OPTIMIZATION APPROACHES TO SENSOR PLACEMENT PROBLEMS

By

Daryn Ramsden

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major Subject: MATHEMATICS

Approved by the
Examining Committee:

John Mitchell, Thesis Adviser

Kristin Bennett, Member

Joseph Ecker, Member

Bülent Yener, Member

Rensselaer Polytechnic Institute
Troy, New York

August 2009
(For Graduation December 2009)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

It wasn't so long ago that I had convinced myself that I would be comfortable if I never finished this dissertation. The events of the interim have been consistently surreal and occasionally harrowing. Fortunately, circumstances had already dictated that I be surrounded by an extraordinary number of good people who wouldn't let me quit.

Foremost among these people is my advisor, Professor John Mitchell. I will always be grateful for the time and energy he has invested in me and my work. Thanks also to the other members of my committee, Professors Kristin Bennett, Bülent Yener and Joseph Ecker for their input.

Thanks to all the friends I made at RPI: Kara, Chris, Matt, Pablo, Billy, Kegan, Gautam, Taylor, John, Kate, Charlie, Charlie, Tara, Christina, Juan, Adnan, Alok, Dimitre, Warren, Kai, Dawnmarie, Martha, Sam, all my intramural soccer teammates, anyone I ever split a rent check with, my many officemates, everyone at WRPI and numerous others. I wasn't in any way prepared to have such a good time while at grad school.

Thanks also to Joe Hetko, Mario Farina, Ann Merante as well as my friends at Ryan's Wake, the Ruck and Positively 4th Street for your invaluable friendship.

I'm grateful to Sam Burer, who was always willing to help with my understanding of his software and to my friend Stephen Kelley, who was always willing to share any computing resources he had available to him.

To my sisters, Danielle and Daleesa, my cousin Joel, the rest of my extended family and many of my friends at home I am also grateful.

But mostly, I'd like to dedicate this thesis to my parents who have always wanted more for me than I knew well enough to want for myself.

ABSTRACT

Recent technological advances have facilitated the widespread use of wireless sensor networks in many applications. Due to battery life concerns it is often of great importance that the network configurations in use minimize energy consumption while meeting some appropriate quality of service threshold. We develop a framework for solving sensor placement problems that separates the solution procedure from geometric or other concerns that may exist in specific problem instances. After an implementation-specific pre-processing stage, coverage problems are reduced to simple combinatorial optimization problems. Further, we develop two approaches for determining solutions for the coverage connectivity problem: the first a cutting-plane algorithm for single connectivity and the second a mixed integer semidefinite formulation of the problem which allows for the solution of the coverage of the coverage-connectivity problem for higher degrees of connectivity. Lastly, we do a brief investigation into the prospect of network lifetime maximization using the feasible solutions developed by these algorithms.

CHAPTER 1
Introduction

1.1 Background: Wireless Sensor Networks

Often, it is desirable that we be able to establish and maintain a certain level of awareness of the objects contained and events taking place in a region. In some scenarios we would like to be notified if a previously identified resource or object is moved or disabled, in other scenarios we would like to be notified of an unexpected/unauthorized intrusion and in yet others we may have an interest in reliably detecting an event.

For example it is of significant advantage that we detect forest fires as close to their origin, in both the spatial and temporal contexts, as possible. The cost of firefighting, the loss of valuable natural resources, and the threat to life and property can combine to debilating effect. The potential contamination of a municipality's water supply serves as another example of an event which would ideally be identified as early as possible. These are but two instances and other applications abound in varied fields including biological detection, environmental monitoring, battlefield surveillance [17], microclimate monitoring, the study of animal populations and many others.

Recent and continuing technological advances have led to the development of wireless sensor network technology to the extent that it can be and has been employed to address the requirements of many of these situations. Some of the important advances facilitating this are in wireless communication and Micro Electrical Mechanical Systems [19].

Wireless sensor networks are composed of numerous sensors and each of these sensors, as the name implies, has the ability to notify or respond to events within their field of sensing. In keeping with the varied nature of the problems the types of devices that come under the umbrella term sensor are widely varied in their make-up. To illustrate, sensing may be done via surveillance cameras, the reception of a radio transmission, the use of infrared technology or by analyzing the contents of a

chemical sample. Despite this diversity, it can be said that the acquisition of these sensors is an increasingly low-cost venture.

The use of these sensors is convenient and comes with benefits: they can be left unattended and therefore they can be used in environments that would be considered dangerous or otherwise unappealing to human beings [51], and in other situations, such as those in which insight into natural phenomena is being chased, they may be prized for the ability to carry out the required surveillance unobtrusively [30, 41].

Whatever the specific means of gathering information, the sensor can, for many purposes, be considered to consist of three modules: one that enables it to carry out the sensing that it gains its name from, one that transmits gathered information and one that serves as an energy supply.

It is this last module that ultimately makes the study of wireless sensor networks interesting. Even as the technology behind the sensing and communicating serves to make rapid and ongoing upgrades in these capabilities, progress in the area of increasing battery capacity is slow and by all indications will remain as such for the foreseeable future [51]. This power scarcity is the driving concern behind the quest for good sensor placement schemes. The continual replacement of these batteries isn't a very practical solution for the very reasons, already mentioned, that we would wish to use wireless sensors in the first place. As such we need to take great care in deciding how to utilize the sensors at our disposal.

The dual goals of any sensor network implementation would be that the network be operational for a sufficiently long period of time and that the quality of service it provides be up to an application-appropriate standard. While the respective weights placed on these considerations may vary from setting to setting they are both important. A sensor network with great longevity is of little practical use if it cannot reliably provide good information in any given time period. Similarly, a sensor network that is only operational briefly has very little possibility of being effective even if the quality of service in that brief period is stellar.

The most basic means of saving energy consists of turning sensors off or, in some applications, reducing the power level in exchange for a reduction in functionality. In some applications it isn't practical to turn sensors off and we have to resort

to a sleep-state in which power is consumed at a drastically lower level. In most cases, this sleep-state power level is a sufficiently small fraction of that of the active sensor that, for modeling purposes, it is often reasonable to treat sleeping sensors as being off.

1.1.1 Problem Requirements

1.1.1.1 Primary Concerns: Coverage and Connectivity

As mentioned before, a key component of the sensor's functionality is the ability to report on the things that it has detected. A sensor that didn't wouldn't be useful. It turns out that the mode by which the sensor makes this information accessible is a key determinant in the nature of the problems we have to solve. In some situations all of the sensors can be assumed to have available to them some inexpensive channel of communication with a central intelligence. Or perhaps, the sensor might be bundled with another device and the information gathered is acted upon at or near the sensor without having to be relayed elsewhere. In situations such as these we are concerned only that we have an appropriate quality of coverage of the region or targets. The problems that result are known as *coverage* problems. As has been widely discussed, coverage problems are related to the *Art Gallery Problem* [15] in which the objective is to use as few guards as possible to secure an art gallery. In other situations, we may desire (possibly multi-hop) communication between sensors as it may be important to keep sensors located elsewhere aware notified of the current state, possibly for the purpose of being vigilant about possible future events in their own vicinity. Alternatively, the sensors may use each other to relay information gathered to a sink node at which higher levels of processing may take place. In situations like these we are concerned not only that all the appropriate targets are covered but that the sensors that are activated be able to communicate with each other. The family of problems yielded by these and similar situations are given the term *coverage-connectivity* problems. Coverage can be seen to be a more primal concern than connectivity because every situation in which we desire that sensors be connected has as its foundation a set of targets that necessarily have to be covered. Problems involving connectivity increase the computational burden

notably.

1.1.1.2 Secondary Concern: Lifetime Maximization

In addition to the desire that the quality of service be up to an application-appropriate standard there is the complementary objective that the network have as prolonged a lifetime as possible. While the respective weights placed on these considerations may vary from setting to setting they are both important. A sensor network with great longevity is of little practical use if it cannot reliably provide good information in any given time period. Similarly, a sensor network that is only operational briefly has very little possibility of being effective even if the quality of service in that brief period is stellar.

Whereas in the short term, energy-saving efforts may consist of finding configurations that provide the appropriate level of service at minimum cost, when we have the long term in mind the situation may dictate that we alternate between different configurations, as a means of prolonging the lifetime of the network[6].

Another viewpoint that we can take when thinking about extending the lifetime of the network is that we should minimize the network's susceptibility of the individual sensors. Perhaps we should seek configurations that de-emphasize cost minimization somewhat, placing some of the displaced emphasis on robustness. The natural way to counteract the possibility that sensors can and will fail from time to time is to require certain levels of redundancy. For example, it might be required that each target be covered by more than one sensor. Or that there be more than one path between any two sensors. When we require that at least k sensors cover each target that is referred to as k-coverage. When we require that at least k paths exist between two sensors this is known as k-connectivity.

Incidentally, in actual applications building redundancy into the network not only improves the network's fault-tolerance but the quality of service also. In practice sensors aren't infallible and at best can only record events within their alleged range with high probability[53]. Having multiple sensors covering a target serves to increase this probability of detection.

1.2 Prior Work

Previous work on coverage and coverage-connectivity problems can be partitioned into centralized and distributed approaches. A distributed algorithm is one in which the decision-making is done locally: the choice to activate any given sensor hinges only on the behaviour of a set of sensors that have been defined to be in the same locality as that sensor. In a centralized approach, all the information available is utilized and decisions are made with their effect on the entire system in mind. Centralized problem formulations allow us to clearly specify an overriding objective but do come with a noticeable drawback: they don't scale up very well and consequently the quest to satisfy the specified objective is a difficult one.

Previous research reflects the wide variety of sensor types and applications and in keeping with this it is the case that the sensor placement problems studied in the literature show great variation. Some approaches work by isolating the worst-covered regions of a given area while others place their focus on finding the areas that are best-covered. In some situations the sensors are immovable and in others they are highly mobile. In fact, in problems there is some sort of hybrid: maybe the sensors can move once (or a finite) number of times after being deployed or in other situations some of the sensors are mobile while others are not. All of these variations serve as proxies for significant situations in the physical realm.

When it comes to connectivity the considerations can also vary subtly and the relevant research follows suit. Some care only that a network is connected while others concern themselves with the cost of transmission or the number of possible routes that information transmission may take or how well the integrity of the data is maintained over the course of transmission.

All of that being said, we endeavor to provide somewhat of a synopsis of the literature.

1.2.1 Coverage Problems

As early as [17] three types of coverage behaviours have been identified for sensor networks: blanket coverage, sweep coverage and barrier coverage. Barrier coverage seeks a static configuration of nodes that maximizes the probability that

intrusion through a barrier is identified correctly and recent recent work in the area can be found in [11, 10, 32]. In sweep coverage the nodes are being moved across the sensing area with the dual goals of maximizing the probability of event detection while minimizing the possibility that events in any one particular area go undetected. An example of sweep coverage can be found in [18] where the focus is on the detection of landmines.

In blanket coverage the aim was defined to be a static arrangement of elements that maximizes the detection rate of targets appearing within the coverage area. This is where our focus will lie.

Even when confining ourselves to the study of blanket coverage there are varied paradigms through which we can quantify the quality of service. Some characterizations of coverage quality are predicated on determining which regions of the sensing field are worst-covered while others focus on detecting the regions that are best-covered.

As a preliminary to discussing different coverage algorithms it is worthwhile to discuss the different sensing models that are in circulation. The most popular of these is the disk model, in which it is assumed that the area that a sensor can cover is isotropic and centered about the sensor's own location. In two dimensions this corresponds to a circle and in three to a sphere. The sensor has a specified coverage radius and all elements within this threshold distance of the sensor are assumed to be covered unless some externality is involved, perhaps an obstacle. In contrast, [31] is an example of a work that allows for the possibility of arbitrarily shaped sensing region. As noted above however, all target/event detection is probabilistic in nature and when we speak of a target being covered we mean that the probability of a noteworthy event being detected meets some application-specific standard.

In practice, this probability of sensing tends to deteriorate at a rate that grows faster than the distance from the sensor [34]. The use of the disk model is tantamount to setting a strict minimum requirement on the likelihood that a point is detected by any given sensor. This approach ignores the fact that a target may be covered poorly by any one sensor but still have a relatively high probability of detection by a group.

Approaches to the problem of coverage can be divided into two categories: centralized and distributed. In centralized approaches, decisions are made with knowledge of the entire network in mind with the goal of extremizing some objective defined as a function of all the nodes. In distributed algorithms on the other hand, decisions are made at each node regarding its own behaviour and the information considered is restricted to that related to nearby nodes (and perhaps a fixed number of other nodes).

Centralized approaches to solving coverage problems are by no means new, with key early contributions in [35] and [8]. In [35] the coverage problem is analyzed from deterministic, statistical, worst and best case perspectives and polynomial-time algorithms are supplied that solve the coverage problem making use of such concepts from computational geometry as Voronoi diagrams and Delaunay triangulations [38, 15]. It is to be noted that computing Voronoi diagrams and Delaunay triangulation is not an easy task and cannot be taken for granted. In [8], the grid-based coverage problem is formulated as an integer linear program. Like many other methods, the assumption is made that sensors have isotropic sensing regions. Distances between locations are input parameters of the integer program. Determination of whether or not a target is in range of a given sensor is done via constraints added to the model which utilize the values of these parameters.

Due to the computational issues that accompany centralized methods the brunt of recent research of the coverage problem has been geared toward the development of good distributed methods. [42] outlines a node-scheduling scheme that uses a rules-based approach to determine when and for how long a sensor is asleep. This is built upon by [25] in which the relationship between sleep schedules and the need for redundancy in the network is explored in more depth. Using the insights gained from both analysis and simulation, [25] is able to provide an improved scheduling scheme that provides continuous coverage with noticeably less expenditure of energy.

More recently, some schemes have been developed to deal with the possibility that some or all of the sensors in the network may possess the ability to relocate. In some scenarios the cost of relocation may be relatively cheap, while in others they might be nonnegligible. If the cost of moving a sensor is sufficiently low this

situation might be treated as being equivalent to one in which we have free choice in choosing the sensor location in the first place. [52] details three different algorithms, VEC, VOR, and MiniMax, which are all distributed algorithms for mobile sensor networks. These algorithms involve the use of Voronoi diagrams. In situations where the cost of using the sensor cannot be reasonably ignored some, notably [44], formulate protocols which allow for hybrid networks in which some sensors are potentially mobile and others are not.

1.2.2 Problems involving connectivity

For blanket coverage of a contiguous domain, it has long been known that if the communication radius of each sensor doubles the coverage radius then k-coverage implies k-connectivity [45]. However, this assumption is not widely applied and as such much work continues to be done toward the establishment of methods that ensure both coverage and connectivity.

To discuss wireless sensor network problems with connectivity requirements we should start by reviewing the different communication model, the most commonplace of which is the binary-disk model. This model has its roots in the assumption that a sensor's field of communication, like its field of coverage, is isotropic. Two sensors are considered to be in communication with each other if they lie within each other's communication field. That is to say that sensors can communicate if the distance between them is bounded above by the smaller of their communication radii.

Just as with the coverage problem, distributed approaches predominate the attempts to find solutions to the problem of having integrated coverage and connectivity. Just as there are protocols that achieve coverage without any assurances regarding connectivity, there are some that achieve connectivity without guaranteeing coverage [7, 49, 50, 12]. The SPAN algorithm formulated in [12] is then utilized in [48] in conjunction with the coverage maintaining protocol CCP [48] to ensure simultaneous coverage and connectivity.

[1] presents a pattern-based approach to sensor placement that guarantees coverage and k-connectivity for $k \leq 4$ that is appropriate for applications in which we have full choice of sensor locations.

Some methods, such as [28] utilize pattern-based deployment of sensors while others such as [26, 33] specify protocols that employ sleep-scheduling and the adjustment of sensors' power levels (and thus communication and coverage ranges) to ensure simultaneous coverage and connectivity.

1.2.3 Lifetime Maximization

While early focus of sensor placement problems was on finding minimum energy configurations, increasing emphasis is being placed on maximizing the lifetime of the network. Finding a single configuration that meets coverage and connectivity thresholds while minimizing total power consumption has been found to have several notable flaws. First, it is sometimes the case that the configuration with the minimum total power consumption is unbalanced in the sense that a few of the sensors are using disproportionately large fractions of the total energy. The natural consequence of this is that these sensors have their batteries depleted relatively early in the network's lifetime with debilitating effect on the rest of the network. Some work is now focused on finding configurations that are more balanced in their levels of power consumption. Other approaches aren't satisfied with finding a single good, or even very good, configuration. If a large number of acceptable configurations are available perhaps the networks's lifetime can be extended by alternating between multiple of these configurations. If replacing multiple batteries isn't significantly more difficult than replacing a single one then it makes sense to deplete as many sensors as possible before going in and renewing the batteries. Preliminary attempts at prolonging network lifetime often involved finding as many disjoint configurations as possible. However, it is often the case that alternating between configurations that aren't necessarily disjoint can have significant impact on prolonging the lifetime of the network. Assuming that a large collection of feasible solutions is available the lifetime maximization problem can be formulated as an linear program [6]. [24] goes one step further and proposes the use of delayed column generation to possibly unearth new feasible configurations that weren't available at the start of the lifetime maximization procedure.

1.2.4 Other Similar Work

Integer programming approaches have also been taken to address similar problems in which network connectivity is an issue. In [23] a cutting plane method is presented as a tool in designing survivable fiber optic communications networks. In these problems it is of importance that for certain pairs of nodes in the network there be a guaranteed path of communication even in the face of a reasonable amount of network failure. [40] subsequently does a study of the multicommodity suvivable network design problem and derives several valid and useful inequalities for the polyhedron of capacity design variables. [14] is a followup to [40] and presents a fully fleshed out cutting plane algorithm for the solving the same problem. The generalized Steiner tree is a closely related problem in which each subset of the nodes in a graph is assigned a value which serves as the lower bound on the number of edges that must exist between that subset and its complement. With this constraint in mind, the objective is to choose a minimum cost subgraph. [29] and [46] provide approximation algorithms to solve this problem and a study of the nature of the feasible polytope in [39]. Lifetime maximization is reminiscent of the task of packing generalized Steiner trees. In this problem the goal is to find multiple generalized Steiner trees which are vertex disjoint. [21] is the source of a good computational approach and [22] investigates the polyhedral properties involved.

1.3 Linear and Integer Programming

Linear Optimization concerns itself with the optimization of a linear function of one or more variables in cases where the set of allowable variable values is determined by a group of linear constraints. A general linear program with n variables and m constraints can be expressed as follows:

$$\begin{array}{lrcccccccl} \text{min} & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & & \\ \text{subject to} & a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \geq & b_1 \\ & \vdots & & & & & & \vdots & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & \geq & b_m \end{array}$$

We can make the formulation more convenient and concise by imposing the

requirement that the x variables be nonnegative and letting:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \text{ and } A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

This leads to a common formulation

$$\begin{array}{rrcll} \min & c^T x & & & \\ \text{subject to} & Ax & \geq & b & \qquad (LP) \\ & x & \geq & 0 & \end{array}$$

Associated with every LP is its dual problem. The dual of the standard-form problem above has the form shown below where A, b and c retain their meaning

$$\begin{array}{rrcll} \max & b^T y & & & \\ \text{subject to} & A^T y & \leq & c & \qquad (LD) \\ & y & \geq & 0 & \end{array}$$

For convenience, we will sometimes refer to the feasible regions of LP and LD as X and Y respectively. It is known that for any $x \in X$ and $y \in Y$, $b^T y \leq c^T x$ and further that if each of LP and LD have a feasible solution then they have the same optimal value. Feasible solutions that may not be optimal can still provide useful bounds for the optimal solution to the primal problem.

(Linear) Discrete Optimization offers another wrinkle in that some of the variables can only be assigned values from a discrete set (often a set of integers). This constraint is very realistic and often follows easily from the nature of the problem being modeled: one wouldn't assign 2.5 people to do a job or manufacture 0.4 pianos for example.

With the introduction of an index set I representing the collection of indices corresponding to x variables that are additionally restricted to be integer-valued we get the following formulation of a mixed integer program:

$$\begin{array}{rrcll} \min & c^T x & & & \\ \text{subject to} & Ax & \geq & b & (MIP) \\ & x & \geq & 0 & \\ & x_i & \in & \mathbb{Z}_+, \forall i \in I & \end{array}$$

1.3.1 Approaches to solving Integer Programming Problems

We deal with two approaches to solving discrete optimization problems: cutting-plane methods and branch-and-bound methods. Nothing prevents these two methods from being hybridized and they can be used to complement each other. For a more detailed treatment see [36, 47].

1.3.1.1 Cutting-plane methods

An essential concept in understanding cutting-plane algorithms is that of a *relaxation* of a mathematical program. For a given MIP we can form relaxations of the problem by doing at least one of the following: 1) expanding the feasible region to become a superset of the original or 2) using a new objective function which is bounded above by the original objective function. Due to the fact that every feasible solution of the original MIP is a feasible solution of the relaxation we can say that the optimal value of any given relaxation is at least as small as the optimal value of the original problem i.e. the optimal solutions to the relaxations provide a lower bound on the optimal solution of the underlying MIP. One fundamental approach to solving mixed-integer programs is to seek out relaxations of the appropriate problem that simultaneously meet the following criteria 1) the relaxation is easier to solve and 2) the solution of the relaxation will provide insight into the solution of the MIP (in some cases, we can also reasonably hope that the relaxation has its optimal solution at a point that is feasible for the original problem).

Due to the fact that integrality constraints provide large algorithmic obstacles to the problem-solving process, a ubiquitous approach to forming relaxations is to remove some or all of the requirements that decision variables be integer.

Cutting-plane methods are an iterative framework for solving Integer Program-

ming problems in which after each solving of a relaxation we seek linear constraints that are valid for the underlying IP that are violated by the solution of the current relaxation. These constraints are added to form a new relaxation which is then solved. The process is repeated until the solution of the current relaxation is a member of the feasible region of the original MIP

Data: A, b, c, I

Result: $x^* = \text{argmin}\{c^T x : x \in X\}$

Let $X_r \supset X$;

Let $x_r = \text{argmin}\{c^T x : x \in X_r\}$;

while $x_r \notin X$ **do**

  Find A_*, b_* s.t. $A_* x \leq b_* \forall x \in X \text{and} A_* x_r > b_*$;

  Update $X_r = X_r \cap \{x : A_* x \leq b_*\}$;

  $x_r = \text{argmin}\{c^T x : x \in X_r\}$;

end

$x^* = x_r$

Algorithm 1: Cutting-plane algorithm framework

1.3.1.2 Branch and Bound Methods

The idea behind branch and bound algorithms is to partition the feasible region of the MIP and solve easier problems over members of the partition. Explicitly optimizing over each of the members of the partition is very intimidating computationally and doesn't work very well as problem sizes get larger. However, it is often possible to avoid much of this work.

The partitioning of the feasible region at any given point in the procedure is known as branching, thus contributing to the name of the method as well as its effectiveness. For a standard minimization problem, we seek a lower bound on the optimal solution of any given branch and this can be found by solving a relaxation of the problem at that branch. There are a few possible scenarios

- The lower bound found for the current branch has a higher value than that of a feasible solution found previously. In this case we can be sure that the optimal solution is not to be found in the current branch.

- The relaxation yields a solution that is feasible for the integer program and provides the smallest-valued optimal solution to the integer program found to date. In this case, we have a candidate for the optimal solution
- The relaxation yields a lower bound that is not provided by a feasible solution but is smaller in value than any of the objective values associated with feasible solutions previously found. In this case, we make an effort to investigate this branch further.

The solving at any given branch may make use of any solution technique available to the implementer(s) and may even include branch and bound itself. Implementation-specific features involve the method used to solve problems at different branches and making decisions regarding which branches should be given priority at any given point in time.

1.4 Semidefinite Programming

Semidefinite Programming is a generalization of Linear Programming in which some of the decision variables are not regarded as the components of a vector but rather as the elements of a positive semidefinite matrix.

We can formulate a general form for primal and dual semidefinite program that bear some resemblance to the general forms LP and LD established earlier.

$$\begin{aligned} \min \quad & C \bullet X & & & \\ \text{subject to} \quad & A_i \bullet X & \geq & b_i & \qquad (SDP) \\ & X & \succeq & 0 & \end{aligned}$$

$$\begin{aligned} \max \quad & b^T y & & & \\ \text{subject to} \quad & \sum_i y_i A_i & \preceq & C & \qquad (SDD) \\ & y & \geq & 0 & \end{aligned}$$

Here $U \bullet V = \text{trace}(U^T V) = \sum_{i,j} U_{i,j} V_{i,j}$ and the notation $U \succeq 0$ is used to signify that U is positive semidefinite.

These programs would both be linear were it not for the requirements that X be positive semidefinite. In fact Linear programming can be recognized as a special case of semidefinite programming in which the off-diagonal entries of X are set to be zero. In such a scenario the semidefiniteness constraint is equivalent to the nonnegativity requirement on the x variables in LP. In fact, it is only the semidefiniteness constraint that distinguishes the general SDP from a linear program. It is to be noted that this constraint allows for the off-diagonal entries of the matrix X to be negative and we will take advantage of this later on. SDP is convex and can be solved in polynomial time.

For a more complete review of Semidefinite Optimization see [43].

1.5 A Few Relevant Preliminaries from Graph Theory

An intuitive model for a connected wireless network is that of a graph and some of our work to follow in subsequent chapters implements this approach and capitalizes on prior results in the field of graph theory. As such, we now present a very brief introduction to graph theory with particular emphasis on results and terminology that happen to be germane.

Definition 1. *A graph is a pair of disjoint sets (V, E) satisfying $E \subset [V]^2$, i.e. the members of E are subsets of V of cardinality 2.*

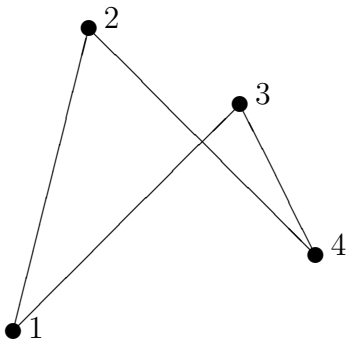
The standard method of visualizing graphs is via the use of dots and lines. The members of V, (which are referred to as vertices, nodes and points), are usually represented as dots. The members of E are represented as lines joining the relevant members of V. In the graph pictured in figure 1.1 for example we would have the following V and E

$$V = \{1, 2, 3, 4\} \quad E = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 4\}\}$$

Definition 2. *A path is a nonempty graph (V, E) of the form*

$$V = \{x_0, x_1, \ldots x_k\} \quad E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$$

Figure 1.1: An example graph



where all the x_i are distinct.

Definition 3. *Two points are said to be connected if there exists a path containing both of these points.*

Definition 4. *A subgraph is said to be connected if any two of its members are connected.*

Definition 5. *A connected subgraph that would lose its connectivity with the addition of any other vertex is called a component.*

Starting with the idea of connectivity, we can then tackle the question of quantifying connectivity where it exists. Perhaps a component that possesses more than a single path between any two points should be regarded as being better-connected. Or maybe the connectivity of a component can be measured by how many edges or vertices can be removed from the graph before it is no longer connected.

Definition 6. *A vertex cut of a graph is a set of vertices whose removal renders the graph disconnected.*

Definition 7. *A component is said to be* k-connected *if there exist k disjoint paths between any two members of the component.*

Definition 8. *A graph is said to have a vertex connectivity κ if at least κ vertices have to be removed from the graph for the graph to become disconnected.*

[16] and [3] provide comprehensive introductions to graph theory.

1.6 Tools

Much of the work is implemented in C and makes use of the ILOG CPLEX callable libraries[27], a library that allows for the interaction with the CPLEX mixed-integer programming solver. In our code, we generate a problem then utilize our algorithm for solving the problem.

The callable libraries allow for the use of CPLEX as a black-box solver to solve the intermediate relaxations of the problem.

Another third-party optimization package we work with is SDPLR[5] which allows us to solve semidefinite programming problems via an implentation of the algorithm presented in [4].

It is our goal to apply Linear and Integer Programming techniques to tackle as many of these sensor placement problems as is feasible. Sensor placement problems have long been formulated as optimization problems [8] however there is relatively little attempt to increase the size of problems that can be solved when formulated in this manner.

CHAPTER 2
Coverage Problems

As has been previously noted in the literature, the term *coverage* has been associated with a wide variety of interpretations commensurate with variation in sensor models and application. A workable conceptualization of the term is as a measure of how much faith can be placed in the sensor network to accurately detect the events taking place within the confines of the region being monitored. A sensor network that is consistently accurate in its detection of relevant events can be considered to be providing a high level of coverage.

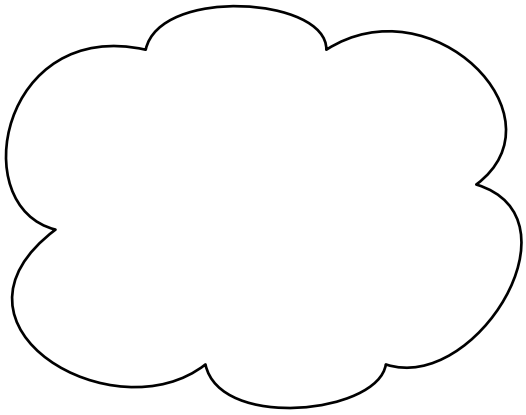
Here we seek to tackle two of the three main types of coverage problems: area coverage and discrete target coverage. The third major type of coverage is boundary coverage. Area coverage, as the name might suggest, asks that the region be monitored in its entirety. Discrete target coverage asks that we monitor a set of discrete points that have already been selected as being of some importance: possibly due to extra information indicating that interesting events are more likely to happen at or near these locations.

The approach that we will take to the area coverage problem will actually be to model the problem as a discrete target problem. The idea being that given a field to cover we can pick a discrete set of points in the region as targets. The efficacy of using this discrete set as a surrogate for the entire field is related to whether or not the discrete points are sufficiently numerous and diffuse. If the maximum distance between any point on the field to one of the target locations is relatively small then we expect the solutions arrived at to be good solutions to the area coverage problem.

For example, figure 2.1 might correspond to an area that we desire to be covered.

While there is no unique way to decide which points on the map we wish to allocate this special target status, it seems reasonable and perhaps even natural to utilize some kind of regular repeated pattern. The building blocks of these patterns can conceivably be of any shape but we will rely on rectangular grids. We lay a grid

Figure 2.1: A possible region to be covered



down over the area to be covered and consider all the grid points that coincide with points within the region to be the targets. A finer mesh would represent a more accurate representation of the problem while adding to the computational burden associated with solving the problem.

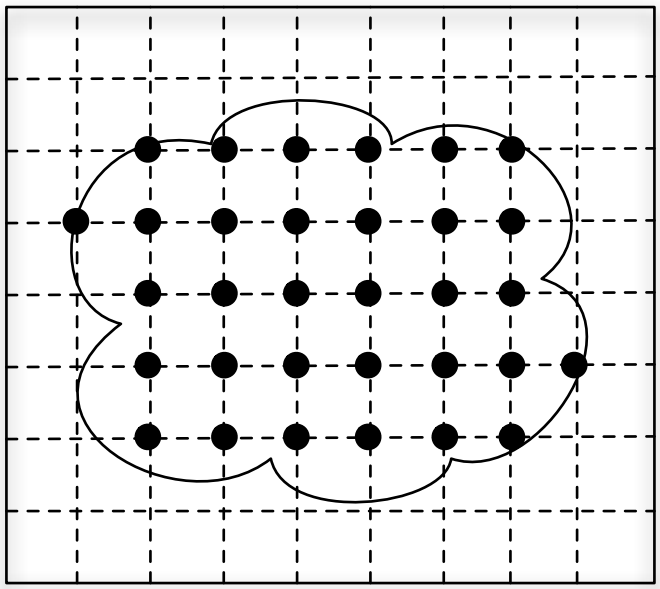
The previously illustrated map might look as in figure 2.2 after we have designated targets according to some grid-based pattern.

The diagrams that follow show a region and how it might alternatively be represented by a grid-based target selection scheme or another target selection scheme. It is conceivable that the seemingly random deployment might be the result of a design decision made by implementers with extra knowledge of the situation or it might actually random. The discrete target coverage problem would probably also feature a map in which the targets seem to be distributed in such a manner.

Figure 2.3 demonstrates how the targets might be dispersed if the target-designation process was not pattern-based.

Good coverage of a region facilitates not only the monitoring of important events but the potential tracking of moving targets. The better covered a region minimizes the probability that a target that is moving through a field can remain undetected and even potentially allows for path reconstruction if detection events are time-stamped.

Figure 2.2: Using grid-based targets to model the region



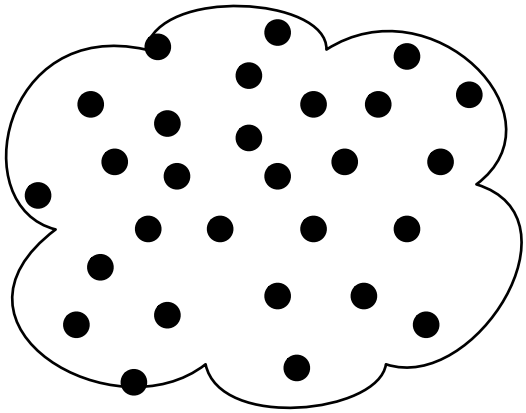
2.0.1 Formulation of the Coverage Problem

A key feature of our approach and the one that allows us to separate the main portions of our algorithms from the realm of geometry is a pre-processing stage in which we collect lists of sensors that cover each target. This pre-processing is of great benefit to us as the constraints generated in the intermediate stages of our algorithms (both for this problem and the coverage-connectivity problem) are heavily dependent on having this information accessible and as such it is advantageous to only have to do those particular computations only once.

The fact that we don't have any interaction with geometric concerns after this pre-processing stage makes this approach very flexible. To the extent that they possess similar numbers of sensors and targets, problems based in three-dimensional contexts aren't more difficult than their two-dimensional counterparts; we are well equipped to deal with obstacles and other irregularities that would seem to otherwise undermine the effectiveness of our traditional models of sensing regions; the window is left open to deal with various types of sensing regions.

We formally define coverage as appropriate for the rest of this work.

Figure 2.3: Using randomly chosen targets to model the region



Definition 9. *Let t be a target location in the sensing field and s be a sensor. s is said to* cover *t if s can reliably be expected to detect the events taking place at t.*

Definition 10. *Let t be a target location in the sensing field and S be a set of sensors. S is said to* cover *t if at least one member of S covers t.*

Definition 11. *Let t be a target location in the sensing field. t is said to be* k-covered *if k or more sensors cover t.*

For ease of notation, we define essential problem parameters as follows:

Parameter	Definition
n_s	number of possible sensor locations
n_t	number of targets
m	number of sensors required for each target
t	number of sensor types available
$R_S(i)$	radius of sensitivity of sensor type i
$R_T(i)$	radius of transmission of sensor type i
$R(i)$	radius of operation of sensor type i when $R_S(i) = R_T(i)$
R	radius of operation when all sensors are identical

Using x_i to denote an indicator variable of whether or not sensor i is activated, we shall give names to the sets of sensors that can possibly cover each target

$$cov(i) = \{x : \text{target i is covered by sensor x}\}$$

As for constraints, what we need now is that sufficiently many members of $cov(i)$ be activated for each target i. Exactly how many of these are needed follows directly from the degree of coverage desired. By the definition of k-coverage we arrive at constraints for the coverage problem

$$\sum_{i \in cov(i)} x_i \geq k$$

The appropriate objective function would now seek to minimize the total cost of activating sensors and that leaves us with the following integer linear program for the coverage problem. Using c_i to denote the cost of using sensor i, we have the following formualation of the coverage problem.

$$\begin{array}{rrcll} \min & c^T x & & & \\ \text{subject to} & \sum_{i \in cov(i)} x_i & \geq & k & (COV) \\ & x_i & \in & \mathbb{Z}_+ & \end{array}$$

This is not the first time that the coverage problem has been formulated as an integer linear program and an earlier formulation can be found in [8]. In fact, covering problems such as these are standard in integer programming particularly when $k = 1$ [36]. Examples of algorithms for more general values of k can be found in [2, 9].

In [8], the model is based on an isotropic model of the sensing regions and the distances between locations are included in the formulation so as to bundle the functionality of the two phases of our approach into one phase. On the same problems solved in that earlier work our formulation provides no advantage. We anticipate that there are benefits to be reaped from our approach via the eventual extension to the coverage-connectivity problem and an enhanced ability to deal with non-isotropic sensing regions.

2.0.2 Types of coverage problem

All of the problems that we are attempting to solve are of the discrete target variety: either a problem that was originally of this form or an area coverage problem

that we are modeling as a discrete target problem. We can classify the problems we solve by the manner in which the the sensors and targets are distributed.

The most basic of the problems solved is that of a grid in which every grid point is a target as well as a potential sensor location. We will solve these problems for square regions as well as regions that are more elongated in one direction or the other. This situation models an area coverage problem. From a sensing point of view this scenario reflects a situation in which either all the points in the region are candidate sensor locations (if the grid mesh is sufficiently fine) or a situation in which sensors have already been laid out according to some regular pattern.

The first modification of this most basic problem would be to randomly discard some percentage of the points generated in an attempt to develop a somewhat more realistic map. At the least this results in a region that isn't symmetrically or regularly shaped. We can then go even further and remove some of the locations in an attempt to model an area coverage problem in which there are unsuitable locations.

We can also use grid-based targets in randomly distributed sensor locations to reflect situations in which we desire area coverage but significant control over the placement of sensors for example the previously mentioned use of aircraft for the purpose of sensor deployment.

2.1 Dealing with heterogeneity among sensors

In many cases there may be some variety in the sensors at our disposal. Sometimes different types are available or equivalently (for our purposes) individual sensors can be operated in a variety of modes. The cost of a sensor is typically a monotonically increasing function of its coverage ability but the particular rate of increase may vary. We deal with heterogeneous sensors in a manner used in [8] which is to split variables and to treat one sensor's potential to operate at different levels as different sensors.

We use $c_{u,v}$ to denote the cost of using sensor u at level v and $x_{u,v}$ to denote the corresponding decision variable. In order to prevent a configuration which features one sensor being operated in multiple modes the following constraints are employed

$$\sum_j x_{i,j} \leq 1 \quad \forall i$$

Lemma 1. *For single coverage no additional constraints are required to deal with heterogeneous sensors.*

Proof. Assuming that the optimal solution asks for the same sensor to operate at more than one power level we can reduce the objective value by de-activating the lower powered version without violating the coverage requirements for any target. Thus, we have a contradiction. □

2.1.1 Results of Problems with Heterogeneous Sensors

Figures 2.4 and 2.5 demonstrate the results of simple grid-based problems involving two types of sensors. The sensor parameters used were taken from [8]. As is noted in [8] sensors that minimize cost per target covered tend to predominate while the other type of sensor that is cheaper tends to be used for the purpose of filling in gaps in the configuration.

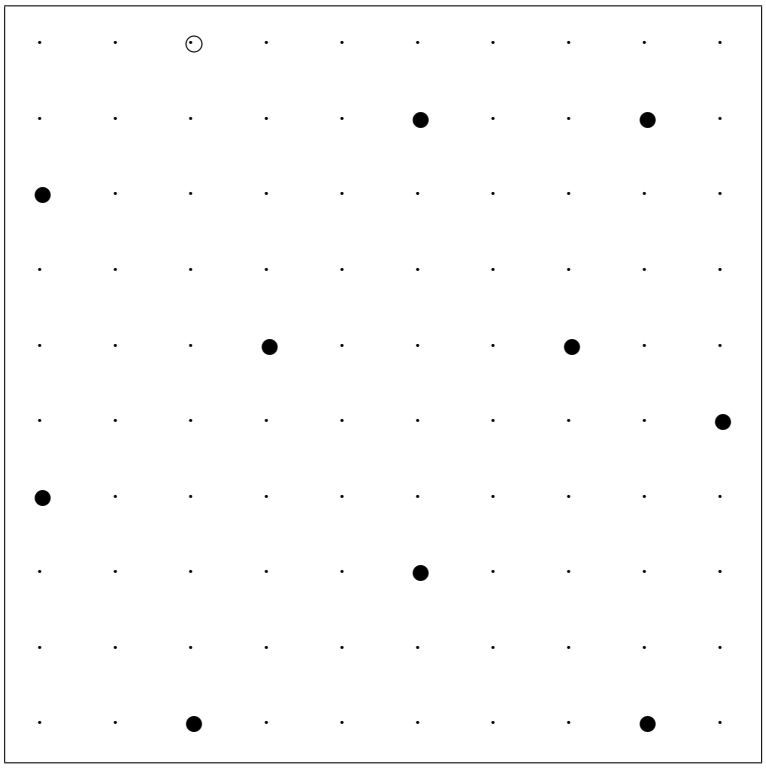
2.2 Unique Coverage

An interesting variation of the coverage problem is one in which the desired sensor network not only covers the targets cheaply but that each target is covered by a unique set of sensors. Some devices that fit under the umbrella description of 'sensor' are restricted in their functionality in that they can only report whether or not they have detected an event. If each target location is covered by a unique set of sensors we can determine the location of the event, a piece of information that one sensor alone would be able to provide, by examining the list of sensors that reported the event. Unique coverage provides a means for us means for us to perform data fusion [8].

Type a: ○ Range: 100, Cost: 150

Type b: ● Range: 200, Cost:200

Figure 2.4: Coverage of a 10x10 grid with 2 types of sensors



2.2.1 A cutting-plane algorithm for Unique Coverage

Our approach to solving the unique coverage problem is predicated on the assumption that a sensor can't choose to ignore a location that its field of perception. With this in mind, for any two targets i and j they cannot both be covered only by sensors in the intersection of the two sets $cov(i)$ and $cov(j)$.

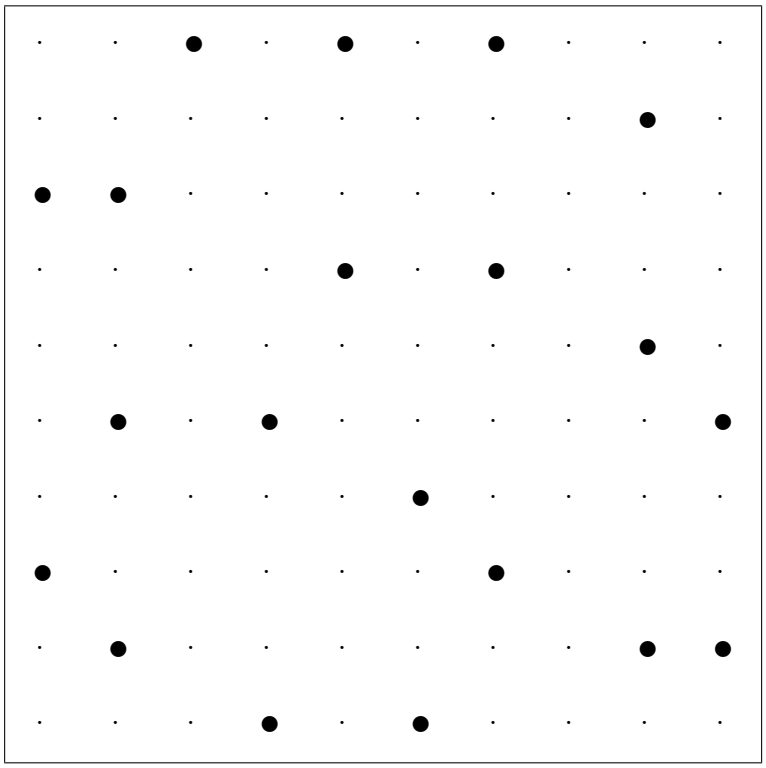
Thus the unique coverage problem can be stated as

$$
\begin{array}{rrcll}
\min & c^T x & & & \\
\text{subject to} & \sum_{i \in cov(i)} x_i & \geq & k & \\
& \sum_{i \in cov(k) \setminus cov(j) \cup cov(j) \setminus cov(k)} x_i & \geq & 1 & \forall j,k \quad \text{such that } cov(j) \cap cov(k) \neq \emptyset \\
& x_i & \in & \mathbb{Z}_+ &
\end{array}
\qquad (UCOV)
$$

Type a: ○ Range: 100, Cost: 150

Type b: ● Range: 200, Cost:200

Figure 2.5: 2-Coverage of a 10x10 grid with 2 types of sensors



Many of the constraints may be unnecessary so we solve the problem via a cutting-plane method. Initially, we solve the coverage problem. We then look for constraints that may be violated in the interim solution and add them to the model and solve again. After each solving we test for feasibility and in response to finding multiple sensors covered by the same set of sensors we add the appropriate constraints. At each iteration a tighter relaxation of the problem is solved and eventually a minimum cost unique coverage configuration is arrived at. The size of the problems that can be conveniently dealt with would be a function of problem sizes, computing resources and the urgency with which results are needed.

Figures 2.6 and 2.7 give two examples of grid-based unique coverage problems solved via the cutting plane method described above.

Figure 2.6: Unique Coverage of a 6x6 grid, m=1

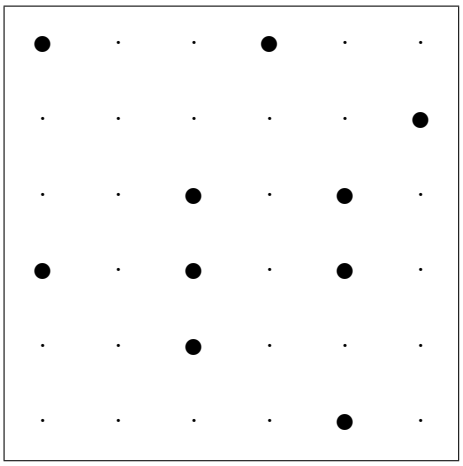
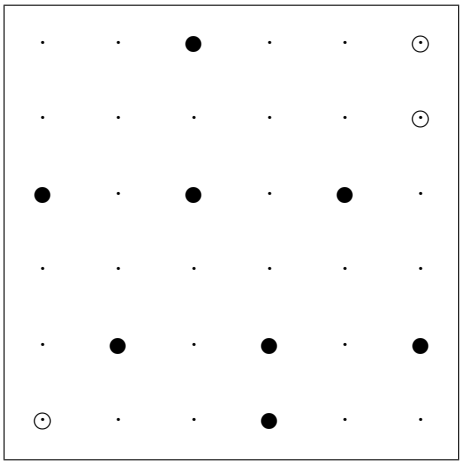


Figure 2.7: Unique Coverage of a 6x6 grid, m=2



CHAPTER 3
A cutting-plane approach to the coverage-connectivity problem.

Now, we move on to tackle the additional problem of connectivity: that there be a path facilitating communication between any pair of sensors that are chosen to be on. If we consider each sensor chosen to be on as a vertex and the possibility of communication between two sensors to be an edge this requirement can be equivalently stated as a need to have the graph induced by the set of active sensors be connected.

We employ a cutting-plane method to solve this problem: we solve relaxations of the problem; at each stage testing for connectivity and if necessary adding constraints that would be valid for any connected solution but are violated by the solution of the current relaxation.

Through inspection of the intermediate solutions it is possible to find sets of edges, all of which are currently inactive, for which we can say that at least one member of the set has to be available in a feasible solution to the coverage-connectivity problem.

3.0.2 Connectivity with m=1

The initial relaxation of our algorithm is none other than the coverage problem formulated previously which had as its objective

$$\sum_{i=1}^{n} c_i x_i$$

and was constrained as follows:

$$\sum_{i \in cov(k)} x_i \geq m \quad , k = 1, 2, \ldots, n_t$$

As before we model the possibility of operating a given sensor at varying

levels of energy consumption by splitting variables. Sensor i operating at power level p is represented by variable $x_{i,p}$ and to prevent the possibility of the algorithm choosing to run the sensor at two different levels simultaneously, we use the following constraint:

$$\sum_{p} x_{i,p} \leq 1 \quad , k = 1, 2, \ldots, n_t$$

Upon solving this problem, we can determine how many components are extant therein. We can now start to seek violated constraints. We do so by focusing on each of the targets in turn. Due to the coverage requirement we can guarantee that, at any given iteration of the algorithm, any given target t is covered by one or more components. If the number of components is greater than 1 then we seek valid constraints that are violated by the current solution to add.

To aid in the formulation of these constraints we introduce the variable $y_{i,j}$ to represent an edge between sensors i and j and $N(i)$ to denote the set of sensors with which sensor i can communicate. If the sensors are not capable of communicating with each other then $y_{i,j}$ is always zero-valued and can be excluded from the model. If sensors i and j are capable of communicating with each other, then having both be on automatically causes the edge to exist and $y_{i,j}$ takes a value of 1. If the sensors are capable of communication but one or both of them is not activated then $y_{i,j}$ takes a value of 0.

Where appropriate, $y_{i,j}$ can be seen to be the product of x_i and x_j.

$$y_{i,j} = x_i x_j$$

As is discussed in [37], we can avoid the use of this quadratic constraint by using the facet-defining inequalities that follow that describe the convex hull of triples $(y_{i,j}, x_i, x_j)$ that adhere to it.

$$y_{ij} \leq x_i \tag{3.1}$$

$$y_{ij} \leq x_j \tag{3.2}$$

$$x_i + x_j - 1 \quad \leq \quad y_{ij} \tag{3.3}$$

We are now better equipped to add constraints that involve the inclusion of edges.

For the formulation of specific constraints that are valid for any feasible solution to the *coverage* − *connectivity* problem that may be violated by the solutions found by the intermediate stages of our cutting plane algorithm we inspect the targets to be covered as well as the components that exist in the intermediate solutions.

Let $C^+(i)$ denote the union of all components covering target i, we can further define a set $C^*(i)$ as follows

$$C^*(i) = C^+(i) \cup cov(i)$$

For any given target, i, exactly one of the following conditions will hold C^* does not cover all the targets $C^*(i)$ covers all the targets

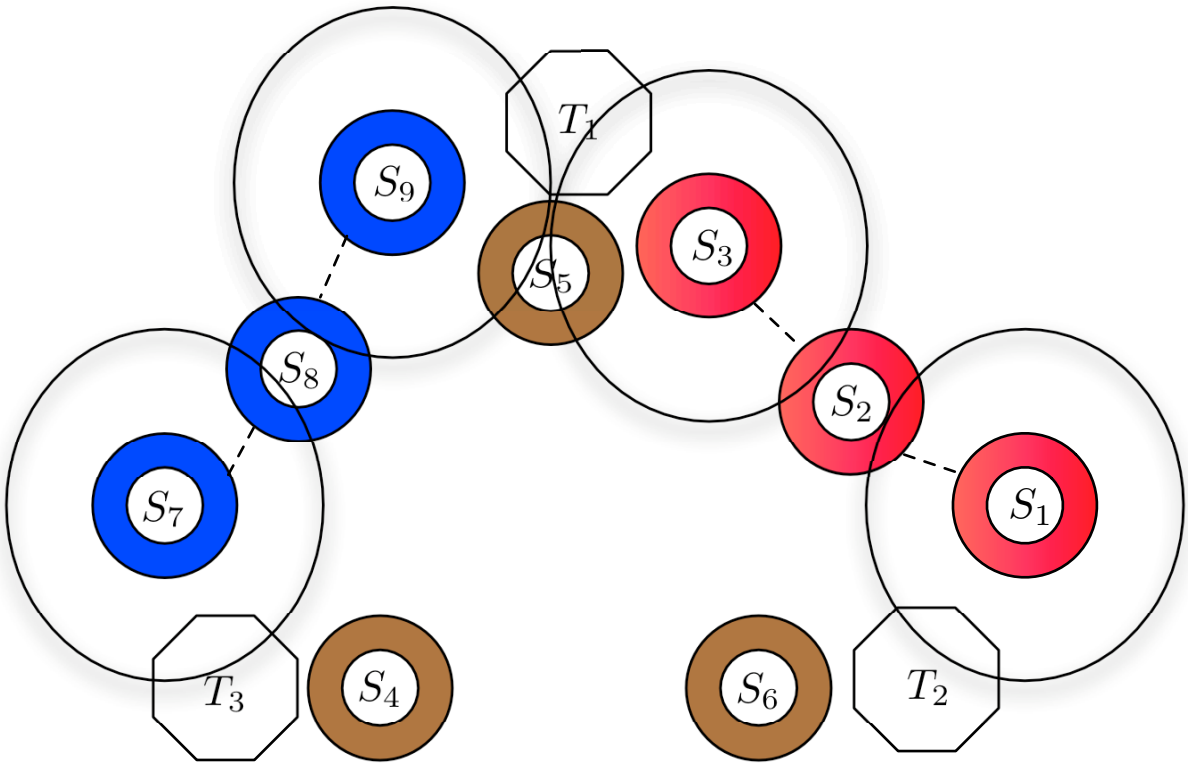
Figure 3.1 illustrates both of these cases. The sensors colored red and blue are on with each color being assigned to a single connected component. Target T_1 is covered by two components which between them cover all the targets i.e. $C^*(T_1)$ covers all targets. On the other hand, targets T_2 and T_3 are each covered by one component each, neither or which covers all of the targets. The brown-colored sensors are currently turned off i.e. $C^*(T_2)$ and $C^*(T_3)$ both fail to cover all the targets.

In each of these cases, we can generate violated constraints.

3.0.2.1 First type of constraint

If $C^*(i)$ does not cover all of the targets then we have some valuable information. First, we know that a member of $C^*(i)$ must be activated to meet the coverage requirement as $C^*(i)$ is a superset of $cov(i)$. Second, we know that at least one sensor that isn't a member of $C^*(i)$ has to be activated as there is a target that $C^*(i)$ doesn't cover. Connectivity necessitates that there be a path between these two sensors and thus there be at least one edge from $C^*(i)$ to its complement. Hence the constraint of the form (3.4).

Figure 3.1: Configuration that meets coverage requirement but fails to meet connectivity. The red sensors are all activated and form a component. The blue sensors are also activated; forming a connected component of their own.



$$\sum_{(i,j)\in\delta(C^*)} y_{i,j} \geq 1, \quad \text{where } \delta(C^*) = \{(i,j) : i \in C^*, j \notin C^*\} \tag{3.4}$$

3.0.2.2 Second Type of Constraint

In the situation in which $C^*(i)$ does cover every target, we can also generate valid cutting-planes which exclude the current solution. We know that in the optimal O a target, t, is covered by a single component: O itself.

The conclusion can then be drawn that at least one sensor that isn't currently activated needs to be activated. One possibility is that this sensor that would be activated in the optimal solution has the power to cover all targets (and is thus a member of $C^*(i)$). In this case, such a sensor would nullify the need for any other

sensors to be activated as a single sensor would constitute a connected configuration. On the other hand, the sensor may conceivably lie outside of $C^*(i)$ and need to form an edge with a member of $C^*(i)$. A third possibility is that this sensor is a member of $C^*(i)$ that isn't capable of covering all the targets and consequently needs to form an edge that isn't currently in existence, either within $C^*(i)$ or not.

Using Ω to represent the set of sensors that can possibly cover all targets, we have the constraint (3.5).

$$\sum_{j\in\Omega} x_j + \sum_{(j,k)\in\delta(C^*(i)),j\notin\Omega} y_{j,k} + \sum_{j\in(C^*(i)\setminus C^+(i))\cap\bar{\Omega},\ k\in N(j)\cap C^*(i)} y_{j,k} \geq 1 \tag{3.5}$$

3.0.3 The Algorithm Specified

Data: c, $cov(i)\forall i \in 1,\ldots,n_t, n, m$
Result: $S =$
set of sensors that satisfy coverage and connectivity requirements
Solve Coverage Problem;
Let $w =$# of components in solution
while $w \neq 1$ **do**
  foreach *t, a target* **do**
    Determine the appropriate type of constraint to add;
    Add the constraint to the model
  end
  Solve new problem;
  Update $w =$#of components in new solution;
end

Algorithm 2: Cutting-plane LP to establish connectivity

Now, we provide a proof that cuts of the types formulated above are sufficient to find the minimum-cost solution to the coverage-connectivity problem. It is to be noted that the cuts that we generate do not exclude any feasible solutions. Now we wish to show if all such solutions are added the optimal solution of the problem we have codified is indeed a feasible solution to the coverage-connectivity problem.

Theorem 1. *The cuts of the two types outlined in (3.4) and (3.5) are sufficient for the convergence of the algorithm*

Proof. Let O be a feasible solution, if O has any connected components that don't cover any targets it is suboptimal for the coverage-connectivity problem, as long as sensor costs are strictly postive, as we can remove these components for a reduction in objective value.

Assume that O is indeed optimal and features 2 or more connected components. There are two cases:

1) $\exists\ i$ such that $C^*(i)$ does not cover all the targets

2) Every target is covered by a collection of components that cover all the targets.

Case 1: We know that at least one member of $C^*(i)$ has to be active in order for the coverage requirement to be met. Also a member of its complement must be activated because there is a target that cannot be covered by $C^*(i)$. There must be a connected path between these two sensors necessitating at least one edge from $C^*(i)$ to its complement and we can add a constraint of the form 3.4.

Case 2: We can consider any target t. The union of the components covering t is disconnected and covers all targets. Unless the problem is infeasible there must be at least one sensor left to turn on which is either powerful enough to cover all the targets or would have to form an edge with a member of $C^*(t)$. Thus we can add a constraint of the form 3.5.

□

3.0.4 Results involving uniform sensors

3.0.4.1 A Note on Determining Feasibility

While utilizing problems that are randomly generated there is always the possibility that the problem instance is infeasible. We found this to be the case especially when smaller numbers of sensors were involved and this is not counterintuitive by any means. In order to detect feasibility we found the number of components that would be formed if all the sensors were used simultaneously. For the problem to be feasible we then need that at least one of the components covers all the targets.

In order to investigate how likely problems were to be infeasible we randomly generated problems on a square or side 500. The number of sensors was varied from 50 to 80, in increments of 10, and the number of targets was varied from 10 to 40, also in increments of 10. Four different values for the radius of the sensors were utilized: 100, 150, 160 and 175. All sensor and target locations were generated using a uniform distribution. Tables

Table 3.1: Infeasibility Rates for randomly generated problems with radius 100

% Infeasible Problems for Radius 100				
	No. of Sensors			
No. of Targets	50	60	70	80
10	47.3	24.2	10.9	4.7
20	61.0	34.4	18.1	7.6
30	68.7	43.1	22.4	9.9
40	71.9	47.6	26.3	11.5

It can be seen that adding more sensors to increases the probability of the problem being feasible while increasing the number of targets has the reverse effect. When the radius was increased to 160 the problems with 50 sensors and 40 targets were infeasible 0.5% of the time and all other combinations of parameters were infeasible even less frequently. When the radius was further increased to 175 only a single problem instance across all combinations of parameters was found to be infeasible. This problem occurred, not surprisingly, with 50 sensors and 40 targets.

In the case of area coverage problems being approximated by discrete targets

Table 3.2: Infeasibility Rates for randomly generated problems with radius 150

% Infeasible Problems for Radius 150				
	No. of Sensors			
No. of Targets	50	60	70	80
10	0.5	0.7	0.4	0
20	1.4	1.0	0.6	0
30	2.3	1.4	0.7	0
40	2.6	1.6	0.8	0.02

Table 3.3: Mean solution values randomly generated problems with radius 150

Mean Solution Values			
	No. of Sensors		
No. of Targets	60	70	80
10	5.27	4.92	5.13
20	7	6.55	5.6
30	6.5	6.2	5.5
40	6.84	7	7

these rates of infeasibility would seem to provide a useful measure of how well we are approximating the region. For any given radius the infeasibility rates tend to plateau as more targets are added. Since the number of targets is a crucial factor in the computational difficulty of the problem it is useful to use these values to determine how few targets we can use while maintaining a certain threshold feasibility rate.

3.0.5 Solution values and Convergence of the Algorithm

.

We solved several instances of connectivity problem on a square region of length and width 500. In the experiments we varied the number of randomly deployed sensors from 60 to 80 and required that they cover a number of targets that varied from 10 to 40, increasing in increments of 10. Each sensor had a coverage and communication radius of 150. We solved 15 instances of each type of problem.

Following are the mean solution values for the different problem instances.

In general, the problems with lower target numbers have lower cost optimal solutions. Perhaps, in conjunction with the previous observation that increases in target numbers provide diminishing marginal returns regarding the quality of area coverage, this can be useful in practical contexts. Larger numbers of sensors provide lower optimal solutions but as we shall see below they correspond to more extended run-times. Further analysis of the trade-off between computational difficulty and solution cost may also prove to be useful.

At each iteration we add a number of constraints equal to the number of

Table 3.4: Mean No. of iterations for randomly generated problems with radius 150

Mean No. of Iterations			
	No. of Sensors		
No. of Targets	60	70	80
10	5.2	3.2	5.4
20	10.8	10.8	10.23
30	10	9.4	14.67
40	16.33	11.6	17.5

targets. For most of the problems the number of iterations prior to convergence is relatively small and the optimal solution is often the first or second feasible solution found. For problems with 70 sensors and 40 targets or 80 sensors the number of iterations blows up.

3.0.6 Sample Problems

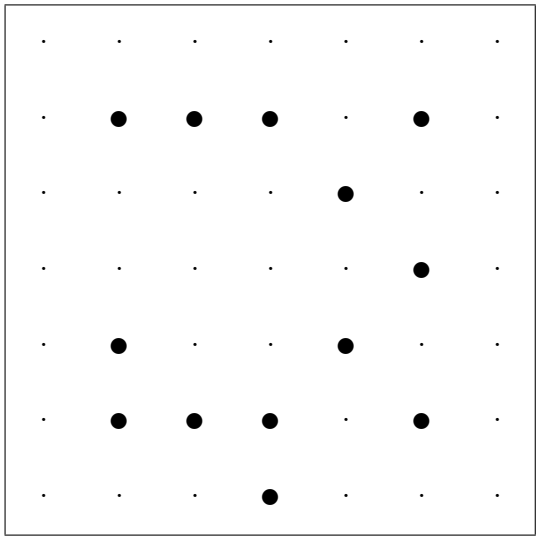
Figures 3.2 and 3.3 illustrate the results of using the previously described algorithm to deploy sensors that are spaced 1 unit apart in both the horizontal and vertical dimensions with ranges of 1.5 units.

Also, we have some examples of the solutions to randomly generated problems. As stated before, in these randomly generated problems we used a square of side 500 units to serve as the region. The number of sensors was varied from 60 to 80 and a radius of 150 was used for both coverage and connectivity. These particular values were used as they represent some of the more substantial problems solved in [6]. In [6] the radii was varied from 100 to 300 and the number of sensors from 50 to 100. For values for the radius greater than 150 the problems were solved even more easily and for values near the lower end of the range it was difficult to generate feasible problems with reliability especially for problems involving less than 60 sensors.

3.0.7 The Cost of using a particular sensor

As noted in [33], two common metrics used for evaluating the cost of operating a sensor at a certain power level are to make the cost of operation proportional to the operating radius or the area of coverage/communication. When the set of values

Figure 3.2: Connected graph for a 7x7 grid-based problem, Radius: 1.5 units

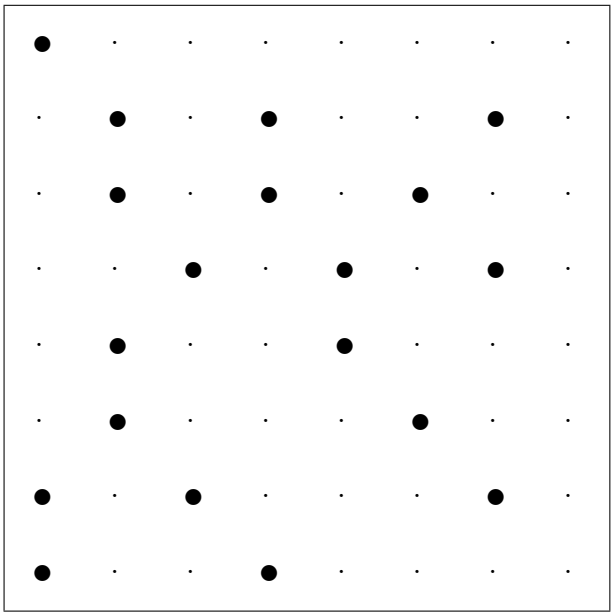


that the sensors range can be set to form a continuum the modeling approach is to discretize these values, for example if the operating radius can be made to be as high as 60 meters we might consider 6 different operating levels: 10, 20, 30, 40, 50 and 60 meters. Sometimes it is possible to use the same number of sensors even when reducing the power of each sensor. Figure 3.7 illustrates a map in which 40 sensors and 30 targets have been placed according to a uniform random distribution. Figure 3.8 illustrates that the optimal number of sensors doesn't increase if the operating radius is reduced from 150 to 125. For a cost metric in which the cost of using a sensor grows linearly with the sensor's area of service a sensor with operating radius of 150 represents a 44% increase in power over one that has a radius of 125. If the cost grows at greater than quadratic rate then the energy savings will be even more drastic.

3.0.7.1 Computational Considerations

Via experimentation it has been found that a cost metric in which the sensor cost is equal or proportional to the area being covered isn't ordinarily as conducive

Figure 3.3: Connected graph for a 8x8 grid-based problem, Radius: 1.5 units



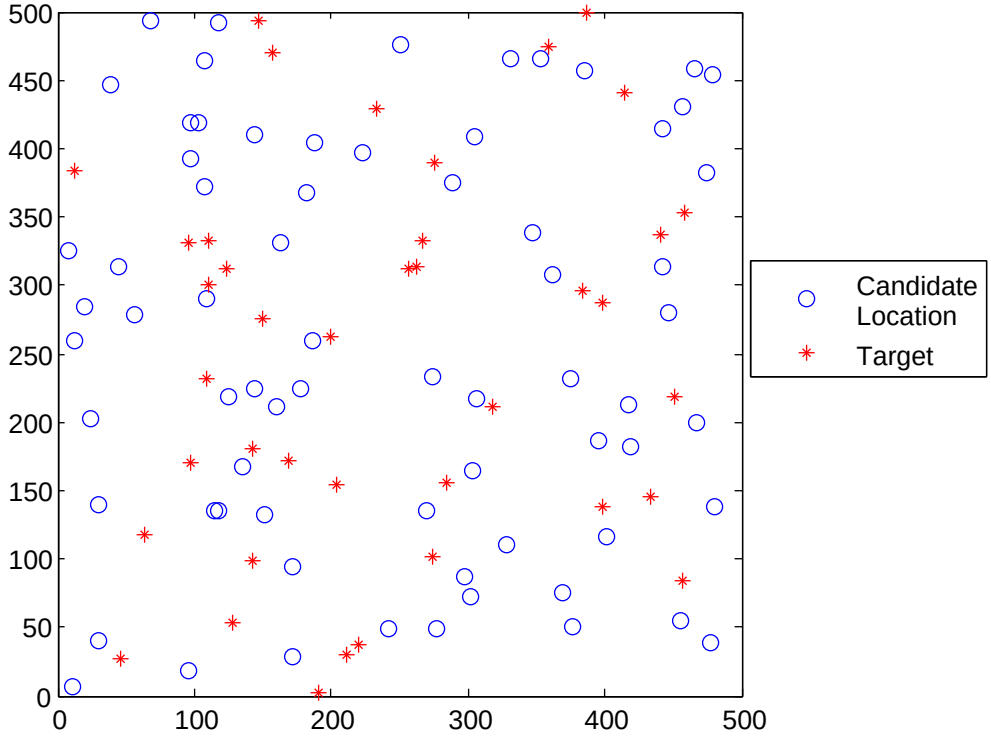
to fast convergence as a metric in which the cost is equal or proportional to the radius. This is by no means counterintuitive as quadratic growth in costs exerts extra pressure on sensors to communicate only with those in close proximity.

Left to its own devices a method seeking only the fulfillment of coverage requirements will tend to emphasize the use of sensors that provide most coverage per unit cost while marginalizing other types of sensors to roles filling gaps. When the cost of a sensor increases as a linear function of the radius, it is actually increasing at a rate that is less than linear when considered as a function of the area being covered. If there is a notable positive correlation between the area of a subregion and the number of sensors contained therein, there is consequently an impetus to use sensors with large ranges which in turn makes it so that components tend to cover more area and thus more components. When the costs grow steeply relative to changes in coverage, the intermediate stages of the algorithm tend to yield numerous small components

Fortunately, we can take advantage of the fact that the cuts we generate are always valid for any feasible connected solution regardless of which costs are used to

bring them about. We have found that solving the problem with linearly increasing costs and then changing the objective function, whilst using all of the previously generated constraints, gives a useful head start. In fact, in practice we actually increase the growth rate of the costs gradually and it seems possible that we could even investigate the use of costs that are monotonically increasing and subadditive as functions of the radius.

Figure 3.4: 70 Sensors, 40 Targets, Radius: 150



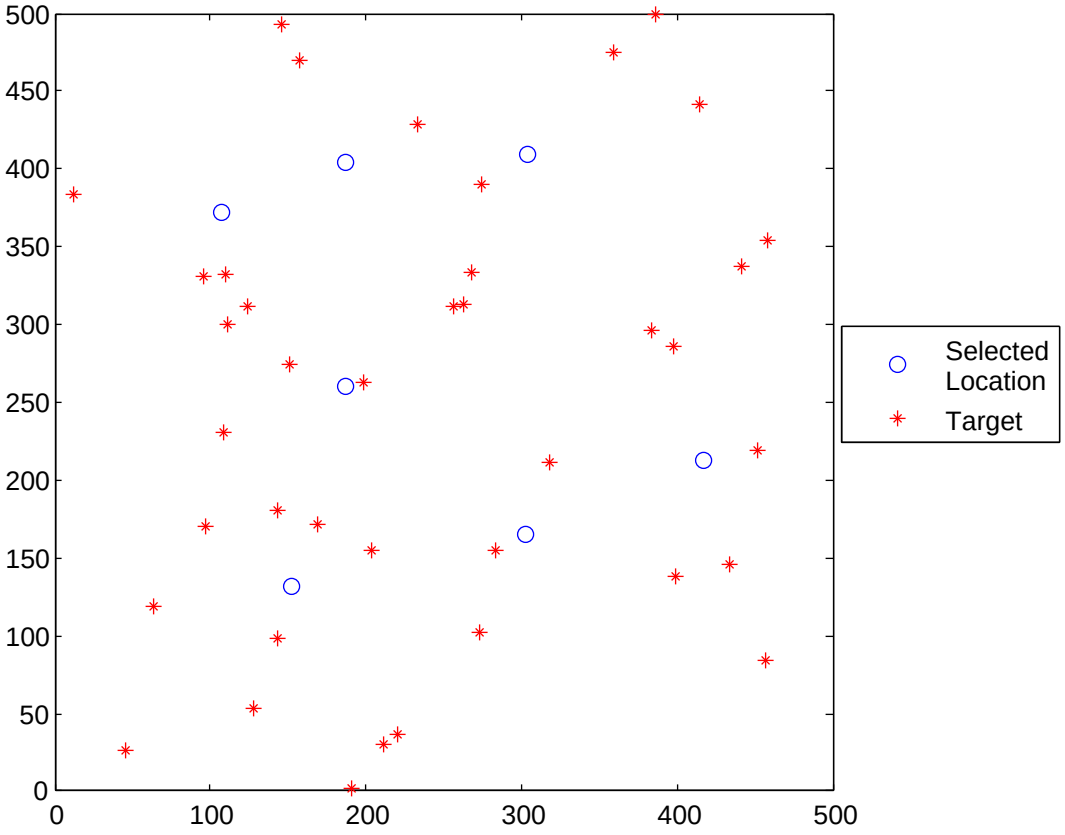


Figure 3.5: 80 Sensors, 10 Targets, Radius: 100

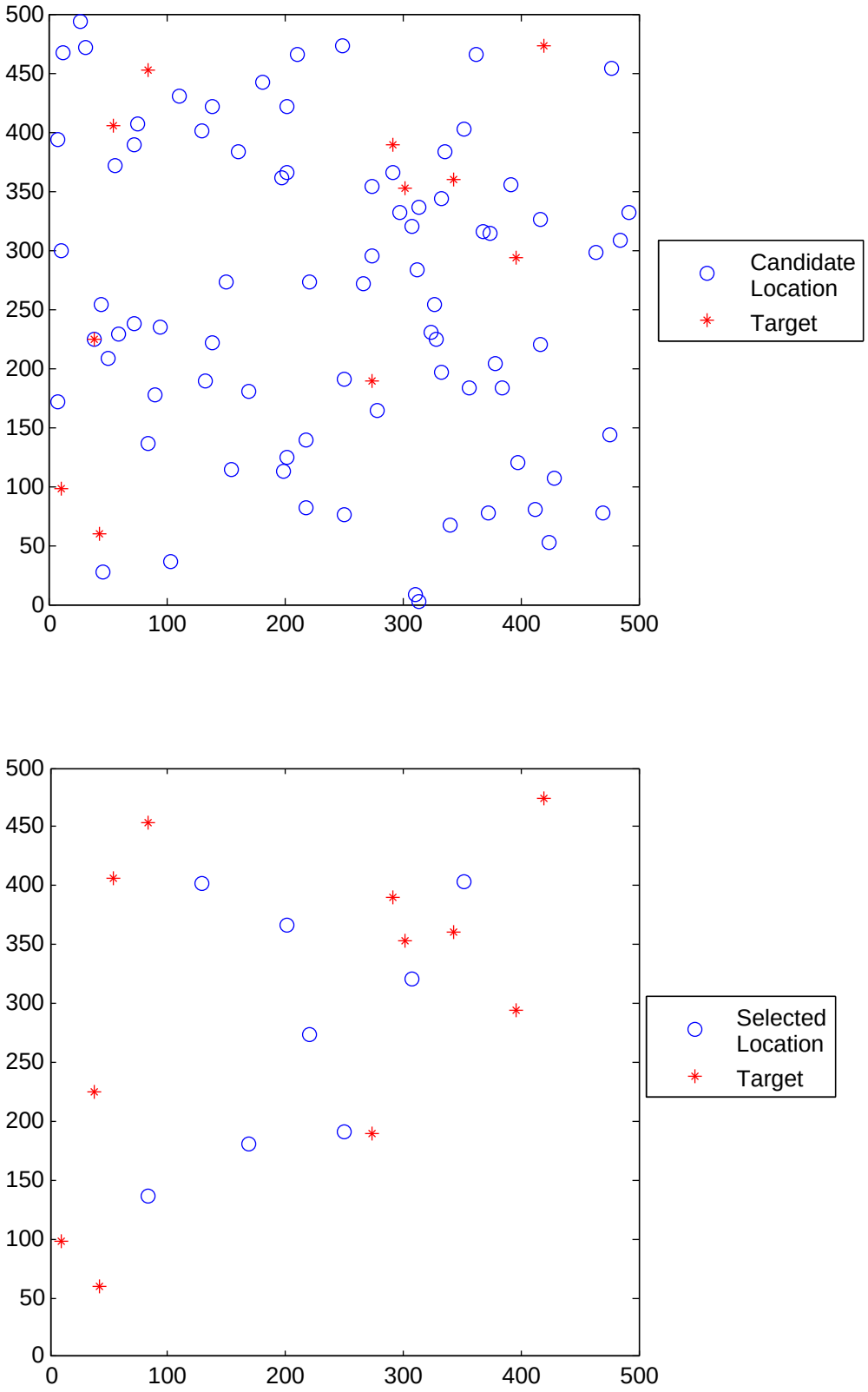
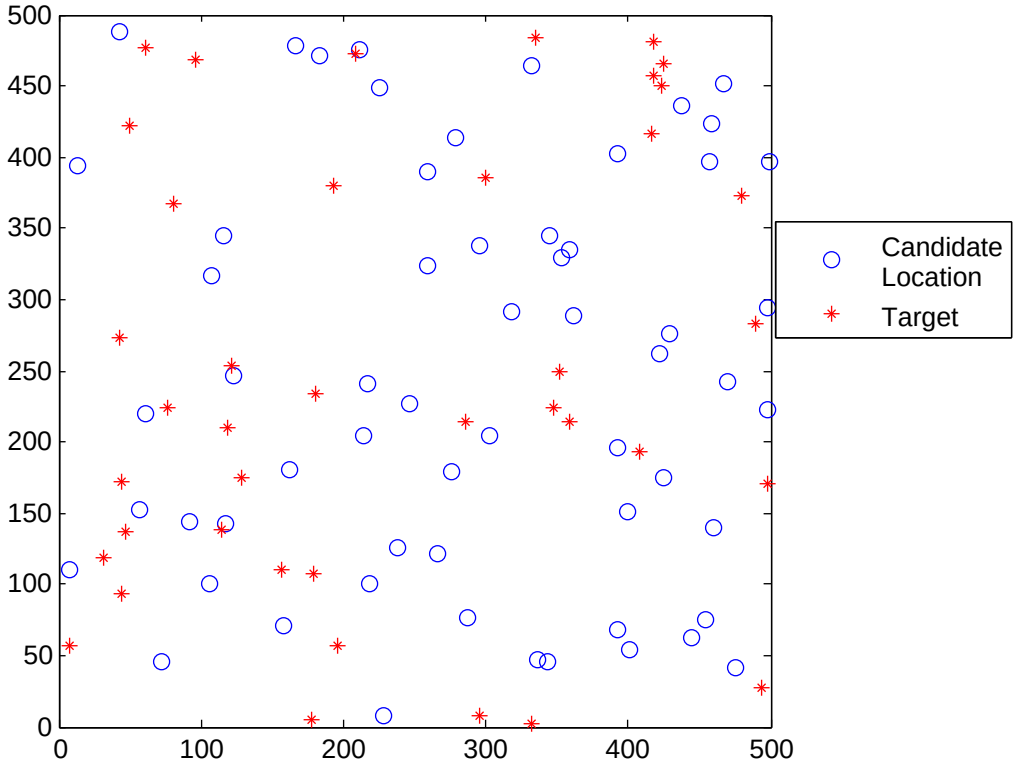


Figure 3.6: 60 Sensors, 40 Targets, Radius: 150



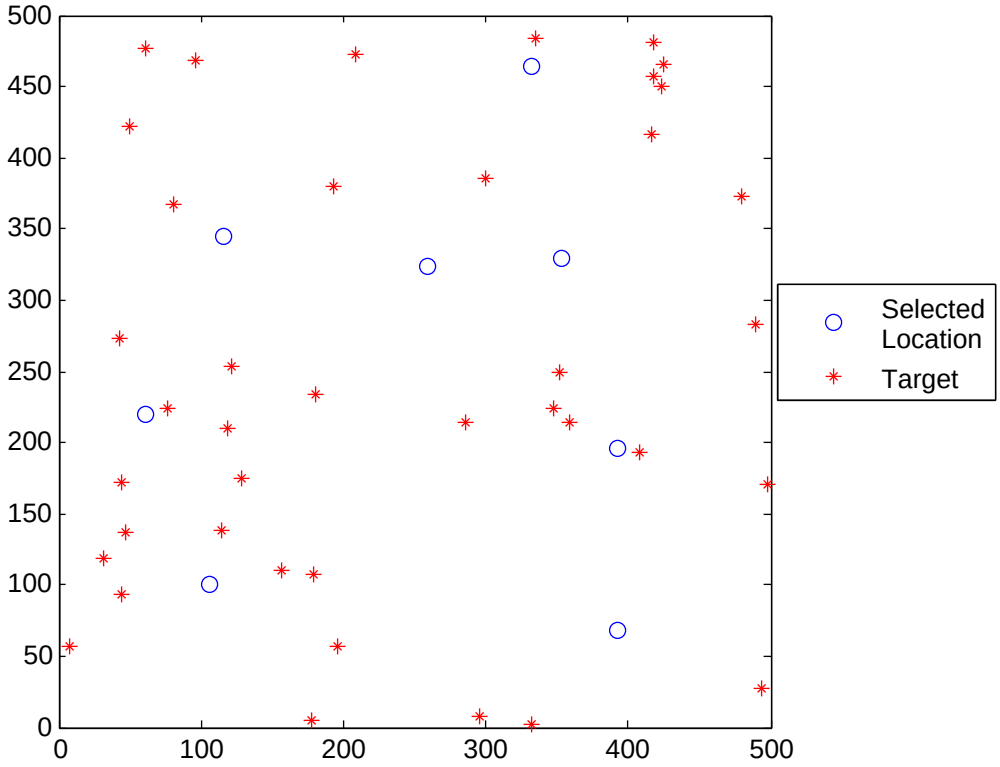


Figure 3.7: 40 Sensors, 30 Targets, Radii: 125 and 150

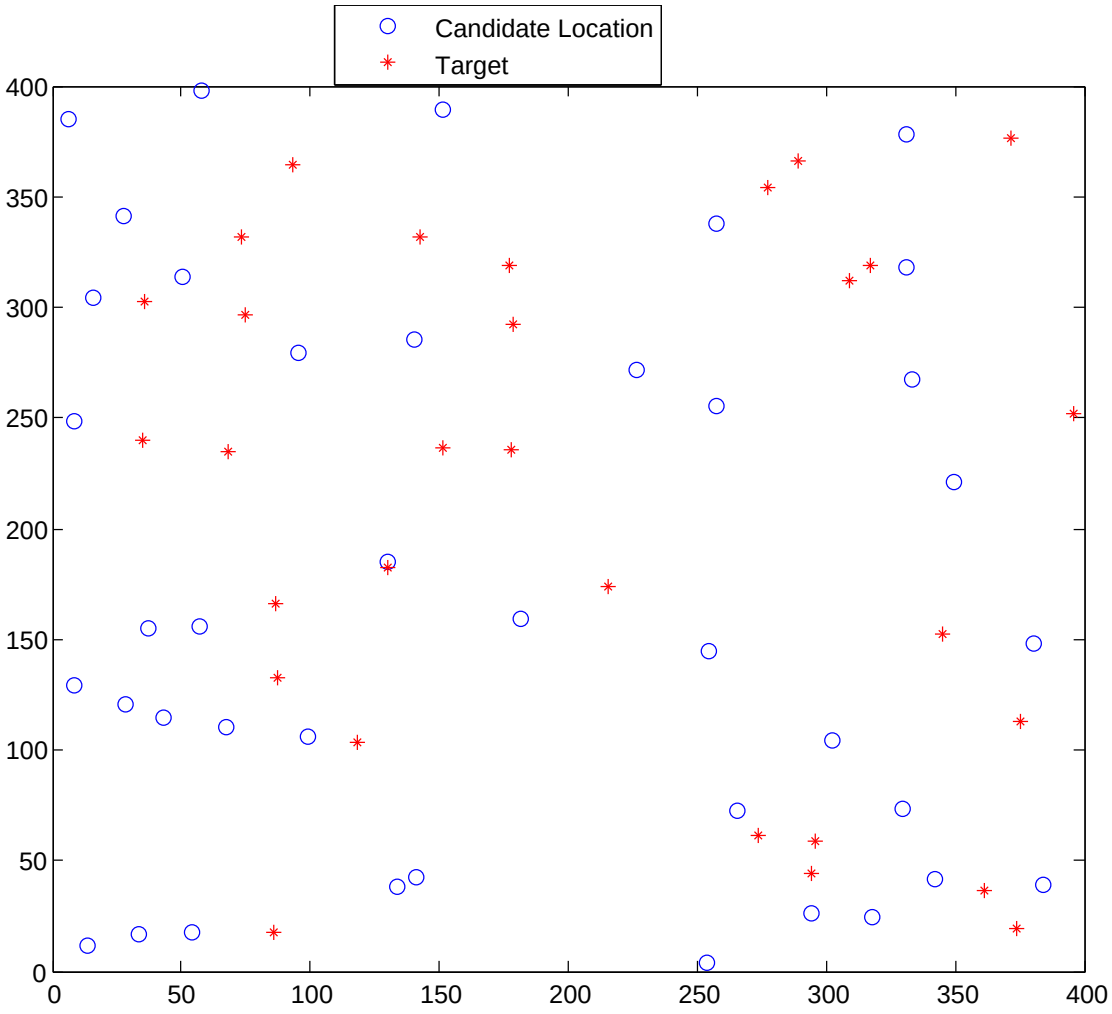


Figure 3.8: 40 Sensors, 30 Targets

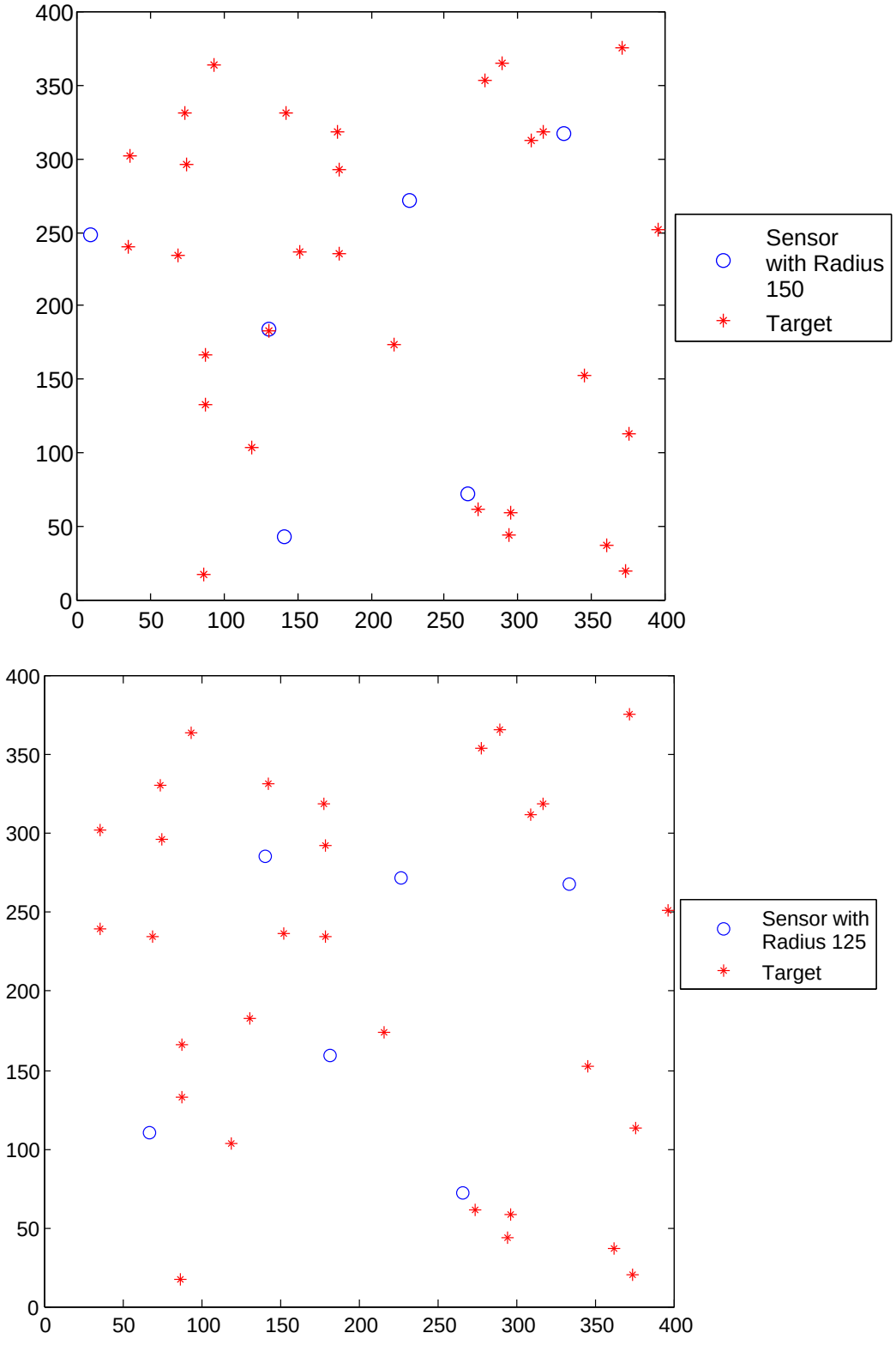
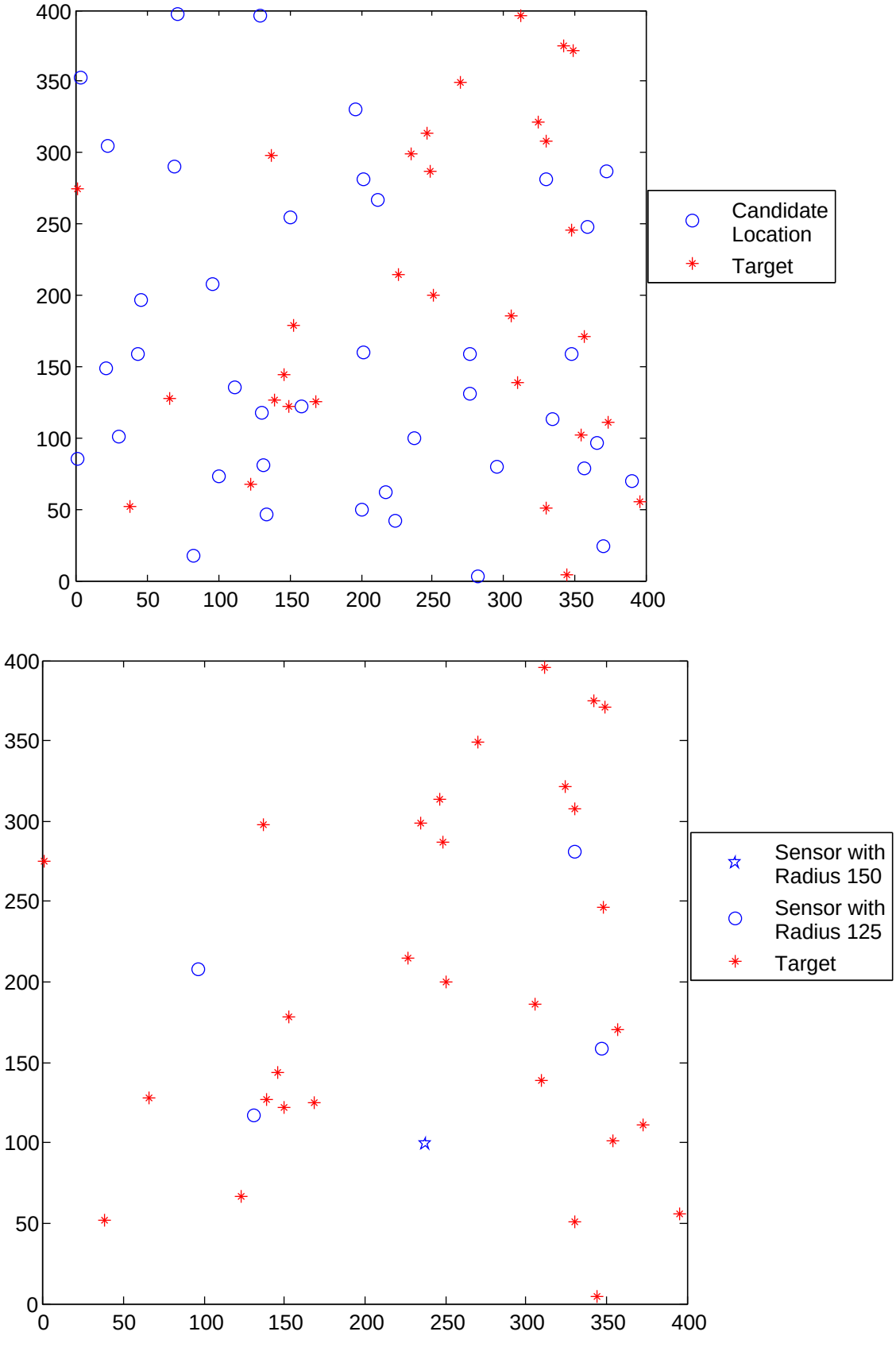


Figure 3.9: 40 Sensors, 30 Targets, Radii: 125 and 150



CHAPTER 4
A Semidefinite Programming Approach for Coverage and k-connectivity

4.1 Background Information

Associated with any graph is a matrix known as its Laplacian. The properties of the Laplacian relay much salient information regarding the connectivity of the graph. It is our intent to make use of these Laplacians to tackle the problem of satisfying coverage requirements while establishing k-connectivity.

Prior to the introduction of the Laplacian we introduce the adjency and degree matrices of a graph. The degree matrix of a graph, D, is a diagonal matrix in which each entry corresponds to the degree of the corresponding vertex. As for the adjacency matrix of a graph, the i, j entry corresponds to the number of edges between vertices i and j. Take for example the graph in figure 4.1.

It has the following degree and adjacency matrices:

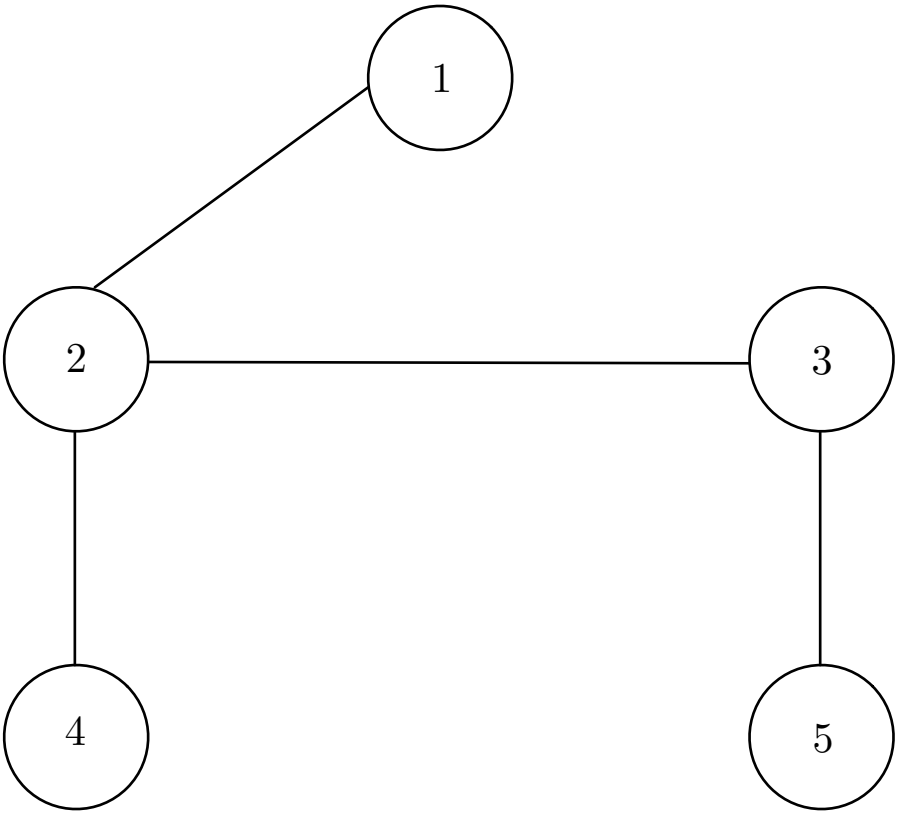
$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The Laplacian matrix is defined as the difference of the degree and adjacency ma-

Figure 4.1: An example of a graph



trices.

In our particular example, the Laplacian is:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

The Laplacian matrix for any given graph is diagonally dominant with positive entries on the leading diagonal and is thus positive semidefinite. Also by definition the sum of each row or column is zero. This means that the linear transformation represented by the Laplacian maps e, the vector of ones, to the zero vector of the same dimension and that $e^T Le = 0$ indicating that e is necessarily in the eigenspace

of the eigenvalue 0.

The information that is most relevant to us relates to a characteristic of the second-smallest eigenvalue that is stated succinctly below.

Theorem 2. *The second-smallest eigenvalue of the Laplacian matrix is known as the algebraic connectivity and provides a lower bound for the vertex connectivity of the graph [20].*

In fact, the geometric multiplicity of 0 eigenvalue is equivalent to the number of connected components in the graph and as such we know that for a connected graph the algebraic connectivity must be strictly positive.

More detailed coverage of the properties of the Laplacian matrix can be found in [13] and [20].

4.2 A semidefinite programming approach

The gist of our approach is to model our problem as a graph, in which vertices represent sensors and adjacencies imply the ability to communicate, and subsequently to find an induced subgraph with the minimum number of nodes that manages to meet constraints implied by our coverage and connectivity requirements.

The formulation of constraints that secure an adequate level of coverage is relatively simple: the constraints are linear and their derivation is straightforward. On the other hand, the constraints that ensure connectivity involve semidefinite variables and their formulation is noticeably more involved.

Using m to denote the number of sensors we wish to have available to cover each target, we first formulate constraints to ensure that our coverage requirements are met

$$\sum_{i \in cov(j)} x_i \quad \geq \quad m \quad , j = 1, 2, \ldots, n_t \tag{4.1}$$

We note that we can generalize on this problem and have each target have its own required level of coverage.

To tackle the problem of ensuring a certain level of connectivity we turn to the Laplacian. In order to guarantee k-connectedness we need to ensure that $\lambda_2 \geq k-1$. A Laplacian matrix, L, that meets this criterion is accurately characterized as follows:

$$d^T L d > (k-1)||d||, \quad \forall d \text{ such that } v_1^T d = 0 \text{ where } Lv_1 = 0 \text{ and } ||v_1|| > 0$$

In order to avoid the use of a strict inequality, we introduce a constant γ, which is of the following form

$$\gamma = (k-1) + \epsilon$$

where ϵ is a small constant to be determined at the time of implementation and we now have the following non-strict inequality

$$d^T L d \geq \gamma||d||, \forall d \text{ such that } v_1^T d = 0 \text{ where } Lv_1 = 0 \text{ and } ||v_1|| > 0$$

Now, as before, we use x to denote the vector of decision variables corresponding to sensors and y to denote the edges between sensors. Both x and y are 0-1 vectors with $x_i = 1$ if and only if sensor i is deployed and $y_{ij} = 1$ if and only if both sensors i and j are deployed and they are capable of communicating with each other.

We use $\mathcal{I}$ to denote a subset of the set of possible indices for x and $x_{\mathcal{I}}$ to represent the configuration in which sensor i is deployed if and only if $i \in \mathcal{I}$. Similarly we use $L^{\mathcal{I}}$ to represent the Laplacian of the subgraph induced by $x_{\mathcal{I}}$. Our job is to sift the set of all possible sensor combinations and come away with the $\mathcal{I}$ that minimizes cost subject to our constraints.

In order to ensure that we have a valid Laplacian matrix we need to have certain constraints in place

$$L^{\mathcal{I}}{}_{ii} = \sum_{j \in N(i)} y_{i,j}, \forall i \in \mathcal{I} \tag{4.2}$$

$$y_{ij} = y_{ji} \tag{4.3}$$

Here we are making the assumption that communication between sensors is symmetric. This lines up well with the standard assumption made when using the disk model that two sensors can communicate with each other if and only if the distance between them is less than or equal to the minimum of their communication radii.

A preliminary requirement for k-connectivity is that each sensor chosen to be on must have k neighbours chosen to be on

$$L^{\mathcal{I}}{}_{ii} \quad \geq \quad k, \forall i \in \mathcal{I} \tag{4.4}$$

In order to ensure the k-connectivity of the induced subgraph we need $L^{\mathcal{I}}$ to satisfy

$$d^T L^{\mathcal{I}} d > (k-1)||d||, \forall d \text{ such that } v^T d = 0 \text{ where } L^{\mathcal{I}} v = 0 \text{ and } ||v|| > 0$$

We know that the eigenvector corresponding to $\lambda_1(L^{\mathcal{I}}) = 0$ is the vector of ones and it follows that the eigenvector corresponding to $\lambda_1(L^*) = 0$ is a vector, v^* where

$$v^*{}_i = \begin{cases} 0 & i \notin \mathcal{I} \\ 1 & i \in \mathcal{I} \end{cases}$$

We can consider a matrix,

$$X = L^* + cW - \gamma I$$

where,

$$L^* = \begin{pmatrix} L^{\mathcal{I}} & 0 \\ 0 & kI \end{pmatrix}$$

$$W = \begin{pmatrix} ee^T & 0 \\ 0 & 0 \end{pmatrix}$$

(The block of ones in the upper left corner of W is of the same size as $L^{\mathcal{I}}$.)

and

c is a positive constant

.

Due to the block-diagonal structure of X, its eigenvalues (and eigenvectors) correspond to those of each of the blocks.

The addition of cW to L^* has the effect of shifting the minimum eigenvalue of the upper left block, i.e. $\mathcal{I}$, from zero to a positive constant that is a function of c. None of the other eigenvalues is affected by this first addition. The subtraction of γI from $cW + L^*$ causes all the eigenvalues to then be shifted γ in the direction of $-\infty$.

Since, k is larger than γ, it is easy to see that,

$$\lambda_2(L^*) \geq \gamma \Leftrightarrow \lambda_2(L^{\mathcal{I}}) \geq \gamma$$

So, if c is chosen to be sufficiently large, we can say that

$$X \succeq 0 \Leftrightarrow \lambda_2(L^{\mathcal{I}}) \geq \gamma$$

and thus the task of ensuring the appropriate level of connectivity can be reduced to ensuring that X is positive semidefinite. If $\mathcal{I}$ has already met the coverage criterion, then it is a feasible solution to the coverage-connectivity problem. What would remain would be the determination of which choice of $\mathcal{I}$ minimizes the appropriate objective function.

So, in order to formulate the problem as an SDP, we'll need some constraints to enforce X fitting the prescribed form.

First, we'll introduce the matrix W defined as follows

$$w_{ij} = x_i xj \tag{4.5}$$

with the elements of W effectively serving as an indicator of whether or not both i and j are in $\mathcal{I}$ and thus whether or not the corresponding elements of L^* are subjected to the extra shift. Due to the 0-1 nature of the entries in x we can use linear equations to represent the conjunction whilst avoiding the computational difficulties that would come along with quadratic constraints

$$w_{ij} \leq x_i \tag{4.6}$$

$$w_{ij} \leq x_i \tag{4.7}$$

$$x_i + x_j - 1 \leq w_{ij} \tag{4.8}$$

$$w_{ij} \geq 0 \tag{4.9}$$

We use W first in the formulation of a constraint concerning the structure of L^* and subsequently in generating an expression for L^* in terms of X.

$$-kx_i + (k - L^*_{ii} + \sum_{j \in N(i)} w_{i,j}) = 0, \ \forall i \tag{4.10}$$

Examining (4.6) and (4.7), we can see that setting any particular x_i to be zero has the effect of forcing any associated w variables to also be zero which in turn would imply that L^*_{ii} has to equal to k for the equality in (4.10) to hold. Setting x_i to one would mean that any associated w variables linked to x_i and any of its neighbours that are simultaneously set to one are thus forced to one themselves. Again, L^*_{ii} is then forced to act appropriately in order for the constraint (4.10) to hold.

Recall that,

$$L^* = X - cW + \gamma I \tag{4.11}$$

Substituting (4.11) in (4.10) we get

$$-kx_i + (k - X_{ii} + cw_{ii} - \gamma + \sum_{j \in N(i)} w_{ij}) = 0, \ \forall i \tag{4.12}$$

Now, $w_{ii} = x_i$ and by definition $k - \gamma = 1 - \epsilon$, so we are left with

$$(c - k)x_i - X_{ii} + \sum_{j \in N(i)} w_{ij} = \epsilon - 1, \ \forall i \tag{4.13}$$

We are also in need of constraints that restrict the off-diagonal entries of X. If

sensors i and j are in range of each other, then an edge exists if both are turned on i.e.

$$(X - cW + \gamma I)_{ij} = -w_{ij} \tag{4.14}$$

and simplifying

$$X_{ij} = (c-1)w_{ij} \text{ if } i \in N(j) \tag{4.15}$$

In the case that sensors i and j don't lie in each other's communication radii, no edge exists between them and the value X_{ij} is determined solely by whether or not both sensors are chosen to be on. Explicitly

$$X_{ij} = cw_{ij} \text{ if } i \notin N(j) \tag{4.16}$$

In both scenarios, X_{ij} is merely a scaling of w_{ij} and plugging these into (4.12), (4.6), (4.7), and (4.8) we have a compact formulation of the problem as an SDP

$$\begin{aligned}
\min \quad & e^T x & & \\
& (c-k)x_i - X_{ii} + \tfrac{1}{c-1} \sum_{j \in N(i)} X_{ij} &=& \epsilon - 1, \ \forall i \\
& tX_{ij} &\leq& x_i \\
& tX_{ij} &\leq& x_j \\
& x_i + x_j - 1 &\leq& tX_{ij} \\
& X_{ij} &\geq& 0 \\
& X &\succeq& 0 \\
& x &\in& \{0,1\}^n
\end{aligned}$$

$$\text{where } t = \begin{cases} \frac{1}{c} & i \notin N(j) \\ \frac{1}{c-1} & i \in N(j) \end{cases}$$

It is to be noted that this formulation is indifferent to factors such as the shape of the sensing region and the layout of the potential sensor locations. If the sensing and communication radii of any given sensor are the same then a single matrix suffices

as input; if a disparity exists then two matrices are adequate. Further, if there are no asymmetries in sensing, then we only need to store the upper triangle of each input matrix.

4.3 Computational Considerations

The problem formulated above is an SDP with the additional constraint that the elements of x are integer and those of X be chosen from a discrete set. Presently, there is no available software for solving problems with both of these characteristics. It is our approach to implement a branch and bound algorithm to solve these problems.

A given node of our branch and bound tree represents a relaxation of the original problem as well as a constriction of the relaxations represented by its ancestor nodes.

Due to the 0-1 nature of the variables we can branch after solving a relaxation by choosing a subset of the x components currently assigned fractional values and fixing them to be integer henceforth. We can get an upper bound on the best feasible solution to be found in any of the descendant nodes of the present relaxation by simply taking a count of the nonzero entries in the current x; a lower bound can be had even more easily by taking the optimal value of the current relaxation.

The set of all nodes at a given depth of the tree has the property that every feasible solution to the problem is a feasible solution for exactly one member of the set. Thus, even if solving the problem to completion is beyond our computational means, we can conceivably get good upper and lower bounds for the solution by examining the solutions to all the nodes of a certain depth: the smallest of the upper bounds serves as a valid upper bound on the optimal solution and a lower bound can be found similarly by taking the minimum of the lower bounds.

The value of ϵ chosen may also prove to be very important. The larger the value of ϵ the smaller the pool of feasible solutions. In fact, making ϵ too large may very result in an unnecessarily infeasible problem.

4.3.1 Obtaining feasible solutions at each node

Due to the computational complexity of the problems, that comes with bundling integrality constraints with semidefinite programming it may not be practical to actually solve these problems to optimality. However, it is possible that we generate a collection of good feasible solutions and in many applications this is superior to having a single optimal solution on hand.

At any given node of the branch and bound tree the optimal solution is more likely than not to be fractional in nature i.e. several of the x variables which act as indicator variables for the sensors may have values that lie in the interval (0,1). It is tempting to simply round these fractional values up to the nearest integer but first we should investigate whether or not this is guaranteed to give a feasible solution to the original problem.

Recalling (4.8), we can see that if the sum of two fractional variables, x_i and x_j, is less than 1 there is a possibility that the associated $w_{i,j}$ variable to remain will remain at its zero lower bound.

We can consider the example of a variable x_i that has a value of $a \in (0, 1)$ in the fractional configuration returned as the optimal solution to one of the relaxations. If none of the w variables are associated with a sensor have positive values then that sensor is not being considered as part of $\mathcal{I}$ for the purpose of calculating the connectivity of the configuration. Instead it is associated with the diagonal block of L^*, the block corresponding to sensors that are not chosen to be on. In particular we know that

$$L^*_{i,i} = (1 - a)k$$

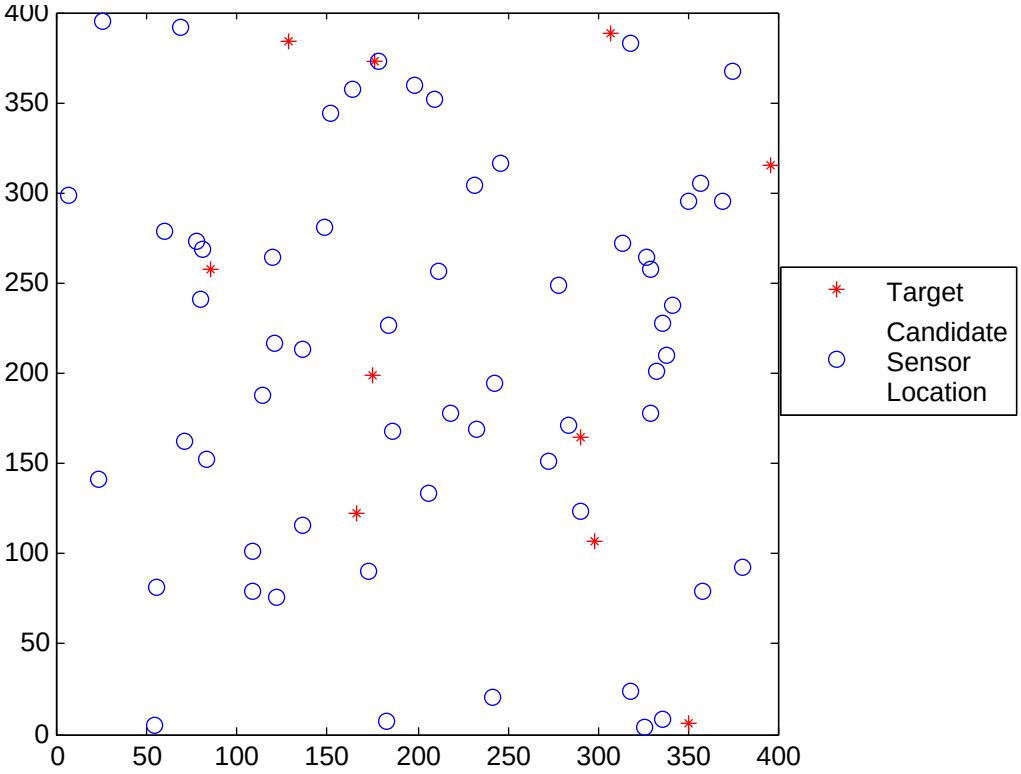
in order for constraint (4.10) to be satisfied. If $(1 - a)k \geq \gamma$ then this L^* satisfies all the feasibility requirements but it does so by considering the contributions of sensor i toward fulfilling the coverage requirement while ignoring the effects that its presence would have on the connectivity of the graph.

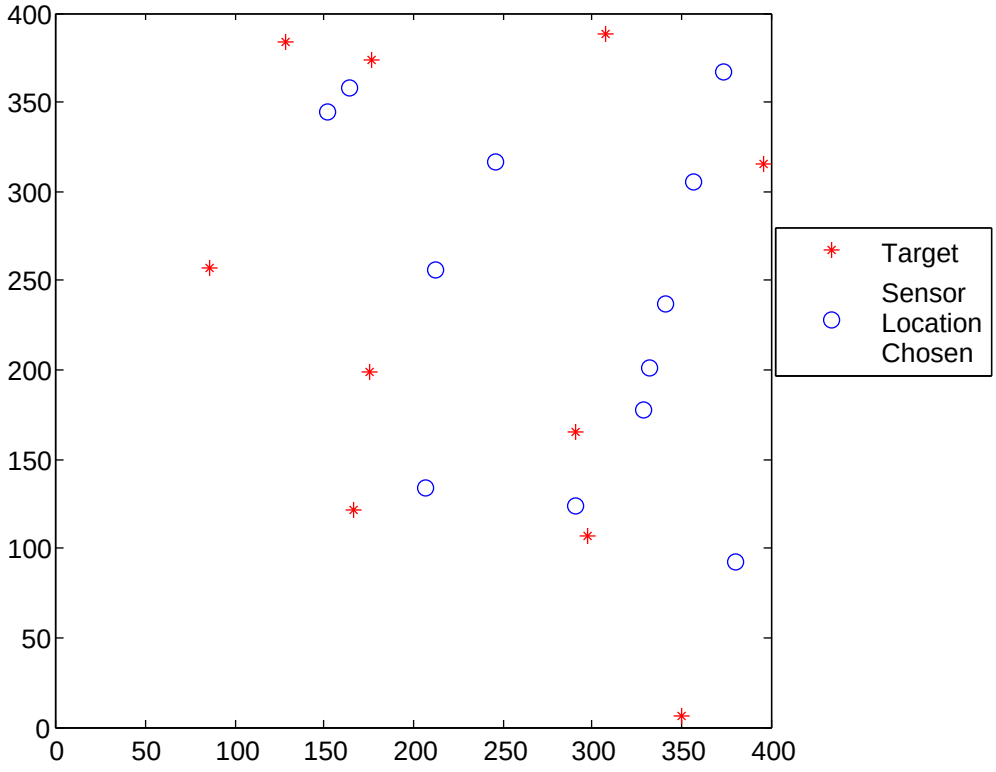
Thus, when presented with any given fractional solution we need to deal with fractional values of the x variables very carefully. Ideally, we would fix some subset of the fractional variables and this would lead to integer solutions upon re-solving.

We have yet to implement a branch and bound algorithm fully but for some

problems with sufficiently large radii relative to the size of the map we can find feasible solutions relatively easily. Figure 4.2 for example depicts a 400x400 map in which 60 sensors and 10 targets have been distributed. Each sensor is assumed to have an operating radius of 150. The coverage requirement is that each target be covered by at least 2 sensors. The connectivity requirement is that the chosen network configuration be a 2-connected graph. The feasible solution shown was obtained via our branch and bound method in its current form.

Figure 4.2: 60 Sensor Locations are available to choose from and there are 10 targets to be covered. The chosen network of 12 sensors can withstand the failure of any given node. Each sensor operates with a radius of 150





CHAPTER 5
Extending Network Lifetime

5.1 Background Information

The most referenced drawback to using centralized approaches to sensor placement problems is that the effort and/or time required to arrive at the optimal solution are exorbitant. It is our belief that this is mitigated by the fact that most optimization algorithms unearth multiple feasible solutions before coming across the optimal solution. In fact, in many cases the long run-times are associated not with discovering the optimal solution but rather with obtaining a certificate of optimality: it is not just important to find the best solution but also to verify that it is indeed a good solution. With that in mind, the questions then arise: how useful are the intermediate feasible solutions generated by our optimization methods? Perhaps, it is possible that the configurations that we come upon prior to optimality can be considered to be good in some appropriate metric. Or maybe, for some scenarios, having several good solutions may trump having a single optimal solution. One such scenario in which this might be the case is one in which the objective is to maximize the lifetime of the network. Other than the previously discussed measure of possibly turning down the power of sensors where appropriate the most common means of prolonging the lifetime of the network is to alternate between valid configurations for as long as possible. In such a situation, it would be of benefit to have a collection of feasible solutions as opposed to a single solution. As such, we tailor all our algorithms to record all the feasible solutions that may be encountered during the solving process. For negligible extra effort we possibly gain a means of prolonging network lifetime.

It is easy enough to see that if two or more disjoint sets of sensors could be found, each meeting both coverage and connectivity requirements, they could be turned on in succession to prolong the lifetime of the network. Finding such disjoint sets is the most venerable approach to solving lifetime maximization problems. However, it is also possible to alternate between configurations that aren't necessarily

disjoint to achieve a gain in network lifetime.

For example consider a scenario in which we have 5 identical sensors, A, B, C, D, and E, available to be used and it has been previously determined that 3 legitimate combinations meet the coverage/connectivity requirements at hand and are specified as follows

$$\begin{aligned} C_1 &= \{A, B, C\} \\ C_2 &= \{C, D, E\} \\ C_3 &= \{E, A, B\} \end{aligned}$$

If we let l denote the lifetime of the sensors, it can be seen that using any of these configurations for period l completely exhausts the sensors involved in that particular configuration while preventing the use of any of the other valid configurations. On the other hand, using each of the configurations for a period $\frac{l}{2}$ and alternating between them allows for the lifetime of the network to be extended to $\frac{3l}{2}$. As is illustrated in this example the relevant constraints for tackling a problem such as this is that each individual sensor has a certain battery capacity (and a commensurate lifetime) that it cannot be reasonably requested to exceed.

Given a collection of feasible configurations it is straightforward to select a subset of these, and a specification of how long a period of time each should be employed for, that maximizes the lifetime of the network. What is key is the generation of the set of configurations: all else being equal, we would like that the quantity and quality of the solution pool be both relatively large.

5.2 Formulating the Problem

Given a list of N configurations, we can begin to formulate the problem. As an objective, we'd like to maximize the sum of the lengths of times we can use each configuration for. Using t_i to denote the time period we wish to use configuration i, the formalized objective is

$$\sum_{i=1}^{N} t_i$$

Now we need constraints to ensure that no one sensor is asked to use more

energy than allowed by its battery's capacity. We will use a and p parameters to represent the battery capacity and power levels associated with specific sensors. These variables will be valued as multiples of some standard capacity for energy and operating power respectively. The p variables are doubly indexed with $p_{i,j}$ representing the rate at which sensor i uses energy in configuration j. If all sensors are of the same type, we can set a to one. If all the sensors operate with the same fixed radius, it is then the case that the p variables are all zero-one valued: zero if sensor i is not used at all in configuration j and one if it used at the standard power level. We then have the following linear programming formulation for the problem:

$$\begin{array}{lrcll} \max & e^T t & & & \\ \text{subject to} & \sum_{j=1}^{N} p_{i,j} t_j & \leq & a_i, \ \forall i & (LM) \\ & t & \geq & 0 & \end{array}$$

This is a straightforward linear program and thus solving even very large instances should not be excessively onerous computationally.

5.2.1 Results

As a measure of lifetime elongation we use the ratio of the network's lifetime to that of a single sensor operating at what we consider to be its default power level. In situations in which one (or more) sensors features in every utilized configuration we have no increase in network lifetime and a magnification factor of 1. The following data gives the lifetime magnification factors for the problems solved in Chapter 3 using just the set of feasible solutions generated by the algorithm developed therein.

These results don't appear to be particularly impressive seeing that the optimal solutions use a very small percentage of the sensors available. Even though the sensors that feature in an optimal solution hardly constitute a random sample it seems intuitive that the probability of the problem remaining feasible once these sensors have been removed should be favourable. Such an occurrence would indicate the existence of disjoint solutions and at least the doubling of network lifetime. Thus

Table 5.1: This table shows the mean number of solutions generated by the algorithm developed in Chapter 3 prior to its termination

Mean Number of Solutions			
	No. of Sensors		
No. of Targets	60	70	80
10	28.07	32.25	13.75
20	51.57	52.27	19.2
30	41.5	26.4	28
40	9.62	60.55	89.83

Table 5.2: This table shows the mean lifetime maximization results using solutions generated by the cutting-place algorithm in Chapter 3.

Mean Lifetime Magnification			
	No. of Sensors		
No. of Targets	60	70	80
10	1.56	1.39	1.38
20	1.63	1.59	1.2
30	1.44	1.2	1.16
40	1.38	1.68	1.47

we are led to believe that there is significant overlap between the feasible solutions generated by the cutting plane algorithm prior to termination. Inspection of the data corroborates this.

This leaves open the possibility of a delayed column generation approach to unearth further feasible solutions that share little if any sensors with the optimal solution found.

5.3 Lifetime Maximization with Mobility

In some applications sensors can potentially change location during the lifetime of the network. The extent of to which they can move about is a defining characteristic of the associated lifetime maximization problems. In some situations all of the sensors are capable of moving an indefinite number of times. Scenarios that meet this description are referred to as featuring total mobility. In other situations each sensor can move only once upon being deployed. These situations are given the moniker partial mobility. In yet other problem types the sensors available

for use are of different types: some are stationary while others are mobile. Networks meeting this criterion are called hybrid networks. We will restrict ourselves to the first two of these three network types.

5.3.1 Lifetime Maximization with Partial Mobility

The Problem of Lifetime Maximization with partial mobility allows us to move each sensor at most once before it is used. This is equivalent to a situation in which we have available to us a fixed number of sensors and a list of candidate locations at which we can place the sensors. The problem can be reduced to choosing how many sensors, if any, to allocate to each possible location. We formulate an integer program that represents this problem.

We will express the problem in a form similar and analogous to that of LM above. With the stationary sensor network we were constrained by the fact that no sensor could use more energy than its capacity dictated. With the partial mobility problem we are constrained by the fact that the quantity of energy spent at a particular location is limited by the number of sensors that we deploy to that location. If we let S denote the number of sensors available, we have the following mixed integer linear program which represents the lifetime maximization problem for partial mobility networks.

$$\begin{array}{lrcll} \max & e^T t & & & \\ \text{subject to} & \sum_{j=1}^{N} p_{i,j} t_j & \leq & n_i a_i, \ \forall i & (LMPM) \\ & \sum_i n_i & = & S & \\ & t & \geq & 0 & \\ & n_i & \in & \mathbb{Z} & \end{array}$$

5.3.1.1 Results

For cases involving uniorm sensors this algorithm almost always returns the floor of the quotient of the number of sensors and the cardinality of the optimal solution i.e. the algorithm chooses to repeat the optimal solution as often as possible. Table 5.3 shows the mean lifetime magnification values obtained for different problems. Each of the problems featured sensors and targets placed on a square of

Table 5.3: Lifetime Magnification Results employing Partial Mobility

Mean Lifetime Magnification			
	No. of Sensors		
No. of Targets	60	70	80
10	11.33	14.75	14.63
20	8.43	10.73	14.4
30	9.07	11.2	12.2
40	9.61	9.82	11.17

side 500 according to a uniform random distribution. The operating radius of each sensor is 150. Further work on heterogeneous sensors can be undertaken if a similar phenomenon is in effect in those situations.

5.3.2 Future Work

The seemingly low lifetime maximization values for problems without mobility might possibly be improved upon in the future with the use of a delayed column generation approach. If the lifetime maximization attained through the approach is deemed to be inadequate we can try to find new configurations that prolong network lifetime that meet the appropriate feasibility requirements.

Using P to denote the constraint matrix of LM the dual problem to LM is

$$\begin{array}{lrcll} \min & a^T y & & & \\ \text{subject to} & P^T y & \geq & e & (LMD) \\ & y & \geq & 0 & \end{array}$$

If $\bar{y}$ is the optimal solution of the problem LMD and we were to find a vector $\bar{p}$ such that $\bar{y}^T \bar{p}$ has a value strictly less than 1 then the configuration corresponding to $\bar{p}$ would lead to an increase in network lifetime if $\bar{p}$ was added as an additional column to the constraint matrix in LM. We would need to verify that $\bar{p}$ actually represented a feasible solution to the problem at hand. That leaves us with the subproblem of minimizing the function $\theta = \bar{y}^T p$ subject to the appropriate feasibility constraints. An objective value less than 1 signifies the ability to prolong the network's lifetime.

Depending on the situation we might already have all the relevant feasibility constraints available. If the pool of configurations was found using the cutting-plane

algorithm developed earlier it may be the case that the new lifetime prolonging configuration isn't actually feasible as it may violate a constraint of the form (3.4) or (3.5) that has yet to be added to the model. If the configuration is feasible we can add it as a column to LM; if it is not we can add the appropriate constraints to the feasibility problem.

CHAPTER 6
Future Work

Thus far we have come up with cleaner IP formulations of several different types of sensor placement problems. These formulations are still prone to the usual difficulties associated with integer programming. It is intended that we will continue to explore means of increasing the problem sizes that we are able to solve in reasonable time. Much of this work will be done in the algorithmic aspect of the problem-solving process: there seems to be significant room for improving the algorithms in use. The following areas in particular seem to be promising

- Further investigation of how best to represent an area by discrete targets. The marginal utility of using discrete targets tends to plateau after a certain number of targets have been employed. Investigation of the relationship between the radii of coverage and the number of targets needed to represent the area adequately may well prove to be fruitful as the number of targets being covered is a significant factor in the tractability of a problem.

- Development of a fully functional mixed integer semidefinite programming solver. At present the sizes of the problems we can get good solutions for are not of much practical value.

- A rigorous investigation of how our methods hold up in situations in which the disk model is not obeyed. In many situations the disk model is not very realistic. The modularity of our approach would seem to make it well suited to be tried with a wider variety of network topologies.

- Investigation of how all of your algorithms perform in situations in which sensors and targets are not uniformly distributed or grid-based.

- Improving separation routines for the unique coverage problem. At present we simply add all the violated cuts that we can find at any given iteration. Many of these may be redundant and thus unnecessary. Also we intend to

investigate the possibility of dropping cuts that havent been active for a series of iterations. Presently, the biggest factor slowing down the runtime of the algorithm is the sheer number of constraints that are in play when the problems are being sent to the solver.

- Use of delayed column generation methods for solving problems involving lifetime maximization. At present there tends to be a lot of commonality among the feasible solutions that our algorithms during their time of execution. This is not ideal for the purpose of lifetime maximization. The constraints of the various types we have formulated however seem to be well suited to be used as the constraints of the subproblems in a delayed column generation approach to lifetime maximization.

References

[1] Xiaole Bai, Chuanlin Zhang, Dong Xuan, Jin Teng, and Weijia Jia. Low-connectivity and full-coverage three dimensional wireless sensor networks. In *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 145–154, New York, NY, USA, 2009. ACM.

[2] Piotr Berman, Bhaskar Dasgupta, and Eduardo Sontag. Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene networks. In *Proc. 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX*, pages 39–50. Springer Verlag, 2004.

[3] Bela Bollobas. *Graph Theory: An Introductory Course*. Springer-Verlag, New York, 1979.

[4] Samuel Burer and Renato D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (Series B)*, 95:2003, 2001.

[5] Samuel Burer and Renato D.C. Monteiro. Sdplr 1.03-beta user's guide (short version). Technical report, University of Iowa, 2009.

[6] Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wirel. Netw.*, 11(3):333–340, 2005.

[7] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sensor networks topologies. Technical report, University of California, Los Angeles, 2004.

[8] Krishnendu Chakrabarty, S. Sitharama Iyengar, Hairong Qi, and Eungchun Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Trans. Comput.*, 51(12):1448–1453, 2002.

[9] Chandra Chekuri, Kenneth L. Clarkson, and Sariel Har-Peled. On the set multi-cover problem in geometric settings. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 341–350, New York, NY, USA, 2009. ACM.

[10] Ai Chen, Santosh Kumar, and Ten H. Lai. Designing localized algorithms for barrier coverage. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 63–74, New York, NY, USA, 2007. ACM.

[11] Ai Chen, Ten H. Lai, and Dong Xuan. Measuring and guaranteeing quality of barrier-coverage in wireless sensor networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 421–430, New York, NY, USA, 2008. ACM.

[12] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *ACM Wireless Networks Journal*, pages 85–96, 2001.

[13] Fan R.K. Chung. *Spectral Graph Theory*. Springer-Verlag, New York, 2001.

[14] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10:1–11, 1998.

[15] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 3rd ed. edition, 2008.

[16] Reinhard Diestel. *Graph Theory*. Springer-Verlag, New York, 2000.

[17] Douglas W. Gage. Sensor abstractions to support many-robot systems. In *Proceedings of SPIE Mobile Robots VII*, pages 235–246, 1992.

[18] E. Garcia and P. Gonzalez de Santos. Hybrid deliberative/reactive control of a scanning system for landmine detection. *Robot. Auton. Syst.*, 55(6):490–497, 2007.

[19] Amitabha Ghosh and Sajal K. Das. Review: Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive Mob. Comput.*, 4(3):303–334, 2008.

[20] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. Springer-Verlag, New York, 2001.

[21] M. Grötschel, A. Martin, and R. Weismantel. Packing Steiner trees: a cutting plane algorithm and computational results. *Mathematical Programming*, 72:125–145, 1996.

[22] M. Grötschel, A. Martin, and R. Weismantel. Packing Steiner trees: polyhedral investigations. *Mathematical Programming*, 72:101–123, 1996.

[23] M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992.

[24] Yu Gu, Yusheng Ji, and Baohua Zhao. Maximize lifetime of heterogeneouswireless sensor networks with joint coverage and connectivity requirement. Technical report, Dept. of Computer Science, University of Science and Technology of China, 2009.

[25] Chih-fan Hsin and Mingyan Liu. Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms. In *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 433–442, New York, NY, USA, 2004. ACM.

[26] Chi-Fu Huang, Yu-Chee Tseng, and Hsiao-Lu Wu. Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. *ACM Trans. Sen. Netw.*, 3(1):5, 2007.

[27] CPLEX Optimization Inc. Using the cplex callable library, 1994.

[28] Rajagopal Iyengar, Koushik Kar, and Suman Banerjee. Low-coordination topologies for redundancy in sensor networks. In *MobiHoc '05: Proceedings*

of the 6th ACM international symposium on Mobile ad hoc networking and computing, pages 332–342, New York, NY, USA, 2005. ACM.

[29] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[30] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 96–107, New York, NY, USA, 2002. ACM.

[31] Loukas Lazos and Radha Poovendran. Stochastic coverage in heterogeneous sensor networks. *ACM Trans. Sen. Netw.*, 2(3):325–358, 2006.

[32] Benyuan Liu, Olivier Dousse, Jie Wang, and Anwar Saipulla. Strong barrier coverage of wireless sensor networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 411–420, New York, NY, USA, 2008. ACM.

[33] Mingming Lu, Jie Wu, Mihaela Cardei, and Minglu Li. Energy-efficient connected coverage of discrete targets in wireless sensor networks. *Int. J. Ad Hoc Ubiquitous Comput.*, 4(3/4):137–147, 2009.

[34] Seapahn Megerian, Farinaz Koushanfar, Gang Qu, Giacomino Veltri, and Miodrag Potkonjak. Exposure in wireless sensor networks: theory and practical solutions. *Wirel. Netw.*, 8(5):443–454, 2002.

[35] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, and Mani B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, pages 1380–1387, 2001.

[36] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley, New York, 1988.

[37] M. Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Math. Program.*, 45(1):139–172, 1989.

[38] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction (Monographs in Computer Science)*. Springer, 1985.

[39] J. J. Salazar. A note on the generalized Steiner tree polytope. *Discrete Applied Mathematics*, 100(1–2):137–144, 2000.

[40] M. Stoer and G. Dahl. A polyhedral approach to multicommodity network design. *Numerische Mathematik*, 68:149–167, 1994.

[41] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM.

[42] Di Tian and Nicolas D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 32–41, New York, NY, USA, 2002. ACM.

[43] M.J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[44] Guiling (Grace) Wang, Guohong Cao, and Piotr Berman. Bidding protocols for deploying mobile sensors. *IEEE Transactions on Mobile Computing*, 6(5):563–576, 2007. Fellow-La Porta, Thomas F.

[45] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 28–39, New York, NY, USA, 2003. ACM.

[46] D. P. Williamson, M. X. Goemans, M. Mihail, and V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15:435–454, 1995.

[47] L. A. Wolsey. *Integer Programming*. John Wiley, New York, 1998.

[48] Guoliang Xing, Xiaorui Wang, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Trans. Sen. Netw.*, 1(1):36–72, 2005.

[49] Y. Xu, J. Heidemann, and D. Estrin. Adaptive energy-conserving routing for multihop ad hoc networks. Technical report, USC/Information Sciences Institute, May 2000.

[50] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84, New York, NY, USA, 2001. ACM.

[51] Ting Yan, Yu Gu, Tian He, and John A. Stankovic. Design and optimization of distributed sensing coverage in wireless sensor networks. *ACM Trans. Embed. Comput. Syst.*, 7(3):1–40, 2008.

[52] Yinying Yang and Mihaela Cardei. Movement-assisted sensor redeployment scheme for network lifetime increase. In *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 13–20, New York, NY, USA, 2007. ACM.

[53] Yi Zou and Krishnendu Chakrabarty. Sensor deployment and target localization in distributed sensor networks. *ACM Trans. Embed. Comput. Syst.*, 3(1):61–91, 2004.