# SENSOR DEPLOYMENT IN DETECTION NETWORKS-A CONTROL THEORETIC APPROACH

by

AHMAD A. ABABNAH

B.S., Mutah University, Jordan, 2003
M.S., Arizona State University, USA, 2007

---

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2010

# Abstract

For any automated surveillance operation to be successful, it is critical to have sensing resources strategically positioned to observe, interpret, react and maybe even predict events. In many practical scenarios, it is also expected that different zones within a surveillance area may have different probability of event detection (or false alarm) requirements. The operational objective in such surveillance systems is to optimize resources (number of sensors and the associated cost) and their deployment while guaranteeing a certain assured level of detection/false alarm performance.

In this dissertation, we study two major challenges related to sensor deployment in distributed sensor networks (DSNs) for detection applications. The first problem we study is the sensor deployment problem in which we ask the following question: Given a finite number of sensors (with a known sensing profile), how can we deploy these sensors such that we best meet the detection and false alarm requirements in a DSN employing a specific information fusion rule? Even though sensor deployment has garnered significant interest in the past, a unified, analytical framework to model and study sensor deployment is lacking. Additionally, the algorithms proposed in literature are typically heuristic in nature and are limited to (1) simplistic DSN fusion architectures, and (2) DSNs with uniform detection/false alarm requirements. In this dissertation, we propose a novel treatment of the sensor deployment problem using concepts from optimal control theory. Specifically, the deployment problem is formulated as a linear quadratic regulator (LQR) problem which provides a rigorous and analytical framework to study the deployment problem. We develop new sensor deployment algorithms that are applicable to a wide range of DSN architectures employing different fusion rules such as (1) logical OR fusion; (2) value fusion; (3) majority decision fusion,

and (4) optimal decision fusion. In all these cases, we demonstrate that our proposed control theoretic deployment approach is able to significantly outperform previously proposed algorithms.

The second problem considered in this dissertation is the "self healing" problem in which we ask the following question: After the failure of a number of sensors, how can one reconfigure the DSN such that the performance degradation due to sensor loss is minimized? Prior efforts in tackling the self healing problem typically rely on assumptions that don't accurately capture the behavior of practical sensors/networks and focus on minimizing performance degradation at a local area of the network instead of considering overall performance of the DSN. In this work, we propose two self healing strategies the first approach relies on adjusting decision thresholds at the fusion center. The second approach involves sensor redeployment based on our control theoretic deployment framework. Simulation results illustrate that the proposed algorithms are effective in alleviating the performance degradation due to sensor loss.

# SENSOR DEPLOYMENT IN DETECTION NETWORKS – A CONTROL THEORETIC APPROACH

by

AHMAD A. ABABNAH

B.S., Mutah University, Jordan, 2003
M.S., Arizona State University, 2007

---

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2010

Approved by:

Major Professor
Balasubramaniam Natarajan

# Copyright

# Abstract

For any automated surveillance operation to be successful, it is critical to have sensing resources strategically positioned to observe, interpret, react and maybe even predict events. In many practical scenarios, it is also expected that different zones within a surveillance area may have different probability of event detection (or false alarm) requirements. The operational objective in such surveillance systems is to optimize resources (number of sensors and the associated cost) and their deployment while guaranteeing a certain assured level of detection/false alarm performance.

In this dissertation, we study two major challenges related to sensor deployment in distributed sensor networks (DSNs) for detection applications. The first problem we study is the sensor deployment problem in which we ask the following question: Given a finite number of sensors (with a known sensing profile), how can we deploy these sensors such that we best meet the detection and false alarm requirements in a DSN employing a specific information fusion rule? Even though sensor deployment has garnered significant interest in the past, a unified, analytical framework to model and study sensor deployment is lacking. Additionally, the algorithms proposed in literature are typically heuristic in nature and are limited to (1) simplistic DSN fusion architectures, and (2) DSNs with uniform detection/false alarm requirements. In this dissertation, we propose a novel treatment of the sensor deployment problem using concepts from optimal control theory. Specifically, the deployment problem is formulated as a linear quadratic regulator (LQR) problem which provides a rigorous and analytical framework to study the deployment problem. We develop new sensor deployment algorithms that are applicable to a wide range of DSN architectures employing different fusion rules such as (1) logical OR fusion; (2) value fusion; (3) majority decision fusion,

and (4) optimal decision fusion. In all these cases, we demonstrate that our proposed control theoretic deployment approach is able to significantly outperform previously proposed algorithms.

The second problem considered in this dissertation is the "self healing" problem in which we ask the following question: After the failure of a number of sensors, how can one reconfigure the DSN such that the performance degradation due to sensor loss is minimized? Prior efforts in tackling the self healing problem typically rely on assumptions that don't accurately capture the behavior of practical sensors/networks and focus on minimizing performance degradation at a local area of the network instead of considering overall performance of the DSN. In this work, we propose two self healing strategies the first approach relies on adjusting decision thresholds at the fusion center. The second approach involves sensor redeployment based on our control theoretic deployment framework. Simulation results illustrate that the proposed algorithms are effective in alleviating the performance degradation due to sensor loss.

# Table of Contents

# List of Figures

# List of Tables

# List of Symbols

- $\mathcal{G}$: Two dimensional grid of points.

  $N_x$/ $N_y$: Number of grid points along the x/y-axis.

- $\mathbf{p}_f^{req}$: False alarm probability requirements vector of size $N_x N_y \times 1$.

- $\mathbf{p}_d^{req}$: Detection probability requirements vector of size $N_x N_y \times 1$.

- $\mathbf{p}_m^{req}$: Miss probability requirements vector of size $N_x N_y \times 1$.

- $p_f^{req}(j)$: False alarm requirement at the $j$-th grid point.

- $p_d^{req}(j)$: Detection probability requirement at the $j$-th grid point.

- $\mathbf{p}_m^{req}(j)$: Miss probability requirement at the $j$-th grid point.

- $K$: Number of available sensors for deployment.

- $\mathbf{p}_f^k$: Achieved false alarm probabilities vector of size $N_x N_y \times 1$, when $k$ sensors are deployed.

- $\mathbf{p}_d^k$: Achieved detection probabilities vector of size $N_x N_y \times 1$, when $k$ sensors are deployed.

- $p_f(j, k(j))$: Achieved false alarm probability at the $j$-th grid point, with $k(j)$ sensors covering that point.

- $p_d(j, k(j))$: Achieved detection probability at the $j$-th grid point, with $k(j)$ sensors covering that point.

- $\mathbf{D}$: Deployment vector of size $N_x N_y \times 1$.

- $R$: Radius of sensor effective coverage area.

- $\tau$: Exponential sensing model decay parameter.

- $A(d)$: Signal amplitude as a function of the distance $d$.

- $S(d)$ : Signal energy as a function of the distance $d$.

- $\mathbf{x}_k$: State of the dynamical control system at the $k$-th discrete step. A vector of size $N_x N_y \times 1$.

- $\mathbf{u}_k$: Control vector at the $k$-th discrete step. A vector of size $N_x N_y \times 1$.

- $J$: Scalar cost function.

- $\mathbf{Q}_k$, $\mathbf{R}_k$: Non-negative square weighting matrices of dimensions $N_x N_y \times N_x N_y$ at the $k$-th step.

- $\mathbf{Q}_f$: Non-negative square weighting matrix of dimensions $N_x N_y \times N_x N_y$ at the final step.

- $\boldsymbol{\lambda}_k$: Lagrangian multiplier vector of size $N_x N_y \times 1$ at the $k$-th discrete step.

- $H_k$: Scalar Hamiltonian function at the $k$-th discrete step.

- $p_m((x, y), (i, j))$: Probability a sensor at point $(i, j)$ misses a target located at the $(x, y)$ grid point.

- $M(x, y)$: Overall miss probability at point $(x, y)$ of the grid.

- $m(x, y)$: Logarithmic overall miss probability at point $(x, y)$ of the grid.

- $\kappa$: Medium propagation constant.

- $N_i$: Measurement noise at the $i$-th sensor.

- $R_c$: Collaboration radius.

- $G(x, n)$: Cumulative distribution function (CDF) of the Chi-Square distribution with $n$ degrees of freedom at point $x$.

- $p_f^s$: Individual sensor false alarm rate.

# Acknowledgments

I would like to express my gratitude to Prof. Bala Natarajan. I thank him not only for his academic role as my advisor but for his friendship and support. The advice that I received from him always proved useful and I hope that they will guide me in my career later in life.

I would also like to thank Prof. Caterina Scoglio, Prof. Noel Schulz, Prof. Jim Neil and Prof. Brett Esry for serving on my committee and for their valuable suggestions.

My thanks go as well to all of my friends from the WiCom group. The discussions that I had with each one of them helped enrich my horizon. I owe much gratitude as well to my friends at the Sunflower Networking Group for their friendship and support. I would also like to thank my friends Mwaffaq Otoom and Mohammed Gharaibeh for their encouragement. Special thanks goes to my friend Dustin Newton for being the best roommate one can have.

Last but not least, I want to thank my family back in Jordan; my mother – Khawla, my brothers – Mohammad and Hamza, my sisters – Manal, Suhair, Amani, Saba, Areen and Hadeel and my aunt – Muntaha for their patience, encouragement and unquestioned support.

# Dedication

To my late father *Abdel-Majied Ababneh*

my mother *Khawla Ebbini*

# Chapter 1

# Introduction

In this chapter, we provide a brief overview of distributed sensor networks - the primary topic of interest in this dissertation. We discuss in section 1.1, the sensor deployment and the self healing challenges associated with detection networks. In section 1.2, we review prior efforts related to sensor deployment and self healing problems. We briefly discuss the limitations and unanswered questions in these two areas. In section 1.3, an overview of our proposed approaches to solving these two problems is discussed. Finally, in section 1.4, we outline our contributions to the areas of sensor deployment and self healing.

## 1.1 Background

Distributed sensor networks (DSNs) consist of a number of sensor nodes that are deployed in a region of interest (ROI) in order to perform a given task [1]-[3]. In general, a typical sensor node consists of four main units; (1) sensing unit (to measure the physical quantity of interest); (2) processing unit (to extract information from measurements); (3) communication unit (to send and share data with other sensors or a fusion center), and (4) power supply unit (e.g., battery, solar cell). A sensor node may also be equipped with other accessories such as location finder and actuators [1]. DSNs can be used in many applications across multiple domains. The operational capability of DSNs include (1) detection of rare events; (2) monitoring of continuous events; (3) tracking of events, and (4) control of processes. As a result, DSNs are suitable for use in many military and civilian applications. A few

examples are given below:

1. Defense: A classic example is the PinPtr system [4] where acoustic sensors are used to detect and locate snipers in the battle field.

2. Agriculture: The LOFAR-Agro project [5] employs a network of sensors to monitor micro-climates in crop fields with the goal of maximizing the crop yield.

3. Health care: Sensor networks are used to monitor sleeping patterns of infants and patients in hospitals. The goal of these applications is to gather medical data or minimize emergency response time.

4. Environment: DSNs are used to monitor fundamental processes, such as volcanic activity (e.g., the Reventador project [6]) and plant growth (e.g. Macroscope of redwood project [7]).

In this dissertation, we are interested in sensor networks for detection applications. Detection refers to the process of making a yes/ no decision regarding the existence of a certain phenomena of interest (e.g., target). In fact, the earliest application of sensor networks was for event/target detection in defense applications. For example, the sound surveillance system (SOSUS) deployed during the Cold War consisted of a network of acoustic sensors placed on the bottom of the ocean to detect Soviet submarines [3]. In some applications (e.g., surveillance systems) the sole goal of the system is to perform detection. Moreover, in most DSN applications the first step is usually detection. For example, in tracking and classification systems, one can not perform tracking nor classification without first asserting the presence of the target or phenomena of interest.

The performance of a detection system is usually specified in terms of its overall false alarm and detection probability. False alarm probability refers to the probability that a detection decision of (yes) is made when the phenomena of interest is absent. False alarms can occur due to imperfection in sensor design or noise and are undesirable. Detection probability refers to the probability that a (yes) detection decision is made when the phenomena

of interest is present. Obviously, correct detection decision are always desirable. Maximizing the overall detection probability and minimizing the overall false alarm probability are competing objectives. Therefore, the end user of a detection system usually provides a set of minimum detection requirements to satisfy and a set of false alarm requirements that are not to be exceeded.

For any detection system to be successful, it is critical to have sensing resources (i.e., sensors) strategically deployed such that performance requirements are met. In a DSN employing low-powered sensors, this becomes more critical since low-powered sensors have a limited sensing range. Therefore, the detection performance is highly dependent on the spatial distribution of sensors within the region of interest. Moreover, having nonuniform requirements exacerbates the importance of having an intelligent deployment strategy that takes this into account. In many situations (e.g., battlefield), a region of interest might have various subregions with different levels of importance assigned to each subregion. For example, within a battle zone, areas around control centers with mission critical staff and infrastructure may require a higher level of protection relative to peripheral regions. Additionally, in some networks (e.g., underwater and underground network), sensors are expensive to deploy and maintain. Therefore, it is advantageous to develop a deployment strategy that takes the limited number of available sensors into consideration as well as attempting to use as few number of sensors as possible. Finally, the deployment strategy should also take into account the fusion rule used in the network. The fusion rule refers to the method used to combine information collected by sensors to make the overall detection decision. In summary, an intelligent deployment strategy should take the following factors into consideration

1. Sensing capability of sensors (i.e, sensing profile).

2. Number of available sensors.

3. Performance requirements as set by the end user.

4. Fusion rule employed

In this dissertation, we are interested in addressing two challenges related to sensor deployment. First, we attempt to answer the following question: *Given a finite number of sensors (with a known sensing profile), how can we deploy these sensor such that we best meet the detection and false alarm requirements in a DSN employing a specific information fusion rule?* Secondly, we investigate methods to mitigate the effects of losing sensors. While an adequate number of sensors can be deployed to meet certain performance requirements, losing sensors can lead to less than optimal performance. Loss of sensors can happen due to exhausted power supply (e.g., battery) or damage due to the nature of the deployment environment (e.g., deep sea, battlefield). Furthermore, it might be neither economical, safe or practical to replace lost sensors. Thus, it is important to devise strategies to mitigate performance degradation due to loss of sensors without the addition of new sensors. In networks, this feature is referred to as the "self-healing" capability of the detection system. In this dissertation, our goal is to develop self healing strategies based on network reconfiguration.

In the following section, we review prior efforts related to sensor deployment in detection networks.

## 1.2 Prior Efforts and Motivation

In general, most sensor deployment efforts start with a grid model for the ROI. This alleviates the complexity associated with considering a space continuum. In this framework, usually the goal is to find the best $M$ positions out of $N$ possible positions at which sensors can be deployed in order to minimize a given cost function. If we assume that only 1 sensor can be deployed at a point, then the number of sensors at any point in the ROI takes only one of two values (0 or 1). In this case, the sensor deployment problem is a binary integer programming problem. Integer programming problems are NP-hard, which implies that the computational complexity associated with solving such a problem increases

exponentially with the number of variables (i.e., $N$ and $M$ in the deployment problem). One possible solution for an integer programming problem is enumeration. All possible solutions are listed and the value of the cost function corresponding to each solution is calculated. The optimal solution is the one that has the minimum cost function value. The enumeration approach requires large storage and computational capabilities which prohibits its use except for problems of a small search space. For example, in a deployment problem if $N = 20$ sensors and $M = 900$ possible locations the set of enumerated solutions will consist of $\binom{900}{20} = 9.8033 \times 10^{55}$ possible sensor positions configurations. For integer programming problems, one can use branch and bound methods to find an optimal solution [8] and [9]. However, the computational complexity of these methods can be quite high. In fact, in the worst case scenario the computational complexity of branch and bound methods is equal to that of the enumeration method.

Sensor deployment problem has been investigated in the context of different applications and approximate solutions have been proposed for the specific problem at hand. For example, [10] presents a survey of deployment methods used in aerospace industry arena. A range of stochastic optimization algorithm that include simulated annealing and genetic algorithms are used to deploy sensors in [11] and [12]. Some of the drawbacks associated with these algorithms include their sensitivity to initial conditions, reliance on heuristics and no guarantee of optimality.

The deployment problem was also studied in the context of distribution networks (e.g., water networks) in which a DSN is used to detect contaminants (e.g., chemical, biological) [13]-[15]. The works in [13] and [14] focus primarily on deploying a fixed number of sensors in order to maximize the network exposure to contaminants. This objective enables an early warning and a faster response time to contamination. The deployment problem in [13] and [14] is modeled as an integer programming problem, which has a high computational complexity as described earlier. In [15], two variations of the sensor deployment problem are investigated. The first version is the sensor constrained problem– here a fixed number

of sensors is to be deployed to minimize a contaminant's detection time. This problem was shown to be equivalent to the asymmetric k-center problem which is an NP-hard problem. The second version of the deployment problem is the time constrained problem– here it is required to detect contaminants within a given time limit while minimizing the number of sensors employed. This problem was shown to be equivalent to a minimum dominating set problem, which is also NP-hard. One can use several approximation algorithms to solve both the asymmetric k-center and minimum dominating set problems. The works in [13]-[15] were based on the assumption that sensors used are perfect point sensors. Point sensors do not have an effective coverage area, this means that a sensor detects a chemical only if it passes through the sensor. One further assumption is that a sensor will always (i.e., 100% of the time) detect the contaminant regardless of its concentration, which is an unrealistic assumption. Furthermore, neither sensor collaboration nor false alarm requirements are incorporated in the deployment framework.

The problem of sensor deployment with sensors having an effective coverage area is presented in [16] -[22]. Examples of sensors that have an effective coverage area include acoustic, seismic and infrared sensors. In these works, the focus is on determining the minimum number of sensors, as well as their positions, needed to satisfy detection requirements. In [18] and [19], the deployment problem is studied with the assumption that sensors have a binary sensing model. In such a model, a target/phenomena is always detected by a sensor if it falls within the sensor's detection radius (i.e., coverage area). This assumption enables the treatment of the deployment problem as a coverage problem. However, in reality a sensor makes a detection decision by comparing a noisy measurement to a detection threshold. The presence of noise implies that it is not always possible to make a correct detection decision. Therefore, the binary sensing model is an unrealistic sensing model.

With the same goal of minimizing the number of sensors used, the authors in [20] and [21] adopt a probabilistic sensing model. In this model, the probability of a sensor detecting a target is an exponentially decaying function of the distance separating them. This is a

reasonable model, since the strength of the sensor's measurement is also a decaying function of distance. Sensors are assumed to have enough computational capabilities, enabling them to make local decisions regarding the presence or absence of a target/phenomena. Both, [20] and [21], employ a simple detection scheme in which a detection at a point within the ROI is declared if at least one sensor that falls within the detection radius of that point reports a detection decision. Though a distance dependent sensing model is more practical than a binary one, it still does not incorporate the effects of noise. Due to the presence of noise, a sensor detection decision might be a false alarm. Neither [20] nor [21] incorporates false alarm rates of individual sensors nor system's overall false alarm requirements. Furthermore, the proposed deployment algorithms; Min_Miss in [20] and Diff_Deploy in [21] are heuristic in nature and have no guarantee of optimality.

In order to account for false alarm probability, one has to incorporate information fusion across sensors. Recently in [22], false alarm and detection probability requirements were incorporated in the sensor deployment problem by assuming a value fusion architecture. The goal in [22] is to deploy as few sensors as possible in order to satisfy uniform false alarm and detection requirements. Sensors with a fixed collaboration radius report their noisy energy measurements to a fusion center (FC). The FC decides on the presence or absence of a target by comparing the average of the measurements against a detection threshold. This simple fusion scheme is known as value fusion. However, the resulting false alarm and detection probabilities are nonlinear with respect to the number and positions of the sensors within the ROI, which complicates the study of the deployment problem in such systems. Therefore, the authors in [22] propose the use of the constrained simulated annealing (CSA) algorithm [23], a stochastic optimization algorithm, to solve the deployment problem. However, the high computational complexity associated with the CSA renders it impractical to use in large scale networks. Therefore, a divide and conquer (D&C) algorithm was introduced as a low complexity alternative to the CSA algorithm. In the D&C algorithm, multiple instances of the CSA algorithm are implemented but with a smaller search space for each instance in

comparison to a single instance of the CSA algorithm with the full search space. Though the D&C algorithm has a lower complexity than the CSA, it is a heuristic algorithm with no guarantee of optimality. Furthermore, one major drawback of the D&C algorithm is it's limited applicability to systems with uniform false alarm and detection requirements.

The value fusion approach presented in [22] assumes that data from sensors can be accurately shared with a fusion center. However, in practice, due to bandwidth and power constraints it is reasonable to (1) expect local sensors to make preliminary detection decisions based on their measurements, and (2) transmit the decisions to the fusion center. This is referred to as decision fusion. In [24], the authors study the deployment problem when decision fusion is used instead of value fusion. Instead of detection probability, the authors use the path exposure criteria and the algorithm is heuristic in nature. In [25], deployment occurs along a one dimensional line and the deployment positions are uniformly spaced. The authors find the optimal spacing distance between sensors so as to satisfy detection and false alarm requirements. However, deployment in a 1-D space is not of much practical use. Additionally, mandating that the sensors are equally spaced might require more sensors than are really needed to satisfy requirements. In summary, prior efforts in sensor deployment strategies that account for decision fusion are either heuristic or overly simplistic.

As stated in section 1.1, in this dissertation, we also investigate self healing methods in DSNs. There have been extensive efforts in this area over the past decade. Employing sensor mobility as an approach for self healing has been proposed and studied extensively in several works [26]-[29]. The main objective in [26] is to maximize the coverage area in the region of interest using mobile sensors. In [26], the area of interest is divided into a set of Voronoi polygons in which each polygon corresponds to a single sensor. The detection task in each polygon falls solely on the sensor that lies within that polygon. In the event of a coverage hole, the authors propose three sensor movement algorithms. The aim of these algorithms is to move sensors from areas with high sensor density to areas with low densities which results in a more uniform sensor distribution. A more uniform sensor density

has the effect of improving the network coverage. In [27], the proposed movement algorithm is used to counter the effects of coverage holes or inadequacy of detection performance (i.e., self-healing). The authors assume a hybrid network consisting of both static and mobile sensors. Mobile sensors move to either; (1) areas with coverage holes (to eliminate them) or (2) areas where static sensors suspect that a target exists (to confirm the existence of such a target). The authors propose a distributed path planning algorithm that guides sensors in their movement. The algorithm minimizes a linear combination of cost functions. Upon running the algorithm, a mobile sensor is (1) pulled towards the target's area (or coverage hole), (2) pushed away from covered areas and (3) pushed away from areas covered by other mobile sensors. In the above mentioned efforts [26]-[27], the performance metric used is the coverage metric. The use of this metric has the advantage of simplifying the treatment of the self healing problem. However, coverage fails to account for the achieved performance of the network with regard to false alarm/ detection requirements. For example, in [26], the movement algorithm is not governed by the exact deviation between achieved and required network performance. In [28] and [29], the authors propose a sensor movement algorithm that aims at meeting given false alarm/ detection requirements while minimizing the total moving distance. Though self-healing is not the specific goal of this algorithm, it can easily be applied to self-healing scenarios. If static sensors make an initial detection decision in an area with a lower than required detection probability, a movement schedule is sent to mobile sensors. Mobile sensors move (reactively) to the area of interest and collaborate with static sensors to achieve the required detection performance. This can be viewed as a self-healing algorithm if the performance requirements are not met initially due to the loss of one or more static sensors. The drawback of this approach is that it does not incorporate the performance degradation experienced at the original mobile sensor locations. Though the movement of sensors can improve the detection performance at the area of interest, it can cause performance degradation at other areas. Therefore, it is desirable to develop a balanced self-healing sensor movement strategy that takes the network's overall performance

(after sensor movement) into account instead of focusing on a limited area of interest.

In summary, even though sensor deployment and self healing problems have garnered significant research interest over the past decade, there are several unanswered questions that still remain. The drawback of prior works that serve as the motivation of this dissertation are summarized below:

1. A unified framework to model and analyze sensor deployment problem is currently lacking.

2. Most of the proposed deployment algorithms are heuristic in nature and offer no insight into the deployment problem. There is a need for a structured and rigorous treatment of sensor deployment.

3. The treatment of sensor deployment problem in literature is limited to either a simple fusion rule or satisfying uniform performance requirements. Both these simplistic assumptions are unrealistic. There is a strong need for analyzing detection networks with nonuniform requirements along with the consideration of practical fusion rules.

4. The majority of research in self healing are based on assumptions that do not accurately capture the behavior of practical sensors and networks. The focus of prior efforts in self healing is to minimize performance degradation in a local area of the network. The overall network performance is often not accurately accounted for.

In the next section, we present an overview of our proposed approach that attempts to address the unanswered questions in the area of sensor deployment and self healing.

## 1.3 Overview of Dissertation

In this dissertation, we propose a novel treatment of the sensor deployment problem. Using concepts from optimal control theory, the deployment problem is modeled as an optimal control problem [30]-[33]. Specifically, the deployment problem is formulated as a linear

quadratic regulator (LQR) problem which is a very well-behaved and studied problem in optimal control theory literature [30]-[34]. The proposed LQR formulation offers a *unified treatment* of the sensor deployment problem. In addition, our LQR formulation provides *a rigorous and analytical framework* to study the deployment problem. A plethora of analytic solution methods for the LQR problem have been proposed in optimal control theory literature. This is in contrast to prior efforts in sensor deployment that are mainly heuristic in nature. Moreover, using the LQR formulation, we are able to consider deployment in networks with *nonuniform performance requirements*. This greatly amplifies the impact of our work, since prior efforts are largely restricted to uniform requirements. Finally, we show that our proposed approach can be applied to networks employing various fusion rules (e.g., value fusion, optimal decision fusion) and sensing models. The steps involved in our LQR formulation of sensor deployment (irrespective of fusion rule used) are as follows

1. The deployment problem is viewed as a sequential problem (i.e., sensors are sequentially deployed). This is advantageous, since it alleviates the high computational complexity associated with deploying sensors in parallel (i.e., simultaneous deployment).

2. We develop a linear approximation of the effect of each additional sensor deployment on the overall performance of the network. In this linearized model, the state (i.e., detection performance) of the network is driven by a control vector that corresponds to the sensor positions. This linearization is a significant step in the formulation and differs based on the fusion rule considered.

3. Since the goal is to satisfy performance requirements, we adopt a squared error cost function as an objective function to minimize. This cost function, measures achieved performance deviations from performance requirements.

4. Using the linearized deployment model in conjunction with the squared error cost function, we are able to formulate the deployment process as a linear quadratic regulator problem (LQR).

We also propose a novel treatment of the self healing problem in sensor networks. Specifically, we propose two self healing approaches which are listed below

1. Decision threshold adjustment: In this approach we update the decision threshold at the fusion center to account for the loss of sensors.

2. Sensor redeployment: If the detection performance after decision threshold adjustment is not satisfactory, we resort to adjusting the spatial distribution of sensors (i.e., redeployment). Sensor redeployment is modeled as an LQR problem in a similar way to what we have discussed earlier.

Our proposed approaches to self healing can be applied to practical detection networks. They can be equally applied to networks employing data fusion or decision fission rules. In addition, using the LQR formulation for redeployment ensures that the overall performance of the network is taken into account. This is in contrast to existing redeployment approaches.

In the next section, we present a summary of our contributions to the problems of sensor deployment and self healing.

## 1.4 Contributions

The major contributions of this dissertation are listed below:

- For the first time, we propose an analytic formulation of the sensor deployment problem. Specifically, the deployment problem is modeled as a linear quadratic regulator (LQR) problem. [see chapter 3, [35]-[43]]

In chapter 4, we consider a DSN employing the logical OR fusion rule, where only detection requirements are incorporated. For this system, we

- Propose a novel LQR-based sensor deployment algorithm along with a low complexity heuristic deployment alternative.

- Simulation results indicate that the proposed algorithms outperform existing methods by using 10% to 30% fewer number of sensors to satisfy detection requirements.

These contributions are published in our papers [35] and [36].

In chapter 5, we study deployment in a DSN employing value fusion with energy measurements as in [22], where both false alarm and detection requirements are considered. For such a system, we

- Derive a novel linear approximation of the effect of a single sensor deployment on the detection performance of the network.

- Propose an LQR-based sensor deployment algorithm using the linearized model. In addition, we develop a low complexity alternative to the LQR-based deployment algorithm.

- Illustrate that the proposed algorithms are effective in addressing both uniform as well as nonuniform false alarm and detection requirements. Simulation results show that the proposed algorithms use as few as 30% fewer sensors than the D&C algorithm [22].

- Derive the optimal collaboration radius that determines which sensor measurements are to be combined. In contrast to prior efforts which assume a fixed collaboration radius, we dynamically update the collaboration radius. The use of a dynamic collaboration radius can save up to 45% of the number of sensors needed to meet performance requirements with a fixed collaboration radius.

- Extend our proposed algorithms to a DSN employing value fusion with amplitude instead of energy measurements (see Appendix A).

These contributions are discussed in detail in our papers [37], [38] and [39].

In chapter 6, we investigate sensor deployment in a DSN employing the majority decision fusion rule.

- We propose a novel approximation of the change in the DSN detection performance resulting from deploying an additional sensor. This is done using results from non-parametric statistical theory.

- For the first time, we introduce a novel LQR-based sensor deployment algorithm that is especially tailored for decision fusion networks. Simulation results indicate that in comparison to a greedy deployment algorithm, the proposed algorithm can save up to 40% in the number of sensors needed to satisfy the same detection requirements.

- The proposed algorithm can be applied to networks with nonuniform performance requirements.

A detailed discussion of these contributions can be found in our papers [40] and [41].

In chapter 7, we examine the deployment problem when the fusion rule employed by the DSN is the optimal decision fusion rule. In this system, our contributions can be summarized as follows:

- We propose a novel closed form approximation of the false alarm and detection probabilities in optimal decision fusion networks.

- We use the proposed approximation to linearize the effect of a single sensor deployment on the overall detection probability of the network.

- For the first time, we introduce a novel LQR-based sensor deployment algorithms for networks employing the optimal decision fusion rule. Simulation results illustrate that it is possible to save up to 45% in the number of sensors required to meet performance requirements by using the proposed algorithm relative to a greedy deployment algorithm.

These contributions can be found in our paper [42].

In chapter 8, we study the self healing problem in an optimal decision fusion network.

- We propose a self healing approach based on adjusting the decision threshold used at the fusion center.

- For the first time, we propose an LQR-based sensor redeployment algorithm that takes the network's overall performance requirements into account. Simulation results illustrate that the proposed approaches can effectively counter the performance degradation resulting from lost/failed sensors.

Our contributions to the study of the self healing problem can be found in our paper [43].

In addition to the contributions listed above, the research presented in this dissertation can serve as a planning tool for many practical DSNs. The proposed approaches can be used to (1) determine the minimum number of sensors needed to satisfy some given performance requirements, and (2) quantify the performance of an existing deployment with low computational complexity.

The rest of this thesis is organized as follows: In chapter 2, we give a brief background of detection in distributed sensor networks. In chapter 3, we introduce the proposed mathematical formulation of the deployment problem. We also provide an overview of optimal control theory in general and the linear quadratic regulator (LQR) problem in particular. In addition, we motivate the use of the LQR formulation in modeling the deployment problem and discuss LQR solution methods. In chapter 4, we examine the deployment problem in a DSN employing the logical OR fusion rule. Sensor deployment in a DSN which employs value fusion with energy measurements is investigated in chapter 5 and the study is extended to the case of amplitude measurements in Appendix A. In chapters 6 and 7, we examine the sensor deployment problem when the majority and optimal decision fusion rules rule, respectively, are used. Our proposed approaches to the self healing problem are discussed in chapter 8. Finally, in chapter 9 we summarize our major contributions and discuss ideas for future work.

# Chapter 2

# Detection in Sensor Networks – Fundamentals

Detection usually refers to making a yes/no type decision regarding the existence of a phenomenon of interest (e.g., gas, intruder, plane). Typically, the decision maker is not in direct contact with the phenomena of interest and relies on noisy measurements from sensors to make a detection decision. Due to noise, the decision making process is susceptible to errors which are not desirable. Therefore, in the decision making process a cost/ performance metric is usually assigned to these errors. The theory of hypothesis testing, also known as statistical inference, provides a mathematical framework for studying the detection process. Hypothesis testing is mainly concerned with the design of optimal decision rules that either minimize a given function of the cost of errors or keep these errors within acceptable levels. One approach to increase detection reliability is by using multiple sensors (i.e., sensor network). Information (e.g., measurements, decisions) from multiple sensors can be combined/ fused in different ways (e.g., centralized or decentralized detection). One major concern in sensor networks is to design optimal decision rules at the local sensor levels and an optimal fusion rule at the the global level (e.g., fusion center) that meet some performance requirement. The principles of hypothesis testing theory can be applied to design such rules.

In section 2.1 of this chapter, we review concepts of hypothesis testing theory. We focus on the design of optimal decision rules under the Bayesian and Neyman-Pearson

frameworks. Section 2.2.1 provides an overview of centralized detection in sensor networks and the design of the optimal fusion rule in such networks. Decentralized detection is the focus of sections 2.2.2, 2.2.3 and 2.2.4. An overview of decentralized detection is provided in section 2.2.2. Various network architectures that are usually used in decentralized detection are discussed in section 2.2.3. The design of optimal fusion rules in decentralized detection networks in the Bayesian and Neyman-Pearson approaches is the focus of section 2.2.4. Finally, in section 2.3, we summarize the ideas discussed in this chapter.

## 2.1 Hypothesis Testing

In hypothesis testing, we are faced with the problem of characterizing the state of an event or phenomena of interest. The state of the phenomena can be any number of discrete states, with each state constituting a hypothesis. We denote the set of $M$-ary hypotheses as $\mathcal{H} = \{H_0, H_1, \ldots, H_{M-1}\}$. We can have any $M$ number of hypotheses, however, we focus our discussion on binary hypotheses (i.e., $M = 2$) problems. Discussions regarding binary testing can be easily extended to the case of $M$-ary hypotheses testing.

Following [44], the basic elements of a hypothesis testing problem are shown in Fig.2.1. The measurement of the phenomena goes through a probabilistic transition mechanism (e.g., noisy sensor measurement). Through this transformation we are able to indirectly observe the phenomena of interest. Based on the resulting observation space, we can then design decision rules that allow us to determine the state of the phenomena.

Usually, the decision rule is designed in order to minimize a certain criteria of interest (e.g. probability of error). One additional design factor is the amount of available prior information (e.g., prior probability of a hypothesis being true). Next, we briefly discuss two decision rule design approaches.

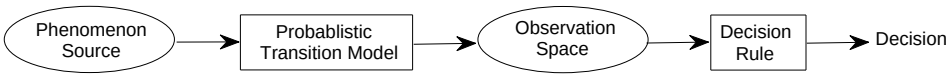



**Figure 2.1**: *Components of hypothesis testing*

### 2.1.1 Bayesian Approach

In the Bayesian approach, we assume the priori probabilities $P_0/P_1$ of hypotheses $H_0/H_1$ are known. Let $\{C_{ij}, i, j = 0, 1\}$, denote the cost associated with making the decision that $H_i$ is present when in fact $H_j$ is true. In the Bayesian approach, we seek to minimize the average risk function $\mathcal{R}$, given as [44]

$$\mathcal{R} = \sum_{i=0}^{1} \sum_{j=0}^{1} C_{ij} P_j \Pr(H_i \text{ decided} | H_j \text{ true}). \tag{2.1}$$

Let $C_{10} > C_{00}$ and $C_{01} > C_{11}$ (i.e., making a wrong decision is more costly than making a correct one). Under such assumption, the optimal decision rule that minimizes $\mathcal{R}$ is a likelihood ratio test (LRT). The non-randomized LRT is given as [44], [45]

$$\text{Decision} = \begin{cases} H_0 & \text{, if } \Lambda(y) < \eta \\ H_1 & \text{, if } \Lambda(y) \geq \eta \end{cases} \tag{2.2}$$

where, $y$ is the observation and $\Lambda(y)$ and $\eta$ are the likelihood ratio and decision threshold, respectively. The likelihood ratio and threshold are given as

$$\Lambda(y) = \frac{p(y|H_1)}{p(y|H_0)} \tag{2.3}$$

$$\eta = \frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})}. \tag{2.4}$$

As a special case, if $C_{00} = C_{11} = 0$ and $C_{01} = C_{10} = 1$ then $\mathcal{R}$ is the average probability of error and is given as [45]

$$\mathcal{R} = P_0 P_f + P_1(1 - P_d) \tag{2.5}$$

where, $P_f$ and $P_d$ are the false alarm and detection probabilities, respectively. $P_f$ and $P_d$ are given as

$$P_f = \Pr(H_1 \text{ decided}|H_0 \text{ true}) \tag{2.6}$$

$$P_d = \Pr(H_1 \text{ decided}|H_1 \text{ true}), \tag{2.7}$$

and the miss probability $P_m = 1 - P_d$ (i.e., $P_m = \Pr(H_0 \text{ decided}|H_1 \text{ true})$).

It is not always possible to know or estimate the hypotheses prior probabilities or the costs. In a situation like this, the Bayesian approach is not applicable. In the next section, we examine the Neyman-Pearson approach, which can be applied when both prior probabilities and costs are not known.

### 2.1.2 Neyman-Pearson Approach

The Neyman-Pearson test can be stated as follows: for a given acceptable level of false alarm probability ($P_f = \alpha$), find the decision rule that maximizes the detection probability $P_d$ [44],[45]. If $y$ denotes the observation, then the detection decision according to the optimal Neyman-Pearson test is given as follows [44]-[46]

$$\text{Decision} = \begin{cases} H_0 & \text{, if } \Lambda(y) < \eta \\ H_1 & \text{, if } \Lambda(y) \geq \eta \end{cases} \tag{2.8}$$

where, $\Lambda(y) = \frac{p(y|H_1)}{p(y|H_0)}$ is the likelihood ratio and $\eta$ is the decision threshold. The decision threshold $\eta$ is chosen such that the false alarm probability is less than or equal to $\alpha$

$$P_f = \Pr(\Lambda(y) \geq \eta|H_0) \leq \alpha \tag{2.9}$$

## 2.2 Detection using Multiple Sensors

The reliability of a detection system is usually measured in terms of the probability of error as in the Bayesian framework or the overall false and detection probability of the system as in the Neyman-Pearson approach. To increase the reliability of the detection process, a network of multiple sensors is usually employed [47]-[49]. Sensors can report their observations

to a fusion center (FC). Combining these measurements, the FC makes a local decision according to a predetermined fusion rule. This is known as centralized detection [50]. Due to increased computational capabilities of sensor, sensors can process their measurements and make local decisions. These decisions can be shared with other sensors or an FC to make a global detection decision. This in known as decentralized detection [50]. In this section, we briefly discuss these two detection schemes. We also discuss different network designs (i.e., architectures) that are commonly used in decentralized detection. Due to its prevalence in the literature of sensor deployment, we discuss the design of optimal fusion rules in a network employing a parallel architecture with a fusion center.

### 2.2.1 Centralized Detection

In centralized detection, sensors send their measurements without processing to the fusion center (FC). The (FC) combines the measurements to reach a global detection decision $u_0$ according to a specified decision rule.

Since sensors do not make local decision (i.e., no local decision rules), the focus in centralized detection is on the design of the fusion rule at the FC. From hypothesis testing in section 2.1, we know that the optimal fusion rule at the FC is the likelihood ratio test (LRT). If the network consists of $N$ sensors $\{s_i, i = 1, 2, \ldots, N\}$, then the optimal LRT rule is given as [44], [45]

$$u_0 = \begin{cases} 0 \text{ (i.e., } H_0 \text{ is decided)} & \text{, if } \Lambda(y_1, y_2, \ldots, y_N) < \eta \\ 1 \text{ (i.e., } H_1 \text{ is decided)} & \text{, if } \Lambda(y_1, y_2, \ldots, y_N) \geq \eta \end{cases} \tag{2.10}$$

where, $\eta$ is the decision threshold and $y_i$ is the measurement of the $i$-th sensor. The likelihood ratio $\Lambda(y_1, y_2, \ldots, y_N)$ is given as

$$\Lambda(y_1, y_2, \ldots, y_N) = \frac{p(y_1, y_2, \ldots, y_N|H_1)}{p(y_1, y_2, \ldots, y_N|H_0)}. \tag{2.11}$$

One common simplifying assumption is to assume that the measurements are conditionally independent, therefore the likelihood ratio can be expressed as

$$\Lambda(y_1, y_2, \ldots, y_N) = \prod_{i=1}^{N} \frac{p(y_i|H_1)}{p(y_i|H_0)}. \tag{2.12}$$

We note that the form of the decision rule depends on the nature of the conditional distributions $\{\mathrm{p}(y_i|H_j),\ i=1,\ldots,N,\ j=0,1\}$. The next example illustrates the structure of the optimal fusion rule when the hypothesis of interest is corrupted by additive Gaussian noise.

*Example 2.1*: Let the $i$-th sensor ($\{i=1,\ldots,N\}$) measurement model be as follows [45]

$$H_0: \quad y_i = n_i, \tag{2.13}$$

$$H_1: \quad y_i = a_i + n_i, \tag{2.14}$$

where, $n_i$ is a zero mean Gaussian noise with a variance of $\sigma^2$. The noise samples $\{n_i,\ i=1,\ldots,N\}$ are uncorrelated. $a_i$ is the signal amplitude at the $i$-th sensor. Then, using Eqn.(2.12) the optimal fusion rule is given as [45]

$$u_0 = \begin{cases} 0 \text{ (i.e., } H_0 \text{ is decided)} & \text{, if } \sum_{i=1}^{N} a_i y_i < \eta' \\ 1 \text{ (i.e., } H_1 \text{ is decided)} & \text{, if } \sum_{i=1}^{N} a_i y_i \geq \eta' \end{cases} \tag{2.15}$$

where, $\eta' = \sigma^2 \ln \eta + \sum_{i=1}^{N} \frac{a_i^2}{2}$ is the decision threshold. The decision rule can be further simplified if the signal amplitudes are equal $\{a = a_i,\ i=1,\ldots,N\}$ as follows

$$u_0 = \begin{cases} 0 \text{ (i.e., } H_0 \text{ is decided)} & \text{, if } \sum_{i=1}^{N} y_i < \frac{\eta'}{a} \\ 1 \text{ (i.e., } H_1 \text{ is decided)} & \text{, if } \sum_{i=1}^{N} y_i \geq \frac{\eta'}{a} \end{cases} \tag{2.16}$$

The new fusion rule is just the summation of the sensor measurements.

### 2.2.2 Decentralized Detection

Decentralized detection differs from centralized detection in the fact that sensors process their measurements and make local detection decisions. These decisions can then be shared with other sensors or sent to a fusion center depending on the network architecture. In comparison to centralized detection, the fusion center has less information about the phenomena of interest, which results in loss of detection performance. However, decentralized detection is advantageous when the network has a limited communication bandwidth and sensors have a limited power supply. In the sections that follow, we review decentralized detection network architectures. In addition, we discuss decision rule design at the sensor and fusion center levels when using Bayesian and Neyman-Pearson approaches.

### 2.2.3 Network Architectures

A network architecture refers to the way information between sensors are shared/combined in the network. There are different architectures that can be used in constructing a multi-sensor network. Examples of these architectures include: (1) parallel architecture with fusion; (2) serial architecture; (3) parallel architecture without a fusion center and (4) tree architecture. In what follows, we briefly discuss these architectures.

1. **Parallel architecture with a fusion center**

   This architecture is depicted in Fig.2.2. Based on the measurement $y_i$ of the $i$-th sensor ($i = 1, \ldots, N$), a decision $u_i = \gamma_i(y_i)$ is made according to the sensor's decision rule $\gamma_i$. The decisions from the $N$ sensors are then sent to a fusion center. The FC makes its final decision $u_0$ according to $u_0 = \gamma_0(u_1, \ldots, u_N)$, where $\gamma_0$ is the global decision rule. The nature of the local and global decision rules depend on the nature of the measurements (e.g., independent or correlated) available at the sensors. If the observations are independent, then the local and global decision rules $\{\gamma_0, \gamma_1, \ldots, \gamma_N\}$ are likelihood ratio tests (LRT) [44], [50]. However, calculating the corresponding decision thresholds $\{\eta_0, \eta_1, \ldots, \eta_N\}$ that minimize the average risk or maximize the detection probability is a computationally intensive process [51]. In the case that observations are correlated, finding the optimal decision thresholds is computationally intractable [50].

2. **Serial architecture**

   A network of $N$ sensors employing a serial architecture is shown in Fig.2.3. In this configuration, the $j$-th sensor sends its decision $u_j$ to the $j+1$-th sensor. The $j+1$-th sensor combines its own observation $y_{j+1}$ with $u_j$ to produce its decision. The decision $u_{j+1}$ is given as $u_{j+1} = \gamma_{j+1}(y_{j+1}, u_j)$, where $\gamma_{j+1}$ is the decision rule at the $j+1$ sensor. The global decision $u_0$ is the one made by the last sensor (i.e., $u_0 = u_N$), which implies that the last sensor effectively acts as the fusion center (FC). Similar to

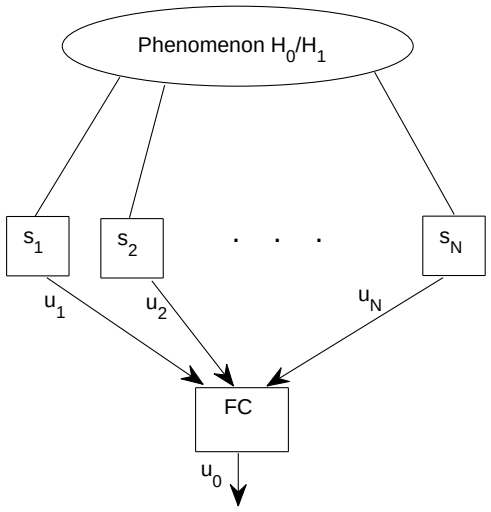



**Figure 2.2**: *Parallel architecture*

the case of parallel fusion, the local decision rules are LRT in the case of conditionally independent observations and are intractable to calculate when the observations are correlated [50],[52] and [51].

3. **Parallel architecture without a fusion center**

   This architecture is illustrated in Fig.2.4. We note that sensors do not communicate with each other and decisions are not fused. Rather, the decision costs at the sensors are coupled and a system wide optimization is carried out to minimize a given performance metric (e.g., probability of error). In general, the local decision rule at a sensor is an LRT. However, the optimal local decision threshold at a sensor depends on both the observations (i.e., data dependent) and decision thresholds at the other sensors [51]. In the special case of conditionally independent observations, a sensor's decision threshold is solely dependent on decision thresholds at the other sensors (i.e., data independent).

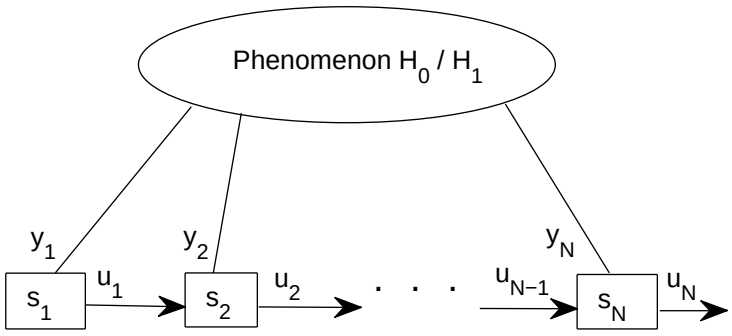



**Figure 2.3**: *Serial architecture*

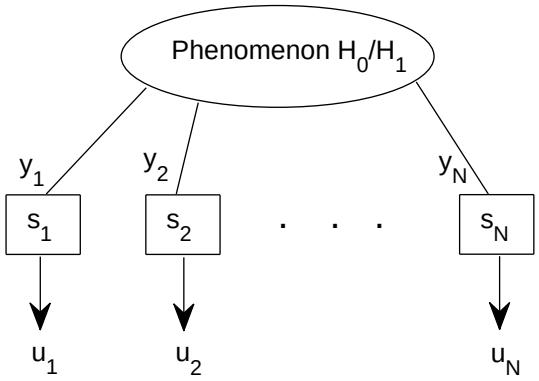



**Figure 2.4**: *Parallel architecture without a fusion center*

4. **Tree architecture**

Fig.2.5 depicts a network employing a tree architecture. In this architecture, sensors correspond to nodes in a directed acyclic graph where the fusion center is the root of this graph [44].

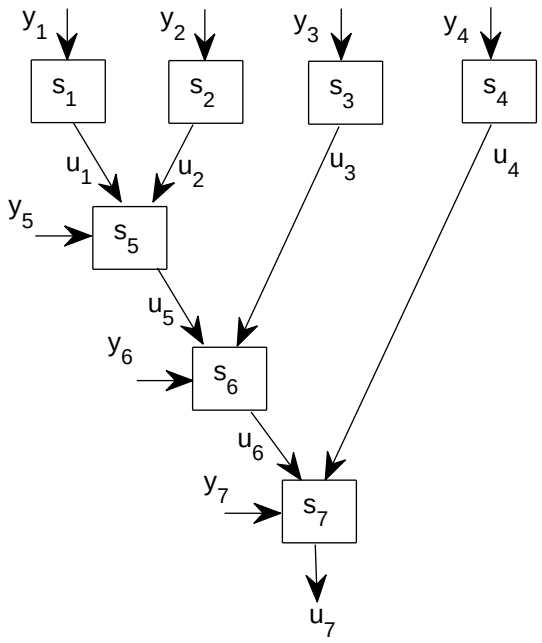



**Figure 2.5**: *Tree architecture*

In this dissertation, the emphasis is on centralized detection and decentralized detection networks employing the parallel architecture with a fusion center. This is because, these networks are the ones generally used in the sensor deployment literature. We next discuss the design of the decision rule in a network employing the parallel architecture with a fusion center.

### 2.2.4 Fusion Rule Design- Parallel Architecture with Fusion Network

From the discussion above, it is evident that the detection performance of a network is dependent on the local and global decision rules employed. In decision rule design, the main problem is to design the fusion rule $\gamma_0$ and local decision rules $(\gamma_1, \ldots, \gamma_N)$ that minimize/maximize a certain performance requirement. Due to high computational complexity we limit our discussion to the design of the global fusion rule (i.e,. finding $\gamma_0$ and threshold

$\eta_0$). Moreover, we consider problems where the observations are conditionally independent since decision design for correlated observations is intractable. Next, we discuss fusion rule design in the Bayesian and Neyman-Pearson frameworks that employ different performance metrics.

### Bayesian Framework

In the Bayesian framework, it is required to design a fusion rule that minimizes the average risk $\mathcal{R}$ as defined in Eqn.(2.1) given some prior probabilities and cost assignments. We follow the same notation used in sections 2.1 and 2.2.3. Also, let $P_f^i$ and $P_d^i$ denote the false alarm and detection probabilities, respectively, of the $i$-th sensor ($i = 1, \ldots, N$). The optimal fusion rule that minimizes the average risk $\mathcal{R}$ is the LRT given as [44],[49]

$$u_0 = \begin{cases} 0 \text{ (i.e., } H_0 \text{ is decided)} & \text{, if } \Lambda(u_1, \ldots, u_N) < \eta_0 \\ 1 \text{ (i.e., } H_1 \text{ is decided)} & \text{, if } \Lambda(u_1, \ldots, u_N) \geq \eta_0 \end{cases} \tag{2.17}$$

where, $\eta = \frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})}$ is the decision threshold. The likelihood ratio $\Lambda(u_1, \ldots, u_N)$ is given as

$$\Lambda(u_1, u_2, \ldots, u_N) = \frac{\Pr(u_1, u_2, \ldots, u_N | H_1)}{\Pr(u_1, u_2, \ldots, u_N | H_0)} \tag{2.18}$$

Assuming decisions are conditionally independent, the likelihood ratio and decision threshold in Eqn.(2.17) can be expressed as [44],[50]

$$\Lambda(u_1, \ldots, u_N) = \sum_{i=1}^{N} \ln\left(\frac{P_d^i(1 - P_f^i)}{P_f^i(1 - P_d^i)}\right) u_i \tag{2.19}$$

$$\eta_0 = \ln\left(\frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})} \prod_{i=1}^{N} \frac{1 - P_f^i}{1 - P_d^i}\right). \tag{2.20}$$

Therefore, the optimal fusion rule is a linear combination of the sensor decisions. The weight of the decision $u_i$ is proportionally related to $P_d^i$ and inversely proportional to $P_f^i$. The fusion threshold $\eta_0$ depends on $\eta$ which contains the prior probability and cost function information.

Since the decisions $\{u_i, i = 1, \ldots, N\}$ are binary (we are considering a binary hypothesis setup), one can also use logical rules to combines these binary decisions. For $N$ binary decisions, there are $2^{2^N}$ unique logical functions that can be used to combine the data. However, not all of these functions are suitable for use [44]. In fact, the set of suitable functions has to satisfy the monotonicity property and are called positive unate functions [50]. Even though the cardinality of the positive unate functions set is less that $2^{2^N}$, it is still computationally intensive to search for a positive unate function that minimizes the risk function $\mathcal{R}$. However, one can attain a satisfactory performance (though no optimality is guaranteed ) by using simple logical functions such as AND, OR and MAJORITY as fusion rules which are discussed below:

1. **And Rule**

   A global detection decision $u_0 = 1$ is made, only if all sensors agree that the phenomenon exists (i.e., $u_i = 1,\ i = 1, \ldots, N$). In this case, the global false alarm $P_f$ and detection probability $P_d$ are given as [44],[49]

   $$P_f = \prod_{i=1}^{N} P_f^i \tag{2.21}$$

   $$P_d = \prod_{i=1}^{N} P_d^i, \tag{2.22}$$

   where, $P_f^i$ and $P_d^i$ are the $i$-th sensor false alarm and detection probabilities, respectively.

2. **OR Rule**

   In the OR rule, a single detection decision by one sensor is sufficient for the fusion center to decide that the phenomenon exists. The global false alarm $P_f$ and miss

probability $P_m$ are given as [44], [49]

$$P_f = 1 - \prod_{i=1}^{N}(1 - P_f^i) \tag{2.23}$$

$$P_m = \prod_{i=1}^{N} P_m^i, \tag{2.24}$$

where, $P_m^i$ is the miss probability of the $i$-th sensor.

3. **Majority Rule**

Assume that sensors have equal false alarm and detection probabilities (i.e., $P_f^i = p_f,\ P_d^i = p_d,\ i = 1, \ldots, N$) and that the decisions are conditionally independent. In this case, the likelihood ratio $\gamma_0(u_1, \ldots, u_N)$ and decision threshold $\eta_0$ in Eqn.(2.17) are given as [44], [49]

$$\gamma_0(u_1, \ldots, u_N) = \sum_{i=1}^{N} u_i \tag{2.25}$$

$$\eta_0 = \frac{\frac{P_0(C_{10}-C_{00})}{P_1(C_{01}-C_{11})}}{N \ln \frac{p_d(1-p_f)}{p_f(1-p_d)}}. \tag{2.26}$$

Since the decisions $\{u_i,\ i = 1, \ldots, N\}$ are binary, the rule in Eqn.(2.25) means that a detection decision $u_0 = 1$ is made only if the number of sensors with a similar decision exceeds a certain number (i.e., counting). Each sensor decision is a Bernoulli random variable, with a success probability of $(1 - p_f)$ under hypothesis $H_0$ and $p_d$ under $H_1$. Therefore, the overall false alarm $P_f$ and detection $P_d$ probabilities are the tail probabilities of two Binomial distributions (one under each hypothesis) and are given as

$$P_f = \sum_{n=\lceil \eta_0 \rceil}^{N} p_f^n (1 - p_f)^{N-n} \tag{2.27}$$

$$P_d = \sum_{n=\lceil \eta_0 \rceil}^{N} p_d^n (1 - p_d)^{N-n} \tag{2.28}$$

where, $\lceil \eta_0 \rceil$ is the smallest integer greater than or equal to $\eta_0$.

In the Bayesian design framework, the decision threshold is a function of the hypotheses prior probabilities and the decision costs. Next, we briefly review the Neyman-Pearson design methodology in which the decision threshold depends on meeting the overall false alarm requirement.

**Neyman-Pearson Framework**

In this framework, no prior probability information or costs are assumed to be known. The goal is design a fusion rule such that the overall false alarm probability $P_f$ is less than a certain level $\alpha$, while maximizing the overall detection probability $P_d$ (or minimizing the miss probability $P_m$) [44].

One can find a solution for this function by using the Lagrangian method and constructing the unconstrained function $F$ given as [44],[50]

$$F = P_m + \eta(P_f - \alpha) \tag{2.29}$$

where, $\eta$ is the Largrange multiplier and is chosen such that $P_f \leq \alpha$.

The non-randomized optimal decision rule that minimizes $F$ is given as

$$u_0 = \begin{cases} 0 \text{ (i.e., } H_0 \text{ is decided)} & \text{, if } \gamma_0(u_1, \ldots, u_N) < \eta_0 \\ 1 \text{ (i.e., } H_1 \text{ is decided)} & \text{, if } \gamma_0(u_1, \ldots, u_N) \geq \eta_0 \end{cases} \tag{2.30}$$

where, the likelihood ratio $\gamma_0(u_1, u_2, \ldots, u_N)$ is given as

$$\gamma_0(u_1, u_2 \ldots, u_N) = \frac{\Pr(u_1, u_2, \ldots, u_N|H_1)}{\Pr(u_1, u_2, \ldots, u_N|H_0)} \tag{2.31}$$

If the sensor decisions $\{u_i,\ i = 1, 2, \ldots, N\}$ are independent, the likelihood ratio $\gamma_0(u_1, u_2 \ldots, u_N)$ and the corresponding decision threshold $\eta_0'$ are given as [48],[44]

$$\gamma_0(u_1, u_2 \ldots, u_N) = \sum_{i=1}^{N} \ln\left(\frac{P_d^i(1 - P_f^i)}{P_f^i(1 - P_d^i)}\right) u_i \tag{2.32}$$

$$\eta_o' = \ln\left(\eta_0 \prod_{i=1}^{N} \frac{1 - P_f^i}{1 - P_d^i}\right). \tag{2.33}$$

We note that the decision rule for the Neyman-Pearson framework is similar in form to the one in the Bayesian framework, with the difference that the decision threshold is chosen

to meet the false alarm requirement. Therefore, one can also use the logical rules (e.g., AND, OR ) discussed earlier as decision rules in the Neyman-Pearson framework.

## 2.3 Summary

In this chapter, we provided a general overview of hypothesis testing and detection in multi-sensor networks. In hypothesis testing, we discussed the design of optimal decision rules in both the Bayesian and Neyman-Pearson frameworks. We saw that the design of the decision rule depends on whether the prior hypotheses probabilities and costs are known or not. Under both frameworks, we saw that the optimal fusion rule is an LRT. We then examined detection in multi-sensor networks and saw that sensors can either send raw measurements to an FC (i.e., centralized detection) or make local detection decision (i.e., decentralized detection). We discussed the design of optimal fusion rules in both centralized and decentralized detection networks. In centralized detection, the optimal fusion rule at the FC is an LRT that is dependent on the statistical distribution of the observations. In decentralized detection, we briefly discussed some of the common network architectures (e.g., detection with fusion). We then discussed the design of fusion rules under the Bayesian and the Neyeman-Pearson frameworks.

# Chapter 3

# Control Theoretic Deployment

In this chapter, we first discuss the general setup of the sensor deployment problem. This consists of modeling the region of interest where sensors are deployed and modeling the sensors themselves in terms of their detection performance. In addition, we present a general mathematical formulation of the sensor deployment problem. Next, we discuss the proposed approach to solve the deployment problem. Specifically, we motivate and visualize concepts from optimal control theory that we use as tools to perform sensor deployment. We introduce the linear quadratic regulator (LQR) problem and briefly review methods that are commonly used to obtain the solution of the LQR problem.

## 3.1 System Setup

In deployment problems, the area to be monitored is usually referred to as the region of interest (ROI) [16],[20]. The ROI is where false alarm/ detection requirements are specified. It is also the area where sensors are deployed. The ROI is usually modeled as a grid $\mathcal{G}$ of points [16] -[21]. Without loss of generality, we assume that $\mathcal{G}$ is rectangular with dimensions $N_x$ and $N_y$ (i.e., $\mathcal{G} = \{(x_{i_x}, y_{i_y}),\ i_x = 1, \ldots, N_x,\ i_y = 1, \ldots, N_y\}$). However, we note that our proposed solution approach is applicable to any distribution of grid points. Each grid point is associated with a certain false alarm and detection probability requirement. Considering all the grid points, we can arrange these false alarm/detection requirements in two $N_x N_y \times 1$ vectors $\mathbf{p}_f^{req}$/ $\mathbf{p}_d^{req}$. Sensors are deployed in the ROI with the goal of satisfying

false alarm/detection requirements. An $N_x N_y \times 1$ vector $\mathbf{D}$ can be used to indicate whether a sensor is deployed at a certain grid point or not. $D(j) = 1$ indicates that a sensor is deployed at the $j$-th grid point ($j = 1, 2, \ldots, N_x N_y$) and a value of $D(j) = 0$ indicates otherwise.

One of the factors that determine the performance of a network is the detection performance of the individual sensors. In the next section, we discuss some of the sensing models used to approximate sensor behavior.

## 3.2 Sensing Models

In the sensor deployment literature, several sensing models have been proposed and used to characterize the detection performance of sensors. One main attribute of any sensing model is its effective coverage area. If a target is present in a sensor's effective coverage area, then it is assumed that the sensor is able to detect that target with an acceptable reliability level. Another attribute that differentiates between sensing models, is the sensor's immunity to variability (e.g., noise). One can also classify sensors according to the nature of the information they report back to the fusion center. For example, a sensor can either report its raw measurement or its quantized local decision.

In what follows, we discuss some of the sensing models most used in literature [13]-[22]

1. **Point model**

   In this model, sensors do not have an effective coverage area [13]-[15]. This means that in order for a sensor to detect a phenomenon, the phenomenon has to pass through the sensor. Examples of point sensors include chemical sensors (e.g., gas sensors). However, point sensing models are not adequate to model different kinds of sensors (e.g., acoustic) that exhibit a coverage area feature.

2. **Disc model**

   In this model, sensors have an effective coverage area (usually a circle/sphere in 2-D/ 3-D). If a target is within the effective coverage area of a sensor, then it is assumed

that the sensor can always (with 100% success probability) detect the presence of the target. On the other hand, the target can not be detected if its outside the sensor's coverage area. Let $d$ denote the distance between the sensor and the target of interest, then the probability of detecting the target $p_{detect}$ is given as

$$p_{detect}(d) = \begin{cases} 1 & \text{if } d \leq R \\ 0 & \text{if } d > R \end{cases} \tag{3.1}$$

where, $R$ is the radius of the sensor effective coverage area. Disc sensing models have been widely used to model the performance of acoustic and seismic sensors [16]. One drawback of this model is the fact that it does not incorporate uncertainty in the decision making process. We also note that a disc sensing model is used when sensors report quantized detection decisions.

3. **Distance-dependent model**

This model is frequently used to approximate a sensor's detection performance in the presence of noise. It is particularly suitable when a sensor measures the signal amplitude emitted by the target of interest. The amplitude of a signal decays as it propagates with a decay rate that depends on the medium of propagation (i.e., distance dependency). Moreover, since ambient noise interferes with the sensor operation, a sensor's detection decision is probabilistic in nature. One frequently used distance-dependent sensing model is the exponential decay model [20], [21]. Let $d$ denote the distance between the sensor and target of interest, then the detection probability $p_{detect}$ is given as [20], [21]

$$p_{detect}(d) = \begin{cases} e^{-\tau d} & \text{if } d \leq R \\ 0 & \text{if } d > R \end{cases} \tag{3.2}$$

where, $\tau$ is a decay parameter that depends on the sensor's design and the environment. $R$ is the radius of the effective coverage area. Similar to the disc sensing model, the model in Eqn.(3.2) is used to model the performance of acoustic and seismic sensors. We also note that this model is used when sensors report detection decisions rather than measurements.

4. **Energy detector model**

   In this model, a sensor measures the energy of the signal emitted by the target of interest. If $d$ is the distance between the sensor and the target of interest, and $S(d)$ is the signal's energy at the sensor then the energy measurement $y$ as measured by the sensor is given as

$$y = S(d) + N^2 \tag{3.3}$$

   where, $N^2$ is the energy of the additive noise. The model in Eqn.(3.3) has been empirically verified and is widely used [22].

In this dissertation, we focus on sensors that have an exponential decay sensing model or which are energy detectors. This is because they better reflect the actual performance of a wide range of senors.

We next turn to mathematically stating the general sensor deployment problem that we examine in this dissertation.

## 3.3 Problem Formulation

The sensor deployment problem that we study can be stated as follows: given false alarm/ detection requirements and $K$ sensors, how can these sensor be deployed (i.e., what is the deployment vector $\mathbf{D}$) such that the difference between achieved and required detection probabilities be minimized while attempting to satisfy false alarm requirements. Let $\mathbf{p}_f^K/\mathbf{p}_d^K$ be two $N_xN_y \times 1$ vectors, that denote the achieved false alarm/ detection over the grid points when $K$ sensors are deployed in the grid. We can then mathematically state the deployment problem as follows

$$\begin{aligned} &\underset{\mathbf{D}}{\operatorname{argmin}} \sum_{j:p_d^K(j)<p_d^{req}(j)} (\mathbf{p}_d^K(j) - \mathbf{p}_d^{req}(j))^2 \\ &\text{subject to} \begin{cases} \mathbf{p}_f^K \leq \mathbf{p}_f^{req} \\ \mathbf{1}^T\mathbf{D} = K. \end{cases} \end{aligned} \tag{3.4}$$

We note, that the summation index in Eqn.(3.4) is taken over the set of grid point where detection requirements are not met. This ensures that grid points where the detection requirements are satisfied do not negatively contribute to our objective.

## 3.4 Proposed Approach

In a sensor deployment problem, usually the goal is to find the best $K$ positions out of $N$ possible positions at which sensors can be deployed in order to minimize a given cost function. Since the number of sensors at any point in the ROI takes only one of two values (0 or 1), the sensor deployment problem is a binary integer programming problem. Integer programming problems are NP-hard, which implies that the computational complexity associated with solving such a problem increases exponentially with the number of variables (i.e., $K$ and $N_xN_y$ in our deployment problem). One possible solution for an integer programming problem is enumeration. All possible solutions are listed and the value of the cost function corresponding to each solution is calculated. The optimal solution is the one that has the minimum cost function value. The enumeration approach requires large storage and computational capabilities which prohibits its use except for problems of a small search space. For example, in our deployment problem if $K = 20$ sensors and $N_x = N_y = 30$ possible locations the set of enumerated solutions will consist of $\binom{900}{20} = 9.8033 \times 10^{55}$ possible sensor positions configurations. Therefore, it is necessary to provide solutions to the deployment problem that do not require high computational complexity to calculate.

In this dissertation, we present a unique and novel approach to sensor deployment. The highlights of the proposed approach are presented below:

1. First, the inherent complexity of determining $K$ sensor positions out of $N_xN_y$ possible grid points is alleviated by switching to a sequential sensor deployment strategy. However, it is important to note that the sequential approach is merely to reduce the search space and does not restrict the end user from temporally staggering the deployment. That is, the entire $K$ steps in the sequential process can be completed

and all $K$ sensors can be deployed similar to the one shot deployment process.

2. Second, we model the sequential deployment process as an optimal control problem with the objective as stated in Eqn.(3.4). This provides a rigorous framework to the deployment problem that has been previously lacking in literature.

3. Thirdly, to simplify the solution of the optimal control based deployment process, we linearize the effects of a single sensor deployment on the overall detection/false alarm performance at each grid point for various fusion architectures. Combined with the objective in Eqn.(3.4), this linearization enables us to model sensor deployment as a linear quadratic regulator (LQR) problem

4. Finally, we exploit the rich literature in LQR problem solutions to design a suite of sensor deployment algorithms that can be applied to various data/decision fusion based detection systems.

In the following section, we present an introduction to optimal control theory and provide details on how one might formulate the deployment problem as an optimal control problem.

## 3.5 Optimal Control Theory

The main objective in optimal control theory is to guarantee that a dynamical system attains a certain desired performance. Optimal control theory is attractive because it offers an analytic approach to solving many design problems. It is also suitable for handling multivariable systems with ease.

Let the dynamical system under consideration be discrete and given as [30]

$$\mathbf{x}_{k+1} = f_k(\mathbf{x}_k, \mathbf{u}_k), \ k = 0, 1, \ldots, K-1, \tag{3.5}$$

$$\mathbf{x}_0 = \mathbf{x}_{initial}, \tag{3.6}$$

where, $\mathbf{x}_k$ is the state vector of the system at the $k$-th instant. The function $f_k(\mathbf{x}_k, \mathbf{u}_k)$ describes the evolution of the system's state (i.e., $\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}$). We note that $f_k(\cdot, \cdot)$ is

dependent on both the current system state (i.e. $\mathbf{x}_k$) and the vector $\mathbf{u}_k$ which is called the control vector. We note that the control vector can be arbitrary in general and is chosen by the system's designer. $\mathbf{x}_0$ represents the initial state of the system. Fig.4.1 shows a block diagram of the system described in Eqn.(3.5). Since the goal is for the system to attain a certain state, a cost function $J$ is usually defined that reflects the end user's preference. The cost function $J$ is given as

$$J = L(\mathbf{x}_K) + \sum_{k=0}^{K-1} V_k(\mathbf{x}_k, \mathbf{u}_k), \tag{3.7}$$

where, $L(\mathbf{x}_K)$ is called the final function and is a function of the final state of the system. The function $V_k(\mathbf{x}_k, \mathbf{u}_k)$ is called the running function and is explicitly dependent on the system state and control vector.

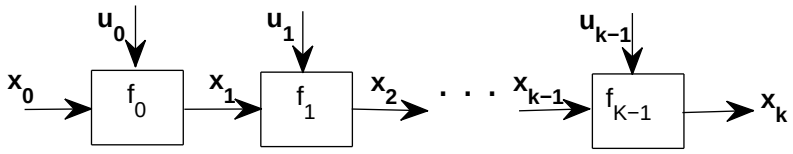



**Figure 3.1**: *A sequential discrete control system*

In optimal control theory, the goal is to find the sequence of optimal control vectors $\{\mathbf{u}_k, k = 0, 1, \ldots, K-1\}$ that results in minimizing $J$. Noting that the system model in

Eqn.(3.5) is effectively a set of equality conditions, the optimal control problem can be viewed as a constrained optimization problem. Thus it is possible to use the methods of Lagrange multipliers and variational calculus to establish the optimality conditions. For convenience, the scalar Hamiltonian function $H_k$ is usually defined as [30]

$$H_k = V_k(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\lambda}_{k+1}^T f_k(\mathbf{x}_k, \mathbf{u}_k), \tag{3.8}$$

where, $\boldsymbol{\lambda}_k$ is the $k$-th Lagrangian multiplier. Using the Hamiltonian , one can derive the following Karush Kuhn Tucker (KKT) optimality conditions [30], [31] and [32]

$$\begin{aligned}
\mathbf{x}_{k+1} &= f_k(\mathbf{x}_k, \mathbf{u}_k) && (3.9)\\
\mathbf{x}_0 &= \mathbf{x}_{initial} && (3.10)\\
\boldsymbol{\lambda}_k &= \nabla_{\mathbf{x}_k}^T f_k(\mathbf{x}_k, \mathbf{u}_k)\boldsymbol{\lambda}_{k+1} + \nabla_{\mathbf{x}_k}^T V_k && (3.11)\\
\boldsymbol{\lambda}_K &= \nabla_{\mathbf{x}_K}^T L && (3.12)\\
0 &= \nabla_{\mathbf{u}_k} V_k + \boldsymbol{\lambda}_{k+1}^T \nabla_{\mathbf{u}_k} f(\mathbf{x}_k, \mathbf{u}_k), && (3.13)
\end{aligned}$$

where, $\nabla_{\mathbf{x}} L$ is the differential of $L$ with respect to $\mathbf{x}$.

Note that the boundary conditions for the set of Eqns.(3.9)-(3.13) are at $k = 0$ since $\mathbf{x}_0 = \mathbf{x}_{initial}$ and at $k = N$ since $\boldsymbol{\lambda}_K = \nabla_{\mathbf{x}_K}^T L$. Therefore, this kind of problems is appropriately called two-point boundary-value problems.

With regards to the deployment problem, the elements of this problem can be mapped into elements of an optimal control problem. We can consider the detection network as a system or a plant. The state of this system ($\mathbf{x}_k$) corresponds to the network's overall detection performance when $k$ sensors are deployed in the grid. Since the detection performance depends on the number of deployed sensors, the effect of deploying an additional sensor $k \to k+1$ is analogous to the effect of a control vector as in Fig.4.1. Thus, the sequential deployment of sensors corresponds to a series of control vectors. The system's evolution function $f_k$ depends on the change in the system state (i.e., detection performance) when a control vector (i.e., deployment of an additional sensor) is applied to the system. This

change depends on the network's design, sensor characteristics and detection requirements. The last element in an optimal control problem is the cost function $J$. In the deployment problem Eqn.(3.4), the goal is to minimize the squared difference between achieved and required detection performance. This can easily be mapped to the cost function $J$, which is the last element of the optimal control problem.

In the next section, we introduce a specific type of optimal control problem - the linear quadratic regulator (LQR). Our proposed algorithms are based on modeling the deployment process as an LQR problem as discussed in the next section.

## 3.6 Linear Quadratic Regulator

The linear quadratic regulator (LQR) problem is a very well-studied problem in optimal control theory. One of the main advantages of the LQR problem is the fact that the resulting KKT optimality conditions in Eqns.(3.9)-(3.13), are both sufficient and necessary [30],[31] and [9]. In addition, there are various approaches to solving the set of optimality conditions for the LQR problem [30] and [53]. In this section, we discuss the LQR problem formulation and motivate modeling the deployment problem as an LQR problem.

In a linear quadratic regulator (LQR) problem, the function $f(\mathbf{x}_k, \mathbf{u}_k)$ in Eqn.(3.9) is linear in both $\mathbf{x}_k$ and $\mathbf{u}_k$ and can be stated as [30],[31]

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k, \;\; k = 0, 1, \ldots, K-1, \tag{3.14}$$

where, $\mathbf{A}_k$ and $\mathbf{B}_k$ are matrices of appropriate dimensions. A block diagram of the LQR state evolution is shown in Fig. 4.2. In addition, the cost function $J$ in Eqn.(3.7) is quadratic and is given as[30],[31]

$$J = \frac{1}{2}\mathbf{x}_K^T\mathbf{Q}_f\mathbf{x}_K + \frac{1}{2}\sum_{k=0}^{K-1}(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}_k\mathbf{u}_k). \tag{3.15}$$

$\mathbf{Q}_f$, $\mathbf{Q}_k$ and $\mathbf{R}_k$ are weighting matrices of appropriate dimensions that the end user chooses according to the performance requirements. In order to yield positive values of $J$ for all

possible combinations of $\mathbf{x}_k$ and $\mathbf{u}_k$, the weighting matrices need to be non-negative. In general, $\mathbf{Q}_f$ and $\{\mathbf{Q}_k,\ k = 0, 1, \ldots, K-1\}$ are chosen to be positive semi-definite, while $\{\mathbf{R}_k, k = 0, 1, \ldots, K-1\}$ are positive definite matrices.

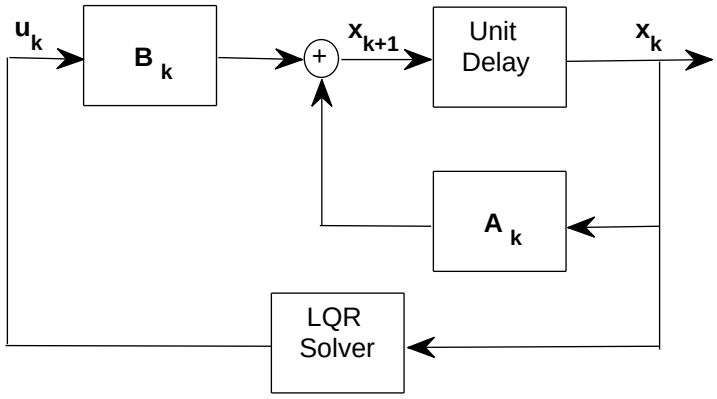



**Figure 3.2**: *An LQR block diagram*

We note that the Hamiltonian function that corresponds to the LQR problem is given as

$$H_k = \frac{1}{2}\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^T\mathbf{R}_k\mathbf{u}_k + \lambda_{k+1}(\mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k). \tag{3.16}$$

Applying the KKT conditions Eqns.(3.9)-(3.13), the LQR optimality conditions are given

as

$$\mathbf{x}_0 = \mathbf{x}_{initial} \tag{3.17}$$

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k,\ k = 0, 1, \ldots, K-1 \tag{3.18}$$

$$\boldsymbol{\lambda}_k = \mathbf{A}_k^T\boldsymbol{\lambda}_{k+1} + \mathbf{Q}_k\mathbf{x}_k \tag{3.19}$$

$$\boldsymbol{\lambda}_K = \mathbf{Q}_f\mathbf{x}_K \tag{3.20}$$

$$\mathbf{u}_k = -\mathbf{R}_k^{-1}\mathbf{B}_k^T\boldsymbol{\lambda}_{k+1}. \tag{3.21}$$

The LQR problem is a very well-studied problem in optimal control theory literature [30]-[33]. It is a well-behaved problem for which a solution (i.e., optimal control vector) can be analytically evaluated using Eqns.(3.17)-(3.21). The linearity of $f_k(\cdot,\cdot)$ and the convexity of $J$ means that the KKT LQR optimality conditions are both necessary and sufficient conditions of optimality. For all of above mentioned advantages, it is desirable to model an optimal control problem as an LQR problem if possible.

The main contribution in this dissertation, is to propose an LQR formulation of the sensor deployment problem. One is motivated to do this by noting that if we set $\mathbf{x}_k = \mathbf{p}_d^k - \mathbf{p}_d^{req}$, then the cost function in Eqn.(3.4) resembles the function $J$ in Eqn.(3.15). Both of the cost functions are convex in the state $\mathbf{x}_k$. Moreover, a sequential deployment of $K$ sensors corresponds to a sequence of $K$ control vectors (i.e., $\mathbf{u}_k, k = 0, 1, \ldots, K-1$). When the $k$-th sensor is deployed (i.e., $\mathbf{u}_k$ is applied), it causes a change in the network detection performance (i.e., $\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}$). This change corresponds to the evolution in the system's state. The remaining feature of an LQR problem is a state evolution function that is linear in both $\mathbf{x}_k$ and $\mathbf{u}_k$ as in Eqn.(3.14). This function is not evident from the deployment problem statement Eqn.(3.4). By constructing this linear function, the deployment problem becomes equivalent to an LQR problem. As we will see later, the nature of this linear state evolution function depends on the sensor sensing, nature of target, detection/false alarm requirements and the detection scheme employed by the network. Formulating this linear function, taking all of the above factors into account, is the main objective of our work in

the next chapters. After having an LQR formulation of the deployment problem, one can then solve for the control vectors (i.e., sensor positions) that minimize the cost function $J$ (i.e., satisfy performance requirements).

To solve for the sensor positions, we need to solve for the optimal control vectors. In the next section, we discuss methods for solving for the optimal control vectors in an LQR problem.

## 3.7 LQR Solution Methods

In this section, we briefly discuss some of the methods one can use to evaluate the optimal control vector in an LQR problem. These solutions differ based on the nature of the LQR problem. LQR problems can be classified as being either static or dynamic. In dynamic/static LQR problems, the system evolution function changes/remains fixed after each application of a control vector. In other words, the matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ are functions of $k = 0, \ldots, K-1$ in a dynamical LQR problem and are constant in a static problem. Therefore, in a dynamical LQR problem the optimal control vector is calculated based on the current system evolution function. Hence, solving a dynamical LQR problem is referred to as a 1-step horizon optimization problem. In contrast, solving a static LQR problem is referred to as a $K$-step optimization problem.

1. **The Sweep Method**

   The sweep method is used to solve LQR static problems (i.e., $K$-step optimization problems). It is based on the observation that the boundary condition on the Lagrange multiplier is given as

   $$\boldsymbol{\lambda}_K = \mathbf{Q}_f \mathbf{x}_K. \tag{3.22}$$

   Based on this, in the sweep method it is assumed that the $k$-th Lagrange multiplier is of similar form

   $$\boldsymbol{\lambda}_k = \mathbf{P}_k \mathbf{x}_k, \ k = 0, 1, \ldots, K-1 \tag{3.23}$$

where, $\mathbf{P}_k$ is a square matrix with appropriate dimensions and will be calculated later. Substituting for $\lambda_k$ in Eqn.(3.21), the optimal control vector is given as

$$\mathbf{u}_k = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}_{k+1}\mathbf{x}_{k+1},\ k = 0, 1, \ldots, K-1 \tag{3.24}$$

since $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$ we can solve for $\mathbf{u}_k$ as follows

$$\mathbf{u}_k = -\mathbf{G}_k\mathbf{x}_k,\ k = 0, 1, \ldots, K-1 \tag{3.25}$$

where, $\mathbf{G}_k$ is a square matrix of appropriate dimensions and is given as

$$\mathbf{G}_k = (\mathbf{R} + \mathbf{B}^T\mathbf{P}_{k+1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}_{k+1}\mathbf{A}. \tag{3.26}$$

Using the above equations, it is possible to describe $\mathbf{P}_k$ in terms of a backward difference equation as follows

$$\mathbf{P}_k = \mathbf{A}^T(\mathbf{P}_{k+1} - \mathbf{P}_{k+1}\mathbf{B}\mathbf{S}^{-1}\mathbf{B}^T\mathbf{P}_{k+1})\mathbf{A} + \mathbf{Q}_k \tag{3.27}$$

where, $\mathbf{S}_k$ is a square matrix of appropriate size and is given as

$$\mathbf{S}_k = \mathbf{R} + \mathbf{B}^T\mathbf{P}_{k+1}\mathbf{B} \tag{3.28}$$

The boundary condition of Eqn.(3.27) at $k = K$ can be obtained from Eqns.(3.22) and (3.23) as

$$\mathbf{P}_K = \mathbf{Q}_f. \tag{3.29}$$

We note that Eqn.(3.27) is called a discrete time algebraic Riccati equation.

The steps of the sweep method are summarized in Algorithm 1 [30].

2. **Non-iterative Riccati Equation Solution Method**

The sweep method is in essence an iterative solution of the Riccati equation in Eqn.(3.27). There are other approaches to solving the Riccati equation that are non-iterative in nature. These approaches fall under two main classes: Hamiltonian and Lyapunov

**Algorithm 1** Sweep method

**Initialization** $\mathbf{P}_K = \mathbf{Q}_f$, $\mathbf{G}_K = \mathbf{0}$ and $\mathbf{S}_K = \mathbf{0}$.
**for** $k = K-1, K-2, \ldots, 0$ **do**
  Calculate $\mathbf{S}_k$ from Eqn.(3.28).
  Calculate $\mathbf{G}_k$ from Eqn.(3.26).
  Store $\mathbf{G}_k$.
  Calculate $\mathbf{P}_k$ from Eqn.(3.27).
**end for**
**for** $k = 0, 1, \ldots, K-1$ **do**
  Calculate $\mathbf{u}_k$ from Eqn.(3.25).
**end for**

methods [53]. Hamiltonian methods do not make use of sparsity or any feature of the Riccati equation. On the other hand, the Lyapunov approach can make some use of matrix structures. Therefore, the focus here is on the latter method.

In a Lyapunov equation, it is required to solve for an unknown matrix $\mathbf{P}$. The Lyapunov equation is given as

$$\mathbf{FP} + \mathbf{PF}^T = -\mathbf{HH}^T \tag{3.30}$$

where, $\mathbf{F}$ and $\mathbf{H}$ are matrices of appropriate dimensions.

It is straightforward to show that the Riccati equation Eqn.(3.27) can be written as a Lyapunov equation

$$\mathbf{A} - \mathbf{BR}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{P}^T\mathbf{P} + \mathbf{PA} - \mathbf{BR}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{P} = -\mathbf{Q} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} \tag{3.31}$$

the right hand side is then

$$\mathbf{TT}^T = \mathbf{Q} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} \tag{3.32}$$

also let $\mathbf{A} - \mathbf{BR}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{P} = \mathbf{F}^T$. Therefore, a Riccati equation is a special case of the Lyapunov equation. Thus one can use Lyapunov equation solution methods to solve for the Riccati equation Eqn.(3.27). Some of the methods one can use include the Bartlet-Stewart and the alternating direction implicit (ADI) methods [53].

3. **Differentiation**

In a dynamic LQR problem, the $\mathbf{A}_k$ and $\mathbf{B}_k$ matrices are dynamic (i.e., change with $k$) and are not known before hand. Therefore, one can not solve the optimality conditions associated with the $K$-steps cost function $J$ Eqn.(3.15). One method to overcome this difficulty, is to sequentially solve for the optimal control vector minimizing the single step cost function $J_k$ defined as follows:

$$J_k = \frac{1}{2}(\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R}_k \mathbf{u}_k). \tag{3.33}$$

subject to

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k. \tag{3.34}$$

Then, setting the gradient of $J_k$ with respect to $\mathbf{u}_k$ equal to zero and solving for $\mathbf{u}_k$, it is straightforward to find:

$$\mathbf{u}_k = -(\mathbf{R}_k + \mathbf{B}_k^T \mathbf{Q}_k \mathbf{B}_k)^{-1} \mathbf{B}_k^T \mathbf{Q}_k \mathbf{A}_k \mathbf{x}_k \tag{3.35}$$

In this dissertation, depending on the nature of the LQR problem, we employ both the Sweep method as well as the Differentiation method to determine optimal control vector. The LQR formulations for various fusion architectures are presented in the following chapters.

## 3.8 Summary

In this chapter, we provided an overview of our sensor deployment problem and our solution approach. We described the sensing models considered in this dissertation. In addition, we provided a mathematical formulation of the deployment problem and motivated the use of the LQR problem as a solution approach. Finally, we provided an overview of some of the methods that can be used to solve an LQR problem and thus the deployment problem. In the following chapters, we use the proposed LQR approach to develop algorithms for various fusion architectures.

# Chapter 4

# Sensor Deployment in Non-Collaborative Detection Networks

In this chapter, we study the sensor deployment problem in a non-collaborative detection system employing the logical OR rule at the fusion center. One of the advantages of considering this network is the relative simplicity of modeling its detection performance in comparison to other network fusion rules. Moreover, modeling the effect of the deployment of an additional sensor is straightforward in a non-collaborative detection network. Using this network model, we are able to formulate sensor deployment as an LQR problem. Based on this formulation, we propose an LQR-based deployment algorithm. To reduce computational complexity, we propose a second heuristic deployment algorithm. The performance of these algorithms is compared against that of the state of the art Diff_Deploy deployment algorithm.

In section 4.1, we introduce the system model. This mainly includes; the sensor sensing model and quantifying the overall detection (or miss) performance of the network. In addition, a linear model of the non-collaborative detection network is discussed. The main focus in section 4.2 is to introduce the LQR formulation of the sensor deployment problem in the non-collaborative network. Based on the LQR formulation, we introduce our LQR-based deployment algorithm in section 4.3. Furthermore, the heuristic deployment algorithm is

described in section 4.4. In section 4.5, simulation results comparing the performance of the proposed algorithms versus that of the Diff_Deploy algorithm are presented.

## 4.1 System Model

The system under consideration consists of an area of interest where region-wise detection requirements are provided by the end user. We model the area of interest as a grid $\mathcal{G}$ of $N_x \times N_y$ points. The detection/miss requirements at every point on the grid are ordered in two $N_x N_y \times 1$ vector $\mathbf{p}_d^{req}/\mathbf{p}_m^{req}$. Additionally, the sensing model and the number of sensors available, serve as inputs to our sensor deployment algorithm. Given these inputs, the objective of this work is to determine the optimal sensor placement that would minimize the square difference between achieved and required detection/miss probabilities. It is important to note that we assume a simple detection model in which a target is declared to be detected if at least a single sensor in the network is able to detect it (i.e., logical OR rule). We assume that the sensors have an exponential decay sensing model as in Eqn.(3.2). In this model, even if a target is within the detection radius (i.e. coverage radius), there is a probability that it will not be detected (i.e., it will be missed). A wide range of practical sensors [54] (e.g., infrared, ultrasound) fit this general model. However, it is important to note that the choice of the sensing model does not affect the basic formulation of the algorithms proposed.

Following the linear shift invariant (LSI) model as in [21], the process of linking individual sensors' detection characteristic to the overall probability of detection requirements on the grid is mathematically quantified using miss probabilities $p_m$ ($p_m = 1 - p_d$, where $p_d$ is the probability of detection). The probability of a target being detected by any sensor on the grid is the complement of the target being missed by all the sensors on the grid. The overall miss probability $M(x, y)$ corresponds to the probability that a target at point $(x, y)$ will be missed by all sensors, i.e.,

$$M(x, y) = \prod_{(i,j)\in\mathcal{G}} p_m((x, y), (i, j))^{D(i,j)}, \tag{4.1}$$

where, $p_m((x,y),(i,j))$ is the probability that a sensor at point $(i,j)$ misses a target located at point $(x,y)$ of the grid. Here, $D(i,j)$ represents the presence or absence of a sensor at the location $(i,j)$ on the grid, and corresponds to

$$D(i,j) = \begin{cases} 1 & \text{, if there is a sensor at } (i,j) \\ 0 & \text{, if there is no sensor at } (i,j) \end{cases} \tag{4.2}$$

Taking the natural logarithm of both sides in Eqn.(4.1) results in

$$m(x,y) = \sum_{(i,j)\in\mathcal{G}} D(i,j) \ln p_m((x,y),(i,j)), \tag{4.3}$$

where $m(x,y)$ is called the overall logarithmic miss probability at point $(x,y)$ [21]. Let us define the function $b(x,y)$ as follows

$$b(x,y) = \begin{cases} \ln p_m((x,y),(0,0)) & \text{, } d((x,y),(0,0)) \leq R \\ 0 & \text{, } d((x,y),(0,0)) > R, \end{cases} \tag{4.4}$$

where, $R$ is the coverage radius and $d((x,y),(0,0))$ is the distance between point $(x,y)$ on the grid and the grid's origin point $(0,0)$.

The overall logarithmic miss probabilities for all points on the grid can be arranged in a vector $\mathbf{m} = [m(x,y), \forall (x,y) \in \mathcal{G}]^T$ of dimension $N_x N_y \times 1$ that corresponds to

$$\mathbf{m} = \mathbf{B}\mathbf{D}. \tag{4.5}$$

Here, $\mathbf{D} = [D(i,j), \forall (i,j) \in \mathcal{G}]^T$ is the deployment vector of dimension $N_x N_y \times 1$. The $((i-1)N_y + j)$-th element of $\mathbf{D}$ indicates the number of sensors deployed at point $(i,j)$ on the grid. The matrix $\mathbf{B}$ is of dimension $N_x N_y \times N_x N_y$, and contains $\{b(x-i,y-j), \forall (x,y) \in \mathcal{G}, (i,j) \in \mathcal{G}\}$. $b(x-i,y-j)$ corresponds to the $(r,c)$-th entry of $\mathbf{B}$, where $r = (x-1)N_y + y$ and $c = (i-1)N_y + j$. Essentially, $b(x-i,y-j)$ quantifies the effect of placing a sensor at point $(i,j)$ on the logarithmic miss probability at point $(x,y)$ on the grid. The logarithmic miss probabilities can be easily converted to detection probabilities at a later stage.

The question we attempt to address in this work is the following: Given a number of sensors, how can the sensors be deployed (i.e., where can the sensors be placed) to minimize the squared error between achieved and required detection/miss probabilities? Once again,

the squared error (SE) between required and achieved detection probabilities at all points in the grid can be mapped to the SE between required and achieved miss probabilities. However, one should note that after sensors have been deployed, the achieved detection probability at some of the grid points will meet/ exceed the detection requirements at these points. Therefore, we emphasize on minimizing the squared error at the points for which detection/miss requirements are not met.

Let $\mathbf{m}_{req}$ be the required miss probability vector, and let $\mathbf{m}_k$ be the achieved miss probability vector resulting from deploying $k$ sensors to the grid. Our sensor deployment problem can be mathematically formulated as follows:

$$\begin{aligned} \underset{\mathbf{u}}{\operatorname{argmin}} & \sum_{j:p_d^K(j)<p_d^{req}(j)} (\mathbf{m}_K(j) - \mathbf{m}^{req}(j))^2 \\ & \text{subject to} \Big\{ \mathbf{1}^T\mathbf{D} = K \end{aligned} \tag{4.6}$$

where, $p_d^K(j)$ is the detection probability at the $j$-th grid point after $K$ sensors have been deployed to the grid. $\mathbf{1}^T$ indicates the transpose of an $N_xN_y \times 1$ vector, with all entries set to 1 and $K$ is the total number of available sensors.

## 4.2 Optimal Control Formulation

The problem of minimizing the square error between achieved and required detection probabilities can be viewed as minimizing the square difference between achieved and required miss or overall logarithmic probabilities. Define $\mathbf{x}_k$ to be the difference between the required log miss probability vector ($\mathbf{m}_{req}$) and the log miss probability vector achieved after deploying $k$ sensors ($\mathbf{m}_k$). i.e., $\mathbf{x}_k = \mathbf{m}_k - \mathbf{m}_{req}$. The system described in Eqn.(4.5) can be written in terms of the dynamic model

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{4.7}$$

where, $\mathbf{u}_k$ is the deployment vector at the $k$-th step. In typical control problems, the index $k$ indicates the time index describing time evolution of the system. In our case, we assume

sequential placing of sensors (i.e., sensors are placed one at a time) with $k$ representing the $k$-th step in this process. In terms of the dynamic state vector $\mathbf{x}_k$, we can define a weighted SE cost function $J$ as:

$$J = \frac{1}{2}\mathbf{x}_K^T \mathbf{Q}_f \mathbf{x}_K + \frac{1}{2}\sum_{k=0}^{K-1}(\mathbf{x}_k^T \mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^T \mathbf{R}\mathbf{u}_k). \tag{4.8}$$

Here, $\mathbf{u}_k$ is the deployment vector for the $k$-th sensor. $\mathbf{Q}$, $\mathbf{Q_f}$ and $\mathbf{R}$ are positive-definite diagonal weighting matrices with dimension $N_x N_y \times N_x N_y$ that are chosen by the designer. In our problem, a good choice of $\mathbf{Q}$,$\mathbf{Q_f}$ and $\mathbf{R}$ is one that reflects the detection requirements on the grid. That is, if the detection requirement at a certain point is relatively large compared to other points, then the entries in the matrices $\mathbf{Q}$,$\mathbf{Q_f}$ and $\mathbf{R}$ that correspond to that point should be in such a way that the resulting solution will be biased towards satisfying that point before other points. One choice that fits well with the above reasoning is the following; $\mathbf{R}(i) = (\frac{\mathbf{m}_{req}(i)}{\mathbf{1}^T\mathbf{m}_{req}})^{-1}$ where $\mathbf{R}(i)$ is the $i$-th diagonal element of $\mathbf{R}$, $\mathbf{m}_{req}(i)$ is the overall logarithmic miss requirement at point $i$ on the grid. The $i$-th diagonal elements of $\mathbf{Q}$ and $\mathbf{Q}_f$ are given as $\mathbf{Q}(i) = \mathbf{Q}_f(i) = (\mathbf{R}(i))^{-1}$, where $(\cdot)^{-1}$ denotes the inverse operation. The goal of the control problem is to determine the sequence of control vectors $\{\mathbf{u}_k, k = 0, 1, \ldots, K-1\}$ that would minimize the cost function $J$. The squared error cost function penalizes both positive and negative deviations from the required detection probability profile. To avoid incurring a penalty for satisfying/exceeding detection requirements, the error terms corresponding to a satisfied point is set to zero in $J$. Therefore, after each sensor deployment, the cost function to be minimized is the squared error evaluated at the points where detection/miss requirements are not satisfied. We refer to this squared error cost function as the effective SE. Therefore, the effective SE corresponds to

$$\text{eSE}(k) = \sum_{j:p_d^k(j)<p_d^{req}(j)} x_k(j)^2 \tag{4.9}$$

where, $p_d^k(j)$ is the achieved detection probability at the $j$-th grid point after $k$ sensors have been deployed in the grid. The formulation of the problem discussed above is known

as a linear-quadratic regulator problem in control theory literature [32][33]. Therefore, we can employ the techniques used for solving LQR problems to solve the sensor deployment problem as described in the previous chapter.

## 4.3 Deployment Algorithm

In this section, we introduce the deployment algorithm which is based on the LQR formulation of the sensor deployment problem. Algorithm2, illustrates the steps of the optimal control based sensor deployment algorithm.

Given the total number of sensors, $K$, and the $\mathbf{B}$ matrix, the algorithm evaluates the feedback gain matrix (i.e., $\mathbf{G}_k, k = K-1:-1:0$) using the sweep method discussed earlier. Sensors are deployed sequentially until all available sensors have been deployed or when detection requirements at all points on the grid have been satisfied (i.e., effective SE equals 0). In the $k$-th iteration, the set of points for which the detection/miss requirements are satisfied is determined and the entries in the vector $\mathbf{x}_{k-1}$ corresponding to these points are set to 0. Afterwards, the $k$-th deployment vector is calculated as in Eqn.(3.25). However, since the deployment vector can only have $\{0, 1\}$ entries, the entry in the deployment vector $\mathbf{u}$ corresponding to the largest entry ( with index $j_{max}$) in the $k$-th deployment vector (i.e., $\mathbf{u}_k$) is set to 1.

We will show that this choice of u serves as the best choice for minimizing the objective function given in Eqn. (4.8). When we evaluate $\mathbf{u}_k$ using the sweep method, we obtain a lower bound on the objective function (since, we relax the integer constraint on u). We denote the corresponding Hamiltonian as $H_k^c$. Once, we discretize $\mathbf{u}_k$, we obtain a solution that yields a higher objective function value. We denote the Hamiltonian corresponding to a discretized u as $H_k^d$. It is desirable to determine a discretization rule that will minimize $\Delta_k = H_k^d - H_k^c$. In Theorem 4.3.1 , we show that forcing the maximum value of $\mathbf{u}_k$ to 1 is the best strategy for discretization from this standpoint.

**Theorem 4.3.1.** *The difference $\Delta_k = H_k^d - H_c^d$ is minimized when the largest entry in $\mathbf{u}_k$*

*is set to* 1 *with all the remaining entries set to* 0.

*Proof.* Substituting for $\lambda_{k+1}$ from Eqn.(3.21) in Eqn.(3.16) , we can express $H_k^c$ as

$$H_k^c = \frac{1}{2}\mathbf{x}_k^T\mathbf{Q}\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^T\mathbf{R}\mathbf{u}_k - \mathbf{u}_k^T\mathbf{R}^T\mathbf{B}^{-1}\mathbf{x}_k - \mathbf{u}_k^T\mathbf{R}^T\mathbf{u}_k. \tag{4.10}$$

We also note that in the sweep method $\mathbf{x}_k = -\mathbf{G}_k^{-1}\mathbf{u}_k$, therefore

$$H_k^c = \mathbf{u}_k^{c^T}\mathbf{F}_H\mathbf{u}_k^c, \tag{4.11}$$

where, $\mathbf{F}_H$ is given as

$$\mathbf{F}_H = \frac{1}{2}\mathbf{G}_k^{-1T}\mathbf{Q}\mathbf{G}_k^{-1} + \frac{1}{2}\mathbf{R} + \mathbf{R}^T\mathbf{R}^{-1}\mathbf{G}_k^{-1} - \mathbf{R}^T. \tag{4.12}$$

We note that the value of the Hamiltonian is minimum when using the continuous control vector $\mathbf{u}_k^c$. Discretizing the control vector to $\mathbf{u}_k^d$ introduces an error $\Delta$. Denote the Hamiltonian when using $\mathbf{u}_k^d$ as $H_k^d = \mathbf{u}_k^{dT}\mathbf{F}_H\mathbf{u}_k^d$ (i.e., $\Delta = H_k^d - H_c^d$)

$$\begin{aligned} \Delta &= \mathbf{u}_k^{dT}\mathbf{F}_H\mathbf{u}_k^d - \mathbf{u}_k^{c^T}\mathbf{F}_H\mathbf{u}_k^c & (4.13)\\ &= (\mathbf{u}_k^d - \mathbf{u}_k^c)^T\mathbf{F}_H(\mathbf{u}_k^d + \mathbf{u}_k^c) & (4.14) \end{aligned}$$

Let $\mathbf{y} = \mathbf{F}_H\mathbf{u}_k^d + \mathbf{u}_k^c$, then $\Delta$ is the inner product of $\mathbf{y}$ and $(\mathbf{u}_k^d - \mathbf{u}_k^c)$. To make $\Delta$ as small as possible, one can use the triangle inequality

$$\Delta \leq \|(\mathbf{u}_k^d - \mathbf{u}_k^c)\|\|\mathbf{y}\|, \tag{4.15}$$

where, $\|\mathbf{y}\|$ is the norm of vector $\mathbf{y}$. Therefore, in order to minimize $\Delta$ one replaces the maximum entry of $\mathbf{u}_k^c$ with 1 and the remaining entries are set to 0. □

After updating the deployment vector, the resulting overall logarithmic miss can be calculated as in Eqn.(4.5). It is also possible to calculate the achieved detection probability vector as $\mathbf{p}_d^k = \mathbf{1} - \exp(\mathbf{m}_k)$, where $\exp(\mathbf{m}_k)$ indicates the exponential of each entry of $\mathbf{m}_k$.

**Algorithm 2** LQR Based Deployment Algorithm

**Input**: $\mathbf{p}_d^{req}$ (detection requirement), $K$ (number of available sensors) and $\mathbf{B}$
**Outputs**: $\mathbf{D}$ (deployment vector).
**for** $k = K - 1 : -1 : 0$ **do**
    Evaluate $\mathbf{G}_k$ using the sweep method.
**end for**
**Initialization**: $k = 0$, $\mathbf{u} = \mathbf{0}$
**while** $k \leq K$ or $\mathbf{p}_d^{req} \not\leq \mathbf{p}_d^k$ (i.e., $eSE \neq 0$) **do**
    Find set of grid points with unsatisfied detection requirements $\{i : p_d^k(i) \geq p_d^{req}(i)\}$
    Set $x_{k-1}(i) = 0$ (i.e., Error at unsatisfied points only is considered)
    Calculate the control vector $\mathbf{u}_k = -\mathbf{G}_k\mathbf{x}_k$
    Find index $j_{max}$, where $j_{max} = \max_{index}(\mathbf{u}_k)$ ( The function $\max_{index}(\mathbf{u}_k)$ returns the index of the largest entry in vector $\mathbf{u}_k$)
    Update the deployment vector (i.e., $\mathbf{D}(j_{max}) = 1$)
    Calculate $\mathbf{m}_k = \mathbf{BD}$
    Evaluate achieved detection profile $\mathbf{p}_d^k = 1 - \exp(\mathbf{m}_k)$
    Calculate $\mathbf{x}_k$
    Increment number of sensors in the grid $k = k + 1$
**end while**

## 4.4 Max_ Deficiency / Greedy Deployment Algorithm

Due to the computational cost associated with the optimal control solution presented earlier, it is advantageous to develop a low complexity algorithm that is relatively simple to implement. In this section, we introduce a new algorithm that we call the Max_Deficiency algorithm.

We assume that given $K$ sensors, we will be deploying them sequentially, until all sensors have been deployed or the detection requirements have been met at all the grid points. In the $k$-th iteration, the Max_Deficiency algorithm calculates the difference $\mathbf{p}_\delta$ between the required $\mathbf{p}_d^{req}$ and achieved detection probabilities $\mathbf{p}_d^{k-1}$ and then deploys the $k$-th sensor to the point $j_{max}$ on the grid where $\mathbf{p}_\delta$ is maximum. The deployment vector $\mathbf{u}$ is updated by placing a 1 instead of 0 at its $j_{max}$ entry. Employing Eqn.(4.5), we calculate the resulting logarithmic miss probability $\mathbf{m}_k$. The resulting detection probability vector $\mathbf{p}_d^k$ is calculated as $\mathbf{p}_d^k = \mathbf{1} - \exp(\mathbf{m}_k)$. In other words, at each step in the deployment algorithm, we identify the point on the grid that is most deficient in terms of meeting the detection requirements

and we place a sensor in that position, calculate its effect and repeat the process. Identifying the point with the maximum deficiency is similar to identifying the location on the grid that will have the maximum impact on the cost function $J$.

The Max_Deficiency algorithm is illustrated in Algorithm 3.

**Algorithm 3** Max_Deficiency Algorithm

**Inputs**: $\mathbf{p}_d^{req}$,$K$
**Outputs**: $\mathbf{D}$ (i.e., deployment vector)
**Initialization**: $\mathbf{p}_d^0 = \mathbf{0}$, $\mathbf{p}_\delta = \mathbf{p}_d^{req}$ and $k = 0$
**while** $k \leq K$ or $\mathbf{p}_\delta \succ \mathbf{0}$ **do**
    Calculate $\mathbf{p}_\delta = \mathbf{p}_d^{req} - \mathbf{p}_d^k$
    Find index $j_{max}$, where $j_{max} = \max_{index}(\mathbf{p}_\delta)$ (The function $\max_{index}(\mathbf{p}_\delta)$ returns the index of the largest entry in vector $\mathbf{p}_\delta$)
    Place a sensor at position $j_{max}$ ($\mathbf{D}(j_{max}) = 1$)
    Calculate $\mathbf{m}_k = \mathbf{BD}$
    Evaluate achieved detection profile $\mathbf{p}_{d,k} = 1 - \exp(\mathbf{m}_k)$
    Increment number of sensors in the grid $k = k + 1$
**end while**

## 4.5 Simulation Results

In this section, the performance of both the optimal control based and the Max_Deficiency algorithm is compared to that of Diff_Deploy algorithm [21].

In the first experiment, we compare the number of sensors needed by the three algorithms to meet the detection requirements as we vary the decay parameter $\tau$. The area of interest is modeled as a grid of $25 \times 25$ points. The area consists of three subregions each with its own detection requirement as is shown in Fig. 4.1. We assume that all sensors employ a detection radius of $R = 5$. Table 4.1 presents the number of sensors needed by the three algorithms as $\tau$ varies. The stopping criteria in both the Diff_Deploy and Max_Deficiency algorithm is meeting the detection/miss requirements at all grid points. The optimal control based algorithm employs the same criteria but in terms of the effective SE (i.e., deployment terminates when the effective SE equals 0). As expected, the optimal control based algorithm outperforms the Diff_Deploy and the Max_Deficiency in terms of the number of sensors

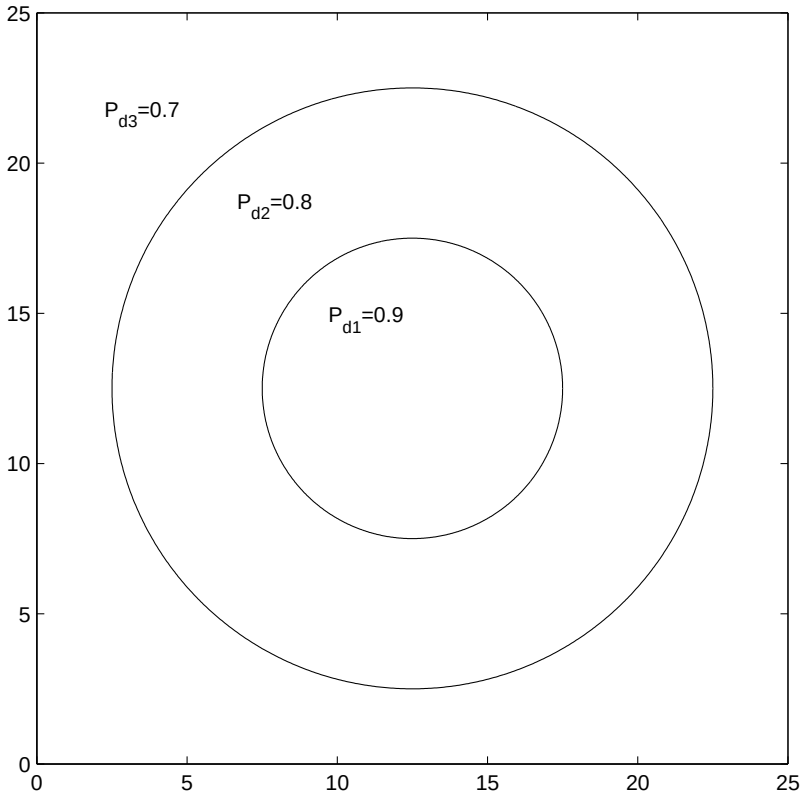



**Figure 4.1**: *Nonuniform detection requirements* $N_x = N_y = 25$

needed to satisfy the detection requirements. This is because, the cost function used in the design of the optimal control algorithm is the effective SE, while there is no clear cost function in the heuristic Diff_Deploy. On the other hand, the effective cost function in the Max_Deficiency algorithm is the error at a single point only, with no regard to the errors at other grid points. In the optimal control deployment algorithm, using the matrix $\mathbf{B}$ along with the sweep method implies that information regarding the effect of each sensor placement on the entire grid is incorporated in the deployment process. This is in contrast to the Max_Deficiency algorithm which makes its deployment decision based solely on the effect of a sensor at its deployment location. Additionally, as $\tau$ increases, the detection sensitivity of a single sensor decreases. Therefore, the number of sensors needed to satisfy the detection requirements increases as $\tau$ increases. This is confirmed in our results presented in Table 4.1.

In order to further compare the performance of the three algorithms, we present the

**Table 4.1**: *Number of required sensor vs.* $\tau$

| $\tau$ | Diff_Deploy | Max_Deficiency | Optimal Control Based |
|---|---|---|---|
| $\tau = 0.05$ | 22 | 20 | 20 |
| $\tau = 0.1$ | 31 | 28 | 27 |
| $\tau = 0.15$ | 32 | 31 | 28 |
| $\tau = 0.2$ | 38 | 38 | 34 |
| $\tau = 0.25$ | 41 | 40 | 39 |

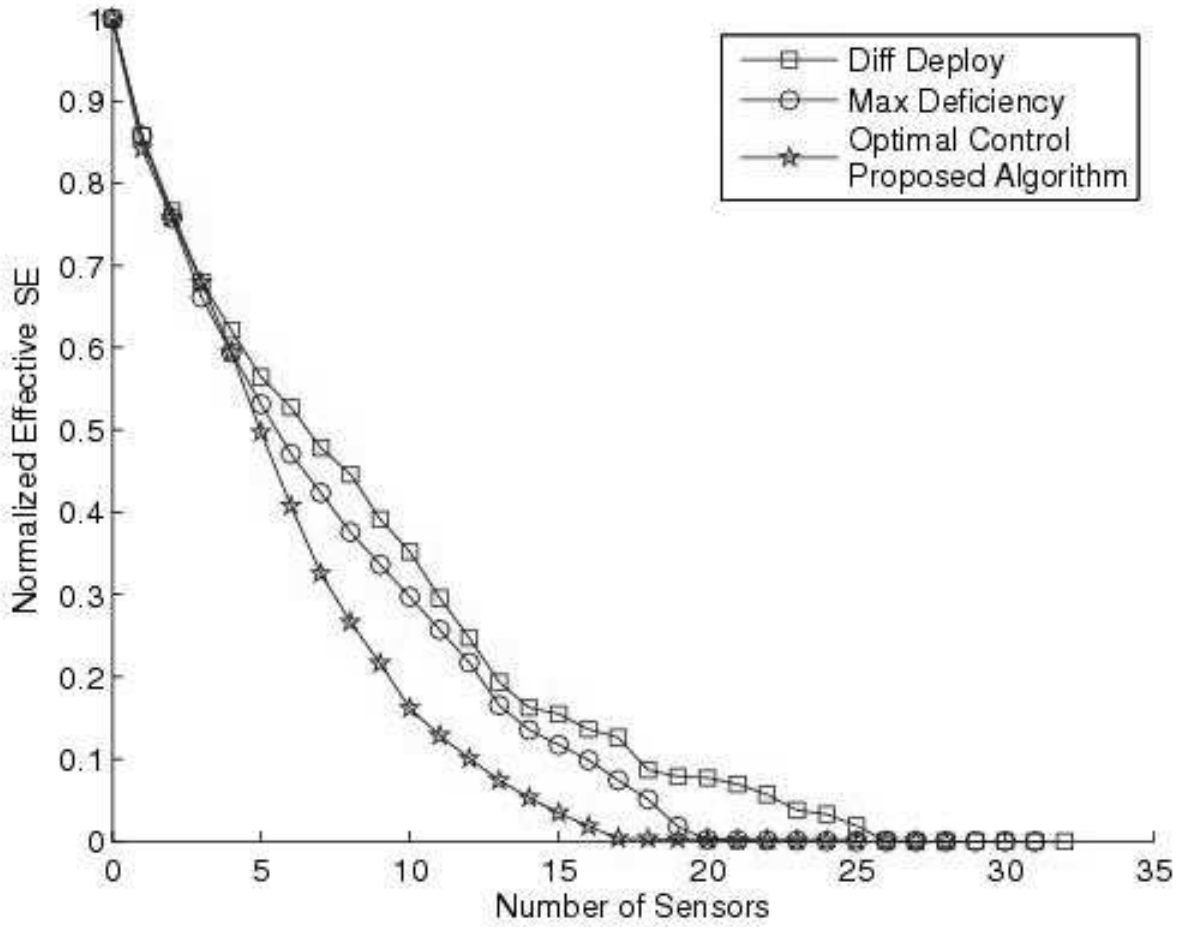



**Figure 4.2**: *SE convergence:* $\tau = 0.15$

evolution of effective SE between achieved and required detection probability profiles as sensors get deployed in the grid (see Fig. 4.2). We specifically examine the SE for the points that are yet to be satisfied in terms of the detection/miss requirements, which we call the effective SE. Fig. 4.2, considers the case of $\tau = 0.15$ listed in Table 4.1. It is

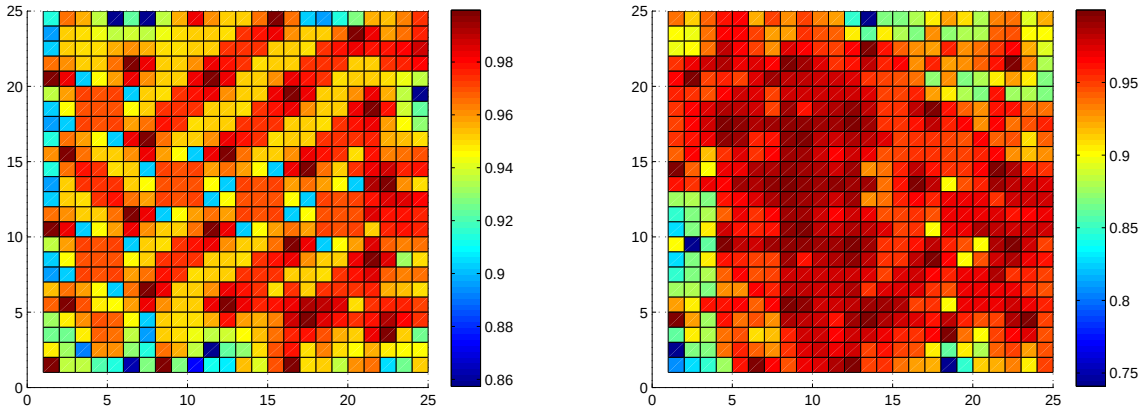
**Table 4.2**: *Number of required sensors for different detection requirements*

| Detection Probabilities | Diff_Deploy | Max_Deficiency | %Savings |
|---|---|---|---|
| p1=0.9,p2=0.8,p3=0.6 | 99 | 72 | 27% |
| p1=0.9,p2=0.7,p3=0.5 | 83 | 66 | 20% |
| p1=0.8,p2=0.9,p3=0.8 | 133 | 121 | 9% |
| p1=0.9,p2=0.5,p3=0.7 | 107 | 92 | 14% |

obvious that the effective SE of the optimal control based algorithm converges faster than that of the Diff_Deploy and the Max_Deficiency. The results indicate that we can satisfy the detection/miss requirements with fewer number of sensors if we employ the proposed optimal control based algorithm. Fig. 4.3 shows the achieved detection probability profile resulting from deploying sensors based on the three algorithms. From Fig. 4.3(c), it is evident that the proposed approach does not overbudget for satisfying the detection requirements in each subregion. For example, in the outer region of the grid where the detection requirement is set to 0.7, we note that achieved detection probability in that subregion is around 0.7. This is in contrast to the Diff_Deploy, where the minimum achieved detection probability is close to 0.86.

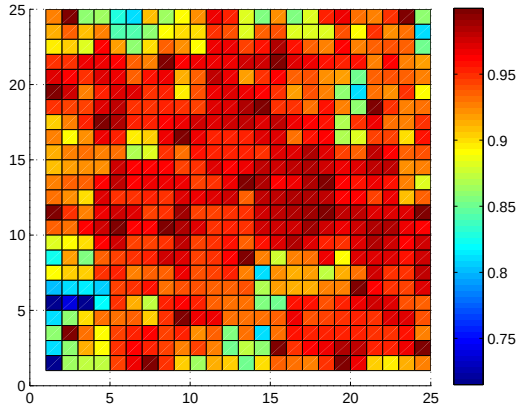
In the second experiment, the performance of the Diff_Deploy is compared to that of the Max_Deficiency algorithm. The grid size is $50 \times 50$. The detection requirements profile is shown in Fig. 4.4. The numerical values of $p1$, $p2$ and $p3$ along with the number of sensors needed to satisfy the detection requirements are listed in Table 4.2. As evident from Table 4.2, the Max_Deficiency algorithm always uses a smaller number of sensors than the Diff_Deploy algorithm. The reduction in the number of sensors depends on the required detection profile, and in our simulations, it ranges from about 10% to 30%.

In the third experiment, the performance of the Max_Deficiency algorithm versus that of the Diff_Deploy is examined in the presence of obstacles. The obstacle positions as well as the required detection probabilities are shown in Fig 4.5. When an obstacle is present between a sensor and a point on the grid that lies within the detection radius of the sensor, the sensor would not be capable of detecting a target at that point (i.e., $p_{miss} = 1$). The




(a) Diff Deploy

(b) Maximum Deficiency




(c) Optimal control based

**Figure 4.3**: *Achieved detection probability profile*

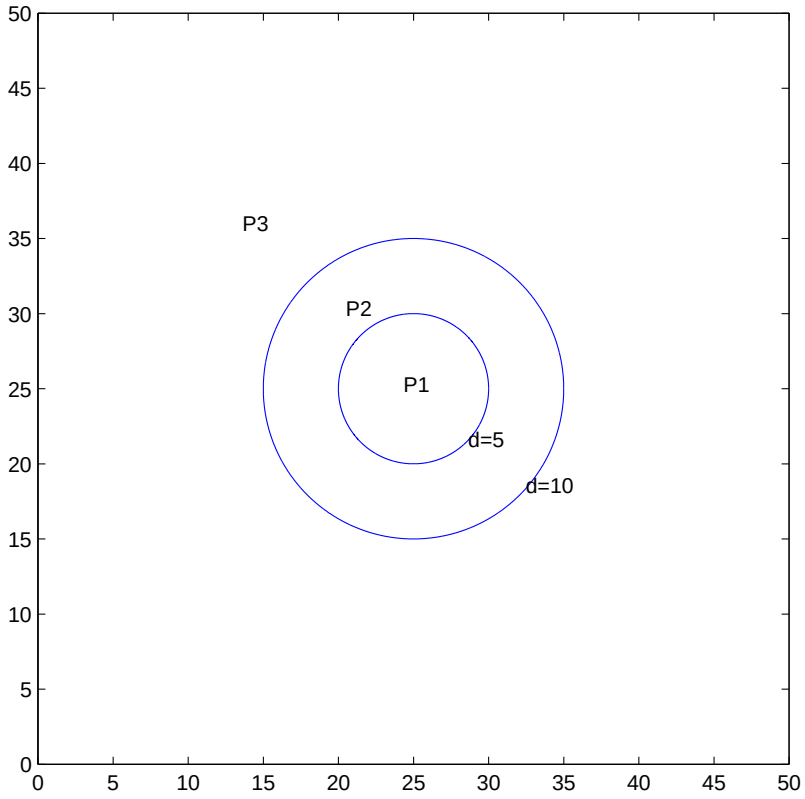



**Figure 4.4**: *Nonuniform detection requirements* $N_x = N_y = 50$

effect of an obstacle being present between a sensor placed at point $(i, j)$ and a point on the grid $(x, y)$ is captured by modifying the entries of the $\mathbf{B}$. The modification is performed by setting to zero the value of $b(x-i, y-j)$, since $b(x-i, y-j)$ corresponds to the logarithmic miss probability (i.e., $\ln(1) = 0$). Simulation results show that Diff_Deploy requires 97 sensors to satisfy the requirements whereas the Max_Defficiency requires 78 sensors only. This corresponds to a savings of approximately 20% in the number of sensors used by the Max_Deficiency in comparison to Diff_Deploy.

## 4.6 Summary

In this chapter, we studied the sensor deployment problem in a non-collaborative detection system. Specifically, given a finite number of sensors, we attempt to determine the locations that the sensors need to be deployed at in order to satisfy the detection requirements in

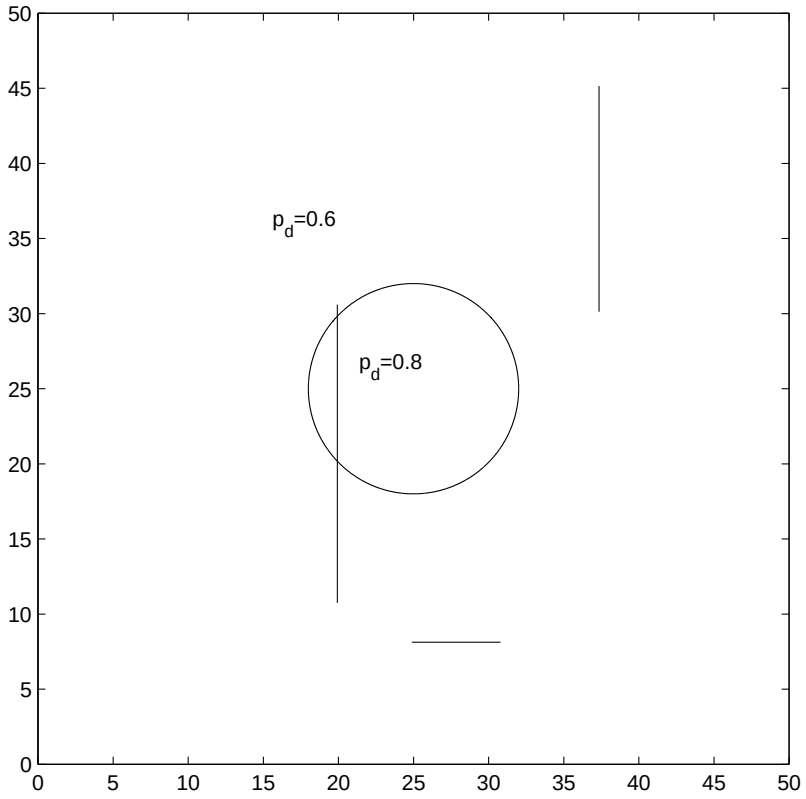



**Figure 4.5**: *Detection probability requirements with obstacles*

a squared error sense. We expressed the deployment problem as a dynamical system and formulated the sensor deployment problem as an optimal control problem (linear quadratic regulator). However, the optimal control based approach is computationally demanding due to the use of the sweep method. Therefore, we introduced a low complexity alternative called the Max_Deficiency algorithm that offers comparable performance relative to the optimal control based approach. Using simulation results, we have shown that the proposed algorithms outperform existing methods by using 10% to 30% fewer number of sensors to satisfy the detection requirements.

In the next chapter, we study the deployment problem in a network employing a centralized fusion rule in which false alarm requirements are incorporated in addition to detection requirements.

# Chapter 5

# Sensor Deployment in Value Fusion Detection Networks – Energy Measurements

In this chapter, we study sensor deployment in a centralized detection system. In this network, noisy energy measurements provided by the sensors are fused to make a global detection decision. In contrast to the deployment problem in the previous chapter, here we consider both false alarm and detection requirements. After approximating the effect of deploying a single sensor on the overall detection performance of the network, we model sensor deployment as an LQR problem. Based on this formulation, we propose two sensor deployment algorithms. We also examine the problem of dynamically selecting the optimal maximum collaboration radius which determines which sensors collaborate to perform detection at a certain grid point. We then study and compare the performance of the proposed algorithms in comparison to the D&C sensor deployment algorithm [22].

In section 5.1, we discuss the detection model when using the value fusion rule. In section 5.2, we propose a linear approximation of the overall detection performance in terms of the sensor positions. This enables us to model deployment as an LQR problem. In section 5.3, we discuss the dynamical update of the collaboration radius as sensors are sequentially deployed. In sections 5.4 and 5.5, we introduce two novel sensor deployment algorithms. Simulation results are presented in section 5.6, where the performance of the

proposed algorithms is compared against that of the D&C deployment algorithm. Finally, a summary of our contribution in this chapter is provided in section 5.7.

## 5.1 System Model

The area of interest is modeled again as grid of $N_x \times N_y$ points. Although the target can be any where in the area of interest, we focus our attention at detecting the target at grid points. By increasing the number of grid points, the resolution of target detection can be improved. At each point on the grid, certain false alarm and detection requirements are specified. Therefore, it is possible to arrange the false alarm and detection requirements over the whole grid in two $N_x N_y \times 1$ vectors, denoted as $\mathbf{p}_f^{req}$ and $\mathbf{p}_d^{req}$, respectively. In our model, the sensors are passive devices. They collect energy measurements emitted in the surrounding environment. For natural phenomena (e.g. seismic activity, electromagnetic radiation, etc,), the energy level of the signal emitted by the source and observed at some distance away from it, is inversely proportional to the distance. Specifically, the energy level at a distance ($d$) away from an energy source can be modeled as [55], [56]

$$S(d) = \begin{cases} S_0 & \text{if } d \leq d_0 \\ \frac{S_0}{(d/d_0)^\kappa} & \text{if } d > d_0. \end{cases} \tag{5.1}$$

where; $S_0$ is the energy level at the source and $\kappa$ is a propagation constant that depends on the environment. Typical $\kappa$ values are between 2 and 5. We note however, that our proposed deployment framework is general and does not depend on any particular choice of the signal energy function.

Furthermore, we assume that the sensor measurements are corrupted by additive Gaussian noise. The source's energy level decay combined with the presence of noise implies that at some distance $R_c$ (referred to as the collaboration radius) away from the source, a sensor's measurement becomes unreliable for detection purposes. Therefore, in making a detection decision at a point on the grid, we disregard a sensor's measurement if it is more than $R_c$ units away from the point of interest. The optimal choice of the collaboration radius $R_c$ is discussed later.

Therefore, for the $k$-th sensor, if a source (i.e., target or phenomena) is present at a point that is $d_k$ units away, the energy measurement $U_k$ at the $k$-th sensor is given as

$$U_k = S(d_k) + N_k^2 \tag{5.2}$$

where, $N_k^2$ is the noise energy. We assume that the measurement noise at the $k$-th sensor is a zero-mean Gaussian noise with a variance of $\sigma_k^2$ (i.e. $N_k \sim \mathcal{N}(0, \sigma_k^2)$). The sensing model in Eqn.(5.2) has been previously presented in [22], [55], [56] and empirically verified in [57]. Note that, we assume that sensors have different measurement noise variances. Obviously, when there is no target or active phenomena, the energy recorded by the $k$-th sensor is

$$U_k = N_k^2. \tag{5.3}$$

In Appendix A, we extend our study of the deployment problem to a network employing value fusion when the sensors measurement is the signal's amplitude rather then its energy. In data fusion (i.e., centralized detection) based detection systems, sensors send their measurements to a fusion center (FC). The FC fuses (combines) the measurements according to a predetermined fusion rule in order to decide on one of two hypotheses, namely: presence of a target/phenomena (hypothesis $H_1$) or absence of a target/phenomena (hypothesis $H_0$). A simple and analytically tractable fusion rule is value fusion. In value fusion, the FC computes the weighted average (denoted by $T$) of the measurements provided by the sensors and then compares it to a detection threshold $\eta$. The non-randomized decision rule is given as

$$\delta(T) = \begin{cases} H_1 & \text{if } T \geq \eta \\ H_0 & \text{if } T < \eta, \end{cases} \tag{5.4}$$

Therefore, the false alarm and detection probabilities for this value fusion based detection system correspond to ($P_f$ and $P_d$, respectively)

$$P_f = Pr(T \geq \eta \mid H_0 \text{ is true}) \tag{5.5}$$

$$P_d = Pr(T \geq \eta \mid H_1 \text{ is true}). \tag{5.6}$$

In our model, we assume that either each sensor has a knowledge about its own variance or that the FC has knowledge about all sensors variances. Therefore, for the $j$-th point on the grid, the FC computes the average $T_j$, given as

$$T_j = \frac{1}{k(j)} \sum_{i=1}^{k(j)} \frac{U_k}{\sigma_k^2} \tag{5.7}$$

where $k(j)$ is the number of sensors that are less than $R_c$ units away from the $j$-th point. The FC, then compares the average $T_j$ to the detection threshold $\eta(j, k(j))$. Therefore, at the $j$-th point the system's overall false alarm rate $p_f(j, k(j))$ is given as

$$\begin{aligned} p_f(j,k(j)) &= Pr[\frac{1}{k(j)} \sum_{i=1}^{k(j)} \frac{U_k}{\sigma_k^2} \geq \eta(j,k(j))] & (5.8)\\ &= 1 - Pr[\sum_{i=1}^{k(j)} \frac{N_k^2}{\sigma_k^2} \leq k(j)\eta(j,k(j))] & (5.9)\\ &= 1 - G\big(k(j)\eta(j,k(j)), k(j)\big). & (5.10) \end{aligned}$$

Here, $G(x, n)$ denotes the cumulative distribution function (CDF) of the Chi-Square distribution $\chi^2$ with $n$ degrees of freedom at point $x$. The system's overall detection probability at the $j$-th point ($p_d(j, k(j))$) corresponds to,

$$\begin{aligned} p_d(j,k(j)) &= Pr[\frac{1}{k(j)} \sum_{k=1}^{k(j)} \frac{U_k}{\sigma_k^2} \geq \eta(j,k(j))] & (5.11)\\ &= Pr[\frac{1}{k(j)} \sum_{k=1}^{k(j)} \frac{S(d_k) + N_k^2}{\sigma_k^2} \geq \eta(j,k(j))] & (5.12)\\ &= Pr[\sum_{k=1}^{k(j)} (\frac{N_k}{\sigma_k})^2 \geq k(j)\eta(j,k(j)) - \sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}] & (5.13)\\ &= 1 - Pr[\sum_{k=1}^{k(j)} (\frac{N_k}{\sigma_k})^2 \leq k(j)\eta(j,k(j)) - \sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}] & (5.14)\\ &= 1 - G\big(k(j)\eta(j,k(j)) - \sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}, k(j)\big) & (5.15) \end{aligned}$$

The FC calculates the decision statistics associated with each point on the grid. The decision threshold is calculated using knowledge of the number of sensors that are within the detection radius, the noise variance and the false alarm requirement associated with every point. In practice, the FC receives measurements from all sensors in the grid and performs a series of sensor measurement averages corresponding to each point on the grid.

The deployment problem that we examine in this work can now be stated as follows: Given $\mathbf{p}_f^{req}$ and $\mathbf{p}_d^{req}$ and a fixed number of sensors $K$, how can we deploy these sensors in a value fusion based detection system, such that the effective SE between achieved and required detection probabilities is minimized while satisfying false alarm requirements? If we denote the achieved false alarm and detection probability vectors after $K$ sensor have been deployed as $\mathbf{p}_f^K$ and $\mathbf{p}_d^K$, then we can mathematically state our problem as

$$\begin{aligned} \underset{\mathbf{D}}{\operatorname{argmin}} & \sum_{j:p_d^K(j)<p_d^{req}(j)} (\mathbf{p}_d^K(j) - \mathbf{p}_d^{req}(j))^2 \\ & \text{subject to} \begin{cases} \mathbf{p}_f^K = \mathbf{p}_f^{req} \\ \mathbf{1}^T\mathbf{D} = K \end{cases} \end{aligned} \tag{5.16}$$

where, $\mathbf{D}$ is an $N_xN_y \times 1$ deployment vector. Its entries indicate the number of sensors at each point on the grid, and take values of either 0 or 1. $\mathbf{1}^T$ indicates the transpose of an $N_xN_y \times 1$, with all entries set to 1.

## 5.2 Optimal Control Formulation

The deployment problem stated earlier can be studied in the context of optimal control theory. The effective SE between achieved and required detection probabilities can be thought of as the cost function which is to be minimized in an optimal control problem. Furthermore, since the detection performance of the DSN detection system is governed by the sensor positions, the deployment vector corresponds to the control vector in an optimal control problem. One of the most studied and well behaved problems in optimal control

literature is the linear quadratic regulator (LQR) problem. In the discrete version of the LQR problem, a system's state evolves linearly with respect to the control vector, while the cost function corresponds to the norm of the system's state. In what follows, we will show that it is possible to approximate the evolution of a logarithmic functional of the achieved detection probability as a linear function of the sensors' positions (i.e., control vector). We then sequentially solve for the optimal control vector by setting the gradient of the cost function to zero.

Due to the nonlinear nature of the CDF $G(x, n)$, it is difficult to quantitatively assess the effect of placing a sensor in the grid on the overall detection and false alarm probabilities. Another difficulty is the change in the degrees of freedom $n$ as an additional sensor is introduced to the grid. As $n$ varies, the relative contribution of any previously deployed sensors to the detection probability varies in a nonlinear fashion.

To overcome these difficulties, we approximate the expression of detection probability in Eqn.(5.15). The approximation provides a better understanding of the effect of placing an additional sensor to the grid. Subsequently, we use the approximation in modeling the deployment problem as an optimal control problem (the LQR model), the solution of which can be analytically determined.

Note that it is possible to approximate the CDF of the Chi-Square distribution using the standard Gaussian CDF denoted as $\Phi$. The approximation is given as [58]

$$G(x, n) \approx \Phi(\frac{x - n}{\sqrt{2n}}). \tag{5.17}$$

Therefore, it is possible to approximate the false alarm rate at the $j$-th point covered by $k(j)$ sensors (i.e., $p_f(j, k(j))$) as

$$p_f(j, k(j)) \approx 1 - \Phi(\frac{k(j)\eta(j, k(j)) - k(j)}{\sqrt{2k(j)}}) \tag{5.18}$$

$$= Q(\frac{k(j)\eta(j, k(j)) - k(j)}{\sqrt{2k(j)}}) \tag{5.19}$$

where $k(j)$ and $\eta(j, k(j))$ are as defined earlier, and $Q(x)$ is the Complementary CDF defined as $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-y^2}{2}} dy$. Similarly, we approximate the detection probability $p_d(j, k(j))$

in terms of the $Q$ function as follows

$$p_d(j,k(j)) \approx 1-\Phi(\frac{k(j)\eta(j,k(j))-\sum_{i=1}^{k(j)}\frac{S(d_k)}{\sigma_k^2}-k(j)}{\sqrt{2k(j)}}) \quad (5.20)$$

$$= Q(\frac{k(j)(\eta(j,k(j))-1)-\sum_{k=1}^{k(j)}\frac{S(d_k)}{\sigma_k^2}}{\sqrt{2k(j)}}) \quad (5.21)$$

Furthermore, the $Q$ function can be approximated as [59]

$$Q(x) \approx 1-\frac{1}{1+e^{-\sqrt{2}x}}. \quad (5.22)$$

which enables us to approximate $p_d(j,k(j))$ as

$$p_d(j,k(j)) \approx 1-\frac{1}{1+e^{-\sqrt{2}(\frac{(\eta(j,k(j))-1)n_j-\sum_{k=1}^{k(j)}\frac{S(d_k)}{\sigma_k^2}}{\sqrt{2k(j)}})}} \quad (5.23)$$

we note that we can use the quantity $\ln(\frac{1}{1-p_d(j,k(j))}-1)$, which has a one-to-one relationship with $p_d(j,k(j))$, and rearranging the terms in Eqn.(5.23), we get that

$$\ln(\frac{1}{1-p_d(j,k(j))}-1)=\frac{1}{\sqrt{k(j)}}\sum_{k=1}^{k(j)}\frac{S(d_k)}{\sigma_k^2}-(\eta(j,k(j))-1)\sqrt{k(j)} \quad (5.24)$$

In order to model the effect that the addition of a sensor has on the detection probability $p_d(j,n_j)$, we initially assume that $n_j=k$. We also assume that the detection threshold $\eta_{j,n_j}$ has been set so as the false alarm requirement $p_f^{req}(j)$ has been met. Therefore, we can express $\ln(\frac{1}{1-p_d(j,k)}-1)$ as

$$\ln(\frac{1}{1-p_d(j,k(j))}-1)=\frac{1}{\sqrt{k(j)}}\sum_{k=1}^{k(j)}\frac{S(d_{k(j)})}{\sigma_k^2}-(\eta(j,k(j))-1)\sqrt{k(j)} \quad (5.25)$$

Suppose that an additional sensor is placed at a distance $d_{k(j)+1}$ from the $j$-th grid point, i.e., $n_j=k+1$. Furthermore, suppose that the detection threshold has been modified accordingly, then we can express $\ln(\frac{1}{1-p_d(j,k(j)+1)}-1)$ after the sensor addition as

$$\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) = \frac{1}{\sqrt{k(j)+1}}\sum_{k=1}^{k(j)}\frac{S(d_k)}{\sigma_k^2}+\frac{1}{\sqrt{k(j)+1}}\frac{S(d_{k(j)+1})}{\sigma_{k(j)}^2}$$
$$- (\eta(j,k(j)+1)-1)\sqrt{k(j)+1} \quad (5.26)$$

using Eqn.(5.25), we can express $\sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}$ in terms of $\ln(\frac{1}{1-p_d(j,k(j))} - 1)$ as

$$\sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_{k(j)}^2} = \sqrt{k(j)} \ln(\frac{1}{1-p_d(j,k(j))} - 1) + k(j)(\eta(j,k(j)) - 1) \tag{5.27}$$

substituting Eqn.(5.27) in Eqn.(5.26), we can express $\ln(\frac{1}{1-p_d(j,k(j)+1)} - 1)$ in terms of $\ln(\frac{1}{1-p_d(j,k(j))} - 1)$ as

$$\begin{aligned} \ln(\frac{1}{1-p_d(j,k(j)+1)} - 1) &= (1-\alpha_{k(j)+1}) \ln(\frac{1}{1-p_d(j,k(j))} - 1) + \frac{1}{\sqrt{k(j)+1}} \frac{S(d_{k(j)+1})}{\sigma_{k(j)+1}^2} \\ &+ \frac{k(j)}{\sqrt{k(j)+1}}(\eta(j,k(j)) - 1) - \sqrt{k(j)+1}(\eta(j,k(j)) - 1) \end{aligned} \tag{5.28}$$

where $1 - \alpha_{k(j)+1} = \sqrt{\frac{k(j)}{k(j)+1}}$.

Since we are interested in the difference between achieved and required detection probabilities(or equivalently their functional), we define the quantity $x(j, k+1)$ as

$$x(j,k(j)+1) = \ln(\frac{1}{1-p_d(j,k(j)+1)} - 1) - \ln(\frac{1}{1-p_d^{req}(j)} - 1) \tag{5.29}$$

where, $p_d^{req}(j)$ is the required detection probability at the $j$-th point on the grid. The quantity $x(j, k+1)$ is in effect a measure of the difference between achieved and required detection probabilities. This is true since the mapping of $y$ to $\ln(\frac{1}{1-y} - 1)$ is a one to one mapping. Subtracting $\ln(\frac{1}{1-p_d^{req}(j)} - 1)$ from both sides of Eqn.(5.28), we can express the evolution of the difference between achieved and required detection probabilities with the addition of a sensor as

$$\begin{aligned} x(j,k(j)+1) &= x(j,k(j)) - \alpha_{k(j)+1} \ln(\frac{1}{1-p_d(j,k(j))} - 1) + \frac{1}{\sqrt{k(j)+1}} \frac{S(d_{k(j)+1})}{\sigma_{k(j)+1}^2} \\ &+ \frac{k(j)}{\sqrt{k(j)+1}}(\eta(j,k(j)) - 1) - \sqrt{k(j)+1}(\eta(j,k(j)+1) - 1) \end{aligned} \tag{5.30}$$

Furthermore, noting that $\ln(\frac{1}{1-p_d(j,k(j))} - 1) = x(j,k(j)) + \ln(\frac{1}{1-p_d^{req}(j)} - 1)$, Eqn.(5.30) can be expressed as

$$\begin{aligned} x(j,k(j)+1) &= (1-\alpha_{k(j)+1}) x(j,k(j)) - \alpha_{k(j)+1} \ln(\frac{1}{1-p_d^{req}(j)} - 1) \\ &+ \frac{1}{\sqrt{k(j)+1}} \frac{S(d_{k(j)+1})}{\sigma_{k(j)+1}^2} + \frac{k(j)}{\sqrt{k(j)+1}}(\eta(j,k(j)) - 1) - \sqrt{k(j)+1}(\eta(j,k(j)+1) - 1) \end{aligned} \tag{5.31}$$

The last four terms in Eqn.(5.31), represent the approximate effect the addition of a sensor that is $d_{k(j)+1}$ units away from the $j$-th point on the grid will have on the difference between achieved and required detection probabilities. Let us denote, the effect of a sensor placed at the $i$-th point on the grid on the difference between achieved and required detection probabilities at the $j$-th point on the grid as

$$\begin{aligned} B(j,i) &= \frac{1}{\sqrt{k(j)+1}}\frac{S(d(j,i))}{\sigma^2_{k(j)+1}} + \frac{k(j)}{\sqrt{k(j)+1}}(\eta(j,k(j))-1) \\ &- \sqrt{k(j)+1}(\eta(j,k(j)+1)-1) - \alpha_{k(j)+1}\ln(\frac{1}{1-p_d^{req}(j)}-1) \end{aligned} \tag{5.32}$$

where, $d(i,j)$ is the distance between the two points. Furthermore, using the deployment vector $\mathbf{u}_k$, which was introduced earlier, it is possible to express the evolution of the system's state (i.e., $x(j,k+1)$ in Eqn.(5.31)) at all points on the grid in vector form as

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k,\ m = 0,1,\ldots,k-1 \tag{5.33}$$

where $\mathbf{x}_k$ is an $N_xN_y \times 1$ vector defined as $\{x(j,k(j)), j = 1,2,\ldots,N_xN_y\}$, where $k(j)$ is the number of sensors whose measurements are used in the fusion process (i.e., sensors that are within the collaboration radius associated with the $j$-th point). The subscript $k$ denotes the number of sensors that are deployed in the grid, and $K$ is the total number of available sensors. The matrix $\mathbf{A}_k$ is a square diagonal matrix of dimension $N_xN_y$, with its $j$-th diagonal entry given as $\mathbf{A}_k(j,j) = 1-\alpha_{k(j)+1}$. The matrix $\mathbf{B}_k$ is a square matrix of dimension $N_xN_y$, whose entry in the $j$-th row and $i$-th column is equal to $B(j,i)$ defined in Eqn.(5.32). Note that the deployment vector in the linear system described in Eqn.(5.33) can be viewed as a control vector.

As stated earlier, our goal in this paper is to deploy a fixed number of sensors such that false alarm requirements are met while the effective SE between the achieved and required detection probabilities is minimized. Meeting false alarm requirements can be easily achieved by choosing a suitable detection threshold for every point after each sensor deployment. As for the effective SE between achieved and required detection probabilities, it

can be equivalently described as the weighted quadratic norm of the state ($\mathbf{x}_m$) of the system described in Eqn.(5.33). Assuming that the weighted quadratic norm of the system's state is chosen as the cost function and the deployment vector corresponds to a control vector, we are motivated to solve the deployment problem as an optimal control problem. Here, the objective is to determine the control vector that would minimize the cost function. That is, the deployment problem in Eqn.(5.16) can be restated as

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}}\, J &= \frac{1}{2}\mathbf{x}_K^T\mathbf{Q}_f\mathbf{x}_K + \frac{1}{2}\sum_{k=0}^{K-1}(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}_k\mathbf{u}_k) \\ \text{subject to} &\begin{cases} \mathbf{p}_f^K = \mathbf{p}_f^{req} \\ \mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k,\ k = 0, 1, \ldots, K-1 \\ \mathbf{1}^T\mathbf{D} = K, \end{cases} \end{aligned} \tag{5.34}$$

where, $\mathbf{Q}_f$,$\mathbf{Q}_k$ and $\mathbf{R}_k$ for $\{k = 0, 1, \ldots, K-1\}$ are symmetric positive definite weighting matrices. The squared error cost function penalizes both positive and negative deviations from the required detection probability profile. To avoid incurring a penalty for satisfying/exceeding detection requirements, the error terms corresponding to a point where the detection requirement has been met/exceeded is set to zero in $J$. The optimal control problem corresponding to our system is the linear quadratic regulator (LQR) problem.

## 5.3 Choice of Collaboration Radius

The collaboration radius $R_c$ for any given point on the grid determines which sensor measurements are combined to make a detection decision. A naive choice of the collaboration radius will result in inefficient use of sensors. For example, having a small collaboration radius implies that for every point only nearby sensor measurements are used in detection resulting in the deployment of a large number of sensors in the grid. On the other hand, choosing a large collaboration radius implies that the measurements of sensors that are far away from the point of interest are used in the decision process. However, due to decaying signal energy, the measurements of sensors that are far away are dominated by noise energy, which reduces their value in the decision process. When the number of sensors used in

the decision process at the $j$-th point increases, the detection threshold $\eta(j, k(j))$ decreases, while $k(j)\eta(j, k(j))$ increases in magnitude. Moreover, Eqn.(5.15) (which is repeated here)

$$p_d(j, k(j)) = 1 - G\big(k(j)\eta(j, k(j)) - \sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}, k(j)\big) \tag{5.35}$$

indicates that the achieved detection probability increases as the argument $k(j)\eta(j, k(j)) - \sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}$ decreases in magnitude. Therefore, in order to either maintain or improve the detection probability, any additional sensors need to be close enough to the $j$-th point, so that the increase in $k(j)\eta(j, k(j))$ is offset by a corresponding increase in $\sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}$. Hence the collaboration radius should be wisely chosen, so that the number of sensors used is minimal while the detection requirements are satisfied.

The selection of optimal collaboration radius was studied in [22]. The collaboration radius was calculated with the assumption of sensors being deployed simultaneously (i.e., in parallel), and that false alarm and detection requirements were met after deployment. This enabled the calculation of the collaboration radius in terms of the required false alarm and detection probabilities. However, the resulting collaboration radius in [22] essentially transforms the deployment problem into a coverage problem.

In what follows, we discuss the calculation of the collaboration radius taking into account the sequential nature of our sensor deployment scheme. In addition to false alarm and detection probability requirement, we incorporate achieved detection probability in the calculation of the collaboration radius. Since our deployment scheme is sequential, the collaboration radius for any given point, is sequentially updated (i.e., dynamically) as sensors are added to the grid. Based on the achieved and required detection probability, false alarm requirement and the number of sensors employed in the decision process, we calculate an upper bound on the collaboration radius. The deployment of a sensor within this upper bound will always improve the achieved detection probability, while placing the sensor outside this upper bound will degrade the achieved detection probability.

Suppose that, for the $j$-th point on the grid, $k(j)$ sensors are used in the decision process. Furthermore, suppose that the achieved detection probability $p_d(j, k(j))$ is less than the

required detection probability $p_d^{req}(j)$, therefore the difference $\varepsilon(j,k(j))$ between the required and achieved detection probabilities can be written as

$$\begin{aligned}\varepsilon(j,k(j)) &= p_d^{req}(j) - p_d(j,k(j)) &(5.36)\\ &= p_d^{req}(j) - 1 - G\big(k\eta(j,k(j)) - \sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}, k(j)\big). &(5.37)\end{aligned}$$

Rearranging the terms in Eqn.(5.37), we can express the weighted sum of the signal energies $\sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2}$ as

$$\sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2} = k(j)\eta(j,k(j)) - G^{-1}\big(1+\varepsilon(j,k(j)) - p_d^{req}(j), k(j)\big) \tag{5.38}$$

where, $G^{-1}(x,n)$ denotes the inverse of the CDF of the Chi-Square distribution of degree $n$ and at point $x$. Now if an additional sensor is added within the collaboration radius, then the change $\delta(j,k(j)+1)$ in the difference between achieved and required detection probabilities using $k$ and $k(j)+1$ sensors can be written as

$$\delta(j,k(j)+1) = \varepsilon(j,k(j)) - \varepsilon(j,k(j)+1). \tag{5.39}$$

A positive value of $\delta(j,k(j)+1)$ implies that the difference between achieved and required detection probabilities has decreased by using the measurement of the additional sensor, whereas a negative value indicates that the use of the additional sensor measurement has actually degraded the detection probability.

Using Eqn.(5.38), one can express the weighted signal energy $\frac{S(d_{k(j)+1})}{\sigma^2_{k(j)+1}}$ of the additional sensor in terms of the change in the detection probability (i.e., $\delta(j,k(j)+1)$) as

$$\begin{aligned}\frac{S(d_{k(j)+1})}{\sigma^2_{k(j)+1}} &= \sum_{k=1}^{k(j)+1} \frac{S(d_k)}{\sigma_k^2} - \sum_{k=1}^{k(j)} \frac{S(d_k)}{\sigma_k^2} &(5.40)\\ &= G^{-1}\big(1+\varepsilon(j,k(j)) - p_d^{req}(j), k(j)\big) + (k(j)+1)\eta(j,k(j)+1) - k(j)\eta(j,k(j))\\ &- G^{-1}\big(1+\varepsilon(j,k(j)) - \delta(j,k(j)+1) - p_d^{req}(j), k(j)+1\big) &(5.41)\\ &= g(\varepsilon(j,k(j)), \delta(j,k(j)+1), p_d^{req}(j)). &(5.42)\end{aligned}$$

Therefore, in order to decrease the difference between achieved and required detection probabilities at the $j$-th point, the signal energy at the additional sensor should be such that

$$S(d_{k(j)+1}) \geq \sigma^2_{k(j)+1} g(\varepsilon(j,k(j)), \delta(j,k(j)+1)=0, p_d^{req}(j)). \tag{5.43}$$

Since the signal energy function in Eqn.(5.1) is a decaying function of distance, we can use Eqn.(5.43) to calculate the new maximum allowed distance (i.e., new collaboration radius) between the $j$-th point and the additional $(k(j)+1)$-th sensor as

$$R_c(j,k(j)+1) \leq S^{-1}(\sigma^2_{k(j)+1} g(\varepsilon(j,k(j)), \delta(j,k(j)+1)=0, p_d^{req}(j))), \tag{5.44}$$

where, $S^{-1}$ denotes the inverse of the signal energy function. The right hand side of Ineq.(5.44) gives the maximum collaboration radius necessary to improve the detection performance at the $j$-th grid point.

## 5.4 Deployment Algorithm

In this section, we introduce the (LQR_Deploy) algorithm which is based on the LQR formulation of the deployment problem and its solution that can be evaluated using the differentiation method. Algorithm 4, illustrates the steps of the LQR_Deploy sensor deployment algorithm.

Given the false alarm/detection requirements ($\mathbf{p}_f^{req}$ and $\mathbf{p}_d^{req}$, respectively) and the number of available sensor $K$, it is required to find the deployment vector $\mathbf{D}$ and the achieved detection probabilities at all points on the grid $\mathbf{p}_d^K$. Each loop of the algorithm is executed if the number of sensors on the grid (i.e., $k$) is less than the number of available sensors $K$, or if the detection requirements are not met at each point on the grid (i.e., $\mathbf{p}_d^{req} \not\preceq \mathbf{p}_d^K$). For the $j$-th grid point $\{j = 1, \ldots, N_x N_y\}$, the $B(j,i), i = 1, \ldots, N_x N_y$ entry will be evaluated as in Eqn.(5.32). At each deployment step, we can construct diagonal matrices $\mathbf{R_k}$ and $\mathbf{Q}_k$ (introduced in Eqn.(8.22)) in the following manner; The diagonal entries in $\mathbf{R}_k$ ($\mathbf{Q}_k$) corresponding to the set of points where detection requirements are met can be set to large

values (small values), whereas the $\mathbf{R}_k$ ($\mathbf{Q}_k$) entries corresponding to the set of points where detection requirements are not met can be set to smaller values (larger values). In essence, this choice reflects our desired deployment preference which is to deploy sensors in regions of the grid that have unsatisfied detection requirements or where sensors are not deployed.

We can now solve for the optimal control vector $\mathbf{u}_k$ as in Eqn.(8.23). The binary deployment vector $\mathbf{D}$ is then constructed by placing a 1 at the $\hat{i}$ index corresponding the index of the maximum value of $\mathbf{u}_k$. After deploying a sensor at the grid point corresponding to the $\hat{i}$ index, one can use the maximum collaboration radius for every point to adjust the detection thresholds $(\eta(j, k(j)), j = 1, \ldots, N_x N_y)$ such that false alarm requirements are met. This is followed by evaluating the detection probabilities at each point as in Eqn.(5.15). The final step in the algorithm is to update $k$ to reflect the fact that an additional sensor has been deployed to the grid. The algorithm terminates when all available sensors have been deployed or detection requirements have been satisfied over all points on the grid.

**Algorithm 4** Opt_Deploy Algorithm

**Input:** $\mathbf{p}_d^{req}$, $\mathbf{p}_f^{req}$, $K$.
**Outputs:** $\mathbf{p}_d^K$, $\mathbf{D}$.
**Initialization:** $k = 1$, $\mathbf{D} = zeros(N^{2,}1)$
**while** $k \leq K$ or $\mathbf{p}_d^{req} \not\preceq \mathbf{p}_d^k$ **do**
  $\mathbf{u}_k^o = \mathbf{0}_{1 \times N_x N_y}$ {% $(1 \times N_x N_y)$ zero vector}
  **for** $j = 1 : N_x N_y$ **do**
    **for** $j = 1 : N_x N_y$ **do**
      Calculate $B_k(j, i)$ as in Eqn.(5.32).
    **end for**
  **end for**
  Solve for $\mathbf{u}_k$ as in Eqn.(8.23).
  Find index $\hat{i}$, where $u_k(\hat{i}) = \max(\mathbf{u_k})$
  Place a sensor at the grid point corresponding to the index $\hat{i}$ (i.e., $\mathbf{D}(\hat{i}) = 1$).
  **for** $j = 1 : N_x N_y$ **do**
    Calculate the maximum collaboration radius $R_{\max}(j, k(j) + 1)$ in Ineq.(5.44).
    Update the detection threshold
    Calculate the achieved detection probability (i.e., $\mathbf{p}_d^k$).
  **end for**
  Increment m (i.e., $k = k + 1$).
**end while**

Using Eqn.(8.23) incurs a computational cost of $\mathcal{O}(K(3N^3 + 4N^2))$ where $N = N_x N_y$. In the following section, we introduce a low complexity sub-optimal sensor deployment algorithm that only uses knowledge of the matrix $\mathbf{B}_k$ in its implementation.

## 5.5 Sub-optimal Algorithm

In this section, we propose a low complexity deployment algorithm which we call the Sub_Opt_Deploy algorithm. To motivate this algorithm, note that it is possible to approximate the system described in Eqn.(5.33) as

$$\mathbf{x}_{k+1} \approx \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k. \tag{5.45}$$

This is especially true when $k$ becomes large. Ideally, it is desirable to deploy sensors such that the resulting $\mathbf{x}_{k+1}$ is equal to the zero vector (i.e., $\mathbf{x}_k = \mathbf{0}$ implies the detection requirements have been satisfied). Substituting $\mathbf{x}_{k+1} = \mathbf{0}$ in Eqn.(A.19) and solving for $\mathbf{u}_k$, we get the following ;

$$\mathbf{u}_k = -\mathbf{B}_k^{-1} \mathbf{x}_k. \tag{5.46}$$

The Sub_Opt_Deploy algorithm is similar to the Opt_Deploy algorithm presented earlier, with the exception that the control vector is calculated as in Eqn.(5.46). The evaluation of $\mathbf{u}_k$ in Eqn.(5.46) depends on the invertibility of the $\mathbf{B}_k$ matrix. In our case, the energy model in Eqn.(5.1) follows a power law decay which in turn ensures that the columns of $\mathbf{B}_k$ are linearly independent. That is, $\mathbf{B}_k$ in our case is full rank and invertible. We note that the computational complexity associated with evaluating $\mathbf{u}_k$ in the Sub_Opt_Deploy algorithm is $\mathcal{O}(K(N^3 + N^2))$, which is lower than that in the Opt_Deploy algorithm. In the next section, we demonstrate that the suboptimal method is comparable to the optimal approach.

## 5.6 Simulation Results

In this section, the performance of the Opt_Deploy and Sub_Opt_Deploy deployment algorithms is compared to that of different algorithms. In the first experiment, we compare the performance of the Opt_Deploy and Sub_Opt_Deploy deployment algorithms to that of the greedy and D&C algorithms introduced in [22]. The greedy deployment algorithm is a simple algorithm, in which a sensor is placed at the point on the grid with the largest difference between required and achieved detection probabilities. False alarm and detection requirements are uniform over the grid, with $P_f = 0.01$ and $P_d = 0.9$. The initial energy $S_0 = 400$ Joules, and the collaboration radius is set to $R_c = 7.76$ meters as in [22]. In this experiment, the number of points at which false alarm and detection requirements is required to be met is varied within a $30 \times 30$ grid of points. Specifically, we consider meeting the requirements at $15 \times 15 = 225$, $10 \times 10 = 100$, $5 \times 5 = 25$ and $4 \times 4 = 16$ regularly spaced points on the grid. Table 5.1, shows the minimum number of sensors needed to meet the requirements when using the four algorithms: greedy, D&C, Opt_Deploy and Sub_Opt_Deploy algorithms. From Table 5.1, it is evident that the use of the Opt_Deploy algorithm saves between 9% to 30% of the minimum number of sensors used by the D&C algorithm to satisfy the same detection/false alarm requirements. In the greedy deployment algorithm, the decision of where to place the sensor is based on the effect the sensor placement will have on the detection probability at just one point. The effect of the sensor on the points that lie in the vicinity of the point it would be placed at is not incorporated. This is in contrast to the Opt_Deploy and Sub_Opt_Deploy deployment algorithms, where the effect of sensor deployment on all points within its vicinity is incorporated through the $\mathbf{B}$ matrix.

In the second experiment, we compare the effect of using a dynamic collaboration radius versus that of a fixed collaboration radius on: (1) the number of sensors needed to satisfy requirements, and (2) the effective SE. The area of interest is a $25 \times 25$ grid of points, and the energy function parameters are: $S_0 = 100$, $d_0 = 1$ and $\kappa = 2$. The measurement noise variance is assumed to be uniform over all sensors and is set to $\sigma^2 = 1$. False alarm and

**Table 5.1**: *Minimum number of sensors required by: Greedy, D&C, Sub_Opt_Deploy and Opt_Deploy algorithms*

| Regularly spaced points | Greedy | D&C | Sub_Opt_Deploy | Opt_Deploy |
|---|---|---|---|---|
| 225 | 15 | 13 | 9 | 9 |
| 100 | 16 | 12 | 11 | 9 |
| 25 | 13 | 11 | 10 | 10 |
| 16 | 16 | 13 | 15 | 11 |

detection requirements are non-uniform as in Fig. 5.1. Table 5.2 lists the number of sensors, needed by the Opt_Deploy and Sub_Opt_Deploy algorithms, to satisfy different false alarm and detection requirements as we use a fixed and a dynamic collaboration radius. Results for the dynamic collaboration radius in Table 5.2, indicate that both the Sub_Opt_Deploy and Opt_Deploy algorithms use a comparable number of sensors to satisfy false alarm/detcetion requirements. In addition, results illustrate that the number of sensors required by the proposed algorithms when using a dynamic collaboration radius can be as much as 45% less than that required when using a fixed collaboration radius. Furthermore, Fig. 5.2 shows the effective SE (for case 1 in Table 5.2) as a function of the number of sensors deployed in the grid. We note that for both the Opt_Deploy and Sub_Opt_Deploy, the effective SE has faster convergence rates when using a dynamic collaboration radius than the rates when using a fixed collaboration radius. The use of a dynamic collaboration radius gives more flexibility for sensors to collaborate. The calculation of the fixed collaboration radius assumes that requirements will be satisfied with the deployment of a single sensor within the collaboration radius. However, the optimal collaboration radius depends on the achieved detection probability which changes with each sensor deployment. Therefore, dynamically updating the collaboration radius provides significant savings in the number of sensors.

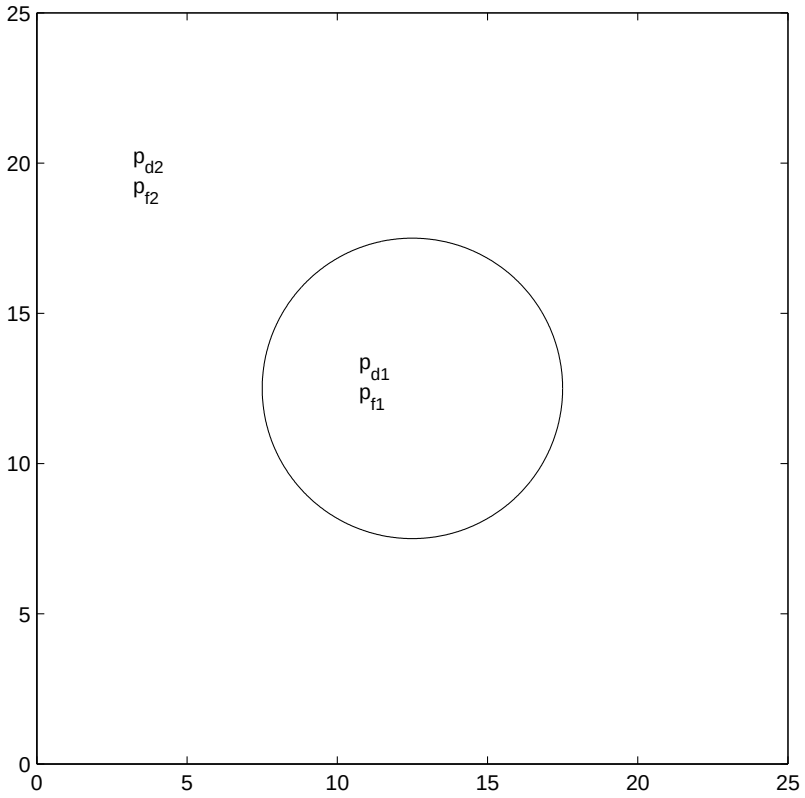



**Figure 5.1**: *Nonuniform detection requirements* $N_x = N_y = 25$

**Table 5.2**: *Comparison of number of sensors-Fixed vs. Dynamic* $R_c$

| Requirements | Sub_Opt_Deploy (Fixed Radius) | Sub_Opt_Deploy (Dynamic Radius) | Opt_Deploy (Fixed Radius) | Opt_Deploy (Dynamic Radius) |
|---|---|---|---|---|
| $p_{d1} = 0.9, p_{d2} = 0.7, p_{f1} = 0.01, p_{f2} = 0.01$ | 34 | 18 | 30 | 18 |
| $p_{d1} = 0.7, p_{d2} = 0.9, p_{f1} = 0.01, p_{f2} = 0.01$ | 35 | 19 | 30 | 19 |
| $p_{d1} = 0.7, p_{d2} = 0.9, p_{f1} = 0.001, p_{f2} = 0.01$ | 31 | 22 | 28 | 22 |
| $p_{d1} = 0.8, p_{d2} = 0.5, p_{f1} = 0.005, p_{f2} = 0.005$ | 34 | 21 | 26 | 19 |

## 5.7 Summary

In this chapter, we studied the sensor deployment problem in a data fusion based DSN detection system. We proposed two novel sensor deployment algorithms; the (Opt_Depoy) and the (Sub_Opt_Deploy) algorithms. In the (Opt_Depoy) algorithm, the deployment problem is modeled as a linear quadratic regulator (LQR) problem. This is achieved by linearizing the effect of sensor deployment on the achieved detection probability and using the effective

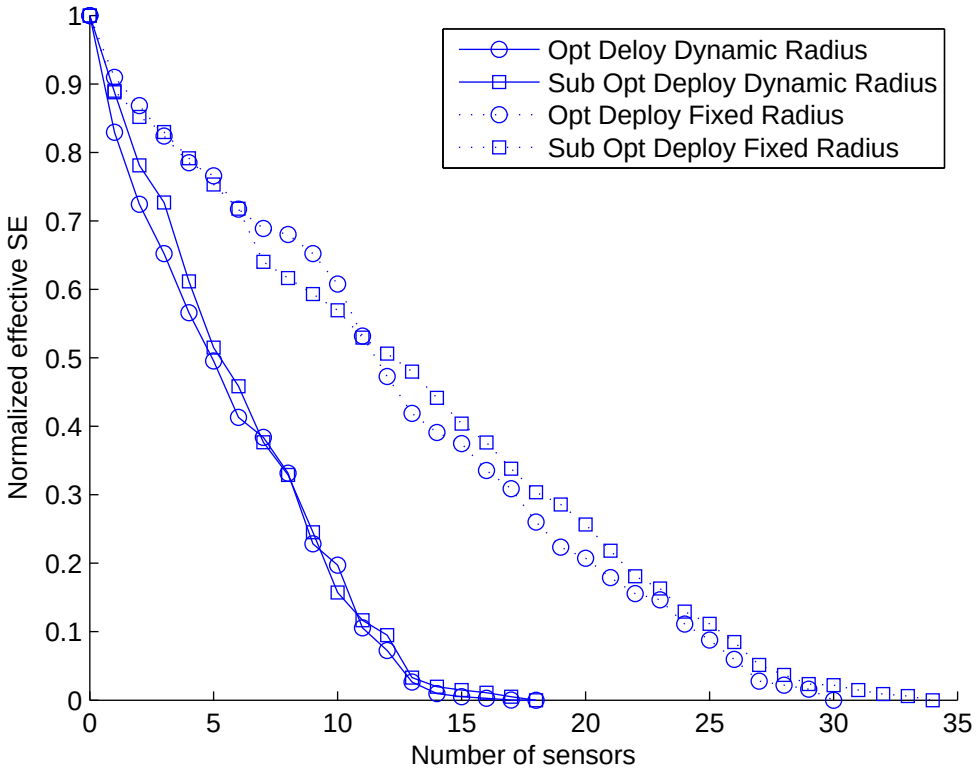



**Figure 5.2**: *SE convergence*

squared error (SE) between achieved and required detection probabilities as the cost function in the LQR formulation. In addition, we proposed the (Sub_Opt_Deploy) in which the sensor positions are calculated using a single matrix inversion operation. In both deployment algorithms, we evaluated the impact of adapting the collaboration radius on the efficiency of sensor usage. Simulation results illustrated that in comparison to the (D&C) algorithm our proposed algorithm used up to 30% fewer number of sensors to satisfy identical false alarm and detection requirements, when using a fixed collaboration radius. Additionally, we illustrated that it is possible to save up to 45% in the number of sensors needed to satisfy the false alarm/detection requirements by using a dynamic collaboration radius instead of a fixed one.

# Chapter 6

# Sensor Deployment in Decentralized Detection Networks – Majority Fusion

In this chapter, we study sensor deployment in a network employing the majority decision fusion rule. We first approximate the detection performance of the network using a result from statistical theory. Then we approximate the effect of the deployment of a single sensor as a function of the sensor position in the grid. This enables us to model the deployment process as an LQR problem. Based on this formulation, we then propose a sequential sensor deployment algorithm.

In section 6.1, the system model and the deployment problem are discussed. In section 6.2, we present our LQR formulation of the deployment problem. Our proposed LQR-based deployment algorithm is presented in section 6.3. Finally, in section 6.4, we compare the performance of the proposed algorithm versus that of a greedy deployment algorithm.

## 6.1 System Model

The area of interest is modeled as a grid $\mathcal{G}$ of $N_x \times N_y$ points. The required false alarm/detection probabilities at all points in the grid are arranged in two $N_x N_y \times 1$ vectors denoted by $\mathbf{p}_f^{req}$/ $\mathbf{p}_d^{req}$, respectively. We assume that sensors have an exponential sensing model [18]. Specifically, if the distance between a sensor and a point of interest is $d$ meters, then the probability

of the sensor detecting a target located at that point $p_{\text{detect}}$ is given as

$$p_{\text{detect}} = \begin{cases} e^{-\tau d} & \text{if } d \leq R \\ 0 & \text{if } d > R \end{cases} \tag{6.1}$$

where, $\tau$ is a decay rate that depends on sensor design and $R$ is the detection radius. We assume that all sensors to be deployed are identical and have a common $\tau$ and $R$. We further assume that all sensors have a common false alarm rate of $p_f^s$.

In a general decision fusion based system, sensors make local decisions regarding the existence (hypothesis $H_1$) or absence (hypothesis $H_0$) of a phenomena/target. The local decision $u_i$ made by the $i$-th sensor corresponds to the index of the hypothesis decided upon. Local decisions from multiple sensors are communicated over an error-free communication channel to a fusion center where a global decision $u_0$ regarding the two hypothesis is made. In our model, the fusion rule we employ is the counting rule. Assuming $k(j)$ sensors report their decisions to the FC are involved in the decision process for the $j$-th grid point, then its associated decision statistics $U$ is given as $U(j) = \sum_{i=1}^{k(j)} u_i$. The decision rule corresponds to

$$u_0(j) = \begin{cases} 1 \text{ (i.e., } H_1 \text{ is true)} & \text{if } U(j) \geq T(j,k(j)) \\ 0 \text{ (i.e., } H_0 \text{ is true)} & \text{if } U(j) < T(j,k(j)) \end{cases} \tag{6.2}$$

where, $T(j,k(j))$ is the decision threshold. Therefore, the system's overall false alarm and detection probabilities at the $j$-th grid point with $k(j)$ sensor decisions ($p_f(j,k(j))$ and $p_d(j,k(j))$, respectively) are given as

$$p_f(j,k(j)) = Pr(U(j) \geq T(j,k(j)) \mid H_0 \text{ is true}) \tag{6.3}$$

$$p_d(j,k(j)) = Pr(U(j) \geq T(j,k(j)) \mid H_1 \text{ is true}) \tag{6.4}$$

The decision threshold $T(j,k(j))$ is chosen so as to satisfy the false alarm requirement $p_f^{req}(j)$. We arrange the achieved false alarm and detection probabilities, after a total of $K$ deployed sensor, in two $N_x N_y \times 1$ vectors $\mathbf{p}_f^K$ and $\mathbf{p}_d^K$, respectively.

The problem of interest in this paper, can now be stated as follows: Determine the positions on the grid where a given number of sensors ($K$) are to be deployed in order to minimize the squared difference between achieved and required detection probabilities, without violating false alarm requirements. Mathematically, it can be stated as follows

$$\underset{\mathbf{u}}{\operatorname{argmin}} \sum_{j:p_d^K(j)<p_d^{req}(j)} (\mathbf{p}_d^K(j) - \mathbf{p}_d^{req}(j))^2$$
$$\text{subject to} \begin{cases} \mathbf{p}_f^K \leq \mathbf{p}_f^{req} \\ \mathbf{1}^T\mathbf{D} = K \end{cases} \tag{6.5}$$

where, $\mathbf{D}$ is the deployment vector. The deployment vector is an $N_xN_y \times 1$ vector. Its entries indicate the number of sensors at each point on the grid, and take values of either 0 or 1. $\mathbf{1}^T$ indicates the transpose of an $N_xN_y \times 1$ vector, with all entries set to 1. Note that in Eqn.(6.5), the squared error is taken at the grid points where detection requirements are not satisfied.

## 6.2 Optimal Control Formulation

Our approach to solving the deployment problem relies on modeling the effect of deploying a sensor on the overall false alarm and detection probabilities. Modeling the change in the overall false alarm rate is straightforward since the overall false alarm rate at a point depends only on the number of sensors covering that point and their individual false alarm rates (i.e., there is no distance dependency). On the other hand, the overall detection probability at any point on the grid depends on the number of sensors incorporated in the decision process for that point and their positions relative to the point of interest. This is due to the distance dependent sensing model of the sensors. This dependency complicates modeling a sensor's effect on the overall detection performance. In this case, the overall detection probability does not have a closed form description. In order to analytically model and solve the deployment problem, we use a theorem by Hoeffding [60] which gives

a bound on the achieved detection probability. Using this bound enables us to linearly approximate the effect of a sensor deployment on the overall detection probability at any given grid point. The linear approximation of the overall detection probability as a function of sensor deployment combined with a squared error cost function equivalent to the one in Eqn.(6.5) enables us to express the deployment problem as a linear quadratic regulator (LQR) problem. We note that this approximation is only used for deployment purposes and after the sensor is deployed, the actual resulting overall detection probability is calculated, and used in our proposed system evolution model. This means that any discrepancy between using the approximate value and the actual one will be accounted for and corrected with the evolution of the system. Details of the system approximation and the control formulation follow in the next few subsections.

Since sensors have a common false alarm rate $p_f^s$, the decision statistics $U(j)$ associated with the $j$-th point, under hypothesis $H_0$, follows a binomial distribution with parameters $p_f^s$ and $k(j)$ where $k(j)$ is the number of sensors covering that point. Therefore, the overall false alarm probability at the $j$-th point, is given as

$$p_f(j,k(j)) = Pr(U(j) \geq T(j,k(j))|H_0) \tag{6.6}$$

$$= \sum_{n_j=\lceil T(j,k(j))\rceil}^{k(j)} \binom{k(j)}{n_j} (p_f^s)^{n_j}(1-p_f^s)^{k(j)-n_j}. \tag{6.7}$$

where, $\lceil T(j,k(j))\rceil$ is the smallest integer greater than or equal to $T(j,k(j))$. As noted earlier, the overall detection probability is difficult to characterize. Under hypothesis $H_1$, the $i$-th ($i = 1,\ldots,k(j)$) sensor reports a detection decision $u_i$ with a distance- dependant detection probability (i.e., success probability) $p(j,i)$. The detection probabilities ($p(j,i),\ i = 1,\ldots,k(j)$) need not be uniform since sensors might be at different distances from the point of interest. Since the success probabilities are not necessarily equal, the distribution of the decision statistics $U(j) = \sum_{i=1}^{k(j)} u_i$ does not follow a binomial distribution. In fact, the distribution of $U(j)$ is a Poisson trial distribution since each random variable $u_i$ is Bernoulli distributed with a success probability of $p(j,i)$. A closed form description of

the poisson trial distribution or its tail does not exist. However, its CDF (and therefore the CCDF) can be bounded based on the theorem described below [60] or approximated as in [61] and references therein. For ease of notation and generality, we drop the dependence on $j$ when stating Theorem 6.2.1 below.

**Theorem 6.2.1.** *If $U = \sum_{i=1}^{k} u_i$ where each $u_i$ is a Bernoulli random variable with a success probability of $p_i$, and $c$ is an integer, then [60]*

$$Pr(U \leq c) \quad \leq \quad Pr(X(k, \overline{p}) \leq c) \quad \text{if } 0 \leq c \leq k\overline{p} - 1 \tag{6.8}$$

$$Pr(U \leq c) \quad \geq \quad Pr(X(k, \overline{p}) \leq c) \quad \text{if } k\overline{p} \leq c \leq k \tag{6.9}$$

*where, $X(k, \overline{p})$ is a binomial random variable with $k$ trials and $\overline{p}$ success probability. The success probability is given as $\overline{p} = \frac{1}{k}\sum_{i=1}^{k} p_i$.*

*Proof.* See [60] for details. □

Theorem 6.2.1 gives upper or lower bounds (depending on the value of $c$) for the distribution of a random variable $U$ which follows a poisson trial distribution. Both bounds are given in terms of a binomial distribution with a number of trials $k$ and a success probability $\overline{p}$.

In our case, the binomial approximation serves as a lower or upper bound for the overall detection probability depending upon the number of sensors ($k(j)$) as well as their individual distance dependant probabilities of detection. However, we note that any approximation error resulting from using bounds will be accounted for after sensor deployment. This is because, in this paper, the actual detection probability (i.e., not its upper/lower bound) is evaluated after each sensor deployment. Let $X \sim Bino(n, p)$ denote a random variable (r.v.) $X$, that follows a binomial distribution with success rate $p$ and $n$ trials. It is possible to approximate the tail probability of $X$ as [58], [62]

$$Pr(X \geq x) \approx Q(\frac{x - 0.5 - np}{\sqrt{npq}}) \tag{6.10}$$

where $q = 1 - p$, and $Q(\cdot)$ is the tail of a standard Gaussian distribution defined as $Q(y) = \frac{1}{\sqrt{2\pi}} \int_y^\infty e^{\frac{-t^2}{2}} dt$. Various rules of thumb have been suggested for use of the approximation in Eqn.(6.10). Two such rules are as follows [62]:

1. When $npq > 9$.

2. When $np > 9$ for $p < 0.5$.

However, it was shown that the maximum approximation error for any value of $n$ and $p$ is $0.14(npq)^{-0.5}$ [62]. We note that in our deployment problem, the value of the success probability $\overline{p}$ is such that $\overline{p} > 0.5$. Therefore, the $Q(\cdot)$ approximation is a lower bound of the detection probability. Hence, in effect we are underestimating the achieved detection probability.

Furthermore, it is possible to approximate the $Q$ function as [59]

$$Q(y) \approx 1 - \frac{1}{1 + e^{-\sqrt{2}y}}. \tag{6.11}$$

Therefore, it is possible to approximate $p_d(j, k(j))$, the detection probability at the $j$-th point when $k(j)$ sensors are involved in the decision process as

$$p_d(j, k(j)) \approx Q\left(\frac{T(j, k(j)) - 0.5 + k(j)\overline{p}(j, k(j))}{\sqrt{k(j)\overline{p}(j, k(j))\,\overline{q}(j)}}\right) \tag{6.12}$$

where, $\overline{p}(j, k(j)) = \frac{1}{k(j)} \sum_{i=1}^{k(j)} p(j, i)$ is the mean of the detection probabilities of the sensors involved in the detection process at the $j$-th point of the grid, and $\overline{q}(j, k(j)) = 1 - \overline{p}(j, k(j))$.

Using the approximation in (6.11) and rearranging the terms in Eqn.(6.12), we get

$$\ln\left(\frac{1}{1 - p_d(j, k(j))}\right) = \sqrt{2}\left(\frac{k(j)\overline{p}(j, k(j)) + 0.5 - T(j, k(j))}{\sqrt{k(j)\overline{p}(j, k(j))\,\overline{q}(j, k(j))}}\right). \tag{6.13}$$

Let $m(j, k(j)) = \ln\left(\frac{1}{1 - p_d(j, k(j))} - 1\right)$. Denoting the decision threshold as $T(j, k(j))$, it is possible using Eqn.(6.13) to express $m(j, k(j) + 1)$ as follows

$$m(j, k(j) + 1) = m(j, k(j)) + \sqrt{2}\left(\frac{\overline{p}(j, k(j)) - \delta^{k(j)+1}(j)}{\alpha_{k(j)}\sqrt{k(j)\overline{p}(j, k(j) + 1)\,\overline{q}(j, k(j) + 1)}}\right) - \gamma_{k(j)} m(j, k(j)) \tag{6.14}$$

where, $\delta^{k(j)+1}(j)$ is the change in the decision threshold (i.e., $T(j,k(j)+1) = T(j,k(j)) + \delta^{k(j)+1}(j)$), $\alpha_{k(j)} = \sqrt{\frac{k(j)+1}{k(j)}}$, and $\gamma_{k(j)} = \frac{1}{\alpha_{k(j)}} - 1$. In essence, Eqn.(6.14) approximates the effect of adding of a sensor on the overall detection probability. The false alarm probability requirement is always met by a proper choice of the decision threshold.

Let $p_d^{req}(j)$ denote the required detection probability at point $(j)$, then $m^{req}(j) = \ln(\frac{1}{1-p_d^{req}(j)} - 1)$. Furthermore, define $x(j,k(j))$ as

$$x(j,k(j)) = m(j,k(j)) - m^{req}(j), \tag{6.15}$$

then it is possible to express the change in $x(j,k(j))$ after adding an additional sensor as

$$x(j,k(j)+1) = (1-\gamma_{k(j)})x(j,k(j)) + \sqrt{2}\left(\frac{\overline{p}(j,k(j)+1) - \delta^{k(i)+1}(j)}{\alpha_{k(j)}\sqrt{k(j)\overline{p}(j,k(j)+1)\,\overline{q}(j,k(j)+1)}}\right) - \gamma_{k(j)}m^{req}(j). \tag{6.16}$$

Considering all points on the grid, then it is possible to express Eqn.(6.16) in matrix form as

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k \tag{6.17}$$

where, the matrix subscripts refer to the total number of sensors deployed in the grid. The matrix $\mathbf{A}_k$ is a square diagonal matrix of dimension $N_xN_y$, where the $j$-th diagonal element corresponds to the $j$-th point on the grid and is given as $\mathbf{A}_k(j,j) = (1-\gamma_{k(j)})$. The matrix $\mathbf{B}_k$ is also a square matrix of dimension $N_xN_y$. The $(j,i)$-th entry corresponds to the change in the average overall detection probability at the $j$-th point on the grid if a sensor were to be deployed at the $i$-th point. If the distance between points $(i)$ and $(j)$ is less than $R$, then

$$\mathrm{B}_k(j,i) = \sqrt{2}\left(\frac{\overline{p}_d(j,k(j)+1) - \delta(j)^{k(j)+1}}{\alpha_{k(j)}\sqrt{k(j)\overline{p}_d(j,k(j)+1)\overline{q}_d(j,k(j)+1)}}\right) - \gamma_{k(j)}m^{req}(j), \tag{6.18}$$

otherwise it is set to 0. Here, $\mathbf{u}_k$ is the deployment vector, which indicates the positions of sensors that are to be deployed at each point of the grid.

Note that the squared error (SE) between achieved and required detection probabilities, can be described as the weighted quadratic norm of the state ($\mathbf{x}_k$) of the system described in Eqn.(6.17). Assuming that the weighted quadratic norm of the system state is chosen as the cost function and the deployment vector corresponds to a control vector, we are motivated to solve the deployment problem as an optimal control problem. Here, the objective is to determine the control vector that would minimize the cost function. That is, the deployment problem in Eqn.(6.5) can be restated as

$$\begin{aligned} \underset{\mathbf{u}_k}{\operatorname{argmin}}\, J &= \frac{1}{2}\mathbf{x}_K^T\mathbf{Q}_f\mathbf{x}_K + \frac{1}{2}\sum_{k=0}^{K-1}(\mathbf{x}_k^T\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k) \\ \text{subject to} &\begin{cases} \mathbf{p}_f^K \leq \mathbf{p}_f^{req} \\ \mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k,\ k = 0, 1, \ldots, K-1 \\ \mathbf{1}^T\mathbf{D} = K \end{cases} \end{aligned} \tag{6.19}$$

where, $\mathbf{Q}$, $\mathbf{Q}_f$ and $\mathbf{R}$ are symmetric positive definite weighting matrices. The squared error cost function penalizes both positive and negative deviations from the required detection probability profile. To avoid incurring a penalty for satisfying/exceeding detection requirements, the error terms corresponding to a point where the detection requirement has been met/exceeded is set to zero in $J$. The optimal control problem corresponding to our system is the linear quadratic regulator (LQR) problem.

In this chapter, we have adopted the sweep method due to its simplicity and intuitive interpretation. The resulting optimal control vector $\mathbf{u}_k$ has continuous entries. In order to have a binary integer solution, a 1 is placed at the index where $\mathbf{u}_k$ is maximum and a 0 is placed at the remaining positions. That is, a sensor is placed at the location corresponding to the index where $\mathbf{u}_k$ is maximum.

In the next section, we propose a sensor deployment algorithm based on our LQR formulation.

## 6.3 Deployment Algorithm

In order to guarantee that both false alarm and detection requirements are met, we propose the optimal control based algorithm illustrated in Algorithm 5.

**Algorithm 5** Optimal Control Based Algorithm

**Inputs**: $\mathbf{p}_f^{req}$, $\mathbf{p}_d^{req}$ and $K$
**Outputs**: $\mathbf{D}$, $\mathbf{p}_f^K$ and $\mathbf{p}_d^K$
**Initialization**: $\mathbf{D} = \mathbf{0}$ and $k = 0$
**while** $\mathbf{p}_f^k \succeq \mathbf{p}_f^{req}$ and $k < K$ **do**
    Find $I_\delta$ s.t. $\mathbf{p}_f^{req}(I_\delta) < \mathbf{p}_f^k(I_\delta)$
    Solve for $\mathbf{u}_k$ using Eqn.(3.25)
    Find $i_{\max}$ s.t. $\mathbf{u}_k(i_{\max}) \geq \mathbf{u}_k(i)$, $\forall i \in I_\delta$
    Deploy a sensor at grid point $i_{\max}$ (i.e., set $\mathbf{D}(i_{\max}) = 1$)
    Increment $k$
    Change detection thresholds
    Calculate $\mathbf{p}_f^k$ and $\mathbf{p}_d^k$
**end while**
**while** $\mathbf{p}_d^k \preceq \mathbf{p}_d^{req}$ and $k < K$ **do**
    Find $I_\delta$ s.t. $\mathbf{p}_d^{req}(I_\delta) > \mathbf{p}_d^k(I_\delta)$
    Set $\mathbf{x}(j) = 0$, $\forall j \notin I_\delta$ (equivalent to taking the SE at unsatisfied points )
    Solve for $\mathbf{u}_k$ using Eqn.(3.25)
    Find $i_{\max}$ s.t. $\mathbf{u}_k(i_{\max}) \geq \mathbf{u}_k(i)$, $\forall i \in I_\delta$
**end while**

Giving more importance to satisfying the false alarm requirements, in the first while loop the algorithm determines the points ($I_\delta$) at which false alarm requirements are not met. Giving a higher priority for sensor deployment at these points will give more freedom in choosing the suitable decision threshold which will facilitate satisfying the false alarm requirements at those points. After calculating the control vector $\mathbf{u}_k$, a sensor is deployed at the the point in $I_\delta$ that corresponds to the entry with the largest value in $\mathbf{u}_k(I_\delta)$. This will have the additional effect of satisfying the detection requirements at these points. In the case that false alarm requirements have been met at all points, the second while loop is concerned with meeting the detection requirements. The contribution of the points $I_\delta$, at which detection requirements are not met, is considered in calculating the deployment vector $\mathbf{u}_k$. This is done by setting entries in $\mathbf{x}$ corresponding to points with satisfied requirements to

0. After each sensor deployment, the decision thresholds are modified in order to satisfy the false alarm requirements. The algorithm terminates when all available sensors are deployed or, if $K$ is sufficiently large, when all detection requirements are met. We note that the complexity associated with the LQR based deployment algorithm is $\mathcal{O}(K(3N^3 + 4N^2))$ where $N = N_x N_y$.

## 6.4 Simulation Results

In this section, we compare the performance of two deployment algorithms, namely: a greedy algorithm and the optimal control based algorithm. In each step of the greedy algorithm, a sensor is deployed at the grid point where the difference between achieved and required detection probabilities is maximum. In the first experiment, the area of interest is modeled as a uniformly spaced $25 \times 25$ grid. Sensors to be used are identical and share the parameters $\tau = 0.1$, $R = 6$, and $p_f^s = 0.05$. The false alarm and detection requirements are uniform over the grid. The number of sensors needed to satisfy different detection requirements ($p_d^{req} = 0.6,\ 0.7$ and $0.9$) by each algorithm is provided in Table 6.1. The false alarm requirement is set to $p_f^{req} = 0.01$ for all 3 cases. From Table 6.1, it is evident that the optimal control based deployment (i.e., LQR approach) algorithm uses up to 34% fewer sensors than the greedy deployment algorithm. In the second experiment, we study the performance of the deployment algorithms as the false alarm rate is varied. The setup of the second experiment is similar to that of the first experiment. The detection probability requirement is set to $p_d^{req} = 0.8$. Using the optimal control based algorithm, the savings in the number of sensors used is as high as 25% relative to the greedy algorithm as shown in Table 6.2 .

In the third experiment, we compare the number of sensors needed by the deployment algorithms as the detection decay rate $\tau$ is varied. The grid is a $25 \times 25$ and the experiment parameters are; $R = 5$, $p_d^{req} = 0.8$, $p_f^{req} = 0.05$ and the sensor's false alarm rate is $p_f^s = 0.1$. The simulation results listed in Table 6.3, illustrate that as $\tau$ increases, the number of

**Table 6.1**: *Number of sensors for different $p_d^{req}$: Greedy vs. Opt_Deploy*

| $p_d^{req}$ | Greedy | Opt_Deploy |
|---|---|---|
| 0.6 | 21 | 16 |
| 0.7 | 32 | 21 |
| 0.9 | 60 | 47 |

**Table 6.2**: *Number of sensors for different $p_f^{req}$: Greedy vs. Opt_Deploy*

| $p_f^{req}$ | Greedy | Opt_Deploy |
|---|---|---|
| 0.05 | 27 | 22 |
| 0.01 | 33 | 23 |
| 0.005 | 48 | 41 |

sensors needed to satisfy the detection requirements increases as well. This is expected as a sensor's ability to detect distant targets is reduced with an increase in $\tau$. We also note that a saving of up to 50% in the number of sensors can be achieved by using the LQR deployment algorithm instead of a greedy algorithm.

We next compare the effect of the individual sensor false alarm probability $p_f^s$ on the number of sensors needed by the deployment algorithms. The false alarm and detection requirements are uniform over a $25 \times 25$ grid and are given as $p_d^{req} = 0.8$ and $p_f^{req} = 0.05$, respectively. The decay rate and the collaboration radius are set to $\tau = 0.05$ and $R = 5$, respectively. Results in Table 6.4, demonstrate that more sensors are needed as $p_f^s$ increases. This is because, information from more sensors need to be fused in order to satisfy the false alarm requirements. Furthermore, using the LQR deployment algorithm can save up to 50%

**Table 6.3**: *Number of sensors for different $\tau$: Greedy vs. Opt_Deploy*

| $\tau$ | Greedy | Opt_Deploy |
|---|---|---|
| 0.01 | 19 | 15 |
| 0.05 | 32 | 16 |
| 0.1 | 40 | 20 |
| 0.15 | 61 | 25 |

**Table 6.4**: *Number of sensors for different $p_f^s$: Greedy vs. Opt_Deploy*

| $p_f^s$ | Greedy | Opt_Deploy |
|---|---|---|
| 0.1 | 32 | 16 |
| 0.3 | 55 | 45 |
| 0.4 | 87 | 68 |

**Table 6.5**: *Number of sensors for different R: Greedy vs. Opt_Deploy*

| $R$ | Greedy | Opt_Deploy |
|---|---|---|
| 3 | 73 | 36 |
| 5 | 40 | 20 |
| 7 | 21 | 15 |

in the number of required sensors in comparison to the greedy algorithm.

In the following experiment, we investigate the effect of changing $R$ (i.e., the collaboration radius) on the performance of the deployment algorithms in terms of the number of sensors needed to meet false alarm and detection requirements. The setup of this experiment is similar to the previous one with an $\tau = 0.1$ and $p_f^s = 0.1$. Numerical results listed in Table 6.5 show a reduction in the number of required sensors as $R$ increases. This is due to the fact that more sensors are able to collaborate, as $R$ is increased, reducing the number of sensors needed to meet false alarm and detection requirements.

We finally compare the performance of the algorithms when the detection requirements are not uniform over the grid as in Fig. 6.1. The parameters are as follows; $R = 5$, $\tau = 0.05$ and $p_f^s = 0.1$. Table 6.4 lists the number of sensors needed to meet the non-uniform performance requirements as the performance requirements are varied. Simulation results indicate that the number of sensors used by the LQR deployment algorithm can be as much as 20% less than that needed by the greedy deployment algorithm. Fig. 6.2 shows the positions of sensors deployed by both algorithms in the grid for Case 4 of Table 6.4 .To illustrate the efficacy of using the deployment algorithms in meeting the detection requirements we use the effective squared error (SE) measure. The effective SE refers to the squared differ-

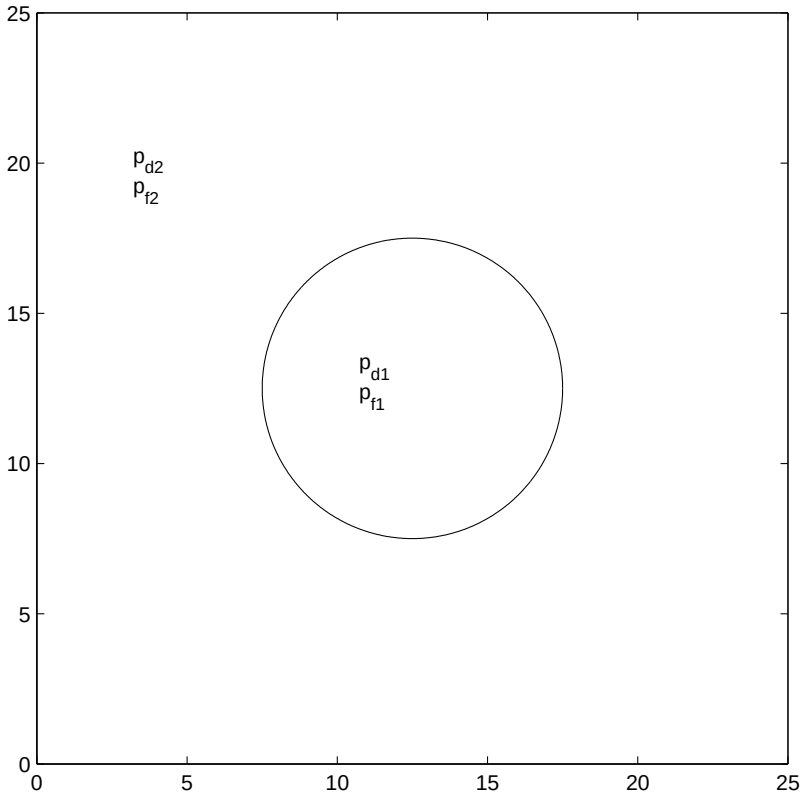



**Figure 6.1**: *Nonuniform requirements* $N_x = N_y = 25$

ence between achieved and required detection probabilities at the points where detection requirements are not met. Fig . 6.3 shows the effective SE as a function of the number of sensors deployed in the grid for Case 4 in Table 6.4 . One can see that the LQR deployment algorithm is more efficient in terms of meeting the requirements than the greedy algorithm since its SE converges to zero at a faster rate than that of the greedy algorithm.

Number of sensors for nonuniform requirements: Greedy vs. Opt_Deploy

| Requirements | Greedy | Opt_Deploy |
|---|---|---|
| $p_{d1} = 0.8, p_{d2} = 0.6, p_{f1} = 0.05, p_{f2} = 0.05$ | 25 | 15 |
| $p_{d1} = 0.6, p_{d2} = 0.8, p_{f1} = 0.05, p_{f2} = 0.05$ | 35 | 24 |
| $p_{d1} = 0.6, p_{d2} = 0.7, p_{f1} = 0.05, p_{f2} = 0.07$ | 25 | 15 |
| $p_{d1} = 0.6, p_{d2} = 0.7, p_{f1} = 0.07, p_{f2} = 0.05$ | 19 | 12 |

Simulation results illustrate that the proposed deployment algorithm consistently outperforms the greedy algorithm with respect to number of sensors needed to satisfy the

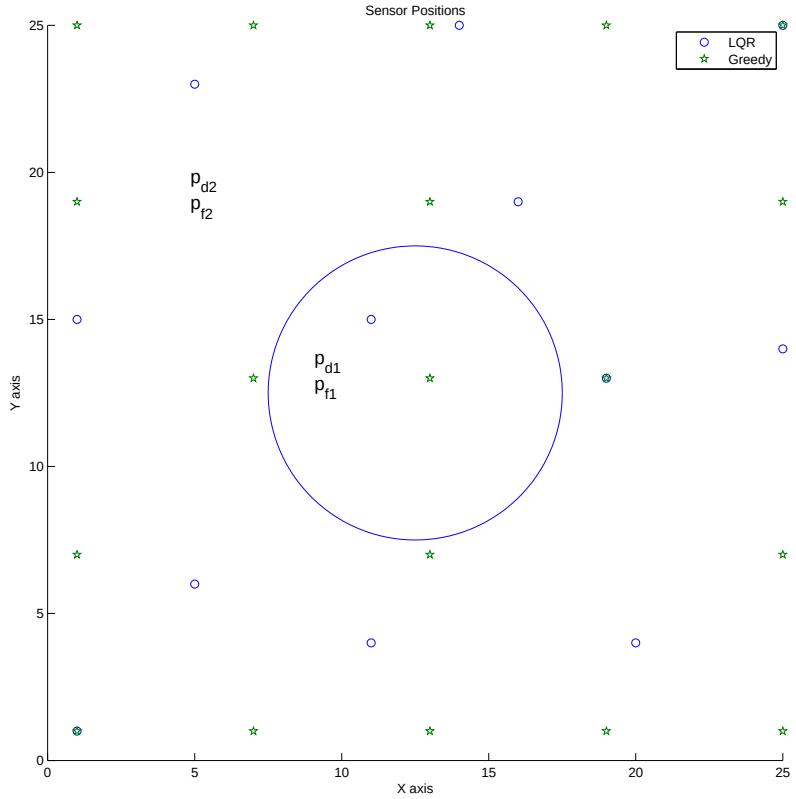



**Figure 6.2**: *Sensor positions for requirements:* $p_{d1} = 0.6, p_{d2} = 0.7, p_{f1} = 0.07, p_{f2} = 0.05$

detection/false alarm requirements. This is due to the fact that the proposed algorithm incorporates more information in making the deployment decision than the greedy algorithm. In a greedy algorithm, sensors are deployed based on the effect the addition of a sensor will have on a single point. In contrast, the proposed algorithm incorporates the approximate effect the deployment of a sensor at any candidate position will have on all the points that fall within its detection radius $R$.

## 6.5 Summary

We investigated the sensor deployment problem in a decision fusion based DSN system. Specifically, we determined the positions where a given number of sensors are to be deployed

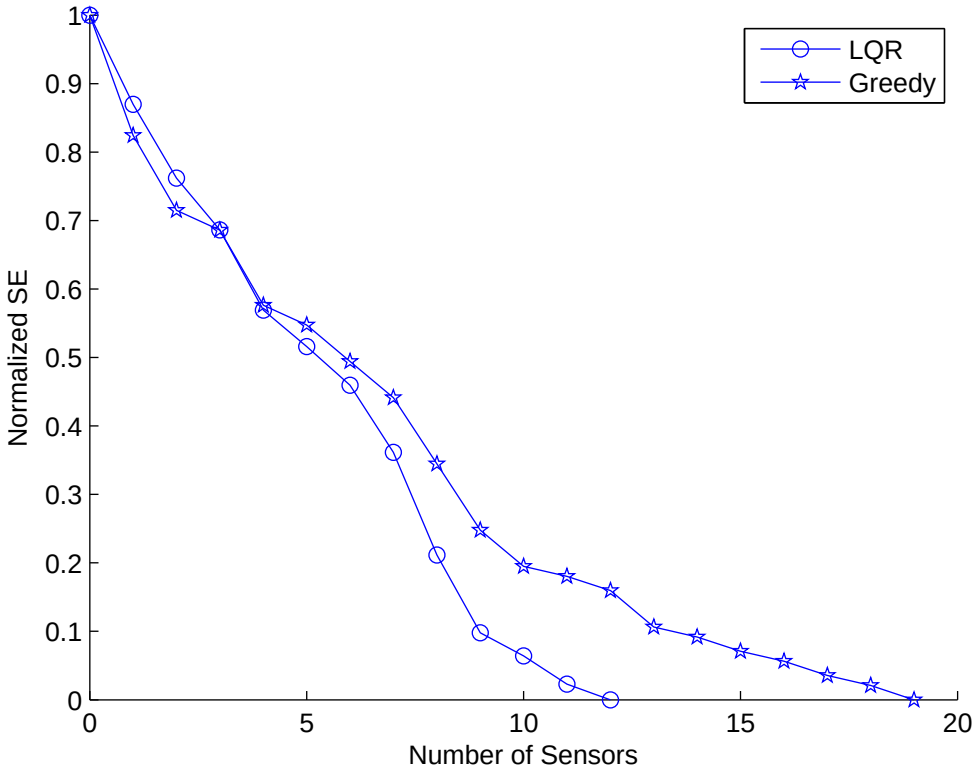



**Figure 6.3**: *SE convergence for requirements:* $p_{d1} = 0.6, p_{d2} = 0.7, p_{f1} = 0.07, p_{f2} = 0.05$

to meet detection and false alarm requirements. Unlike prior efforts in this area, we present a rigorous approach to the sensor deployment problem. Using results from non-parametric statistical theory, we linearly approximated the effect of a single sensor deployment. Using this approximation in conjunction with a squared error cost function enabled us to model the deployment problem as an LQR problem. Based on this, we proposed a sequential LQR-based sensor deployment algorithm that aims at satisfying both false alarm and detection requirements. Finally, we compared the performance of this algorithm against a greedy deployment algorithm in terms of the efficiency of meeting detection requirements (as measured by the effective SE ) and the number of sensors needed to satisfy performance requirements. Simulation results illustrated that using the proposed LQR algorithm the savings in the number of sensors required can be as much as 50% relative to a greedy algorithm.

In this chapter, the decision fusion rule used was the majority or counting rule which is suboptimal. In the next chapter, we investigate sensor deployment in a network employing the optimal decision fusion rule.

# Chapter 7

# Sensor Deployment in Decentralized Detection Networks – Optimal Decision Fusion

In this chapter, we consider the sensor deployment problem when using the optimal decision fusion rule [44]. Due to the fact that the tail of the decision statistic distribution has no closed form expression, this problem received little attention. In this work, we propose a closed form approximation of the tail probability of the decision statistics distribution. Using this approximation, we are able to model sensor deployment as an LQR problem. Based on this formulation, we propose a sensor deployment algorithm to be used in a network employing the optimal decision rule.

In section 7.1, we review the system setup and the deployment problem formulation. In section 7.2, we present our proposed closed form approximation of the false alarm and detection probabilities when using the optimal decision fusion rule. Based on this approximation, we discuss our LQR formulation of the deployment problem. In section 7.3, we propose an LQR-based sensor deployment algorithm. Simulation results are discussed in section 7.4 where the performance of our proposed algorithm is shown to outperform that of a greedy deployment algorithm.

## 7.1 System Model

We model the area of interest as a grid $\mathcal{G}$ of $N_x \times N_y$ points. The false alarm and detection requirements are given in two $N_x N_y \times 1$ vectors, $\mathbf{p}_f^{req}$ and $\mathbf{p}_d^{req}$, respectively with each entry corresponding to the false alarm and detection requirement at each grid point. The DSN network employs a parallel decision fusion detection scheme [44],[63]. In a network of $N$ sensors, the sensor deployed at the $i$-th grid point makes a local decision $u_i$ on the absence of a target (hypothesis $H_0$) or its presence (hypothesis $H_1$). We assume that every individual sensor has the same false alarm rate $p_f$. However, the analysis that follows can be easily extended to the case of sensors with different false alarm probabilities. On the other hand, the probability of a sensor detecting a target depends on the distance $d$ separating them. Assume a sensor is located at the $i$-th grid point, and let $d(i,j)$ denote the distance separating it from a target located at the $j$-th grid point, then the probability of the sensor detecting the target denoted as $p_d(i,j)$ is given as

$$p_d(i,j) = \begin{cases} e^{-\tau d(i,j)} & \text{if } d(i,j) \leq R \\ 0 & \text{if } d(i,j) > R \end{cases} \tag{7.1}$$

where, $\tau$ is the sensor's detection decay rate and $R$ is a detection radius. We assume that the detection radius $R$ is such that the detection probability at a point further from $R$ is negligible.

If $k(j)$ sensors are involved in the decision process for the $j$-th grid point (i.e., $k(j)$ sensors are at a distance less than $R$ from the $j$-th point), the fusion center (FC) combines the sensors' individual decisions and makes an overall detection decision $u_0(j)$ regarding the existence/absence of a target at the $j$-th grid point. In this paper, the FC uses the optimal decision fusion rule in order to make its decision [44], [64]. The FC constructs the decision statistic $Z(j)$ given as

$$Z(j) = \sum w(j,i) u_i, \tag{7.2}$$

where,

$$w(j,i) = \begin{cases} \log \frac{p_d(j,i)(1-p_f)}{p_f(1-p_d(j,i))} & \text{if } d(i,j) \leq R \\ 0 & \text{if } d(i,j) > R \end{cases} \tag{7.3}$$

The weight $w(j,i)$ quantifies the relative importance that the decision from the $i$-th sensor gets in comparison to the decisions coming from the other sensors. Note that the weight $w(j,i)$ becomes negative infinity if the detection probability $p_d(j,i)$ is zero. However, in Eqn.(8.3) the detection probability is equal to zero only if the distance between target and sensor is greater than $R$. Therefore, during the construction of the decision statistic $Z(j)$ for the $j$-th grid point, only those sensors that are within a radius of $R$ of the $j$-th point are included in evaluating $Z(j)$ and the weights. This insures that no negative infinite weights are used in calculating $Z(j)$. In addition, we note that the weight $w(j,i)$ is positive in practice. The weight $w(j,i)$ is negative only if the detection probability is less than that of the false alarm probability (i.e., $p_d(j,i) < p_f$ - which is highly undesirable in a practical system). We note that a practical value for a sensor's false alarm probability $p_f$ lies in the range $(0, 0.1]$. Using a worst case value of $p_f = 0.1$ implies that a negative weight only occurs if the detection probability $p_d(j,i) < 0.1$. This is a extremely low detection probability that may motivate us to reject the sensor for any detection application. In practice, it is reasonable to expect that sensors with a high detection probability (e.g., $p_d(j,i) > 0.5$) will be used. Therefore, the random variable $Z(j)$ will always have a non-negative support for practical values of detection and false alarm probabilities.

The final decision rule corresponds to

$$u_0(j) = \begin{cases} 1 \text{ (i.e., } H_1 \text{ is true)} & \text{if } Z(j) \geq \eta(j,k(j)) \\ 0 \text{ (i.e., } H_0 \text{ is true)} & \text{if } Z(j) < \eta(j,k(j)) \end{cases} \tag{7.4}$$

where, $\eta(j,k(j))$ is the decision threshold. The false alarm probability $p_f(j,k(j))$ and detection probability $p_d(j,k(j))$ at the $j$-th point are given as

$$p_f(j,k(j)) = Pr(Z(j) \geq \eta(j,k(j)) \ / H_0 \text{ is true}) \tag{7.5}$$

$$p_d(j,k(j)) = Pr(Z(j) \geq \eta(j,k(j)) \ / H_1 \text{ is true}). \tag{7.6}$$

Since the calculation of $Z(j)$ under both hypothesis involves using the weights $\{w(j,i)$, both false alarm and detection probability depend on the distance separating the sensors from the point of interest (i.e., the detection performance is affected by the sensors' spatial distribution). Therefore, it is important to devise a deployment strategy that will take into account the performance requirements and sensor characteristics. We can now state the problem that we study in this paper as follows: Given false alarm and detection requirements, where should a given number of sensors $K$ be deployed in order to meet/minimize the difference between achieved and required detection probabilities while attempting to satisfy false alarm requirements. That is,

$$\begin{aligned} \underset{\mathbf{u}}{\operatorname{argmin}} & \sum_{j:p_d^K(j)<p_d^{req}(j)} (\mathbf{p}_d^K(j) - \mathbf{p}_d^{req}(j))^2 \\ \text{subject to} & \begin{cases} \mathbf{p}_f^K \leq \mathbf{p}_f^{req} \\ \mathbf{1}^T\mathbf{D} = K \end{cases} \end{aligned} \tag{7.7}$$

where, $\mathbf{p}_f^K$ and $\mathbf{p}_d^K$ is the achieved false alarm and detection probability vector after $K$ sensors are deployed in the grid, respectively. $\mathbf{D}$ is an $N_xN_y \times 1$ deployment vector, each element of which indicates the number of sensors at a corresponding grid point (i.e., $D(j) = 1$ if a sensor is deployed at the $j$-th point and 0 otherwise). The squared error cost is taken at the points where the requirements are not met, this will ensure that we don't penalize satisfying the requirements.

## 7.2 Optimal Control Formulation

One of the difficulties associated with using the optimal decision fusion rule, is the computational complexity involved in evaluating the false alarm and detection probabilities for a given $K$. The distribution of the decision statistic $Z$ has no closed form expression for its tail probability (i.e., false alarm/detection probabilities) [65],[66], [67] and [68]. We propose approximating the decision statistics using a binomial distribution, we then use a Gaussian

approximation to approximate the tail probability of the decision statics $Z$. Using this approximation, we model the change in the overall detection performance of the network as a linear function of the sensor positions. This, in combination with a suitable squared error cost function results in a linear quadratic regulator (LQR) interpretation of the sensor deployment problem, in which the control vector corresponds to the sensors positions. One can use optimality conditions to calculate the optimal control vector (i.e., sensor positions). In the next subsections, we will detail our effort in formulating the deployment problem as an LQR problem.

As mentioned earlier, the distribution of the decision statistic has no closed form expression. The distribution of $Z$ is a weighted combination of sensor decisions. The non-zero weights $\{w(i), i = 1, \ldots, k$ (we drop the dependence on $j$ for reducing the clutter in notation) are different for different sensors (as given by Eqn.(7.3)). In order to evaluate the distribution of $Z$, one has to consider all weight combinations and the probabilities associated with each combination. As a result, it is computationally expensive to characterize the distribution of $Z$. This in turn makes the evaluation of the false alarm and detection probabilities after each sensor deployment prohibitively expensive. The high computational complexity necessitates the use of approximations or bounds to estimate the tail probability of $Z$ (i.e., false alarm/detection probability). A class of approximation methods, asymptotically approximates the tail probability of the $Z$ distribution when a large number of sensors are involved in the decision process [69], [70]. This is of little practical relevance since a real network consists of a limited number of sensors. Another approach for approximating the tail probability is based on using a saddle point approximation as in [71]. Even though the approximation in [71] does not assume the presence of a large number of sensors, it still requires the evaluation of a number of non-linear equations which adds significant computational complexity. Another class of bounds relies on using results, related to large deviation theory (LDT), such as that of Hoeffding [65] and Talagrand [66] . However, these bounds are applicable as long as the decision threshold is larger than the mean of the de-

cision statistic distribution. The farther the threshold is from the mean, the more accurate is the approximation. This can be problematic in our case. For instance, while the decision threshold under hypothesis $H_0$ might be far away from the mean (since false alarm probability requirement is usually small), there is no guarantee the decision threshold under $H_1$ hypothesis will be larger than the mean.

It is desirable to have a tail probability approximation that (1) does not assume asymptotic conditions on the number of sensors; (2) can be easily evaluated, and (3) does not depend on the value of the decision threshold. We propose using a binomial approximation followed by a Gaussian approximation of the the tail probability that satisfies conditions (1)-(3). Our motivation for using the Binomial approximation for Z(j) stems from the following observations:

1. We note that in the special case where the weights $w(i)$ are identical and the success probabilities for all Bernoulli random variables $u_i$ are the same (i.e., $p_d(j,i) = p_d$), the distribution of $Z$ under each hypothesis is that of a Binomial distribution.

2. We also note that when the weights are identical, but the success probabilities of the Bernoulli random variables are not identical, the resulting distribution of $Z$ is the Poisson trial distributions [60]. The tail of the Poisson trial distribution has no closed form expression [67], [68]. However, Hoeffding in 1956 [60] has proposed bounding the tail of the Poisson trial distribution using a Binomial distribution.

Therefore, the use of a Binomial to approximate the $Z$ distribution for non- uniform weights and success probabilities is a natural progression of the use of the Binomial approximation for the above mentioned special cases.

Let $Z$ denote the scaled decision statistic

$$Z' = \sum w'(i)u_i \tag{7.8}$$

where, $w'(i) = \frac{w(i)}{\max\{w(i)\}}$ is the set of non-zero weights. A binomial random variable $X \sim Bino(N_x, p_x)$ can be fully described in terms of its number of trials $N_x$ and its success

probability $p_x$. Exploiting the structure of $Z'$, we wish to determine values of $N_x$ and $p_x$ that approximate the distribution of $Z'$ using a binomial distribution. We note that in the most general case, when the weights $\{w(i),\ i = 1, \ldots, k\}$ are all distinct, we have $2^k - 1$ distinct non-zero weight combinations. We claim that an $N_x$ value of $2^k - 1$ accurately models the number of values encountered in the distribution of $Z'$. Based on this choice of $N_x$, we can now calculate the success probability $p_x$.

Under hypothesis $H_0$ , we are interested in using the proposed approximation to calculate the decision threshold that results in meeting the false alarm requirement. Since false alarm requirements are usually small (say less than 0.5), we are interested in the CCDF of the binomial distribution $X$ being an upper bound for the CCDF of $Z'$ in the range of $[E[Z'] = \sum w'(i)p_f, \max Z' = \sum w'(i)]$, where $E[Z']$ denotes the expected value of $Z'$.

We note that after choosing the number of trials of the approximation to be $2^k - 1$, our task is then to calculate the success probability $p_{x,0}$ of $X$ under hypothesis $H_0$. There are two computationally efficient methods for calculating this quantity:

1. Mandate that $E[Z'] = E[X] = \sum w'(i)p_f$ , then
$$p_{x,0} = \frac{\sum w'(i)p_f}{2^k - 1} \tag{7.9}$$

2. The max value of $Z'$ occurs when all $k$ sensors report a positive decision. That is, $\max Z' = \sum w'(i)$. The tail probability of $Z'$ corresponding to this case can be easily calculated as $\Pr(Z' \geq \max Z') = p_f^k$. The new success probability is then equal to $p_{x,0} = (p_f)^{\frac{k}{(2^k-1)}}$.

So, in order to have an upper bound on the CCDF of Z', under $H_0$, in the range $[E[Z'] = \sum w'(i)p_f, \max Z' = \sum w'(i)]$ using a binomial $X_0$, the success probability $p_{x,0}$ should be chosen as
$$p_{x,0} = \max\{\ \frac{\sum w(i)p_f}{(2^k - 1)},\ p_f^{\frac{k}{(2^k-1)}}\}. \tag{7.10}$$
We note that $0 \leq p_{x,0} \leq 1$ (as $Z$ has been scaled by the maximum weight). One can take an additional step further and approximate the tail of the binomial approximation using a

Gaussian density as follows

$$Pr(X_0 \geq t) \approx Q(\frac{t-(2^k-1)p_{x,0}}{\sqrt{(2^k-1)p_{x,0}q_{x,0}}}), \tag{7.11}$$

where, $Q(\cdot)$ is the complementary cumulative distribution (CCDF) of a standard normal distribution and $q_{x,0} = 1-p_{x,0}$. Additionally, the process of choosing $p_{x,0}$ based on Eqn.(7.10) can be simplified by adding a correction factor in the Gaussian approximation of $X_0$. That is, if we pick $p_{x,0} = \frac{\sum w'(i)p_f}{2^k-1}$ and use a correction factor of $+0.5$ in Eqn.(7.11), the overall tail probability will automatically increase making the resulting tail probability approximation as an upper bound on the actual false alarm probability as:

$$Pr(X_0 \geq t) \approx Q(\frac{t-(2^k-1)p_{x,0+0.5}}{\sqrt{(2^k-1)p_{x,0}q_{x,0}}}), \tag{7.12}$$

We can extend this argument to detection probability, under hypothesis $H_1$, bound by using $p_{x,1} = \frac{\sum w'(i)p_d(i)}{2^k-1}$ in conjunction with a correction factor of $-0.5$ as:

$$Pr(X_1 \geq t) \approx Q(\frac{t-(2^k-1)p_{x,1-0.5}}{\sqrt{(2^k-1)p_{x,1}q_{x,1}}}), \tag{7.13}$$

As an example, we approximate the CCDF of the distribution of the $Z$ distribution under both hypotheses. We assume that the number of sensors involved is $k = 7$; the sensor's false alarm probability is set to $p_f = 0.2$, and the detection probabilities are assumed to be $p_d(i) = 0.8 - 0.05 * i, i = 1, 2, \ldots, 7$. Fig.7.2 shows the CCDF (i.e., tail probability) of the distribution of $Z$ as well as the CCDF of the approximation in Eqn.7.11. We note that under hypothesis $H_0$ the CCDF of the approximation is an upper bound of the false alarm probability (i.e., CCDF of $Z$). We also note that under hypothesis $H_1$, the CCDF of the binomial-Gaussian approximation is a lower bound of the CCDF of $Z$ (i.e., detection probability). This will be helpful since we will be able to compute, using the approximation, a decision threshold such that the resulting false alarm rate is lower than the false alarm requirement. The detection probability calculated using the approximation, will also be lower than the actual detection probability.

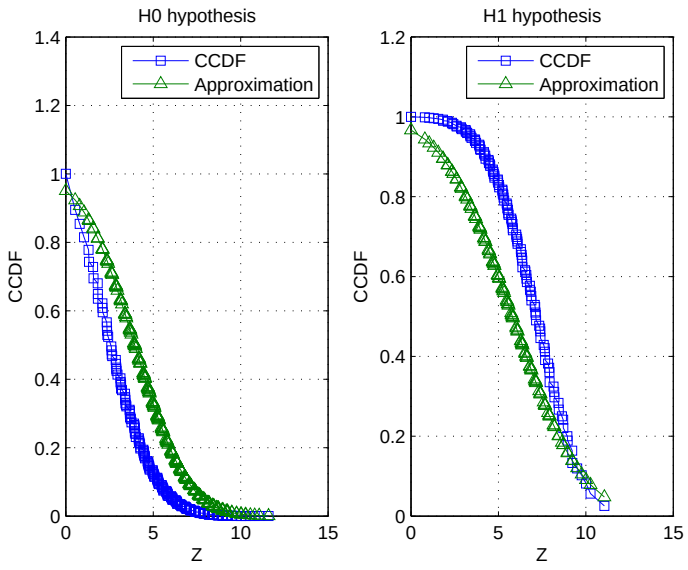



**Figure 7.1**: *Actual CCDF vs Approximation*

### 7.2.1 LQR Problem Formulation

In this section, we go through the steps that enable us to formulate the deployment problem as a linear quadratic regulator (LQR) problem. Using the Gaussian approximation of the detection probability, we approximate the effect of a single sensor deployment on the network's overall detection probability.

Towards this end, we first note that it is possible to approximate the $Q(\cdot)$ function as follows [59]

$$Q(y) \approx 1 - \frac{1}{1 + e^{-\sqrt{2}y}}. \tag{7.14}$$

The approximation in Eqn.(8.10) is a widely used approximation of the $Q$ function [18] and [72]. Below is a graph that shows both the $Q(y)$ function and its approximation. It is evident that the approximation of $Q(y)$ is valid over a wide range of values. We will use this approximation in evaluating the decision threshold and in estimating the detection probability at each grid point.

In a detection system that employs a Neyman-Pearson rule, the decision threshold is chosen such that the false alarm requirement is satisfied and then the system's detection

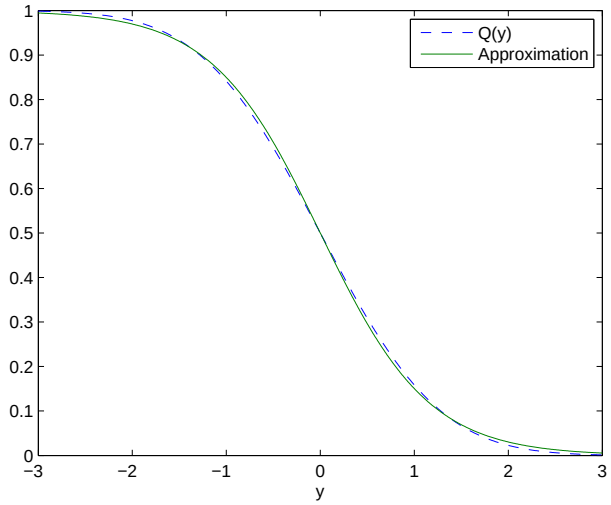



**Figure 7.2**: *Q Approximation*

probability is maximized assuming that decision threshold. In this paper, we adopt a similar approach.

Under hypothesis $H_0$, the scaled decision statistics $\eta'(j,k(j))$ that would enable us to satisfy the false alarm requirement $p_f^{req}(j)$ at the $j$-th point according to our proposed approximation Eqns.(7.11) and (8.10) is given as

$$\begin{aligned}\eta'(j,k(j)) &= (2^{k(j)}-1)p_{x,0}(j,k(j)) \\ &- \sqrt{\frac{(2^{k(j)}-1)p_{x,0}(j,k(j))q_{x,0}(j,k(j))}{2}}\ln(\frac{1}{1-p_f^{req}(j)}-1)\end{aligned} \tag{7.15}$$

Under hypothesis $H_1$, using the approximation in Eqn.(7.14) one can express $\ln(\frac{1}{1-p_d(j,k(j))}-1)$ (which has a one to one relationship with the detection probability $p_d(j,k(j))$) as follows

$$\ln(\frac{1}{1-p_d(j,k(j))}-1) = -\sqrt{2}(\frac{\eta'(j,k(j))-(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)}-1)p_{x,1}(j,k(j))q_{x,1}(j,k(j))}}). \tag{7.16}$$

If an additional sensor is to be deployed (i.e., $k(j)\rightarrow k(j)+1$), one can use Eqn.(7.15) to calculate the new decision threshold $\eta'(j,k(j)+1)$ and Eqn.(7.16) to get that

$$\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) = -\sqrt{2}(\frac{\eta'(j,k(j)+1)-(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}). \quad (7.17)$$

Eqn.(7.17) can be rewritten as

$$\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) = -\sqrt{2}(\frac{\eta'(j,k(j))+\Delta\eta'-(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)+(2^{k(j)}-1)p_{x,1}(j,k(j))-(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}), \quad (7.18)$$

where, $\Delta\eta' = \eta'(j,k(j)+1) - \eta'(j,k(j))$ is the difference in the decision threshold. In addition, Eqn.(7.18) can be restated as

$$\begin{aligned}\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) &= -\sqrt{2}(\frac{\eta'(j,k(j))-(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}) \\ &- \sqrt{2}(\frac{\Delta\eta'-(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)+(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}) \quad (7.19)\end{aligned}$$

multiplying the first term of the right hand side of Eqn.(7.19) by $\sqrt{\frac{(2^{k(j)}-1)p_{x,1}(j,k(j))q_{x,1}(j,k(j))}{(2^{k(j)}-1)p_{x,1}(j,k(j))q_{x,1}(j,k(j))}} = 1$, we can state Eqn.(7.19) as

$$\begin{aligned}\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) &= -\frac{\sqrt{2}}{\alpha(j)}(\frac{\eta'(j,k(j))-(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)}-1)p_{x,1}(j,k(j))q_{x,1}(j,k(j))}}) \\ &- \sqrt{2}(\frac{\Delta\eta'-(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)+(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}) \quad (7.20)\end{aligned}$$

where,

$$\alpha(j) = \sqrt{\frac{2^{k(j)+1}p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}{2^{k(j)}p_{x,1}(j,k(j))q_{x,1}(j,k(j))}}.$$

Finally, since $\ln(\frac{1}{1-p_d(j,k(j))}-1) = -\sqrt{2}(\frac{\eta'(j,k(j))-(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)}-1)p_{x,1}(j,k(j))q_{x,1}(j,k(j))}})$, Eqn.(7.20) can be rewritten as

$$\begin{aligned}\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) &= \frac{1}{\alpha(j)}\ln(\frac{1}{1-p_d(j,k(j))}-1) \\ &- \sqrt{2}(\frac{\Delta\eta'-(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)+(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}) \quad (7.21)\end{aligned}$$

Let $p_d^{req}(j)$ be the required detection probability at the $j$-th point, then one can define $x(j,k(j))$ as

$$x(j,k(j)) = \ln(\frac{1}{1-p_d(j,k(j))} - 1) - \ln(\frac{1}{1-p_d^{req}(j)} - 1). \tag{7.22}$$

Since there is a one to one relationship between $y$ and $\ln(\frac{1}{1-y} - 1)$, $x(j,k(j))$ effectively measures the difference between the achieved detection probability $p_d(j,k(j))$ and the required detection probability $p_d^{req}(j)$. In terms of $x(j,k(j))$, it is possible to write the change (evolution) in $x(j,k(j))$ when a sensor is added to the grid as

$$x(j,k(j)+1) = \frac{1}{\alpha(j)} x(j,k(j)) + B(j,i), \tag{7.23}$$

where, $B(j,i)$ is given as

$$\begin{aligned} B(j,i) &= (1 - \frac{1}{\alpha(j)}) \ln(\frac{1}{1-p_d^{req}(j)} - 1) \\ &- \sqrt{2}(\frac{\Delta\eta' - (2^{k(j)+1}-1)p_{x,1}(j,k(j)) + (2^{k(j)}-1)p_d(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}). \end{aligned} \tag{7.24}$$

$B(j,i)$ in essence quantifies the change in $x(j,k(j))$ resulting from deploying a sensor at the $i$-th grid point. Eqn.(7.24) dependency on $i$ is manifested by the terms $p_{x,1}(j,k(j)+1)$ and $q_{x,1}(j,k(j)+1)$. The term $p_{x,1}(j,k(j)+1)$ is the success probability calculated by incorporating the contributions of the sensors that are already deployed with an $R$ radius of the $j$-th point and a sensor that is possibly to be deployed at the $i$-th grid point.

Considering all points on the grid, it is possible to write Eqn.(7.23) in matrix form as

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k \tag{7.25}$$

where, the matrix subscript $k$ denotes the total number of deployed sensors. The matrix $\mathbf{A}_k$ is a diagonal matrix of dimension $N_x N_y$, where the $j$-th diagonal entry is $A(j,j) = \frac{1}{\alpha(j)}$. The matrix $\mathbf{B}_k$ is a square matrix of dimension $N_x N_y$, with its $(j,i)$ entry is equal to $B(j,i)$ in Eqn.(7.24). Each element in the $N_x N_y \times 1$ $k$-th deployment vector $\mathbf{u}_k$ corresponds to one of the grid points. The $j$-th element in $\mathbf{u}_k$, in essence, indicates the contribution of placing a sensor at the $j$-th grid point on the overall detection probability.

In an optimal control problem, the goal is to calculate an optimal control vector that minimizes a certain cost function. In our original problem statement in Eqn.(7.7), the cost function is a squared error cost function. Exploiting the one to one relationship between $p_d(j, k(j))$ and $\ln(\frac{1}{1-p_d(j,k(j))} - 1)$, we define an updated cost function, associated with the system evolution model in Eqn.(7.25)

$$J = \frac{1}{2}\mathbf{x}_K^T \mathbf{Q}_f \mathbf{x}_K + \frac{1}{2}\sum_{k=1}^{K-1}(\mathbf{x}_k^T \mathbf{Q}_k \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R}_k \mathbf{u}_k), \tag{7.26}$$

where, $\mathbf{Q}_k$, $\mathbf{Q}_f$ and $\mathbf{R}_k$ are symmetric positive definite weighting matrices of dimension $N_x N_y \times N_x N_y$. In the cost function $J$, the matrices $\mathbf{Q}_k$ and $\mathbf{Q}_f$ are directly related to the $\mathbf{x}_k$ and $\mathbf{x}_K$ vectors. In fact, $\mathbf{x}_k^T \mathbf{Q}_k \mathbf{x}_k$ is a weighted norm of the state $\mathbf{x}_k$. Since the system is dynamic and the detection requirements at some of the grid points might be satisfied while others might not be satisfied, it is important to not penalize satisfying the requirements. In the cost function $J$, this can be achieved by setting the elements in $\mathbf{Q}_k$ corresponding to points where requirements have been met to small values. In contrast, to penalize not meeting detection requirements at the remaining grid points, we set the elements in $\mathbf{Q}_k$ that correspond to these points to relatively larger values. This scheme ensures that when making the deployment decision the areas where requirements are not met are given more importance in making that decision. We can also see that the matrix $\mathbf{R}_k$ is connected directly to the control vector $\mathbf{u}_k$. Choosing a large value for an entry of $\mathbf{R}_k$ indicates that it is costly to deploy a sensor at that point. This can be useful in steering the deployment away from areas where requirements have been met or points where sensors have been already deployed. In contrast, assigning relatively small values of entries of $\mathbf{R}_k$ corresponding to points where requirements are not met or where sensors are not deployed indicates a reasonable bias for deploying sensors at those points.

The optimal control problem corresponding to our system is the linear quadratic regulator (LQR) problem which we state as follows

$$\underset{\mathbf{u}_k}{\text{argmin}}\, J = \frac{1}{2}\mathbf{x}_K^T\mathbf{Q}_f\mathbf{x}_K + \frac{1}{2}\sum_{k=1}^{K-1}(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}_k\mathbf{u}_k)$$
$$\text{subject to} \begin{cases} \mathbf{p}_f^K \leq \mathbf{p}_f^{req} \\ \mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k,\ k = 0, 1, \ldots, K-1 \\ \mathbf{1}^T\mathbf{u} = K \end{cases} \tag{7.27}$$

The LQR problem described above is dynamic and therefore can be solved using the differentiation method discussed earlier.

## 7.3 Deployment Algorithm

Based on the LQR formulation and its solution discussed above, we propose a sequential LQR based sensor deployment algorithm. Given a number of sensors $K$, false alarm/detection requirements ($\mathbf{p}_f^{req}/\mathbf{p}_d^{req}$), the algorithm sequentially deploys sensors until either all $K$ sensors have been deployed or when the requirements are satisfied. The algorithm deploys the $k$-th sensor by first constructing the matrices $\mathbf{A}_k$ and $\mathbf{B}_k$. Afterward, the $k$-th deployment vector $\mathbf{u}_k$ is calculated as in Eqn.(3.25). Giving more importance to satisfying false alarm requirements, in each deployment stage we determine the points at which false alarm requirements are not met, and only consider these points for sensor deployment. The entry values of $\mathbf{u}_k$ are compared for these points and a sensor is placed at the index $\hat{i}$ where $\mathbf{u}_k$ is maximum (i.e., $\mathbf{D}(\hat{i}) = 1$). In case all false alarm requirements are met, the algorithm simply places a sensor at the grid point corresponding to the index where $\mathbf{u}_k$ is maximum. The proposed deployment algorithm is described in Algorithm 6.

**Algorithm 6** LQR Based Algorithm

**Initialization** $k = 0$, $\mathbf{D} = \mathbf{0}$, $\mathbf{p}_d^0 = \mathbf{0}$, $\mathbf{p}_f^0 = \mathbf{1}$
**while** $(\mathbf{p}_f^k \succeq \mathbf{p}_f^{req}$ and $\mathbf{p}_d^k \preceq \mathbf{p}_d^{req})$ OR $(k \leq K)$ **do**
  Calculate $\mathbf{u}_k$ as in Eqn.(3.25)
  Find the index set $I_\Delta = \{i_j, j = 1, 2, \ldots, n\}$, where $\mathbf{p}_f^k(I_\Delta) \geq \mathbf{p}_f^{req}$
  **if** $I_\Delta \neq \Phi$ ($\Phi$ indicates null set)
  Find $\hat{i} \in I_\Delta$ where $u_k(\hat{i}) \geq u_k(i) \forall i \in I_\Delta$
  **else**
  Find $\hat{i}$ where $u_k(\hat{i}) \geq u_k(i) \forall i = 1, 2, \ldots, N_x N_y$
  **end**
  Deploy sensor at $\hat{i}$ (i.e., $D(\hat{i}) = 1$)
  Update decision thresholds
  Calculate $\mathbf{p}_f^k$ and $\mathbf{p}_d^k$
  Increment $k$ to $k + 1$
**end while**

## 7.4 Simulation Results

In this section, we compare the performance of the LQR based deployment algorithm to that of a greedy algorithm. The greedy algorithm sequentially deploys sensors by calculating the difference between achieved and required detection probabilities at all the grid points and then deploying a sensor at the grid point with the largest difference. We use two metrics to compare the performance of the LQR and greedy deployment algorithms. The first metric is the number of sensors needed by each algorithm to meet performance requirements. Though our original problem statement assumes that the total number of sensors is already determined before deployment, the LQR sequential formulation ( with the single-step horizon model) does not require the previous knowledge of the number of sensors. Therefore, we can deploy sensors until performance requirements are met. The same holds for the greedy deployment algorithm. The second metric, is the effective squared error (SE) which measures the squared difference between achieved and required detection probabilities at grid points where requirements are not satisfied.

In the first experiment, the false alarm/detection requirements are uniform over the grid. We compare the performance of the LQR and greedy deployment algorithms as the

detection requirement is varied while the false alarm requirement is kept constant. We assume that the (1) grid is of size $20 \times 20$, (2) sensor false alarm rate of $p_f = 0.05$, (3) sensing profile decay rate of $\tau = 0.15$ and (4) the detection radius $R = 5$. The false alarm requirement is set to $p_f^{req} = 0.01$. Fig. 7.3 lists the number of sensors needed to satisfy performance requirements. We note that as the detection requirements become more stringent, the number of sensors needed by each algorithm increases as well. This is expected as for a fixed $\tau$, sensors need to be closer to each other (i.e., more compactly distributed) in order to meet detection requirements. Therefore, more sensors are needed when detection requirements become more demanding. Simulation results in Fig. 7.3, illustrate that the LQR based deployment algorithm requires up to 20% fewer number of sensors than does greedy algorithm to satisfy the same false alarm/detection requirements. The deployment decision in the greedy algorithm aims at minimizing the detection error at a single point (i.e., localized cost function). Whereas in the LQR deployment algorithm, the cost function in the LQR problem takes into account the errors at all the grid point where requirements are not met. Moreover, in the LQR algorithm the effect of a sensor's deployment on all the points within its detection radius is taken into consideration by constructing the $\mathbf{B}_k$. This is in contrast to the greedy algorithm which does not incorporate this information. To compare the effective SE of both algorithms, we consider case 1 (i.e., $p_d^{req} = 0.6$) in Fig. 7.3. The effective SE as a function of the number of sensors deployed in the grid is depicted in Fig.7.4. Even though both algorithms require the same number of sensors in this case, it is evident that the LQR deployment algorithm is more efficient in satisfying detection requirements. This is true since the LQR algorithm results in a lower SE than the greedy algorithm for a given number of sensors.

In the second experiment, we study the performance of the algorithms as the false alarm requirement is varied. The parameters in this experiment are as follows; (1) a $20 \times 20$ grid, (2) sensor false alarm rate $p_f = 0.1$, (3) a sensing decay rate $\tau = 0.05$ and (4) the detection radius is set to $R = 7$. The detection requirement at the grid points is set to $p_d^{req} = 0.8$.

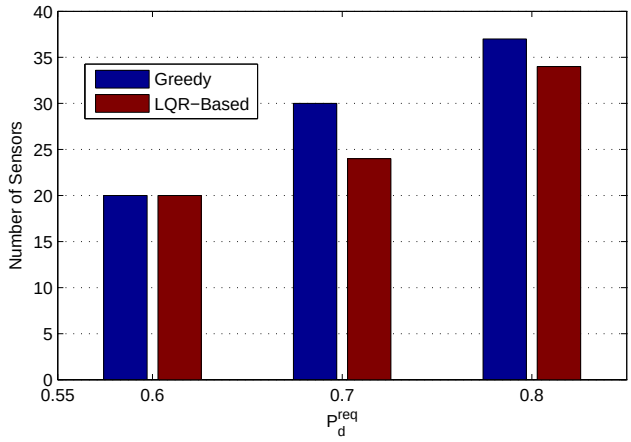



**Figure 7.3**: *Number of sensors for different $p_d^{req}$: Greedy vs. LQR-based*

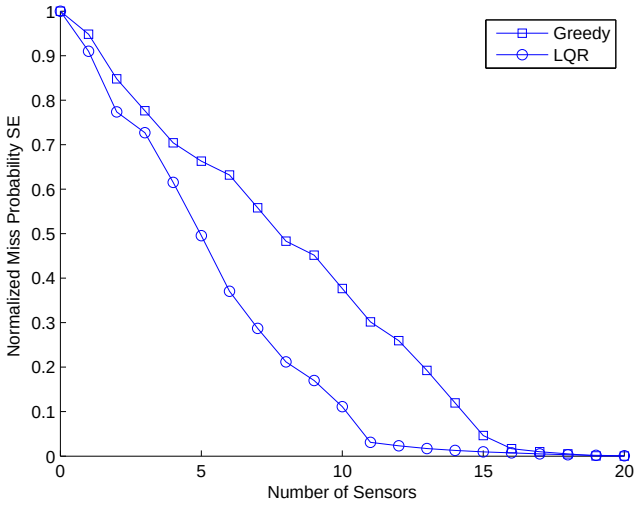



**Figure 7.4**: *SE convergence for uniform requirements*

The number of sensors needed to satisfy performance requirements are listed in Fig. 7.5. The trend is similar to what is expected and what is seen in the first experiment. The more stringent a requirement is ( in this case, a lower false alarm probability), the more sensors are needed. This is because a smaller false alarm requirement results in a larger decision threshold. However, under hypothesis $H_1$, we desire a larger tail probability in order to meet the detection requirement. The larger decision threshold mandates the use of more sensors covering a grid point. From Fig. 7.5, we see that we can save up to 30%

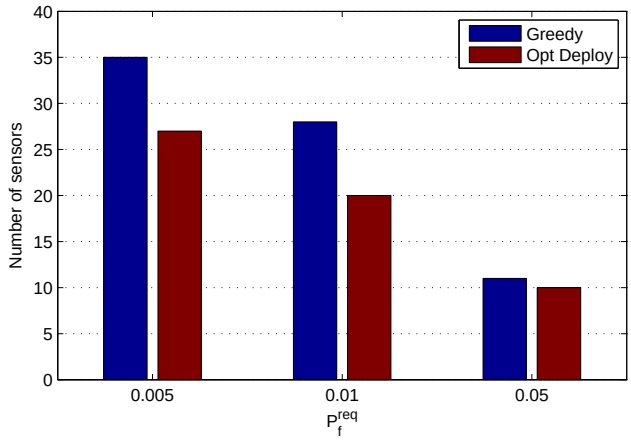



**Figure 7.5**: *Number of sensors for different $p_f^{req}$: Greedy vs. LQR-based*

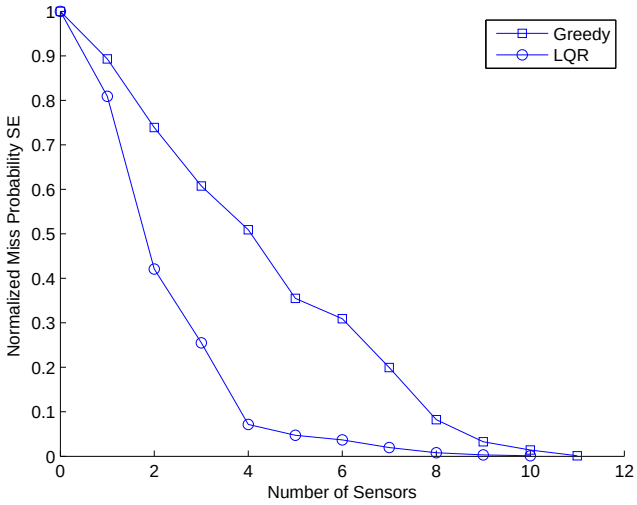



**Figure 7.6**: *SE convergence for uniform requirements*

in the number of required sensors by using the LQR deployment algorithm instead of the greedy algorithm. Next, we examine the effective SE corresponding to case 1 of Fig. 7.5. In Fig.7.6, we note that even though both algorithms require comparable number of sensors for this case, the LQR deployment algorithm consistently outperforms the greedy algorithm in terms of effective SE. This further illustrates the advantage of using the LQR algorithm when provided with a limited number of sensors.

In the third experiment, we consider a grid of non-uniform false alarm/detection require-

**Table 7.1**: *Number of sensors for non-uniform performance requirements*

| Requirements | Greedy | LQR |
|---|---|---|
| $p_d^1 = 0.75, p_f^1 = 0.01, p_d^2 = 0.6, p_f^2 = 0.01$ | 37 | 32 |
| $p_d^1 = 0.9, p_f^1 = 0.05, p_d^2 = 0.6, p_f^2 = 0.01$ | 37 | 30 |
| $p_d^1 = 0.9, p_f^1 = 0.05, p_d^2 = 0.8, p_f^2 = 0.01$ | 59 | 45 |

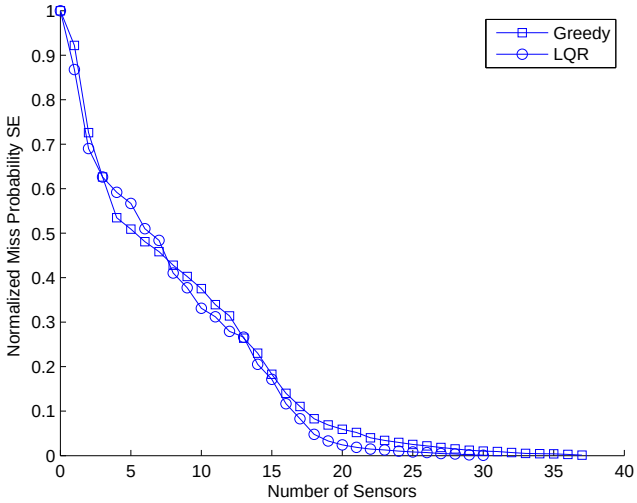



**Figure 7.7**: *SE convergence for non-uniform requirements*

ments. The experiment setup is as follows; (1) a $20 \times 20$ grid, (2) $p_f = 0.1$, (3) $\tau = 0.1$ and (4) $R = 5$. The grid is divided into two areas with different false alarm/ detection requirements as in Fig.7.8 . Simulation results listed in Table 7.1 indicate that with increasing requirements, the number of sensors needed to satisfy these requirements grows. The justification is similar to what was discussed in the previous two experiments. Furthermore, we note that even when the requirements are not uniform, the LQR algorithm consistently outperforms the greedy deployment algorithm by using a fewer number of sensors to meet the same requirements. In fact, results in Table 7.1 show that a 23% reduction in the number of sensors is possible by using the LQR algorithm instead of the greedy algorithm. The effective SE for case 2 of Table 7.1 is plotted in Fig.7.7 and it is evident that the LQR algorithm is more efficient than the greedy algorithm in satisfying the detection requirements. We also show the sensor positions as they are deployed by each algorithm in Fig.7.8.

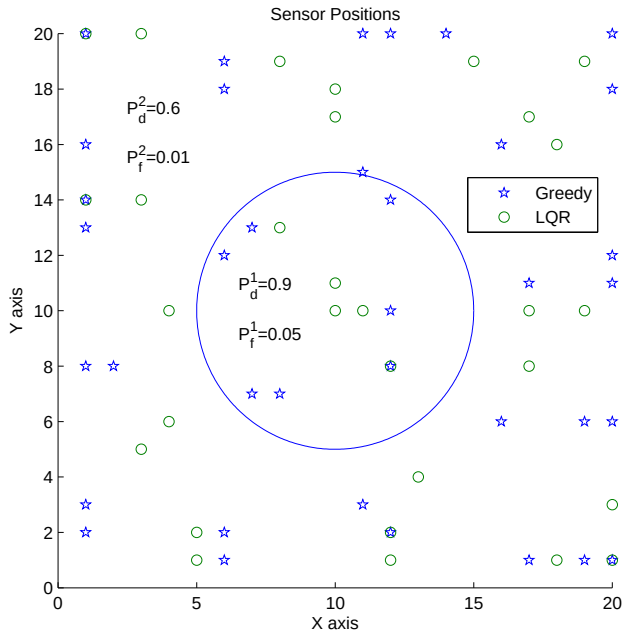



**Figure 7.8**: *Sensor positions*

**Table 7.2**: *Number of sensors for square non-uniform performance requirements*

| Requirements | Greedy | LQR |
|---|---|---|
| $p_d^1 = 0.6, p_f^1 = 0.05, p_d^2 = 0.9, p_f^2 = 0.01, p_d^3 = 0.6, p_f^3 = 0.05$ | 31 | 28 |
| $p_d^1 = 0.9, P_f^1 = 0.01, p_d^2 = 0.7, p_f^2 = 0.01, P_d^3 = 0.7, p_f^3 = 0.05$ | 53 | 44 |

In the fourth experiment, we consider a grid with nonuniform requirements as in Fig.7.9. The experiment parameters are as follows: (1) grid size is $20 \times 20$, (2) $p_f = 0.1$ (3) $\tau = 0.1$ and $R = 5$. Table 7.2 lists the number of sensors needed by each algorithm to satisfy the performance requirements. As before, it is evident that the LQR deployment algorithm uses a fewer number of sensors than the greedy algorithm to meet the same performance requirements. The spatial distribution of sensors when the requirements are as in case 2 of Table 7.2 is depicted in Fig. 7.9.

## 7.5 Summary

We examined the sensor deployment problem in a DSN employing the optimal decision fusion rule. The goal was to find the positions where a given number of sensors can be deployed

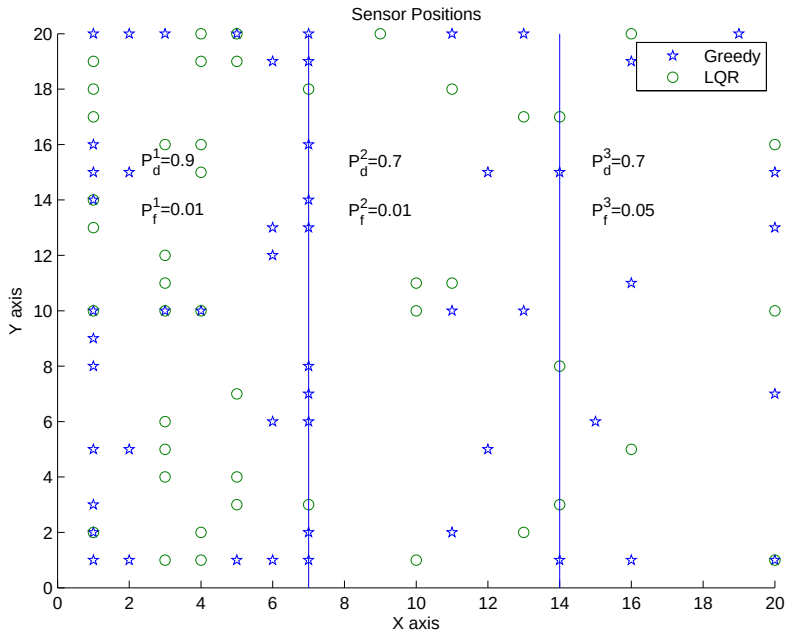



**Figure 7.9**: *Sensor positions for square non-uniform requirements*

in order to meet some given detection and false alarm rate requirements. In prior efforts, deployment in an optimal decision fusion network has not been addressed due to mathematical complexity. For the first time, we proposed a rigorous treatment of the deployment problem with optimal decision fusion by incorporating ideas from optimal control theory. We first proposed a novel approximation of the detection and false alarm rates and used this approximation to model the change in the detection performance of the network as a linear function of the deployment position of a single sensor. Adopting a suitable squared error cost function, we modeled the deployment problem as a linear quadratic regulator (LQR) problem. Using this model, we proposed a sequential sensor deployment algorithm with the goal of satisfying both false alarm and detection requirements. To illustrate the advantages of using the LQR based algorithm, we compared its performance against a greedy algorithm. Simulation results indicated that the proposed LQR algorithm can save up to 30% in the number of sensors needed by the greedy algorithm to satisfy the same performance requirements. Moreover, we illustrated that the LQR based algorithm is more efficient in

terms of the effective SE than the greedy algorithm.

In the next section, we study the self healing problem in sensor networks. We specifically show that it is possible to use our proposed LQR formulation of the deployment problem as a self healing approach.

# Chapter 8

# Self Healing in Sensor Networks – Optimal Decision Fusion

In this chapter, we study the self-healing problem in sensor networks employing the optimal decision fusion rule. We propose two self healing methods. The first method is based on adjusting the decision threshold at the fusion center. Since the decision statistics distribution has no closed form expression, updating the decision threshold is computationally intensive. Therefore, we propose using the closed form approximation used in the previous chapter to update the decision threshold. The second method uses sensor mobility for self healing. Sensors are redeployed such that the performance degradation from sensor loss is minimized. The redeployment of sensors is modeled as an LQR problem. Based on these two approaches, we propose a self healing algorithm for sensor network.

The system model is similar to the one used in the previous chapter and is reviewed in section 8.1. The proposed approaches are discussed in section 8.2. In section 8.3, we introduce our proposed self-healing algorithm. Simulation results illustrating the advantages of using our proposed algorithms are listed in section 8.4.

## 8.1 System Model

An $N_x \times N_y$ grid $\mathcal{G}$ is used to describe the area of interest where the sensors are deployed. The $j$-th ($j = 1, \ldots, N_x N_y$) grid point is associated with a given false alarm and detection

requirement, $p_f^{req}(j)$ and $p_d^{req}(j)$, respectively. Considering all grid points, these requirements can be arranged in two $N_xN_y \times 1$ vectors $\mathbf{p}_f^{req}$ and $\mathbf{p}_d^{req}$. Without loss of generality, we assume that a sufficient number $N$ of sensors has been deployed in the area of interest such that the false alarm and detection requirements are met. The binary $N_xN_y \times 1$ vector $\mathbf{D}$ indicates the positions $\{(x_n, y_n), n = 1, \ldots, N\}$ of the sensors in the grid. An entry with a value of 1 indicates that a sensor is deployed at the grid point corresponding to that entry, while a 0 value indicates that no sensor is deployed at that point. The binary local decision $u_i$ made by the sensor located at the $i$-th grid point indicates the absence/ existence of the target (hypothesis $H_0$ and $H_1$ respectively). We assume that the sensors share a common false alarm rate $p_f$. On the other hand, we assume that the sensors have a distance dependent detection profile. If $d(i, j)$ denotes the distance between the sensor, located at the $j$-th grid point, and the $i$-th grid point of interest, then the probability that the sensor detects a target at that point is given as

$$p_d(i,j) = \begin{cases} e^{-\tau d(i,j)} & \text{if } d(i,j) \leq R \\ 0 & \text{if } d(i,j) > R \end{cases} \tag{8.1}$$

where, $\tau$ is the sensor's detection decay rate and $R$ is a detection radius. We assume that the detection radius $R$ is such that the detection probability at a point further from $R$ is negligible.

Assuming error free communication, sensors send their decisions to the fusion center (FC) which combines their individual decisions and makes an overall detection decision $u_o(j)$ regarding the existence/absence of a target at the $j$-th grid point. In this work, we assume that the FC employs the optimal decision fusion strategy in making the decision $u_o(j)$ [44],[64]. The decision statistic $Z(j)$ corresponding to the $j$-th point is given as

$$Z(j) = \sum w(j,i)u_i, \tag{8.2}$$

where $k(j)$ is the number of sensors involved in the decision process at the $j$-th point, and

the weight $w(j,i)$ is given as

$$w(j,i) = \begin{cases} \log \frac{p_d(j,i)(1-p_f)}{p_f(1-p_d(j,i))} & \text{if } d(i,j) \leq R \\ 0 & \text{if } d(i,j) > R \end{cases} \tag{8.3}$$

The weight $w(j,i)$ quantifies the relative confidence in the accuracy of the decision from the sensor located at the $i$-th grid point in comparison to the decisions coming from the other sensors. Since $p_d(j,i)$ is distance dependent, it follows that the weight is also distance dependent. The closer a sensor is to the point of interest, the more importance is given to its decision in making the overall decision.

The final decision $u_0(j)$ rule corresponds to

$$u_0(j) = \begin{cases} 1 \text{ (i.e., } H_1 \text{ is true)} & \text{if } Z(j) \geq \eta(j,k(j)) \\ 0 \text{ (i.e., } H_0 \text{ is true)} & \text{if } Z(j) < \eta(j,k(j)) \end{cases} \tag{8.4}$$

where, $\eta(j,k(j))$ is the decision threshold. The false alarm probability $p_f(j,k(j))$ and detection probability $p_d(j,k(j))$ at the $j$-th point are given as

$$p_f(j,k(j)) = Pr(Z(j) \geq \eta(j,k(j)) \mid H_0 \text{ is true}) \tag{8.5}$$

$$p_d(j,k(j)) = Pr(Z(j) \geq \eta(j,k(j)) \mid H_1 \text{ is true}). \tag{8.6}$$

Note that in Eqns.(8.5) and (8.6), the false alarm and detection probabilities depend on the distribution of the decision statistic $Z(j)$ and the decision threshold $\eta(j,k(j))$. Furthermore, $Z(j)$ depends on the number of sensors $k(j)$ and their positions in the grid. Losing one or more sensors changes the distribution statistic $Z(j)$ and can lead to undesirable change in the detection performance at many grid points. In light of Eqns.(8.5) and (8.6), one can minimize performance degradation by: (1) changing the decision threshold $\eta(j,k(j))$, or (2) changing the decision statistic distribution $Z(j)$. Changing the decision threshold can be achieved by recalculating the distribution of $Z(j)$ and finding a suitable threshold $\eta(j,k(j))$ that ensures satisfying the performance requirements. The second option to consider is changing the distribution of $Z(j)$. The distribution of $Z(j)$ can be changed by modifying

the weights $w(j,i)$ in Eqn.(8.2) or the number of sensors $k(j)$. Either approaches can be done by moving sensors in the grid. Moving the $i$-th sensor closer to the $j$-th point that it already covers changes its weight $w(j,i)$. In addition, a sensor can move to cover the $j$-th point that it previously did not, effectively changing $k(j)$. In this chapter, we develop self healing strategies that are based on decision threshold adjustment and sensor mobility.

## 8.2 Proposed Approaches

In this section, we discuss two self-healing approaches; (1) decision threshold adjustment and (2) sensor redeployment.

### 8.2.1 Decision Threshold Adjustment

In decision threshold adjustment, the FC recalculates the decision threshold it employs for every point affected by loss of sensors. The main difficulty in this approach is the high computational complexity associated with evaluating the decision statistic distribution. We propose using an approximation of the decision statistic distribution, which allows us to evaluate the decision threshold with a relatively low computational complexity. In order to recalculate the decision threshold, it is necessary to evaluate the distribution of the decision statistic $Z$ (we drop the dependence on $j$ for ease of notation). However, this is a computationally intensive process since the distribution of $Z$ is non-parametric. Evaluating $Z$ requires calculating all weight combinations as well as the probability of each combination. As an alternative to exact evaluation of $Z$, we consider using a binomial-Gaussian approximation proposed in the previous chapter.

Consider the scaled $Z'$ distribution

$$Z' = \sum_{i=1}^{k} w'(i)u_i \tag{8.7}$$

where, $w'(i) = w(i)/\max w(i)$. The $Z'$ distribution is then approximated as a binomial $X \sim Bino(N_x, p_x)$ where $N_x = 2^k - 1$ is the number of trials and $p_x$ is the success probability

given as

$$p_x = \frac{E[Z']}{n_x}. \tag{8.8}$$

Moreover, we can approximate the CCDF of a binomial distribution using the CCDF of a Gaussian random variable as [58]

$$\Pr(Z' \geq \eta') \approx \Pr(X \geq \eta') \approx Q(\frac{\eta' - n_x p_x}{\sqrt{n_x p_x q_x}}) \tag{8.9}$$

where, $\eta' = \frac{\eta}{\max w(i)}$ is the scaled decision threshold and $q_x = 1 - p_x$. Finally, we can approximate the $Q$ function as follows [59]

$$Q(y) \approx 1 - \frac{1}{1 + e^{-\sqrt{2}y}}. \tag{8.10}$$

Under a Neyman-Pearson framework, the decision threshold is calculated such that the false alarm requirement is met (if possible) while maximizing the achieved detection probability. In this paper, we follow a similar framework. Using the approximations in Eqns.(8.9) -(8.10), and under hypothesis $H_0$ it is possible to calculate the scaled decision threshold $\eta'(j, k(j))$ for the $j$-th point when it is covered by $k(j)$ sensors as

$$\begin{aligned} \eta'(j,k(j)) &= (2^{k(j)} - 1)p_{x,0}(j,k(j)) \\ &- \sqrt{\frac{(2^{k(j)} - 1)p_{x,0}(j,k(j))q_{x,0}(j,k(j))}{2}} \ln(\frac{1}{1 - P_f^{req}(j)} - 1) \end{aligned} \tag{8.11}$$

where, $p_{x,0} = \frac{E[Z'|H_0]}{2^{k(j)}-1}$.

Note that the maximum value of $Z'$ is $\sum_{i=1}^{k} w'(i)$. Therefore, if $\eta' \leq \sum_{i=1}^{k} w'(i)$, then it is possible to meet the false alarm requirement while at the same time attaining a value of $p_d(j, k(j)) > 0$. On the other hand, if $\eta' > \sum_{i=1}^{k} w'(i)$ a false alarm probability of $p_f(j, k(j)) = 0$ is achieved (i.e., meet any false alarm requirement). However, with regards to the detection probability, choosing such a threshold implies that $p_d(j, k(j)) = 0$. We argue that this is not a wise choice, since it effectively implies that sensors covering the

point of interest are not contributing to the detection process. Therefore, after initially calculating $\eta'(j, k(j))$ as in Eqn.(8.11), the final decision threshold is given as

$$\eta'(j, k(j)) = \min(\eta'(j, k(j)), \sum_{i=1}^{k(j)} w'(i)). \tag{8.12}$$

Eqn.(8.12), in effect, represents a tradeoff between false alarm and detection performance. A priority is always given to satisfying the false alarm requirement, except in the case where detection performance is completely compromised. Using Eqns.(8.11) and (8.12), it is possible to adjust the decision thresholds in the network to minimize the degradation in the network's performance after the loss of one or more sensors.

Depending on the position of the failed sensors and the requirements within their area, the threshold readjustment might not offset the degradation in the system's performance. In the next subsection, we discuss sensor redeployment as an alternative self-healing approach.

### 8.2.2 Sensor Redeployment

In the sensor redeployment approach, we move mobile sensors to grid points such that the difference between the required and achieved performance profiles is minimized. Approximating the detection probability at a point as a function of the sensor positions, we are able to model sensor redeployment as a linear quadratic regulator (LQR) problem.

Note in Eqns.(8.5) and (8.6), that the false alarm and detection probabilities depend both on the decision threshold and the density of the decision statistic $Z$. We also note that the density of $Z$, which the FC evaluates independently for each grid point, depends on the number of sensors covering each point and the relative distances between sensors and points of interest. Assuming that the network consists of a number of static and mobile sensors, it is possible to change the density of $Z$ by moving sensors in the grid. Moving sensors closer to a point makes the tail of the corresponding $Z$ distribution heavier, for a fixed threshold. This increases the achieved detection probability and allows more flexibility in choosing a suitable decision threshold. Thus improving the detection performance. On the other hand,

moving sensors can also cause a degradation in performance at the original locations of the mobile sensors. Therefore, one should develop a redeployment strategy that considers not just the improvement in performance from moving sensors, but also the possible degradation caused by sensor movement.

We can state the main problem in the sensor redeployment approach as follows: Let $N_o$ denote the total initial number of sensors in the network, we assume that there are $S$ static sensors and $M$ mobile sensors. In the event of losing $l$ sensors, the question we seek to address is the following; where can the $M$ mobile sensors be placed (i.e., redeployed) such that the degradation in the detection performance as well as the total energy consumed in moving the sensors are minimized? That is,

$$\underset{\mathbf{D}}{\operatorname{argmin}} \sum_{j:p_d^n(j)<p_d^{req}(j)} (\mathbf{p}_d^n(j) - \mathbf{p}_d^{req}(j))^2 + \sum_{m=1}^{M} E_m$$
$$\text{subject to} \begin{cases} \mathbf{p}_f^n \leq \mathbf{p}_f^{req} \\ \mathbf{1}^T\mathbf{D} = n \end{cases} \tag{8.13}$$

where, $n = N_o - l$ and $\mathbf{p}_f^n$/ $\mathbf{p}_d^n$ denotes the achieved false alarm/ detection profile when $n$ sensors are deployed in the grid. $\{E_m, m = 1, \ldots, M\}$ denotes the energy that the $m$-th sensor consumes in relocating. The $N_xN_y \times 1$ deployment vector $\mathbf{D}$ indicates the sensor deployment positions. The squared error cost is taken at the points where the requirements are not met, this will ensure that we don't penalize satisfying the requirements.

Our approach towards sensor redeployment is based on modeling the redeployment problem as a linear quadratic regulator (LQR) problem. First, we model the change in the detection probability as a linear function of the position of the mobile sensor. We then use the difference between achieved and required detection performance as a cost function to be minimized. Thus, the redeployment problem becomes equivalent to an LQR problem. We can then solve for the new mobile sensor positions that minimize the cost function.

Let $k(j)$,$\eta'(j, k(j))$ and $p_d(j, k(j))$ denote the number of sensors, scaled decision threshold and detection probability at the $j$-th grid point after losing $l$ sensors. Using the approxima-

tion in Eqn.(8.10), we can approximate $\ln(\frac{1}{1-p_d(j,k(j))}-1)$ as follows

$$\ln(\frac{1}{1-p_d(j,k(j))}-1) = -\sqrt{2}(\frac{\eta'(j,k(j))-(2^{k(j)}-1)p_{x,1}(j,k(j))}{2^{k(j)}p_{x,1}(j,k(j))q_{x,1}(j,k(j))}), \tag{8.14}$$

note that $\ln(\frac{1}{1-p_d(j,k(j))}-1)$ has a one to one relationship with the detection probability $p_d(j,k(j))$.

If a sensor is to be moved such that it covers the $j$-th point (i.e., $k(j) \rightarrow k(j)+1$), one can use Eqn.(8.12) to calculate the new decision threshold $\eta'(j,k(j)+1)$ and Eqn.(8.14) to express the change from $\ln(\frac{1}{1-p_d(j,k(j))}-1)$ to $\ln(\frac{1}{1-p_d(j,k(j)+1)}-1)$ as follows

$$\begin{aligned}\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) &= \frac{-\sqrt{2}}{\alpha(j)}\ln(\frac{1}{1-p_d(j,k(j))}-1) \\ &- \sqrt{2}(\frac{\Delta\eta'-(2^{k(j)+1}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p^1_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}) \\ &- \sqrt{2}(\frac{(2^{k(j)}-1)p_d(j,k(j))-(2^{k(j)+1}-1)\Delta p_s(j)}{\sqrt{(2^{k(j)+1}-1)p^1_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}),\end{aligned} \tag{8.15}$$

$$\begin{aligned}\ln(\frac{1}{1-p_d(j,k(j)+1)}-1) &= -\frac{\sqrt{2}}{\alpha(j)}(\frac{\eta'(j,k(j))-(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)}-1)p_{x,1}(j,k(j))q_{x,1}(j,k(j))}}) \\ &- \sqrt{2}(\frac{\Delta\eta'-(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)+(2^{k(j)}-1)p_{x,1}(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}})\end{aligned} \tag{8.16}$$

where, $\Delta\eta' = \eta'(j,k(j)+1)-\eta'(j,k(j))$ is the difference in the decision threshold, and

$$\alpha(j) = \sqrt{\frac{2^{k(j)+1}p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}{2^{k(j)}p_{x,1}(j,k(j))q_{x,1}(j,k(j))}}.$$

Let $p_d^{req}(j)$ be the required detection probability at the $j$-th point, then one can define $x(j,k(j))$ as

$$x(j,k(j)) = \ln(\frac{1}{1-p_d(j,k(j))}-1) - \ln(\frac{1}{1-p_d^{req}(j)}-1). \tag{8.17}$$

In terms of $x(j,k(j))$, it is possible to write the change (evolution) in $x(j,k(j))$ when a a sensor is moved to the vicinity of the $j$-th point as

$$x(j,k(j)+1) = \frac{1}{\alpha(j)}x(j,k(j)) + B(j,i), \tag{8.18}$$

where, $B(j,i)$ is given as

$$\begin{aligned} B(j,i) &= (1-\frac{1}{\alpha(j)})\ln(\frac{1}{1-p_d^{req}(j)}-1) \\ &- \sqrt{2}(\frac{\Delta\eta' - (2^{k(j)+1}-1)p_{x,1}(j,k(j)) + (2^{k(j)}-1)p_d(j,k(j))}{\sqrt{(2^{k(j)+1}-1)p_{x,1}(j,k(j)+1)q_{x,1}(j,k(j)+1)}}). \end{aligned} \tag{8.19}$$

$B(j,i)$ in essence quantifies the change in $x(j,k(j))$ resulting from placing a sensor at the $i$-th grid point.

Considering all points on the grid, it is possible to write Eqn.(8.18) in matrix form as

$$\mathbf{x}_{m+1} = \mathbf{A}_m\mathbf{x}_m + \mathbf{B}_m\mathbf{u}_m \tag{8.20}$$

where, the matrix subscript $m$ denotes the total number of sensors that have been redeployed. The matrix $\mathbf{A}_m$ is a diagonal matrix of dimension $N_xN_y$, where the $j$-th diagonal entry is $A(j,j) = \frac{1}{\alpha(j)}$. The matrix $\mathbf{B}_m$ is a square matrix of dimension $N_xN_y$, with its $(j,i)$ entry is equal to $B(j,i)$ in Eqn.(**??**). Each element in the $N_xN_y \times 1$ $m$-th deployment vector $\mathbf{u}_m$ corresponds to one of the grid points. The $j$-th element in $\mathbf{u}_m$, in essence, indicates the contribution of placing a sensor at the $j$-th grid point on the overall detection probability.

As stated in Eqn.(**??**), our goal is to minimize the difference between the achieved and required performance. To do this, we choose to minimize $m$-th step cost function $J_m$, given as

$$J_m = \frac{1}{2}(\mathbf{x}_m^T\mathbf{Q}_m\mathbf{x}_m + \mathbf{u}_m^T\mathbf{R}_m\mathbf{u}_m), \tag{8.21}$$

where, the matrices $\mathbf{Q}_m$, $\mathbf{R}_m$ are square positive definite matrices of dimension $N_xN_y$. In essence, $J_m$ quantifies the deviation error between achieved and required detection performances. To avoid incurring a penalty for satisfying/exceeding detection requirements, the error terms corresponding to a point where the detection requirement has been met/exceeded is set to zero in $J_m$. Noting that there are $M$ mobile sensors to be moved, we can state the

$M$-step system model and cost function as

$$\underset{\mathbf{u}_m}{\text{argmin}}\, J = \frac{1}{2}\mathbf{x}_M^T\mathbf{Q}_f\mathbf{x}_M + \frac{1}{2}\sum_{m=0}^{M-1}(\mathbf{x}_m^T\mathbf{Q}_m\mathbf{x}_m + \mathbf{u}_m^T\mathbf{R}_m\mathbf{u}_m)$$
$$\text{subject to} \begin{cases} \mathbf{p}_f^n = \mathbf{p}_f^{req} \\ \mathbf{x}_{m+1} = \mathbf{A}_m\mathbf{x}_m + \mathbf{B}_m\mathbf{u}_m,\ m = 0, \ldots, M-1 \\ \mathbf{1}^T\mathbf{D} = n \end{cases} \tag{8.22}$$

The quadratic cost function $J$ along with the linear system model in Eqn.(8.20) constitute a linear quadratic regulator (LQR) problem. Solving an LQR problem, is equivalent to finding the optimal control vector that minimizes the cost function. One can use several methods [30], [31] to solve the LQR problem. However, due to the fact that the system model is dynamic (i.e., $\mathbf{A}_m$ and $\mathbf{B}_m$ vary with $m$), solving the LQR problem can be done by solving a single step (one step horizon) optimization problem. This can be done by setting the gradient of the single step cost function $J_m$ with respect to $\mathbf{u}_m$ to zero, and solving for $\mathbf{u}_m$ as

$$\mathbf{u}_m = -(\mathbf{R}_m + \mathbf{B}_m^T\mathbf{Q}_m\mathbf{B}_m)^{-1}\mathbf{B}_m^T\mathbf{Q}_m\mathbf{A}_m\mathbf{x}_m. \tag{8.23}$$

We note, that in calculating the new positions of the mobile sensors we need to account for any performance degradation that might result from sensor movement. This can be done in our approach, simply by calculating the achieved performance that results when not taking the mobile sensors contribution into account. Neglecting any contribution mobile sensors make in the decision process, we calculate $\mathbf{x}_0$ which acts as the initial condition of the LQR system evolution model. This ensures that the degradation in performance in the original areas covered by the mobile sensors is given an equal weight in making the redeployment decision as the areas that suffered performance degradation due to loss of sensors. Furthermore, it is desirable that mobile sensors move such that their total moving distance, which is proportional to their energy consumption, is minimized. To ensure this, sensor move only after the calculation of the all new mobile sensor positions. Every mobile sensor is moved to the closest new position as calculated from the LQR solution. This ensures that the total moving distance as well as the energy required for redeployment is

minimized.

## 8.3 Self-Healing Algorithm

In this section, we discuss our self-healing algorithm. Assuming that the FC knows the positions of the compromised sensors, a first attempt at healing the network is made by adjusting the decision thresholds associated with the affected points. This can be done using Eqn.(8.12). The network's user can specify a minimum performance improvement goal for both false alarm and detection requirements, which are given in vectors $\epsilon_{p_f}$ and $\epsilon_{p_d}$, respectively. If the performance improvement is not satisfactory, sensor redeployment can be incorporated. We first assume that the mobile sensors are removed from the grid, and then the decision thresholds are recalculated. The removal of these mobile sensors allows us to quantify the performance change within their immediate area resulting from moving these sensors to other points in the grid. We then proceed with the redeployment of the $M$ mobile sensors. The new achieved false alarm and detection profiles are used in constructing $\mathbf{x}_{N_0-l-m}$ and $\mathbf{B}_{N_0-l-m}$ for $m = 1, \ldots, M$ as in Eqns.(8.17) and (**??**). The optimal control vector $\mathbf{u}_{N_0-l-m}$ is calculated as in Eqn.(8.23) . We then store the index of the grid point $\hat{i}_m$ corresponding to the maximum entry of $\mathbf{u}_{N_0-l-m}$. We choose to deploy a sensor at the $\hat{i}_m$-th point since placing a sensor at that point has the largest contribution in minimizing the difference between the achieved and required detection performances. The redeployment process is repeated till the set $\{\hat{I} = \hat{i}_m, m = 1, \ldots, M\}$ of all redeployment positions are evaluated. To minimize the total energy consumed in moving the sensors to their new locations, we compare the set $\hat{I}$ to the set $\{\hat{J} = \hat{j}, j = 1, \ldots, M/$ which indicates the initial positions of the mobile sensors. A mobile sensor is then moved to the point in $\hat{I}$ which is closest to its original location.

**Algorithm 7** Self-healing Algorithm

Determine locations of the $l$ compromised sensors
Adjust decision thresholds as in Eqn.(8.12).
Calculate new false alarm $\mathbf{p}_f^{N_0-l}$ and detection probability $\mathbf{p}_d^{N_0-l}$ profiles.
**if** $\mathbf{p}_f^{N_0-l} < \epsilon_{p_f}$ OR $\mathbf{p}_f^{N_0-l} < \epsilon_{p_d}$
Assume the $M$ mobile sensors are compromised.
Recalculate decision thresholds using Eqn.(8.12).
Calculate false alarm $\mathbf{p}_f^{N_0-(l+M)}$ and detection $\mathbf{p}_d^{N_0-(l+M)}$ profile.
Set $m = 0$
**for** $m = 1 : M$
Calculate control vector $\mathbf{u}_m$ as in Eqn.(8.23)
Find $\hat{i}_m$ where $\mathbf{u}_m(\hat{i}_m) = \max \mathbf{u}_m$, s.t. $\mathbf{D}(\hat{i}_m) = 0$
Store $\hat{i}_m$ (i.e., $I = [I\hat{i}_m]$).
Update $\mathbf{D}$ (i.e., $\mathbf{D}(\hat{i}_m) = 1$).
**end**
Calculate the distance $d(\hat{i}_m, \hat{j}_{\hat{m}})$ $\forall \hat{i}_m, \hat{j}_m$
Redeploy sensors
**end**

## 8.4 Simulation Results

In the first experiment, we examine the improvement in performance resulting from using our proposed self-healing approaches; decision threshold adjustment and sensor redeployment. The area of interest is a $15 \times 15$ grid. The performance requirements are not uniform and are as shown in Fig.8.1 and are set to $P_f^1 = 0.01$, $P_d^1 = 0.8$, $P_f^2 = 0.05$ and $P_d^1 = 0.6$. The sensing parameters are given as $\alpha = 0.05$, $p_f = 0.1$ and $R = 5$. Initially, we assume that 23 sensors are deployed such that both false alarm and detection requirements are satisfied. We assume that $M = 1$ sensor is mobile while the remaining sensors are static. Fig.8.2 (a) and (b) show the degradation in performance after the loss of 3 sensors. Though the degradation in false alarm performance was minimum, the degradation in the detection performance is significant and affects several points. Fig.8.2 (c) and (d) depict the performance of the system after threshold adjustment is performed. We note that the detection performance has improved at several points in the grid. However, this improvement is at the cost of degradation in the false alarm performance of the system. Though a false alarm requirement

can be achieved by setting the decision threshold high enough, it is detrimental to the detection performance. Therefore, the improvement in the detection performance is at the expense of degradation in false alarm performance. The performance of the system after redeployment of the mobile sensor is shown in Fig.8.2 (e) and (f). We note that in comparison to both detection performances before and after threshold adjustment, the detection performance after redeployment has improved considerably with very few points not meeting their detection requirements. In addition, we note that the redeployment algorithm was efficient in meeting the false alarm requirements as well. In comparison to the false alarm performance after threshold adjustment, we note that using redeployment resulted in meeting the false alarm requirements at a great number of the points where requirements were not met earlier. Fig.8.3 shows the original distribution of sensors, the failed sensors and the new position of the mobile sensor for the simulation case corresponding to Fig.8.2. Next, we quantify the performance improvement after using sensor redeployment. We define the percentage improvement as the ratio between the number of points where false alarm/detcetion requirements have been met after redeployment to the total number of grid points. We compare the improvement in the system's performance with respect to the percentage improvement in the number of points where false alarm/detection requirements have been met after redeployment in comparison to their number before redeployment. Fig.8.4 illustrates the average percentage performance improvement as we vary the number of compromised sensors. Since the performance degradation in system performance depends on the positions of compromised sensors, we assume that sensors are compromised randomly. The results in Fig.8.4 are the average of 250 Monte Carlo runs. We note that when using the proposed redeployment algorithm one can achieve significant improvement in both the false alarm and detection performances of the network.

We next illustrate the improvement in performance when the number of mobile sensors is set to $M = 3$. Fig. 8.5 (a) and (b), show the degradation in performance when 3 sensors are lost. The detection and false alarm error profiles after threshold adjustment are shown

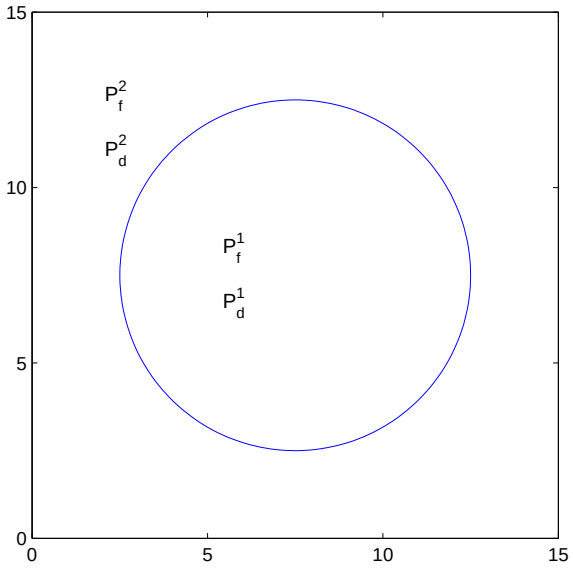



**Figure 8.1**: *Nonuniform detection requirements* $N_x = N_y = 15$

in Fig.8.5 (c) and (d), respectively. The detection and false alarm error profiles after sensor redeployment are shown in Fig.8.5 (e) and (f). We note that the same discussion for Fig.8.4 can be applied for Fig.8.5. Fig.8.6, illustrates the average performance improvement as a function of the number of lost sensors. We note that similar to the previous case of $M = 1$, redeploying 3 sensors results in improvement in the system's performance. In fact, a better improvement in performance is possible using 3mobile sensors instead of 1. For example, for $M = 1$ the average improvement for both false alarm/ detection performance is almost 7% when 6 sensors are lost. On the other hand, when $M = 3$ the average improvement in false alarm and detection performance is approximately 18% and 14%, respectively. This agrees with our expectation that the more mobile sensors one can move, the better one can mitigate performance degradation by having more flexibility in redeployment.

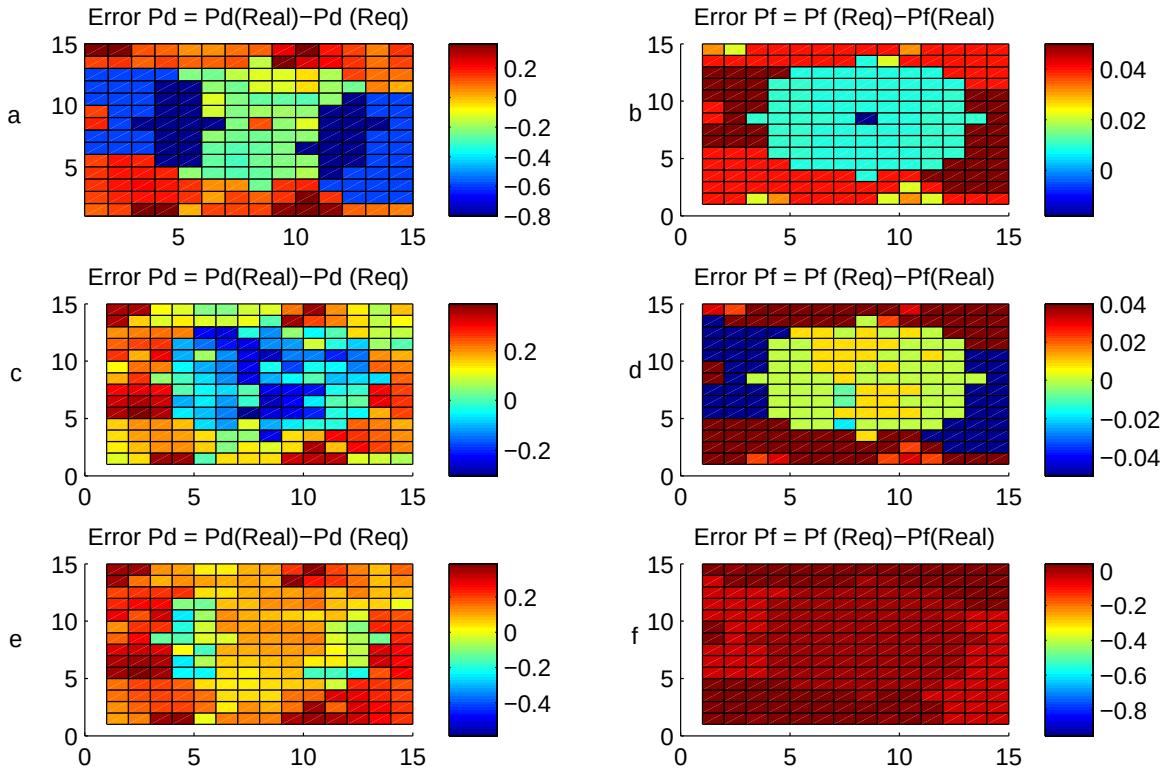



**Figure 8.2**: *Change in performance* $M = 1$, $l = 3$

## 8.5 Summary

In this chapter, we proposed two approaches for self-healing in a DSN employing the optimal decision rule. The first approach is decision threshold adjustment, where the decision threshold at the FC is updated to account for sensor failures. We proposed using an approximation of the decision statistic distribution, which enabled us to re-evaluate the decision threshold with low computational complexity. The second approach that we proposed is sensor redeployment. Assuming that a number of the deployed sensors are mobile, we determine the positions where these sensors should move to in order to mitigate performance degradation. Towards this, we modeled redeployment as a linear quadratic regulator (LQR) problem. To minimize the total moving energy, a mobile sensor moves from its original location to

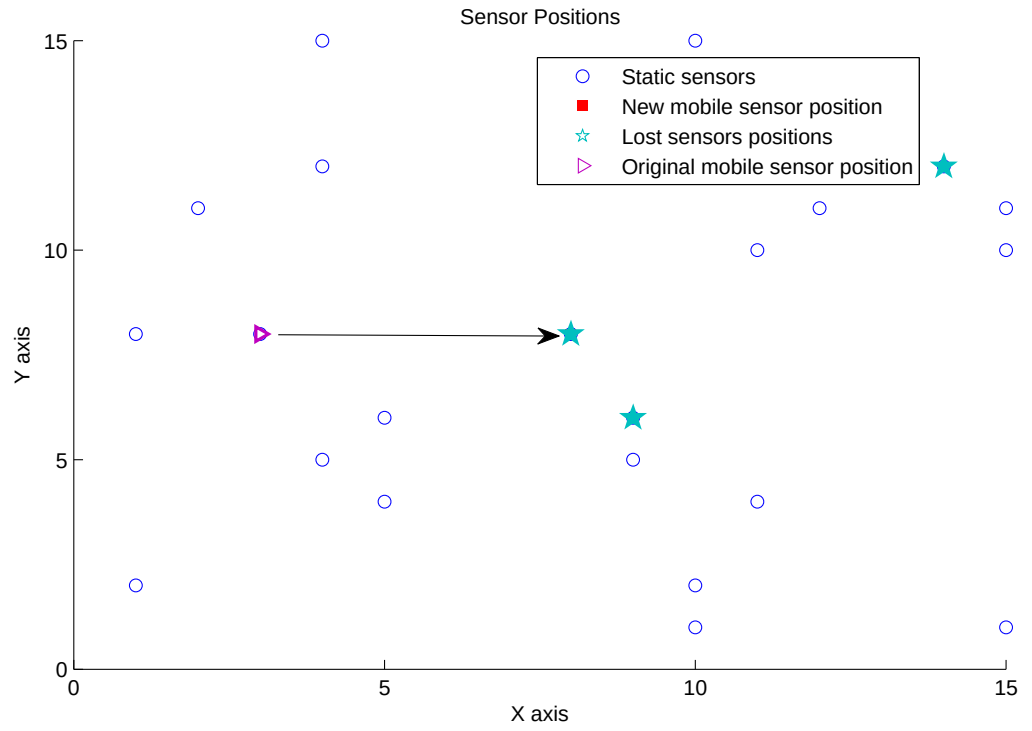



**Figure 8.3**: *Sensor redeployment*

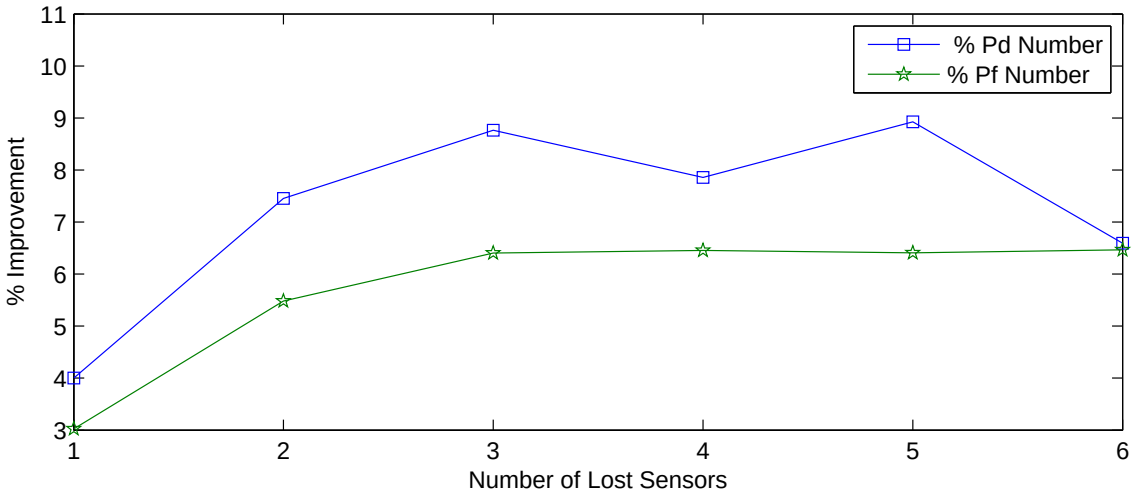



**Figure 8.4**: *% Improvement after redeployment-1 mobile sensor*

the location closest to it among the new sensor locations. Simulation results illustrated the efficacy of using the proposed approaches in self-healing the network by improving its

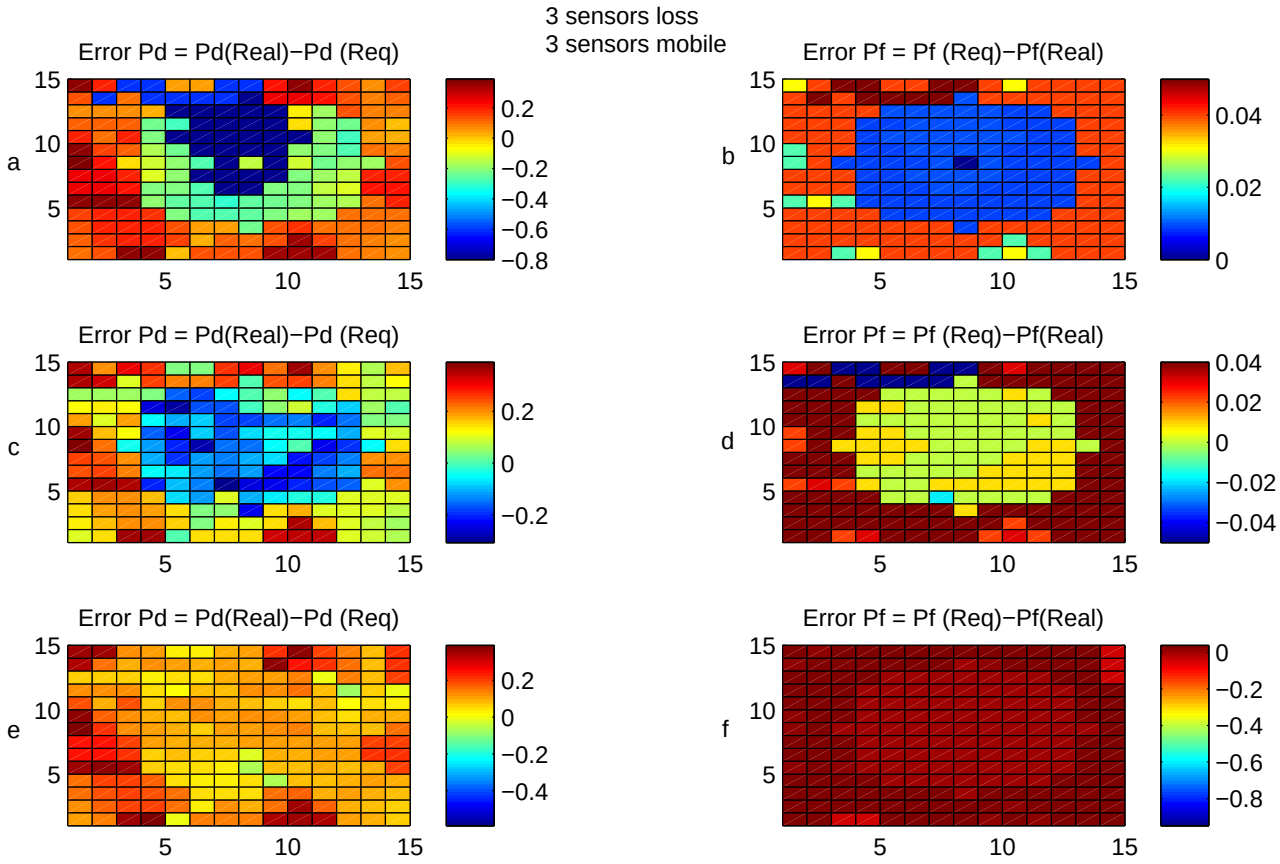



**Figure 8.5**: *Change in performance* $M = 3$, $l = 3$

detection performance after sensor failures.

In the next chapter, we briefly outline our contributions in this dissertation and future research directions.

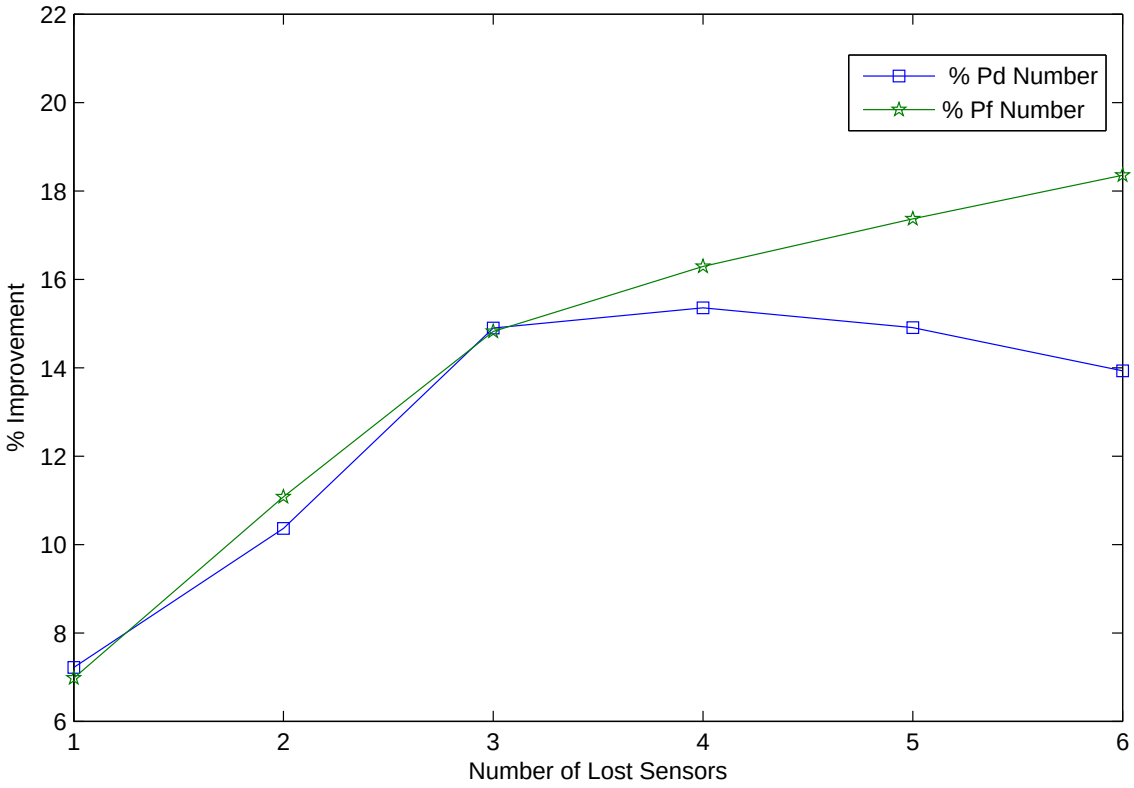



**Figure 8.6**: *% Improvement after redeployment-3 mobile sensors*

# Chapter 9

# Conclusion and Future Work

In this chapter, we summarize the contributions of this dissertation and discuss future research directions.

## 9.1 Summary of Key Contributions

In this dissertation, we considered two major challenges related to deployment in sensor networks for detection applications. First, we investigated the sensor deployment problem. Using concepts from optimal control theory, the deployment problem was modeled as an optimal control problem. Specifically, the deployment problem was formulated as a linear quadratic regulator (LQR) problem which is a very well-behaved and studied problem in optimal control theory literature. The proposed LQR formulation offered a unified treatment of the sensor deployment problem. In addition, our LQR formulation provided a rigorous and analytical framework to study the deployment problem. This is in contrast to prior efforts in sensor deployment that are mainly heuristic in nature. We first investigated a DSN employing the logical OR fusion rule with detection requirements only (no false alarm requirements). We proposed an LQR-based deployment algorithm along with a low complexity heuristic alternative. Simulation results showed that using our proposed algorithms can save up to 30% in the number of sensors in comparison to the Diff_Deploy algorithm [21]. Next, we considered a network employing value fusion with both false alarm and detection requirements. Once again, we proposed an LQR formulation and a deployment algorithms

based on it. In contrast to prior efforts, our proposed algorithm can be applied to networks with nonuniform false alarm and detection requirements. We also considered the problem of dynamically calculating the maximum collaboration radius instead of a fixed collaboration radius (which was a common assumption in prior works). Simulation results illustrated the advantages of using our proposed algorithms and a dynamic collaboration radius in reducing the number of sensors needed to meet performance requirements in comparison to the D&C algorithm [22]. We next considered sensor deployment in a DSN employing the majority decision fusion rule. Using results from nonparametric statistical theory, we were able to model the effect of deploying a single sensor on the network overall detection performance. This enabled us to model sensor deployment as an LQR problem. Subsequently, we proposed an LQR-based sensor deployment algorithm. Simulation results showed the advantages of using the proposed algorithm versus using a greedy deployment algorithm. Finally, we examined deployment for a network employing the optimal decision fusion rule [44]. To the best of or knowledge, this is the first analytical attempt to study sensor deployment incorporating optimal decision fusion. One reason for the lack of prior efforts in this arena is the complexity associated with the probability of detection and false alarm analysis. To reduce the computational complexity, we proposed a novel closed form approximation of false alarm/ detection probabilities. Using this approximation, we were able to develop an LQR formulation of the sensor deployment problem and propose an LQR-based deployment algorithm. The performance of the proposed algorithm was shown to outperform that of a greedy deployment algorithm.

Secondly, we investigate methods to mitigate the effects of losing sensors. While an adequate number of sensors can be deployed to meet certain performance requirements, losing sensors can lead to less than optimal performance. Thus, it is important to devise strategies to mitigate performance degradation due to loss of sensors without the addition of new sensors. In networks, this feature is referred to as the "self-healing" capability of the detection system. For the first time, we considered self healing in a network employing the

optimal decision fusion rule. The first proposed approach relies on adjusting the decision threshold at the fusion center after the loss of sensors. Using the closed form approximation proposed in chapter 7, we can efficiently update the decision threshold. The second proposed approach, sensor redeployment, uses sensor mobility. We were able to model the redeployment problem as an LQR problem. This LQR formulation allows us to consider the network's overall detection performance instead of focusing on a localized part as in prior efforts. Simulation results illustrated that the proposed algorithms are able to mitigate performance degradation to a large extent.

In summary, a novel treatment of the sensor deployment problem has been proposed. This control theoretic approach provides a unified to study sensor deployment with (1) uniform and nonuniform false alarm and detection requirements, and (2) a wide range of fusion rules. The proposed approach can serve as a planning tool for many practical DSNs. The proposed approaches can be used to (1) determine the minimum number of sensors needed to satisfy some given performance requirements, and (2) quantify the performance of an existing deployment with low computational complexity.

## 9.2 Future Work

Some possible future research directions are provided in this section. We first discuss extensions to our work proposed in chapters 5-8.

- Throughout this dissertation, we assumed that sensors have a common false alarm rate and detection sensing profile. We can extend our work to the case of heterogeneous sensors with different false alarm rates and/or detection sensing profiles.

- In chapter 4, we have studied deployment for a basic terrain model. We can extend our work to more realistic terrain models under which the various proposed deployment algorithms can be evaluated.

- In chapter 5, we evaluated the dynamic collaboration radius in a DSN employing value

fusion. We can extend this by evaluating the collaboration radius for DSNs employing the majority fusion or optimal decision fusion rules.

- In chapter 8, we developed approaches for self healing to mitigate performance degradation due to loss of sensors. Consider a situation where detection/ false alarm requirements change from their initial state for some reason (e.g., change in mission priorities). In this case, how can we "self-adapt" the DSN to meet the modified requirements?

In addition to the extensions discussed above, we can investigate the deployment and self healing problems under new dimensions. A subset of possible next steps are discussed below:

- *Modeling effects of communications:* In this dissertation, we assumed communication between sensors and the fusion center to be error free. One can can consider the deployment problem when the communication channel is noisy/unreliable. Understanding the effects of communication delay on the fusion process as well as deployment is a problem that needs further study.

- *Network architectures:* The networks examined in this dissertation use a parallel architecture with a fusion center. It would be an interesting problem to investigate sensor deployment in networks using other architectures (e.g., serial architecture).

- *Target localization/tracking*: In the target localization problem, noisy measurements from multiple sensors are processed to determine the position of a target of interest. In general, sensor measurements are distance dependent and therefore it is important to determine the deployment positions where sensors are placed to minimize the error in estimating the target position.

- *Game-theoretic self healing:* We have considered the problem of minimizing the total energy consumed in sensor redeployment. However, we did not incorporate the

amount of residual energy available to each sensor. Incorporating this in the solution of the redeployment problem can significantly improve the network's lifetime, since the movement of a sensor will be dependent on its residual energy. We note that this problem can be studied using game-theoretic approaches.

# Bibliography

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, pp. 2292–2330, 2008.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[3] C. Chong and S. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings Of the IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug 2003.

[4] G. Simon, M.Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *Proceedings of the Second International Conference on Embedded Networked Sensor Systems*, 2004.

[5] A. Baggio, "Wireless sensor networks in precision agriculture," Delft University of Technology, The Netherland, Tech. Rep., 2005.

[6] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M.Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, pp. 18–25, 2006.

[7] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, and D. G. andW. Hong, "A macroscope in the redwoods," in *Proceedings of the Third International Conference on Embedded Networked Sensor Systems*, 2005.

[8] M. Brusco and S. Stahl, *Branch-and-Bound Applications in Combinatorial Data Analysis*. Springer, 2005.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[10] S. L. Padula and R. K. Kincaid, "Optimization strategies for sensor and actuator placement," NASA, Tech. Rep., 1999.

[11] X. Liu, D. Begg, and D. Matravers, "Optimal topology/actuator placement design of structures using sa," *Journal of Aerospace Engineering*, vol. 10, no. 3, pp. 119–125, 1997.

[12] M. Simpson and C. H. Hansen, "Use of genetic algorithms to optimize vibration actuator placement for active control of harmonic interior noise in a cylinder with floor structure," *Noise Control Eng. J.*, vol. 44, no. 4, pp. 169–184, 1996.

[13] J. Berry, W. E. Hart, C. E. Phillips, J. G. Uber, and J. Watson, "Sensor placement in municipal water networks with temporal integer programming models," *J. Water Resources Planning and Management*, 2006.

[14] D. Hamel, M. Chwastek, B. Farouk, M. Kam, and K. Dandekar, "A computational fluid dynamics approach for optimization of a sensor network," in *Proceedings of the IEEE International Workshop for Homeland Security, Contraband Detection and Personal Safety*, 2006.

[15] T. Berger-Wolf, W. E. Hart, and J. Saia, "Discrete sensor placement problems in distribution networks," *Journal of Mathematical and Computer Modelling*, vol. 42, no. 13, pp. 1385–1396, Dec 2005.

[16] V. Isler, S. Kannan, and K. Daniilidis, "Sampling based sensor-network deployment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*, 28 Sept.-2 Oct. 2004 2004, pp. 1780 – 1785.

[17] S. H. Q. E. C. Chakrabarty, K. Iyengar, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448 – 1453, Dec 2002.

[18] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, pp. 1448–1453, 2002.

[19] S. Shakkottai, S. Srikant, and N. Shroff, "Unreliable sensor grids: Coverage, connectivity and diameter," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, vol. 2, 30 March -3 April 2003, pp. 1073–1083.

[20] Y. Zou and K. Chakrabarty, "Uncertainty-aware and coverage-oriented deployment for sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, no. 7, pp. 788–798, July 2004.

[21] J. Zhang, T. Yan, and S. H. Son, "Deployment strategies for differentiated detection in wireless sensor networks," in *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks SECON*, vol. 1, 28 Sept 2006, pp. 316–325.

[22] Z. Yuan, R. Tan, G. Xing, C. Lu, Y. Chen, and J. Wang, "Fast sensor deployment for fusion-based target detection," in *IEEE Real-Time Systems Symposium (RTSS'08)*, Dec 2008.

[23] B. W. Wah and T. Wang, "Simulated annealing with asymptotic convergence for nonlinear constrained global optimization," in *Principles and Practice of Constraint Programming*. Springer-Verlag, 1999, pp. 461–475.

[24] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja, "Sensor deployment strategy for target detection," in *Proceedings of the 1st ACM international workshop on wireless sensor networks and applications*, 2002, pp. 42–48.

[25] S. K. Jayaweera, "Optimal node placement in decision fusion wireless sensor networks for distributed detection of a randomly-located target." Military Communications Conference, MILCOM, 29-31 Oct 2007, pp. 1–6.

[26] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640 –652, June 2006.

[27] T. P. Lambrou and C. G. Panayiotou, "Collaborative area monitoring using wireless sensor networks with stationary and mobile nodes," *EURASIP Journal on Advances in Signal Processing*, pp. 1–16, Jan 2009.

[28] G. Xing, J. Wang, Z. Yuan, R. Tan, L. Sun, Q. Huang, X. Jia, and H. C. So, "Mobile scheduling for spatiotemporal detection in wireless sensor networks," *IEEE Transactions Parrllel and Distributed Systems*, 2010, to be published.

[29] R. Tan, G. Xing, J. Wang, and H. C. So, "Collaborative target detection in wireless sensor networks with reactive mobility," Jan 2008, pp. 150 –159.

[30] M. Mahmoud and M. Singh, *Discrete Systems: Analysis, Control and Optimization*. Springer-Verlag, 1984.

[31] A. Sage and C. White, *Optimum Systems Control*. Prentice-Hall, 1977.

[32] P. Dorato, C. T. Abdallah, and V. Cerone, *Linear Quadratic Control: An Introduction*. Krieger Pub Co, 2000.

[33] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Prentice-Hall International, 1989.

[34] V. Sima, *Algorithms for Linear-Quadratic Optimization*. Marcel Dekker, 1996.

[35] A. Ababnah and B. Natarajan, "Sensor deployment as an optimal control problem," in *Proceedings of 18th Internatonal Conference on Computer Communications and Networks, (ICCCN)*, Aug 2009, pp. 1 –5.

[36] ——, "Optimal control-based strategy for sensor deployment," *IEEE Transactions on on Systems, Man and Cybernetics, Part A*, no. 99, pp. 1 –8, 2010.

[37] ——, "Optimal sensor deployment for value-fusion based detection," *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pp. 1 –6, Nov 2009.

[38] ——, "Optimal control sensor deployment incorporating centralized data fusion," *IEEE Transactions on Aerospace and Electronic Systems*, accepted for publication.

[39] ——, "Control theoretic sensor deployment approach for data fusion based detection," in *The 6th IEEE International Conference Distributed Computing in Sensor Systems, DCOSS*, ser. Lecture Notes in Computer Science, vol. 6131. Springer, 2010, pp. 92–101.

[40] ——, "An LQR formulation of sensor deployment for decision fusion based detection," Dec 2010, accepted for publication.

[41] ——, "On sensor deployment for decision fusion based detection," *IEEE Transactions on Parallel and Distributed Systems*, to be submitted.

[42] ——, "Sensor deployment in detection networks employing optimal decision fusion rule," *IEEE Transactions on Signal Processing*, under review.

[43] ——, "Mobility assisted self healing in sensor networks for detection applications," *IEEE Transactions on Signal Processing*, to be submitted.

[44] P. Varshney, *Distributed Detection and Data Fusion*. Springer-Verlag, 1997.

[45] H. V. Poor, *An Introduction to Signal Detection and Estimation*. Springer, 1994.

[46] B. C. Levy, *Principles of Signal Detection and Parameter Estimation*. Springer, 2008.

[47] R. R. Tenney and N. R. Sandell, "Detection with distributed sensors," in *The 19th IEEE Conference on Decision and Control*, vol. 19, Dec 1980, pp. 433 –437.

[48] Z. Chair and P. Varshney, "Optimal data fusion in multiple sensor detection systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-22, no. 1, pp. 98 –101, Jan 1986.

[49] A. Aziz, M. Tummala, and R. Cristi, "Optimal data fusion strategies using multiple-sensor detection systems," in *The Thirty-First Asilomar Conference on Signals, Systems Computers*, vol. 1, Nov 1997, pp. 941 –945.

[50] R. Viswanathan and P. Varshney, "Distributed detection with multiple sensors I. fundamentals," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 54–63, Jan 1997.

[51] A. Ansari, "Some problems in distributed detection," Master's thesis, Dept. Electrical Engineering, S.Illinois Univ., Carbondale, IL, 1987, MS Thesis.

[52] R. Viswanathan, S. Thomopoulos, and R. Tumuluri, "Optimal serial distributed decision fusion," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 24, no. 4, pp. 366 –376, July 1988.

[53] P. Benner, J.-R. Li, and T. Penzl, "Numerical solutions of large-scale lyapunov equations, riccat equations and linear-quadratic optimal control problems," *Numerical Linear Algebra with Applications*, vol. 15, pp. 755–777, 2008.

[54] A. Elfes, "Occupancy grids: A stochastic spatial representation for active robot perception," in *Proceedings of the Sixth Conference on Uncertainty in AI*, 1990.

[55] M. F. Duarte and Y. H. Hu, "Vehicle classifcation in distributed sensor networks," *Journal of Parrallel and Distributed Computing*, vol. 64, no. 7, pp. 826– 838, 2004.

[56] D. Li and Y. H. Hu, "Energy based collaborative source localization using acoustic micro-sensor array," *EUROSIP J. Applied Signal Processing*, vol. 2003, pp. 321–337, 2003.

[57] ——, "Energy based collaborative source localization using acoustic micro-sensor array," *J. Applied Signal Processing*, vol. 2003, pp. 321–337, 2003.

[58] J. K. Patel and C. B. Read, *Handbook of the Normal Distribution.* Marcell Dekker. Inc, 1982.

[59] V. C. Raykar, R. Duraiswami, and B. Krishnapuram, "A fast algorithm for learning a ranking function from large-scale data sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 7, pp. 1158–1170, July 2008.

[60] W. Hoeffding, "On the distribution of the number of successes in independent trials," *Ann. Math. Statis*, vol. 27, 1956.

[61] K. P. Choi and A. Xia, "Approximating the number of successes in independent trials: Binomial versus poisson," *Annals of Applied Probability*, vol. 12, no. 4, pp. 1139–1148, 2002.

[62] N. L. Johnson, A. W. Kemp, and S. Kotz, *Univariate Discrete Distributions.* John Wiley, 2005.

[63] M. E. Liggins, D. L. Hall, and J. Llinas, Eds., *Handbook of Multisensor Data Fusion: Theory and Practice.* CRC Press, 2009.

[64] Z. Chair and P. Varshney, *IEEE Transactions on Aerospace and Electronic Systems.*

[65] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Stat. Assoc*, vol. 58, pp. 13–30, 1963.

[66] M. Talagrand, "The missing factor in hoeffding's inequalities," *Ann. Inst . Henri Poincare*, vol. 31, pp. 689–702, 1995.

[67] Kh.Batirov, D. Manevich, and S.V.Nagaev, "The Esseen inequalityfor sum of a random number of differently distributed random variables," *Matematicheskie Zametki*, vol. 22, pp. 143–146, 1980.

[68] P.Vellaisamy and B.Chaudhuri, "Poisson and compound poisson approximations for random sums of random variables," *J. Applied Prob.*, vol. 33, pp. 127–137, 1996.

[69] H. Delic, P. Papantoni-Kazakos, and D. Kazakos, "Fundamental structures and asymptotic performance criteria in decentralized binary hypothesis testing," *IEEE Transactions on Communications*, vol. 43, no. 1, pp. 32 –43, Jan 1995.

[70] R. Niu, P. Varshney, M. Moore, and D. Klmaer, "Distributed fusion in a wireless sensor network with a large number of sensors," in *7th Int. Conf. Information Fusion*, June 2004.

[71] S. A. Aldosari and J. M. F. Moura, "Detection in sensor networks: The saddlepoint approximation," *IEEE Transactions on Signal Processing*, vol. 55, no. 1, pp. 327 –340, Jan 2007.

[72] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables.* Dover Publications, 1972.

# Appendix A

# Sensor Deployment in Value Fusion Detection Networks – Amplitude Measurements

## A.1 System Model

The measurement ($U_i$) of the $i$-th sensor, under the two hypothesis (i.e., $H_0$ and $H_1$), is given as

$$U_i = N_i \mid H_0 \text{ true} \tag{A.1}$$

$$U_i = A(d_i) + N_i \mid H_1 \text{ true} \tag{A.2}$$

where, $N_i \sim \mathcal{N}(0, \sigma^2)$. Furthermore, we assume that sensor measurement noise is i.i.d. The signal amplitude $A(d_i)$ is distance dependent, and is given as

$$A(d_i) = \begin{cases} A_0 & \text{if } d_i \leq d_0 \\ \frac{A_0}{(d_i/d_0)^\kappa} & \text{if } d_0 < d_i \leq d_{max} \\ 0 & \text{if } d_i > d_{max} \end{cases} \tag{A.3}$$

where; $d_i$ is the distance between the i-th sensor and the target/phenomena, $d_{max}$ is the detection radius and $\kappa$ is a decay factor that depends on the environment. We also assume that the collaboration radius $R_c$ is equal to $d_{max}$. We note however, that our proposed deployment framework is general and does not depend on any particular choice of the signal amplitude function.

A detection decision is made regarding the presence or absence of a target at the $j$-th point on the grid by combining the available sensor measurements. This is done at a fusion center (FC), which calculates a decision statistic $T_j$ and compares it to a decision threshold $\eta(j, n_j)$, where $n_j$ is the number of sensors reporting measurements regarding the j-th point (i.e., less than $d_{max}$ meters from the $j$-th point). A decision is made according to the following non-randomized decision rule,

$$\delta(T_j) = \begin{cases} H_0 & \text{if } T_j \leq \eta(j, n_j) \\ H_1 & \text{if } T_j > \eta(j, n_j). \end{cases} \tag{A.4}$$

In this paper, we adopt the value fusion detection strategy. The decision statistic $T_j$ is given as the average of the measurements reported by the $n_j$ sensors, i.e.,

$$T_j = \frac{1}{n_j} \sum_{i=1}^{n_j} U_i \tag{A.5}$$

therefore, the detection probability at the $j$-th point, denoted as $p_d(j, n_j)$, is given as

$$\begin{aligned} p_d(j, n_j) &= Pr(T_j \geq \eta(j, n_j) |\ H_1 \text{ true}) && \text{(A.6)} \\ &= Pr(\frac{1}{n_j} \sum_{i=1}^{n_j} [A(d_i) + N_i] \geq \eta(j, n_j)) && \text{(A.7)} \\ &= Q(\frac{\sqrt{n_j}\eta(j, n_j)}{\sigma} - \frac{1}{\sigma\sqrt{n_j}} \sum_{i=1}^{n_j} A(d_i)) && \text{(A.8)} \end{aligned}$$

where, $Q(x)$ is given as $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-t^2}{2}} dt$. Similarly, the false alarm probability, denoted as $p_f(j, n_j)$, at point $(j)$ is given as

$$\begin{aligned} p_f(j, n_j) &= Pr(T_j \geq \eta(j, n_j) |\ H_0 \text{ true}) && \text{(A.9)} \\ &= Pr(\frac{1}{n_j} \sum_{i=1}^{n_j} N_i \geq \eta(j, n_j) |\ H_0 \text{ true}) && \text{(A.10)} \\ &= Q(\frac{\sqrt{n_j}\eta(j, n_j)}{\sigma}). && \text{(A.11)} \end{aligned}$$

The FC calculates the decision statistics associated with each point on the grid. The decision threshold is calculated using knowledge of the number of sensors that are within the detection

radius, the noise variance and the false alarm requirement associated with every point. In practice, the FC receives measurements from all sensors in the grid and performs a series of sensor measurement averages corresponding to each point on the grid.

The deployment problem that we examine in this work can now be stated as follows: Given $\mathbf{p}_f^{req}$ and $\mathbf{p}_d^{req}$ and a fixed number of sensors $K$, how can we deploy these sensors in a value fusion based detection system, such that the squared error (SE) between achieved and required detection probabilities is minimized while satisfying false alarm requirements? If we denote the achieved false alarm and detection probability vectors after $K$ sensor have been deployed as $\mathbf{p}_{f,K}$ and $\mathbf{p}_{d,K}$, then we can mathematically state our problem as

$$\begin{aligned} \underset{\mathbf{u}}{\operatorname{argmin}} & \sum_{j:p_{d,K}(j)<p_d^{req}(j)} (\mathbf{p}_{d,K}(j) - \mathbf{p}_d^{req}(j))^2 \\ & \text{subject to} \begin{cases} \mathbf{p}_{f,K} = \mathbf{p}_f^{req} \\ \mathbf{1}^T\mathbf{u} = K \end{cases} \end{aligned} \tag{A.12}$$

where, $\mathbf{u}$ is the deployment vector. The deployment vector is an $N_xN_y \times 1$ vector. Its entries indicate the number of sensors at each point on the grid, and take values of either 0 or 1. $\mathbf{1}^T$ indicates the transpose of an $N_xN_y \times 1$, with all entries set to 1.

## A.2 Optimal Control Formulation

The deployment problem stated earlier can be thought of as a optimal control problem. The SE between achieved and required detection probabilities can be mapped into the cost function to be minimized in an optimal control problem. Furthermore, the set of optimal control vectors correspond to the sensor positions on the grid (i.e., the deployment vector). In an LQR problem, the optimal control vectors are solved sequentially, this means that in the proposed framework sensors are sequentially placed on the grid. In the next section, we illustrate that it is indeed possible to approximate the deployment problem as a linear quadratic regulator (LQR) problem. We will also discuss solving for the optimal control vectors (i.e., deployment vector in our problem) in the LQR problem.

In an LQR problem, the evolution of the state of the system (i.e., difference in detection probabilities in our problem) is governed by a linear relationship. In this section, we show that it is possible to linearly approximate the change in the difference between $\ln(\frac{1}{1-p_d(j,n_j)}-1)$ and $\ln(\frac{1}{1-p_d^{req}(j)}-1)$. Due to the monotone nature of the logarithmic function, minimizing the difference between these two quantities is equivalent to minimizing the difference between $p_d(j,n_j)$ and $p_d^{req}(j)$. Furthermore, we quantify the effect each entry in the control vector will have on the system's evolution.

Noting that we can approximate the $Q(\cdot)$ function as [59]

$$Q(x) \approx 1 - \frac{1}{1+e^{-\sqrt{2}x}}, \tag{A.13}$$

we can approximate $\ln(\frac{1}{1-p_d(j,n_j)}-1)$ as follows

$$\ln(\frac{1}{1-p_d(j,n_j)}-1) \approx \sqrt{\frac{2}{\sigma^2 n_j}}\sum_{i=1}^{n_j} A(d_i) - \sqrt{\frac{2n_j}{\sigma^2}}\eta(j,n_j). \tag{A.14}$$

Eqn.(A.14), illustrates that for a fixed $n_j$ and $\eta(j,n_j)$, the change in $\ln(\frac{1}{1-p_d(j,n_j)}-1)$ is approximately linear with respect to the signal amplitudes measured by the $n_j$ sensors.

Define $m(j,n_j)=\ln(\frac{1}{1-p_d(j,n_j)}-1)$ and $m^{req}(j)=\ln(\frac{1}{1-p_d^{req}(j)}-1)$. Having $n_j=k-1$, then it is possible to write $x(j,k)=m(j,k)-m^{req}(j)$ after the $k$-th sensor has been added within the detection radius of point $(j)$, as follows

$$\begin{aligned} x(j,k) &= x(j,k-1) + \sqrt{\frac{2}{\sigma^2 k}}A(d_k) \\ &\quad + \sqrt{\frac{2}{\sigma^2}}(\frac{1}{\sqrt{k}} - \frac{1}{\sqrt{k-1}})\sum_{i=1}^{n_j-1} A(d_i) \\ &\quad + \sqrt{\frac{2k}{\sigma^2}}\eta(j,k) + \sqrt{\frac{2(k-1)}{\sigma^2}}\eta(j,k-1). \end{aligned} \tag{A.15}$$

It is possible to write Eqn.(A.15) in matrix form as follows

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k. \tag{A.16}$$

where, $\mathbf{x}_K = [x(j, n_j), \forall j \in \mathcal{G}]^T$. $k$ is the total number of sensors in the grid. We note that, depending on the detection radius, the number of sensor covering a point $(j)$ might be in general less than $k$. The matrix $B$ is of dimension $N_x N_y \times N_x N_y$. The elements of the $B$, quantify the contribution of possible sensor positions to the detection probability. For example, the $(r, c)$th element of $\mathbf{B}$ is given as

$$\begin{aligned}\mathbf{B}(r,c) &= \sqrt{\frac{2}{\sigma^2(n_r+1)}}A(d(r,c)) - \sqrt{\frac{2(n_r+1)}{\sigma^2}}\eta(j, n_r+1) \\ &+ \sqrt{\frac{2}{\sigma^2}}\left(\frac{1}{\sqrt{n_r+1}} - \frac{1}{\sqrt{n_r}}\right)\sum_{i=1}^{n_r} A(d(r,i)) \\ &+ \sqrt{\frac{2(n_r)}{\sigma^2}}\eta(j, n_r)\end{aligned} \tag{A.17}$$

where, $d(r, c)$ is the distance between points $r$ and $c$ on the grid. The deployment vector $\mathbf{u}$ is an $N_x N_y \times 1$ vector, with either 0 or 1 entries. The entry value indicates the number of sensors at the point on the grid that corresponds to that entry.

Note that the SE between achieved and required detection probabilities, can be described as the weighted quadratic norm of the state $(\mathbf{x}_k)$ of the system described in Eqn.(A.16). Assuming that the weighted quadratic norm of the system's state is chosen as the cost function and the deployment vector corresponds to a control vector, we are motivated to solve the deployment problem as an optimal control problem. Here, the objective is to determine the control vector that would minimize the cost function. That is, the deployment problem in Eqn.(A.12) can be restated as

$$\begin{aligned}\underset{\mathbf{u}_k}{\text{argmin}}\, J &= \frac{1}{2}\mathbf{x}_K^T\mathbf{Q}_f\mathbf{x}_K + \frac{1}{2}\sum_{k=1}^{K-1}(\mathbf{x}_k^T\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k) \\ \text{subject to } &\begin{cases}\mathbf{p}_{f,K} = \mathbf{p}_f^{req} \\ \mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{B}_k u_k,\ k = 1, \ldots, K \\ \mathbf{1}^T\mathbf{u} = K\end{cases}\end{aligned} \tag{A.18}$$

where, $\mathbf{Q}$,$\mathbf{Q}_f$ and $\mathbf{R}$ are symmetric positive definite weighing matrices. The squared error cost function penalizes both positive and negative deviations from the required detection

probability profile. To avoid incurring a penalty for satisfying/exceeding detection requirements, the error terms corresponding to a point where the detection requirement has been met/exceeded is set to zero in $J$. The optimal control problem corresponding to our system is the linear quadratic regulator (LQR) problem. We note here, that the cost function $J$ does not incorporate the false alarm requirements. However, false alarm requirements can be always met by choosing a suitable detection threshold at the FC. Since the LQR problem described above is dynamic, we can use either the sweep method or differentiation to calculate the optimal control vector.

## A.3 Suboptimal Deployment Algorithm

Our system equation is as follows

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k. \tag{A.19}$$

Ideally, it is desirable to deploy sensors such that the resulting $\mathbf{x}_k$ is equal to the zero vector (i.e., $\mathbf{x}_k = \mathbf{0}$ implies the detection requirements have been satisfied). Substituting $\mathbf{x}_k = \mathbf{0}$ in Eqn.(A.19) and solving for $\mathbf{u}_k$, we get the following ;

$$\mathbf{u}_k = -\mathbf{B}_k^{-1} \mathbf{x}_{k-1}. \tag{A.20}$$

Similar to the optimal control based algorithm, the resulting deployment vector consists of continuous real-valued elements. Therefore, a 1 is placed at the index where $\mathbf{u}_k$ is maximum and a 0 is placed at the remaining positions. Once again, a sensor is placed at the location corresponding to the index where $\mathbf{u}_k$ is maximum. The computational complexity of the suboptimal algorithm is $\mathcal{O}(K(N^3 + N^2))$, where $N$ is as defined earlier. Furthermore, simulation results show that the performance of this algorithm is comparable to that of the optimal control based algorithm.

## A.4 Simulation Results

In this section, we compare the performance of the greedy, suboptimal and optimal control based algorithms. In the greedy algorithm, a sensor is placed at the point with the largest difference between required and achieved detection probability.

In the first experiment, the area of interest is modeled as a grid of $25 \times 25$ points. The false alarm and detection probability are uniform and are set to $p_f^{req} = 0.01$ and $p_d^{req} = 0.9$, respectively. The noise variance is set to $\sigma^2 = 1$ and $d_0 = 1$. A discussion of the choice of $R_c$ can be found in [22]. Table A.1, lists the number of sensors needed by each algorithm, to meet the false alarm and detection requirements as the initial signal amplitude and detection radius are varied. The minimum numbers of sensors can be found by assuming, in the problem statement and LQR formulation, a large number of sensors $K$, and deploying sensors till detection and false alarm requirements are met. Results in Table A.1 indicate, that the optimal control based algorithm uses 25% fewer sensors than the greedy algorithm. This is due to the fact, that in the greedy algorithm a sensor is deployed by anticipating the effect the sensor deployment will have at a single point (i.e., the point with the largest difference between required and achieved detection probabilities). In contrast, in the proposed algorithms, the deployment process takes into account the effect of each sensor deployment on the whole grid, which is embedded in the matrix $\mathbf{B}$. Fig. A.1 shows the convergence of the SE between required and achieved detection probabilities as a function of the number of sensors deployed in the network by each algorithm, for the case of $A_0 = 100$ and $R_c = 5.26$ meters. We note that the suboptimal and optimal control based algorithms have a faster convergence rate than the greedy algorithm.

In the second experiment, we consider a setup similar to the first experiment with a fixed initial amplitude of $A_0 = 50$. However, the false alarm and detection requirements are not uniform over the grid (see Fig. A.2). In Table A.2, the number of sensors needed by each algorithm to satisfy the same $\mathbf{p}_d^{req}$ and $\mathbf{p}_f^{req}$ requirements is indicated. Results indicate that even with nonuniform requirements, the proposed algorithm uses fewer number of sensors

**Table A.1**: *Minimum number of sensors to satisfy uniform requirements for different $A_0$ and $R_c$: Greedy, Sub-optimal and Optimal-control based algorithms*

| Parameters | Greedy | Sub-optimal | Optimal-control based |
|---|---|---|---|
| $A_0 = 30, R_c = 2.9$ | 58 | 56 | 53 |
| $A_0 = 50, R_c = 3.72$ | 36 | 35 | 30 |
| $A_0 = 100, R_c = 5.26$ | 20 | 16 | 15 |
| $A_0 = 250, R_c = 8.3$ | 9 | 8 | 8 |

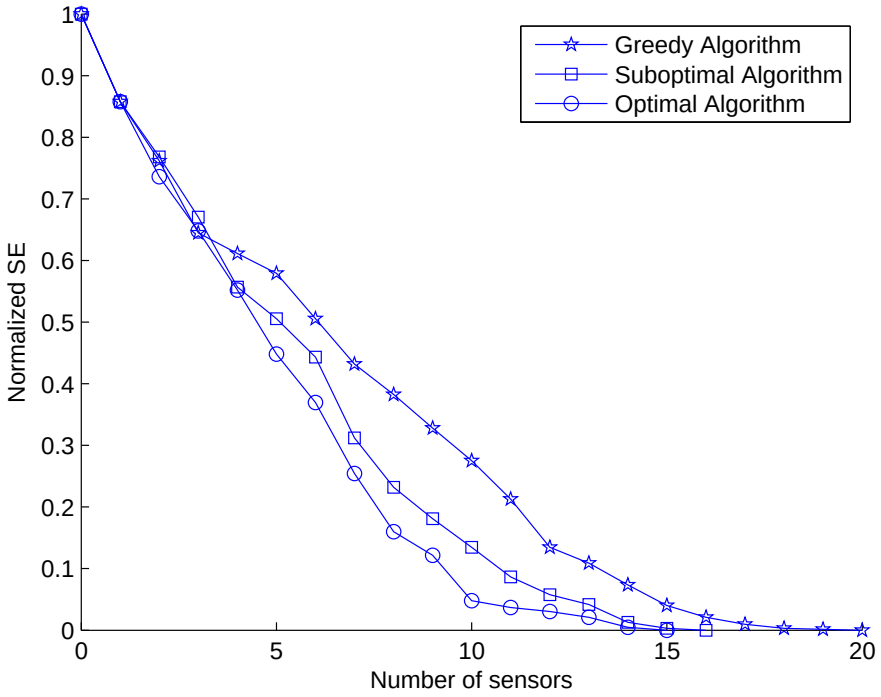



**Figure A.1**: *SE convergence*

than the greedy algorithm.

**Table A.2**: *Minimum number of sensors to satisfy various nonuniform requirements for fixed $A_0$ and $R_c$: Greedy, Sub-optimal and Optimal-control based algorithms*

| Requirements | Greedy | Suboptimal | Optimal-control based |
|---|---|---|---|
| $p_{d1} = 0.9, p_{d2} = 0.7, p_{f1} = 0.01, p_{f2} = 0.001$ | 35 | 33 | 31 |
| $p_{d1} = 0.9, p_{d2} = 0.7, p_{f1} = 0.001, p_{f2} = 0.01$ | 30 | 27 | 26 |
| $p_{d1} = 0.9, p_{d2} = 0.9, p_{f1} = 0.01, p_{f2} = 0.001$ | 40 | 37 | 34 |

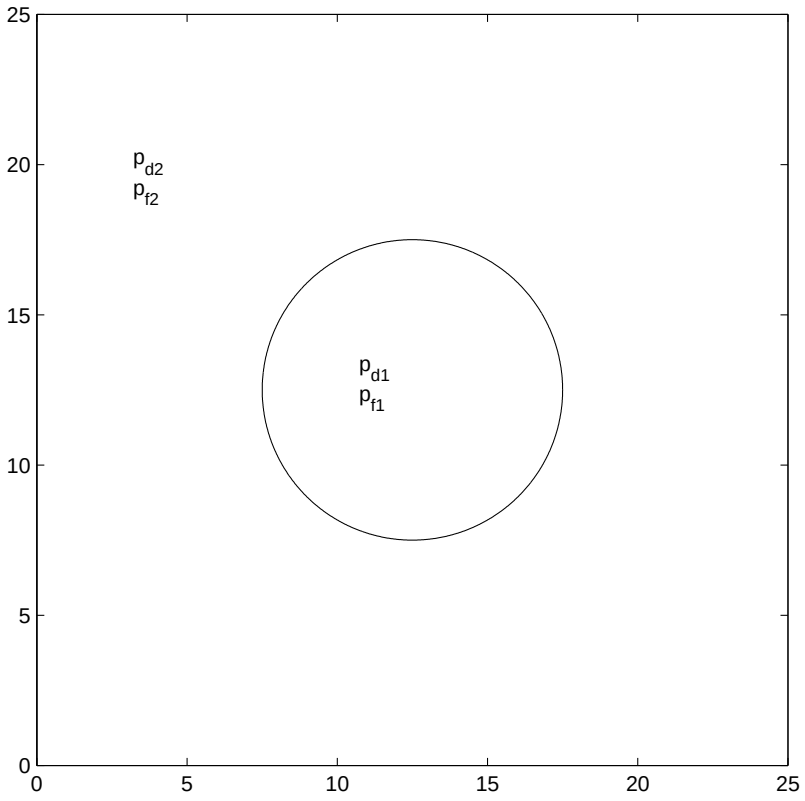



**Figure A.2**: *Nonuniform requirements* $N_x = N_y = 25$