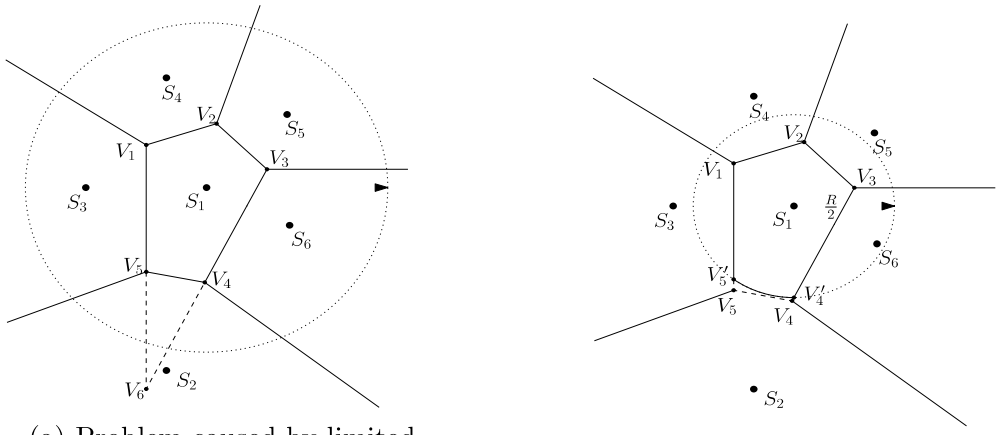
end for

▷*Jacobi phase ends*

▷*Movement phase starts*

Move according to EE-steps computed in the Jacobi phase

▷*Movement phase ends*



(a) Problem caused by limited communication range

(b) Using $R/2$-circle to bound Voronoi cell

FIGURE 2.5.1: An example showing incorrect Voronoi cell computation due to limited communication range of sensor R and the proposed algorithm to deal with limited communication range.

rect computation of Voronoi cells may lead to oscillatory movement in both Lloyd's method and DEED; thereby increases the traveling distance and prevents algorithms from converging. For example, in Lloyd's method, we can see that S_1 moves more than needed towards S_2 and may travel back after it hears from S_2.

To address the issue of distributed Voronoi cell computation, an intuitive algorithm is proposed as follows. This algorithm is applicable to both Lloyd's method (the original Lloyd's method and Lloyd-α) and DEED. Let R denote the communication ranges of sensors; a circle with radius $R/2$ and centered at a sensor is referred to as the "$R/2$-circle" of the sensor. Clearly, if two sensors are out of the communication range of each other, then their $R/2$-circles do not overlap. In the proposed method, each sensor uses the overlapped area of its computed Voronoi cell and its $R/2$-circle as its Voronoi cell. For example, the resultant Voronoi cell computed by S_1 in Fig. 2.5.1(a) is the polygon $V_1V_2V_3V_4'V_5'$ in Fig. 2.5.1(b). In this special case, the resultant Voronoi cell is very close to the correct one. The intuition behind this

algorithm is that the resultant Voronoi cell will not contain points that are in the Voronoi cells of other sensors. In this way, a sensor with no neighbors will stay at its position, while a sensor with incomplete neighbor information will take steps that tend to form an even deployment within the connected neighbors bounded by the $R/2$ circles of the sensor and its neighbors. Since there is no overlapped area between the $R/2$-circles of two sensors out of each other's communication range, the Hessian matrix H and ϵ_k in Eq. (2.4.1) computed for DEED algorithm will be locally correct within the connected part of the network. As the sensors spread out and learn about new neighbors gradually, the global even deployment is finally achieved.

To make sure that all sensors finally compute their correct Voronoi cells and form a global even deployment, it is assumed that in the final even deployment, the $R/2$-circles of all sensors cover the whole area. In practice, if the boundaries of the target area are known, then the required communication range R of sensors can be computed a priori. Even if the boundaries are unknown, a larger area that contains the target area can be used to compute the value of R. The transmission power of the sensor can be adjusted to get the required R. In general, sensors may have different communication ranges, the value of R here should be the minimum communication range of all sensors. For a sensor with anisotropic communication range, the value of R should be chosen as the radius of the maximum circle within the communication range of the sensor.

One should note that in [71], an approach where sensor moves at most $R/2$ during movement is proposed to solve the same problem. The method we proposed employs the $R/2$-circle instead of limiting the step size is to approximate the actual Voronoi cell so that the accuracy of Voronoi related computation can be improved.

From the simulations it is observed that Lloyd method and DEED algorithm do

not converge in a large number of deployment steps if the network formed by the initial deployment of the sensors is disconnected. When the above method is used to deal with the limited communication range, then both algorithms converge quickly.

2.6 Performance Evaluation

This section compares the performance of Lloyd's method and DEED algorithm through simulation in NS-2 [2] (version 2.35). The simulation settings are described first, followed by the simulation results.

2.6.1 Simulation Settings

The DEED and Lloyd's algorithms are implemented in two agent models. The mobile node model is used with minor modifications so that the agents can control the movement of sensors directly. Sensors are synchronized by scheduling the start time of their next step in every step.

The settings of the physical model follow IEEE 802.11p [4], the MAC protocol is IEEE 802.11. The sensors use omnidirectional antenna. The two ray ground model is used as the propagation model that considers reflection from the ground [59].

The computation of Voronoi partition is based on the "qvoronoi" program from Qhull [3]. Modifications have been made so that the program can compute Voronoi cells in a bounded region. At the end of the neighbor discovery phase, each sensor calls the program and uses its collected neighbor information as the input to the program. Thus, each sensor computes the Voronoi tessellation of the area from its local point of view. Subsequently, each sensor determines the intersection area of the

computed Voronoi cell and its $R/2$-circle, which is used as the Voronoi cell in the computation of its movement step. To simplify the computation, the $R/2$ circle of each sensor is approximated by a regular hexagon.

Stopping Criteria

In both algorithms, a stopping criteria is defined for the movement of sensors. If the distance between a sensor and the centroid of its Voronoi cell (i.e., the movement step of Lloyd's method) is less than 1m, then the sensor stops moving. Thus, when all sensors are close to the centroids of their Voronoi cells, i.e., close to a CVT, then they stop moving and the even deployment is completed. The reason for choosing the threshold of 1m as the stopping criteria is two-fold. Firstly, lowering the threshold further leads to an increasing number of deployment steps with little progress towards achieving a CVT. Secondly, based on the results obtained from the animation tool — network animator (NAM) [2], it is observed that with this criteria, the sensors visually form an even deployment. Furthermore, the deployment quality is visually much better than using a larger threshold, e.g., 2m.

Energy Consumption

The energy consumption statistics is accumulated over three sources — communication, movement, and start/stop operations (roughly equals to the number of deployment steps). The unified energy consumption setting is used as that in [71]. The moving distance and the number of deployment steps are normalized into message complexity. Thus, the energy consumed by movement is presented by how many packets can be transmitted with the same amount of energy. As calculated from

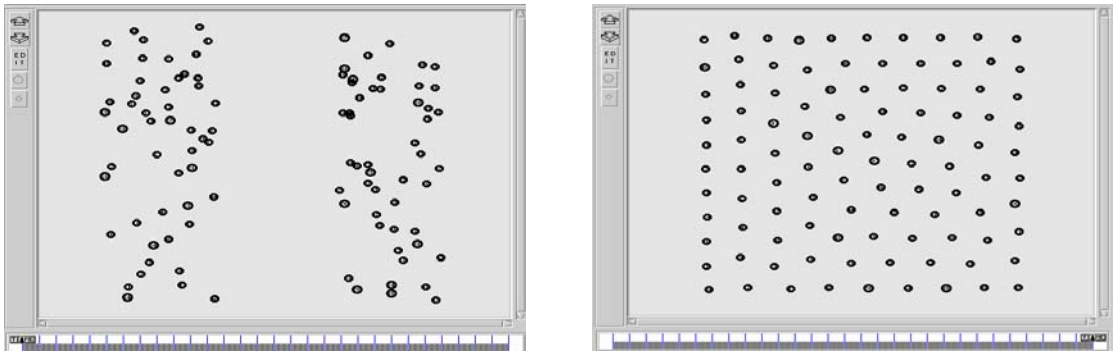
Robomote [67], moving a sensor one meter consumes a similar amount of energy as transmitting 300 messages. Thus, unless otherwise stated, the energy consumed by movement is defined as 300 messages/meter. The energy consumption in start/stop operations differs in different systems. In the unified energy consumption result as reported later, unless otherwise stated, the start/stop operation is equivalent to movement by one meter, i.e., 300 messages/step.

2.6.2 Simulation Scenarios

The simulations consist of two scenarios. The first scenario contains 100 mobile sensors that have to be evenly deployed in a 500m $\times$ 500m area. In order to make sure that the $R/2$-circles of sensors cover the whole area when they form an even deployment, the communication ranges of sensors are set to 160m by adjusting the transmission power of the sensors. The area is divided into three stripes with the width of the middle stripe being 165m. The sensors are randomly distributed in the other two stripes. In this way, the initial deployment of the sensors forms a disconnected network topology. The initial deployment and desired deployment of the scenario are shown in Fig. 2.6.1(a) and Fig. 2.6.1(b) respectively, both of which are captured in the network animator (NAM) from one of the obtained results.

The second scenario contains 25 mobile sensors that needs to be evenly deployed in a 200m $\times$ 200m area. Similarly, the communication ranges of sensors are set to 90m. As before, three stripe areas are used for initial deployment and the width of the middle area is 95m.

For each scenario, all algorithms use the same initial locations and run one by one. The simulation time is sufficiently long (approximately the time of 200 deployment

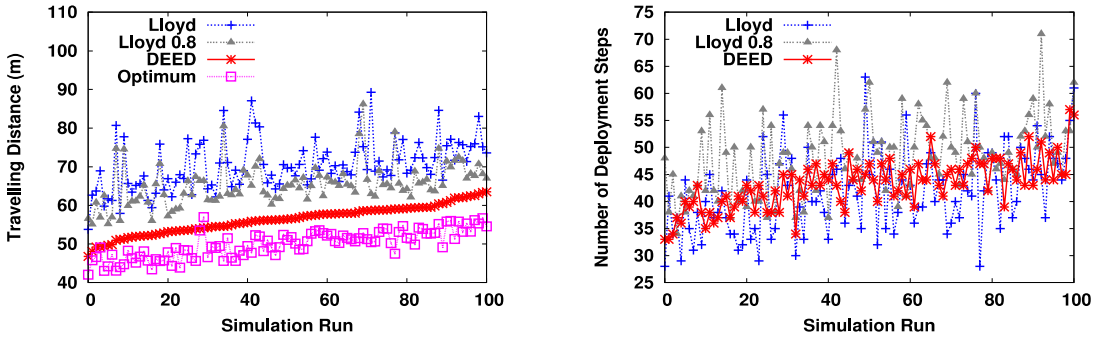


(a) Initial Deployment (b) Final Deployment

FIGURE 2.6.1: Illustration of an initial and final deployment for the simulation scenario with 100 sensors (results captured in NAM). The final deployment is obtained using DEED.

steps) to make sure that all sensors achieve the steady state within the simulation time. The process is repeated for 100 runs. In each run, we collect three statistics: the traveling distance, the number of deployment steps, and the energy consumption of sensors. These statistics are collected when all sensors stop.

Now we briefly discuss the choice of the lengths of the phases for the scenario that contains 100 sensors (the same setting is used for the scenario that contains 25 sensors). In the neighbor discovery phase, the possible packet loss is dealt with a simple mechanism. We let sensors send packets of their location information twice and wait a random time period before any transmission. Since the transmission time of one packet under the simulation setting is less than 1ms, the length of the neighbor discovery phases is set to 10s, which is enough for all 100 sensors to complete all transmissions. The speed of the sensor is set to be 1m/s. The length of the movement phase is set to 750s so that all sensors can complete the movement before next iteration starts. In DEED, the Jacobi phase consists of 3 computation sub-phases. We let sensors send one packet in each computation sub-phase and set the



(a) (b)

FIGURE 2.6.2: Average travelling distance and number of deployment steps over 100 simulation runs for the scenario with 100 sensors.

length of the Jacobi phase to 10s. We choose multiple values for the positive constant δ in (2.4.1), and find that it has little impact on the simulation results. The results reported below use $\delta = 10$.

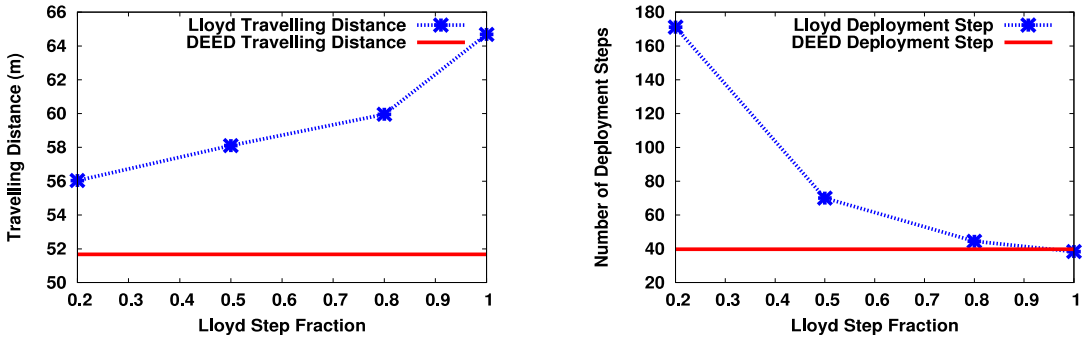
2.6.3 Simulation Results

For both simulation scenarios, the results are generated from 100 simulation runs. Both DEED and Lloyd's method converge quickly when the method described in Section 2.5 is used to adjust Voronoi cells to deal with limited sensor communication range (when not using this method, both methods fail to converge within the given simulation time). All simulation results reported below use the adjusted Voronoi cells.

For the sake of clarity, the simulation runs are indexed in increasing order according to the average traveling distance under DEED algorithm.

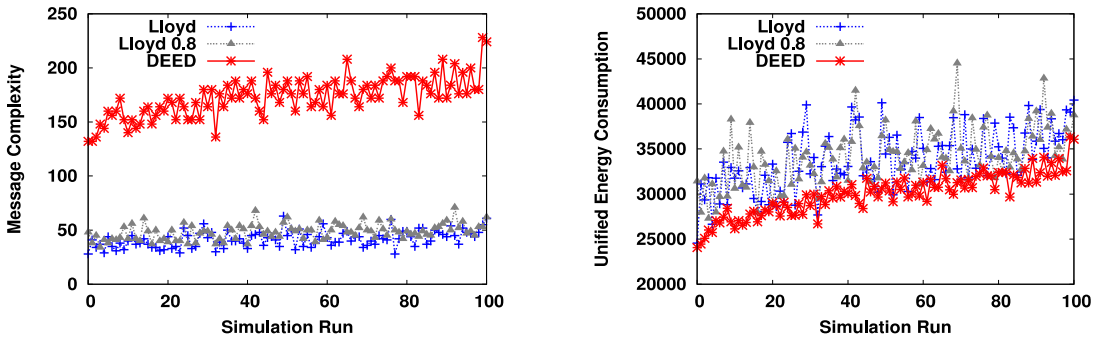
Fig. 2.6.2, 2.6.3 and 2.6.4 presents the results generated from Scenario I with 100 sensors in a 500m $\times$ 500m area.

Fig. 2.6.2(a) plots the average traveling distance over all sensors for each simulation run. The "Optimum" curve represents the average length of the optimal



(a) (b)

FIGURE 2.6.3: Comparison of Lloyd-α methods with different step fractions α.



(a) (b)

FIGURE 2.6.4: Average message complexity and unified energy consumptions over 100 simulation runs for the scenario with 100 sensors (represented using the number of messages that can be transmitted).

traveling path in Lloyd's method, i.e., the average linear distance between the initial locations and the final destinations in Lloyd's method. It is to be noticed that the traveling distance under DEED is closer to the optimum one. It results in 9%-40% less traveling distances as compared to Lloyd's method for all simulation runs with an average saving of 19%. In comparison with Lloyd-0.8, the average distance saving from DEED is 13%. Note that in some cases, DEED results in even less traveling distances than the optimum one. This is because the sensors converge to different destinations under DEED (recall that the CVT energy function can have multiple

minima in the same area).

Fig. 2.6.2(b) plots the number of deployment steps for different algorithms. Observe that DEED requires similar numbers of deployment steps when compared to Lloyd's method and the average difference is small.

The performance of Lloyd-α method was also measured with respect to different step sizes α. The results are plotted in Fig. 2.6.3 and are obtained by averaging over all simulation runs. Fig. 2.6.3 (a) shows the average traveling distance vs the Lloyd step fraction value α, while Fig. 2.6.3 (b) shows the total number of deployment steps. As expected, more distance savings are obtained for smaller step sizes. Lloyd-α algorithm requires approximately $1/\alpha$ times the number of steps than that of Lloyd's method. Among the tested Lloyd algorithms, Lloyd-0.8 is considered the most energy-efficient because it has the highest ratio of the distance saving over the increment in the number of deployment steps. For comparison, the average traveling distance and the number of deployment steps for DEED are also shown in Figs. 2.6.3 (a) and (b), respectively. The results indicate that DEED outperforms Lloyd's method and all its variants in both the average traveling distance and the number of deployment steps.

Fig. 2.6.4 (a) plots the message complexity and Fig. 2.6.4 (b) plots the average unified energy consumption over all sensors. As seen from Fig. 2.6.4 (a) and (b), DEED requires more message exchange compared to Lloyd's method and Lloyd-0.8 method due to the additional Jacobi phase; however, it still saves overall energy. This is due to the less traveling distance under DEED. Lloyd-0.8 algorithm consumes more energy on average than Lloyd's method due to larger number of deployment steps. When compared to Lloyd's method, the energy saving from DEED is up to 28% with an average saving of 13%. When compared to Lloyd-0.8, the average energy saving is 15%.

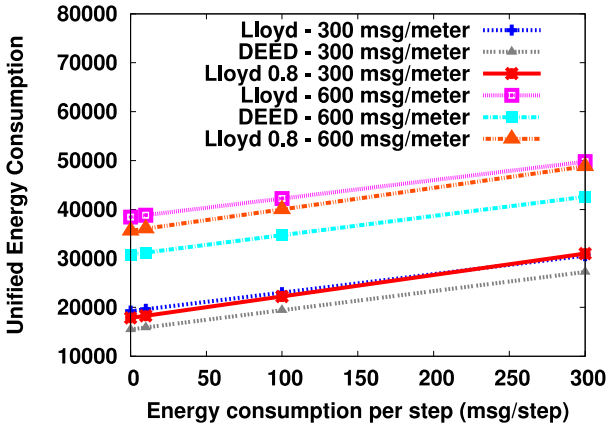


FIGURE 2.6.5: Unified energy consumption under different energy consumption models.

All the results presented so far assume that the energy consumption for moving a sensor one meter and the energy consumption per deployment step are both equivalent to the energy consumption of sending 300 messages. We next vary the energy consumption model as follows. The energy consumption for moving a sensor one meter is set to equivalence of sending either 300 or 600 messages; the energy consumption per deployment step is varied to be the equivalence of sending zero (zero energy consumption per deployment step) to 300 messages. The results are shown in Fig. 2.6.5. Clearly, DEED saves energy under all the settings. It is to be noticed that while Lloyd-0.8 consumes more energy than Lloyd's method under the default simulation settings, in other settings where the energy consumption of start/stop operation is relatively low, Lloyd-0.8 requires less energy consumption on average due to the distance savings when compared to Lloyd's method.

The results obtained in the second scenario with 25 sensors in 200m $\times$ 200m area are similar to those in the first scenario. We briefly summarize the result here. DEED algorithm performs the best overall. When compared to Lloyd's method, it saves up to 54% traveling distance with an average saving of 28%. The energy saving is up to

46% with an average saving of 18%.

We also explore different initial deployment settings, where sensors start from four corners of the target area, e.g., in the 500m $\times$ 500m target area, initial locations of sensors are set to be in four 100m $\times$ 100m squares located in the four corners of the area respectively. The obtained results are similar and thus are omitted in the interest of space.

2.7 Summary

In this chapter, we studied the problem of energy-efficient even self-deployment in mobile sensor networks. In order to address the issue of energy-efficient deployment, which is still a challenge in the widely used Lloyd's method, a new algorithm, DEED algorithm, is proposed. Simulation results demonstrate that DEED performs well in different scenarios. Specifically, it leads to up to 54% less traveling distance and 46% less energy consumption than Lloyd's method.

Chapter 3

Energy-balanced Mobile Sensor Deployment to A Priori Known Target Locations

3.1 Introduction

How to deploy sensors to achieve certain coverage goals, be it area coverage, point coverage or barrier coverage, has been studied extensively in the literature. Typically, the problem setting is to determine where to place the sensors under the assumption that once the target locations are determined, sensors can be placed precisely at the designated target locations. This assumption, however, may not hold in practice. For instance, in hostile regions (e.g., remote harsh fields, disaster areas or toxic urban regions), it is often infeasible to manually place sensors at designated target locations. In such settings, sensors are often air dropped or projected into a region, with no control on precise positioning. To achieve the desired coverage goals, one strategy,

motivated by the availability of cheap mobile sensors (e.g., Robomote [67]), is to use mobile sensors. Once the mobile sensors land in the region of interest, they employ mobility to deploy themselves to the desired target locations. In this case, the problem is to determine, for a given set of desired target locations and a set of mobile sensors, which target location a mobile sensor should move to.

In the above scenario, while having only one sensor at each desired target location suffices to achieve the coverage goal, allowing multiple sensors to move to a desired target location provides additional benefits. This is because sensors are often battery powered and hence have limited lifetime. Furthermore, sensors are prone to failures, particularly in harsh environments. Therefore, allowing multiple sensors at a desired target location is helpful to improve fault tolerance as well as prolong the sensing time at the target location through role rotation [78]. The duration for which a target location can be covered is the sum of the remaining energy of the sensors at the target location (the remaining energy of a sensor is its original energy subtracted by the amount of energy consumed to reach the target location). A natural question in this setting is: for each sensor, which target location should it move to so that each target location is covered by at least one sensor, and the duration for which a target location can be covered is balanced? We henceforth refer to the problem as *energy-balanced mobile sensor deployment to a priori known target locations*, or DEPLOY problem for short. Since the starting locations of the sensors are not known beforehand (the sensors are typically projected or air dropped into an hostile area), the optimal solution cannot be obtained beforehand. Rather, it needs to be computed after sensors are deployed.

The above problem formulation differs from existing coverage problems (see Section 3.2 for more details). It can be used as a second step after solving a coverage

problem to achieve the solution through sensor mobility in an energy balanced manner. Surprisingly, this problem has not received much attention in the literature. In this chapter, we take the first step in developing centralized algorithms to solve the above problem. The computation can be done at one or multiple sensors, and the solutions will be broadcast to the other sensors.

As we shall see, the above problem is a special case of Max-min Fair Allocation (MMA) problem [6, 15, 29] (a known NP-hard problem, see Section 3.2). However, most existing approximation algorithms for MMA requires solving a relaxed linear programming formulation [6, 29], which are computational intensive, and hence are unsuitable for resource-constrained sensors. In this chapter, we propose light-weight algorithms that are not less computational extensive than linear programming. Specifically, we start with the simple scenario where sensors have the same starting location and the same initial energy, and propose an optimal algorithm to solve it. We then consider the generalized scenario, prove that the problem is NP-hard, and propose a light-weight heuristic algorithm to solve it. Extensive simulation results demonstrate that our heuristic algorithm achieves similar performance as the best known approximation algorithm while being much more computation-efficient. To the best of our knowledge, we are the first to formulate the above deployment problem, and propose computational efficient algorithms to solve it.

The rest of the chapter is organized as follows. Section 3.2 describes related work. Section 3.3 describes the problem formulation. Section 3.4 presents an optimal algorithm when sensors have the same starting location and energy. Section 3.5 presents algorithms for the general scenarios where sensors have different starting locations and energy. Last, Section 3.7 concludes the chapter and presents future work.

3.2 Related Work

We next briefly review several directions of research that are related to our study.

Coverage and mobile sensor deployment. Most existing studies on coverage problems in sensor networks is to determine the target locations of the sensors to achieve certain coverage goals. Broadly, coverage problems fall into two categories [45]: *static coverage*, where sensors are static, and *mobile coverage*, where sensors are mobile.

Depending on the coverage goals, static coverage can be classified into three classes: area coverage (e.g., [10, 37, 73, 79]), point coverage (e.g., [19, 75]), and barrier coverage [42, 63]. The goal of area coverage is to place the sensors to cover an entire area, the goal of point coverage (also referred to as target coverage) is to cover a set of points (or targets), while the goal of barrier coverage is to let the sensors form a barrier to prevent intruders from crossing the thin strip. Typically, deterministic solutions for the above three types of coverage problems is to determine the minimum number of sensors that are needed and the optimal target locations of the sensors so that the coverage goal is achieved, and in some cases additional conditions (e.g., connectivity or k-connectivity) are satisfied. The assumption is that once the target locations are determined, sensors can be placed precisely at the designated target locations to achieve the coverage goals. When this is not the case (e.g., in hostile conditions), several studies [16, 17, 25, 80] assume that a large number of sensors are randomly deployed in the region of interest, and study how to schedule the sensors so that the lifetime of the sensor network is maximized. In our study, motivated by the availability of cheap mobile sensors (e.g., Robomote [67]), we explore how to use sensor mobility to move the sensors to the desired target locations (which can be

predetermined by solving a coverage problem). Our solution can be used as a second step after solving any of the above static coverage problems when manual deployment of the sensors is not feasible.

In mobile coverage, the goal can be covering a region fully or sweep coverage [45]. In the first category, existing studies (e.g., [32,34,71,82]) move sensors iteratively and step by step to achieve certain formation or to improve the quality of the coverage. In these studies, the final target locations of the sensors are not known beforehand, rather will be determined in the process. Our study differs from them in that we assume the target locations (which are the desired target locations) are known beforehand. In sweep coverage [45], sensors only need to monitor certain points of interest periodically so that a small number of mobile sensors are needed to achieve sweep coverage among a much larger number of points of interest. In our study, we focus on determining which target location a sensor needs to move to, and sensors do not move afterwards.

Motion planning and navigation. Several studies in the robotics community are on robot path planning and navigation (e.g., [14,62,69]). The goal is to plan the path for a robot to move to a target location with a field of obstacles. Our study differs from them in scope, and can be combined with them to determine the paths for a mobile sensor after determining which target location a sensor should move to.

Max-min fair allocation. The DEPLOY deployment problem formulated in our study is a special case of the Max-min fair allocation (MMA) problem as discussed in [6,15,29]. An MMA problem considers m indivisible goods and n agents, where the agents' utilities for each good is given, and the goal is to fairly allocate goods to agents (i.e., the utility of the agent with the minimum utility is maximized). The study in [6] proposes an approximate algorithm for MMA with the best known approximation ra-

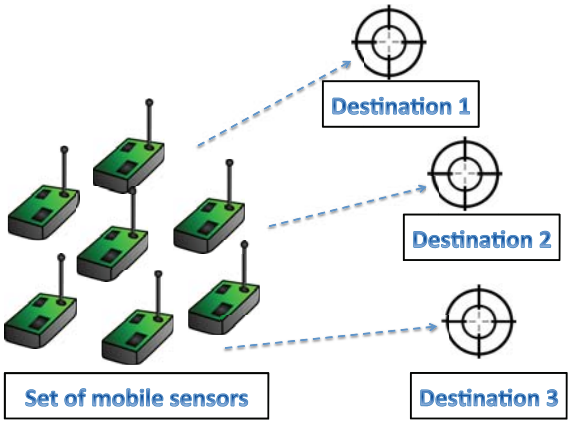


FIGURE 3.3.1: Illustration of the problem setting.

tio. The approximation ratio, $\Omega(1/(\sqrt{n}\log^3 n))$, is obtained by rounding the results of a relaxed linear programming formulation (LP-relax) [6]. The time complexity of the algorithm is lower bounded by the time complexity of solving the relaxed linear programming formulation, for which the best known complexity is $\Omega((nm)^{3.5})$ [40]. Another algorithm proposed in [15] is based on maximal matching, and has approximation ratio of $1/(m-n+1)$. Both algorithms are computational intensive and thus not applicable to resource constrained sensors.

3.3 Problem Formulation

Consider a set of sensors S and a set of a priori known target locations T. As each target location must be covered by at least one sensor, we have $|S| \geq |T|$. Let g_s and l_s denote respectively the initial energy and the starting location of sensor $s \in S$. Sensors consume energy to travel from their starting location to their selected target location. We assume that for a travel distance of d, the energy consumed is $U(d)$ where $U(d)$ is a monotonically increasing function of d. Let d_{st} denote the distance

TABLE 3.3.1: Key notation.

Notation	Definition
S	set of sensors
T	set of target locations
l_s	starting location of sensor s
g_s	initial energy of sensor s
r_{st}	remaining energy of sensor s after reaching target location t
e_t	total remaining sensor energy at target location t
$e_{\min}$	minimum total remaining energy over all target locations, i.e., the objective value

for sensor s to travel to target location t. Let r_{st} denote the remaining energy of sensor s if it travels to target location t. Then $r_{st} = g_s - U(d_{st})$. If a target location t is unreachable for sensor s, i.e., the energy depletes before sensor s reaches t, we let $r_{st} = 0$. If a sensor cannot reach any target location, it is excluded from the computation. Similarly, if a target location cannot be reached by any sensor, then we remove the target location from the computation. Henceforth, we assume $\forall s \in S$, there exists at least one target location, t, such that $r_{st} > 0$. Similarly, $\forall t \in T$, there exists at least one sensor, s, such that $r_{st} > 0$. An illustration of the problem is shown in Figure 3.3.1. The key notation is summarized in Table 3.3.1 for easy reference.

Our goal is to determine which target location a mobile sensor should move to so that each target location is covered by at least one sensor, and the duration for which a target location can be covered is balanced. Let x_{st} denote whether sensor s travels to target location t or not. Specifically,

$$x_{st} = \begin{cases} 1, & \text{if sensor } s \text{ travels to target location } t \\ 0, & \text{otherwise} \end{cases}$$

Then the problem is formulated as

$$\begin{aligned} \text{maximize:} \quad & \min_{t \in T} \sum_{s \in S} x_{st} r_{st} \\ \text{s.t.} \quad & \sum_{t \in T} x_{st} = 1, \forall s \in S \\ & x_{st} \in \{0, 1\} \end{aligned}$$

The *total remaining energy* at a target location is the sum of the remaining energies of the sensors that have moved to this target location, which is directly related to the duration for which the target location can be monitored. We denote the total remaining energy at target location t as e_t. Then, $e_t = \sum_{s \in S} x_{st} r_{st}$. In the above formulation, the objective function is to maximize the minimum remaining energy over all the target locations. The constraint specifies that each sensor travels to only one target location. Clearly, at least one sensor has to move to a target location, otherwise the objective value is zero, and clearly not optimal. When the number of sensors is the same as the number of target locations (i.e., $|S| = |T|$), the problem degenerates to the case where a target location is covered by exactly one sensor. Henceforth, we refer to the above problem as the DEPLOY problem. We refer to a solution to the DEPLOY problem as an *allocation*.

The initial energy of a sensor depends on its initial battery capacity. The initial location of a sensor depends on the initial deployment. The sensors may be dropped in packages; the ones that are dropped in the same package are placed closely to each other. Sensors in the same package can communicate with each other. In addition, at least one sensor in a package has localization capability; and the sensors that do not have localization capabilities approximate their locations through the locations of other nodes in the package (e.g., through range-free localization techniques [31, 46]).

In this chapter, we consider centralized solutions where the global knowledge of the network is known; distributed solutions are left as future work.

We consider two settings, *homogeneous* and *heterogeneous* settings. In both settings, we assume the target locations are known by all the sensors beforehand. In homogeneous setting, sensors have the same initial energy and initial location. This corresponds to a scenario where sensors that have the same initial battery capacity are projected or dropped as a whole package to the initial location. In this setting, only a single sensor needs to gather the global information, solve the DEPLOY problem, and informs the solution to other sensors. In heterogeneous setting, sensors may have different initial energy and/or different initial locations. When sensors have different initial locations (i.e., they are dropped in multiple packages), the sensors in the same package elect one leader (many leader election algorithms have been proposed in the literature, e.g., [24, 53, 70]), which gathers the information of other sensors in the package and communicate with the leaders in other packages. In this way, sensors in the network can communicate with each other. Similar as the homogeneous setting, only one sensor needs to gather the global information, solve the DEPLOY problem, and informs the solution to other sensors.

3.4 Homogeneous Setting

In the homogeneous setting, i.e., when all sensors have the same initial energy and initial location, we develop an algorithm to solve the DEPLOY problem and show that it obtains the optimal solution. The algorithm, as shown in Algorithm 2, is greedy in nature. It runs in steps. In each step, it finds the target location that

has the minimum total remaining energy, assigns a sensor to it, and updates the total remaining energy at this target location. The steps repeat until all sensors have been assigned. This algorithm is a more generalized version of the progressive filling algorithm that has been used to achieve max-min fairness [68]. The difference is that the utility of the sensors to the target locations depends on the target locations, and may not be the same. The complexity of the algorithm is $\mathrm{O}(|S||T|)$.

Algorithm 2 Greedy algorithm (homogeneous setting)

```
1: while sensor set S is not empty do
2:     find a target location t that currently has the lowest total remaining energy.
3:     assign a sensor s to t.
4:     remove sensor s from S.
5: end while
6: output obtained allocation.
```

We next present a theorem that states a sufficient and necessary condition for an allocation to be optimal in the homogeneous setting. We then show that Algorithm 2 is an optimal algorithm. In the homogeneous setting, the remaining energy of any two sensors moving to the same target location is the same. That is, for any two sensors s_i and s_j, $r_{s_i t} = r_{s_j t}$, $\forall t \in T$. We therefore use r_t to denote r_{st}.

Theorem 3.4.1. *In the homogeneous setting, an allocation is optimal if and only if it satisfies that*

$$e_t - e_{\min} \leq r_t, \quad \forall t \in T, \tag{3.4.1}$$

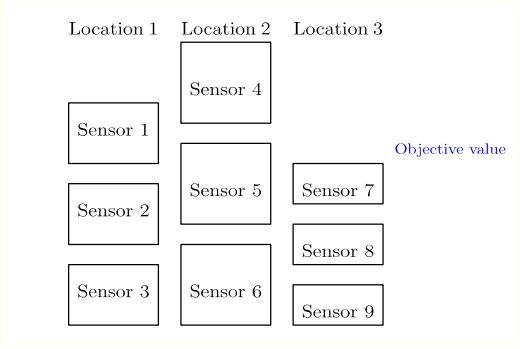
where $e_{\min} = \min_{t \in T} e_t$.

Proof. We first prove that if an allocation is optimal, then it satisfies inequality (3.4.1). The proof is by contradiction. Consider an optimal allocation where the target location that has the minimum total remaining energy is $t^*_{\min}$, and the total

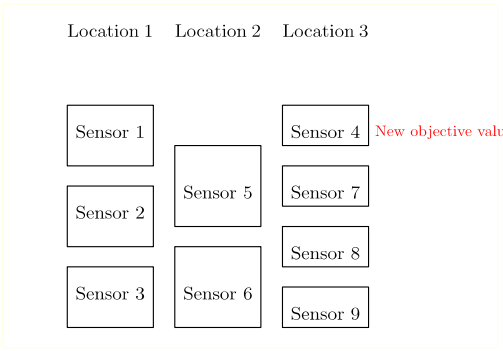
remaining energy at this location is $e^*_{\min}$. Suppose that inequality (3.4.1) is not satisfied under this allocation. That is, there exists at least one target location t that satisfies $e_t - e^*_{\min} > r_t$. Then, we can reassign a sensor from target location t to target location $t^*_{\min}$. In the new allocation, the total remaining energy at target location $t^*_{\min}$ is increased, while the total remaining energy at target location t becomes $e_t - r_t \geq e^*_{\min}$. Hence the new allocation achieves an objective value larger than $e^*_{\min}$, which contradicts with the assumption that the original allocation is optimal.

We then prove that if an allocation satisfies inequality (3.4.1), then it is optimal. The proof is again by contradiction. Consider an allocation, X. Suppose that it satisfies inequality (3.4.1), but is not optimal. Suppose the objective value of this allocation is $e_{\min}$ and the target location that has the minimum remaining energy is $t_{\min}$. Let $e^*_{\min}$ denote the objective value of an optimal solution, X^*. Then $e^*_{\min} > e_{\min}$. Therefore target location $t_{\min}$ must have obtained more sensors in allocation X^* than what it gets in allocation X. This implies that, in X^*, there exists (at least) one target location other than $t_{\min}$, denoted as t, that gets at least one sensor less than what it gets in X. Since X satisfies inequality (3.4.1), we have $e_t - e_{\min} \leq r_t$, $\forall t \in T$, in allocation X. Let e^*_t denote the remaining energy of target location t in allocation X^*. Then $e^*_{\min} \leq e^*_t \leq e_t - r_t \leq e_{\min}$, contradicting with the assumption that X^* is an optimal solution. □

An example of inequality (3.4.1) is shown in Figure 3.4.1. In this example, we use a bar to represent the remaining energy of a sensor. The total remaining energy at a target location can then be represented as stacked bars. Fig. 3.4.1(a) shows an allocation where inequality (3.4.1) is not satisfied. If we reassign sensor 4 from target location 2 to target location 3, we can obtain a better allocation as shown in



(a) Allocation not satisfying inequality (3.4.1)



(b) Allocation satisfying inequality (3.4.1)

FIGURE 3.4.1: The bars are used to represent remaining energy of sensors. Give an allocation does not satisfying inequality (3.4.1), it is not optimal and there exists a allocation satisfying inequality (3.4.1) which is optimal.

Fig. 3.4.1(b) which satisfies inequality (3.4.1) and is optimal.

It is easy to see that the allocation obtained by Algorithm 2 satisfies inequality (3.4.1). From Theorem 3.4.1, we have the following corollary.

Corollary 3.4.1. *The allocation obtained by Algorithm 2 is optimal.*

We next show that Algorithm 2 also provides an optimal solution for *incremental deployment* in the homogeneous setting, i.e., when some sensors have already been deployed at the target locations, and a new set of homogeneous sensors (i.e., with the same initial energy and initial location) need to be deployed.

Corollary 3.4.2. *Algorithm 2 provides an optimal solution for incremental deployment in the homogeneous setting.*

The proof follows directly that from Theorem 3.4.1 since the allocation of Algorithm 2 satisfies inequality (3.4.1), and hence is optimal for incremental deployment in the homogeneous setting.

In addition to incremental deployment, the above corollary serves as the basis to solve the *limited heterogeneous* setting where most of the sensors have the same initial energy and initial location; only a small number of sensors are heterogeneous (i.e., have different initial energy and/or initial location). Specifically, suppose that there are k heterogeneous sensors. If we know which target locations the k heterogeneous sensors travel to, then by running Algorithm 2, we can obtain the optimal allocation according to Corollary 3.4.2. Thus in order to obtain a global optimal allocation, we can enumerate all possible assignment of the k sensors and then run Algorithm 2. The allocation with highest objective value is the optimal allocation for the limited heterogeneous setting. Since there are $|T|^k$ different assignments of k heterogeneous sensors, the algorithm runs Algorithm 2 $|T|^k$ times on all regular sensors for each possible assignments of k heterogeneous sensors. The complexity of the algorithm is $\mathrm{O}\left(|S||T|^{k+1}\right)$.

3.5 Heterogeneous Setting

In this section, we first prove that the DEPLOY problem is NP-hard in general. We then present a light-weight heuristic algorithm to solve the general DEPLOY problem. At the end, we propose two enhancing techniques that can be applied to the heuristic algorithm.

3.5.1 Hardness of the Deploy Problem

Theorem 3.5.1. *The* DEPLOY *problem is NP-hard in general. It remains NP-hard when sensors have the same initial location but different initial energies or when*

sensors have the same initial energy but different initial locations.

Proof. We prove the lemma by reducing a known NP-hard problem "Big goods/Small goods MMA" [29] (BS-MMA) problem, to DEPLOY problem. In BS-MMA problem, we considers m indivisible goods and n agents, where the agents' utilities for each good is given and can be partitioned into two sets $\mathcal{G}_B$ and $\mathcal{G}_S$. Let $u(a, g)$ denote the utility of agent a for good g. The utilities in such problem satisfy that for some $x > 1$, $u(a, g) = 1$ if $g \in \mathcal{G}_S$ and $u(a, g) = x$ if $g \in \mathcal{G}$. The goal is to fairly allocate goods to agents (i.e., the utility of the agent with the minimum utility is maximized).

We denote the DEPLOY problem where sensors have the same starting location but different initial energies as DEPLOY $- \alpha$ problem. We denote the DEPLOY problem where the sensors start from different locations but have the same initial energy as DEPLOY $- \beta$ problem.

Suppose we have an optimal algorithm A_α to DEPLOY$-\alpha$ problem and an optimal algorithm A_β to DEPLOY $- \beta$ problem. Our reduction is by showing that A_α and A_β are both optimal algorithms to BS-MMA problem.

Consider a BS-MMA problem with m indivisible goods and n agents. We can design a DEPLOY $- \alpha$ problem in following way. We put n agent randomly at 2 random locations. We put all m goods at a single start location having the same distance d from both agent locations. If a good $g \in \mathcal{G}_B$, its initial energy is set to $x + U(d)$ and its remaining energy at any agent is x. If a good $g \in \mathcal{G}_S$, its initial energy is set to $1 + U(d)$ and its remaining energy at any agent is 1. It's easy to show that the resultant problem is a DEPLOY $- \alpha$ problem and an optimal allocation to the problem corresponds to an optimal allocation to the BS-MMA problem. Thus, A_α is optimal algorithm to the BS-MMA problem.

Similarly, consider another BS-MMA problem with m indivisible goods and n agents. We can design a DEPLOY $-\beta$ problem in following way. We put n agents randomly at 2 random locations. We put m goods at two start locations. Suppose all goods have same initial energy g. One of the start location l_1 have the same distance d_1 to two agent locations satisfying that $g - U(d_1) = x$. The other start location l_2 have the same distance d_2 to two agent locations satisfying that $g - U(d_2) = 1$. If good $g \in \mathcal{G}_B$, we put it at l_1 such that its remaining energy at any agent location is x. If a good $g \in \mathcal{G}_S$, we put it at l_2 such that its remaining energy at any agent location is 1. It's easy to show that the resultant problem is a DEPLOY $-\beta$ problem and an optimal allocation to the problem corresponds to an optimal allocation to the BS-MMA problem. Thus, A_β is optimal algorithm to the BS-MMA problem.

□

3.5.2 Heuristic Algorithm

We propose a heuristic algorithm for DEPLOY problem. This algorithm applies to both heterogeneous and homogeneous settings, and it degenerates to Algorithm 2 in homogeneous settings. Specifically, as summarized in Algorithm 3), this algorithm is also greedy in nature and runs in steps. In each step, it finds the target location that has the minimum total remaining energy, and allocates a sensor from the remaining set of sensors to it. The choice of the sensor is also greedy in that the algorithm gives higher priority to a sensor that is closer to the target location and has higher initial energy. Specifically, it searches sensor initial locations in the order of increasing distance to the target location, find the first initial location that has at least one remaining sensor and chooses the sensor that has the highest initial energy. The

complexity of the algorithm is $\mathrm{O}\left(|S||T|^2\right)$.

Algorithm 3 Heuristic greedy algorithm (general setting.)

1: **while** sensor set S is not empty **do**
2: find a target location t that currently has the lowest remaining energy.
3: find a sensor s, closer starting location first, sensor with higher initial energy first and assign s to t
4: remove sensor s from sensor set S
5: **end while**
6: output the obtained allocation.

We next show that Algorithm 3 satisfies two properties. Before that, we first show that any optimal allocation provides a *steady allocation*, as defined below.

Definition 3.5.1. (Steady allocation) *Let $e_{\min}$ denote the minimum total remaining energy over all target locations, i.e., the objective value of an allocation. An allocation is steady if it satisfies*

$$e_t - e_{\min} \leq \min_{s \in S_t} r_{st}, \quad \forall t \in T, \tag{3.5.1}$$

where S_t is the set of sensors that move to target location t.

Lemma 3.5.1. *An optimal allocation is a steady allocation.*

Proof. The proof is similar to the proof for Theorem 3.4.1. The proof is by contradiction. Suppose an allocation is optimal, with minimum total remaining energy $e^*_{\min}$, and the target location $t^*_{\min}$ with minimum total remaining energy, but is not a steady allocation. Then there exists at least one target location t and a sensor s at t that satisfies $e_t - e^*_{\min} > r_{st}$. In such case, we can remove s from target location t and assign it to target location $t^*_{\min}$. Then in the new allocation, for target location $t^*_{\min}$, its remaining energy is increased, while for target location t, its remaining energy

becomes $e_t - r_{st} \geq e^*_{\min}$. Hence the new assignment achieves an objective value larger than $e^*_{\min}$, contradicting with the assumption that $e^*_{\min}$ is optimal. □

Proposition 3.5.1. *When all the sensors have the same initial energy, Algorithm 3 outputs a steady allocation.*

Proof. When all the sensors have the same initial energy, the closer the sensor to the target location t, the higher the remaining energy of the sensor will have at t. Since in Algorithm 3 target location t always finds a sensors from a closer starting location, target location t obtains sensors with their remaining energies at t monotonically decreasing.

Suppose that Algorithm 3 outputs an allocation X which is not steady, it implies at least one target location t violates inequality 3.5.1. We denote in this allocation the target location with the minimum total remaining energy $e_{\min}$ as $t_{\min}$. Consider the state right before t obtains its last sensor x. In this state, t is the target location with the minimum total remaining energy. It implies that in this state $e_t \leq e_{\min}$ or $e_{\min}$ will increase in the final state. In the final state, the sensor x has been allocated to t and the total remaining energy at t is $e_t + r_{xt}$. Since t violates inequality 3.5.1 and $r_{xt} = \min_{s \in S_t} r_{st}$ as we show previously, we have $e_t + r_{xt} - e_{\min} > r_{xt}$ and thus $e_t > e_{\min}$ which contradicts to the fact that $e_t \leq e_{\min}$. □

We next show that Algorithm 3 outputs an optimal allocation in certain cases.

Proposition 3.5.2. *When all the sensors have the same initial energy, if an allocation obtained by Algorithm 3 satisfies that, for every target location t, the sensor assigned to t is from the initial location that is closest to t, then this allocation is optimal.*

To prove Proposition 3.5.2, we propose a lemma first.

Let l_t^* denote the closest starting location of location t. We use r_t^* to denote the remaining energy of sensor at t from l_t^*. We use $e_{\min}^X$ to denote the objective value of allocation X.

Lemma 3.5.2. *Consider a location t in two allocations X and Y with total remaining energy e_t^X and e_t^Y. In X, k^X sensors at location t are from t's closest starting locations. In Y, k^Y sensors at location t are from arbitrary locations. We have (i) If $e_t^X < e_t^Y$, then $k^X < k^Y$. (ii) If $k^X > k^Y$, then $e_t^X - e_t^Y \geq r_t^*$*

Proof. Firstly we show statement i is correct. Suppose $e_t^X < e_t^Y$ and $k^X \geq k^Y$. Then in Y there is at least one sensor s satisfies that $r_{st} > r_t^*$, which contradicts to the fact that l_t^* is the closest starting location of t.

The correctness of Item ii can be shown by the fact that $e_t^Y \leq k^Y r_t^*$. Then we have $e_t^X = k^X r_t^* - e_t^Y \geq (k^X - k^Y) r_t^* \geq r_t^*$ □

We now prove Proposition 3.5.2.

Proof. Suppose allocation X with objective value $e_{\min}$ is not optimal, then there exists an optimal allocation X^* with objective value $e_{\min}^* > e_{\min}$. It means that the t_{min} in X must get at least one more sensor in X^* according to Lemma 3.5.2 statement i. Since the number of sensors is same in X and X^*, this implies, among the locations other than $t_{\min}$ in X, there is at least one location t which get at least one sensor less in X^*. Let e_t and e_t^* denote the total remaining energy of t in X and X^* respectively. We have $e_t - e_t^* \geq r_t^*$, i.e., $e_t^* \leq e_t - r_t^*$ according to Lemma 3.5.2 statement ii. Since the X is a steady allocation following Proposition 3.5.1, we have $e_t - r_t^* \leq e_{\min}$ based on equation 3.5.1. Then we have $e_t^* \leq e_t - r_t^* \leq e_{\min}$, which means objective

value of X is not smaller than the total remaining energy at location t in X^*. Thus, the objective value of X^* cannot be larger than the objective value of X, which is a contradiction to the assumption that $e^*_{\min} > e_{\min}$. □

3.5.3 Enhancing Algorithms

We next propose two enhancing algorithms that can improve the allocations obtained from Algorithm 3. By the definition of steady allocation (Definition 3.5.1), we see that if an allocation is not steady, we can always find a sensor that can be reassigned to increase the objective value. The first enhancing algorithm is summarized in Algorithm 4. It repeatedly finds and reassigns sensors as long as the current allocation is not steady.

Algorithm 4 Enhancing algorithm – Reassign sensors

```
Input: an allocation
while the allocation is not a steady allocation do
    locate the target location, t_min, that has the minimum remaining energy
    locate a target location, t, that does not satisfy inequality (3.5.1).
    reassign the sensor that has the minimum remaining energy at target location
t to t_min
    update the allocation
end while
output the obtained allocation
```

Let $e_{\min}$ and $e'_{\min}$ denote respectively the objective value before and after running Algorithm 4. Then it is clear that $e'_{\min} \geq e_{\min}$ (the equality only holds when the original solution is already stable). Specifically, suppose that the target location, $t_{\min}$, has the minimum remaining energy, then after reassigning a sensor s from target location t to $t_{\min}$, the total energy at $t_{\min}$ will be increased by $r_{st_{\min}} \geq r_{\min}$. Algorithm 4 will terminate when the current solution is steady. Before that, each round

of finding potential sensor and reassigning it will improve the objective value by at least $r_{\min} = \min_{\forall s,t} r_{st}$. Since the total amount of improvement is bounded (e.g., it is no more than $\sum_{s \in S} g_s$, where g_s is the initial energy of sensor s), Algorithm 4 will terminate in a finite number of rounds.

For an allocation obtained by Algorithm 3, it is possible that there are two sensors, sensor s_1 allocated to target t_1 and sensor s_2 to target t_2, satisfying that swapping the assignments of s_1 and s_2 may increase both e_{t_1} and e_{t_2}. The second enhancing algorithm (summarized in Algorithm 5) involves swapping two sensors that are assigned to two different target locations. Since the allocation after such a swap may not be steady. In such cases, we further run Algorithm 3 until the resultant allocation is steady. The above procedure repeats until no two sensors need to be swapped. Since each such procedure improves the objective value by at least a positive amount and the total amount of improvement is bounded, it is clear that Algorithm 5 will after a finite number of rounds.

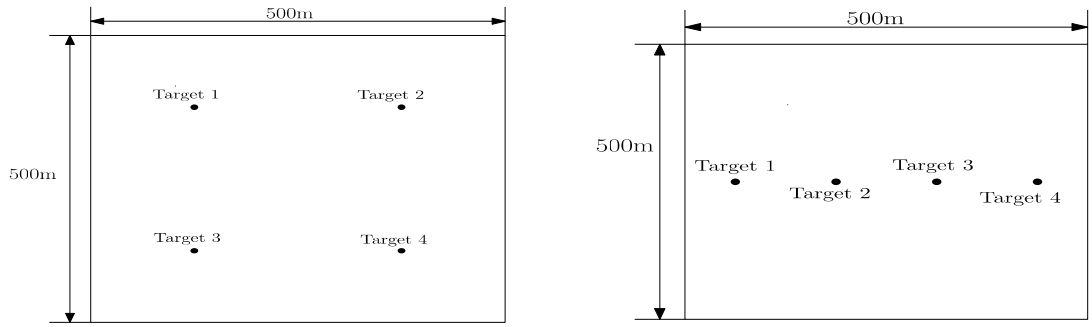
Algorithm 5 Enhancing algorithm – Swap sensors

Input: an allocation
while there are sensors s_1 on t_1 and s_2 on t_2 satisfying that swapping s_1 with s_2 increases both e_{t_1} and e_{t_2} **do**
    swap s_1 with s_2, i.e., reassign s_1 to t_2 and reassign s_2 to t_1.
    if the allocation is not a steady allocation **then**
        run Algorithm 4 to update allocation.
    end if
end while
output the obtained allocation

3.6 Performance Evaluation

We evaluate the performance of our proposed algorithms using extensive simulation. All simulations are done in our self-implemented simulator written in C++. Specifically, we consider deploying mobile sensors in a 500m $\times$ 500m square region. There are two types of sensors. The first type of sensor models the energy setting of regular sensor similar to that in Robomote [67]. That is, the initial energy is 26487J (equivalent to the energy of three 1.5V AAA alkaline batteries (1635mAh). We use b_l to denote the number of batteries on these regular sensors, i.e., $b_l = 3$. We also modeled a second type of sensor that has higher initial energy: the initial energy is equivalent to the energy of b_h batteries, $b_h \geq 3$. We use n_l to denote the number of regular sensors. The number of sensors with higher energy is denoted as n_h. We assume energy consumption is a linear function of the traveling distance. The energy consumption for traveling one meter is 28J as computed from Robomote. Thus, the maximum traveling distance for the first type of sensors is around 946m, sufficient for the deployment in the 500m $\times$ 500m area. We assume the sensors start from l locations, with equal number of sensors at each location.

We consider two layouts of target location, each with 4 target locations. In the first layout, the target locations are evenly placed in the square region, at target locations $(125, 375)$, $(125, 125)$, $(375, 125)$, and $(375, 375)$. In the second layout, the target locations are linearly placed, at $(100, 250)$, $(200, 250)$, $(300, 250)$, and $(400, 250)$. The two layouts are shown in Fig. 3.6.1.



(a) Even layout. (b) Linear layout.

FIGURE 3.6.1: Target location layouts.

3.6.1 Homogeneous Settings

In this section, we use the optimal deployment algorithm, Algorithm 2, in the simulation. We consider even layout of target locations as in Figure 3.6.1(a). In each simulation run, homogeneous sensors start from the same location and have the same initial energy. We compare the objective values of different initial energy settings, i.e., different b_l. To make a fair comparison, we set the total number of batteries to be 324 in all simulation runs. For b_l ranging from $[3, 6, 9]$, there are 108, 54, 36 sensors respectively.

We also tested the objective values using different starting locations for the sensors. We limit the starting location of the sensors to be *within* one of 5 different starting areas as shown in Figure 3.6.2. The distance to the center of the region decreases from the Area 1 to Area 5. In each simulation run in one of the area, we randomly generate starting locations sensors. The objective value of an area is averaged over 100 runs in the area.

Figure 3.6.3 plots the average optimal objective value result under different starting areas of the sensors. We can see that the average objective value increases as the

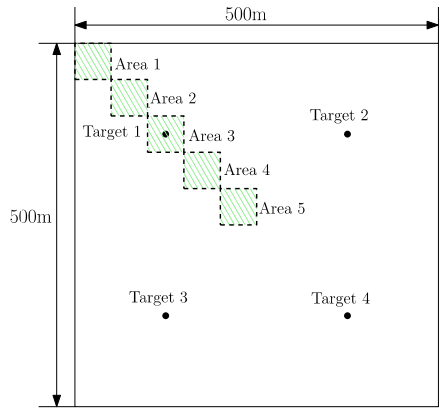


FIGURE 3.6.2: Different starting areas of homogeneous sensors for even layout of target locations.

starting area of the sensors become closer to the center of the area. This coincides with our expectation as in even layout of target locations, the closer to the center, the smaller the average travelling distance for sensors will be. Compared with Area 1, the average objective value for Area 5 increased up to 19%. We can also observe that higher initial energy leads to better objective values. This is because the sensor with higher initial energy carries the same number of batteries as multiple sensors with lower initial energy, but it only move once and thus accounts for only one time movement costs. Another observation is that the difference of objective values among different starting points diminishes as the sensors starting closer to the center of the area. At Area 5, the difference in object values under different starting points is quite small. This is because the moving cost decreases as the sensors starting closer to the center of the area. The saving of the moving cost of the sensors with higher energies diminishes.

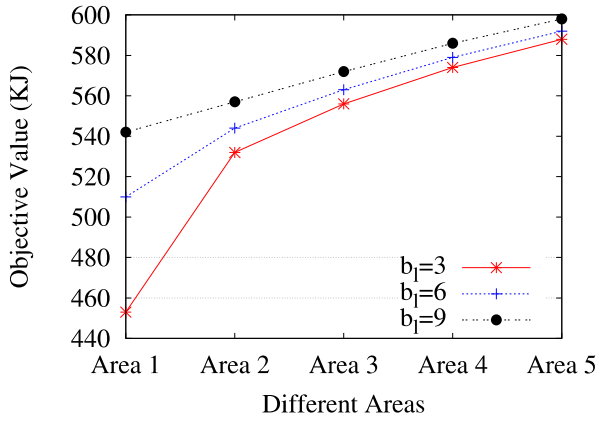


FIGURE 3.6.3: Objective values under different number of initial batteries and different starting areas.

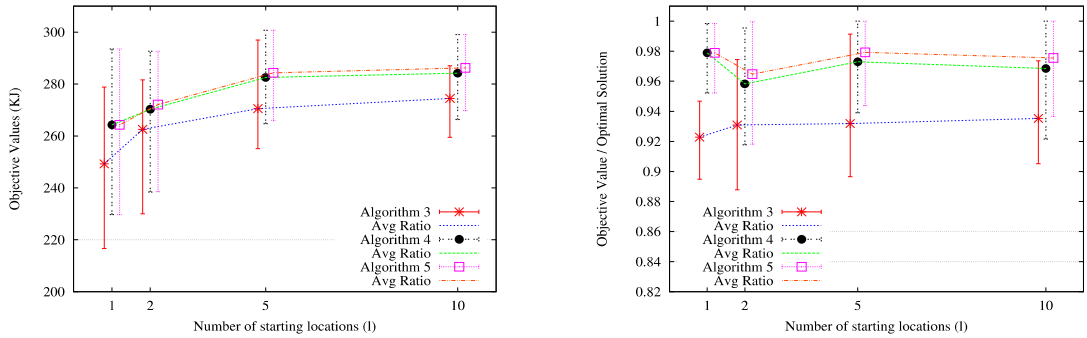
3.6.2 Heterogeneous Settings

Small-scale Settings

In this section, we use the deployment algorithm for the general settings, Algorithm 3, in the simulation. We also evaluate the performance of enhancing algorithm 4 and enhancing algorithm 5. We consider totally 40 sensors starting from l starting locations, with $n_l = 10$ and $n_h = 30$. There are $40/l$ regular sensors per location and l ranges in $[1, 2, 5, 10]$. At each location, there are $10/l$ sensors have higher initial energy and $b_h = 6$.

The objective values obtained by the algorithms will be compared with the optimal solution, which is obtained by solving integer linear programming formulation directly using IBM ILOG CPLEX [1]. For each l, we run Algorithm 3, Algorithm 4, Algorithm 5 and IBM ILOG CPLEX 100 times with random distribution of starting locations.

We present the objective values of Algorithm 3, Algorithm 4 and Algorithm 5 as *normalized objective value*, defined as the ratio of the objective value from our



(a) Objective values.

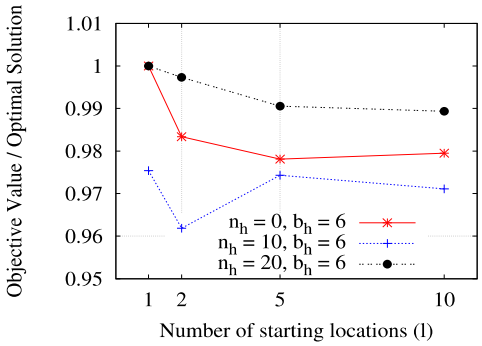
(b) Normalized objective values.

FIGURE 3.6.4: Objective values and normalized objective values of Algorithms 3, 4 5 for even layout of target locations ($b_h = 6$, $n_l = 30$ and $n_h = 10$).

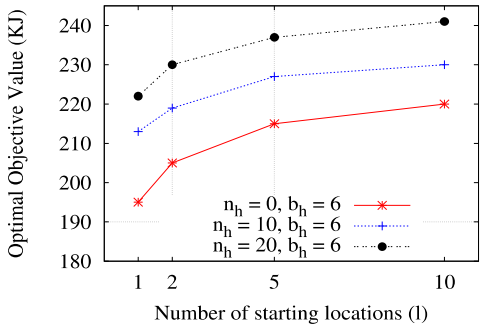
algorithms over the optimal objective value obtained using CPLEX. At each run, the normalized objective values are computed for all three algorithms. The minimum, maximum and the average of normalized objective values over 100 runs are collected for each l.

We consider relatively small number of sensors (i.e., 40 sensors) because with larger number of sensors (e.g., 100 sensors), some instances of the ILP formulation may take days to complete on our testing platform, which makes it impossible for us to collect sufficient results within reasonable time. We will consider larger scale simulations with more sensors in later section, in which we will compare Algorithm 3 with another known approximation algorithm.

Figure 3.6.4 plots the normalized objective value ranges of Algorithms 3, 4, 5. We use *even layout* of target locations as in Figure 3.6.1(a) with $b_h = 6$, $n_l = 30$ and $n_h = 10$. Figure 3.6.4(a) and Figure 3.6.4(b) plot objective values and normalized objective values respectively. In all of the simulation runs, the normalized objective values are close to optimal. The normalized objective values are above 0.88 in all runs for Algorithms 3. Both enhancing algorithm improves the simulation results. The



(a) Normalized objective values obtained by Algorithms 5



(b) Optimal objective values

FIGURE 3.6.5: Normalized objective values obtained by Algorithms 5 and optimal objective values under different number of starting locations (l) and different number of sensors with higher initial energy (n_h), $b_h = 6$

enhancing algorithm 4 improves the average to above 0.95. The enhancing algorithm 5 improves the average to above 0.96. The normalized objective value is best when there is only one starting location, which is expected since the Algorithm 3 produce optimal results when sensors are homogeneous. The result shows that the algorithms still performs well in heterogeneous case.

We also evaluate the performance of Algorithms 5 under different number of sensors with higher energy, i.e., different n_h. To make a fair comparison, we fixed the total number of batteries to be 120. We set $b_l = 3$ and $b_h = 6$. The number of regular sensor, i.e., the sensor with lower energy, can be calculated by $(120 - n_h * 6)/3$. Both regular sensor and the sensor with higher initial energy are evenly distributed at l starting locations. Similar as in previous setting, we collect the normalized objective value averaged over 100 random simulation runs for each data point.

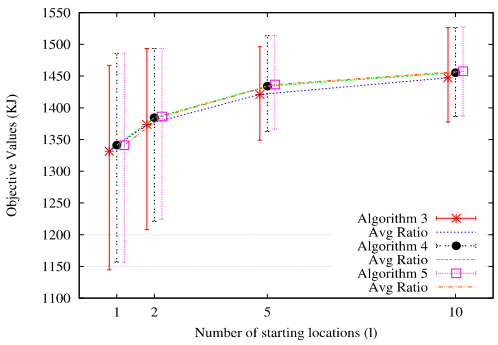
Figure 3.6.5 plots normalized objective values obtained by Algorithms 5 and optimal objective values under different number of starting locations (l) and different number of sensors with higher initial energy (n_h). The n_h varies from 0 to 20. Fig-

ure 3.6.5(a) plots the normalized objective values obtained by Algorithms 5. The average normalized objective values are all above 0.96. The average normalized objective value is higher if all the sensors are same type, i.e., when $n_l = 0$ or $n_h = 0$. The normalized objective values are very close to optimal in all settings. An interesting observation is that the optimal objective values increase if there are larger number of sensors with higher energy or there are larger number of starting locations. We plot the collected optimal objective values from CPLEX in Figurer 3.6.5(b) We can see that the objective value increases as l increases, which shows that the randomly generated starting location helps with deployment when target locations are evenly distributed. When n_h increase, the objective value increases as the energy cost in moving batteries to target locations decreases. The difference is not large. Using only sensors with higher initial energy, the objective values increase between $8\% - 13\%$ compared with using regular sensors only. Also, since mobile sensors are prone to fail, only using a few sensors which carry large battery to save overall travelling cost may fail to achieve deployment goal. It's the trade-off one should consider in the deployment.

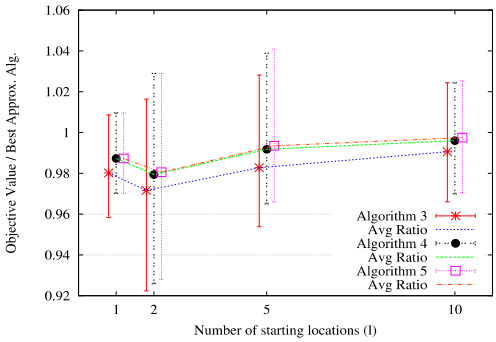
We also evaluate the performance of the algorithms by using similar settings in *linear layouts* of target locations. The results are similar to the even layout and we briefly summarize the result. The normalized objective value after enhancing algorithms are all above 0.92.

Large-scale Settings

In this section, we evaluate the Algorithm 3, Algorithm 4 and Algorithm 5 in large scale scenario. In such scenario, solving integer linear programming formulation is



(a) Objective values.



(b) Normalized objective values over the best approximation algorithm.

FIGURE 3.6.6: Objective values of Algorithm 3, 4 and 5 and normalized objective values over the best approximation algorithm under different number of starting locations (l).

not practical as some instances take forever to finish on our testing platform. Thus, we implement the approximation algorithm with best known approximation ratio as described in [6]. The algorithm requires to solve relaxed linear programming formulation first and we use IBM ILOG CPLEX [1] to solve it. We denote the algorithm as the *best approximation algorithm*.

We still consider even layout of target locations. We increase the number of sensors to 200. There are l starting locations, with $200/l$ sensors per location and l ranges in $[1, 2, 5, 10]$. At each location, there are $50/l$ sensors have higher initial energy and $b_h = 6$. We normalize objective values of Algorithm 3, Algorithm 4 and Algorithm 5 over the objective values of the best approximation algorithm.

We collect and compare normalized objective values of three algorithms. We also collect results of running time ratio of the algorithms over the best approximation algorithm, which shows how much times the best approximation algorithm is slower than our algorithms. All data points are averaged from 100 random simulation runs.

Figure 3.6.6 plots the performance comparison of objecive values among Algorithm

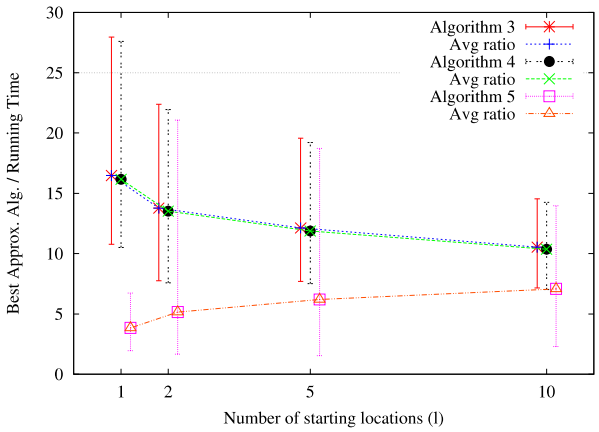


FIGURE 3.6.7: Running time ratio under different number of starting locations (l).

3, 4, 5 and the best approximation algorithm. Figure 3.6.6(a) and Figure 3.6.6(b) plot objective values and normalized objective values respectively. We can see from Figure 3.6.6(b) that all algorithms perform on average quite similar as the best approximation algorithm. The objective values of the best approximation algorithm are around %1 better on average.

Figure 3.6.7 plots the performance comparison of running time among Algorithm 3, 4, 5 and the best approximation algorithm. We can see from Figure 3.6.7 that the best approximation algorithm runs much slower than our algorithms. Under our simulation setting, it takes up to 27 times more time in solving an instance than the Algorithm 3. This is because that the approximation algorithm requires to solve relaxed linear programming formulation, which itself runs much slower than the Algorithm 3. Due to the big difference in computation complexity. It is expected the difference will be even larger in larger scale settings. We also notice that the Algorithm 5 takes much more extra time than Algorithm 4. Considering both the objective value performance and the running time, Algorithm 4 should be a good choice in most scenarios.

3.7 Summary

In this chapter, we studied the problem that determines, for a given set of a priori known target locations and a set of mobile sensors, which target location a mobile sensor should move to so that each target location is covered by at least one sensor, and the duration for which a target location can be covered is balanced. We proposed a set of light-weight deployment algorithms that are suitable for resource-limited sensors.

Specifically, we start with the simple scenarios where all sensors have the same starting location and initial energy or only a small number of sensors have different starting location and initial energy, and propose optimal algorithms to solve them. We then consider the general scenario, prove that the problem is NP-hard and propose light-weight heuristic algorithms for the scenario. Extensive simulation results demonstrate that our proposed algorithms either achieve close-to-optimal performance or achieve similar performance as the best known approximation algorithm while being much more time-efficient. As future work, we will explore centralized algorithms that do not depend on the locations of the sensors. We will also explore distributed and randomized algorithms.

Chapter 4

Cost-efficient Target Monitoring and Data Offloading through Mobile Phone Sensing

4.1 Introduction

With the increasing prevalence of smart phones, mobile sensor networks formed by mobile phones become a great alternative to traditional mobile sensor networks. The richly equipped sensors can enable sensing applications in various domains such as environmental monitoring, social network, health-care, transportation, safety, etc [41, 44]. For example, The authors of [52] have proposed a system called SoundSense, which collects sound events through mobile phone's microphone. It uses various learning techniques to classify the sound into different categories to recognize sound events in users' daily lives. The authors of [11] have built MoVi, which improves social event coverage by capturing video highlights of users' events. It detects the

event by correlating the sensed data with the data from other phones in the same social group. Once an event is confirmed, the phone with best view will be recruited for taking the video of the event.

An important assumption in these delay-sensitive mobile phone applications is that mobile phones can offload the data right at the sensing site, via either an available WiFi access point or the cellular network. However, this is not always true. The offloading capability of mobile phones is quite limited. Although WiFi network is a fast and preferred way to offload the data, a free and unencrypted WiFi access point is not easy to locate, even in the urban area. The authors of [60] have designed an algorithm for delay-tolerant applications to store the data first and then offload the data when users are close to an available WiFi access points. As for the delay-sensitive applications, mobile phones may have to resort to the widely-available cellular network to offload the data. The celluar networks, mostly referred to as 3G/4G networks, has its own limitations. The offloading capability of a mobile phone on 3G/4G networks is limited by its available data plans on the corresponding phone carrier. Since unlimited data plans are no longer prevalent [48], often times the users will not be able to offload the sensed data without limitation. Even the users have large available data plans, it is reasonable to assume that they will not contribute all of their data plans in sensing tasks.

In this chapter, we design a generic delay-sensitive system which employ mobile phones to monitor the status of a set of targets. The status information of the targets are sensed by the mobile phone and uploaded to a set of cloud servers for processing. We assume mobile phones offload the data via the cellular networks and they have limited 3G/4G data plans available. When the sensing task incurs large amount of data and a mobile phone is unable to offload the data on its own, we propose the

mobile phones to rely on opportunistic data exchange, such as Bluetooth and WiFi ad hoc communication, to collaboratively offload the data. As all activities of mobile phones in sensing tasks may incur certain cost, we also consider the cost during the whole sensing process. The costs in the system include cost of data sensing, data exchange and data offloading, which are all considered generic. For example, the costs may be energy consumptions of phone activities, which directly impacts the battery life of mobile phones. They may vary among different phones as the energy consumptions of phone activities depends on the type of mobile phones and the hardwares it is built with. The costs may also be the financial compensations that the system has to pay to the participants on their phone activities. They may also vary among different phones as different users may ask for different prices for participation. The costs of the sensing tasks should be carefully managed or the costs may become unacceptable to either system or user. User may not expect the sensing task depletes its battery life and the system may not spend too much on a single sensing task.

We study how to cost effectively schedule mobile phones to monitor a set of targets and upload the collected data. The goal of the problem is to obtain a cost-effective schedule of phone activities. We start with the offline problem, assuming the trajectories of the phones are known beforehand. We propose to overcome the limited data plan by resorting to opportunistic communication among mobile phones. We then formulate the problem as a minimum cost flow problem and obtain optimal solutions. We also consider cost fairness among phones by proposing algorithms to balance the costs among the phones. We then investigate the online version of the problems under realistic assumptions where only the past trajectories of the phones are known. We develop two heuristic algorithms one aims to minimize cost while

the other aims to achieve cost fairness. Extensive simulation results show that the proposed algorithms perform well in terms of both minimizing total cost and achieving cost fairness.

Our main contributions are:

- We propose a generic target monitoring system through mobile phones. The system considers practical limitations of mobile phones such as limited 3G/4G data plan and generic costs of phone activities.
- We propose optimal algorithms with idealized assumptions for two problems: minimizing total cost and achieving cost fairness, respectively. They serve as benchmark algorithms in performance evaluations.
- We propose heuristic algorithms with practical assumptions for both problems, which renders these algorithms immediately applicable in practice.

The rest of the chapter is organized as follows. We describe the state of the art related works in Section 4.2. We present the problem formulation and the main notations used through the chapter in Section 4.3. We present optimal algorithms in Section 4.4. After that, we present heuristic algorithms in Section 4.5. We describe our simulation settings and results in Section 4.6 and conclude the chapter in Section 4.7.

4.2 Related Works

Mobile phone have been the focus of recent attention in sensing and mobile computing applications. Besides applications described in Section 4.1, The authors of [57] have

proposed METIS platform. It implements a sensing task distribution scheme that dynamically decides whether to perform sensing on the phone or in the infrastructure, considering the energy consumption, accuracy, and mobility patterns of the user. The authors of [74] have proposed StreamShaper. It employs mobile phones to collect the environmental data, which provides a virtual sensor abstraction to applications. They also proposed coordinating sensing algorithms to achieve quality requirements and energy efficiency. The authors of [23] have designed a citizen journalist application based on the proposed Platform for Remote Sensing using Smartphones (PRISM). It provides location-based triggers to alert mobile phone users, who are in the vicinity of a location of interest, to respond to the application so that the sensing data at the location can be collected.

Target monitoring is also known as point coverage problems in sensor networks. Different from other two types of coverage problems, area coverage and barrier coverage, its objective is to cover a set of points [18]. Previous works on point coverage problems mainly focus on deploying static or mobile sensors to achieve targets coverage goals [16, 72]. In this chapter, we leverage the mobile phone sensing to monitor the targets, which incurs minimal deployment costs. The coverage goal and mathematical model of the system is completely different from closely related work such as that in [65], which has developed a energy-efficient collaborative sensing model which schedules mobile phones to achieve area coverage.

In this work, we also consider generic costs of phone activities. The generic costs allow the system to be adopted in various applications since the costs involved in sensing tasks may vary. For example, the authors in [65] considers the cost in terms of energy consumption. The authors of [76] consider the cost in terms of incentives. They have designed incentive mechanisms for mobile phone sensing to attract more

user participation. There are two system models: the platform-centric model where the platform provides a reward shared by participating users, and the user-centric model where users have more control over the payment they will receive.

We address limited 3G data plans of mobile phones in this work as well. The downlink limitation of 3G data plan of mobile phones has been addressed in a few works. the authors of [30] have proposed to exploit opportunistic communications to facilitate information dissemination in the emerging Mobile Social Networks (MoSoNets) and thus reduce the amount of mobile data traffic. In [60], the authors have proposed to exploit delay tolerance and tries to download contents when users are close to Wi-Fi access points. Our works differs from these work in that we consider uplink limitation and uploading cost. In order to achieve system wide cost optimization, the mobile phone sensing and data offloading should be considered at the same time. To the best of our knowledge, we are the first to address both mobile phone sensing in target monitoring and 3G data plan limitation together to minimize total cost.

4.3 Problem Formulation

We consider a mobile phone system consists of a small number of central cloud servers and a large number of mobile phone users as participants in the system. The system is primarily used to monitor a set of targets or small areas. Since the mobile phones connects to the internet via either cellular or WiFi network, the cloud servers are assumed to be capable of communicating with these participated phones at anytime. We denote these small areas as N targets (points of interest). The system monitors the targetss by periodically querying the status of the points. The cloud servers for-

ward queries to the participating phones. The mobile phones then provide sensing service for the system. The cloud servers, in turn, collect sensing data from the participated phones. The system may choose to offer financial compensation to attract participation of users. Such system could be easily deployed via a phone app.

The locations of the targets are assumed to be close to or along the roads so that the mobile phone users that pass by can collect the sensing data. The phones users may collect the sensing data actively by moving to the target point, i.e., through participatory sensing. Alternatively, the system may forward the query of a specific target to those phones which is currently at the target or may pass the target in the near future and schedule the sensing. The phone will then be able to collect the sensing data without user intervention, i.e., through opportunistic sensing. The opportunistic sensing is quite user-friendly in that it does not put any requirement on user actions. But due to the unpredictability of the users' travel paths, the sensing data may be missing or inaccurate. Also, some types of sensing data such as pictures and videos, which requires users' operations, may not be available in such way. In contrast, the participatory sensing guarantees accurate sensing data and has no constraints on the types of sensing data it can collect. Since it requires a lot of user interventions, how to keep users engaged in the sensing will be a problem in participatory sensing.

After the sensing data is collected by the phones, the system will take charge of the scheduling of data offloading. When a phone is unable to upload all the sensing data due to its limited data plan, the system will also schedule data exchange among phones so that other phones can upload the remaining data with their available data plan. The data exchange can also be achieved in an opportunistic manner in that mobile phones exchange data when they get closer with each other.

TABLE 4.3.1: Key notation.

Notation	Definition
N	number of target points
T	time limit before status of all target points are uploaded
M	number of participating mobile phones
l_i	data plan limit of phone i
d_i	available data storage of phone i
c_{ij}	sensing cost when phone i senses target j
o_{ij}	communication cost between phone i and phone j per data unit
w_i	data offloading cost per data unit
r	communication rate for all data exchanges in data unit per time unit
V	set of vertices in flow network
E	set of edges in flow network
f_e	amount of flow on edge e in flow network
c_e	capacity of edge e in flow network
γ_e	cost of edge e in flow network
E_v^{in}	set of edges incoming to vertex v in flow network
E_t^{out}	set of edges outgoing from vertex v in flow network

Since all phone activities, including data sensing, data exchange and data offloading, may take a certain amount of time to complete. The system allows a tolerance period T before the status of all targets get sensed and uploaded. The larger the number of mobile phones traveling in the area, a smaller T may apply to closely resemble the real time monitoring.

The key notation used through the chapter is summarized in Table 4.3.1 for easy reference. We denote the number of participating mobile phones traveling around in the area as M. The available data plan limit of phone i is denoted as l_i. The available data storage of phone i is denoted as d_i.

Without loss of generality, we assume the size of the sensed data is the same at all the targets and is set to 1 data unit. Most sensing tasks involve taking a snapshot of

the data, such as taking a picture, measuring the temperature or noise level. These activities usually can be completed instantly and takes only a small amount of time. We assume that the phone i can collect the sensing data of the target j as long as the target falls in its sensing range. The opportunistic communication among mobile phones can either rely on Bluetooth or WiFi techniques. WiFi is generally considered better as its communication range is relatively large and energy consumption is relatively small. To simplify the model, we assume that the communication rate is fixed among all opportunistic communications, denoted as r. It does not depend on the distance between the phones that are involved in the communication or the signal strength of the communication.

All phone activities involve certain cost. The sensing cost is denoted as c_{ij} per data unit. We assume that the phone i can exchange the sensed data with another phone j in transmission range. The communication cost is denoted as o_{ij} per data unit. We assume that the phone i can offload the sensed data to the cloud server anytime as long as the cellular network is available. The offloading cost is denoted as w_i per data unit.

A feasible schedule of phone activities should enable that the sensing data at all targets are collected and uploaded to the cloud servers within time limit T. The optimal schedule should be the feaible schedule with the minimal cost among all feasible schedules. The design goal of the system is to find the optimal schedule of phone activities and apply it on the phones. We refer to the problem as *Minimum Cost Target Sensing* problem, denoted as MinSense problem.

Minimizing the total cost of the system may no longer be a preferable goal when we consider the cost of an individual phone. For a schedule that minimizes the total cost, the cost of a particular phone may be much larger than those of other phones.

We thus propose another design goal that is to find the optimal schedule of phone activities that leads to fair costs among mobile phones. We denote the problem as FairSense problem.

4.4 Optimal Algorithms

In this section, we formulate MinSense problem as a minimum cost flow problem under an idealized assumption, i.e., we assume the trajectories of the phones during time period T are known beforehand. We assume that we know mobile phones' trajectories during the period T. We can create a flow network based on mobile phones' trajectory then obtain optimal sensing schedule by solving the minimum cost flow problem on created network. Although it is impossible to foreknow mobile phones' trajectories, the solution to the problem helps the design of our heuristic algorithm as we will describe later. It also serves as our benchmark problem when we evaluate the performance of the heuristic algorithms to be described in Section 4.5.

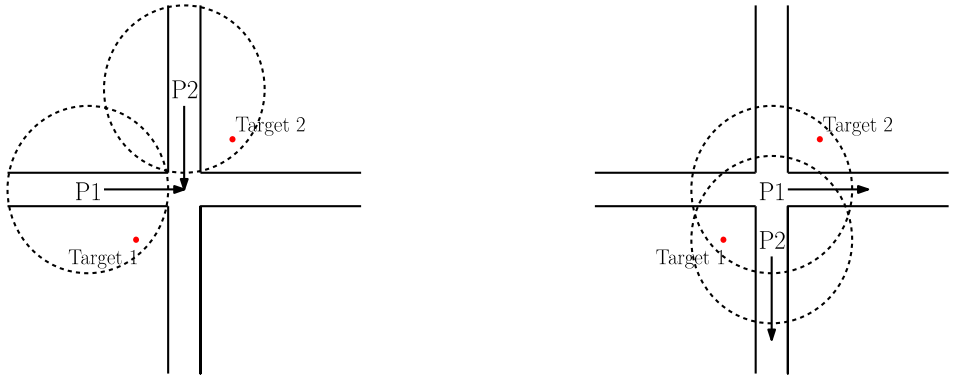
We now describe how to create the flow network $G(V, E)$ based on mobile phones' trajectories. We consider all mobile phone activities, including data sensing, data exchange and data offloading, as data transfers among target, phones and cloud servers. Data exchange is considered as data transfer among phones and data offloading is considered as data transfer from phone to cloud server. Data sensing can be represented as data transfer from target to phone. Naturally, we can use network flow to represent the data transfers. We will firstly describe how to add vertices V of flow network G and describe next how to add edges E among vertices.

There are mainly two types of vertices. The first type of vertices are *target vertices*.

We add one vertex for each target i, denoted as a_i. The second type of vertices are *phone vertices*. If we simply adding one vertex for one phone, it will be difficult to represent the sequences of data transfers. To account for the activity sequence, we take a time-expanded approach. We adde multiple vertices for one phone along the timeline. We evenly divide the period T to small time slots. The length of the time slots should be sufficiently small so that any connectivity change among phones only happens at the end of the time slot and there is no connectivity change within the time slot. This is to make sure that the network topology within one time slot will not change. The topology change within a time slot may lead to that an invalid data transfer being considered or a better data transfer path is missed. The connectivity change includes both the creation of a new connection and the disconnect of an established connection. The new created connection may either be that a target enters a phone's sensing range or that a phone enters the communication range of another phone. The disconnected connection may be that a target leaves a phone's sensing range or that a phone leaves the communication range of another phone. Without loss of generality, we assume the length of the time slot is 1 time unit and thus there are totally T time slots. We then add T vertices for each phone, i.e., totally $T \times M$ phone vertices. To represent the mobile phones' activity over time period T, the vertex representing the status of phone i at jth time slot is denoted as $p_{(i,j)}$. In addition, we add a *virtual source* vertex S and a *virtual sink* vertex D to the network.

The addition of edges basically follows the flow representation of data transfers among targets, mobile phones and cloud servers. An directed edge between two vertices represents a possible directed data transfer between the two. We denote the flow amount on the edge e as f_e. Specifically, if at time slot t, phone i can sense target j, we add an directed edge e from vertex a_j to $p_{(i,t)}$. The capacity of the edge

is set to 1 and the cost is set to the sensing cost of phone i at target j, i.e., c_{ij}. A non-zero flow value f_e represents that phone i senses target j at time slot t. The phone i have exactly 1 data unit of sensing data at target j under our assumption. It will be responsible for sending f_e portion of sensing data at target j to other phones or cloud server (data offloading). If at time slot t, phone i can send data to phone j, we add a directed edge e from vertex $p_{(i,t)}$ to $p_{(j,t)}$. The capacity of the edges is set to the communication rate r, i.e., the amount of data unit can be transferred per time unit, between two phones. The cost of the edge is set to the sum of communication cost of two phones, i.e., $o_{ij} + o_{ji}$. A non-zero flow value f_e represents that phone i send f_e amount of data in its data storage to phone j. Similarly, if phone j can send data to phone i, we add a directed edge from vertex $p_{(j,t)}$ to $p_{(i,t)}$. For every phone i and every time slot t except last time slot, we add a directed edge e from $p_{(i,t)}$ to $p_{(i,t+1)}$. Its capacity is set to amount of data unit phone i can store during period T, i.e., d_i. Its cost is set to 0. A non-zero flow value f_e represents that phone i holds f_e amount of data from t to $t+1$. For every target i, we add a directed edge from virtual source S to a_i. Its capacity is set to 1 and cost is set to 0. A non-zero flow value f_e represents that f_e amount of sensing data at target i is finally uploaded. If $f_e = 1$, one data unit of sensing data at target i is fully uploaded. If $f_e < 1$, it means only f_e amount of data is uploaded. For every phone i at time slot T, i.e., the last time slot, we add a directed edge e from $p_{(i,T)}$ to virtual sink D. Its cost is set to the uploading cost of phone i, i.e., w_i. Its capacity is set to phone i's available data plan limit, i.e., l_i. A non-zero flow value f_e represents phone i upload f_e amount of data to cloud server at the end of period T. Although mobile phones can complete data offloading any time during the period T, we prefer them to do it at the end of T so that cloud server can calculate optimal offloading schedule based on the sensing data



(a) System status at time point 1. (b) System status at time poine 2.

FIGURE 4.4.1: An example scenario where there are two phones passing a crossroads and two targets required to be monitored.

distribution.

An example is shown in Figure 4.4.1. There are two phones passing a crossroads, denoted as P1 and P2. Their moving directions are shown in arrows. The communication ranges and sensing ranges are assumed to be same and are shown as dashed circles. At time point 1, phone 1 can sense target 1 and phone 2 can sense target 2. At time point 2, after short movements, both phones fall into the communication range of each other. Phone 1 can sense both targets while phone 2 can sense target 1. The created flow network is shown in Figure 4.4.2.

We now show that a maximum flow solution represents a feasible schedule of phone activities to MINSENSE problem if the maximum flow equals to N. We have the following lemma.

Lemma 4.4.1. *If the maximum flow on flow network $G(E, V)$ is equal to N, the maximum flow represents a feasible schedule of phone activities to* MINSENSE *problem.*

Proof. If the maximum flow equals to N, it implies that the flow value of the incoming

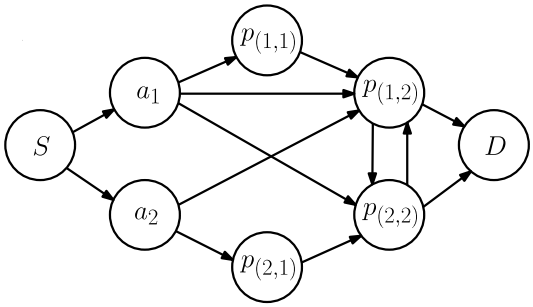


FIGURE 4.4.2: Time expanded flow network based on scenario shown in Figure 4.4.1

edge of vertex a_i is 1 for all target i. Due to the flow constraint, there must be some outgoing edges from a_i that have non-zero flow values connected to one or more mobile phones. Based on the way we create flow network G, it implies that all the targets are sensed by one or more mobile phones.

If the maximum flow equals to N, it also implies that the sum of flow values of the edges from $p_{(i,T)}$ to D is N. Based on the way we create flow network G, it implies that all the sensing data are uploaded by one or more phones, which concludes the proof. □

We can then define a minimum cost flow problem on flow network $G(E, V)$. We denote the set of edges outgoing from vertex v and incoming to v as E_v^{out} and E_v^{in} respectively. We denote the capacity and cost of the edge e as c_e and γ_e respectively. The linear programming formulation of MINSENSE problem, denoted as LP-MIN, is as follows.

$$\text{minimize: } \sum_{e \in E} f_e \gamma_e$$

$$\text{s.t. } \sum_{e \in E_S^{out}} f_e = N, \tag{4.4.1}$$

$$\sum_{e \in E_v^{in}} f_e = \sum_{e \in E_v^{out}} f_e, \forall v \in V \backslash \{s, z\} \tag{4.4.2}$$

$$0 \leq f_e \leq c_e, \forall e \in E \tag{4.4.3}$$

We have the following theorem.

Theorem 4.4.1. *If the maximum flow on flow network $G(V, E)$ is equal to N, the solution of* LP-Min *represents an optimal schedule of phone activities to* MinSense *problem.*

Proof. Since the maximum flow on flow network $G(E, V)$ is equal to N, there exists at least one feasible schedule based on Lemma 4.4.1. Due to the way we create flow network G, the sensing cost, communication cost and uploading cost of mobile phones' activities are associated with corresponding edges. The solution obtained by solving the min cost flow problem LP-Min represent a feasible schedule with minimum cost. Thus the solution represents the optimal schedule of phone activities to MinSense problem. □

We can create a similar flow network for the FairSense problem. Since the goal is now to obtain the schedule with fair costs among mobile phones, the minimum cost flow algorithm no longer applies. To obtain the schedule with fair costs, we take a two steps approach. We firstly convert the problem to a max-min problem where the

objective is to minimize the maximum cost among mobile phones. Let E_i denote the edge set where phone i incurs cost. It includes edges associated with data sensing, data transmission and data offloading of phone i during the whole period T. The linear programming formulation of this max-min problem, denoted as LP-Maxmin is as follows.

$$\begin{aligned} \text{minimize: } & \mathcal{X} \\ \text{s.t. } & \sum_{e \in E_i} f_e \gamma_e \leq \mathcal{X}, \forall \text{ phone } i \\ & \text{constraints (4.4.1)-(4.4.3) in LP-Min} \end{aligned} \tag{4.4.4}$$

After solving the LP-Maxmin problem, we obtain a fair cost schedule. Since there can be multiple fair schedule with the same $\mathcal{X}$ value, the obtained schedule may not be the one with the minimum cost. We add an additional step. We use $\mathcal{X}$ as an input to the following linear programming problem, denoted as LP-Fair.

$$\begin{aligned} \text{minimize: } & \sum_{e \in E} f_e \gamma_e \\ \text{s.t. } & \sum_{e \in E_i} f_e \gamma_e \leq \mathcal{X}, \forall \text{ phone } i \\ & \text{constraints (4.4.1)-(4.4.3) in LP-Min} \end{aligned}$$

Solving LP-Fair formulation in addition to LP-Maxmin ensures that we find the fair schedule with the minimum total cost, which is the optimal schedule for FairSense

problem.

4.4.1 Flow Network Optimization

In the flow network we create as above, there are multiple vertices for the same phone at different time slots. If period T is relatively long or the time slot is relatively short, the resultant flow network will be too large in scale for algorithms to compute solution efficiently. We will now show that, instead of creating a set of phone vertices at every time slots, we only need to create a new set of phone vertices only when there is a connectivity change among mobile phones or among the mobile phones and the targets. Specifically, consider the time points when there is at least one connectivity change. We denote these time points as $\{t_1, t_2, t_3, \ldots, t_T\}$. There is no connectivity change between two consecutive time points. We can create a reduced flow network $G(V, E)$ as follows.

Similarly as in previous section, we create N target vertices, a virtual source S and virtual sink D. As for the phone vertices, for every time point t_k and phone i, we create a phone vertex $p_{(i,t_k)}$. Edges are created in a similar way as in previous section except that time points used are $\{t_1, t_2, t_3, \ldots\}$. For the edge between two phone vertices at time point t_k, the capacity is set to $r \times (t_{k+1} - t_k)$, i.e., the amount of data unit can be transferred during period $t_{k+1} - t_k$ between two phones.

We have following lemma.

Lemma 4.4.2. *Consider a set of phones and their trajectories in period T. We denote the flow network created as described in previous section as G and the flow network created following description in this section as G'. A solution A to the minimum cost flow problem on the flow network G can be converted to a solution A' to the minimum*

cost flow problem on the flow network G' and vice versa.

Proof. A and A' are inter-convertible implies that flow amount and paths in A and A' are inter-convertible. Consider converting edges in A to edges in A'. We only need to show that edges among targets and phones and edges among phones are inter-convertible between A and A'. Consider a time point t_k when at least one connectivity change happens and its following period $t_{k+1} - t_k$ without connectivity change. Since there is no connectivity change within time slots of graph G, t_k should be start of some time slot in G.

Consider edges outgoing from a target i with non-zero flows in A during $t_{k+1} - t_k$. Since we assume that phone can get all sensing data instantly once connection between the phone and the target is created, for target i, the corresponding outgoing edges in A to phone j can be combined into one edge with sum of the flows to phone j at time t_k. The sensing costs are the same as before. All the outgoing flows from phone j during $t_{k+1} - t_k$ in A will not be affected as phone j hold all the data at the beginning of the period.

Consider edges from phone i to phone j with non-zero flows in A during $t_{k+1} - t_k$. We can combine these edges into one edge from phone i to phone j at t_k the capacity as the sum of the capacities and the flow as the sum of the flows. The communication costs are the same as before. If we combine all such edges during $t_{k+1} - t_k$, all such edges will be removed and there is only one edge between any pair of phones. All flow amount remain same as the sum of the flow values is used as the new flow value on combined edges. Since there are no connectivity changes during $t_{k+1} - t_k$, if two phones are connected at t_k, then they are connected at all time slots during $t_{k+1} - t_k$. Any flow paths in A will not be affected in converted A.

Notice that by combining edges, we convert a solution A into a solution A' with same flow amounts and same flow paths, which concludes the proof of conversion from A to A'. The conversion from A' to A is similar and is omitted. □

With Theorem 4.4.1 and Lemma 4.4.2, we then have the following theorem which shows that minimum cost maximum flow on the reduced flow network is an optimal schedule to MinSense problem too.

Theorem 4.4.2. *If we create flow network $G(V, E)$ as described in this section and the maximum flow on G is equal to N, the solution of* LP-Min *is an optimal schedule to* MinSense *problem.*

4.5 Heuristic Algorithms

Since it is impossible to foreknow the trajectories of the mobile phones, optimal algorithms may not be directly applied in practice. In this section, we propose several practically feasible heuristic algorithms to solve both MinSense and FairSense problems.

4.5.1 Estimation-based Algorithms

Although foreknowing the trajectories of the mobile phones cannot be achieved, we can always estimate and predict the trajectories of mobile phones. The estimation can be based on mobile phones' current trajectories and other status information such as traveling speed and moving direction. The estimation may apply some predefined patterns such as walking along the road with current speed without turns. If the

mobile phone user happens to pass the area frequently, historical moving pattern may also help with improving the accuracy of the estimation. The existing trajectories can be obtained from mobile phones, e.g., we may ask mobile phones to report their locations to cloud servers periodically via built-in GPS, which is widely available nowadays on phones.

The estimation-based heuristic algorithm is described in Algorithm 6. The mobile phones are required to report their status information periodically for cloud servers to estimate their moving trajectories. The algorithm runs periodically during time period T. Every time when it runs, it firstly estimates phones' trajectories and creates flow network based on estimated trajectories. It then solves LP-Min formulation and send obtained schedule to mobile phones for execution.

Since the algorithm runs for multiple rounds, in later rounds, mobile phones may have already had some data stored in their storage. It may either be collected by sensing the target or transmitted from another neighbor in previous rounds. If we don't consider these previous phone activities in creating new flow network, it may cause problems. For instance, a target may have been sensed and its data is now in the data storage of phone i. The obtained schedule on newly created flow network may still ask some phone to sense it as it has never been sensed before, although phone i may simply upload it at the end of T. To reflect previous phone activities in new flow network, we take an approach where we treat previous phone activities as edges in the new flow network. Specifically, in addition to normal status that are useful for trajectory estimation, we require phones to report their activities in previous report periods. When we create new flow network at time t, we create a flow network starting from start of the period T. We create all edges as normal for time period after t. As for the edges corresponding to the time period prior to t, we

only keep those edges that are "executed" by the phones. The capacities on these edges should be set to the actual amount of data in the corresponding data transfer. For instance, suppose phone i sensed target j at previous time slot t_1. And suppose phone k send x unit of data to phone l at previous time slot t_2. When we create new flow network, other than the edges and vertices based on predicted trajectories, we also add a edge from a_j to $p_{(i,t_1)}$ with capacity 1 and a edge from $p_{(k,t_2)}$ to $p_{(l,t_2)}$ with capacity x.

Due to the unpredictable nature of phones' trajectories, it is unavoidable that a mobile phone may miss some scheduled phone activities. To decrease the chance of missing a scheduled activity for phones, we may require phones to report their locations more frequently and require cloud servers to calculate and push the new schedule more frequently. This may result in more energy consumption of phones due to the frequent communication with cloud server and frequent use of GPS device, which is rather energy-hungry. A possible way to decrease the energy consumption is to use GPS less and apply method as introduced in [21]. It relies on the information from inertial sensors (such as accelerometer and digital compass) of mobile phones to approximate current location. The phones only need to use GPS once when it enters the area.

To avoid unnecessary transmissions and save communication cost, in any data exchange, the sender should always send only the data that the receiver currently does not have. The information can either be retrieved from the cloud server or right before the data exchange time.

We found that in our simulation, employing FairSense in estimation-based algorithm results in a large number of sensing / communication failures when the trajectory estimation is not accurate, which leads to ineffectiveness in balancing phone

Algorithm 6 Estimation-based algorithm

1: **repeat**
2: predict mobile phones trajectories based on mobile phone's status report.
3: create flow network based on predicted trajectories starting from beginning of the period T.
4: remove all edges prior to current time.
5: **if** previous actions are executed by some mobile phone i **then**
6: Add a edge for corresponding action with capacity set as the data amount collected / transferred.
7: **end if**
8: solves LP-Min or formulation and send solution to mobile phones.
9: **for all** mobile phone i **do**
10: **if** a non-zero flow is found on i's incoming or outgoing edges from now to next report time or T **then**
11: execute actions.
12: report actions executed and data amount collected / transferred at next report time.
13: **end if**
14: **end for**
15: wait until next calculation time.
16: **until** end of the period T

costs. We further propose an adaptive cost algorithm in next section to balance the phone cost with the help of MinSense.

4.5.2 Adaptive Cost Algorithm

In order to achieve the cost fairness, the estimation-based heuristic algorithm tries to achieve cost fairness by solving max-min problem LP-Maxmin. In this section, we propose an adaptive cost algorithm which takes an alternative approach, as described in Algorithm 7. The algorithm runs in multiple rounds. It dynamically adjusts the cost of phones in each round and run minimize cost formulation LP-Min. The cost of a phone will be increased in the current round if it participates in some activities in previous round. With the cost adjustment, when solving LP-Min in the current round, the probability the phone gets selected in activities will decrease. It gives other phones opportunities to participate and thus balance the cost among all phones. As the costs of phones are fixed and will not increase, the costs of phones should still be calculated based on their actual costs.

The cost γ of a phone could be adjusted by any predefined functions $\mathcal{F}(\gamma)$. To balance the costs among phones, the $\mathcal{F}(\gamma)$ should be monotonic increasing function of γ. A simple choice could be $\mathcal{F}(\gamma) = \alpha\gamma$, $\alpha > 1$. When α gets closer to 1, the algorithm produces the schedule that is close to the goal of minimizing cost. The larger the α is, the algorithm gets more aggressive in balancing the costs among phones. This also provides us with the opportunities to adjust the costs to achieve the different design goals of the system.

An interesting variant of the algorithm is to decrease the cost of certain phones. The algorithm will pick the phone more frequently than other phones. This may

be useful when there are phones that can provide the data with better quality than that provided by other phones. For instance, such "phone" could be a special patrol vehicle equipped with high quality sensors that shuttles back and forth among target points.

Algorithm 7 Adaptive cost algorithm

repeat
    predict mobile phones trajectories based on mobile phone's status report.
    create flow network based on predicted trajectories starting from beginning of the period T.
    remove all edges prior to current time.
    if previous actions are executed by some mobile phone i **then**
        Add a edge for corresponding action with capacity set as the data amount collected / transferred.
    end if
    solves LP-Min formulation and send solution to mobile phones.
    for all mobile phone i **do**
        if a non-zero flow is found on i's incoming or outgoing edges from now to next report time or T **then**
            execute actions.
            report actions executed and data amount collected / transferred at next report time.
            adjust its cost γ with function $\mathcal{F}(\gamma)$
        end if
    end for
    wait until next calculation time.
until end of the period T

4.5.3 Baseline Algorithm

We also designed a baseline algorithm as described in Algorithm 8. The algorithm is primarily used for performance comparison in our simulations. It is an algorithm trying to complete sensing tasks without much consideration of the costs. The algorithm does not require phones to periodically report their locations. The cloud

servers only send notifications to phones so that phones know that how much of the sensing data of targets has been uploaded. The phones senses the target whenever a target is in sensing range and the data of the target has not been fully uploaded. It uploads whenever there is data in its storage when it still has available data plan left. If it has data in storage not uploaded either due to limitation of data plan or transmissions from other phones, it sends the data out to every phone it meets when it allows.

To avoid unnecessary data transfers, in any data exchange, the sender should always send only the data that the receiver currently does not have. Also, all phones get notified when any portion of the data gets uploaded by some phone. The phone erases the data portion in its storage when server notifies that the portion has been uploaded.

Algorithm 8 Baseline algorithm

```
1: for all mobile phone do
2:     repeat
3:         if the data in storage that has been uploaded then
4:             discard the data.
5:         end if
6:         if a target is in range and the data of the target has not been fully uploaded.
   then
7:             sense the target.
8:         end if
9:         if there is data in storage and data plan is still available then
10:            upload the data as much as possible
11:        end if
12:        if there is data in storage and other phones is in range then
13:            for all other phones do
14:                send as much as possible data that are not on the receiving phone
15:            end for
16:        end if
17:    until end of the period T
18: end for
```

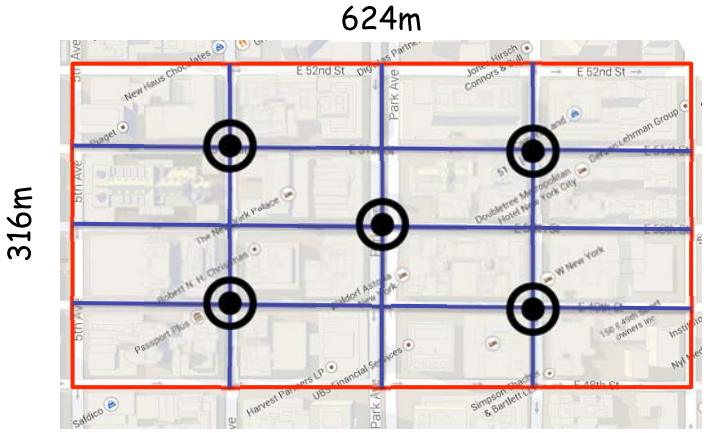


FIGURE 4.6.1: Simulation region at Manhattan, NYC between East 48th and East 52nd street, 3rd and 5th avenue (obtained from Google Map)

4.6 Performance Evaluation

In this section, we evaluate the performance of the various algorithms through simulation. We pick a region from Google Map as the target region as shown in Figure 4.6.1. The region is located in Manhattan, NYC. It is a rectangular area between East 48th and East 52nd street, 3rd and 5th avenue. The length and width fo the region, obtained with the help of Google Map API, are 624 meter and 316 meter respectively. There are totally 9 intersections in the region and we pick 5 targets as shown in circles in Figure 4.6.1.

The sensing range and communication range of mobile phones are both set to 40 meters. Data transfer rate among phones is set to 0.5 data unit per second. As our algorithms may be associated with different types of cost such as energy consumption and financial compensation. We take a generic approach in choosing cost values in our simulations. The costs of data sensing, data exchange and data offloading are all randomly picked from $\{1.0, 3.0\}$. Available data plan of mobile phones is randomly picked from $\{0.1, 0.3\}$ data unit. Each target has 1 unit of data required to be

uploaded in 15 minutes.

The mobility model we use to generate mobile users moving trajectories is similar to the well-known Manhattan model [8]. Specifically, each user is assumed to enter the target region from a road at a random time in the first 10 minutes, randomly pick a speed from $\{0.5, 1.5\}$ meters per second and start moving into the region. The user always move towards the intersection, and then when arrives at the intersection, he moves straight ahead with a probability of 50% and turn left or right with equal probabilities (i.e., 25%).

We compare the proposed heuristic algorithms with the optimal solutions for both the Minsense and Fairsense problems (see Section 4.4). In both Algorithm 6 and 7, the algorithms estimate the trajectories of phones following a simple pattern that the phones walk straight without turns. The cloud servers start considering a phone in computation only after it enters the region. Algorithm 6 runs MinSense optimal algorithm in each round. Algorithm 7 use cost adjust function $\mathcal{F}(\gamma) = \alpha\gamma$. The cost multiplier, α, ranges from 1.05 to 1.25. The mobile phones report their status to cloud servers once per minute and the cloud server pushes the new schedule to mobile phones every minute.

In the simulation scenario, we increase the number of phones from 45 to 75 with 5 as the step size. The total available data plan limit of all phones are approximate from 9 to 15 data units (recall that data plan limit is randomly picked from $\{0.1, 0.3\}$ data unit). For a specific number of phones, we randomly generate 10 trajectories of phones and the data points collected for all algorithms are averaged over 10 scenarios.

Figure 4.6.2 shows the result of total cost of the three algorithms under different number of phones. We can see that the MinSense optimal algorithm saves most as expected, up to 80% compared the baseline algorithm, Algorithm 8. Both Algorithm

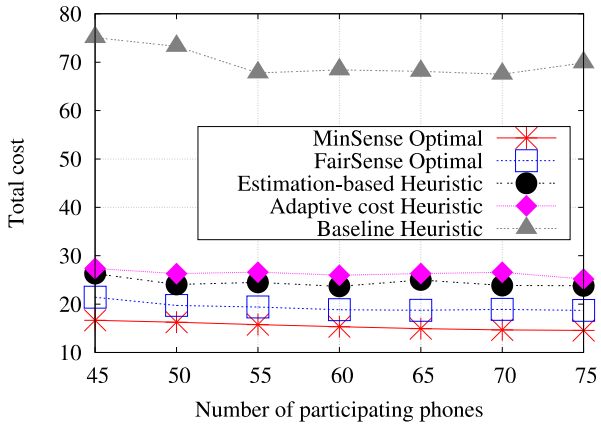


FIGURE 4.6.2: Total cost of algorithms under different number of phones.($\alpha = 1.25$)

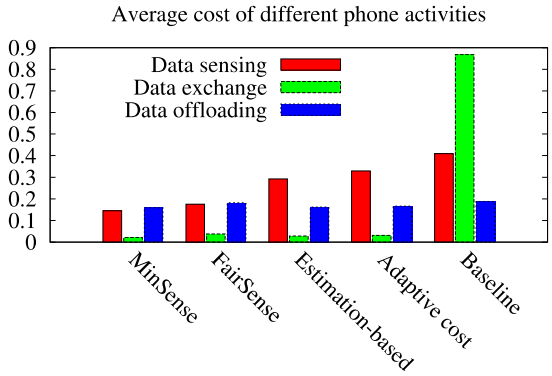
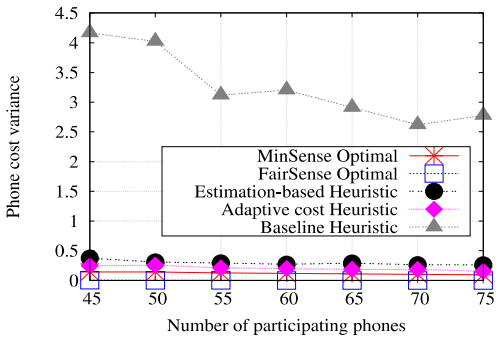


FIGURE 4.6.3: Average cost of different phone activities with 50 phones.($\alpha = 1.25$)

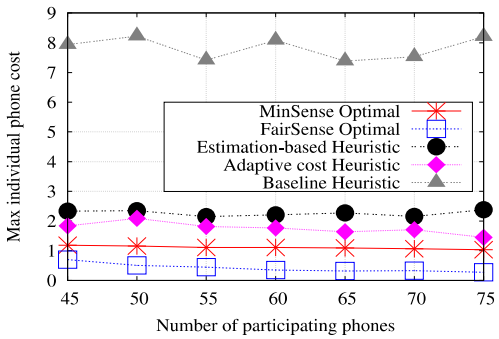
6 and 7 perform much better than the baseline algorithm. The savings of Algorithm 6 and 7 compared to the baseline algorithm are up to 67% and 65% respectively. The overall cost of all algorithms tends to decrease as number of phones increases. With more phones that are available to be chosen, cloud servers has higher chance to pick the phones with lower cost for the same task. The overall cost of Naive algorithm increases together with the number of phones. The baseline algorithm also have higher chance to complete uploading all data earlier before it spends more costs on data exchange.

Figure 4.6.3 shows the result of the average cost of different phone activities with 50 phones. We can see that baseline algorithm spends lots of cost on data exchange, almost 40 times more than other algorithms. Without the schedule optimization, the mobile phones under baseline algorithm have to disseminate data as much as possible so that the sensing task will not fail. Compared to baseline algorithm, Algorithm 6 and 7 saves 30% and 22% data sensing cost respectively. They both save 16% on data offloading cost. Compared to MinSense optimal algorithm, Algorithm 6 and 7 spends more on data sensing tasks. This is because that there are few phones available at early stage of the simulation, cloud servers has no knowledge that more phones are coming to the area and thus the phones have to sense target whenever in range. The uploading costs of Algorithm 6 and 7 are quite similar to MinSense optimal algorithm as mobile phones only upload at the end of the period, right after they receives the optimal schedule from cloud servers.

Figure 4.6.4 shows the cost fairness of all algorithms under different number of phones. Figure 4.6.4(a) shows the variance of the costs among all phones. As expected, the FairSense optimal algorithm produces fairest schedule. Compared to baseline algorithm, Algorithm 6 and 7 achieves 1.8 and 2.3 times smaller in terms

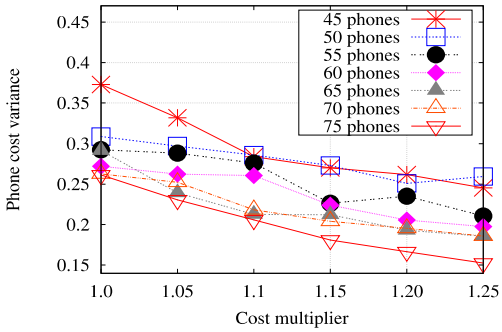


(a) Variance of costs among phones.

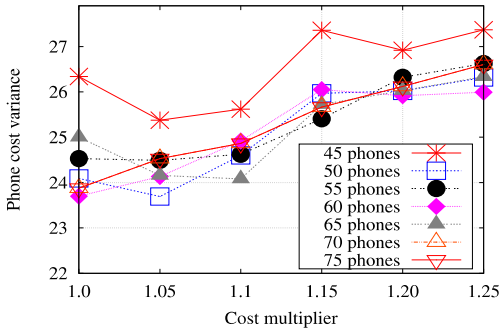


(b) Maximum individual phone cost.

FIGURE 4.6.4: The variance of the costs among phones and the maximum individual phone cost under different number of phones. ($\alpha = 1.25$)



(a) Trend of variance of costs.



(b) Trend of total costs.

FIGURE 4.6.5: The trend of variance of costs and total costs under different cost multiplier α

of variance respectively Figure 4.6.4(b) shows the maximum individual phone cost among all phones. The performance of all algorithms in terms of maximum individual phone cost matches that in terms of variance of the costs.

We also evaluate the performance of Algorithm 7 under different cost multiplier α. Figure 4.6.5 shows how the variance of costs and total costs change over different cost multiplier α. Figure 4.6.5(a) shows the trend of variance of costs. As expected, the variance decreases, i.e., the fairness of costs increase, as the multiplier increases. We notice in Figure 4.6.5(b) that although not always, the total costs increase slowly

as the multiplier increases. This demonstrates that balancing the costs among the phones may incurs extra cost. This is because that a phone with low cost in current round may not be preferred in next round as its cost will increase once it participate in current round. The increase in total cost as α increases from 1.0 to 1.25 is around 9% , while the variance is about 1.8 times smaller, in other words, the fairness becomes 1.8 times better.

4.7 Summary

In this chapter, we study how to cost effectively schedule phones to monitor a set of targets. We proposed two optimal algorithms in the offline setting, i.e., assuming the trajectories of the phones are known beforehand, addressing minimum cost and cost fairness, respectively. We then develop one estimation-based algorithm and one adaptive cost algorithm to address both problems, which are under realistic assumptions and can be applied immediately in practice. Extensive simulation results show that all proposed algorithms perform well in terms of both minimizing total cost and achieving cost fairness among the phones.

Chapter 5

Conclusion and Future Work

Employing mobile sensors in deployment and sensing tasks provides many additional benefits over the static sensors. The locomotive abilities of mobile sensors greatly facilitates the desired deployment to be achieved in unknown and hostile areas. The mobile phones, which are naturally mobile sensors, help completing sensing tasks without any deployment cost. In this dissertation, we have presented cost-effective deployment and sensing problems in mobile sensor networks. Specifically, we presented distributed algorithms for achieving even energy efficient self-deployment. We presented computation efficient algorithms for energy-balanced deployment to a priori known target locations. We designed generic target monitoring system through mobile phone sensing and presented cost-efficient practical algorithms for the system. Our work is summarized as follows.

- *Distributed algorithms for energy-efficient even self-deployment in mobile sensor networks.* We studied the problem of energy-efficient even self-deployment in mobile sensor networks. In order to address the issue of energy-efficient de-

ployment, which is still a challenge in the widely used Lloyd's method, a new algorithm, DEED algorithm, is proposed. Simulation results demonstrate that DEED performs well in different scenarios. Specifically, it leads to up to 54% less traveling distance and 46% less energy consumption than Lloyd's method.

As future work, we will explore even self-deployment of sensors in areas with obstacles.

- *Energy-balanced mobile sensor deployment to a priori known target locations.* We studied the problem that determines, for a given set of a priori known target locations and a set of mobile sensors, which target location a mobile sensor should move to so that each target location is covered by at least one sensor, and the duration for which a target location can be covered is balanced. We propose a set of light-weight deployment algorithms that are suitable for resource-limited sensors. Specifically, we start with the simple scenarios where all sensors have the same starting location and initial energy or only a small number of sensors have different starting location and initial energy, and propose optimal algorithms to solve them. We then consider the general scenario, prove that the problem is NP-hard and propose light-weight heuristic algorithms for the scenario. Extensive simulation results demonstrate that our proposed algorithms either achieve close-to-optimal performance or achieve similar performance as the best known approximation algorithm while being much more time-efficient.

  As future work, we will explore centralized algorithms that do not depend on the locations of the sensors. We will also explore distributed and randomized algorithms.

- *Cost-efficient target monitoring and data offloading through mobile phone sens-*

ing. We studied how to cost effectively schedule phones to monitor a set of targets. We propose two optimal algorithms in the offline setting, i.e., assuming the trajectories of the phones are known beforehand, addressing both minimum cost and cost fairness problems. We then develop one estimation-based algorithm and one adaptive cost algorithm to address the problems, which are under realistic assumptions and can be applied immediately in practice. Our extensive simulation results demonstrate that all proposed algorithms perform well in terms of both minimizing total cost and achieving cost fairness among the phones.

As future work, we plan to consider probability distribution of trajectories of mobile phone users. Incorporating the probability distribution in trajectory estimation is expected to improve the accuracy of the estimation, which is vital to the performance of the estimation based algorithms. We will also explore different cost functions in the adaptive cost algorithm and evaluate how they impact the performance of the algorithm.

Bibliography

[1] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[2] The network simulator ns-2. http://www.isi.edu/nsnam/ns/.

[3] Qhull. http://www.qhull.org.

[4] IEEE P802.11p/D2.01, Draft Amendment for Wireless Access in Vehicular Environments (WAVE), February 2007.

[5] K. S. A. Okabe, B. Boots and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley, 2nd edition, 2000.

[6] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM Journal on Computing*, 39(7):2970–2989, 2010.

[7] Y. Asami. A note on the derivation of the first and second derivatives of objective functions in geographical optimization problems. *Journal of The Faculty of Engineering, The University of Tokyo*, XLI(1):1–13, 1991.

[8] F. Bai, N. Sadagopan, and A. Helmy. The important framework for analyzing the impact of mobility on performance of routing protocols for adhoc networks. *AdHoc Networks Journal*, 1:383–403, 2003.

[9] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '06, pages 131–142, New York, NY, USA, 2006. ACM.

[10] X. Bai, Z. Yun, D. Xuan, B. Chen, and W. Zhao. Optimal multiple-coverage of sensor networks. In *Proc. of IEEE Infocom*, 2011.

[11] X. Bao and R. Roy Choudhury. MoVi: Mobile phone based video highlights via collaborative sensing. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 357–370, New York, NY, USA, 2010. ACM.

[12] N. Bartolini, T. Calamoneri, T. La Porta, and S. Silvestri. Mobile sensor deployment in unknown fields. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5, March 2010.

[13] B. A. Bash and P. J. Desnoyers. Exact distributed Voronoi cell computation in sensor networks. In *Proc. IPSN*, 2007.

[14] M. Bennewitz, W. Burgard, and S. Thrun. Optimizing schedules for prioritized path planning of multi-robot systems. 2001.

[15] I. Bezáková and V. Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18, 2005.

[16] M. Cardei and D. Du. Improving wireless sensor network lifetime through power aware organization. *ACM Wireless Networks*, 11(3), 2005.

[17] M. Cardei, M. Thai, Y. Li, and W. Wu. Energy-efficient target coverage in wireless sensor networks. In *Proc. of IEEE Infocom 2005*, volume 3, pages 1976–1984 vol. 3, 2005.

[18] M. Cardei and J. Wu. Coverage in wireless sensor networks. *Handbook of Sensor Networks*, pages 422–433, 2004.

[19] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *Computers, IEEE Transactions on*, 51(12):1448–1453, 2002.

[20] X. Chu and H. Sethu. A new distributed algorithm for even coverage and improved lifetime in a sensor network. In *Proc. IEEE Infocom*, pages 361–369, June 2009.

[21] I. Constandache, R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[22] J. Cortés, S. Martínez, T. Karatas, and B. Francesco. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20:243–255, April 2004.

[23] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. PRISM: Platform for remote sensing using smartphones. In *Proceedings of the 8th In-*

ternational Conference on Mobile Systems, Applications, and Services, MobiSys '10, pages 63–76, New York, NY, USA, 2010. ACM.

[24] A. Derhab and N. Badache. A self-stabilizing leader election algorithm in highly dynamic ad hoc mobile networks. *Parallel and Distributed Systems, IEEE Transactions on*, 19(7):926–939, July 2008.

[25] L. Ding, W. Wu, J. Willson, L. Wu, Z. Lu, and W. Lee. Constant-approximation for target coverage problem in wireless sensor networks. In *Proc. of IEEE Infocom*, pages 1584–1592, 2012.

[26] S. Doumit and D. Agrawal. Self-organized criticality and stochastic learning based intrusion detection system for wireless sensor networks. In *Military Communications Conference, 2003. MILCOM '03. 2003 IEEE*, volume 1, pages 609–614 Vol.1, Oct 2003.

[27] C. Gao, F. Kong, and J. Tan. HealthAware: Tackling obesity with health aware smart phone systems. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 1549–1554, Dec 2009.

[28] A. Gersho. Asymptotically optimal block quantization. *IEEE Transactions on Information Theory*, 25:373–380, July 1979.

[29] D. Golovin. Max-min fair allocation of indivisible goods. Technical Report CMU-CS-05-144, Carnegie Mellon University, 2005.

[30] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, and A. Srinivasan. Mobile data offloading through opportunistic communications and social participation. *Mobile Computing, IEEE Transactions on*, 11(5):821–834, May 2012.

[31] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom '03, pages 81–95, New York, NY, USA, 2003. ACM.

[32] N. Heo and P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man and Cybernetics*, 35:78–92, January 2005.

[33] R. Holman, J. Stanley, and T. Ozkan-Haller. Applying video sensor networks to nearshore environment monitoring. *Pervasive Computing, IEEE*, 2(4):14–21, Oct 2003.

[34] A. Howard, M. J. Matarić, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Journal Autonomous Robots*, 13(2), September 2002.

[35] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem. In *Proc. IEEE/RSJ International Conference on Distributed Autonomous Robotic Systems (DARS02)*, pages 299–308, 2002.

[36] M. Iri, K. Murota, and T. Ohya. A fast Voronoi-diagram algorithm with applications to geographical optimization problems. In *Proc. IFIP Conference on System Modelling and Optimzation*, pages 273–288, 1984.

[37] R. Iyengar, K. Kar, and S. Banerjee. Low-coordination topologies for redundancy in sensor networks. In *Proc. of ACM MobiHoc*, 2005.

[38] A. Jadbabaie, A. Ozdaglar, and M. Zargham. A distributed Newton method for network optimization. In *Proc. the 48th IEEE Conference on Decision and Control*, pages 2736–2741, December 2009.

[39] R. B. S. J.E. Dennis, JR. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentie-hall, 1983.

[40] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[41] W. Khan, Y. Xiang, M. Aalsalem, and Q. Arshad. Mobile phone sensing systems: A survey. *Communications Surveys Tutorials, IEEE*, 15(1):402–427, First 2013.

[42] S. Kumar, T. H. Lai, and A. Arora. Barrier coverage with wireless sensors. In *Proc. of Mobicom*, pages 284–298, New York, NY, USA, 2005.

[43] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04, pages 144–158, New York, NY, USA, 2004. ACM.

[44] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, Sept 2010.

[45] M. Li, W. Cheng, K. Liu, Y. He, X. Li, and X. Liao. Sweep coverage with mobile sensors. *Mobile Computing, IEEE Transactions on*, 10(11):1534–1545, 2011.

[46] M. Li and Y. Liu. Rendered path: Range-free localization in anisotropic sensor networks with holes. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, pages 51–62, New York, NY, USA, 2007. ACM.

[47] B. Liu, O. Dousse, P. Nain, and D. Towsley. Dynamic coverage of mobile sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 24(2):301–311, Feb 2013.

[48] H. Liu, S. Hu, W. Zheng, Z. Xie, S. Wang, P. Hui, and T. Abdelzaher. Efficient 3G budget utilization in mobile participatory sensing applications. In *INFOCOM, 2013 Proceedings IEEE*, pages 1411–1419, April 2013.

[49] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal Voronoi tessellation— energy smoothness and fast computation. *ACM Transactions on Graphics*, 28(4), August 2009.

[50] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[51] D. M. Y. Louis A. Hageman. *Applied iterative methods.* Academic Pressl, 1981.

[52] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. SoundSense: Scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, pages 165–178, New York, NY, USA, 2009. ACM.

[53] N. Malpani, J. L. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th International Workshop on Discrete*

Algorithms and Methods for Mobile Computing and Communications, DIALM '00, pages 96–103, New York, NY, USA, 2000. ACM.

[54] J. Nocedal and S. Wright. *Numerical optimizaion.* Springer, 2nd edition, 2006.

[55] A. Pantelopoulos and N. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(1):1–12, Jan 2010.

[56] V. F. Qiang Du and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[57] K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. Rentfrow. METIS: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications. In *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*, pages 85–93, March 2013.

[58] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-phone: An end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 105–116, New York, NY, USA, 2010. ACM.

[59] T. S. Rappaport. *WIreless communications, principles and practice.* Prentice Hall, 1996.

[60] N. Ristanovic, J.-Y. Le Boudec, A. Chaintreau, and V. Erramilli. Energy efficient offloading of 3G networks. In *Proceedings of the 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, MASS '11, pages 202–211, Washington, DC, USA, 2011. IEEE Computer Society.

[61] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14:877–898, 1976.

[62] M. Ryan. Exploiting subgraph structure in multi-robot path planning. *J. Artificial Intelligence Research*, 31:497–542, 2008.

[63] A. Saipulla, C. Westphal, B. Liu, and J. Wang. Barrier coverage of line-based deployed wireless sensor networks. In *INFOCOM 2009, IEEE*, pages 127–135, 2009.

[64] R. Schnabel and E. Eskow. A revised modified Cholesky factorization algorithm. *SIAM Journal on Optimization*, 9(4):1135–1148, 1999.

[65] X. Sheng, J. Tang, and W. Zhang. Energy-efficient collaborative sensing with mobile phones. In *INFOCOM, 2012 Proceedings IEEE*, pages 1916–1924, March 2012.

[66] O. Shental, P. H. Siegel, J. K. Wolf, D. Bickson, and D. Dolev. Gaussian belief propagation solver for systems of linear equations. In *Proc. the IEEE International Symposium on Information Theory*, pages 1863–1867, August 2008.

[67] G. T. Sibley, M. H. Rahimi, and G. S. Sukhatme. Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks. In *Proc. the IEEE ICRA*, pages 1143–1148, May 2002.

[68] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[69] P. Svestka and M. Overmars. Coordinated path planning for multiple robots. *Robotics and Autonomous Systems*, 23:125–152, 1998.

[70] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pages 350–360, Oct 2004.

[71] G. Wang, G. Cao, and T. F. L. Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5(6):640–652, June 2006.

[72] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 28–39, New York, NY, USA, 2003. ACM.

[73] Y. Wang, C. Hu, and Y. Tseng. Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks. In *Proc. of WICON*, 2005.

[74] H. Weinschrott, F. Durr, and K. Rothermel. StreamShaper: Coordination algorithms for participatory mobile urban sensing. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 195–204, Nov 2010.

[75] X. Xu and S. Sahni. Approximation algorithms for sensor deployment. *Computers, IEEE Transactions on*, 56(12):1681–1695, 2007.

[76] D. Yang, G. Xue, X. Fang, and J. Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 173–184, New York, NY, USA, 2012. ACM.

[77] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Netw.*, 6(4):621–655, June 2008.

[78] S. Yu and Y. Zhang. R-Sentry: providing continuous sensor services against random node failures. In *Proc. of 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2007.

[79] H. Zhang and J. Hou. Maintaining sensing coverage and connectivity in large sensor networks. In *NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wirelsss, and Peer-to-Peer Networks*, 2004.

[80] Q. Zhao and M. Gurusamy. Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 16(6):1378–1391, Dec. 2008.

[81] Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. In *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*, pages 373–378, Oct 2004.

[82] Y. Zou and K. Chakrabarty. Sensor deployment and target localization based on virtual forces. In *Proc. IEEE Infocom*, pages 1293–1303, July 2003.