New Optimization Models for Directional and Biological Wireless Sensor Networks

by

Yahya Osais, M.Sc., B.Sc.

A dissertation submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada, K1S 5B6

June 2010

Your file Votre référence
ISBN: 978-0-494-70558-2
Our file Notre référence
ISBN: 978-0-494-70558-2

The undersigned recommend to
The Faculty of Graduate Studies and Research
acceptance of the thesis

New Optimization Models for Directional and Biological Wireless Sensor Networks

submitted by
Yahya Osais, M.Sc., B.Sc.
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering

Thesis Supervisor
Dr. Fei Richard Yu

Thesis Supervisor
Dr. Marc St-Hilaire

External Examiner
Professor Alejandro Quintero
Department of Computer Engineering, École Polytechnique de Montréal

---

Chair, Department of Systems and Computer Engineering
Professor Howard Schwartz

Carleton University
June 2010

Abstract

Wireless Sensor Networks (WSNs) are embedded networks made up of tiny wireless devices referred to as sensors. They can be distinguished based on the type of sensing elements used in their sensors. Two important classes of WSNs are directional and biological WSNs. The former is given its name because the sensing region of a directional sensor is viewed as a sector in a two-dimensional plane. The latter, however, is given its name because its sensing elements are biological materials like enzymes.

One of the major challenges introduced by directional WSNs is the fact that the existing mathematical models developed for conventional WSNs cannot directly be used for solving the directional sensor placement and configuration problems. Therefore, new optimization models which capture the primary parameters characterizing directional sensors are necessary. On the other hand, a major challenge introduced by biological WSNs is the heat generated as a result of power consumption and wireless radiation. When biological WSNs are operated in temperature-sensitive environments like the human body, the temperature of the surrounding tissues might rise. The tissues might be damaged if the maximum safe temperature level is exceeded. Therefore, thermal management techniques are indispensible.

The first objective of the work presented in this dissertation is to develop optimization models for the planning of directional WSNs. Toward that end, the existing literature is critiqued and gaps are identified. Then, three fundamental planning problems are presented. The problems are formulated as Integer Linear Programming

(ILP) models. The wide applicability of the proposed models and their effectiveness are also discussed.

The second objective in this dissertation is to develop stochastic optimization models for the control of biological WSNs. The framework of Markov Decision Processes (MDPs) is used for building the optimization models. The solution of an MDP model is a policy which can be used to operate the biological WSNs in such a way that the maximum safe temperature level is never exceeded. When compared to existing policies, the policies produced by the proposed MDP models are superior in terms of network lifetime and temperature increase. Several techniques for handling large-size MDP models are also investigated.

Acknowledgments

First of all, I would like to thank God (ALLAH) for providing me with faith, guidance, strength and patience to complete this work. After that, I would like to thank my co-supervisors Dr. Fei Richard Yu and Dr. Marc St-Hilaire for their trust and support. Their guidance and extremely helpful feedback was critical in helping me formulate my research topics.

I would like also to thank my sponsor, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, for the financial support through the duration of my PhD study.

Members of the Saudi community in Ottawa also deserve a mention for their support. Especially, I would like to thank Dr. Fahad Al-Dosary, then a psychiatrist in the royal Ottawa mental health center, for his help and encouragement.

Last but not least, to my wife Mesbah and our children Renad and Retal goes my deepest gratitude for their patience and support. I have always felt guilty towards you. But, I promise I will make it up to you.

Contents

List of Figures

List of Tables

Acronyms

BTM	Biosensor Thermal Management
BWSN	Biological Wireless Sensor Network
DSP	Directional Sensor Placement
FOV	Field of View
ILP	Integer Linear Programming
MCSP	Minimum Cost Sensor Placement
MDP	Markov Decision Process
MSP	Minimum Sensor Placement
OSC	Optimal Sensor Configuration
RL	Reinforcement Learning
WSN	Wireless Sensor Network

Notation

Sets

Ω	Control points
Π	Placement sites for sensors
B	Locations for base stations
Ψ	Sensor types
Σ	Base station types
S	Sensing ranges
F	FOVs
D	Directions (or orientations) for a sensor
D_k	Directions (or orientations) for a type-k sensor
χ	Biosensors surgically implanted in the body of a patient
Υ_i	Biosensors which are neighbors to biosensor i

Decision Variables

x_{id}	A 0-1 variable such that $x_{id} = 1$ if and only if a sensor installed at location $i \in \Pi$ is oriented towards direction $d \in D$
x_{id}^k	A 0-1 variable such that $x_{id}^k = 1$ if and only if a type-k sensor is installed at site $i \in \Pi$ and oriented towards direction $d \in D_k$

x_i^{sfd}	A 0-1 variable such that $x_i^{sfd} = 1$ if and only if a sensor is installed at site $i \in \Pi$ and has a sensing range $s \in S$, a FOV $f \in F$ and is oriented toward direction $d \in D$
x_b^k	A 0-1 variable such that $x_b^k = 1$ if and only if a base station of type-k ($k \in \Sigma$) is installed at location $b \in B$
σ_{ij}^k	A 0-1 variable such that $\sigma_{ij}^k = 1$ if and only if control point $j \in \Omega$ is covered by a type-k sensor installed at site $i \in \Pi$
y_1^{il}	A 0-1 variable such that $y_1^{il} = 1$ if and only if the sensor at location $i \in \Pi$ is communicating with sensor at location $l \in \Pi/\{i\}$
y_2^{ib}	A 0-1 variable such that $y_2^{ib} = 1$ if and only if the sensor at location $i \in \Pi$ is communicating with the base station at location b
y_2^{ibk}	A 0-1 variable such that $y_2^{ibk} = 1$ if and only if a sensor at location $i \in \Pi$ can communicate with a type-k ($k \in \Psi$) base station at location $b \in B$
f_1^{il}	The flow from sensor node $i \in \Pi$ to sensor node $l \in \Pi/\{i\}$
f_2^{ib}	The flow from sensor node $i \in \Pi$ to the base station b

Constants

Δ	Adjacency matrix between placement sites where $\delta_{il} = 1$ if the Euclidean distance between two placement sites $i, l \in \Pi; i \neq l$ is less than or equal to the uniform transmission range of sensor nodes
Γ	The adjacency matrix between placement sites and the base station where $\gamma_{ib} = 1$ if the Euclidean distance between the base station b and a placement site $i \in \Pi$ is less than or equal to the uniform transmission range of sensor nodes

Γ_k	Adjacency matrix for a type-k base station where $\gamma_k^{ib} = 1$ if the Euclidean distance between a placement site $i \in \Pi$ and a type-k base station installed at location $b \in B$ is less than or equal to the reception range of the base station
Θ	Adjacency matrix between control points and a sensor node where $\theta_{ij}^d = 1$ if a sensor node installed at site $i \in \Pi$ covers control point $j \in \Omega$ when oriented towards direction $d \in D$
Θ_k	The adjacency matrix between control points and a type-k sensor where $\Theta_{ij}^{kd} = 1$ if a type-k sensor installed at site $i \in \Pi$ covers control point $j \in \Omega$ when oriented towards direction $d \in D_k$
Λ	Distance matrix $[d_{ij}]$ where d_{ij} is the distance between a placement site $i \in \Pi$ and a control point $j \in \Omega$
$C_{il}(C_{ib})$	Capacity of the wireless link between sensor node at site $i \in \Pi$ and sensor node at site $l \in \Pi/\{i\}$ (base station b)
C_{max}	Maximum amount of data that a sensor node can handle (transmit and/or receive) per time unit (i.e., node capacity)
R_i	Rate at which data is generated by a sensor placed at site $i \in \Pi$

Parameters

ϕ	FOV for all sensors
r	Sensing range for all sensors
r_t	Transmission range for all sensors
c_k	Cost of a type-k sensor
r_k	Sensing range of a type-k sensor
ϕ_k	FOV of a type-k sensor
r_s	Value of sensing range $s \in S$

ϕ_f	Value of FOV $f \in F$
c	Base cost of a sensor
c_s	Cost of one meter of sensing range
c_f	Cost of one degree of FOV
c_k	Cost of a type-k base station
C_k^{max}	Capacity of a type-k base station
r_k^r	Reception range of a type-k base station
$\mathcal{E}_0$	Initial energy
κ	Decrease in temperature of a biosensor
τ	Maximum safe temperature level

Chapter 1

Introduction

A Wireless Sensor Network (WSN) is typically made up of a large number of energy-constrained and possibly temperature-constrained sensor nodes and one or more sink nodes (also referred to as base stations). Sensor nodes (hereafter, referred to as sensors) are tiny devices that are equipped with multiple on-board sensing elements, a microcontroller and a wireless interface. They are deployed to study and monitor a variety of phenomena and environments at close proximity. They generate, process and forward data to sink nodes which are mainly responsible for data gathering and network management.

The list of possible applications of WSNs continues to grow as we learn more about them. The following are some examples:

1. Military and defense: battlefield surveillance and ground reconnaissance,
2. Environment: fire detection and precision agriculture,
3. Health: telemonitoring and patient tracking, and
4. Disaster recovery: search and rescue.

Regardless of the type of sensors used in the above applications, two fundamental issues always arise when using WSNs. These two issues are *coverage* and *manage-*

ment. Coverage is concerned with whether any location or object is not in the sensing region of at least one sensor. Management, on the other hand, is concerned with effectively controlling resources (like energy) and the effect of using them (like heat). The importance of these two issues varies between the classes of sensors and their applications. For example, in an application using video sensors for surveillance, energy management is much more important than the effect caused by energy consumption. However, in applications requiring the implantation of sensors into a patient's body, the heat resulting due to energy consumption is the first and most important concern.

In this work, two important classes of sensors are considered: (1) directional and (2) biological. The class of directional sensors contains all sensors whose sensing region is represented as a sector. In fact, this class includes all conventional sensors with a disk-like sensing region since a sector geometrically is a portion of a disk. Coverage is the most frequent issue arising in the applications of directional sensors.

On the other hand, the class of biological sensors (*biosensors*, for short) contains sensors whose sensing elements are biological materials such as enzymes and antibodies. Biosensors can be subclassified further as either directional or non-directional. Examples of directional biosensors include optical and tube-like biosensors. For this class of sensors, the heat generated due to power consumption is the most important issue, especially when operating in sensitive environments such as the human body.

This chapter is organized as follows. First, an overview of the research conducted in this dissertation is given. Then, the main contributions are summarized. Finally, the overall organization of the dissertation is described.

1.1 Overview

Directional and biological WSNs inherit all the technical challenges introduced by conventional WSNs. In addition, they introduce new ones that are unique to them.

For example, directional WSNs are expected to be capable of carrying video data which typically requires much higher bandwidth than that required by conventional WSNs whose main purpose is to observe and sample the surrounding environment. Also, the directionality of sensors requires a careful tuning of sensor parameters in order to achieve the performance objectives under a reasonable cost.

As for biological WSNs, however, a major challenge to realizing their full potential is the heat they generate as a result of power dissipation and wireless radiation. The effect of the generated heat is balanced by the human thermoregulatory system. However, if the generated heat is larger than what can be drained, the temperature of the tissues rises. The affected tissues might be damaged if the blood flow is not sufficient. As a consequence, the maximum safe temperature level which the tissues can withstand becomes a limiting factor in the operation of biological WSNs.

From the above, it is clear that the mathematical models developed for conventional WSNs cannot directly be used for optimizing directional and biological WSNs. This is because the applications and parameters involved are different. Hence, in this dissertation, two broad research themes are explored. They are the following:

1. Directional sensor placement and configuration and

2. Thermal management in biological WSNs.

The first research theme was chosen because of the little attention the area of directional WSNs planning has received. Thus far, most of the published works on directional WSNs focus on optimizing them after they are randomly deployed in a sensor field (e.g., see [1], [2] and [3]). Another reason is that deterministic placement might be favored over random placement. For example, there might be a limit on the number of sensor nodes which can be used in a mission due to a limited budget or payload[1]. Also, the required coverage and connectivity might not arise since some

[1]The current cost of an ordinary Crossbow sensor node consisting of the MICAz mote and

sensors might not survive the air drop. Besides, random placement can introduce nonefficient spending of energy.

The second research theme, on the other hand, was chosen because of the need for intelligent thermal management techniques in biological WSNs. Such techniques would, for example, enable long-term measurements to be performed and thus help in avoiding the prohibitive cost of continually visiting the doctor's office. It is also essential to understand how the performance of these techniques is affected by different design parameters like the maximum safe temperature level.

1.2 Contributions

In this section, the main problems addressed in this dissertation are presented and our technical contributions are summarized. Also, the details of our published and submitted works are given.

1.2.1 Directional Sensor Placement

Sensor placement (or deployment) is a network planning problem which arises out of the need to reduce deployment, installation and maintenance costs. In this problem, the number of necessary sensors, their types and (approximate) locations within a sensor field are determined so that the overall operating cost of the WSN is minimum while requirements such as coverage, lifetime and connectivity are satisfied. Sensor placement can be either deterministic or random. In deterministic placement, a network planner selects the sites where the sensors are to be placed. On the other hand, in random placement, sensors are scattered in the sensor field. For example, sensors may be dropped from an aircraft traveling over the sensor field.

MTS400 multi-sensor board is over US$300 [4]. In addition, if we take into consideration the cost of batteries and networking devices, the cost of deploying a large-scale ($>$ 3,000 nodes) WSN would easily exceed a million US dollars.

Planning of conventional WSNs has received excellent attention from the WSNs community. However, when a conventional WSN is equipped with directional sensors, network planning problems might not be treated in the same way as in conventional WSNs. Also, the algorithms designed for conventional WSNs behave differently when used in directional WSNs (for example, see [5] and [6]). In addition to that, the mathematical models developed for conventional WSNs might not directly be used for optimizing directional WSNs due to the different parameters involved.

In this part of the dissertation, two fundamental questions are addressed. They are as follows:

1. What is the minimum number of directional sensors necessary to achieve a full coverage and connectivity[2]?

2. What is the minimum possible cost if more than one directional sensor type is available?

Two optimization models are proposed to answer the above two questions. They are formulated as Integer Linear Programming (ILP) models. In the first model, there is only one type of directional sensors. The objective is to minimize the number of sensors by choosing the best subset of a given set of locations where sensors can be installed. This model turns out to be a generalization of sensor placement and grid coverage problems in conventional WSNs. In the second model, however, there is more than one type of sensors. The objective is to reduce the cost of sensors by appropriately choosing where to install a sensor and what sensor type to install at the selected site. Two refereed conference papers resulted from our work in this part:

1. Yahya Osais, Marc St-Hilaire and F. Richard Yu, "The Minimum Cost Sensor Placement Problem for Directional Wireless Sensor Networks", *In Proc. IEEE Vehicular Technology Conference (VTC)*, pp. 1-5, Sept. 2008.

[2]By full connectivity, it is meant that each sensor can deliver its data to the base station.

2. Yahya Osais, Marc St-Hilaire and F. Richard Yu, "On Sensor Placement for Directional Wireless Sensor Networks", *In Proc. IEEE International Conference on Communications (ICC)*, pp. 1-5, June. 2009.

1.2.2 Optimal Directional Sensor Configuration

A directional sensor has three tunable parameters: sensing range, field of view and direction. The ability to tune these parameters enables the WSN designer to meet application requirements like coverage and connectivity. A directional sensor is said to be optimally configured if the values assigned to its tunable parameters are optimal.

This part of the dissertation seeks to answer the question of what the optimal configuration of a directional sensor is before it is installed in the sensor field. This question is captured by a problem we refer to as the Optimal Sensor Configuration (OSC) problem. The solution to this problem is an ILP model that eases the parameter value selection process for directional sensors. In this model, the network cost is determined by the number of sensors as well as their configuration. The following publications resulted from our work in this part:

1. Yahya Osais, Marc St-Hilaire and F. Richard Yu, "On the Sensor Placement Problem in Directional Wireless Sensor Networks", *Planning and Optimisation of 3G and 4G Wireless Networks*, pp. 305-330, Johnson I. Agbinya, Editor, ISBN 978-87-92329-24-0, River Publishers, 2010.

2. Yahya Osais, Marc St-Hilaire and F. Richard Yu, "Directional Sensor Placement with Optimal Sensing Range, Field of View and Orientation", *Mobile Networks and Applications*, Vol. 15, No. 2, pp. 216-225, 2009.

3. Yahya Osais, Marc St-Hilaire and F. Richard Yu, "Directional Sensor Placement with Optimal Sensing Range, Field of View and Orientation", *In Proc. IEEE*

International Conference on Wireless and Mobile Computing, Networking and Communication (WIMOB), pp. 19-24, Oct. 2008.

1.2.3 Thermal Management Via Dynamic Sensor Scheduling

Biological WSNs will find many applications in human and animal healthcare. However, there are obstacles that must be overcome before their full potential can be realized. One such obstacle is that the heat generated by implanted biosensors may damage the tissues around them. Therefore, thermal management techniques are necessary. One such technique is based on the notion of dynamic sensor scheduling in which biosensors are dynamically scheduled to transmit their measurements. The scheduler takes into consideration the state of each biosensor and its wireless channel state information.

In this part of the dissertation, the dynamic sensor scheduling problem is formulated as a Markov Decision Process (MDP). Not like previous works, the state information of the wireless channel and the temperature increase in the tissues caused by the generated heat is incorporated into the model. The solution of the model gives an optimal policy that when executed, it will result in the maximum possible network lifetime under a constraint on the maximum temperature tolerable by the patient's body. The optimal policy is compared with two policies one of which is specifically designed for biological WSNs. The optimal policy turns out to be superior in terms of both network lifetime and temperature increase.

The proposed MDP model can be solved only for networks with a small number of biosensors. This is due to the large number of possible system states. Fortunately, multiple system states can be combined (or aggregated) if they are equivalent. In this way, the size of the MDP model can significantly be reduced. In this regard, two types of system states are identified and shown to result in a considerable reduction in model size when they are aggregated. The equivalence of the optimal policy produced by solving the reduced MDP model is also established. Two papers resulted from our work in this part:

1. Yahya Osais, F. Richard Yu and Marc St-Hilaire, "Dynamic Sensor Scheduling for Thermal Management in Biological Wireless Sensor Networks", *Submitted to the Journal of Wireless Networks*, Mar. 2009.

2. Yahya Osais, F. Richard Yu and Marc St-Hilaire, "Thermal Management of Biosensor Networks", *In Proc. IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 1-5, Jan. 2010.

1.2.4 Optimally Operating a Rechargeable Biosensor

Biosensors are powered by either rechargeable built-in batteries or by continuously sending electric energy in the form of electromagnetic waves. The use of batteries necessitates periodic recharging which can be performed using energy resulting from vibration, motion, light and heat. However, a more mature approach is to wirelessly collect energy from a radio frequency (RF) source and then convert it into usable power. This approach is widely used in the industry to transfer data and power to biosensors. It is also more practical since many sensors can be recharged simultaneously.

When implanted in a temperature-sensitive environment like the human body, a rechargeable biosensor must be operated with extreme caution at no more than the allowed maximum safe temperature level. This requirement gives rise to a stochastic control problem whose objective is to find a policy for operating the rechargeable biosensor in such a way that a performance criterion is maximized while the temperature in the surrounding region of the biosensor does not exceed the allowed maximum limit. This problem can be modeled as an MDP with three possible actions: *Sample*, *Recharge* and *Sleep*. The average number of samples is used as the performance criterion. The solution is an optimal policy that maximizes the average number of samples which can be generated by the biosensor while observing the constraint on the maximum safe temperature level.

Due to the exponential nature of the problem and the difficulty of describing the structure of the optimal policy, two heuristic policies are proposed. The first one is greedy and is used to provide insight into the design of any heuristic policy. On the other hand, the second one closely mimics the behavior of the optimal policy. In addition, it is shown how Q-learning, which is a form of reinforcement learning, can be used for learning the optimal policy. The average number of time slots needed to generate a sample is used as a measure to distinguish between the performance of the different policies. The following papers resulted from our work in this part:

1. Yahya Osais, F. Richard Yu and Marc St-Hilaire, "Optimal Management of Rechargeable Biosensors in Temperature-Sensitive Environments", *Submitted to the IEEE Transactions on Network and Service Management*, May 2010.

2. Yahya Osais, F. Richard Yu and Marc St-Hilaire, "Optimal Management of Rechargeable Biosensors in Temperature-Sensitive Environments", *To Appear in the Proceedings of the IEEE Vehicular Technology Conference*, Sept. 2010.

1.3 Organization

The rest of the dissertation is organized as follows. First, in Chapter 2, the literature related to both sensor placement and biosensors is reviewed. Second, in Chapter 3, the directional sensor placement problem is introduced along with the necessary background information. Then, in Chapters 4 and 5, the proposed optimization models for directional sensor placement and configuration are described, respectively. After that, in Chapter 6, the definition of the biosensor thermal management problem and necessary background information are given. Next, in Chapters 7 and 8, the problems of thermal management via dynamic sensor scheduling and optimally operating a rechargeable biosensor are discussed, respectively. Finally, in Chapter 9, conclusions are given and directions for further research are suggested.

Chapter 2

Related Work

This chapter provides a review of the literature related to our work. First, sensor coverage and the different notions of coverage are reviewed. Then, the works pertaining to the sensor placement problem are described. After that, the works concerned with biosensors and their effects are summarized. Finally, the notion of network lifetime is discussed.

2.1 Sensor Coverage

Coverage can be thought of as a measure of the quality of the sensing service provided by a WSN [7]. Different notions of coverage exist. However, the most used ones are area coverage, point coverage and barrier coverage. In area coverage problems, the objective is to cover (or monitor) a region. On the contrary, in point coverage problems, the objective is not to cover a whole region. It is rather to monitor some interesting locations within the region. Finally, in barrier coverage problems, the objective is to minimize the probability of undetected penetration through the barrier; i.e., sensor network. The notion of barrier coverage is inspired by Gage's classification [8].

Coverage was also studied in other fields such as computational geometry and robotics. The art gallery problem is a notable example in computational geometery. Its objective is to find the minimum number of observers necessary to cover an art gallery room such that each item is seen by at least one observer [9]. The art gallery problem is contrasted with the DSP problem in the next section.

As for the area of robotics, the notion of coverage was introduced by Gage for studying the coverage achievable by a many-robot system [8]. He defined three types of coverage: blanket coverage, barrier coverage and sweep coverage. In the blanket coverage, the goal is to achieve a static arrangement of robots that maximizes the total detection area. In barrier coverage, the goal is to achieve a static arrangement of robots that minimizes the probability of undetected penetration through the barrier. The sweep coverage is equivalent to a moving barrier.

The above notions of coverage are considered conservative since they define a location to be covered if it is within the sensing region of at least one sensor. This kind of coverage is referred to as physical coverage. Information coverage, on the other hand, is a relaxed notion of coverage. It was introduced in [10]. The motivation behind this new notion of coverage is the fact that for some applications like target detection, some parameters of the event of interest decay with distance (e.g., the sound generated by the movement of a tank). Therefore, even the sensor closest to the position of the event might not be able to detect the event. However, several sensors can cooperate to detect the event.

Information coverage is based on estimation theory where sensors cooperate to make an estimate of the data at a particular location. A location is considered covered by a set of sensors if an event occurring at that location can be estimated with a guaranteed estimation error. Clearly, with this definition of coverage, the concept of sensing region of a sensor is not applicable any more. This is what caused the introduction of the notion of occupation region in [11]. A point belongs to the occupation region of a sensor if it can be information-covered by the sensor and its neighbors.

Several network services and capabilities depend on the notion of coverage used in designing WSNs. For example, in [10], it was shown that significant savings in terms of sensor density for complete coverage can be achieved by using information coverage instead of physical coverage. Also, it was shown in [12] that the notion of information coverage helps in balancing between communication energy consumption and network coverage requirement.

2.2 Sensor Placement

In this section, three fundamental problems similar to the DSP problem are reviewed. They are the art gallery, camera placement and conventional sensor placement problems. It is argued that the DSP problem cannot be reduced to any one of them. Also, a brief description of some other relevant works involving directional sensors is given.

2.2.1 Guard Placement

The DSP problem is similar to the art gallery problem which is the theoretical study on how to place guards (or devices with vision capabilities) in an arbitrarily shaped polygon so as to cover the entire area. Chvatál [13] proved that for a simple polygon P with n sides, at most $\lfloor \frac{n}{3} \rfloor$ guards with a $360°$ FOV and an unlimited visibility range are needed such that every point of P is visible from at least one of them. Later, Fisk [14] showed that if P is triangulated, its vertices can be colored with three colors. Therefore, as a result, the number of guards is equal to the number of vertices colored with the least used color.

Algorithm 3 shows the general steps for solving the art gallery problem. It is based on Fisk's proof given in [14]. The solution assumes that each triangle in the triangulation of P (denoted by $\mathcal{T}(P)$) will have a guard at one of its vertices. Thus,

Algorithm 1 Solving the art gallery problem.

```
1: Triangulation
   $\mathcal{T}(P) \leftarrow$ Polygon $P$ is triangulated
2: 3-Coloring
   Each $v \in \mathcal{T}(P)$ is colored with a distinct color
3: Guard Placement
   Guards are placed at the vertices belonging to the class of the least used color
```

a guard placed on a vertex $v \in \mathcal{T}(P)$ can see at least each point inside the triangles whose corner points contain v. Of course, this is possible because there are no restrictions on the FOV and range of vision of the guards.

A notable application of the art gallery problem is in the area of antenna placement in wireless networks, such as GSM and UMTS. For example, in [15], the authors formulated the antenna placement problem as a 3D art gallery problem and provided a polynomial time approximate solution for it. The solution gives the number of antennas necessary to cover the whole service area and where they should be placed. The solution is obtained as follows. First, the set of control points visible to each antenna is computed. Then, graph coloring is used to make a first placement of antennas. Finally, a set covering model is used to reduce the number of antennas needed to cover the service area.

Clearly, the above results and algorithms cannot be applied to the DSP problem. This is because of the following reasons:

1. In the DSP problem, the sensing region (or visibility) of a directional sensor is restricted due to the constraints on sensing range and FOV.

2. The polygon formed by connecting the points corresponding to placement sites may not enclose all the control points in the sensor field. Further, it may self-intersect.

3. Sensors can be of different types (e.g., different ranges). In the art gallery problem, however, guards are assumed to be of the same type (i.e., similar

capabilities).

4. The location of sensors is restricted to a finite number of sites. Also, the number of points to be covered is finite.

2.2.2 Camera Placement

The DSP problem is also similar to the camera placement problem in computer vision where the goal is to determine the optimal number and location of cameras for a region to be monitored. Horster and Lienhart [16, 17] developed a binary integer programming model by discretizing the region into a grid. Each grid point represents a potential placement site for a camera. They used a simple visibility model where they described the range of vision of a camera by a triangle. The proposed model is focused on coverage with respect to a predefined sampling rate. It guarantees that an object in the monitored region will be imaged at a minimum resolution.

Inspired by the above work, Zhao and Cheung [18] developed an iterative grid based binary integer programming model for the visual tagging problem where the goal is to identify distinctive visual features of objects in two or more camera views. They also presented a comprehensive visibility model for computing the visibility of a single tag. The proposed model finds the minimum number of cameras, their poses and their locations in the monitored region in order to achieve a desired level of visibility.

We might not use the models developed for the camera placement problem to solve the DSP problem. However, we can still be inspired by them. The reasons that might prevent us include the following:

1. Cameras are mounted on walls and ceilings. Sensors, however, are placed on the ground. This requires new visibility models.

2. Sensors have a limited energy supply. Cameras do not have this problem since

they are typically powered from wall outlets. Thus, energy must be considered in optimization.

3. Sensors must be connected among themselves. Thus, connectivity is another element that must be considered in optimization.

4. Sensors can process data before it is delivered to the base station. This gives rise to issues such as routing and redundancy reduction.

2.2.3 Conventional Sensor Placement

Several models exist in the literature for the placement problem of conventional sensors. Chakrabarty et al. [19] were the first to study this problem on sensor fields comprising discrete points that are grid points. A 2D or 3D grid of points is to be covered. Sensors can be placed only at grid points. Each grid point is to be covered by at least α sensors, where $\alpha \geq 1$. There are $|\Upsilon|$ sensor types. A sensor of type $k \in \Upsilon$ costs c_k dollars and has a sensing range of r_k^s meters. Only one sensor can be placed at any grid point. They made an assumption such that all sensors can directly communicate with the base station. Therefore, the connectivity constraints were not considered. The objective is to find the least cost sensor placement which provides the required α-coverage. The problem is formulated as an ILP with $O(|\Upsilon|(|\Omega| + |\Pi|)^2)$ variables and $O(|\Upsilon|(|\Omega| + |\Pi|)^2)$ constraints, where $|\Omega|$ and $|\Pi|$ are the number of control points and placement sites, respectively. Control points and placement sites are combined into one set corresponding to the grid points.

Sahni and Xu [20] proposed another model that reduces the number of variables to $O(|\Upsilon|(|\Omega| + |\Pi|))$ and the number of constraints to $O(|\Omega| + |\Pi|)$. That is possible because the set of neighbors for each grid point is computed beforehand and given to the model as an input. The proposed model can be applied to any set of discrete points.

The above two models are practically solvable only for a small number of points. Wang and Zhong [21] compared the runtime of the two models. The models are implemented using *lpsolve* 5.5 [22]. It was observed that when the sensor field is bigger than 9×9 (i.e., 81 grid points), execution of the Chakrabarty et al.'s model is suspended with an out-of-memory error message. On the other hand, for a 20×20 sensor field, the runtime of Sahni and Xu's model for $\alpha = 1$ surges to 14 hours.

Sahni and Xu [23] proposed another ILP formulation for finding the minimum cost placement of sensors. The model handles the problem of placing sensors at a subset of preselected sites so as to minimize the sensor cost while providing a specified degree of coverage of the control points. The sets of placement sites and control points are merged together to form a single set. Each sensor is capable of directly communicating with the base station which is situated within the communication range of all sensors. The total number of variables is $O(|\Upsilon|(|\Omega|+|\Pi|))$ and constraints is $O((|\Upsilon| + m)(|\Omega| + |\Pi|))$, where m is the number of modalities. A modality is the quantity to be monitored, such as temperature, humidity and sound.

The above models might not be used to solve the DSP problem because they assume that sensors have a circular sensing range. Also, they only consider coverage. In our case, we consider coverage as well as connectivity. In addition, our model captures the limited visibility of sensors and their ability to change their orientation.

2.2.4 Other Relevant Works

There are some works that especially deal with directional sensor networks. For example, deterministic placement for connected coverage was discussed in [24]. The maximization of target coverage while minimizing the number of active sensors was studied in [1]. Area coverage was considered in [2]. The goal was to maximize the covered area by scheduling the directions of sensors. Prolonging the network lifetime was discussed in [3]. The proposed solution is to organize the directions of sensors into a group of non-disjoint cover sets. Each cover set covers all the targets in the sensor field. One cover set is activated at a time.

Other relevant works include node mobility for enhancing data collection and energy performance in WSNs. For example, applications like mapping [25, 26] and mine-removal [27] require mobile sensors that are capable of moving to accomplish their primary mission. Mobility may be random (e.g., using animals [28, 29]), predictable (e.g., public transport vehicle [30]) or controlled by the network (e.g., robotic sensors [31]).

Mobile sensor networks or robotic sensor networks have enormous potential for positive impact in our society. They can be employed for search-and-rescue operations in emergency situations [32, 33]. In environmental monitoring, they can be used to detect forest fires or monitor rare species [34]. In health care, they can monitor patients or the elderly over extended periods of time without confining them to a small area. A number of automation tasks, such as surveillance, monitoring energy consumption and quality inspection, can also benefit from robotic sensor network technology.

2.3 Biosensors

The research on the possible biological effects caused by biosensors and how to mitigate those effects is very recent. Most of the existing research deals with other technical issues such as energy efficiency and quality of service. In this section, the limited available literature is briefly reviewed.

The effect of leadership rotation in a cluster-based BWSN was studied in [35]. It was observed that rotating the role of which node collects measurements from other sensors and deliver them to the base station can significantly reduce the temperature increase in tissues due to wireless communication. The computation of an optimal rotation sequence involves using the Pennes's bioheat equation and the FDTD method to calculate the temperature increase due to a sequence. Because of its time requirement, the authors proposed another scheme to calculate the temperature increase. It is referred to as the Temperature Increase Potential (TIP). It efficiently estimates the temperature increase of a sequence. Using this scheme and a genetic algorithm, the authors were able to find the minimum temperature increase rotation sequence. However, they did not consider the effect of the wireless channel and limited energy.

The issue of routing in BWSNs was studied in [36]. The authors proposed a thermal-aware routing protocol that routes the data away from high temperature areas referred to as hot spots. The location of a biosensor becomes a hot spot if the temperature of the biosensor exceeds a predefined threshold. The proposed protocol achieves a better balance of temperature increase and shows the capability of load balance. The effect of the wireless channel and limited energy, on the other hand, is not incorporated into the operation of the protocol.

The sensor scheduling problem is considered in [37]. It is formulated as an MDP and the objective is to find an operating policy which maximizes the network lifetime. The state of a sensor is characterized by its current energy level only. An interesting observation is that since the total residual energy in the WSN decreases in each time slot, the value iteration algorithm used to solve the MDP model converges in one iteration. This is not guaranteed if the temperature is included in the model because when a sensor cools down (i.e., its temperature decreases), it transitions back to a less hot state.

Dynamic sensor activation in networks of rechargeable sensors is considered in [38]. The objective is to find an activation policy which maximizes the event detection probability under the constraint of slow rate of recharge of the sensor. The state of the system is characterized by the energy level of the sensor and whether or not an event would occur in the next time slot. The recharge event is random and recharges the sensor with a constant charge. The model does not include the state of the wireless channel which is very crucial when temperature is considered.

Body sensor networks [39] with energy harvesting capabilities are another kind of WSNs in which each sensor has an energy harvesting device that collects energy from ambient sources such as vibration, light and heat. In this way, the more costly recharging method which uses radiation is avoided. The interaction between the battery recharge process and transmission with different energy levels is studied in [40]. The proposed policies utilize the sensor's knowledge of its current energy level and the state of the processes governing the generation of data and battery recharge to select the appropriate transmission mode for a given state of the network.

The above works have motivated us to explore further the biological effects of BWSNs. As a result, we have noticed a lack of information on how to optimally operate an implanted BWSN when bounds such as the maximum safe temperature increase exist. Most of the existing works assume that energy is the only limiting factor in the operation of WSNs. However, this is not the case in BWSNs where the increase in temperature is a serious limiting factor. In addition, the effect of the wireless channel is not included explicitly in the current models used to compute optimal operating policies for BWSNs such as in [35, 36].

2.4 Network Lifetime

Different performance meterics have been used in the design of WSNs. Examples include mutual information between the a priori and the a posteriori target states [41], mean-squared error in state estimation [42], the difference between the desired and the estimated covariance matrix [43], sensor usage cost [44] and sum capacity [45]. Network lifetime was used as a performance measure in [37, 46–48]. In this dissertation, network lifetime is used as one of the performance measures for BWSNs.

The lifetime of a WSN is highly dependent on the rate at which sensors drain their energy. Sensors consume energy to perform sensing, process the sensed data and transmit the data to the base station. Radio communication is considered to be the single biggest source of energy consumption when compared to sensing and processing [49]. The radio interface uses energy when it transmits, receives or is idle. The energy required to transmit is usually the highest. According to the MICA2 sensor platform specifications [50], the current draw for the radio interface is as follows:

1. 27mA for transmission with maximum power,
2. 10mA for reception, and
3. $<$ 1uA for sleeping.

As a result, a significant research effort has been focused on designing energy efficient mechanisms at all protocol layers: physical, medium access control, routing and application. In order to prolong network lifetime, different strategies should be adopted at all network layers including hardware improvement at physical layer, energy efficient medium access control protocol, topology control and routing. These strategies include avoiding packet collision [51] and idle listening [52], limiting carrier-sense energy consumption [53], optimal routing path selection and content filtering or error tolerance [54, 55] and data aggregation at application layer [56].

Finally, it should be pointed out that the definition of lifetime depends on the underlying network application. One of the commonly used lifetime definitions is the number of data collections (i.e., time slots in which samples are generated by the network) until the number of dead sensors reaches a threshold [57]. We assume that the network lifetime terminates when a failure in data collection occurs, i.e., there is no active sensor in a data collection slot. In this case, two scenarios are possible. The first one is to recharge the sensors off-line and then re-start the network. The second scenario, however, is to allow the recharging of sensors during the normal operation of the network.

Chapter 3

The Directional Sensor Placement Problem

Several variations of the Directional Sensor Placement (DSP) problem could be envisioned by WSN designers. A basic but fundamental model of the problem is studied in this dissertation. Before delving into more details, however, the necessary background information is given in this chapter. Also, a description of the DSP problem as we envision it is stated.

3.1 Sensor Coverage Model

The behavior of a sensor is dictated by many parameters. However, only parameters that play a role in the issues relevant to the DSP problem are considered. The following parameters completely characterizes the sensing region of a sensor (see Figure 3.1):

1. (x_i,y_i): the Cartesian coordinates which denote the location of the sensor in a two-dimensional plane,

2. $\vec{v}_{id}$: a unit vector which cuts the sensing sector into half. This parameter defines

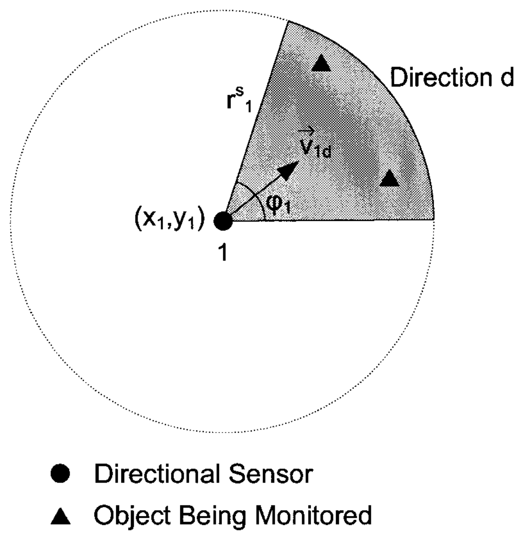


Figure 3.1: A directional sensor monitoring two objects.

the direction (or orientation) of the sensor,

3. ϕ_i: the maximum angle of sensing which can be achieved by the sensor. It is also called Field of View (FOV), and

4. r_i^s: the maximum sensing range of the sensor beyond which an object cannot be monitored.

The direction of a sensor is defined by the angle between the positive x-axis and its unit vector. The set of possible directions include the angles from 1° to 360°. The FOV of a sensor, however, is defined as the angle whose sides define the range of vision of the sensor. The set of possible FOVs include the set of all possible angles (i.e., 0°−360°).

A sensor is said to be directional if its FOV is less than 360°. The sensing region of a directional sensor is represented as a sector (see Figure 3.2(a)). On the other hand, a sensor whose FOV is intrinsically 360° is also a directional sensor but its sensing region is represented as a disk (see Figure 3.2(b)). In both figures, the shadowed area is the current sensing region of the sensor.

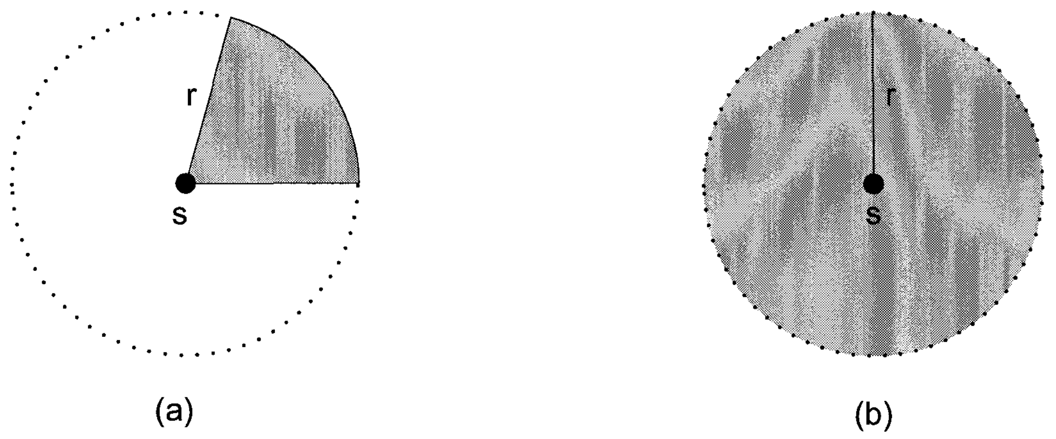


Figure 3.2: Two possible representations of the sensing region of a sensor (a) a sector and (b) a disk.

Video, ultrasonic and infrared sensors are all examples of directional sensors. Also, sensors with a 360° FOV might become directional in their sensing capability if their FOV is partially obstructed by physical objects or natural phenomena such as fog and snow. Surveillance and monitoring are examples of applications which require directional measurements in order to achieve their goals with a high degree of accuracy. For example, consider a standard Charge-Coupled Device (CCD) video camera. This camera is a directional sensor since its FOV is less than 360°. Thus, in order for a control point to be covered, the line segment connecting it and the camera must lie inside a *cone* oriented along the camera axis. The volume of the cone is determined by the sensing range and FOV of the camera.

Finally, it should be pointed out that in a real WSN design task, the sets of possible sensing ranges, FOVs and directions are finite and contain only discrete values. That is because the first two sets are dictated by the available sensors. The last set, in contrast, is dictated by the network designer who typically requires a small number of directions.

3.2 Test for Coverage

The four parameters described in the previous section determines the relationship between a sensor and object in the sensor field. Mathematically, an object at position j is covered by a sensor at position i if the following two conditions are true [1]:

$$\|\vec{d_{ij}}\|_2 \leq r_i^s \tag{3.1}$$

$$\vec{v}_{id} \cdot \vec{d_{ij}} \geq \|\vec{d_{ij}}\|_2 * \cos\left(\frac{\phi_i}{2}\right) \tag{3.2}$$

where $\vec{d_{ij}}$ is the distance vector from the sensor to the object.

The first condition checks if the object is within the sensing range of the sensor. The second condition checks if the distance vector is within the FOV of the sensor. This is done by performing the inner product operation with equality when the object is along one of the two edges of the sensing region of the sensor.

3.3 Problem Description

We consider the following scenario for the DSP problem. A directional WSN is to be deployed in a field. An engineer is assigned the task of studying the field and coming up with a floorplan for it. Figure 3.3 shows an example of a floorplan for a two-dimensional sensor field. The floorplan contains three kinds of sites (or points of interest). The first set of sites is referred to as *placement sites*. These are locations where sensors can be installed. Only one sensor can be installed at a placement site. The second set of sites represent *control points* where an event of interest might occur or a target might touch. These sites are monitored by the sensors. The last set of sites are used to host base stations which are used for collecting information from the sensors.

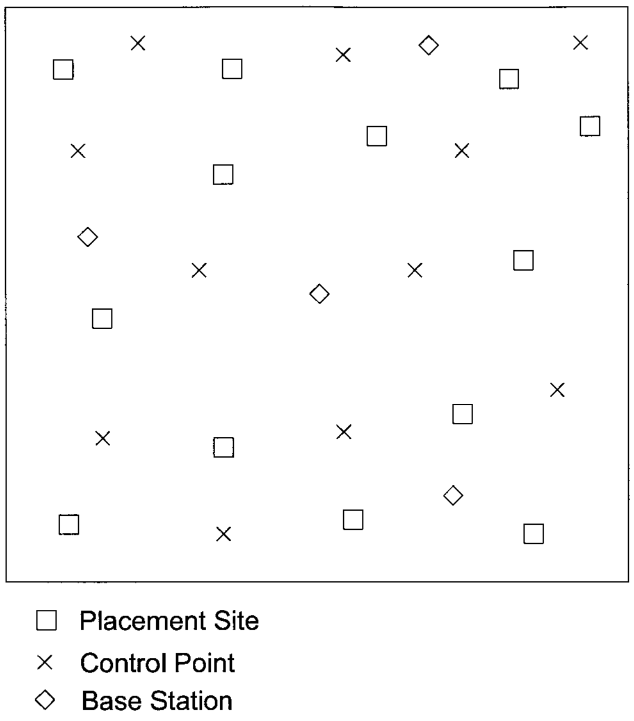


Figure 3.3: A typical floorplan for a 2D sensor field.

Given the floorplan of a sensor field, we would like to develop optimization models to answer the following questions:

1. What is the minimum number of directional sensors necessary to achieve a full coverage and connectivity?

2. What is the minimum possible cost if more than one directional sensor type is available?

3. What is the optimal configuration of a directional sensor?

Chapter 4

Reducing Number of Sensors and their Cost

This chapter presents two optimization problems which naturally arise when planning directional WSNs. In the first problem, the objective is to minimize the number of sensors. In the second problem, however, the objective is to minimize the cost of sensors. In both problems, there are constraints on the minimum required coverage of control points and connectivity of the resulting WSN. The two problems are formulated as ILP models.

4.1 Minimum Sensor Placement

The Minimum Sensor Placement (MSP) problem is a fundamental problem in which the goal is to find the minimum number of directional sensors such that every control point in the sensor field is covered by at least one sensor and the resulting network is connected. Consider, for example, Figure 4.1. There are five placement sites, five control points and one base station. Four sensors are required to cover the five control points. They are sensors 1, 2, 3, and 4. In addition, four wireless links are established to carry traffic from the sensor nodes to the base station. They are shown by the

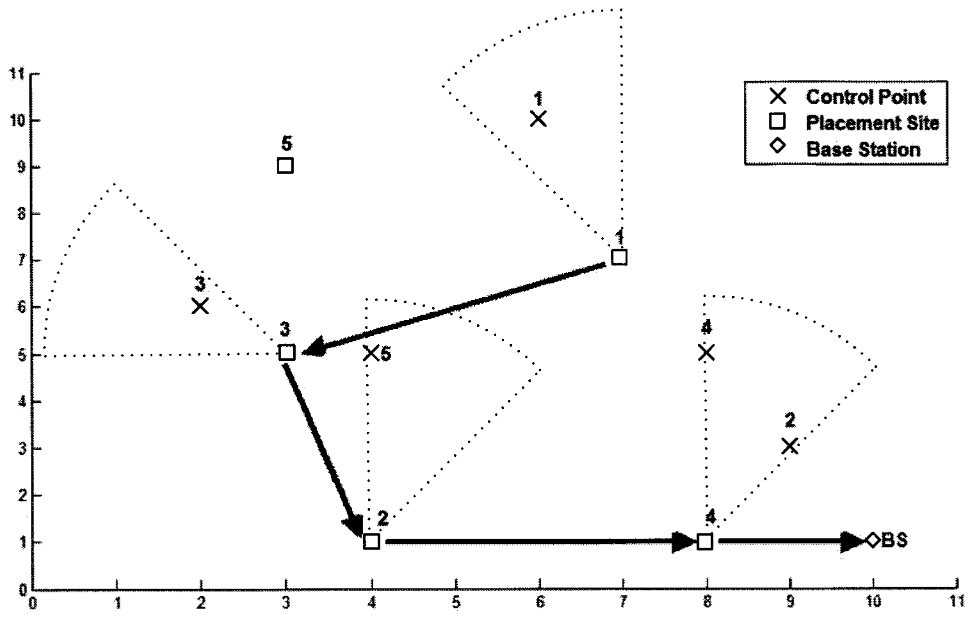


Figure 4.1: Four out of five sensors are necessary to cover the five control points and establish a connected network.

bold arrows in Figure 4.1.

4.1.1 Mathematical Formulation

The objective function is given by the following equation

$$F = min \sum_{\{i \in \Pi\}} \sum_{\{d \in D\}} x_{id} \tag{4.1}$$

which represents the number of sensors in the sensor field. The goal is to *minimize* this function while satisfying a set of constraints and requirements.

The first constraint is captured by the following inequality

$$\sum_{\{d \in D\}} x_{id} \leq 1 \qquad \forall \quad i \in \Pi \tag{4.2}$$

which states that a sensor can be oriented toward one direction only. The coverage requirement is represented by the following inequality

$$\sum_{\{i\in\Pi\}}\sum_{\{d\in D\}} x_{id}\cdot\theta_{ij}^{d}\geq 1 \qquad \forall \quad j\in\Omega \tag{4.3}$$

which states that a control point must be covered by at least one sensor. A control point is assigned to a sensor only if it can be covered by one of its possible directions.

A wireless link is established between two placement sites if two sensors are installed at those sites and they can communicate with each other. This is ensured by the following constraint.

$$y_1^{il}\leq\frac{\delta_{il}}{2}\left(\sum_{\{d\in D\}} x_{id}+\sum_{\{d\in D\}} x_{ld}\right) \qquad \forall \quad i,l\in\Pi; i\neq l \tag{4.4}$$

Similarly, a wireless link can be established between the base station and a placement site only if there is a sensor installed at that site and it can communicate with the base station. This is ensured by the following constraint.

$$y_2^{ib}\leq\gamma_{ib}\cdot\sum_{\{d\in D\}} x_{id} \qquad \forall \quad i\in\Pi \tag{4.5}$$

Flow constraints are introduced into the model to ensure the resulting network is connected. Since the base station does not generate any traffic, only flows between placement sites and between placement sites and the base station are considered. The following constraint is generated for every placement site.

$$R_i\sum_{\{d\in D\}} x_{id}+\sum_{\{l\in\Pi;l\neq i\}} f_1^{li}=\sum_{\{l\in\Pi;l\neq i\}} f_1^{il}+f_2^{ib} \qquad \forall \quad i\in\Pi \tag{4.6}$$

The following constraints are the capacity constraints. They ensure that the capacity of the wireless links and nodes is not exceeded.

$$f_1^{il} \leq C_{il} \cdot y_1^{il} \qquad \forall \quad i,l \in \Pi; i \neq l \tag{4.7}$$

$$f_2^{ib} \leq C_{ib} \cdot y_2^{ib} \qquad \forall \quad i \in \Pi \tag{4.8}$$

$$\sum_{\{l \in \Pi; l \neq i\}} f_1^{il} \leq C_{max} \cdot \sum_{\{d \in D\}} x_{id} \qquad \forall \quad i \in \Pi \tag{4.9}$$

The last two constraints respectively indicate that decision variables are binary and flow variables are positive integer.

$$x_{id}, y_1^{il}, y_2^{ib} \in \mathbf{B} \qquad \forall \quad j \in \Omega; d \in D; i,l \in \Pi; i \neq l \tag{4.10}$$

$$f_1^{il}, f_2^{ib} \in \mathbf{N}^+ \qquad \forall \quad i,l \in \Pi; i \neq l \tag{4.11}$$

The number of variables in the above ILP model is $O(|\Pi|(|D| + |\Pi|))$ and the number of constraints is $O(|\Pi|^2 + |\Omega|)$.

4.1.2 Numerical Results

In this section, we present preliminary results to assess the performance of the ILP model. The model is implemented using CPLEX 10.1.1 (see [58] for more details about CPLEX). For the computing platform, we used a Unix workstation equipped with a 3 GHz CPU and 1 GB memory.

First, since the proposed ILP model is a generalization of the grid coverage problem and the isotropic MSP, we used it to solve the grid coverage problem presented in [19], [20] and [21]. For this case, sensors are isotropic (i.e. the FOV is set to 360°) and only the coverage is considered. As a result, the grid ILP model is simply composed of the objective function (equation (1)) and the first and second constraints (inequalities (2) and (3)).

Table 4.1: Percentage of reduction and runtime for the grid coverage problem.

Grid Size	#Grid Points	#Sensors	%Reduction	Runtime (sec.)
9×9	81	12	85.12	0.55
10×10	100	14	86	1.57
11×11	121	16	86.78	1.83
12×12	144	18	87.5	6.13
13×13	169	20	88.16	26.5

Table 4.1 shows the results. The first column is the grid size. The second column is the number of grid points. This number is also the number of placement sites and control points in the sensor field. The next three columns respectively give the number of sensors needed to cover the control points, the percentage of reduction with respect to the initial number of placement sites and the runtime of the ILP model.

Secondly, we evaluate the effect of the number of placement sites on the runtime of the ILP model. We also evaluate the percentage of reduction in the number of sensors required to cover all the control points. There are 50 control points that need to be covered and one base station where all the traffic must be sent. Each sensor has four directions (i.e., FOV = 90 degrees), a transmission range of 10m and a sensing range of 6m. The node capacity of each sensor is 40 Kbps and the link capacity of each wireless link is 10 Kbps. The rate of data generation is 512 bps for each sensor. For each problem instance, the locations of control points, placement sites and base station were uniformly generated for a sensor field of size 20m $\times$ 20m. For each number of placement sites, five instances of the problem were randomly generated and the average was calculated.

Table 4.4 shows the results. The first column indicates the number of possible placement sites in the sensor field. The next three columns respectively represent the minimum number of sensors required in order to cover the 50 control points, the percentage of reduction with respect to the initial number of placement sites and the runtime of the model.

Table 4.2: Percentage of reduction achieved by the ILP model along with the runtime.

#PSs	#Sensors	%Reduction	Runtime (sec.)
30	14	53.33	0.31
40	14	65	1.36
50	13	74	3.77
60	13	78.33	3.97
70	12	82.86	8.58
80	12	85	17.13
90	12	86.67	23
100	12	88	50.18
150	11	92.67	99.12
200	11	94.5	503.61

As can be seen from Table 4.4, the execution time is increasing with respect to the problem size. Even with 200 potential placement sites, CPLEX only took, on average, 504 seconds. We can conclude that CPLEX is quite fast to solve large size instances of the problem. We also noticed that, from one instance to another, the execution time is variable. This can be explained by the fact that CPLEX is using the branch and bound algorithm. As far as the reduction in the number of sensors is concerned, we can see that as the number of potential placement sites increases, the reduction percentage also increases. Since more placement sites are available, better locations can be selected. As a result, one sensor node will be able to cover more control points, thus reducing the total number of sensors.

Next, the number of placement sites is fixed at 50 and the FOV is varied. Table 4.3 shows the results. The first column gives the size of the FOV in degrees. The second column gives the number of directions which a sensor can take for the corresponding FOVs. The last column gives the number of sensors necessary to cover the control points. For each FOV, five instances of the problem were randomly generated and the average was calculated.

Clearly, as the size of the FOV decreases, more sensors are needed to cover the same set of control points. For example, with FOV = $45°$, 20 sensors are needed,

Table 4.3: Number of sensors required for different FOVs.

FOV (degrees)	#Directions	#Sensors
360	1	5
180	2	9
120	3	12
90	4	12
72	5	13
60	6	17
45	8	20
36	10	19
20	18	26
12	30	25
5	72	30

compared to 5 when FOV = 360°. The time and memory required to solve the ILP model also increase when the size of the FOV decreases.

Finally, the relationship between the sensing range and number of sensors is studied. The FOV is fixed at 90°. Figure 4.2 shows that as the sensing range increases, less sensors are needed to cover the same number of control points. This behavior is as expected since a given sensor node can cover more control points.

4.2 Minimum Cost Sensor Placement

The MCSP is another fundamental problem in the planning of directional WSNs. In this problem, the goal is to minimize the overall cost of the WSN by appropriately selecting sensors from a set of different sensor types. Each sensor type is characterized by its cost, FOV and set of directions. Each control point must be covered by at least one sensor and the resulting network must be connected.

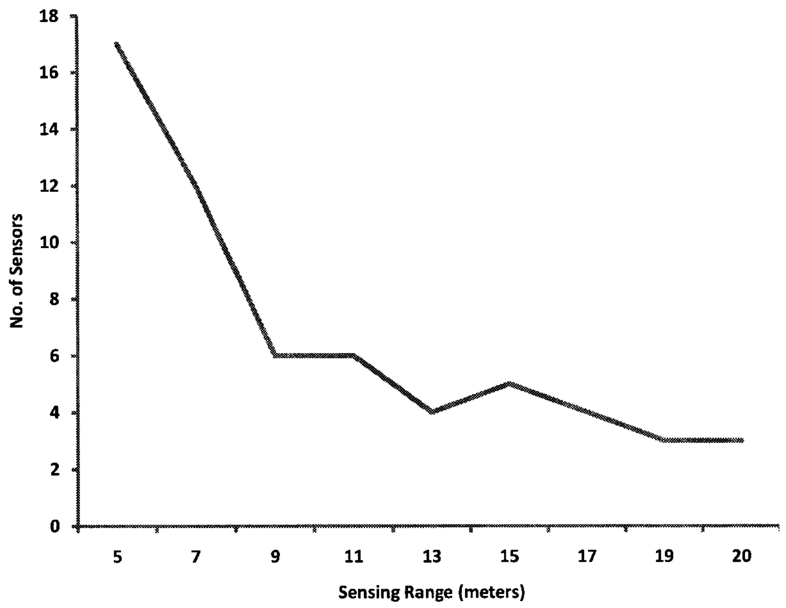


Figure 4.2: Number of sensors required for different sensing ranges.

4.2.1 Mathematical Formulation

The objective function is given by the following equation

$$F = \sum_{\{k\in\Psi\}} c_k \sum_{\{i\in\Pi\}} \sum_{\{d\in D_k\}} x_{id}^k \tag{4.12}$$

which represents the total cost of sensors. The goal is to *minimize* this function subject to a set of constraints and requirements.

A type-k sensor installed at a placement site can be oriented toward one direction only. This requirement is represented by the following inequality.

$$\sum_{\{k\in\Psi\}} \sum_{\{d\in D_k\}} x_{id}^k \leq 1 \qquad \forall \quad i \in \Pi \tag{4.13}$$

A control point can be assigned to a type-k sensor only if it can be covered by one of the directions of that sensor. This is represented by the following inequality.

$$\sigma_{ij}^k \leq \sum_{\{d \in D_k\}} x_{id}^k \cdot \Theta_{ij}^{kd} \qquad \forall \quad i \in \Pi; j \in \Omega; k \in \Psi \tag{4.14}$$

A control point must be covered by exactly one sensor. This requirement is given by the following equation.

$$\sum_{\{i \in \Pi\}} \sum_{\{k \in \Psi\}} \sigma_{ij}^k = 1 \qquad \forall \quad j \in \Omega \tag{4.15}$$

This equation can be changed to one that ensures each control point is covered by a minimum number of sensors.

The following eight constraints are similar to the last eight constraints in the formulation for the MSP problem.

$$y_1^{il} \leq \frac{\delta_{il}}{2} \left[\sum_{\{k \in \Psi\}} \sum_{\{d \in D_k\}} x_{id}^k + \sum_{\{k \in \Psi\}} \sum_{\{d \in D_k\}} x_{ld}^k \right] \qquad \forall \quad i, l \in \Pi; i \neq l \tag{4.16}$$

$$y_2^{ib} \leq \gamma_{ib} \sum_{\{k \in \Psi\}} \sum_{\{d \in D_k\}} x_{id}^k \qquad \forall \quad i \in \Pi \tag{4.17}$$

$$R_i \sum_{\{k \in \Psi\}} \sum_{\{d \in D_k\}} x_{id}^k + \sum_{\{l \in \Pi; l \neq i\}} f_1^{li} = \sum_{\{l \in \Pi; l \neq i\}} f_1^{il} + f_2^{ib} \qquad \forall \quad i \in \Pi \tag{4.18}$$

$$f_1^{il} \leq C_{il} \cdot y_{il}^1 \qquad \forall \quad i, l \in \Pi; i \neq l \tag{4.19}$$

$$f_2^{ib} \leq C_{ib} \cdot y_2^{ib} \qquad \forall \quad i \in \Pi \tag{4.20}$$

$$\sum_{\{l \in \Pi; l \neq i\}} f_1^{il} \leq C_{max} \cdot \sum_{\{k \in \Psi\}} \sum_{\{d \in D_k\}} x_{id}^k \qquad \forall \quad i \in \Pi \tag{4.21}$$

$$x_{id}^k, \sigma_{ij}^k, y_1^{il}, y_2^{ib} \in \mathbf{B} \qquad \forall \quad i, l \in \Pi; i \neq l; j \in \Omega; k \in \Psi; d \in D_k \tag{4.22}$$

$$f_1^{il}, f_2^{ib} \in \mathbf{N}^+ \qquad \forall \quad i, l \in \Pi; i \neq l \tag{4.23}$$

4.2.2 Numerical Results

The ILP model is implemented using CPLEX 10.1.1 (see [58] for more details about CPLEX). All of our experiments are performed on a UNIX workstation equipped with a 3 GHz CPU and 1 GB memory.

In the first set of experiments, we evaluate the effect of the number of placement sites on the runtime of the ILP model. We also evaluate the percentage of reduction in the number of sensors required to cover the control points. The experiments are setup as follows. Locations of control points, placement sites and sink node are uniformly generated for a sensor field of size 100m $\times$ 100m. There are two sensor types, 30 control points and one sink node where all the traffic must be sent. A type-1 sensor costs \$20 and has a 20m sensing range, 45° FOV and eight directions. A type-2 sensor costs \$40 and has a 50m sensing range, 60° FOV and six directions. The transmission range and node capacity of all sensor types respectively are 25m and 40 Kbps. The link capacity of each wireless link is 10 Kbps. The rate of data generation is 512 bps for each sensor.

Table 4.4 shows the results for the experiments. The first column indicates the number of possible placement sites in the sensor field. The number of sensors from each type is respectively shown in the second column for type-1 and third column for type-2. The fourth column gives the minimum total number of sensors required to cover the control points. The next three columns respectively represent the percentage of reduction with respect to the initial number of placement sites, total cost of sensors and runtime of the ILP model. For each problem size (i.e. number of placement sites), five instances of the problem were randomly generated.

As can be seen from Table 4.4, the execution time is increasing with respect to the problem size. Even with 200 potential placement sites, CPLEX only took 372 seconds. We can conclude that CPLEX is quite fast to solve large size instances of the problem. We also noticed that, from one instance to another, the execution time is variable. This can be explained by the fact that CPLEX is using the branch and bound algorithm. As far as the reduction in the number of sensors is concerned, we can see that as the number of potential placement sites increases, the reduction percentage generally increases. Since more placement sites are available, better locations can be selected. As a result, one sensor node will be able to cover more control points, thus reducing the total number of sensors.

To illustrate the viability of the ILP model, we show the details of the solution for a problem instance with 70 placement sites. Seven sensors are required to cover the 30 control points in the sensor field. Table 4.5 gives the placement sites where these sensors are to be installed. It also gives the type and direction for each sensor. Figure 5.1 shows the network layout.

Consider, for example, the sensor installed at placement site 54. It is of type 2. In this type, a sensor has a FOV of 60°. The direction vector of this sensor is at an angle of 150° with the positive x-axis. The sensing region of the sensor is delimited by a sector.

The solution for this problem instance contains seven wireless links which are established to carry traffic from the sensor nodes to the sink node. They are shown by the bold arrows in Figure 5.1. The routing scheme is 6 $\rightarrow$ 29 $\rightarrow$ 54 $\rightarrow$ 55 $\rightarrow$ 11 $\rightarrow$ 5 $\rightarrow$ 14 $\rightarrow$ SINK. The total traffic delivered to the sink node is 3584 bps.

For the second set of experiments, the number of placement sites is fixed at 50 and the number of control points is varied. For each number of control points, five instances of the problem were randomly generated and the average cost was calculated. We briefly discuss the results.

Our main observations are the following. First, as the number of control points increases, the sensor cost increases. This behavior is as expected since the directions of the existing sensors may not cover the newly added control points. Therefore, more sensors are added to the sensor field. Secondly, when the number of control points

Table 4.4: Percentage of reduction achieved by the ILP model along with the runtime.

#	#Sensors			%	Cost	Runtime
PSs	Type-1	Type-2	Total	Reduction	($)	(sec.)
40	1	5	6	85	220	4.26
50	3	5	8	84	260	4.93
60	3	5	8	86.67	260	11.71
70	2	4	6	91.43	200	20.43
80	1	6	7	91.25	260	33.12
90	1	5	6	93.33	220	49.31
100	1	5	6	94	220	71.72
150	6	6	12	92	240	209.61
200	3	5	8	96	260	372.17

Table 4.5: Placement sites, sensor types and directions of sensors used in the solution for the example problem instance with 70 placement sites.

Sensor	Placement Site	Type	Direction (degrees)
1	5	1	22.5
2	6	1	292.5
3	29	1	202.5
4	11	2	30
5	14	2	330
6	54	2	150
7	55	2	210

reaches a certain threshold (500 in our experiments), the number of placement sites becomes insufficient. That is, even if a sensor is installed at every placement site, the resulting set of directions would not be enough to cover all the control points. Therefore, more placement sites must be introduced into the floorplan of the sensor field.

4.3 Conclusions

In this chapter, two optimization models were proposed for minimizing the number of directional sensors and their cost. The solution produced by both models represent a WSN which is connected and covers all control points in the sensor field. The model

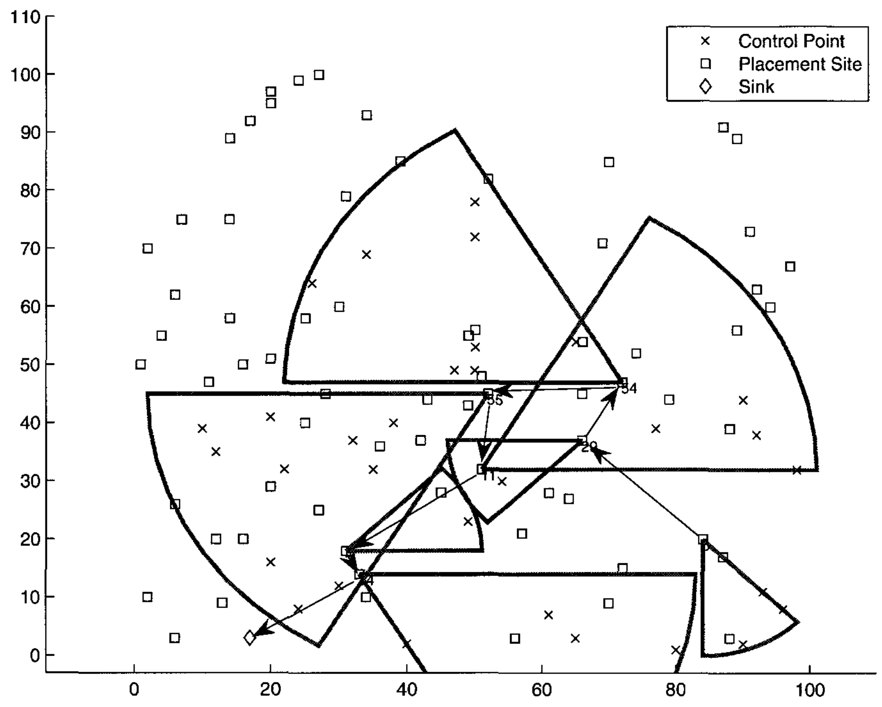


Figure 4.3: Network layout produced by the ILP model for the example problem instance in Table 4.5.

for the MSP problem is a generalization of the classical grid coverage problem and the coverage problem with isotropic sensors. The experimental results show that CPLEX is quite effective for solving large size instances of the MSP and MCSP problems.

Chapter 5

Optimal Sensor Configuration

5.1 Mathematical Formulation

The objective function of the OSC problem is given by the following equation:

$$F = min\{C_N + C_B\} \tag{5.1}$$

where

$$C_N = \sum_{\{s \in S\}} \sum_{\{f \in F\}} (c + c_s r_s + c_f \phi_f) \sum_{\{d \in D\}} \sum_{\{i \in \Pi\}} x_i^{sfd}$$

and

$$C_B = \sum_{\{k \in \Psi\}} c_k \sum_{\{b \in B\}} x_b^k.$$

The first term (C_N) is the cost of sensor nodes. The cost of a single directional sensor is determined by its sensing range and FOV and the sensor base cost. The second term (C_B), however, accounts for the cost of base stations which is determined by their number and type. The goal is to minimize this function subject to the following set of constraints and requirements.

A control point $j \in \Omega$ is covered by a sensor installed at a placement site $i \in \Pi$ if the following two constraints are satisfied.

$$\sum_{\{s\in S\}} r_s \sum_{\{f\in F\}} \sum_{\{d\in D\}} x_i^{sfd} \geq d_{ij} \sum_{\{d\in D\}} \sigma_{ij}^d \qquad \forall \quad i \in \Pi; j \in \Omega \tag{5.2}$$

$$\sum_{\{f\in F\}} \phi_f \sum_{\{s\in S\}} x_i^{sfd} \geq 2\cos^{-1}\left(\frac{\vec{v}_{id}\cdot\vec{d}_{ij}}{||\vec{d}_{ij}||_2}\right)\sigma_{ij}^d \qquad \forall \quad i \in \Pi; j \in \Omega; d \in D \tag{5.3}$$

Constraint (5.2) states that the sensing range assigned to the sensor must be greater than or equal to the Euclidean distance between the sensor and control point. Similarly, constraint (5.3) states that the FOV and direction assigned to the sensor must be such that the control point lies in the range of vision of the sensor.

Each control point must be covered by at least one sensor. This is ensured by the following two constraints.

$$\sum_{\{i\in\Pi\}} \sum_{\{d\in D\}} \sigma_{ij}^d \geq 1 \qquad \forall \quad j \in \Omega \tag{5.4}$$

$$\sigma_{ij}^d \leq \sum_{\{s\in S\}} \sum_{\{f\in F\}} x_i^{sfd} \qquad \forall \quad i \in \Pi; j \in \Omega; d \in D \tag{5.5}$$

The following two constraints respectively ensure that at most one sensor and one base station can be installed at any placement site and base station location.

$$\sum_{\{s\in S\}} \sum_{\{f\in F\}} \sum_{\{d\in D\}} x_i^{sfd} \leq 1 \qquad \forall \quad i \in \Pi \tag{5.6}$$

$$\sum_{\{k\in\Psi\}} x_b^k \leq 1 \qquad \forall \quad b \in B \tag{5.7}$$

A wireless link can be established between two placement sites if two sensors are installed at those sites and they can communicate with each other. This is ensured by the following constraint.

$$y_1^{il} \leq \frac{\delta_{il}}{2}\left(\sum_{\{s\in S\}}\sum_{\{f\in F\}}\sum_{\{d\in D\}} x_i^{sfd} + \sum_{\{s\in S\}}\sum_{\{f\in F\}}\sum_{\{d\in D\}} x_l^{sfd}\right) \qquad \forall \quad i,l \in \Pi; i \neq l \quad (5.8)$$

Similarly, a wireless link can be established between a placement site and a base station location if they contain a sensor and base station, respectively. Besides, the Euclidean distance between the two locations must be less than or equal to the reception range of the installed base station. These are ensured by the following constraint.

$$y_2^{ibk} \leq \frac{\gamma_k^{ib}}{2}\left(\sum_{\{s\in S\}}\sum_{\{f\in F\}}\sum_{\{d\in D\}} x_i^{sfd} + x_b^k\right) \qquad \forall \quad i \in \Pi; b \in B; k \in \Psi \qquad (5.9)$$

Flow constraints are introduced into the model to ensure that the resulting network is connected. Since the base station does not generate any traffic, only flows between placement sites and between placement sites and base station locations are considered. The following constraint is generated for every placement site.

$$R_i \sum_{\{s\in S\}}\sum_{\{f\in F\}}\sum_{\{d\in D\}} x_i^{sfd} + \sum_{\{l\in\Pi; l\neq i\}} f_1^{li} = \sum_{\{l\in\Pi; l\neq i\}} f_1^{il} + \sum_{\{b\in B\}} f_2^{ib} \qquad \forall \quad i \in \Pi \quad (5.10)$$

The following constraints are the capacity constraints. Constraints (5.11) and (5.12) ensure that the capacity of the wireless links is not exceeded. On the other hand, constraints (5.13) and (5.14) respectively ensure that the capacity of sensors and base stations is not exceeded.

$$f_1^{il} \leq C_{il} \cdot y_1^{il} \qquad \forall \quad i,l \in \Pi; i \neq l \qquad (5.11)$$

$$f_2^{ib} \leq C_{ib} \cdot \sum_{\{k\in\Psi\}} y_2^{ibk} \qquad \forall \quad i \in \Pi; b \in B \qquad (5.12)$$

$$\sum_{\{l\in\Pi;l\neq i\}} f_1^{il} + \sum_{\{b\in B\}} f_2^{ib} \leq C_{max} \cdot \sum_{\{s\in S\}} \sum_{\{f\in F\}} \sum_{\{d\in D\}} x_i^{sfd} \qquad \forall \quad i \in \Pi \tag{5.13}$$

$$\sum_{\{i\in\Pi\}} f_2^{ib} \leq \sum_{\{k\in\Psi\}} C_{max}^k \cdot x_b^k \qquad \forall \quad b \in B \tag{5.14}$$

The last two constraints respectively indicate that decision variables are binary and flow variables are positive integer.

$$x_i^{sfd}, x_b^k, \sigma_{ij}^d, y_1^{il}, y_2^{ibk} \in \mathbf{B} \qquad \forall \quad i,l \in \Pi; i \neq l; b \in B; k \in \Psi; \\ j \in \Omega; d \in D; s \in S; f \in F \tag{5.15}$$

$$f_1^{il}, f_2^{ib} \in \mathbf{N}^+ \qquad \forall \quad i,l \in \Pi; i \neq l; b \in B \tag{5.16}$$

The above formulation has $O(|\Pi||S||F||D|)$ variables and $O(|\Omega||\Pi||D|)$ constraints.

5.2 Numerical Results

In this section, we present numerical results to assess the performance of the ILP model which is implemented using CPLEX 10.1 [58]. For the computing platform, we used a Unix workstation equipped with a 3 GHz CPU and 1 GB memory. For each problem size (i.e. number of placement sites and number of control points), five instances of the problem were randomly generated and the average was calculated.

First, we study the effect of the number of placement sites and control points on the runtime of the ILP model and the percentage of reduction in the number of sensors required to cover the control points. The experiments are set up as follows.

Locations of control points, placement sites and base stations are uniformly generated for a sensor field of size 100m $\times$ 100m. Each sensor has a transmission range of 30m. The node capacity of each sensor is 10 Kbps and the link capacity of each wireless link is 5 Kbps. The rate of data generation is 1 Kbps for each sensor.

In order to speed up computation time, only two possible locations are available for the base stations. There are three types of base stations. They are characterized by their cost, capacity and coverage. They are as follows:

- Type 1 = {\$500, 10 Kbps, 25m},
- Type 2 = {\$1200, 15 Kbps, 50m}, and
- Type 3 = {\$2000, 20 Kbps, 80m}.

The sets of possible FOVs, directions and sensing ranges are as follows:

- $F = \{45°, 90°, 180°\}$,
- $D = \{90°, 180°, 270°, 360°\}$, and
- $S = \{5\text{m}, 10\text{m}, 15\text{m}, 20\text{m}\}$.

The base cost of a sensor is set to \$100. The costs of 1m of sensing range and 1 degree of FOV are set to \$0.1 and \$0.2, respectively.

Table 5.1 shows the results for the experiments. The number of control points is shown along the top of the table. The number of placement sites is shown on the left side. For each combination of placement sites and control points, the table shows the number of sensors needed to cover the control points, percentage of reduction with respect to the initial number of placement sites, cost of the sensor network and runtime of the ILP model.

As can be seen from Table 4.4, the execution time is increasing with respect to the problem size. This is as expected since the complexity of the problem grows with

Table 5.1: Percentage of reduction achieved by the ILP model along with the cost and runtime.

			No. of Control Points		
			30	50	70
No. of Placement Sites	20	#Sensors	9	9	8
		%Reduction	55	55	60
		Cost ($)	2031	2065	1974
		Runtime (sec.)	14	67	177
	40	#Sensors	12	14	17
		%Reduction	70	65	57.5
		Cost ($)	2481	2730	3083
		Runtime (sec.)	30	91	263
	60	#Sensors	11	14	16
		%Reduction	81.7	76.7	73.3
		Cost ($)	2389	2745	3015
		Runtime (sec.)	295	344	354
	80	#Sensors	11	13	15
		%Reduction	86.2	83.8	81.2
		Cost ($)	2420	2664	2911
		Runtime (sec.)	3678	3486	3340
	100	#Sensors	9	13	14
		%Reduction	91	87	86
		Cost ($)	1633	2670	2817
		Runtime (sec.)	5333	6731	8468
	200	#Sensors	11	12	13
		%Reduction	94.5	94	93.5
		Cost ($)	2380	2583	2704
		Runtime (sec.)	25476	26505	30826

the number of placement sites and the number of control points. Another interesting aspect is that the number of placement sites, the cost of the network and the reduction percentage are closely related. In fact, if the number of placement sites increases, the total cost of the WSN will generally decrease (up to a certain limit) and the reduction percentage will increase. This can be explained by the fact that since more placement sites are available, better locations can be selected. As a result, a given sensor node will be able to cover more control points, thus reducing the total number of sensors. On the other hand, increasing the number of control points leads to a slight increase in the number of sensors and thus cost.

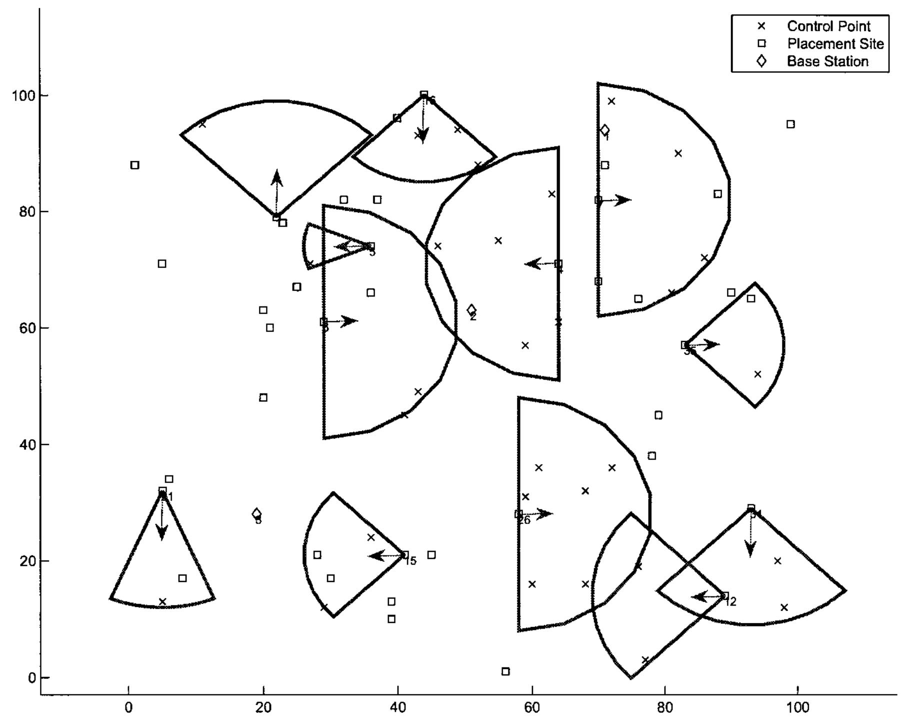


Figure 5.1: Network layout of the WSN produced by the ILP model for a 40 placement sites and 30 control points instance of the OSC problem.

To illustrate the viability of the ILP model, we show the details of the solution for a 40 placement sites and 30 control points instance of the OSC problem. Eleven sensors are required to cover all the control points in the sensor field. Two base stations of type one need to be installed at locations two and three. Figure 5.1 shows the network layout of the WSN produced by the model. The arrows indicate the directions of the sensors.

5.3 Conclusions

The total cost of directional sensor networks can be reduced by appropriately choosing base station type and adjusting sensor parameters. These parameters include the sensing range, field of view and direction. In this chapter, we have described the optimal sensor configuration problem and proposed an integer linear programming model for it. The experimental results show that using the proposed model, a significant reduction in the number of required sensors can be achieved. Also, they show that the runtime of the model increases with the number of placement sites and control points. The number of possible locations for base stations also adds to the runtime.

Chapter 6

The Biosensor Thermal Management Problem

Biosensors are tiny wireless devices which are attached or implanted into the body of a human or animal to monitor and control biological processes. Biosensors are energy- as well as temperature-constrained. Also, their sensing elements are made from biological materials. In this chapter, the Biosensor Thermal Management (BTM) problem is first described. Then, the background information necessary to understand the optimization models presented in the next chapters is given.

6.1 Problem Description

A biosensor typically contains four essential components (see Figure 6.1): biorecognition, transducer, radio and battery. The biorecognition system is made of elements such as enzymes and antibodies whose role is to produce a physio-chemical change which is detected and measured by the signal transducer. The transducer carries out signal processing tasks. The radio circuitry is responsible for wireless communication. The battery provides power for all active modules in the biosensor and is recharged using RF energy. During a recharging period, the biosensor uses its radio

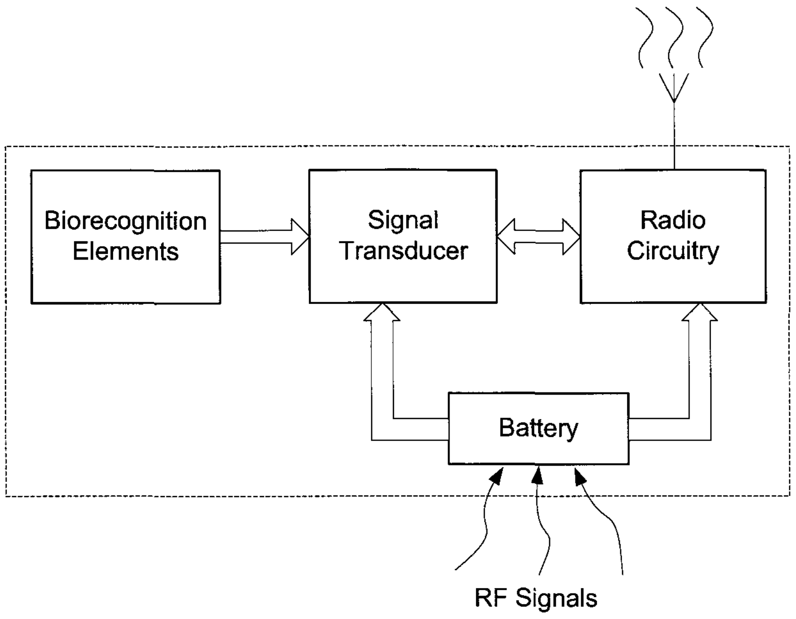


Figure 6.1: Components of a biosensor.

module to collect energy and recharge the battery. Therefore, while its battery is being recharged, the biosensor cannot perform sensing and communication.

The location of a biosensor implanted into the body of a human or animal represents a critical point since it experiences the maximum temperature increase. The tissues surrounding the biosensor might even be heated continuously due to the local radiation generated by the biosensor itself and the radiation generated by its neighbors. The biosensor becomes incapable of detecting and reporting events if the increase in its temperature exceeds the maximum safe temperature level. This condition causes a halt in system operation to allow the system to cool down. The BTM problem is concerned with finding optimal policies for operating biological WSNs in order to mitigate the thermal effect on the tissues surrounding the biosensors.

There is a gap in the current knowledge about biological WSNs. For example, most of the existing works on WSNs assume that energy is the only limiting factor in the operation of WSNs. However, this is not the case in biological WSNs where the increase in temperature is an additional serious limiting factor. In addition, the interplay between the initial energy a biosensor has, its temperature and the state of the wireless channel are not yet characterized.

6.2 Calculating Temperature Increase

RF signals used for wireless communication and recharging of implanted biosensors produce electrical and magnetic fields. When a human gets exposed to electromagnetic fields, the absorbed radiation gets converted into heat which manifests itself as a temperature increase inside the tissues. This phenomenon is balanced by the human thermoregulatory system. If the generated heat is larger than what can be drained, the temperature of the tissues will rise. The tissues might be damaged if the generated heat cannot be regulated by the blood circulation system.

The level of radiation absorbed by the human body when exposed to RF radiation is measured by the Specific Absorption Rate (SAR) which is expressed in units of W/Kg. SAR records the rate at which radiation energy is absorbed per unit mass of tissue [59]. The mathematical relationship between SAR and radiation is given by

$$SAR = \frac{\sigma|E|^2}{\rho}, \tag{6.1}$$

where E is the induced electric field due to radiation and ρ and σ are the density and electrical conductivity of the tissue, respectively. As an example to appreciate the importance of this measure, it was reported in [60] that an exposure to a SAR of 8 W/Kg in any gram of tissue in the head for 15 minutes may result in tissue damage.

SAR is a point quantity. That is, its value varies from one location to another. In this paper, we consider only the SAR in the near field (i.e., the space around the antenna of the biosensor). The near field radiation due to the antenna of the biosensor causes the heating of the tissue surrounding the biosensor. SAR in the near field is used to estimate this effect. The extent of the near field is given by $d_0 = \frac{\lambda}{2\pi}$, where

λ is the wavelength of the carrier signal used in wireless communication. SAR in the near field is given by the following equation [61]:

$$SAR_{NF} = \frac{\sigma\mu\omega}{\rho\sqrt{\sigma^2 + \epsilon^2\omega^2}}\left(\frac{Idl\sin\theta e^{-\alpha R}}{4\pi}\left(\frac{1}{R^2} + \frac{|\gamma|}{R}\right)\right)^2, \tag{6.2}$$

where μ and ϵ are the permeability and permittivity of the tissue, respectively. dl is the length of the wire representing the antenna, I is the current provided to the antenna, α is the attenuation constant, R is the distance from the biosensor to the observation point, θ is the angle between the observation point and the x-y plane, γ is the propagation constant and ω is the angular frequency. We assume that the radiation patterns are omnidirectional on the 2D plane and thus $\sin\theta = 1$.

The Pennes's bioheat equation [62] is the standard for calculating the temperature increase in the body due to heating. The general form of the equation is

$$\rho C_p \frac{dT}{dt} = K \bigtriangledown^2 T - b(T - T_b) + \rho SAR + P_c + Q_m, \tag{6.3}$$

where ρ is the mass density, C_p is the specific heat of the tissue, $\frac{dT}{dt}$ is the rate of temperature increase, K is the thermal conductivity of the tissue, b is the blood perfusion constant which indicates how fast the heat can be taken away by the blood flow inside the tissue and T_b is the temperature of the blood and the tissue. The terms on the right side indicate the heat accumulated inside the tissue. The terms $K \bigtriangledown^2 T$ and $b(T - T_b)$ are the heat transfer due to the thermal conduction and the blood perfusion, respectively. The terms ρSAR, P_c, and Q_m are the heat generated due to radiation, the power dissipation of circuitry, and the metabolic heating, respectively. In this equation, SAR includes both the SAR in the near field and the SAR in the far field.

The Finite-Difference Time-Domain (FDTD) method is a technique that transforms the bioheat equation to a discrete form with discrete time and space steps [63].

The area under consideration is divided into cells of side δ and the temperature is evaluated in a grid of points defined at the centers of the cells. Temperatures are computed at equally spaced time instants with a time step equal to δ_t. Therefore, from [35], the new bioheat equation is

$$\begin{aligned} T_{m+1}(i,j) &= \left[1 - \frac{\delta_t b}{\rho C_p} - \frac{4\delta_t K}{\rho C_p \delta^2}\right] T_m(i,j) \\ &+ \frac{\delta_t}{C_p} SAR_{NF} + \frac{\delta_t b}{\rho C_p} T_b + \frac{\delta}{\rho C_p} P_c \\ &+ \frac{\delta_t K}{\rho C_p \delta^2} \left[\begin{array}{c} T_m(i+1,j) + T_m(i,j+1) \\ +T_m(i-1,j) + T_m(i,j-1) \end{array} \right], \end{aligned} \tag{6.4}$$

where $T_{m+1}(i,j)$ is the temperature of cell (i,j) at time $m+1$, δ_t is the time step and δ is the space step.

Using (6.2) and (6.4), the temperature increase at the location of the biosensor $((i,j))$ can be found. It is assumed that the temperature of the surrounding cell points is the normal body temperature (i.e., $37°C$).

A simplified scheme to estimate the possible temperature increase was proposed in [35]. It is referred to as the Potential Temperature Increase (TIP). Basically, the temperature increase of any biosensor depends on two parameters:

1. The Euclidean distance from the currently transmitting biosensor and

2. The time elapsed since the last transmission made by the currently transmitting biosensor.

Clearly, the larger the values of these two parameters, the smaller the temperature increase is. On the other hand, a smaller Euclidean distance between biosensors i and j indicates that they have more influence on the temperature increase of each other. Also, if biosensor j follows biosensor i in the transmission schedule, then after being heated by by biosensor i, biosensor j becomes the transmitting node and further heats the tissues surroundint it.

The influence of the transmission made by biosensor i on the temperature increase of biosensor j (denoted by P_{ij}) is a function of the Euclidean distance between biosensors i and j (denoted by d_{ij}) and the distance in the transmission schedule between biosensors i and j (denoted by r_{ij}). r_{ij} is the number of transmissions which will occur before biosensor j makes its transmission. Thus, the mathematical expression for P_{ij} is the following:

$$P_{ij} = \frac{1}{c_1 + c_2\sqrt{d_{ij}} + c_3 r_{ij}^2 + c_4 r_{ij}^{2.5} + c_5 r_{ij}^3} \tag{6.5}$$

The values of the coefficients are as follows:

- c_1 = 14.8827
- c_2 = 8.8633
- c_3 = 3.1134
- c_4 = -1.5933
- c_5 = 0.2471

The joint influence on one biosensor is the sum of the individual influences from all other heating sources. This can be expressed mathematically as follows:

$$P_j = \sum_{i=1}^{N} P_{ij} \tag{6.6}$$

where N is the total number of biosensor.

It should be pointed out that eqn. (6.5) is just an approximation of the possible temperature increase. Thus, it cannot be used to calculate the exact temperature increase value.

6.3 Markov Decision Processes

An MDP is a model of a dynamic system whose behavior varies with time. The elements of an MDP model are the following [64]:

1. System states,
2. Possible actions at each system state,
3. A reward or cost associated with each possible state-action pair, and
4. Next state transition probabilities for each possible state-action pair.

The solution of an MDP model (referred to as a policy) gives a rule for choosing an action at each possible system state. If the policy chooses an action at time t depending only on the state of the system at time t, it is referred to as a stationary policy. An optimal stationary policy exists over the class of all policies if every stationary policy gives rise to an irreducible Markov chain. This means that one can limit the attention to the class of stationary policies.

An interesting class of MDPs is the class of MDPs with a terminating state. This state is reached with probability one in a finite number of steps. The number of steps represents the lifetime of the Markovian process induced by the MDP model (hence, the lifetime of the modeled system). The solution of the model is a policy which drives the system into the terminating state while optimizing an objective function which may include the lifetime of the system as a parameter.

In order to obtain a policy from an MDP model, it is necessary to form and solve the so called optimality equation (or Bellman equation). The following is the standard form of this equation with the maximization operator [65]:

$$V_n(s) = \max_{a \in A(s)} \Big[f(s,a) + \sum_{s' \in S} \mathbb{P}(s,s',a) V_{n-1}(s') \Big], \tag{6.7}$$

where n is the iteration index, S is the set of system states ($s \in S$), $A(s)$ is the set of actions possible when the system is at state s, $f(s,a)$ is the reward/cost per step, $\mathbb{P}$ is the system state transition probability matrix and $V(s)$ is the optimal value of the objective function when the system is started at state s and the optimal policy is followed.

Eqn. (8.3) can be solved using the classical policy iteration, value iteration and relative value iteration algorithms [65]. However, these algorithms become impractical when the number of system states is large. In such situations, one typically resorts to approximate techniques such as in [66–69]. Another solution for the problem of state explosion is *state aggregation* [70–74]. In this technique, using some notion of equivalence, equivalent states are combined into one class which is represented by a single state in the reduced MDP model. The reduced MDP model is equivalent to the original one but with significantly fewer states.

Reinforcement Learning (RL) techniques (like Q-learning) offers an alternative for obtaining the optimal policy at a significantly lower computational cost. Using a simulation model of the system under study, the decision maker in an MDP is viewed as a learning agent whose task is to learn the optimal action in each possible state of the system. As Figure 6.2 shows, the optimal policy is learned while the system is being driven (i.e., simulated) by the actions selected by the learning agent which stores the results of its actions in a knowledge base. The actions of the decision maker becomes better over time as new knowledge is obtained. Eventually, the RL algorithm converges to an optimal policy which can be used in the physical system.

6.4 Wireless Channel

The communication between the biosensor and base station occurs over a Rayleigh fading channel with additive Gaussian noise. Hence, the instantaneous received

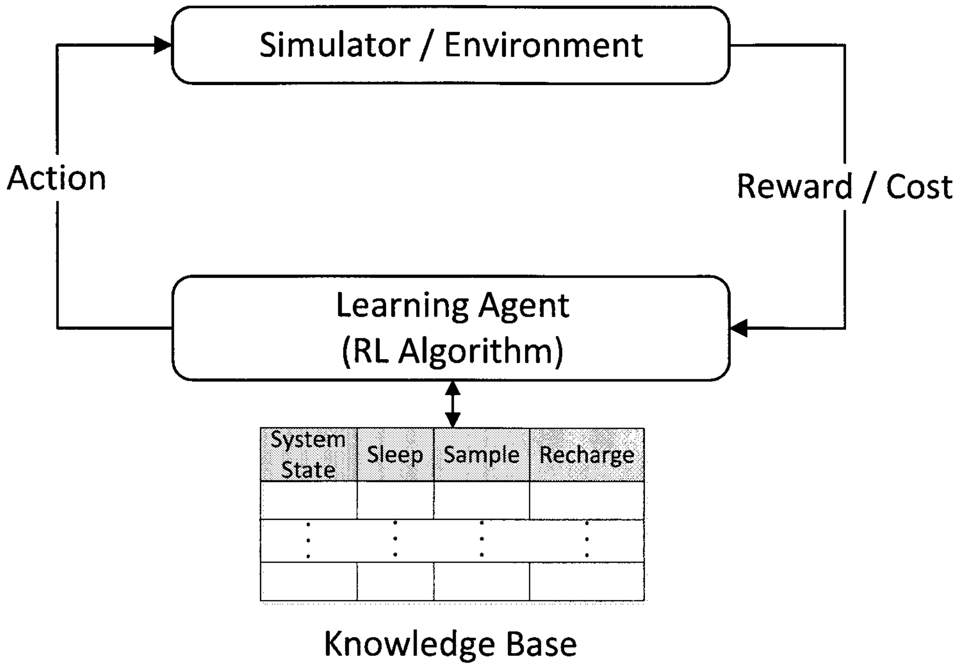


Figure 6.2: Using an RL algorithm (like Q-learning), the decision-making agent gradually learns the optimal policy. An action is applied to the system or simulated and then the resulting reward/cost is fed back into the knowledge base of the decision maker. The new knowledge obtained over time helps the decision maker to make better actions.

Signal-to-Noise Ratio (SNR) denoted by γ is exponentially distributed with the following probability density function [75]:

$$P(\gamma) = \frac{1}{\gamma_0} \exp\Big(- \frac{\gamma}{\gamma_0} \Big), \tag{6.8}$$

where γ_0 is the average received SNR.

Such a wireless channel can be modeled as a Finite-State Markov Chain (FSMC) [76, 77]. The model can be built as follows. For a wireless channel with K states, the state boundaries (i.e., SNR thresholds) are denoted by $\Gamma_1, \Gamma_2, ..., \Gamma_K, \Gamma_{K+1}$ where $\Gamma_1 = 0$ and $\Gamma_{K+1} = \infty$. The channel is said to be in state s_i if the SNR is between Γ_i and Γ_{i+1} where i = 1, 2, ..., K. It is assumed that the SNR remains the same during packet transmission and only transitions to the current or adjacent states are allowed.

The steady-state probability of the i^{th} state of the FSMC is given by

$$P(s_i) = \exp\Big(-\frac{\Gamma_i}{\gamma_0}\Big) - \exp\Big(-\frac{\Gamma_{i+1}}{\gamma_0}\Big) \tag{6.9}$$

and thus the state transition probabilities are

$$P(s_{i+1}|s_i) \approx \frac{N(\Gamma_{i+1})\Delta t}{P(s_i)} \qquad \text{and} \tag{6.10}$$

$$P(s_{i-1}|s_i) \approx \frac{N(\Gamma_i)\Delta t}{P(s_i)}, \tag{6.11}$$

where $N(\Gamma_i)$ is the average number of times per unit interval that the SNR crosses level Γ_i and Δt is the packet duration. $N(\Gamma_i)$ can be computed using the following equation [78]:

$$N(\Gamma_i) = \sqrt{2\pi\Gamma_i} f_d e^{-\Gamma_i}, \tag{6.12}$$

where f_d is the maximum Doppler frequency defined as $f_d = \frac{v}{\lambda}$ with v being the speed of the subject and λ being the wavelength.

The above channel model has been verified to be precise when the fading process is slow [76], such as in biosensor applications.

Chapter 7

Heat Management Via Dynamic Sensor Scheduling

7.1 System Model

Figure 7.1 shows a BWSN consisting of three biosensors implanted into the body of a patient. The biosensors communicate with an access point (or base station) over a wireless channel. The wireless access point initiates the data collection process by determining which biosensor is going to transmit the next measurement. A biosensor is selected for transmission based on the current network state and some policy. The wireless access point is assumed to know the global Channel State Information (CSI) of the wireless channel and the state of each biosensor at each point in time. Clearly, the location of a biosensor represents a critical point since it experiences the maximum temperature increase. This is because the tissues surrounding a biosensor might be heated continuously due to the local radiation generated by the biosensor itself and the radiation generated by its neighbors.

The setup in Figure 7.1 can mathematically be modeled as a discrete-state system which evolves in discrete time. Thus, the time axis is divided into slots of equal

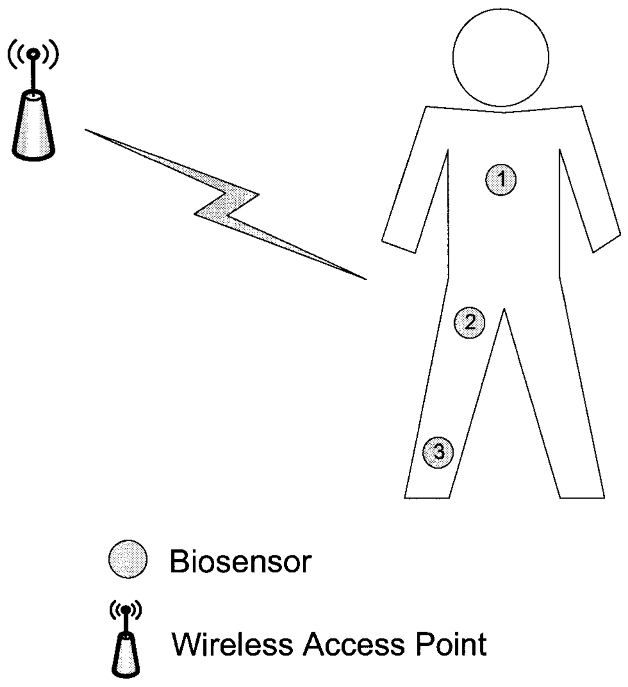


Figure 7.1: A patient with three biosensors implanted into his body.

duration ΔT and time $t \in \mathbb{Z}^+$ is the time interval $[t\Delta T, (t+1)\Delta T)$. The state of the system represents its condition at the beginning of a time slot. Control (i.e., which biosensor to choose next) can only be exercised at the beginning of a time slot and not at any other time during the slot. The current temperature and remaining energy of each biosensor and the CSI of the wireless channel are used to represent the state of the system in Figure 7.1. The number of biosensors, on the other hand, is used to represent the number of possible control actions that can be used to control the evolution of the system.

The system in Figure 7.1 works as follows. At the beginning of each time slot, a biosensor is selected by the base station to transmit its measurement. As a result, the energy and temperature of the selected biosensor change according to its transmission energy requirement which is determined by the state of the wireless channel. Also, the temperature of the neighbors of the selected biosensor increases based on the amount of energy used in the transmission. On the other hand, the temperature of the non-neighboring biosensors decreases. The change in the temperature of the biosensors can be calculated using the Pennes's bioheat equation and the FDTD method. However, due to the large simulation time required before the temperature change reaches a steady state, this approach is not followed here. Instead, the temperature decrease is assumed to be a constant reduction which occurs whenever the biosensor is not transmitting and not a neighbor of a transmitting biosensor. The temperature increase is also assumed to be directly proportional to the energy consumed by the transmitting biosensor.

Let χ be the set of biosensors which have been surgically implanted in the body of a patient and at known locations. Also, let Υ_i be the set of biosensors which are neighbors to biosensor i. Different criteria can be used to compute this set. In this work, the Euclidean distance between biosensors is used. Each biosensor $i \in \chi$ has a battery with an initial energy of $\mathcal{E}_0$ and can withstand a maximum temperature increase of τ units. The parameter τ represents the maximum safe temperature level that must not be exceeded.

In each time slot t, the state of a biosensor i is characterized by two variables which are the current temperature $T_t(i)$ and remaining energy $E_t(i)$. The energy required for a biosensor i to successfully transmit its measurement to the base station is determined by the state of the wireless channel in time slot t in which the biosensor is scheduled. This transmission energy is a random variable that is denoted by $W_t(i)$ and is independent and identically distributed over all sensors and time slots. Due to hardware and power limitations, $W_t(i)$ is discretely distributed over a finite set $\{\epsilon_1, \epsilon_2, ..., \epsilon_L\}$, where $0 < \epsilon_1 < \epsilon_2 < ... < \epsilon_L < \infty$ and ϵ_j is the energy consumed by a biosensor in transmitting its measurement at the j^{th} power level.

At the beginning of the next time slot, the energy level at each biosensor i is given by the following equation:

$$E_{t+1}(i) = \begin{cases} E_t(i) & if \quad i \neq a \\ E_t(i) - W_t(a) & if \quad i = a \end{cases} \tag{7.1}$$

where a is the index of the sensor selected for transmission. Similarly, at the beginning of the next time slot, the temperature of each biosensor i is given by the following equation:

$$T_{t+1}(i) = \begin{cases} \mathcal{F}(T_t(i), W_t(a)) & if \quad i = a \mid i \in \Upsilon_a \\ T_t(i) - \kappa & if \quad i \neq a \;\&\; i \notin \Upsilon_a \end{cases} \tag{7.2}$$

where $\mathcal{F}$ is a function of the transmission power and current temperature of the sensor scheduled for transmission and κ is the amount by which the temperature of a non-neighboring sensor decreases. The symbol | denotes the logical OR operator. It should be noted that the change in temperature experienced by the scheduled biosensor and its neighbors is assumed to be the same.

Finally, the communication between the biosensor and base station occurs over a Rayleigh fading channel with additive Gaussian noise. Such a wireless channel can be modeled as an FSMC. Thus, the transmission energy requirement for a biosensor i follows a Markov chain with L states and transition probabilities $P[W_{t+1}(i) = w' | W_t(i) = w]$, where $w, w' \in \{\epsilon_j\}_{j=1}^{L}$.

7.2 MDP Model

The purpose of the MDP formulation of the system described in the previous section is to find a policy π that prescribes the best action to take in each state of the system so as to maximize the long-term expected lifetime of the system which is defined as the number of accumulated time slots before the system enters a terminating state. The policy π is a stationary policy which means that it is independent of time and depends only on the state of the system. Next, we give the details of the MDP model.

7.2.1 State Set

The state of the system with $|\chi|$ biosensors at time t is described by a $(3 \times |\chi|)$-dimensional vector. That is,

$$s_t = \{(T_t(1), E_t(1), W_t(1)), (T_t(2), E_t(2), W_t(2)), ..., \\ (T_t(|\chi|), E_t(|\chi|), W_t(|\chi|))\} \tag{7.3}$$

Let S be the set of possible system states. Then, the number of possible system states is $|S| = |T|^{|\chi|} \times |E|^{|\chi|} \times |W|^{|\chi|}$, where $|T|$, $|E|$ and $|W|$ are the numbers of possible temperatures, residual energies and transmission energy levels, respectively.

The system enters a terminating state when any one of the following two conditions is true:

i) Temperature of any biosensor i is harmful (i.e., $T_t(i) > \tau$, where τ is an upper limit on the allowed temperature increase), and

ii) A biosensor i cannot transmit its measurement due to the lack of enough energy (i.e., $E_t(i) < W_t(i)$). This condition also accounts for the case when $E_t(i) = 0$.

Once the system is in a terminating state, the system must be halted to protect the patient. The system can then be restored to an initial state by recharging the biosensors and letting them cool down.

7.2.2 Action Set

At the beginning of each time slot, based on the current state of the system, the base station chooses an action (i.e., a biosensor to transmit its measurement). The set of possible actions consists of the indexes of all biosensors. In other words, the set of actions available in each state s which is not a terminating state is $A(s) = \{1, 2, ..., |\chi|\}$.

7.2.3 Reward Function

Let $R(s, a)$ be the instantaneous reward earned by the network due to action $a \in A(s)$ when the system is in a non-terminating state s. Since the goal is to maximize the expected network lifetime, the reward function can be defined as

$$R(s, a) = 1$$

which assigns a unit reward to each time slot as long as the network is in a non-terminating state. Therefore, the expected sum of rewards obtained before the network reaches a terminating state represents the network lifetime. It should be pointed out that the expectation is taken over all possible state sequences generated by a given policy.

7.2.4 Transition Probability Function

The behavior of the system is described by $|A|$ $|S| \times |S|$ transition probability matrices. Each matrix is denoted by $\mathbb{P}_{s_t, s_{t+1}}(a)$ which is the probability that choosing an action a when in state s_t will lead to state s_{t+1}. More formally, $\mathbb{P}_{s_t, s_{t+1}}(a)$ can be rewritten as follows:

$$\begin{aligned} \mathbb{P}[s_{t+1}|s_t, a = k] &= \prod_{i \in \chi} \Big\{ P[T_{t+1}(i)|T_t(i), W_t(i), a = k] \\ &\times \; P[E_{t+1}(i)|E_t(i), W_t(i), a = k] \\ &\times \; P[W_{t+1}(i)|W_t(i)] \Big\}, \end{aligned} \tag{7.4}$$

where k is the index of the biosensor selected for transmitting its measurement.

The above equation can be further simplified to:

$$\mathbb{P}[s_{t+1}|s_t, a = k] = \prod_{i \in \chi} \Big\{ P[W_{t+1}(i)|W_t(i)] \Big\}. \tag{7.5}$$

This is possible because the change in remaining energy and temperature happens with a probability of one for each biosensor. However, for each biosensor, the change in the state of the wireless channel is random. Thus, the next state transition probabilities for the system under study depends only on the state transition probabilities of the wireless channel.

7.2.5 Value Function

The thermal management problem is formulated as an infinite-horizon MDP using the average reward criterion [64]. So, let $V_\pi(s_0)$ be the expected network lifetime given that the policy π is used with an initial state s_0. Then, the maximum expected network lifetime $V^*(s_0)$ starting from state s_0 is given by

$$V^*(s_0) = \max_{\pi} V_\pi(s_0) \tag{7.6}$$

The optimal policy π^* is the one that achieves the maximum expected network lifetime at all non-terminating states. Hence, it gives the optimal sensor transmission schedule.

The Relative Value Iteration (RVI) algorithm [65] is used to numerically solve the following recursive equation for $n > 0$:

$$V_n(s) = \max_{a \in A(s)} \Big[R(s,a) + \sum_{s_{t+1} \in S} \mathbb{P}(s_t, s_{t+1}, a) V_{n-1}(s_{t+1}) \Big] \tag{7.7}$$

In (8.8), the subscript n denotes the iteration index. As $n \to \infty$, $V_n \to V^*$.

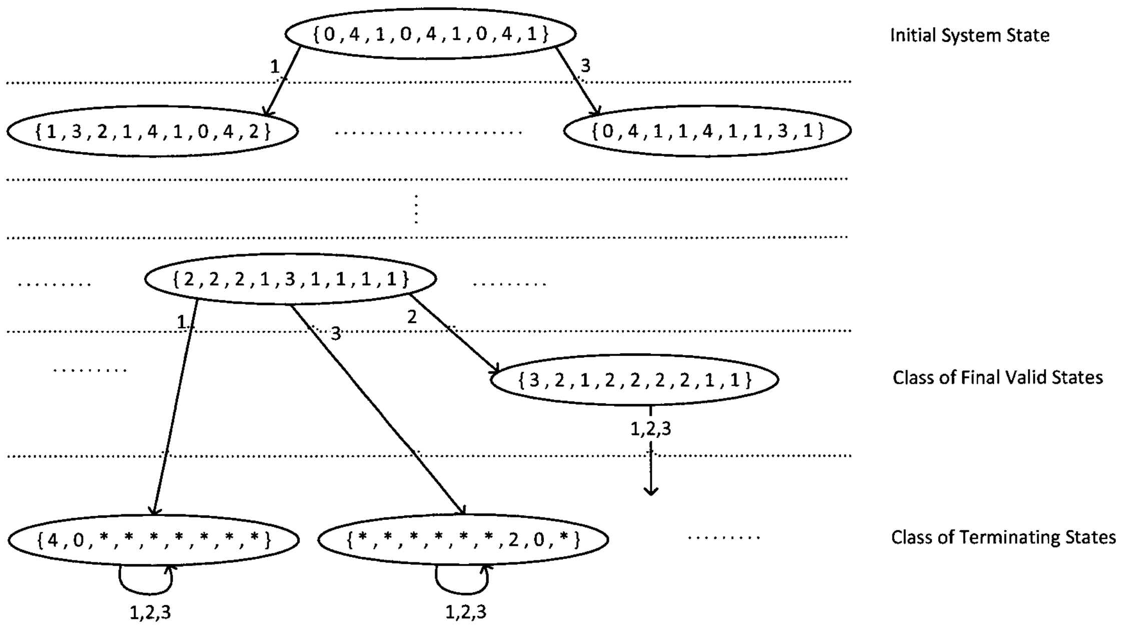


Figure 7.2: Excerpt of the system state space showing three classes of states.

7.3 Minimizing the Size of the MDP Model Through State Aggregation

The large state space of the MDP model makes the computation of the optimal policy a highly intensive process and thus only feasible for small-scale networks. This is due to the storage and runtime requirements which are both function of the number of possible system states. State aggregation can be used to mitigate this problem. With this technique, the state space is partitioned and the states belonging to the same partition are aggregated into a single new state. Partitioning is performed by using some notion of equivalence between system states. The final result is a reduced MDP model with the same properties as the original one but significantly fewer states.

In this work, the definition of state equivalence in MDPs introduced in [71] is utilized. This definition can be stated as follows.

Definition 1. *(State Equivalence [71])*

Two states are equivalent if and only if for every action:

1. They achieve the same immediate reward, and

2. They transit to the same next states with the same transition probabilities.

For example, consider Figure 7.2 which shows an excerpt of the state space of an instance of the MDP model of the system in Figure 7.1. In this case, τ and $\mathcal{E}_0$ are both 4. The state space has a tree-like structure in which the root is the initial state and the leaves are the terminating states. Two important classes of states are the class of *terminating* states and the class of *final valid*[1] states. In the former, the states are equivalent since for each action, no reward is generated and the next state is the same as the present one with a probability of one. This class of states can be identified in $O(|S|)$ time. Similarly, in the latter, the states are equivalent since for each action, a reward of one unit is generated and the next state is a terminating state with probability one. This class of states can be identified in $O(|S||\chi|)$ time. Additional classes of states can be identified but they are costly to compute in practice. Therefore, we consider only the classes of final valid states and terminating states since they are not costly to compute and provide a considerable reduction in the size of the MDP model.

The following theorem asserts that system states identified as final valid (terminating) are equivalent and thus can be represented by a single final valid (terminating) state in the reduced MDP model.

Theorem 1. *The system states in the class of final valid states (terminating states) are equivalent.*

Proof. We provide the proof for any two system states belonging to the class of final valid states. The proof for any two states belonging to the class of terminating states is the same.

[1]The name is just a convention to indicate that the final working state of the system before entering a terminating state always belongs to this class of states.

By definition, a valid system state is a non-terminating state. That is, it is a state at which each biosensor can make a transmission (i.e., all actions are possible). Also, by definition, a final valid system state is one at which the execution of an action generates a reward of one unit and causes the system to enter a terminating state. Since all terminating states are equivalent, the system transits to a terminating state with a probability of one. □

The equivalence of the optimal policy produced by solving the reduced MDP model is established by the following theorem.

Theorem 2. *The reduced MDP model produced by aggregating the final valid states and terminating states induces an optimal policy for the original MDP model.*

Proof. Let S^* be the new reduced set of system states. Also, let i and j be two equivalent system states such that $i \in S$ and $j \in S^*$. Using mathematical induction, it can be shown that i and j have the same optimal value. First, we start with the base case where $n = 0$ and $V_0(k) = 0 \;\; \forall \; k \in S^*$. In this case, the optimal value for any state is just the reward for that state; i.e., $V_1(k) = \max_{a \in A(k)} R(k, a)$. Since states i and j are equivalent, it is implied that $R(i, a) = R(j, a) \; \forall \; a \in A$ and thus $V_1(i) = V_1(j)$. This proves the base case.

For the inductive case (i.e., $n \geq 2$), using the induction hypothesis, the following can be shown for states i and j.

$$\begin{aligned} V_n(j) = \;& \max\nolimits_{a \in A(j)} \left[R(j,a) + \textstyle\sum_{k \in S^*} \mathbb{P}(j,k,a) V_{n-1}(k)\right] \\ = \;& \max\nolimits_{a \in A(i)} \left[R(i,a) + \textstyle\sum_{k \in S^*} \mathbb{P}(i,k,a) V_{n-1}(k)\right] \\ = \;& \max\nolimits_{a \in A(i)} \left[R(i,a) + \textstyle\sum_{l \in S} \mathbb{P}(i,l,a) V_{n-1}(l)\right] \\ = \;& V_n(i) \end{aligned}$$

This proves the inductive case. Therefore, it can now be established that any optimal action for state $j \in S^*$ is also an optimal action for state $i \in S$. □

Table 7.1: Reduction in the number of system states when terminating states and final valid states are aggregated. τ and L are 7 and 2, respectively.

$\mathcal{E}_0$	Total No. of States	Reduced No. of States	Percentage of Reduction
5	884736	184319	79.17
6	1404928	341802	75.67
7	2097152	569849	72.83
8	2985984	881510	70.48
9	4096000	1289835	68.51
10	5451776	1807874	66.84

Table 7.1 shows the percentage reduction obtained by varying $\mathcal{E}_0$ while fixing τ and L at 7 and 2, respectively. This considerable reduction is achieved just by aggregating the final valid and terminating states. Clearly, most of the system states fall into these two classes of system states. This can be attributed to the fact that the state space of the MDP model has a tree-like structure in which the number of leaf nodes representing terminating states is substantially large. The next substantially large number is the number of final valid states.

7.4 Numerical and Simulation Results

In this section, numerical and simulation results are presented in the context of an illustrative example. First, the example is described and solved numerically. Second, the example BWSN is simulated to compare the performance of the optimal policy obtained by the MDP model to that of TIP-based and most residual energy policies.

7.4.1 Illustrative Example

The numerical and simulation results are obtained by using the following example. Consider again the BWSN shown in Figure 7.1. The biosensors are indexed from one to three. The neighbors of each biosensor are as follows:

- $\Upsilon_1 = \{2\}$
- $\Upsilon_2 = \{1, 3\}$
- $\Upsilon_3 = \{2\}$

Also, the $\mathcal{F}$ function in (8.2) is defined for each biosensor i as

$$\mathcal{F}(T_t(i), W_t(a)) = T_t(i) + W_t(a).$$

The channel for each biosensor is modeled as a two-state Markov chain whose state boundary is randomly generated. A biosensor requires k units of energy to successfully transmit its measurement when its channel is in state $k \in \{1, 2\}$. The transition probability matrix is the following.

$$\begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{bmatrix}$$

The MDP model of the biosensor network is solved using the RVI algorithm. The initial state of the network is assumed to be $\{(0,\ \mathcal{E}_0,\ 1),\ (0,\ \mathcal{E}_0,\ 1),\ (0,\ \mathcal{E}_0,\ 1)\}$. The expected network lifetime is the value calculated by the RVI algorithm for the initial state.

Figure 7.3 shows the expected network lifetime for different levels of initial energy ($\mathcal{E}_0$) and maximum allowed temperature increase (τ). For example, for $\tau = 3$ (i.e., a maximum temperature increase of three units is allowed), the maximum expected network lifetime is about 2.49. This can be achieved with an initial energy of 4 units. As the curve for $\tau = 3$ shows, increasing the initial energy will not increase the expected lifetime due to the limit on the maximum allowed temperature increase.

The initial energy of a biosensor might also become a limiting factor. For example, for $\tau = 7$, $\mathcal{E}_0$ limits the maximum expected lifetime over the range of initial energies

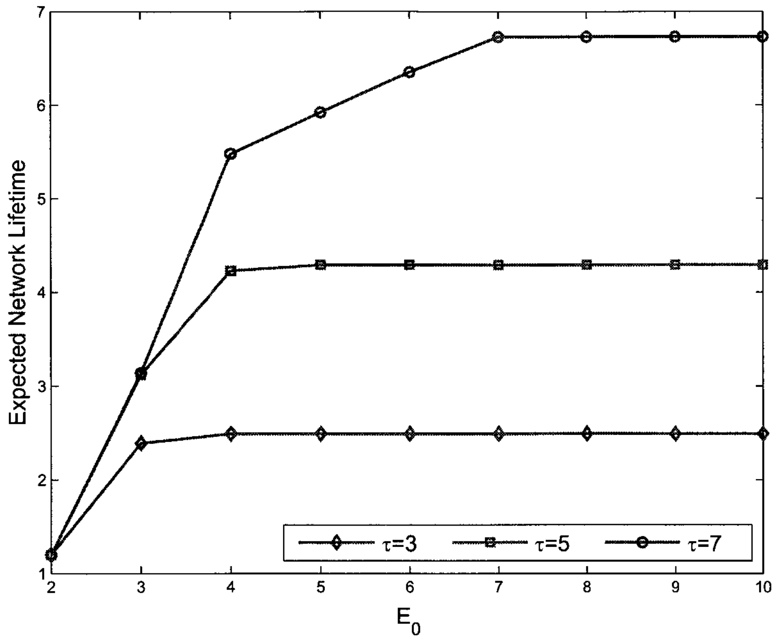


Figure 7.3: Expected network lifetime vs. initial energy for different values of τ.

from 2 to 6. After that, τ becomes the limiting factor. In this example, the maximum expected network lifetime which can be achieved with $\tau = 7$ is 6.73 with an initial energy of 7 units.

Another interesting issue is the amount of energy which remains in biosensors after the system is halted due to a high temperature increase. For example, from Figure 7.3, it can be seen that for $\mathcal{E}_0 = 4$, increasing τ leads to a noticeable increase in the expected lifetime of the network. This indicates that the amount of initial energy must be determined carefully. This is because an excessive amount of remaining energy means that the patient has been exposed to an unnecessary temperature increase when the biosensors implanted in his body were charged. Thus, the measurement process has been started on already heated organs.

Figure 7.4 shows the actions the optimal policy makes when the remaining energy at each biosensor is fixed at three and the transmission energies of biosensors 1 and 2 are both two and that of biosensor 3 is one. $\mathcal{E}_0$ and τ are both 5. After analyzing the data, it is found that biosensor 3 is selected for transmission in 64% of the system

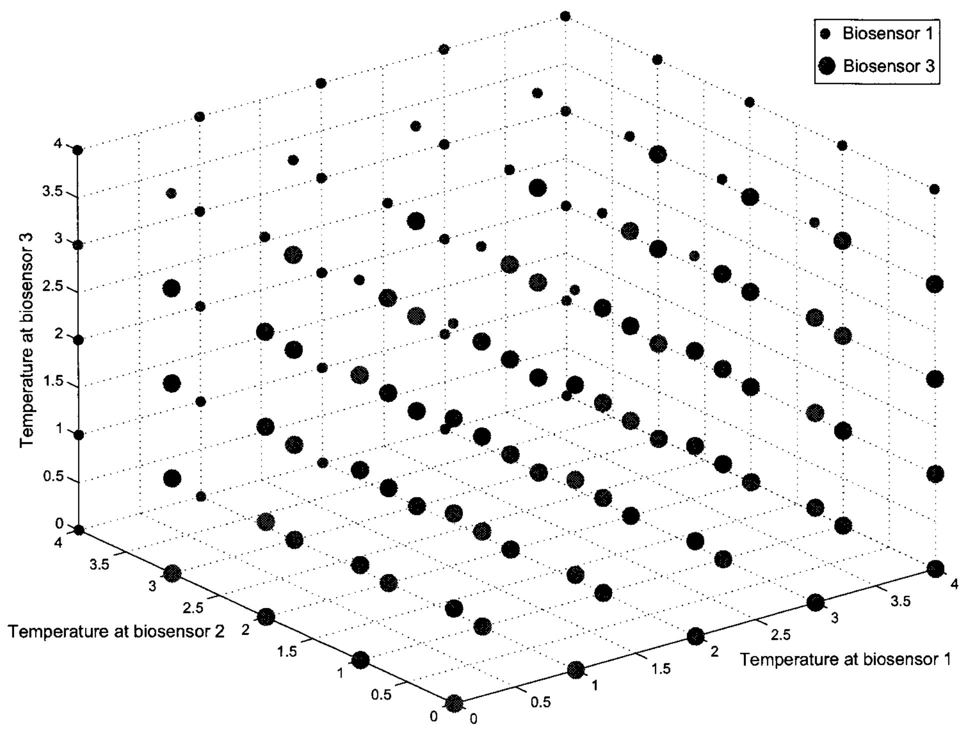


Figure 7.4: Optimal actions when E(1) = E(2) = E(3) = 3, W(1) = W(2) = 2 and W(3) = 1, T = 5 and $\mathcal{E}_0 = 5$.

states since it results in the minimum temperature increase. This is obvious since only one unit of energy is required for a successful transmission and the size of its neighborhood is one. Biosensor 2 is never selected. Biosensor 1, however, is selected when the temperature at biosensor 3 or its neighbor (biosensor 2) is 4. This is because if any one of them is selected, the system will enter a terminating state. So, biosensor 1 is selected to let biosensor 3 cool down and thus lengthen the network lifetime or to distribute heat evenly if the network is going to enter a terminating state.

7.4.2 Comparative Analysis

The BWSN in Figure 7.1 is simulated to compare the performance of the optimal policy with that of the TIP-based and most residual energy policies. Also, the impact of varying the initial energy and maximum safe temperature level is evaluated. The

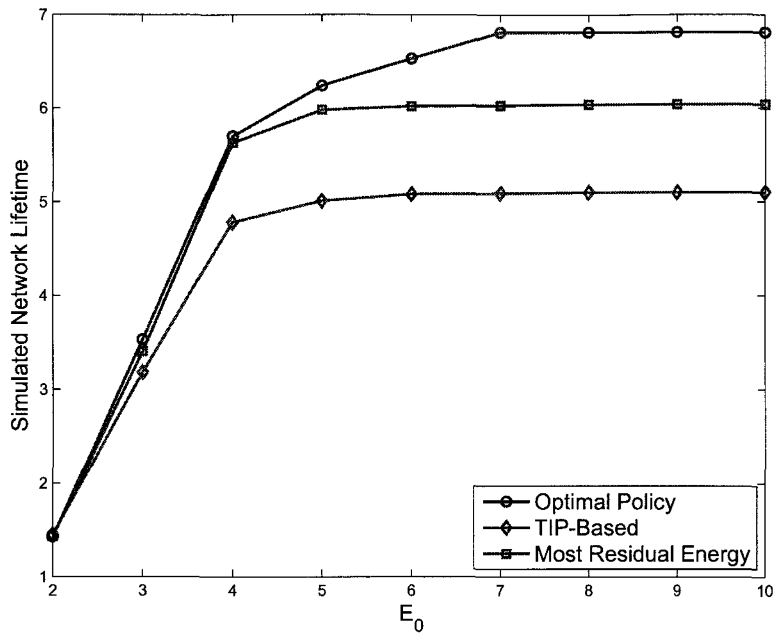


Figure 7.5: Simulated network lifetime vs. initial energy for different policies.

simulator is written in Matlab [79] and each data point is the average of 1000 simulation runs. The TIP-based policy (or the optimal rotation sequence) is computed as described in [35]. The optimal sequence is (3,1,2). The peak potential is 0.148 and is experienced by biosensor 2.

First, the impact of varying the initial energy on the network lifetime is studied. Figure 7.5 shows the simulated lifetime of the network when the initial energy is varied from 2 to 10 and τ is fixed at 7. Essentially, the network lifetime increases as the initial energy increases. However, after a threshold (around 4), the lifetime curve starts to level off for all policies. This is because the limit on the maximum allowed temperature increase is reached. Therefore, unless τ is increased, the average network lifetime will not increase with the increase of the initial energy.

Figure 7.5 also shows that the optimal policy outperforms the other two policies. The TIP-based policy performs the worst. The main reason for its poor performance is that the TIP-based policy does not account for the effects of the wireless channel. On the other hand, the policy based on the most residual energy performs better than

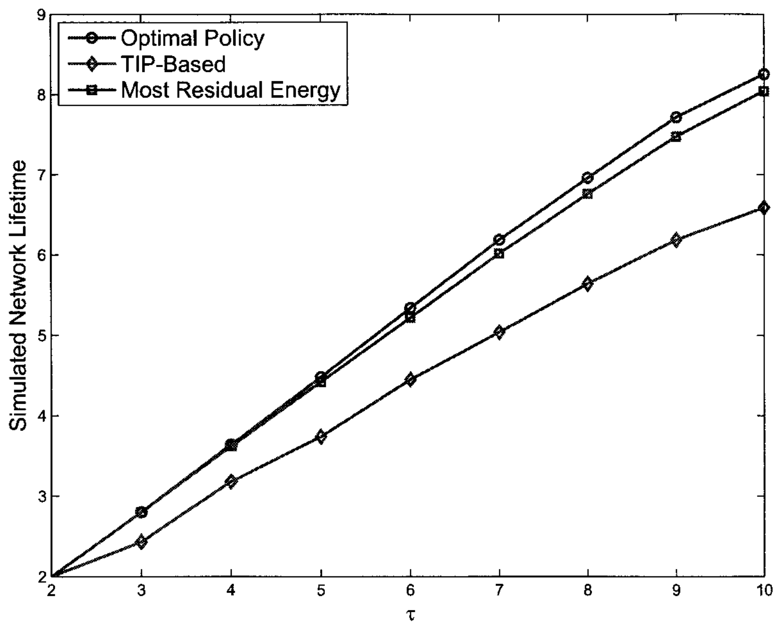


Figure 7.6: Simulated network lifetime vs. maximum safe temperature level for different policies.

the TIP-based policy. This is because it always chooses the sensor which consumes the least amount of energy for transmission. Hence, the gap between its curve and that of the optimal policy is smaller. Nevertheless, its performance cannot reach the performance of the optimal policy since temperature is not considered explicitly.

Figure 7.6 shows the impact on the network lifetime when fixing the initial energy at 5 and varying the upper limit on the safe temperature level from 2 to 10. As expected, the network lifetime increases as τ increases. The optimal policy clearly gives the best network lifetime. The policy based on most residual energy gives the next best network lifetime. The worst network lifetime is achieved by the TIP-based policy. Compared to Figure 7.5, Figure 7.6 tells that with $\mathcal{E}_0 = 5$, longer lifetime can be achieved if τ is increased. This indicates that τ has more impact of the lifetime of the network.

The performance of the three policies in terms of temperature increase is compared next. The initial energy is fixed at $\mathcal{E}_0 = 7$. The temperature at biosensor 2 is chosen as

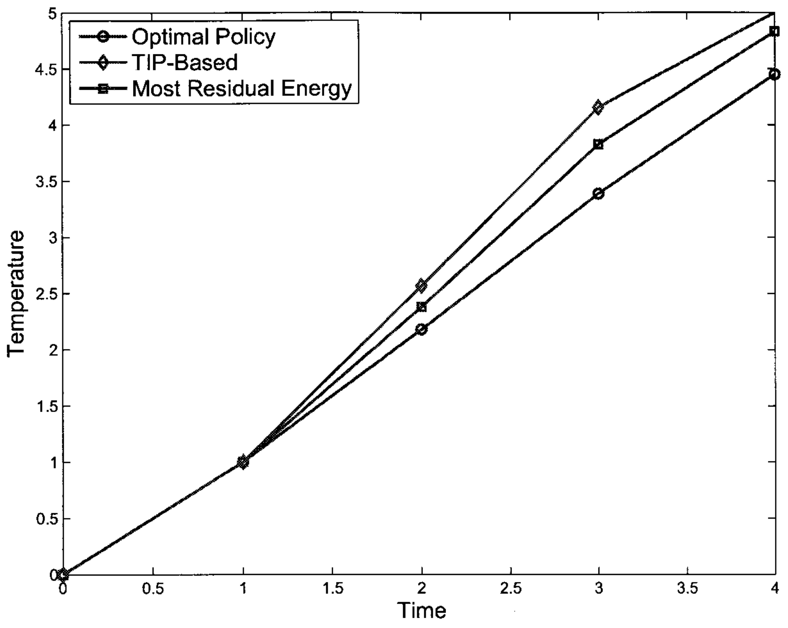


Figure 7.7: Temperature at biosensor 2 for different policies.

a metric. This is because biosensor 2 belongs to the neighborhoods of both biosensors 1 and 2. Thus, it might be heated continuously.

Figure 7.7 shows the temperature at biosensor 2 over four time slots. As expected, the TIP-based policy gives the maximum temperature increase. A closer examination of the simulation data reveals that biosensor 2 has indeed been continuously heated. This in turns leads to a larger temperature increase and thus shorter lifetime since the maximum allowed temperature is approached very fast. Both the most residual energy and optimal policies give a significant improvement over the TIP-based policy. The performance of the two policies is slightly the same over the first two time slots. Then, the optimal policy shows a lower temperature increase over the remaining time slots.

The above observation is very interesting since the goal of the TIP-based policy is to give a minimal temperature increase rotation sequence. However, since the wireless channel and its dynamics are not taken into account, the precomputed rotation sequence will most probably lead to a larger temperature increase when implemented in practice.

7.5 Conclusions

The work in this chapter is one step further in understanding the thermal management problem in BWSNs. The thermal management problem is modeled as an MDP and then solved to obtain the optimal operating policy for the network. Further, the aggregation of final valid and terminating system states is proposed as a way for minimizing the number of states in the proposed MDP model. The equivalence of the reduced MDP model is established. Also, numerical results show a substantial reduction in model size which is obtained by aggregating just two types of system states. The optimal policy produced by the MDP model outperforms the policies based on the most residual energy and temperature increase potential. This is because the optimal policy gives the best balance between transmission energy consumption and the resulting temperature increase.

Chapter 8

Maximizing the Average Number of Samples Generated by a Rechargeable Biosensor

8.1 System Model

Figure 8.1 shows the system under study where a mobile subject, in this case an animal, has a biosensor implanted into its body. The biosensor has a built-in battery which is recharged by an RF power source. The role of the biosensor is to monitor and report interesting physiological events such as heart rate and blood pressure. The biosensor becomes incapable of detecting and reporting events if it does not have enough energy for transmission under any channel condition or the increase in its temperature exceeds a prespecified threshold. The latter condition causes a halt in system operation to allow the system to cool down.

Both the biosensor and RF power source are under the control of the base station which initiates the measurement process. The base station generates three control signals: *Sleep* and *Sample* targeted at the biosensor and *Recharge* targeted at the RF

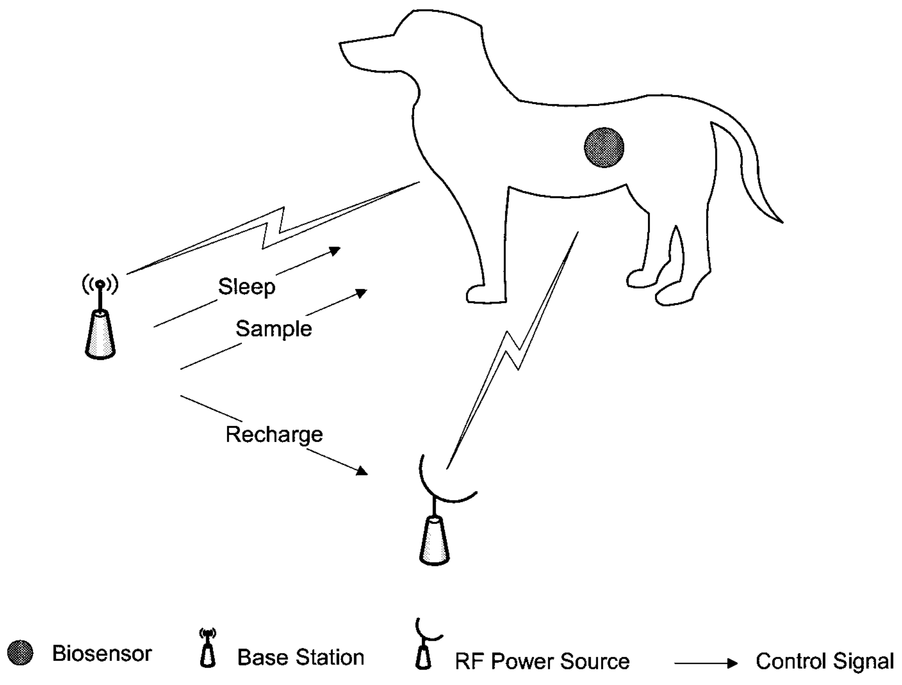


Figure 8.1: Setup of the system under study.

power source. The system state information is assumed to be available to the base station before it generates a control signal.

Mathematically, the system can be modeled as a discrete-state system which evolves in discrete time. Therefore, the time axis is divided into slots of equal duration Δt. At the beginning of each time slot, the state of the system is observed and a control signal is generated by the base station accordingly. Each time slot is long enough to transmit a complete packet carrying a measurement.

The location of a biosensor represents a critical point since it experiences the maximum temperature increase. This is because the tissues surrounding the biosensor might be heated continuously due to the local radiation generated by the biosensor itself and the radiation generated by the base station while recharging the biosensor. In each time slot t, the state of the biosensor is characterized by two variables which are the current temperature T_t and energy level E_t. There are $\tau+1$ safe temperature levels; i.e., $T_t \in \{0, 1, ..., \tau\}$ where the zero temperature level represents the normal body temperature and τ is an upper limit which must not be exceeded. Initially, the biosensor has a total energy of $\mathcal{E}_0$ which is also the capacity of its battery. The energy required for the biosensor to successfully transmit its measurement to the base station is determined by the state of the wireless channel at the time of transmission. This transmission energy is denoted by $\mathcal{E}_{w_i}$, where w_i is the i^{th} state of the wireless channel. The temperature increase due to a transmission energy of $\mathcal{E}_{w_i}$ units is denoted by $\mathcal{T}_{w_i}$.

At the beginning of each time slot, the base station may decide to recharge the biosensor, let it transmit its measurement or put it into sleep. The time required for a full recharge is random since it depends on the current temperature and energy levels. During this time, interesting events may occur but they will not be reported by the biosensor since it is being recharged. Also, the biosensor may be put into sleep for a random amount of time during which no measurements can be produced.

At the beginning of the next time slot (i.e., $t+1$), the energy level at the biosensor is given by the following equation:

$$E_{t+1} = \begin{cases} E_t & if \quad a_t = Sleep \\ E_t - \mathcal{E}_{w_i} & if \quad a_t = Sample \\ E_t + \mathcal{E}_r & if \quad a_t = Recharge \end{cases} \tag{8.1}$$

where a_t is the action taken by the base station at time t and $\mathcal{E}_r$ is the amount of energy gained by the biosensor. Similarly, the temperature of the biosensor at $t+1$ is given by the following equation:

$$T_{t+1} = \begin{cases} \max\{T_t - \mathcal{T}_s, 0\} & if \quad a_t = Sleep \\ T_t + \mathcal{T}_{w_i} & if \quad a_t = Sample \\ T_t + \mathcal{T}_r & if \quad a_t = Recharge \end{cases} \tag{8.2}$$

where $\mathcal{T}_s$ is the amount by which the temperature of the biosensor decreases when it is put to sleep. In the same way, $\mathcal{T}_r$ and $\mathcal{T}_w$ are the amounts by which the temperature of the biosensor increases when it is recharged and when it is allowed to transmit its measurement, respectively. $\mathcal{T}_r$ and $\mathcal{T}_w$ can be calculated using eqn. (6.4). $\mathcal{T}_r$ is constant since the SAR due to the base station is assumed to be constant. On the other hand, $\mathcal{T}_w$ is not constant since the SAR due to the biosensor changes with the change in transmission energy. Therefore, before $\mathcal{T}_w$ can be calculated, the SAR due to the radiation from the antenna of the biosensor is calculated using (6.2). Since (6.2) is a function of I, it is assumed that the current (I) corresponding to each transmission power level is known.

8.2 MDP Formulation

An MDP is a model of a dynamic system whose behavior varies with time. The elements of an MDP model are the following [64]:

1. System states,
2. Possible actions at each system state,
3. A reward or cost associated with each possible state-action pair, and
4. Next state transition probabilities for each possible state-action pair.

The solution of an MDP model (referred to as a policy) gives a rule for choosing an action at each possible system state. If the policy chooses an action at time t depending only on the state of the system at time t, it is referred to as a stationary policy. An optimal stationary policy exists over the class of all policies if every stationary policy gives rise to an irreducible Markov chain. This means that one can limit the attention to the class of stationary policies.

In order to obtain a policy from an MDP model, it is necessary to form and solve the so called optimality equation (or Bellman's equation). The following is the standard form of this equation with the maximization operator [65]:

$$V_n(s) = \max_{a \in A(s)} \Big[f(s,a) + \sum_{s' \in S} \mathbb{P}(s,s',a) V_{n-1}(s') \Big], \tag{8.3}$$

where n is the iteration index, S is the set of system states ($s \in S$), $A(s)$ is the set of actions possible when the system is at state s, $f(s,a)$ is the reward/cost per step, $\mathbb{P}$ is the system state transition probability matrix and $V(s)$ is the optimal value of the objective function when the system is started at state s and the optimal policy is followed. Eqn. (8.3) can be solved using the classical policy iteration, value iteration and relative value iteration algorithms [65]. Next, the details of the MDP model are given.

8.2.1 State Set

The state of the system at time t is described by the following 3-dimensional vector:

$$s_t = (T_t, E_t, W_t), \tag{8.4}$$

where T_t, E_t and W_t are the current temperature of the biosensor, its energy level and transmission power required for successful transmission at time t, respectively. The total number of possible states is $|S| = |T| \times |E| \times |W|$, where $|T|$, $|E|$ and $|W|$ are the numbers of possible temperatures, residual energies and transmission energy levels, respectively.

8.2.2 Action Set

In each time slot, the base station chooses an action based on the current state of the system. In each state s, there are three possible actions.

$$A(s) = \{Sample, Recharge, Sleep\}, \tag{8.5}$$

where the *Sample* action lets the biosensor generate a measurement and transmit it to the base station, *Recharge* action recharges the biosensor and *Sleep* action puts the biosensor into sleep.

The *Sleep* action can be performed at every system state. The other two actions, however, can only be performed at system states where the next temperature of the biosensor is within the safe temperature range. In addition, the *Sample* action can only be performed at system states where the remaining energy is sufficient to make a successful transmission.

8.2.3 Reward Function

Since the objective is to maximize the expected number of samples that can be generated by the biosensor, the reward function is defined as

$$R(s, Sample) = 1.$$

This means that one unit of reward is earned every time the *Sample* action is performed. The long-run expected sum of rewards represents the average number of samples that can be generated by the biosensor with an initial energy of $\mathcal{E}_0$ units and maximum temperature increase of τ units.

8.2.4 Transition Probability Function

After the action taken by the base station is performed, the system transits to a new state according to the transition probabilities of the present state of the wireless channel. Thus, the behavior of the system is described by $|A|$ transition probability matrices and each such matrix is of size $|S| \times |S|$. Each matrix is denoted by $P_{s_t,s_{t+1}}(a)$ which is the probability that choosing an action a when in state s_t will lead to state s_{t+1}. More formally, $P_{s_t,s_{t+1}}(a)$ can be written as the following:

$$P[s_{t+1}|s_t, a_t] = P[W_{t+1}|W_t] \tag{8.6}$$

8.2.5 Value Function

The problem of finding an optimal policy for maximizing the average number of samples is formulated as an infinite-horizon MDP using the average reward criterion [64]. So, let $V_\pi(s_0)$ be the expected number of samples given that the policy π is used with an initial state s_0. Then, the maximum expected number of samples $V^*(s_0)$ starting from state s_0 is given by

$$V^*(s_0) = \max_{\pi} V_\pi(s_0) \tag{8.7}$$

The optimal policy π^* is the one that achieves the maximum expected number of samples at all system states.

The famous value iteration algorithm [65] is used to numerically solve the following recursive equation for $n > 0$

$$V_n(s) = \max_{a \in A(s)} \left[R(s,a) + \sum_{s_{t+1} \in S} \mathbb{P}(s_t, s_{t+1}, a) V_{n-1}(s_{t+1}) \right] \tag{8.8}$$

In (8.8), the subscript n denotes the iteration index. As $n \to \infty$, $V_n \to V^*$.

8.3 Handling Large-Size MDP Models

The size of the proposed MDP model depends on the number of biosensor states which is a function of the number of possible temperature and energy levels. As the number of biosensor states increases, the process of computing the transition probability matrices for the system becomes very time consuming. Also, the value iteration algorithm used for solving the MDP model becomes impractical. This section presents two methods (namely, Q-learning and heuristics) for handling MDP models with a large number of states.

8.3.1 Q-Learning

Reinforcement Learning (RL) offers an alternative for obtaining the optimal policy at a significantly lower computational cost. Using a simulation model of the system under study, the decision maker in an MDP is viewed as a learning agent whose task is to learn the optimal action in each possible state of the system. Q-Learning is an RL algorithm which was introduced in [80]. It is used for learning from experience. It requires that each entry in the decision-maker's knowledge base corresponds to a state-action pair. The value stored in each entry is referred to as the *Q-value* and is a measure of the goodness of executing an action in a particular system state. The Q-value for a state-action pair (s, a) is updated as follows:

$$Q(s,a) = Q(s,a) + \alpha * (r + \gamma * \max_{j \in A(s')} Q(s',j) - Q(s,a)) \tag{8.9}$$

where r is the immediate reward obtained after executing action a in state s, s' is the next state and $A(s')$ is the set of possible actions in state s'. α and γ denote the learning rate and discount factor $(0 < \gamma < 1)$, respectively.

The Q-learning algorithm is shown as Algorithm 2. The interaction between the learning agent and the simulator (or environment) is divided into episodes. In each

Algorithm 2 The Q-Learning Algorithm

```
for $i = 1$ to $NumEpisodes$ do
  $ps \leftarrow InitialState$;    {E.g., [0 $\mathcal{E}_0$ 1]}
  for $j = 1$ to $NumIter$ do
    if $rand \leq 1 - \epsilon$ then
      $a \leftarrow \max_{j \in A(ps)} Q(ps, j)$;    {rand $\in$ (0,1)}
    else
      $a \leftarrow Random(A(ps))$;    {Random Action}
    end if
    if $a == Sample$ then
      $r \leftarrow 1$;    {Reward}
    else
      $r \leftarrow 0$;
    end if
    $ns \leftarrow SimulateAction(ps, a)$;
    $Update(Q(ps, a))$;    {Using eqn. (8.9)};
    $ps = ns$
  end for
end for
```

episode, the system transits through a sequence of states. The length of this sequence is controlled by the parameter $NumIter$ which is the number of simulated time slots. In each simulated time slot, based on the current state of the system, the learner chooses an action either based on the ϵ-policy or randomly. If the former is selected, the action with the highest Q-value is selected. After that, if the action is *Sample*, a reward of one unit is earned; otherwise, the reward is zero. The action is then simulated and the next system state is observed. Next, the Q-value is updated using eqn. (8.9). Also, the new system state becomes the current one and the cycle repeats.

Although Q-learning is theoretically guaranteed to obtain an optimal policy, it requires that each state-action pair be tried infinitely often in order to learn the optimal policy. The quality of the learned policy depends on how much time is spent in learning and if every state-action pair can be tried. On the other hand, depending on the application, a certain percental difference between the learned and optimal policies might be tolerated. This is because the system states differ in the likelihood of being visited. Thus, a default action (like *Sleep*) can be assigned to system states

Algorithm 3 Greedy Policy

```
Require: $S$: Set of possible system states
         $A$: Set of possible actions at each system state
  for $i = 1$ to $|S|$ do
    if Action(i, Sample) is True then
      $Policy(i) = Sample$
    else if Action(i, Recharge) is True then
      $Policy(i) = Recharge$
    else
      $Policy(i) = Sleep$
    end if
  end for
```

with a low likelihood of being visited.

8.3.2 Heuristic

Since it is difficult to describe the structure of the optimal policy, a heuristic policy is proposed in this section. The goal is to design a policy which mimics the behavior of the optimal policy as close as possible. However, before presenting such a policy, a greedy one is given to provide insight into the design of any heuristic policy.

The greedy policy is computed using Algorithm 3. The inputs to this algorithm are the set of possible system states and set of feasible actions for each system state. The computed policy is greedy in the sense that for each system state, the feasibility of actions is checked in the following order: *Sample*, *Recharge* and then *Sleep*. The first feasible action is associated with the corresponding system state.

As will be shown by simulations in the next section, the greedy policy is poor since it is based on a fixed order of actions. Therefore, Algorithm 3 needs to be extended to allow for a dynamic selection of actions. This objective is accomplished by introducing two control parameters: α and β. With these two control parameters, the *Sample* and *Recharge* actions are not selected in a specific order or whenever they are feasible. Algorithm 4 shows how the control parameters and new heuristic policy

Algorithm 4 Heuristic Policy

```
Require: $S$: Set of possible system states
         $A$: Set of possible actions at each system state
  $\alpha = \frac{T}{\tau}$
  $\beta = \frac{E}{E_0}$
  for $i = 1$ to $|S|$ do
    if Action(i, Sample) is True & $\alpha \leq \beta$ then
      $Policy(i) = Sample$
    else if Action(i, Recharge) is True & $\alpha \geq \beta$ then
      $Policy(i) = Recharge$
    else
      $Policy(i) = Sleep$
    end if
  end for
```

are computed.

The essence of Algorithm 4 is as follows. If the current temperature (denoted by T) of the biosensor is low and its current energy level (denoted by E) is high, then the condition $\alpha \leq \beta$ would more likely be true and thus the *Sample* action could be executed. However, this would not be the case when the available energy is very close to zero. In this case, the opposite condition (i.e., $\alpha \geq \beta$) would more likely be true and thus a *Recharge* could be performed. If neither of the two conditions is true, the biosensor is put to sleep and thus its temperature decreases.

8.4 Numerical and Simulation Results

In this section, an example is first presented to illustrate the viability of the proposed MDP model. Then, the performance of the optimal policy is compared to that of the approximate policies using simulation. The impact of various system parameters on the performance of the system is also evaluated. The simulation was performed using a simulator written in Matlab [79]. Each simulation was run for a duration of 100000 time slots and each data point is the average of 10 simulation runs. The number of

Table 8.1: Values assigned to the Parameters of the Example and Q-learning algorithm.

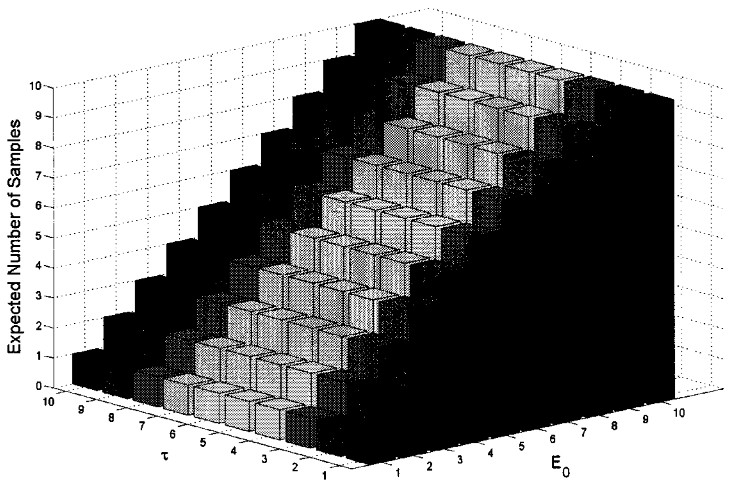
	Parameter	Value
Example	Δt	0.25 ms
	f_d	60 Hz
	$\mathcal{E}_0$	5
	T	5
	$\mathcal{E}_{w_1}$, $\mathcal{E}_{w_2}$	1, 2
	$\mathcal{T}_{w_1}$, $\mathcal{T}_{w_2}$	1, 2
	$\mathcal{T}_s$	1
	$\mathcal{T}_r$	1
	$\mathcal{E}_r$	1
Q-Learning	$NumEpisodes$	100
	$NumIter$	10000
	ϵ	0.5
	α	0.2
	γ	0.1

channel states (W) is four and the channel state boundaries are randomly generated.

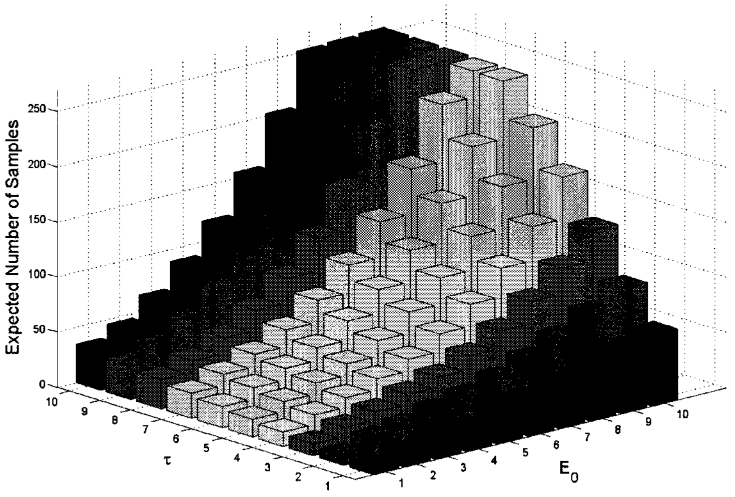
8.4.1 Illustrative Example

In this example, a wireless channel with two states is considered. The channel state transition probabilities are calculated using equations (6.10) and (6.11). Table 8.1 shows the values of the parameters involved. Figures 8.2(a) and 8.2(b) show the expected number of samples when there is no recharge and when recharge is allowed, respectively. The expected number of samples is expressed as a function of the maximum safe temperature level (τ) and initial energy ($\mathcal{E}_0$). The first observation is that if recharge is allowed, more samples are expected to be generated by the biosensor. In the case when recharge is not allowed, the expected number of samples is limited only by the amount of initial energy. This is confirmed by Figure 8.2(a) where for the same initial energy, the same expected number of samples is obtained when τ is varied.

When recharge is allowed, the maximum safe temperature level (τ) plays a critical



(a)



(b)

Figure 8.2: Expected number of samples. (a) No recharge. (b) With recharge.

role. This is due to the temperature increase caused by the recharge action. In Figure 8.2(b), for the same initial energy, the expected number of samples increases as τ is varied. Increasing τ enables the *Recharge* action to be performed more often. On the other hand, as one would expect, if τ is fixed and ($\mathcal{E}_0$) is varied, the expected number of samples slightly increases when τ is small. However, when τ is large (≥ 6), the maximum possible expected number of samples can be achieved when $\mathcal{E}_0$ is at its maximum value. Therefore, for this particular example, if $\mathcal{E}_0 = 10$, the optimal value for τ is 6.

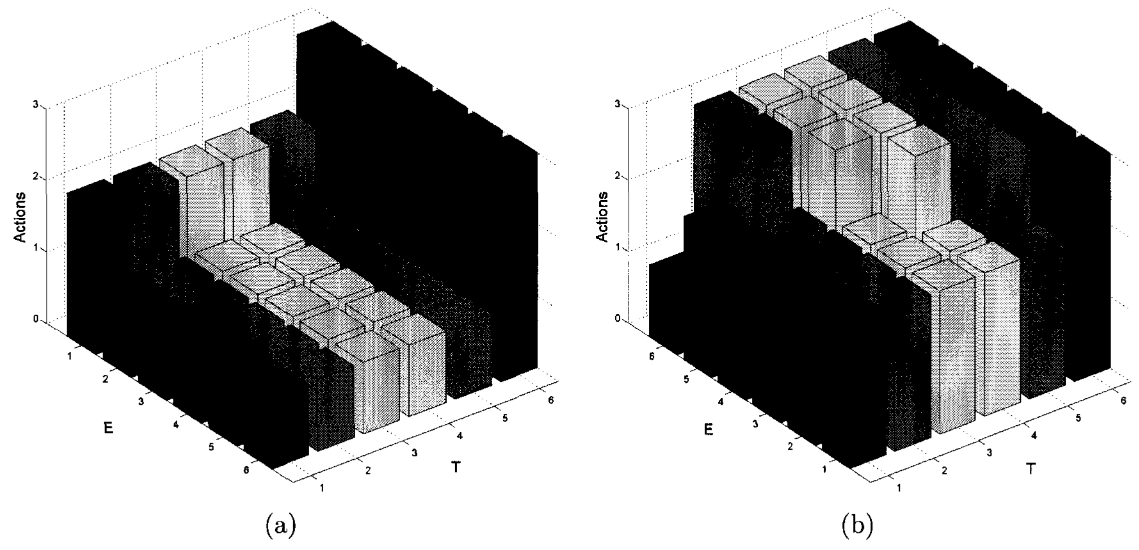


Figure 8.3: Optimal policy for $\tau = 5$ and $\mathcal{E}_0 = 5$. Actions 1, 2 and 3 denote *Sample*, *Recharge* and *Sleep*, respectively. (a) Policy for channel state 1. (b) Policy for channel state 2.

Figures 8.3(a) and 8.3(b) show the optimal action for each possible system state. In Figure 8.3(a), for channel state 1, the *Sample* action is performed in 70% of the system states. The *Sleep* action is performed whenever the temperature reaches the maximum safe level (τ) and the *Recharge* action is performed when the remaining energy is zero and the temperature is below τ.

By contrast, in Figure 8.3(b), for channel state 2, the *Sample* action is performed only once at the initial system state. For this channel state, due to the higher cost of transmission, the biosensor is put in the sleep mode most of the time. However, since the cost of the *Recharge* action is independent of the channel state, the system recharges itself more often to enable more samples to be generated when the wireless channel switches to a state with a lesser transmission energy requirement (i.e., channel state 1).

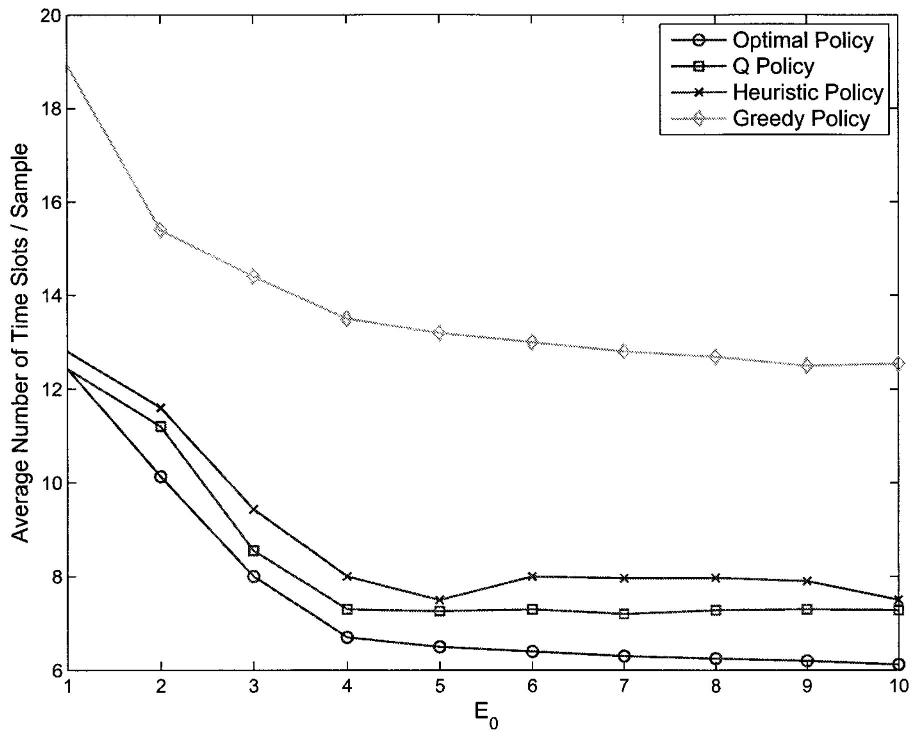


Figure 8.4: Average number of time slots needed to generate a sample when τ is fixed at five and $\mathcal{E}_0$ is varied.

8.4.2 Comparative Analysis

In order to be able to appreciate the merit of any approximate policy, a more meaningful performance criterion is needed. In this work, *the average number of time slots needed to generate a sample* is used as a criterion to distinguish between the different policies available to run a system. It is calculated as the total simulation time divided by the average number of samples generated by the system while being operated by a certain policy. This measure takes into account the effect of the *Recharge* and *Sleep* actions.

For example, consider Figure 8.4. In this figure, τ is fixed at five while $\mathcal{E}_0$ is varied. The greedy policy is very costly since it requires the largest amount of time before a sample can be generated. The difference in the amount of time required by the heuristic policy and that required by the optimal policy stays around two time slots.

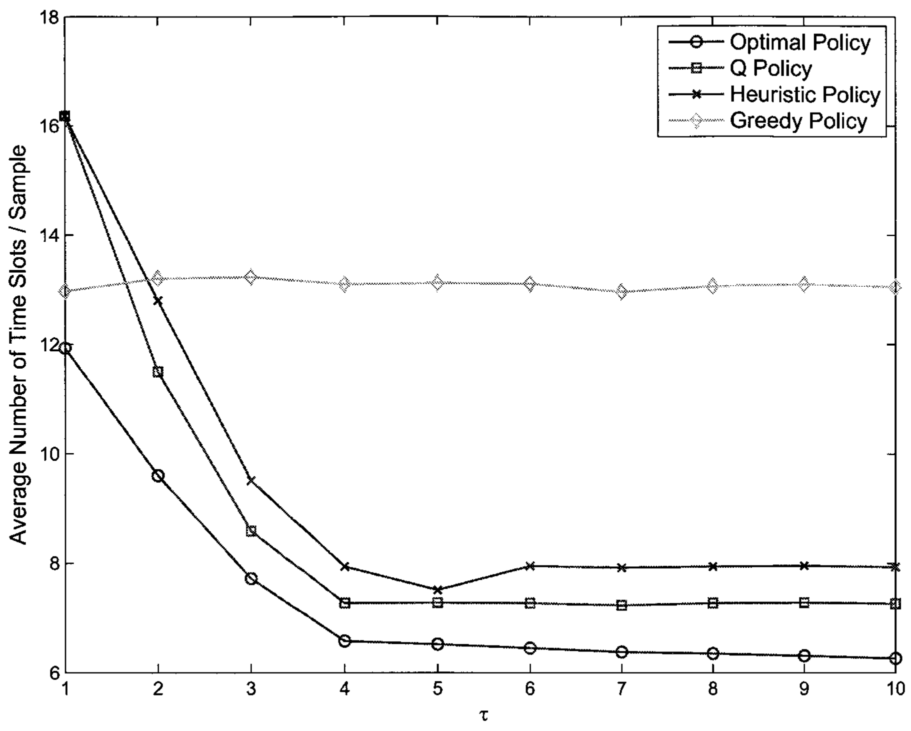


Figure 8.5: Average number of time slots needed to generate a sample when $\mathcal{E}_0$ is fixed at five and τ is varied.

This is a 75% reduction in time when compared to the greedy policy. The Q policy is the best approximate policy. On average, the difference with the optimal policy stays around 1.1 time slots.

Figure 8.5 shows the amount of time required to generate a sample when $\mathcal{E}_0$ is fixed at five and τ is varied. In this figure, when $\tau = 1$, the greedy policy outperforms both the Q policy and heuristic policy. A difference of three time slots is observed. This can be explained as follows. In the Q and heuristic policies, the *Recharge* action can be performed in one state only (i.e., when $T = E = 0$). On the other hand, with the greedy policy, the *Recharge* action can be performed in more than one state (i.e., whenever $T = 0$). This, of course, leads to a reduction in the average amount of time needed to generate a sample. Other than that, for $\tau \geq 2$, the Q and heuristic policies are always better than the greedy policy and their performance is close to that of the optimal policy.

8.5 Conclusions

The increase in temperature due to the heat generated by biosensors is a limiting factor in the operation of biosensor networks. This problem can be modeled as a stochastic control problem using the framework of Markov decision processes. The solution is an optimal policy which ensures that the maximum safe temperature level is not exceeded. In order to handle large-size MDP models, it is shown how Q-learning can be used for obtaining the optimal policy. In addition, a heuristic policy is proposed. Its performance is comparable to that of the policies obtained by the MDP model and Q-learning.

Chapter 9

Conclusions and Directions for Further Work

In this dissertation, two kinds of WSNs were considered: (1) directional and (2) biological. For directional WSNs, three planning problems were identified: (1) MSP, (2) MCSP and (3) OSC. They were formulated as ILP models which were then solved to obtain directional WSNs with either the minimum number of sensors, minimum overall cost or optimal sensor configuration. The solutions obtained by the three models represent WSNs which are connected and cover all the critical sites in the sensor field.

The MSP problem turned out to be a generalization of the classical grid coverage problem in conventional WSNs. It is also a generalization of the coverage problem with sensors whose sensing fields can be represented as a disk. The OSC problem also turned out to be a generalization of the MSP and MCSP problems. For all the proposed models, the experimental results showed a significant reduction in the number and cost of sensors necessary for deployment.

The above models represents a foundation for further exploration. Other parameters like energy consumption, reliability and fault-tolerance can easily be added to the proposed models without any modifications. Also, the mobility of sensors based on preplanned routes can be considered. In this case, only the input data to the models need to be computed at each point along the preplanned routes. On the other hand, the proposed deterministic ILP models will not be appropriate if stochastic mobility models are considered.

As for biological WSNs, two control problems were identified. The two problems arise when biological sensors are operated in temperature-sensitive environments like the human body. They were formulated as MDPs and solved to obtain optimal policies for operating the WSN while the constraint on the maximum safe temperature level is not exceeded. The effectiveness and viability of the MDP models were demonstrated through numerical and experimental results. Further, the policies obtained by the proposed models were superior when compared with published policies.

The proposed MDP model assumes that biosensors can only communicate with the base station. Thus, the measurements collected by the biosensors are processed by the base station. Biosensors can be allowed to process the measurements and only the results are delivered to the base station. If inter-sensor communication is allowed, the set of possible actions for a biosensor will include its neighboring nodes. Furthermore, the communication between the biosensors must be represented by a different wireless channel model since it happens inside the body of the patient.

The proposed MDP model also assumes that the decision maker (i.e., the base station) knows with absolute certainty the current state of the system. This is, however, not the case in many real-world situations where the decision maker can only make observations about the current state of the system. Such observations typically provide only incomplete information due to noise or uncertainty. In this case, we say that the state of the system is only partially observable. Such systems are modeled by paritally observable MDPs which are MDPs with an additional set of observations and their probability distribution at each time instant.

In order to handle large-size MDP models, state aggregation and Q-learning were used. Also, heuristics for computing approximate policies were proposed. In the case of state aggregation, two classes of system states were identified as crucial in reducing the size of the MDP model. In addition, the equivalence of the policy obtained by the reduced MDP model was established.

The work reported in this dissertation can be extended in several directions. The following are some suggestions:

1. The OSC problem becomes computationally infeasible for large problem instances. Search techniques such as genetic algorithms and tabu search can be used for obtaining optimal or near-optimal solutions.

2. The proposed models for directional WSNs are based on a simple transmission model. The use of more realistic transmission models would be very useful.

3. The inclusion of k-coverage, energy consumption and network lifetime needs to be considered.

4. The proposed ILP models are deterministic. Thus, they cannot be used with stochastic mobility models. New optimization models need to be developed.

5. Modeling the sensing region of a sensor as a two-dimensional surface might be inadequate. Some applications like undersea monitoring require that the sensing region of a sensor be modeled as a three-dimensional space. Thus, the area of three-dimensional coverage with directional sensors needs to be explored. The works in [81, 82] can be used as a starting point.

6. The notion of state equivalence used in Chapter 8 is too strict and too sensitive. It is too strict because it requires that its conditions be met exactly. And, it is too sensitive because any perturbation of the transition probabilities can

make two equivalent states no longer equivalent. More flexible metrics for state equivalence are needed. The works in [73, 74] can be used as a starting point.

7. The state transition probability matrix is built programmatically. This means a runtime which largely grows with the number of system states and thus state aggregation might not always be helpful. Hence, approximate techniques based on reinforcement learning are recommended (see [65–69]).

8. The possibility of obtaining effective policies based on simple heuristic techniques should be investigated. Heuristic techniques are typically characterized by their low runtime and storage requirements.

9. If inter-sensor communications are allowed, the set of possible actions for each biosensor might become large. This would be a limiting factor. A natural extension would be to develop techniques for aggregating action spaces.

References

[1] J. Ai and A. A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *Journal of Combinatorial Optimization*, vol. 11, no. 1, pp. 21–41, February 2006.

[2] W. Cheng, S. Li, X. Liao, S. Changxiang, and H. Chen, "Maximal coverage scheduling in randomly deployed directional sensor networks," in *Proc. International Conference on Parallel Processing Workshops (ICPPW).* IEEE, 2007, pp. 68–73.

[3] Y. Cai, W. Lou, M. Li, and X.-Y. Li, "Target-oriented scheduling in directional sensor networks," in *Proc. International Conference on Computer Communications (INFOCOM).* IEEE, 2007, pp. 1550–1558.

[4] *The WSN Starter Kit, Crossbow Technology, Inc.*, http://www.xbow.com.

[5] S. Soro and W. Heinzelman, "On the coverage problem in video-based wireless sensor networks," in *Proc. International Conference on Broadband Networks.* IEEE, 2005, pp. 932–939.

[6] M. P. J. Adriaens, S. Megerian, "Optimal worst-case coverage of directional field-of-view sensor networks," in *Proc. International Conference on Sensor and Ad Hoc Communications and Networks.* IEEE, 2006, pp. 336–345.

[7] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proc. International Conference on Computer Communications (INFOCOM).* IEEE, 2001, pp. 1380–1387.

[8] D. W. Gage, "Command control for many-robot systems," in *Proc. the Nineteenth Annual AUVS Technical Symposium*, 1992, pp. 22–24.

[9] J. O'Rourke, *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[10] B. Wang, W. Wang, V. Srinivasan, and K. C. Chua, "Information coverage for wireless sensor networks," *IEEE Communications Letters*, vol. 9, no. 11, pp. 967–969, Nov. 2005.

[11] H. Bai, X. Chen, Y.-C. Ho, and X. Guan, "Information coverage configuration with energy preservation in large scale wireless sensor networks," in *Proc. the International Conference on Computer and Information Technology.* IEEE, 2006, pp. 221 –221.

[12] B. Wang, V. Srinivasan, K. C. Chua, and W. Wang, "Information coverage and network lifetime in energy constrained wireless sensor networks," in *Proc. Conference on Local Computer Networks.* IEEE, 2007, pp. 512 –519.

[13] V. Chvatal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory Series B*, vol. 18, pp. 39–41, 1975.

[14] S. Fisk, "A short proof of chavtál's watchman theorem," *Journal of Combinatorial Theory Series B*, vol. 24, p. 374, 1975.

[15] M. Marengonia, B. Draperb, A. Hansona, and R. Sitaramana, "A system to place observers on a polyhedral terrain in polynomial time," *Journal of Image and Vision Computing*, vol. 18, pp. 773–780, 2000.

[16] E. Horster and R. Lienhart, "Approximating optimal visual sensor placement," in *Proc. International Conference on Multimedia and Expo.* IEEE, 2006, pp. 1257–1260.

[17] E. Horster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *Proc. International Workshop on Video Surveillance and Sensor Networks.* ACM, 2006, pp. 111–120.

[18] J. Zhao and S.-C. S. Cheung, "Multi-camera surveillance with visual tagging and generic camera placement," in *Proc. International Conference on Distributed Smart Cameras.* ACM/IEEE, 2007, pp. 259–266.

[19] K. Chakrabarty, S. Sitharama, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448–1453, 2002.

[20] S. Sahni and X. Xu, "Algorithms for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 1, no. 1, pp. 35–56, January 2005.

[21] J. Wang and N. Zhong, "Efficient point coverage in wireless sensor networks," *Journal of Combinatorial Optimization*, vol. 11, no. 3, pp. 291–304, May 2006.

[22] M. Berkelaar, P. Notebaert, and K. Eikland, *LP Solve: Linear Programming Code, version 5.5*, Eindhoven University of Technology, The Netherlands, 2005.

[23] X. Xu and S. Sahni, "Approximation algorithms for sensor deployment," *IEEE Transactions on Computers*, vol. 56, no. 12, pp. 1681–1695, December 2007.

[24] X. Han, X. Cao, E. Lloyd, and C.-C. Shen, "Deploying directional sensor networks with guaranteed connectivity and coverage," in *Proc. Sensor, Mesh and Ad Hoc Communications and Networks.* IEEE, 2008, pp. 153–160.

[25] I. Solis and K. Obraczka, "Isolines: energy-efficient mapping in sensor networks," in *Proc. Symposium on Computers and Communications (ISCC).* IEEE, 2005, pp. 379 – 385.

[26] K. Kotay, R. Peterson, and D. Ru, "Experiments with robots and sensor networks for mapping and navigation," in *Proc. International Conference on Field and Service Robotics*, 2005.

[27] J. D. Nicoud and P. Machler, "Robots for anti-personnel mine search," *Control Engineering Practice*, vol. 4, pp. 493–498, 1996.

[28] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Oeh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Architectural Support for Programming Languages and Operating Systems*, 2002.

[29] T. Small and Z. J. Haas, "The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way)," in *Proc. International Symposium on Mobile Ad Hoc Networking and Computing.* ACM, 2003, pp. 233–244.

[30] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using predictable observer mobility for power efficient design of sensor networks," in *Proc. International Conference on Information Processing in Sensor Networks.* Springer, 2003.

[31] A. Savkin, "Coordinated collective motion of groups of autonomous mobile robots: analysis of vicsek's model," *IEEE Transactions on Automatic Control*, vol. 49, no. 6, pp. 981 – 982, June 2004.

[32] J. Reich and E. Sklar, "Toward automatic reconfiguration of robot-sensor networks for urban search and rescue," in *Proc. Workshop on Autonomous Agents and Multi-Agent Systems (Agent Technology for Disaster Management)*, 2006.

[33] J. Reich and J. Reich, "Robot-sensor networks for search and rescue," in *Proc. International Workshop on Safety, Security and Rescue Robotics.* IEEE, 2006.

[34] F. Amigoni, G. Fontana, and S. Mazzuca, "Robotic sensor networks: An application to monitoring electro-magnetic fields," in *Proc. Conference on Emerging Artificial Intelligence Applications in Computer Engineering.* IOS Press, 2007, pp. 384–393.

[35] Q. Tang, N. Tummala, S. Gupta, and L. Schwiebert, "Communication scheduling to minimize thermal effects of implanted biosensor networks in homogeneous tissue," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 7, pp. 1285–1294, July 2005.

[36] Q. Tang, N. Tummala, S. Gupta, and L. Schwiebert, "Tara: Thermal-aware routing algorithm for implanted sensor networks," in *Proc. Distributed Computing in Sensor Systems*, vol. 3560. Springer, 2005, pp. 206–217.

[37] Y. Chen, Q. Zhao, V. Krishnamurthy, and D. Djonin, "Transmission scheduling for optimizing sensor network lifetime: A stochastic shortest path approach," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 2294–2309, May 2007.

[38] N. Jaggi, K. Kar, and A. Krishnamurthy, "Rechargeable sensor activation under temporally correlated events," *Wireless Networks*, vol. 15, no. 5, pp. 619–635, July 2009.

[39] G. Z. Yang, *Body Sensor Networks.* Springer, 2006.

[40] A. Seyedi and B. Sikdar, "Energy efficient transmission strategies for body sensor networks with energy harvesting," in *Proc. Conference on Information Sciences and Systems.* IEEE, 2008, pp. 704–709.

[41] A. Logothetis and A. Isaksson, "On sensor scheduling via information theoretic criteria," in *Proc. of the American Control Conference*, vol. 4. IEEE, 1999, pp. 2402 –2406.

[42] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden markov model sensors," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1382 –1397, Jun. 2002.

[43] M. Kalandros, "Covariance control for multisensor systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 4, pp. 1138 – 1157, oct 2002.

[44] Y. He and K. Chong, "Sensor scheduling for target tracking in sensor networks," in *Proc. Conference on Decision and Control*, vol. 1. IEEE, 2004, pp. 743–748.

[45] R. Knopp and P. Humblet, "Information capacity and power control in single-cell multiuser communications," in *Proc. International Conference on Communications (ICC)*, vol. 1. IEEE, 18-22 1995, pp. 331 –335.

[46] Y. Chen and Q. Zhao, "Distributed transmission protocol for lifetime maximization in sensor networks," in *Proc. Workshop on Signal Processing Advances in Wireless Communications*. IEEE, 2005, pp. 895 – 899.

[47] R. Madan, S. Cui, S. Lal, and A. Goldsmith, "Cross-layer design for lifetime maximization in interference-limited wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 11, pp. 3142 –3152, Nov. 2006.

[48] J. H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 609–619, Aug. 2004.

[49] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102 – 114, aug 2002.

[50] *MICA Wireless Measurement System. Datasheet, Crossbow Technology, Inc.*, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf.

[51] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *Proc. International Conference on Embedded Networked Sensor Systems.* ACM, 2003, pp. 181–192.

[52] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proc. International Symposium on Mobile Ad Hoc Networking and Computing.* ACM, 2006, pp. 322–333.

[53] S. Datta and S. Biswas, "Limiting carrier-sense energy consumption by low-power interface idling in 802.11 protocols," in *Proc. International Conference on Wireless Networks.* IEEE, 2004.

[54] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *Proc. International Conference on Data Engineering.* IEEE, 2003, pp. 429 – 440.

[55] Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy efficient data collection in distributed sensor environments," in *Proc. International Conference on Distributed Computing Systems.* IEEE, 2004, pp. 590 – 597.

[56] S. Park and R. Sivakumar, "Energy efficient correlated data aggregation for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 4, pp. 13 – 27, 2008.

[57] Q. Dong, "Maximizing system lifetime in wireless sensor networks," in *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE, 2005, pp. 13 – 19.

[58] *ILOG Inc., Incline Village, NV*, http://www.ilog.com/products/cplex.

[59] National Council on Radiation Protection and Measurements (NCRP), "A practical guide to the determination of human exposure to radiofrequency fields," Report No. 119, 1993.

[60] International Electrotechnical Commission (IEC), "Medical electrical equipment, part 2-33: Particular requirements for the safety of magnetic resonance equipment for medical diagnosis," IEC 60601-2-33, 2nd Edn., 1995.

[61] S. Gupta, S. Lalwani, Y. Prakash, E. Elsharawy, and L. Schwiebert, "Towards a propagation model for wireless biomedical applications," in *Proc. International Conference on Communications*. IEEE, 2003, pp. 1993–1997.

[62] H. H. Pennes, "Analysis of tissue and arterial blood temperature in the resting human forearm," *Journal of Applied Physiology*, vol. 1, no. 1, pp. 93–122, July 1948.

[63] D. M. Sullivan, *Electromagnetic Simulation Using the FDTD Method.* IEEE Press, 2000.

[64] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, 2005.

[65] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume I.* Athena Scientific, 2000.

[66] W. B. Powell, *Approximate Dynamic Programming - Solving the Curse of Dimensionality.* Wiley, 2007.

[67] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, *Simulation-based Algorithms for Markov Decision Processes.* Springer, 2007.

[68] X.-R. Cao, *Stochastic Learning and Optimization: A Sensitivity-Based Approach.* Springer, 2007.

[69] J. Si, A. G. Barto, W. B. Powell, and D. W. (Editors), *Handbook of Learning and Approximate Dynamic Programming.* Wiley-IEEE Press, 2004.

[70] Z. Ren and B. Krogh, "State aggregation in markov decision processes," in *Proc. Conference on Decision and Control.* IEEE, 2002, pp. 3819–3824.

[71] R. Givan, T. Dean, and M. Greig, "Equivalence notions and model minimization in markov decision processes," *Artificial Intelligence*, vol. 147, no. 1-2, pp. 163–223, July 2003.

[72] P. Castro, P. Panangaden, and D. Precup, "Equivalence relations in fully and partially observable markov decision processes," in *Proc. International Joint Conference on Artificial Intelligence.* Morgan Kaufmann Publishers Inc., 2009, pp. 1653–1658.

[73] N. Ferns, P. Panangaden, and D. Precup, "Metrics for finite markov decision processes," in *Proc. Conference on Uncertainty in Artificial Intelligence.* AUAI Press, 2004, pp. 162–169.

[74] N. Ferns, P. Castro, D. Precup, and P. Panangaden, "Methods for computing state similarity in markov decision processes," in *Proc. Conference on Uncertainty in Artificial Intelligence.* AUAI Press, 2006, pp. 13–16.

[75] J. G. Proakis, *Digital Communications.* McGraw-Hill, 2000.

[76] H. S. Wang and N. Moayeri, "Finite-state markov channel-a useful model for radio communication channels," *IEEE Transactions on Vehicular Technology*, vol. 44, no. 1, pp. 163–171, Feb 1995.

[77] Q. Zhang and S. A. Kassam, "Finite-state markov model for rayleigh fading channels," *IEEE Transactions on Communications*, vol. 47, no. 11, pp. 1688–1692, Nov. 1999.

[78] W. C. Jakes, *Microwave Mobile Communications.* Wiley, 1974.

[79] The MathWorks, Inc. [Online]. Available: http://www.mathworks.com

[80] C. Watkins, *Learning from delayed rewards.* Springer, 1989. [Online]. Available: http://www.cs.rhul.ac.uk/~chrisw/thesis.html

[81] M. Watfa and S. Commuri, "The 3-dimensional wireless sensor network coverage problem," in *Proc. International Conference on Networking, Sensing and Control.* IEEE, 2006, pp. 856 –861.

[82] M. Watfa and S. Commuri, "Optimality measures for coverage in 3d wireless sensor networks," in *Proc. International Symposium on Wireless Pervasive Computing.* IEEE, 2006.

Appendix A

The Relative Value Iteration Algorithm

The relative value iteration algorithm is a variation of the famous value iteration algorithm. This appendix gives the details of its implementation in visual C++. The *parallel_for* function from the new Microsoft concurrency namespace is used to speed up different parts of the algorithm. This implementation was used for computing the results in Chapter 7.

A.1 Matrix

```
template <class T>
class Matrix
{
public:
    typedef std::map<size_t, std::map<size_t , T> > mat_t;
    typedef typename mat_t::iterator row_iter;
    typedef std::map<size_t, T> col_t;
typedef typename col_t::iterator col_iter;
Matrix(size_t i){ m=i; n=i; }
Matrix(size_t i, size_t j){ m=i; n=j; }
inline T& operator()(size_t i, size_t j)
{
    if(i>=m || j>=n) throw;
    return mat[i][j];
}
inline T operator()(size_t i, size_t j) const
{
    if(i>=m || j>=n) throw;
    return mat[i][j];
}
std::vector<T> operator*(const std::vector<T>& x)
{  //Computes y=A*x
    if(this->m != x.size()) throw;
    std::vector<T> y(this->m);
    T sum;
    row_iter ii;
    col_iter jj;
    for(ii=this->mat.begin(); ii!=this->mat.end(); ii++){
        sum=0;
        for(jj=(*ii).second.begin(); jj!=(*ii).second.end(); jj++){
            sum += (*jj).second * x[(*jj).first];
        }
        y[(*ii).first]=sum;
    }
```

```
        return y;
    }
    void printMat()
    {
        row_iter ii;
        col_iter jj;
        for(ii=this->mat.begin(); ii!=this->mat.end(); ii++){
            for( jj=(*ii).second.begin(); jj!=(*ii).second.end(); jj++){
                std::cout << (*ii).first << ' ';
                std::cout << (*jj).first << ' ';
                std::cout << (*jj).second << std::endl;
            }
        } std::cout << std::endl;
    }
protected:
    Matrix(){}
private:
    mat_t mat;
    size_t m;
    size_t n;
};
template <class T>
class Matrix3D
{
public:
typedef std::map<size_t, std::map<size_t, std::map<size_t , T> > > mat_t;
    Matrix3D(size_t k, size_t i, size_t j){ a=k; m=i; n=j; }
    //mat[#Actions][#Rows][#Cols]
T getElement(size_t a, size_t i, size_t j)
{
return mat[a][i][j];
}
void setElement(size_t a, size_t i, size_t j, T value)
{
mat[a][i][j] = value;
}
private:
    mat_t mat;
size_t a; //#Actions
    size_t m; //#Rows
    size_t n; //#Columns
};
template <class T>
class Matrix2D
{
public:
typedef std::map<size_t, std::map<size_t , T> > mat_t;
  Matrix2D(size_t i, size_t j){ m=i; n=j; } //mat[#Rows][#Cols]
T getElement(size_t i, size_t j)
{
return mat[i][j];
}
void setElement(size_t i, size_t j, T value)
{
```

```
mat[i][j] = value;
}
private:
    mat_t mat;
size_t m; //#Rows
    size_t n; //#Columns
};
```

A.2 Definitions

```
#include <stdlib.h>
#include <iostream>
#include <map>
#include <string>
#include <cstdlib>
#include <vector>
#include <iomanip>
#include <sstream>
#include <ppl.h>
#include "Matrix.h"
using namespace std;
using namespace Concurrency;

int T = 2;
int E0 = 2;
int W = 2; //Number of channel states (Not amount of transmission energy)
double epsilon = 0.01;
double ChTransMat[2][2] = {{0.3 , 0.7} , {0.2 , 0.8}};
//Transition Probability Matrix for the Channel
int TransEnergy[3] = {0, 2, 3};
//State 1 of channel requires 2 units of energy
typedef struct _SystemState
{
int T1, E1, W1, T2, E2, W2, T3, E3, W3;
}SystemState;
map<int, SystemState*> StateTable;
map<string, int> IndexTable;
SystemState* CloneSystemState(SystemState* s)
{
SystemState* new_s = (SystemState*) malloc(sizeof(SystemState));
new_s->T1 = s->T1;
new_s->E1 = s->E1;
new_s->W1 = s->W1;
new_s->T2 = s->T2;
new_s->E2 = s->E2;
new_s->W2 = s->W2;
new_s->T3 = s->T3;
new_s->E3 = s->E3;
new_s->W3 = s->W3;
return(new_s);
}
double GetStateTransProb(SystemState* ps, SystemState* ns)
//Get the state transition probability
{
```

```
return(ChTransMat[ps->W1-1][ns->W1-1]
* ChTransMat[ps->W2-1][ns->W2-1]
* ChTransMat[ps->W3-1][ns->W3-1]);
}
bool IsTerminatingState(SystemState* s)
{
if(s->T1 >= T || s->T2 >= T || s->T3 >= T || s->E1 <= 0
|| s->E2 <= 0 || s->E3 <= 0 || s->E1 < TransEnergy[s->W1]
|| s->E2 < TransEnergy[s->W2] || s->E3 < TransEnergy[s->W3])
return true;
else
return false;
}
SystemState* ApplyAction(SystemState* s, int a)
{
SystemState* ns = CloneSystemState(s);
if(a == 1)
{
ns->T1 = ns->T1 + TransEnergy[ns->W1];
ns->E1 = ns->E1 - TransEnergy[ns->W1];
ns->T2 = ns->T2 + TransEnergy[ns->W1];
if(ns->T3 > 0)
ns->T3 = ns->T3 - 1;
}
else if(a == 2)
{
ns->T2 = ns->T2 + TransEnergy[ns->W2];
```

```
ns->E2 = ns->E2 - TransEnergy[ns->W2];
ns->T1 = ns->T1 + TransEnergy[ns->W2];
ns->T3 = ns->T3 + TransEnergy[ns->W2];
}
else if(a == 3)
{
ns->T3 = ns->T3 + TransEnergy[ns->W3];
ns->E3 = ns->E3 - TransEnergy[ns->W3];
ns->T2 = ns->T2 + TransEnergy[ns->W3];
if(ns->T1 > 0)
ns->T1 = ns->T1 - 1;
}
return(ns);
}
map<int, SystemState*> GetNextStates(SystemState* s)
{
map<int, SystemState*> NS; //Set of next states given a base state
int index = 0;
for(int w1=1;w1<=W;w1++)
{
for(int w2=1;w2<=W;w2++)
{
for(int w3=1;w3<=W;w3++)
{
index++;
SystemState* ns = CloneSystemState(s);
ns->W1 = w1; ns->W2 = w2; ns->W3 = w3;
NS[index] = ns;
}
}
}
return(NS);
}
string GetSystemStateID(SystemState* s)
{
stringstream ss;
    ss.str("");
ss << s->T1;
ss << s->E1;
ss << s->W1;
ss << s->T2;
ss << s->E2;
ss << s->W2;
ss << s->T3;
ss << s->E3;
ss << s->W3;
   return(ss.str());
}
double GetMax(vector<double> v)
{
double tmp = v[0];
for(int i=1;i<v.size();i++)
{
if(v[i] > tmp)
```

```
{
tmp = v[i];
}
}
return tmp;
}
double CalcVariation(vector<double> Unext, vector<double> U)
{
double diff, MaxV = -99999.99, MinV = 99999.99;
for(int i=1;i<U.size();i++)
{
diff = Unext[i] - U[i];
if(diff > MaxV)
MaxV = diff;
else if(diff < MinV)
MinV = diff;
}
return(MaxV-MinV);
}
```

A.3 Program

```
#include "defs.h"
int main()
{
int index = 0;
int NumStates, IndexOfTerminatingState;
```

```
//Buidling the State Table

for(int t1=0;t1<T;t1++)

{

for(int t2=0;t2<T;t2++)

{

for(int t3=0;t3<T;t3++)

{

for(int e1=1;e1<=E0;e1++)

{

for(int e2=1;e2<=E0;e2++)

{

for(int e3=1;e3<=E0;e3++)

{

for(int w1=1;w1<=W;w1++)

{

for(int w2=1;w2<=W;w2++)

{

for(int w3=1;w3<=W;w3++)

{

if(e1 >= TransEnergy[w1] && e2 >= TransEnergy[w2]

&& e3 >= TransEnergy[w3])

{

index++;

StateTable[index] = (SystemState*) malloc(sizeof(SystemState));

StateTable[index]->T1 = t1;

StateTable[index]->E1 = e1;

StateTable[index]->W1 = w1;
StateTable[index]->T2 = t2;
StateTable[index]->E2 = e2;
StateTable[index]->W2 = w2;
StateTable[index]->T3 = t3;
StateTable[index]->E3 = e3;
StateTable[index]->W3 = w3;
string id = GetSystemStateID(StateTable[index]);
IndexTable[id] = index;
}}}}}}}}}
NumStates = index + 1;
IndexOfTerminatingState = index + 1;
Matrix3D<double>* P = new Matrix3D<double>(3,NumStates,NumStates);
Matrix3D<double>* R = new Matrix3D<double>(3,NumStates,NumStates);
Matrix3D<int>* NextStatesIndexes = new Matrix3D<int>(3,NumStates,9);
P->setElement(1,IndexOfTerminatingState, IndexOfTerminatingState, 1.0);
P->setElement(2,IndexOfTerminatingState, IndexOfTerminatingState, 1.0);
P->setElement(3,IndexOfTerminatingState, IndexOfTerminatingState, 1.0);
R->setElement(1,IndexOfTerminatingState, IndexOfTerminatingState, 0.0);
R->setElement(2,IndexOfTerminatingState, IndexOfTerminatingState, 0.0);
R->setElement(3,IndexOfTerminatingState, IndexOfTerminatingState, 0.0);
//Computing P & R
parallel_for(1, NumStates, 1, [&](int i) {
SystemState* ps = StateTable[i];
for(int a = 1; a <= 3; a++)
{
SystemState* base_state = ApplyAction(ps, a);
map<int, SystemState*> NS = GetNextStates(base_state);
```

```
bool FoundTerminatingState = false;
NextStatesIndexes->setElement(a, i, 0, 0);
for(int j=1;j<=8;j++)
{
if(IsTerminatingState(NS[j]))
{
double tmp = P->getElement(a, i, IndexOfTerminatingState);
tmp = tmp + GetStateTransProb(ps, NS[j]);
P->setElement(a, i, IndexOfTerminatingState, tmp);
R->setElement(a, i, IndexOfTerminatingState, 1.0);
if(!FoundTerminatingState)
{
FoundTerminatingState = true;
int count = NextStatesIndexes->getElement(a, i, 0);
count++;
NextStatesIndexes->setElement(a, i, 0, count);
NextStatesIndexes->setElement(a, i, count, IndexOfTerminatingState);
}
}
else
{
int count = NextStatesIndexes->getElement(a, i, 0);
count++;
NextStatesIndexes->setElement(a, i, 0, count);
string id = GetSystemStateID(NS[j]);
int index_ns = IndexTable[id];
NextStatesIndexes->setElement(a, i, count, index_ns);
double prob = GetStateTransProb(ps, NS[j]);
P->setElement(a, i, index_ns, prob);
//P->setElement(a, i, index_ns, 0.125);
R->setElement(a, i, index_ns, 1.0);
}
}
free(base_state);
}
});
Matrix2D<double>* ARM = new Matrix2D<double>(NumStates, 3);
bool done = false;
for(int a=1;a<=3;a++){
parallel_for(1, NumStates+1, 1, [&](int row) {
double v = 0.0;
for(int col=1;col<=NextStatesIndexes->getElement(a, row, 0);col++)
{
int ns_index = NextStatesIndexes->getElement(a, row, col);
v = v + P->getElement(a, row, ns_index) * R->getElement(a, row, ns_index);
}
ARM->setElement(row, a, v);
});
}
vector<double> U(NumStates+1);
vector<double> Unext(NumStates+1);
vector<int> Policy(NumStates+1);
vector<double> Q1(3);
Matrix2D<double>* Q2 = new Matrix2D<double>(NumStates, 3);
```

```
double g, variation;

while(!done)

{

parallel_for(1, 4, 1, [&](int a) {

double v = 0.0;

for(int i=1;i<=NumStates;i++)

{

v = v + P->getElement(a, IndexOfTerminatingState, i) * U[i];

}

Q1.push_back(ARM->getElement(IndexOfTerminatingState,a) + v);

});

g = GetMax(Q1);

parallel_for(1, NumStates+1, 1, [&](int i) {

double MaxValue = -999999.99;

int BestAction = 0;

for(int a=1;a<=3;a++)

{

double v = 0.0;

for(int col=1;col<=NextStatesIndexes->getElement(a, i, 0);col++)

{

int ns_index = NextStatesIndexes->getElement(a, i, col);

v = v + P->getElement(a, i, ns_index) * U[ns_index];

}

if(ARM->getElement(i,a)+v > MaxValue)

{

MaxValue = ARM->getElement(i,a)+v;

BestAction = a;
}
}
Unext[i] = MaxValue - g;
Policy[i] = BestAction;
});
variation = CalcVariation(Unext, U);
if(variation < epsilon)
done = true;
else
U = Unext;
}
SystemState* s = (SystemState*) malloc(sizeof(SystemState));
s->T1 = 0;
s->E1 = E0;
s->W1 = 1;
s->T2 = 0;
s->E2 = E0;
s->W2 = 1;
s->T3 = 0;
s->E3 = E0;
s->W3 = 1;
string id = GetSystemStateID(s);
index = IndexTable[id];
printf("\nExpected System Lifetime = %f", U[index]);
return 0;
}
```

Appendix B

Essential Functions

This appendix shows the Matlab implementations for some essential concepts and algorithms.

B.1 Computing SAR in the Near Field

```
function v = SAR
F = 2 * 10^6; % Frequency
sigma = 0.5476;
omega = 2 * pi * F;
mu = 1; % permeability
epsilon = 826;
rho = 1040;
I = 0.01; % Sinusoidal current driving the antenna (Ampers)
dl = 0.04; % Length of antenna in meters
% Attenuation constant
alpha = omega * sqrt((mu * epsilon / 2)
   * (sqrt(1 + (sigma / omega * epsilon)^2) - 1));
% Phase constant
```

```
beta = omega * sqrt((mu * epsilon / 2)
   * (sqrt(1 + (sigma / omega * epsilon)^2) + 1));
e = 2.7182;
R = 0.02;
gamma = sqrt(alpha^2 + beta^2);
v = ((sigma * mu * omega)
   / (rho * sqrt(sigma^2 + epsilon^2 * omega^2)))
   * (((I * dl * e^(-1 * alpha * R)) / 4 * pi)
   * ((1 / R^2) + gamma / R))^2;
```

B.2 Computing the Temperature Increase

```
function dT = Temperature
delta_t = 10;   % Time step of FDTD
b = 2700;       % Blood perfusion constant
rho = 1040;     % Mass density
Cp = 3600;      % Specific heat of the tissue
K = 0.498;      % Thermal conductivity
delta = 0.005;  % Cell size
Tb = 37;        % Fixed blood temperature
Pc = 0.02;       % Power dissipation of circuitry (mW)
SAR = 2.5205e-106;     % Specific Absorption Rate (W/Kg)
Tn = 37;        % Normal human body temperature
T(1) = 37;
for n = 1:1000
    T(n+1) = (1 - ((delta_t * b) / (rho * Cp))
     - (4 * delta_t * K) / (rho * Cp * delta^2)) * T(n)
    + (delta_t / Cp) * SAR + ((delta_t * b) / (rho * Cp))
    * Tb + (delta / (rho * Cp)) * Pc
    + ((delta_t * K) / (rho * Cp * delta^2)) * (4 * Tn);
end
dT = T(1000) - Tn;
```

B.3 Computing the State Transition Probabilities for the Wireless Channel

```
% N = Number of channel states
% SNR = SNR thresholds
% Example:
%       N = 5
%       SNR = [0.05 2.02 3.65 5.05 6.27];
function p = ComputeChannelStateTransProb(N, SNR)
p = zeros(N);
Tp = 0.003; % Packet duration (3 msec)
Ps = 1 / N;  % Steady-state probability for each channel state
fd = 60;     % Doppler frequency (Hz)
for i = 1:N
    if i == 1
        p(i, i+1) = ((sqrt(2 * pi * SNR(i+1)) * fd
         * exp(-1 * SNR(i+1))) * Tp) / Ps;
        p(i,i) = 1 - p(1, 2);
    elseif i == N
        p(i, i-1) = ((sqrt(2 * pi * SNR(i)) * fd
         * exp(-1 * SNR(i))) * Tp) / Ps;
        p(i, i) = 1 - p(N, N-1);
    else
        p(i, i+1) = ((sqrt(2 * pi * SNR(i+1)) * fd
         * exp(-1 * SNR(i+1))) * Tp) / Ps;
        p(i, i-1) = ((sqrt(2 * pi * SNR(i)) * fd
         * exp(-1 * SNR(i))) * Tp) / Ps;
        p(i, i) = 1 - (p(i, i+1) + p(i, i-1));
    end
end
```

B.4 Computing the Transition and Reward Matrices for the MDP in §7.2

```
P = cell(1, Num_Sensors);
R = cell(1, Num_Sensors);
parfor a = 1:Num_Sensors
    Prob = sparse(Num_System_States, Num_System_States);
    Reward = sparse(Num_System_States, Num_System_States);
    for i = 1:Num_System_States
        if(i == IndexTerminatingState)
            Prob(i, i) = 1.0;
            Reward(i, i) = 0.0;
        elseif(i == IndexFinalValidState)
            Prob(i, IndexTerminatingState) = 1.0;
            Reward(i, IndexTerminatingState) = 1.0;
        else
            ps = StateTable{i, 1};
            base_state = ApplyAction(ps, a);
            NextStates = GetNextStates(base_state);
            for j = 1:size(NextStates,1)
                ns = NextStates{j, 1};
                ns_index = GetIndex(ns, StateTable, StateIndex);
                Prob(i, ns_index) = Prob(i, ns_index)
                 + NSTransProb(ps, ns, ChTransMat);
                Reward(i, ns_index) = 1.0;
            end
        end
    end
    P{1, a} = Prob;
    R{1, a} = Reward;
end
```

B.5 Computing the Transition and Reward Matrices for the MDP in §8.2

```
for i = 1:size(StateTable,1)
    for a = 1:A   % A = Num_Actions
      % Check if action a can be performed at state i
        if(Action(i,a) == true)
            ps = StateTable{i, 1};
            BaseState = ApplyAction(ps, a);
            for j = 1:W   % W = Num_Chan_States
                BaseState(3) = j;
                index = GetStateIndex(BaseState, StateTable);
                P(i, index, a) = ChanStateTranProb(ps(3), j);
                if(a == 1)  % Sample
                    R(i, index, a) = 1;
                else
                    R(i, index, a) = 0;
                end
            end
        else
            P(i, i, a) = 1;
            R(i, i, a) = 0;
        end
    end
end
```

B.6 Q-Learning

```
NumActions = 3;
Q = zeros(NumStates, NumActions);
N = 10000; % Number of iterations (i.e., steps) in each episode
NumEpisodes = 100;
epsilon = 0.5;
alpha = 0.2; % Learning rate
gamma = 0.1; % Discount factor
for i = 1:NumEpisodes
    ps = [0 E0 1]; % Initial state
    for j = 1:N
        ps_index = GetStateIndex(ps, StateTable);
        if(rand <= 1 - epsilon)
            [v a] = max(Q(ps_index, :));
        else
            a = randi(NumActions);
        end
        while(Action(ps_index,a) == false)
           if(rand <= 1 - epsilon)
                [v a] = max(Q(ps_index, :));
           else
                a = randi(NumActions);
           end
        end
        if(a == 1)
            r = 1;
        else
            r = 0;
        end
        base_state = ApplyAction(ps,a);
        base_state(3) = randi(W); % W = Num_Chan_States
        ns = base_state;
        ns_index = GetStateIndex(ns, StateTable);
        Q(ps_index, a) = Q(ps_index, a) + alpha
         * (r + gamma * max(Q(ns_index, :)) - Q(ps_index, a));
        ps = ns;
    end
end
[QVlaue QPolicy] = max(Q, [], 2);
```

B.7 Network Lifetime Simulation

```
NumSimRuns = 1000;

LifeTime = zeros(NumSimRuns,1);

for i = 1:NumSimRuns

    % Initial network state (three biosensors)

    ps = [0 E0 1 0 E0 1 0 E0 1];

    while(~IsTerminatingState(ps))

        ps_index = GetIndex(ps, StateTable);

        a = Policy(ps_index);

        ns = ApplyAction(ps, a);

        % Update state of wireless channel for each biosensor

        ns(3) = Next_Channel_State(1);

        ns(6) = Next_Channel_State(2);

        ns(9) = Next_Channel_State(3);

        ps = ns;

        LifeTime(i) = LifeTime(i) + 1;

    end

end

fprintf('-> Average Simulated Lifetime (Optimal Policy) = %f\n',

   sum(LifeTime)/NumSimRuns);
```