國立臺灣大學管理學院資訊管理學研究所

博士論文

Department of Information Management

College of Management

National Taiwan University

Doctoral Dissertation

應用於無線感測網路中之物體監控與追蹤演算法

Object Monitoring and Tracking Algorithms in Wireless Sensor Networks

李政達

Cheng-Ta Lee

指導教授：林永松 博士

Advisor: Frank Yeong-Sung Lin, Ph.D.

中華民國 99 年 7 月

July, 2010

應用於無線感測網路中之物體監控與追蹤演算法

Object Monitoring and Tracking Algorithms in Wireless Sensor Networks

By Cheng-Ta Lee

A dissertation submitted to

the Graduate School of Information Management

of National Taiwan University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

July, 2010

國立臺灣大學博士學位論文
口試委員會審定書

應用於無線感測網路中之物體監控與追蹤演算法

Object Monitoring and Tracking Algorithms in Wireless Sensor Networks

本論文係 李政達 君（學號 D90725001）在國立臺灣大學資訊管理學系、所完成之博士學位論文，於民國九十九年七月十九日承下列考試委員審查通過及口試及格，特此證明

口試委員：[illegible]

[illegible] [illegible]

[illegible] [illegible]

[illegible] [illegible]

所　　長：[illegible]

謝辭

「即使是一株雜草也可以學習開花」，這是李小平教授送給我《慧眼初開》一書裡面的一句話，正說明了我前半生的人生路程：從小至今一直是在挫折中成長，每遇到難關時，除了靠自己樂天的個性之外，最重要的是身旁有一群關心與鼓勵我的家人、師長以及朋友，讓我能克服困難，繼續前進。

回顧這九年來博士班的求學歷程，要感謝許多人的鼓勵與幫忙。首先感謝恩師林永松教授，恩師在學術研究上認真的態度，對學生細心指導的付出，還有對學生身教上的典範，不僅將我帶入學術領域的最高殿堂，也是我從事教職的最佳表率。此外，還有孫雅麗老師、謝清佳老師、李瑞庭老師、莊裕澤老師、蔡益坤老師、陳靜枝老師與梁定澎老師，我在每一位老師的課堂上都獲益良多，在此致上我深深的謝意。

再者，感謝擔任口試委員的孫雅麗教授、呂俊賢教授、林一平教授、莊東穎教授、趙啟超教授和鐘嘉德教授，由於教授們的細心指導與建議，使我的博士論文益臻完善。此外，還要感謝王國光教授在一些數學方面的問題，適時地為我解惑，以及洪瑞文老師與許宏仁老師在英文潤稿上的協助。

另外，我還要感謝博碩士班同門師兄弟姊妹們：顏宏旭、林志浩、祝國忠、鄭旭成、彭國維、邱佩玲、溫演福、臧柏皓、林俊甫、許明宗、陳霈語、陳澤龍、黃健誠、蕭邱漢、王猷順、林書平、林岦毅、許宴毅等，及其他博士班的同學們，你們每一位都給了我許多的幫忙，謝謝你們。

我更要感謝我的外婆、父母親和阿姨，沒有你們辛勞的培育和教養，我不可能會有今日的一點成就。在此，也感謝愛妻平雯，由於你的全心體諒與幫忙，才能讓我無後顧之憂來攻讀博士；還有愛子尚澔，在博士班最後階段帶給我無限的歡笑與求學研究的動力，在此特別感謝家人在背後的支持。此外還要感謝愛心爸爸游火金、愛心媽媽林清翠與愛心奶奶李詠妤對尚澔的照顧，讓我有更多時間來撰寫論文。

最後，謹以本論文獻給往生的父親，感謝您從小到大對我的呵護，如今我身為人父，更能體會您對我的關心與愛心，願您能一同分享這份喜悅。

李政達 謹識

于國立臺灣大學資訊管理學研究所

中華民國九十九年七月

論文摘要

論文題目：應用於無線感測網路中之物體監控與追蹤演算法
作者：李政達　　　　　　　　　　　　　　　　　　　　　　九十九年七月
指導教授：林永松 博士

無線感測網路(Wireless Sensor Networks, WSNs)的規劃設計上，有兩個重要的研究議題：首先是如何建構一個能滿足服務品質之應用需求的無線感測網路，第二則是如何延長無線感測網路之生命期。從應用的觀點來看，改善服務品質之需求，必須考慮無線感測網路對於應用的支援，如環境監測、物體入侵監控與物體追蹤等的能力。此外，由於感測器的電力有限，一般而言，很難再充電，故如何延長無線感測網路的生命期，也是規劃無線感測網路的重要議題。

在本論文中，我們提出物體監控與追蹤相關應用服務之演算法，首先我們發展了五個演算法，是先將問題描述為數學最佳化模型，這些都是複雜的無線感測網路規劃問題，我們採用啟發式、系統模擬與拉格蘭日鬆弛法來解決這一系列最佳化問題。此外，我們發展了一個以預測為基礎的演算法來支援物體追蹤服務。茲將每一研究主題之內容與成果簡述如下：

- 在邊緣監控(boundary monitoring)服務中，我們提出 BMAFS 與 BMAMS 演算法來支援該服務，BMAFS 演算法是考慮在一個任意拓樸的無線感測網路中，找出監控範圍（monitoring region）之邊緣感測器節點（boundary nodes），為了滿足生命期最大化之服務品質，即以電能效率(energy efficiency)為考量，配置具有 *k* 個群(group)的無線感測網路之邊緣感測器節點，每個群輪流支援入侵監測服務。實驗結果顯示，此機制可以有效地延長無線感測網路入侵監測服務之生命期。在上述研究議題中，我們將容錯的問題考量進來，BMAMS 演算法考慮當有節點故障或是節點電力耗盡時導致檢核點未被覆蓋，我們可以規劃將鄰近的節點移動覆蓋至未被覆蓋之系統檢核點(check points)，使得整個網路的服務不致於中斷。
- 我們亦將縱深防禦(in-depth defense)的觀念加諸在上述研究議題中，亦即同

時考量有縱深的監控範圍，當有入侵者進入監控範圍時，提供早期預警的功能，讓防禦者有更充裕的時間來因應，此問題並同時將整體系統的防禦率加入整個縱深防禦系統的規劃中，在此我們提出 LDA 與 NLDA 演算法來支援縱深防禦服務，LDA 演算法配置具有 *k* 個群(group) 的無線感測網路之多層監控範圍節點，每個群輪流支援入侵監測服務。實驗結果顯示，此機制可以有效地延長無線感測網路入侵監測服務之生命期。此外，NLDA 演算法是將一個入侵情境轉化成數學規劃問題，用以描述系統之整體防禦率與早期預警率，並且透過三階段的評估流程找出能有效地延長無線感測網路入侵監測服務之生命期之群組配置模式。此外，該法能夠用於解決具備不完美資訊特質的問題，透過適當的情境描述，加入隨機的變異性情況，使問題更貼近於真實情況，有效地提升縱深防禦系統之生命期。

- 在物體追蹤(object tracking)服務中，我們提出 TOTA 與 POTA 演算法來支援該服務，TOTA 演算法是一個以樹為基礎(tree-based)的物體追蹤演算法，此研究的實驗結果顯示，所提演算法不但可得到高品質的解，且具有效力(effectiveness)、擴展性(scalability)與強固性(robustness)。另外，我們亦發展 POTA 演算法以動態預測為基礎(dynamic prediction-based)的物體追蹤演算法，此法使用喚醒較少的節點來進行物體追蹤，並利用動態預測模式來提升預測的準確性，利用此機制可以有效地延長感測網路物體追蹤服務之生命期。

由實驗結果顯示，我們所提出的六個演算法均可有效地支援物體監控與追蹤之相關應用服務。

關鍵詞：無線感測網路、物體監控、入侵偵測、縱深防禦、物體追蹤、服務品質、電能效率、系統模擬、拉格蘭日鬆弛法、數學規劃、網路最佳化

Dissertation Abstract

Object Monitoring and Tracking Algorithms in Wireless Sensor Networks

By Cheng-Ta Lee
July, 2010
ADVISER: Dr. Frank Yeong-Sung Lin

There are two important challenges in WSNs design. One is to construct an efficient WSN for applications to guarantee desired quality of service (QoS). The other challenge is to prolong the lifetime of WSNs. From application viewpoint, the abilities of environment surveillance, object intrusion detection, and object tracking have to support the QoS. Besides, it is difficult to recharge or replace the battery for numerous sensors in the most scenarios. Therefore, how to prolong the lifetime of WSNs also becomes a key issue.

In this dissertation, we focus on the network planning problem to support object monitoring and object tracking services from various perspectives. We develop five algorithms to solve optimization problems based on Lagrangean relaxation method, simulation techniques, and heuristic approaches. In addition, we develop one prediction-based algorithm based on modified Viterbi algorithm to solve object tracking problem. We present each topic briefly as follows:

- For boundary monitoring problem, we propose two algorithms, BMAFS and BMAMS, to support boundary monitoring services. The BMAFS is to construct boundary monitoring for grouping capabilities, and it tries to find the maximum *k* groups of sensors for boundary monitoring of the sensor field to prolong the system lifetime. In the test problems, the experiment results show that the proposed algorithm achieves optimality in the boundary monitoring for grouping

capabilities. The BMAMS is to address the problem of boundary node relocation, and it can move previously deployed sensors to cover uncovered check points due to failure of other nodes or battery exhaustion of other nodes. The mechanism can further prolong the system lifetime. The experiment results show that the proposed BMAMS gets effectiveness in the boundary monitoring services for mobile and grouping capabilities.

- For in-depth defense problem, we propose two algorithms, LDA and NLDA, to support in-depth defense services. The LDA is to construct layered defense for wireless sensor networks of grouping capabilities. It tries to find the maximum *k* groups of sensors for layered defense of the monitoring region to prolong the system lifetime. The experiment results show that the proposed LDA gets efficiency in the layered defense for grouping capabilities. The NLDA is to construct non-layered defense of supporting different types of intruders for grouping capabilities, and it tries to find the maximum *k* groups of sensors for non-layered defense subject to the constraints of defense rate, early warning rate, battery capacity, intruder behaviors, and defender strategies. The NLDA can prolong the system lifetime and provide lead time alarms. The experiment results show that the proposed NLDA gets applicability and effectiveness in the non-layered defense services of supporting different types of intruders for grouping capabilities.
- For object tracking problem, we propose two algorithms, TOTA and POTA, to support object tracking services. The TOTA is to construct an object tracking tree for object tracking. Such tree-based algorithm can achieve energy-efficient object tracking for given arbitrary topology of sensor networks. The experiment results show that the proposed TOTA gets a near optimization in the energy-efficient object tracking. Furthermore, the algorithm is efficient and scalable in terms of the running time. The POTA is to construct a dynamic prediction-based algorithm for object tracking. Such the POTA can minimize the number of nodes participating in the tracking activities, minimize out of tracking

probability, and maximize the accuracy of object predicted position. The POTA can prolong the system lifetime.

The experiment results show that all six algorithms can support object monitoring and tracking services efficiently.

Keywords: Wireless sensor networks, object monitoring, intrusion detection, in-depth defense, object tracking, quality of services, energy-efficiency, system simulation, Lagrangean relaxation, mathematical modeling, network optimization.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Object monitoring and tracking are important applications in wireless sensor networks (WSNs) since 1) the object monitoring is one important issue for overseeing hostile intrusions and attacks in order to protect the core field; and 2) the object tracking such as tracking of moving objects has many military and civil applications. In these applications, sensor nodes collectively track the movements of moving objects. In this dissertation, we focus on the problem of boundary monitoring, in-depth defense, and object tracking. In this chapter, the motivations of the dissertation are described in Section 1.1; the contributions is presented in Section 1.2; the overview is described in Section 1.3; the research scope presented in Section 1.4; and the dissertation layout is organized in Section 1.5.

1.1 Motivations

Because of fast develop in the wireless sensor networks (WSNs) techniques, from either theoretical or practical perspective are new and important research issues. Numbers of interesting applications for WSNs have been investigated, e.g., environment surveillance, object positioning, object intrusion detection, object tracking, anti-terrorism, and health care. Sensor networks have been forecasted to apply to various usages, such as the civilian and military domains.

There are two important challenges in WSNs design. One is to construct an efficient WSN for applications to guarantee desired quality of service (QoS). The other challenge is to prolong the lifetime of WSNs. From application perspective, the abilities of environment surveillance, object intrusion detection, and object tracking have to support the QoS. Besides, it is difficult to recharge or replace the battery for numerous sensors in the most scenarios. Therefore, how to prolong the lifetime of

WSNs also becomes a key issue.

Therefore, in this dissertation, we focus on the network planning problem to support object monitoring and object tracking services from various perspectives. We develop five algorithms to solve optimization problems based on Lagrangean relaxation method, simulation techniques, and heuristic approaches. In addition, we develop one prediction-based algorithm based on modified Viterbi algorithm to solve object tracking problem.

1.2 Contributions

We summarize the contributions of this dissertation as follows.

1. The interesting issues of object monitoring and tracking services in wireless sensor networks are addressed.
2. We propose two algorithms, BMAFS and BMAMS, to support boundary monitoring services. The BMAFS is to construct boundary monitoring for grouping capabilities. In the test problems, it achieves optimal solutions. The BMAMS is to address the problem of boundary nodes relocation. It can move previously deployed sensors to cover uncovered check points due to failure of other nodes or battery exhaustion of other nodes. The mechanism can further prolong the system lifetime. The experiment results show that the proposed BMAFS and BMAMS get effectiveness in the boundary monitoring services for grouping capabilities.
3. The proposed LDA and NLDA to support in-depth defense services. The LDA is to construct layered defense for wireless sensor networks of grouping capabilities. The NLDA is to construct non-layered defense of supporting different types of intruders for grouping capabilities. The NLDA can prolong the system lifetime and provide lead time alarms. The experiment results show that the proposed LDA and NLDA can improve system lifetime.

4. We propose the TOTA and POTA, we use the TOTA to support tree-based object tracking services. The experiment results show that the proposed algorithm can achieve the near optimal solutions. Furthermore, the algorithm is efficient and scalable in terms of the running time. The POTA is to construct a dynamic prediction algorithm for object tracking. Such prediction-based can minimize the number of nodes participating in the tracking activities, minimize out of tracking probability, and maximize the accuracy of object predicted position in the tracking activities. The experiment results show that the POTA can prolong the system lifetime.
5. At last, in tree-based object tracking and non-layered defense problems, due to their non-linear and non-convex natures, are hard to solve by traditional mathematical programming methods directly. Based on Lagrangean relaxation and simulation methods, we successfully developed heuristic algorithms, TOTA and NLDA, to solve these optimization problems.

1.3 Overview

In an object monitoring and tracking sensor networks, a number of sensor nodes are deployed over a monitoring region with predefined geographical boundaries. The sink (base station) acts as the interface between the sensor networks and applications by issuing commands and collecting the data of interests. A sensor node has the responsibility for objects monitoring and tracking in the monitoring region, and reporting the states of the mobile objects [1][2]. The object monitoring and tracking sensor networks are shown in Figure 1.1.

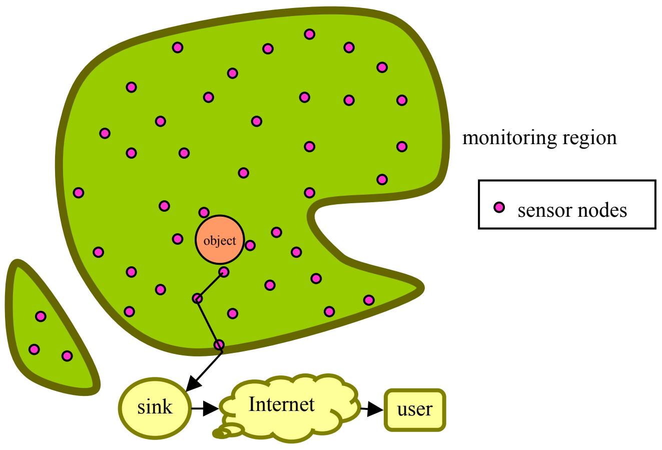


Figure 1.1. An object monitoring and tracking sensor networks.

In some applied circumstances, we just need to record the objects that enter or leave the boundary of monitored area [3][4][5]. For example, the preservation area administrators must be notified while the hunters enter or leave the wildlife preservation area in order to take necessary action. Besides, intrusion detection of enemies is also required to record whether the objects enter or leave the boundary of monitored area for further notification and tracking.

In some other applied circumstances, we need to detect the objects that intrude the safeguard area of in-depth defense [6][7][8][9]. For example, the commander must be notified while the enemies enter the safeguard area of in-depth defense in order to take necessary action. Besides, intrusion detection of enemies is also required to record whether the objects enter monitored area for further notification and tracking. The in-depth defense includes both layered defense and non-layered defense.

In many applications, a wireless sensor network needs to detect, track, and predict mobile objects, and reports the sensing data to sink(s) [10][11][12][13][14][15]. For example, detecting illegal intruders and tracking enemy vehicles in military applications, and tracking the movement of wild animals in wildlife preservation area.

Therefore, there are three main research issues in object monitoring and tracking sensor networks. 1) the boundary monitoring services. It needs to record the objects that enter or leave the boundary of monitored area. Figure 1.2 illustrates a scenario of boundary monitoring for fixed sensor and Figure 1.3 illustrates a scenario of boundary monitoring for mobile sensor. 2) the in-depth defense services. It needs to detect the objects that intrude the safeguard area of in-depth defense. Figure 1.4 and Figure 1.5 illustrate two scenarios of layered defense and non-layered defense. 3) the object tracking services. It needs to detect, track, predict mobile objects, and reports the sensing data to sink. Scenarios of tree-based and prediction-based object tracking are shown in Figure 1.6 and Figure 1.7.

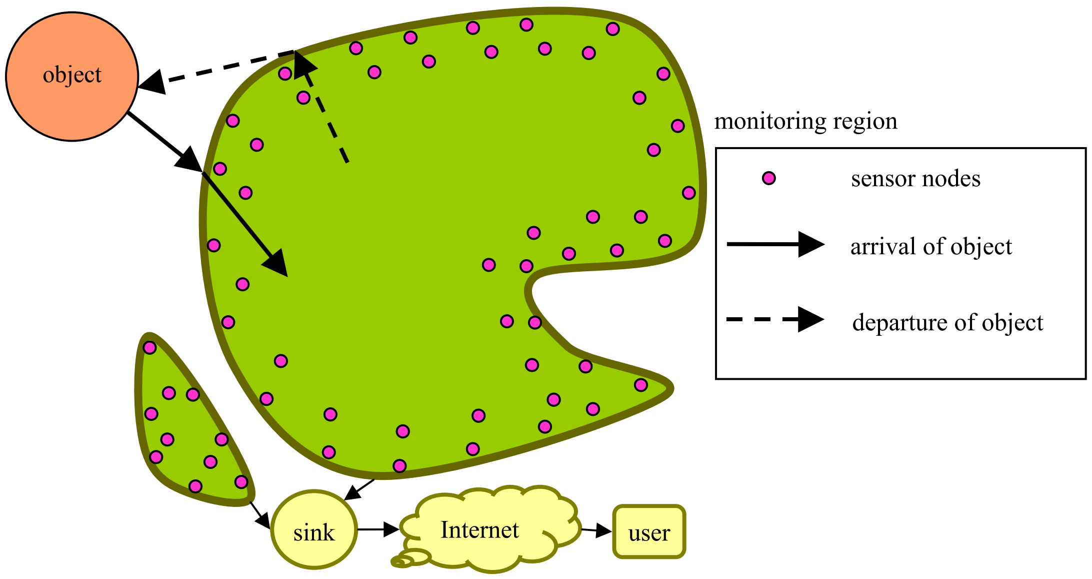


Figure 1.2. A scenario of boundary monitoring services for fixed sensors.

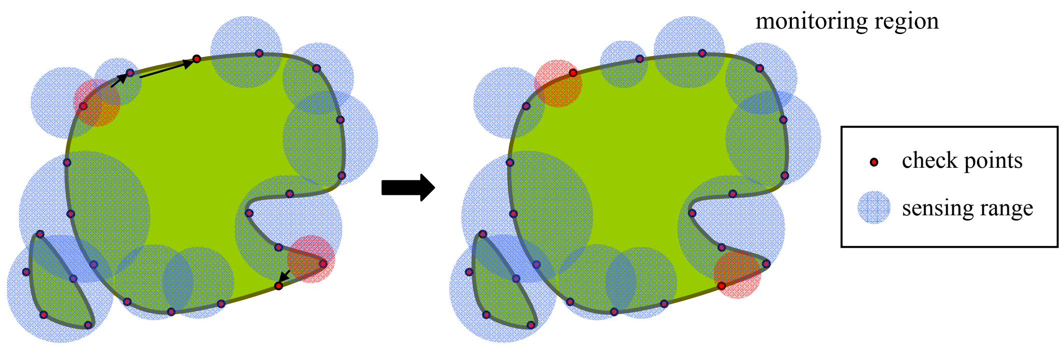


Figure 1.3. A scenario of boundary monitoring for mobile sensors.

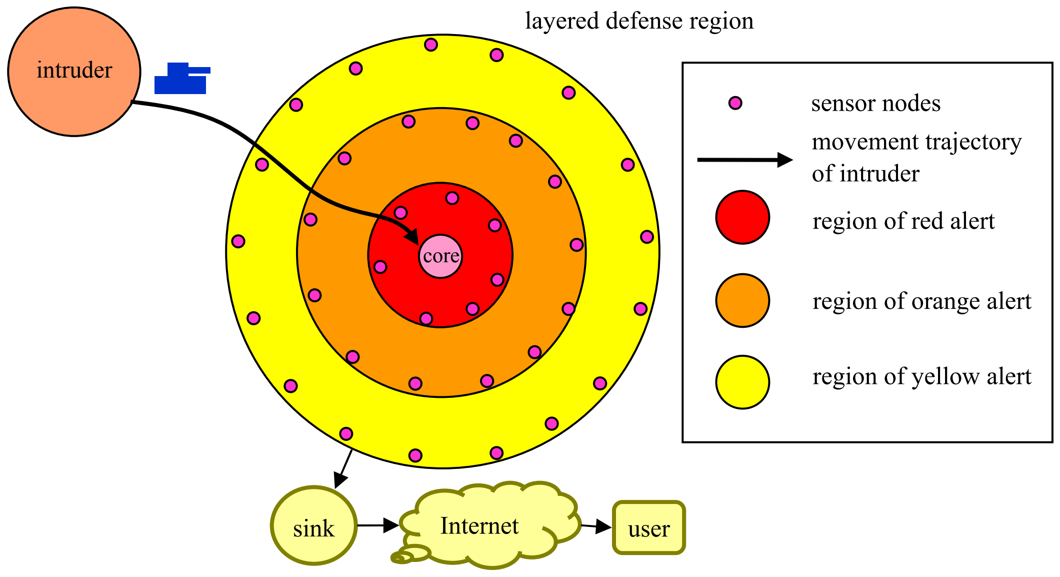
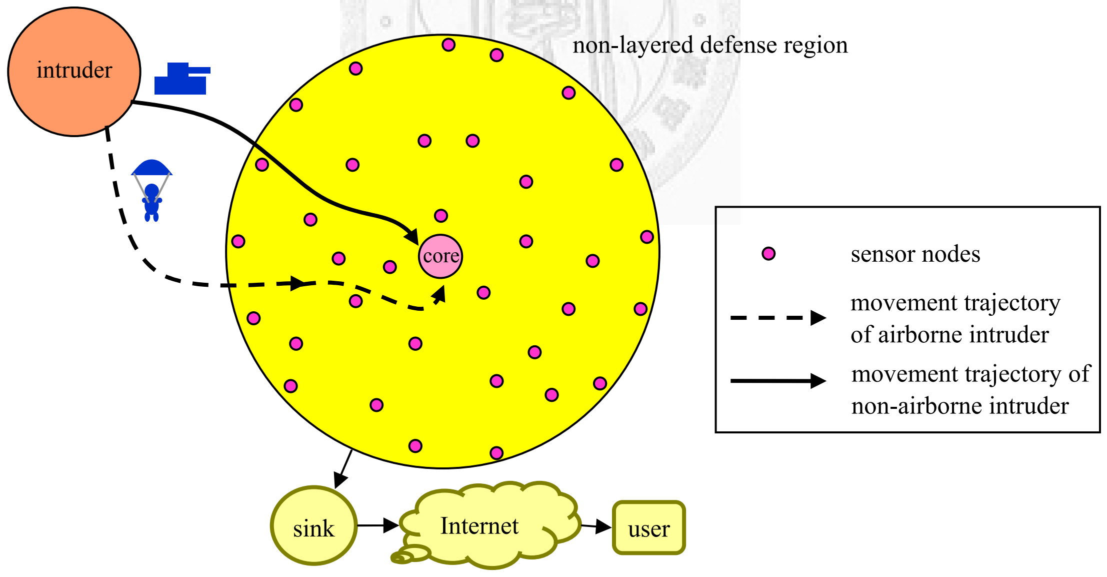


Figure 1.4. A scenario of layered defense services.



intruder
non-layered defense region
core
sensor nodes
movement trajectory of airborne intruder
movement trajectory of non-airborne intruder
sink
Internet
user

Figure 1.5. A scenario of non-layered defense services.

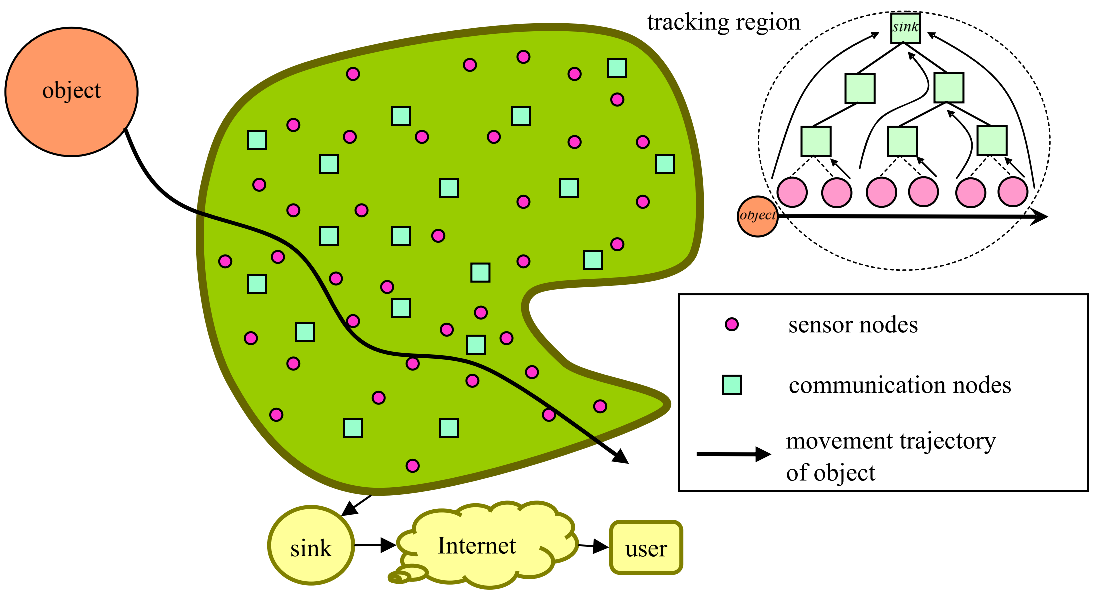


Figure 1.6. A scenario of tree-based object tracking services.

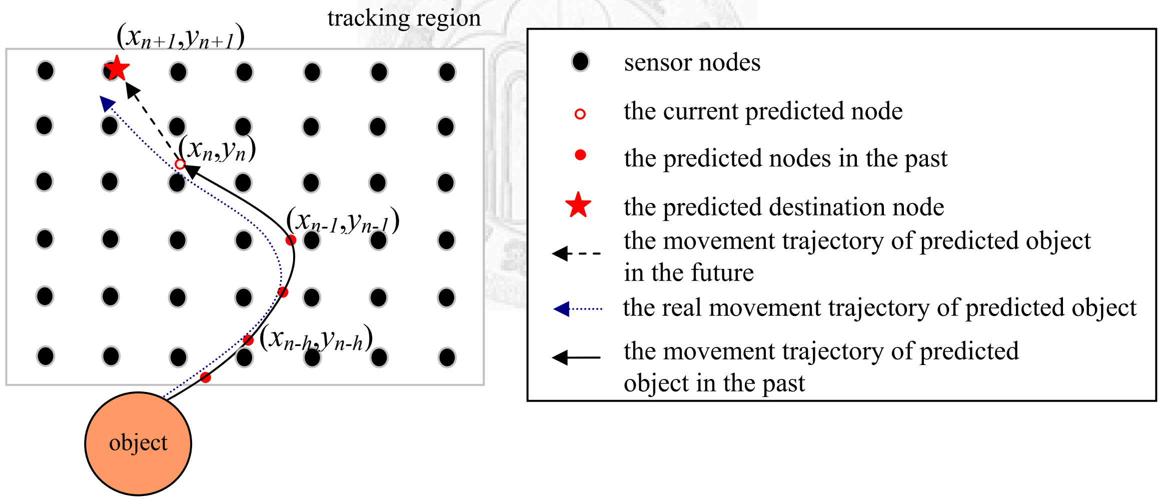

tracking region
(x_{n+1},y_{n+1})
(x_n,y_n)
(x_{n-1},y_{n-1})
(x_{n-h},y_{n-h})
object
sensor nodes
the current predicted node
the predicted nodes in the past
the predicted destination node
the movement trajectory of predicted object in the future
the real movement trajectory of predicted object
the movement trajectory of predicted object in the past

Figure 1.7. A scenario of prediction-based object tracking services.

1.4 Research Scope

In this section, we discuss the object monitoring and tracking problems from types of services, illustrated in Figure 1.8. First is boundary monitoring, it includes boundary nodes grouping algorithm and boundary nodes mobility algorithm. Second is in-depth defense, it includes layered defense algorithm and non-layered defense algorithm. Third is object tracking, it includes tree-based object tracking and prediction-based object tracking.

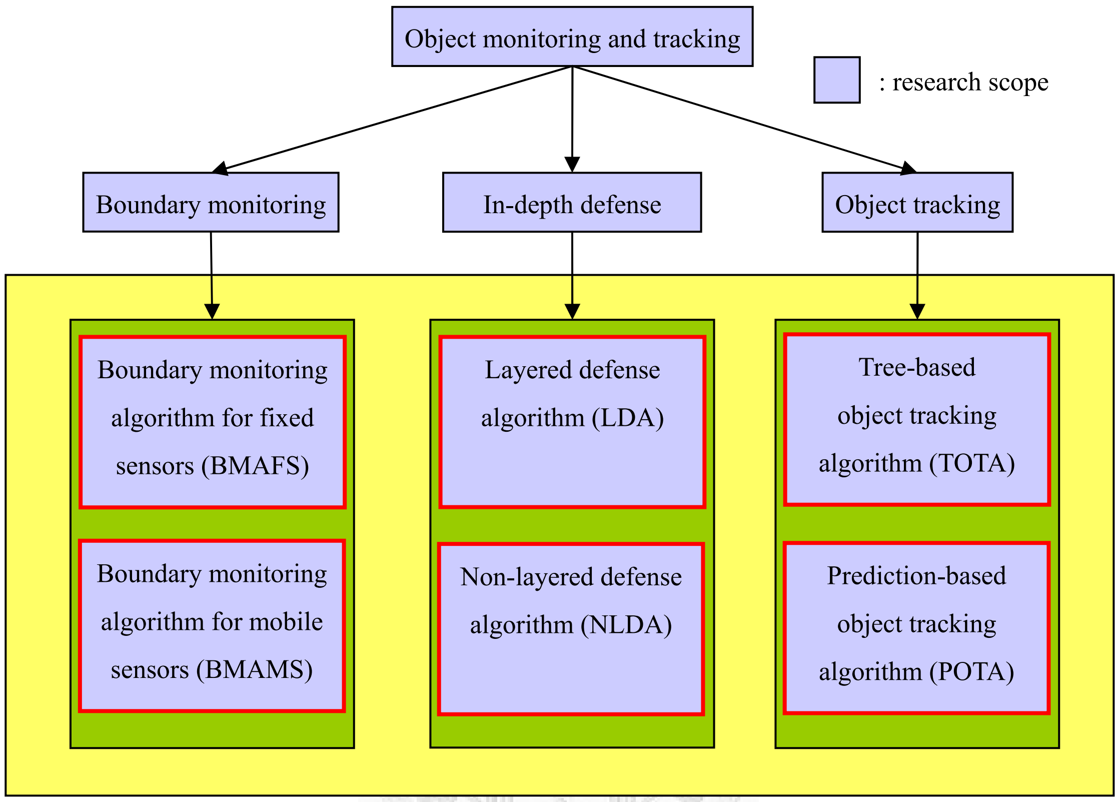

Object monitoring and tracking
: research scope
Boundary monitoring
In-depth defense
Object tracking
Boundary monitoring algorithm for fixed sensors (BMAFS)
Boundary monitoring algorithm for mobile sensors (BMAMS)
Layered defense algorithm (LDA)
Non-layered defense algorithm (NLDA)
Tree-based object tracking algorithm (TOTA)
Prediction-based object tracking algorithm (POTA)

Figure 1.8. Research scope.

In this dissertation, we study several object monitoring and tracking problems (summarized in Table 1.1). Mathematical formulations are used to model these problems. Based on the proposed mathematical models, Lagrangean relaxation, simulation techniques, heuristic approaches, and predicted algorithm are adopted to solve the object monitoring and tracking problems.

Table 1.1. Scope and problem definition of this dissertation.

Problem 1: Boundary monitoring algorithms for fixed sensors	
Given parameters	The set of check points, the set of sensor nodes, initial energy level of each sensor node, energy consumption for sensor nodes to sense data in each round, and detection radius of each sensor.
Constraints	Full coverage boundary check points in each round and battery capacity.
Objective	To maximize the boundary monitoring services lifetime.
To determine	To determine whether sensor *s* is awake or not in the round *r*.
Algorithm	Boundary monitoring algorithm for fixed sensors (BMAFS).
Problem 2: Boundary monitoring algorithm for mobile sensors	
Given parameters	The set of check points, the set of sensor nodes, residual energy level of each sensor node, energy consumption for sensor nodes to sense data in each round, energy consumption for sensor node to move one unit, and detection radius of each sensor.
Constraints	Full coverage of boundary check points in each round and battery capacity.
Objective	To maximize the boundary monitoring services lifetime.
To determine	To determine 1) whether sensor *s* is awake or not in the round *r*, and 2) whether sensor node s moves to cover check point *a* or not.
Algorithms	Boundary monitoring algorithm for mobile sensors (BMAMS).
Problem 3: Layered defense algorithms	
Given parameters	The set of check points, the set of sensor nodes, initial energy level of each sensor node, energy consumption for sensor nodes to sense data in each round, detection radius of each sensor, total number of layers, total defense rate, and the detectability.
Constraints	Defense rate, detectability and battery capacity.
Objective	To maximize the layered defense services lifetime.
To determine	To determine whether sensor *s* is awake or not in the round *r*.

Algorithms	Simple algorithm 1 (SA1), simple algorithm 2 (SA2), and layered defense algorithm (LDA).
Problem 4: Non-layer defense algorithms	
Given parameters	The set of sensor nodes, initial energy level of sensor node, energy consumption for sensor nodes to sense data in each round, the total evaluation number of times for all intruder categories in each round, all possible defense strategies, strategies of an intruder, total defense rate, distance of early warning, early warning rate, false positive rate, false negative rate, and location of core field.
Constraints	Defense rate, distance of early warning, early warning rate, battery capacity, all possible defense strategies, and total evaluation frequency.
Objective	To maximize the non-layered defense services lifetime.
To determine	To determine whether sensor s is awake or not in the round r.
Algorithm	Simple algorithm (SA) and non-layered defense algorithm (NLDA).
Problem 5: Tree-based object tracking algorithm	
Given parameters	The set of sensor nodes, the communication nodes, the set of the object moving frequency, and the set of transmission cost associated with links.
Constraints	Routing, tree, and variable-transformation constraints.
Objective	To minimize the total communication cost.
To determine	Object tracking tree.
Algorithm	Tree-based object tracking algorithm (TOTA).
Problem 6: Prediction-based object tracking algorithm	
Given parameters	The set of sensor nodes and policies of prediction.
Objective	1. To minimize the number of nodes participating in the object tracking. 2. To maximize the accuracy of object predicted position. 3. To minimize out of probability.
To determine	The h value at time interval n.
To predict	The location of object at time interval n.
Algorithm	Prediction-based object tracking algorithm (POTA).

1.5 Dissertation Layout

The remainder of this dissertation is organized as follows. Chapter 2 introduces the background knowledge for object monitoring and tracking in WSNs, and reviews a number of previously proposed approaches of boundary monitoring, in-depth defense, and object tracking in WSNs. In Chapter 3, we present the grouping and mobile algorithms for boundary monitoring. In Chapter 4, we propose two grouping algorithms for in-depth defense. In Chapter 5, we develop the tree-based and prediction-based algorithms for object tracking. Finally, we describe the conclusions and the directions of the future work in Chapter 6.

Chapter 2 Background Knowledge and Literature Survey

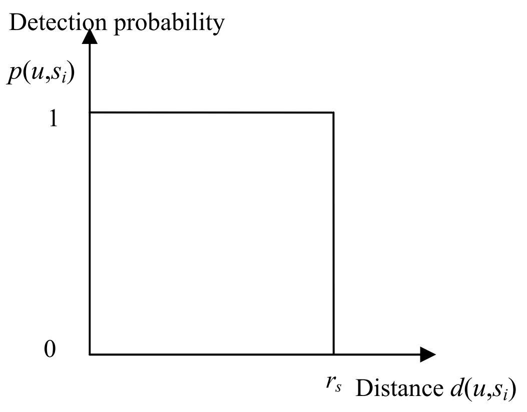
In this chapter, we introduce the background knowledge in Section 2.1 and describe literature survey in Section 2.2.

2.1 Background Knowledge

In this section, we introduce the background knowledge of object monitoring and tracking. The section is organized as follows. The detection and location models are described in Sections 2.1.1 and 2.1.2, respectively. The sensing energy consumption model is discussed in Section 2.1.3. Additionally, the gauss-markov motion model is presented in Section 2.1.4. The routing model is discussed in Section 2.1.5. The location awareness and energy awareness are described in Sections 2.1.6 and 2.1.7, respectively. Furthermore, the impacting factors are discussed in Section 2.1.8, and the quality of service is presented in Section 2.1.9.

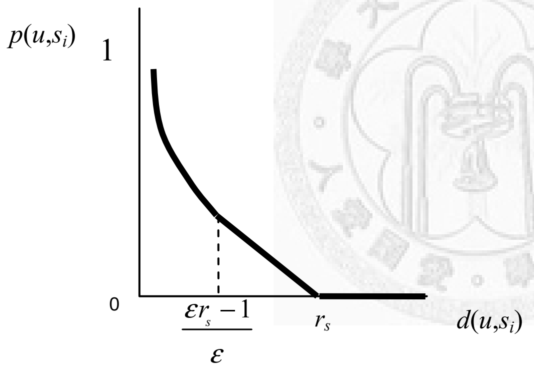
2.1.1 Detection Model

WSNs have three types of sensing models. 1) The binary sensing model [16]. A location can be either monitored or not monitored by a sensor, depending on whether the location is within the sensing range of sensor, illustrated in Figure 2.1. 2) The probabilistic sensing model [17]. A location will be monitored by a sensor according to some probability function. Figure 2.2 shows the probabilistic sensing model. 3) The hybrid sensing model [18][39]. For nominal sensing range R, the object is always detected when it is $R - e$ away or closer, never detected beyond $R + e$, and has a non-negative chance of detection between $R - e$ and $R + e$. Kirill et al. found that setting $e = 0.1R$ comes fairly close to the actual behavior of the sensors used in their experiments, illustrated in Figure 2.3.



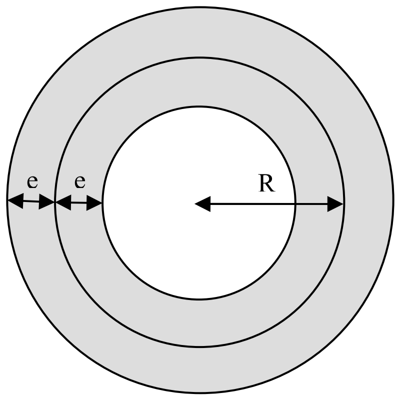
$$p(u,s_i)=\begin{cases}1, & \text{if } d(u,s_i)\le r_s\\ 0, & \text{otherwise}\end{cases}$$

Figure 2.1. Binary sensing model.



$$p(u,s_i)=\begin{cases}e^{-\varepsilon d(u,s_i)}, & \text{if } d(u,s_i)\le \dfrac{\varepsilon r_s - 1}{\varepsilon}\\ -\varepsilon e^{1-\varepsilon r_s}(d(u,s_i)-r_s), & \text{if } \dfrac{\varepsilon r_s - 1}{\varepsilon} < d(u,s_i)\le r_s\\ 0, & \text{otherwise}\end{cases}$$

Figure 2.2. Probabilistic sensing model.



$$p(u,s_i)=\begin{cases}1, & \text{if } d(u,s_i)\le R-e\\ e^{-\varepsilon d(u,s_i)}, & \text{if } R-e< d(u,s_i)\le R+e\\ 0, & \text{if } d(u,s_i)> R+e\end{cases}$$

Figure 2.3. Hybrid sensing model.

2.1.2 Location Model

An example of location model is shown in Figure 2.4, it has four types [15]. 1) The sensor cell, sensor ID, e.g., S_3. 2) The triangle, T_{34}, in S_3 and adjacent to S_4 represents the location of the mobile object. 3) The grid, G_{13}, indicates the ID of the grid where the object is detected. 4) The coordinates, e.g., (2.8, 2.2).

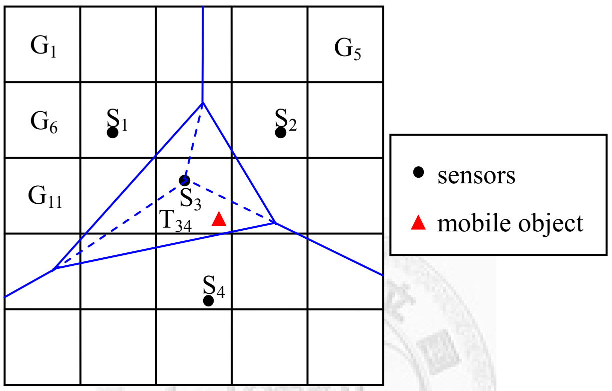


Figure 2.4. An example of location model.

2.1.3 Sensing Energy Consumption Model

There are two sensing energy consumption models in WSNs. We denote $e_s = f(r_s)$, where e_s is the energy consumption and r_s is the sensing radius of sensor s. The function f can be linear or quadratic. The first model is linear model which energy consumption is a linear function of the sensing radius. The second model is quadratic model which energy consumption is a quadratic function of the sensing radius [56][57][58].

2.1.4 Gauss-Markov Motion Model

A Gauss-Markov motion model uses one tuning parameter α to vary the degree of randomness in the mobility pattern [19].

$$s_n = \alpha s_{n-1} + (1-\alpha)\overline{s} + \sqrt{(1-\alpha^2)}s_{x_{n-1}}$$

$$d_n = \alpha d_{n-1} + (1-\alpha)\overline{d} + \sqrt{(1-\alpha^2)}d_{x_{n-1}}$$

where s_n and d_n are the new speed and direction of the intruder at time interval n. α is the tuning parameter used to vary the randomness, where $0 \le \alpha \le 1$. $\overline{s}$ and $\overline{d}$ are constants representing the mean value of speed and direction as $n \rightarrow \infty$. $s_{x_{n-1}}$ and $d_{x_{n-1}}$ are random variables from a Gaussian distribution. Totally random values (or Brownian motion) are obtained by setting $\alpha = 0$ and linear motion is obtained by setting $\alpha = 1$. Intermediate levels of randomness are obtained by varying the value of α between 0 and 1.

At each time interval the next location is calculated based on the current location, speed, and direction of movement. Specifically, at time interval n, the position of an object is given by the equations:

$$x_n = x_{n-1} + s_{n-1}\cos d_{n-1}$$
$$y_n = y_{n-1} + s_{n-1}\sin d_{n-1}$$

where (x_n, y_n) and (x_{n-1}, y_{n-1}) are the x and y coordinates of the positions of an object at the n^{th} and $(n - 1)^{th}$ time intervals, respectively, and s_{n-1} and d_{n-1} are the speeds and directions of the object, respectively, at the $(n - 1)^{th}$ time interval.

2.1.5 Routing Model

The routing model includes direct communication routing, multi-hop routing, and hierarchical routing architecture.

Sensor nodes can directly communicate with sink in direct communication routing model. Figure 2.5 shows an example of a direct communication routing model.

In multi-hop routing model, sensor nodes are hop by hop communication to sink. Figure 2.6 shows an example of a multi-hop routing model.

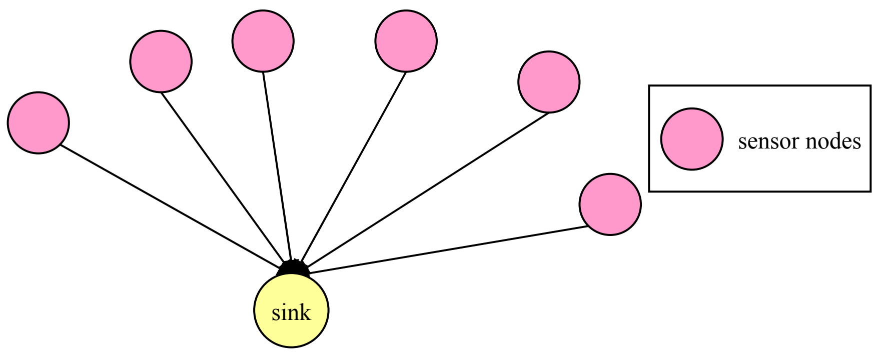


Figure 2.5. Direct communication routing model.

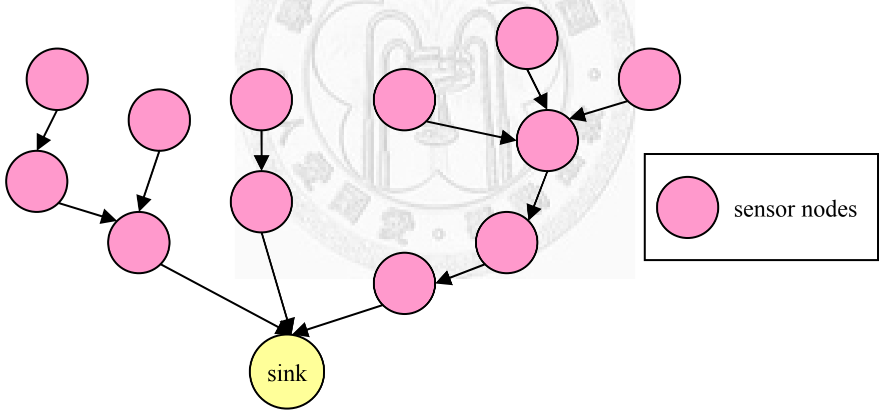


Figure 2.6. Multi-hop routing model.

In the proposed boundary monitoring, in-depth defense, and prediction object tracking models, we assume that the monitoring and tracking systems have a hierarchical routing architecture to forward sensing data to sink [66][67]. In a hierarchical routing architecture, nodes will play different roles in the WSNs. The cluster heads are closer to the sensor nodes than the sink. The cluster heads can do some aggregation and reduction of data in order to save energy. Figure 2.7 shows an example of a two-tiered hierarchical routing model. Besides, we have calculated energy consumption for sensor nodes to send sensing data to cluster heads in NLDA and POTA. However, the energy consumption is not calculated in BMAFS, BMAMS, and LDA.

In the two-tiered hierarchical routing model, the cluster heads are assumed to communicate with the sink directly [68]. The sensors use a binary sensing model. A location can be either monitored or not monitored by a sensor, depending on whether the location is within the sensing range of sensor, illustrated in Figure 2.1.

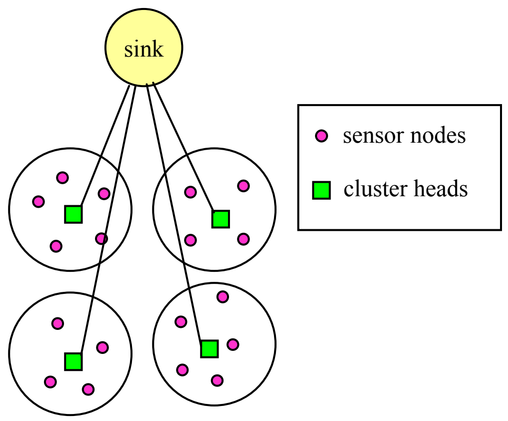


Figure 2.7. Two-tiered hierarchical routing model.

2.1.6 Location Awareness

The location awareness includes GPS (Global Positioning System) and anchor approach [69][70].

The GPS is a space-based positioning system by a group of satellites in earth orbit that transmit precise signals, allowing GPS receivers to calculate and display accurate location to sensors. However, GPS is an unattractive solution due to cost and power constraints.

In this anchor approach [70], a few of the sensor nodes called beacons know their coordinates in advance, either from satellite information (GPS) or pre-deployment.

The anchor approach scheme relies on signal strength information. The method is embedded in the inherent radio frequency communication capabilities of the nodes to approximate neighbor distances. Each node can hear three beacon neighbors and determines its own location by tri-angular algorithm and becomes a beacon. The tri-angular method is used iteratively to find all locations of each node.

2.1.7 Energy Awareness

The energy awareness can be classified according to different protocol layers.

1. Physical layer

In energy aware modulation scheme, A.Y. Wang et al. presented several energy minimization techniques derived from the unique properties of a practical short range asymmetric micro sensor system [72]. The techniques include energy efficient modulation schemes, appropriate multiple access protocols, and fast turn-on transmitter architecture.

In energy aware packet forwarding, V. Tsiatsis et al. proposed a node architecture that takes advantage of both the intelligence of the radio hardware and the needs of applications to efficiently handle the packet forwarding [73].

2. Data link layer

To design a good MAC protocol for the sensor networks, the energy awareness must be considered. The energy awareness protocols are used to prolong the system lifetime. In [74], I. Demirkol et al. proposed several MAC protocols for sensor networks to emphasize their strengths and weaknesses, such as S-MAC, T-MAC, DSMAC, WiseMAC, TRAMA, SIFT, and DMAC protocols.

3. Network layer

Energy awareness is an essential consideration in routing protocols. J.N.A.

Karaki et al. proposed the design tradeoffs between energy and communication overhead savings in every routing protocol [75]. For example, LEACH is a cluster-based routing protocol. LEACH intends to minimize network level energy consumption and improve the network utilization by balancing communication load over the whole network. The approach in this protocol is to cluster the whole network to avoid frequent expensive communications of single nodes.

2.1.8 Impact Factors

The object monitoring and tracking impact factors are described as follows [14][15]:

1. Number of moving objects

   The more moving objects inside the monitoring region is incurred the higher the total number of sampling and reporting.

2. Reporting frequency

   Keeping the reporting frequency low can reduce the number of transmissions. Hence, it can increase the lifetime of the object monitoring and tracking. They are two types of report, regular report and event-driven report.

3. Data precision [40]

   A higher data precision requires more data collection, more computation and more update packets, which results in more energy consumption on sensing, computing and communications

4. Sensor sampling frequency

   High sampling frequency incurs more energy consumptions.

5. Object moving speed

An object monitoring and tracking algorithm needs to sample more frequently on a high speed moving object.

6. Location models

Based on the location identification techniques used in the object monitoring and tracking services, location model can be categorized as geometric model (e.g., Coordinate) and symbolic model (e.g., Sensor ID).

2.1.9 Quality of Services

The quality of services is described as follows [20]:

1. Power consumption [43][48]

Sensor nodes are highly energy-constrained, because of the limitation of hardware and the infeasibility of recharging the battery under a harsh environment. Therefore, energy consumption of sensor nodes becomes one of the popular issues. Unused sensor nodes turn to sleeping mode in order to prolong the system lifetime. The types of power consumption of MICAz 2.4GHz are illustrated in Table 2.1 [21].

Table 2.1. The types of power consumption of MICAz 2.4GHz.

Types	CPU		RF Transceiver					
	Active	Sleep	Receive	TX -10dBm	TX -5dBm	TX dBm	Idle	Sleep
Power	8mA	<15μA	19.7mA	11mA	14mA	17.4mA	20μA	1μA

2. Accuracy [50]

In WSNs, the position error exists in the predicted position and the real position. To improve the accuracy of object position needs to merge more nodal data. This causes higher energy consumption.

3. Cost per detected position

It is the ratio of the energy consumption to the number of detected positions.

4. Lifetime [41][47]

It is the time when the first node of the network runs out of energy or the network can not provide services.

5. Monitoring resolution

The requirements of monitoring resolution are different for various applications. The granularity of data resolution is highly related to the sampling rate of sensors. The higher the data resolution is demanded, the more monitoring information is needed. Hence, the sampling rate needs to be set higher.

6. Scalability

Most applications consist of a great amount of sensors. The communication load and system load are scale to the size of the sensor network. It is an important factor to measure the performance of the applications. A principle of designing applications is to avoid waking up all sensors.

7. Response time

The main challenge in developing a real-time control system using sensor networks is the inconsistency in sensor measurements due to packet loss, communication delay, and false detections.

8. Fault tolerance

Moving sensors to cover uncovered regions while the nodes failed or node battery exhausted. The mechanism can prolong the system lifetime.

2.2 Literature Survey

2.2.1 Boundary Monitoring

Sink is to be noticed that some applications may need to record the information of objects entering or leaving the boundary of the monitoring region.

Under some applied circumstances, we just need to record the objects that enter or leave the boundary of monitored area [3][4][5][59]. For example, the preservation area administrators must be notified when the hunters enter or leave the wildlife preservation area in order to take necessary action. Besides, intrusion detection of enemies and in-depth defense are also required to record whether the objects enter or leave the boundary of monitored area for further notification and following track.

In the prior studies [22][23][24], In [22], Sam, et al. proposed a optimized communication and organization method called OCO to find the boundary nodes. The authors develop the border detection algorithm to identify a list of points that traverse the border of the geographic image, called border points. The border detection algorithm is shown in Figure 2.8.

```
Step1. For each pixel in an image, check if the color value = 1.
Step2. If true, scan all their neighbors to see if any of them having the color value = 0.
       If true, this pixel belongs to the border.
```

Figure 2.8. Algorithm for finding the border.

In [23], Sahoo, et al. proposed two boundary node selection algorithms, called SBNS and DBNS, to find out the boundary nodes. The two methods have three phases to find out the boundary nodes. The DBNS approach tries to find out the boundary nodes by distributed method. The algorithm of three phases is shown in Figure 2.9

The initial phase: Each sensor node in the monitoring region could be classified as boundary nodes or non-boundary nodes after the initial phase is executed.

The selection phase: The ring of boundary node can be found.

The pruning phase: The redundant boundary nodes are changed to non-boundary nodes.

Figure 2.9. The boundary node selection algorithm of three phases.

In [24], P.L. Chiu, et al. construct the sensor network such that it includes *k* mutually exclusive sets (number *k* is given). These sets are called covers. The covers are disjoint for each other. The method can find out the boundary nodes and prolong the system lifetime.

From papers review, we find that this study differs from prior works in several points. First, we consider both the energy conservation and lifetime extending during the sensor deployment phase of boundary monitoring. Second, we present a mathematical model to describe the optimization problem. Third, the relationship between the grouping capabilities of boundary node and the maximum extension of system lifetime is investigated. Fourth, we present a new concept of the check point. Fifth, we can find boundary nodes in user define disjoint monitoring region. A comparison among the OCO, SBNS, DBNS, BMAFS, and MBAMS are listed in Table 2.2.

Table 2.2. A comparison among the OCO, SBNS, DBNS, BMAFS, and MBAMS.

Factors of consideration / Algorithms	Boundary monitoring	User define disjoint monitoring region	Arrival and departure of objects	Maximizing the lifetime	Mobility capability	Researchers
Optimized communication and organization (OCO)	●	●				Tran, et al. [22]
Sequential boundary node selection (SBNS) and distributed boundary node selection (DBNS)	●		●			Sahoo, et al. [23]
Our work (BMAFS)	●	●	●	●		Lee, et al. [59]
Our work (BMAMS)	●	●	●	●	●	Lee, et al.

2.2.2 In-Depth Defense

An in-depth defense also called a defense in depth. Under some applied circumstances, we need to detect the objects that intrude the safeguard area [6][7][8][9][61]. For example, the commander must be notified when the enemies enter the safeguard area in order to take necessary action. Besides, intrusion detection of enemies is also required to record whether the objects enter monitored area for further notification and following track.

In WSNs security, the in-depth defense is used to describe a security system that is built using multiple rings or a group of neighboring nodes and policies to safeguard core area of the WSNs against multiple threats including enemy attacks and other security considerations.

In the prior studies [6][7][8][24][51], In [6], Yun, et al. analyze the intrusion detection problem in both homogeneous and heterogeneous WSNs. The work provides insights in designing homogeneous and heterogeneous WSNs and helps in selecting critical network parameters so as to meet the application requirements. The study can be summarized as follows:

1. The authors propose an analytical model for intrusion detection in wireless sensor networks, and mathematically analyzing the detection probability with respect to various network parameters such as node density and sensing range.
2. Using the analytical model to single-sensing detection and multiple-sensing detection scenarios for homogeneous and heterogeneous wireless sensor networks.
3. The authors discuss the network connectivity and broadcast reachability.

In [7][8], Li, et al. proposed a distributed group-based intrusion detection scheme that meets all the above requirements by partitioning the sensor networks into many groups. The group-based intrusion detection scheme involves two phases: grouping the sensor networks and running the group-based intrusion detection algorithm in each group. The group-based intrusion detection scheme can save power consumption.

In [24], Chiu, et al. construct the sensor network such that it includes *k* mutually exclusive sets (number *k* is given). These sets are called covers. The covers are disjoint for each other. The method can find out the group of nodes for in-depth defense and prolong the system lifetime.

In [51], Li, et al. focus on the survivability of wireless sensor network and develop a model to evaluate the tradeoffs between the cost of defense mechanisms for Wireless Sensor Network and the resulting expected survivability after a network attack.

From papers review, we find that this study differs from prior works in four points. First, we consider both the energy conservation and lifetime extending during the sensor deployment phase of in-depth defense. Second, we present a mathematical model to describe the optimization problem. Third, the relationship between the grouping capabilities of in-depth defense and the maximum extension of system lifetime is investigated. Fourth, we present a new concept of the check point. A comparison among the IDHH, GIDA, LDA, and NLDA are listed in Table 2.3.

Table 2.3. A comparison among the IDHH, GIDA, LDA, and NLDA.

Factors of consideration / Algorithms	In-depth defense	Quality of intrusion detection	Maximizing the lifetime	Behaviors of intruders	Researchers
Intrusion detection in homogeneous and heterogeneous wireless sensor networks (IDHH)	●	●			Yun, et al. [6]
Group-based intrusion detection algorithm (GIDA)	●		●		Li, et al. [7][8]
Our work (LDA)	●	●	●		Lee, et al.
Our work (NLDA)	●	●	●	●	Lee, et al. [61]

2.2.3 Object Tracking

Object tracking is the key application issue of WSNs which is widely deployed for military and wildlife animal tracking. Object tracking wireless sensor networks have two critical operations [13][14][15]. One is monitoring. Sensor nodes are required to detect and track the moving states of mobile object. The other is reporting. The nodes sensing the object need to report their discoveries to the sink. These two operations are interleaved during the entire object tracking process.

Object tracking algorithm has two cases. One is tree-based. For example, optimized communication and organization (OCO) algorithm [22], scalable tracking using networked sensors (STUN) [10], and deviation-avoidance tree (DAT) [11][12].

Another is prediction-based. For example, dual prediction-based reporting (DPR) [15], and distributed prediction tracking (DPT) [52].

2.2.3.1 Tree-based Object tracking

Our focus, in the prior studies [10][11][12], has been on developing strategies for reducing the energy consumption in reporting operations. In [10], H.T. Kung, et al. proposed a scalable tracking method using network sensors called STUN for sensor tracking system. The tracking system is a scalable tracking architecture that employs hierarchical structure to allow the system to handle a large number of tracked objects. Furthermore, authors proposed a drain-and-balance (DAB) method to construct a hierarchical structure of STUN based on expected properties of the object movement patterns such as the frequency of object movements over a monitoring region.

For example, consider those detection messages from sensors that detect the arrival of the object. Message of sensor 1 will update the detected sets of all its ancestors. The messages from sensors 2 and 4 do not update the detected sets of their parents and thus will be pruned there. The message from sensor 3 updates only its parent *z* and thus will be pruned at *x*. An example of a message-pruning hierarchy is shown in Figure 2.10.

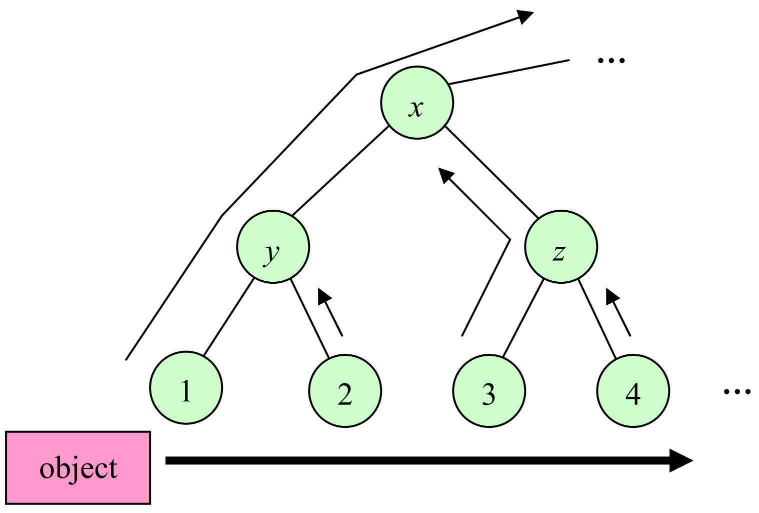


Figure 2.10. An example of a message-pruning hierarchy

In [11][12], C.Y. Lin, et al. proposed two message-pruning tree structures called DAT and Z-DAT for object tracking. The two methods are used to construct an object tracking tree for reducing the communication cost of location update. The Z-DAT approach tries to divide the sensing area into square-like zones and recursively combine these zones into a tree.

This study is an extension of the work in [10][11][12]. The prior studies are expanded to the energy-efficient object tracking in wireless sensor networks. We focus on the problem of constructing an energy-efficient wireless sensor networks for object tracking services using the object tracking tree. This tree is to propose a data aggregation model for object tracking [25][26][27][28][29][30][60]. Therefore, we motivate to propose a heuristic strategy to cope with the problem. With a given sensor network arbitrary topology, we particularly consider the bi-directed moving objects with given frequencies for each pair of sensor nodes and link transmission cost. The total communication cost can be computed and minimized by object tracking tree.

The object tracking tree is a weighted spanning graph of given sensor and communication nodes [49]. The tree is used to minimize total communication cost. Therefore, constructing the object tracking tree is an NP-complete problem [31]. A method called Lagrangean relaxation which has been successfully adopted to solve many famous NP-complete problems [32][33][34].

From papers review [10][11][12], this study differs from the prior works in two points. First, we consider the bi-directed moving objects with given frequencies for each pair of sensor nodes and link transmission cost. Second, we present a LR mathematical model to describe the optimization problem and propose LR-based heuristic algorithm to solve the problem. A comparison among the STUN, DAB, DAT, ZDAT, and TOTA are listed in Table 2.4.

Table 2.4. A comparison among the STUN, DAB, DAT, ZDAT, and TOTA.

Factors of consideration / Algorithms	Object tracking	Update cost	Query cost	Bi-directed moving objects and link transmission cost	Entering and leaving of object in monitoring region	Researchers
Tracking using networked sensors (STUN) and drain-and-balance (DAB)	●	●				Kung, et al. [10]
Deviation-avoidance tree (DAT) and zone-based deviation-avoidance tree (Z-DAT)	●	●	●			Lin, et al. [11][12]
Our work (TOTA)	●	●	●	●	●	Lee, et al. [60]

2.2.3.1 Prediction-based Object tracking

Prediction can minimize the number of nodes participating in the tracking [13][14][15]. The wake-up mechanisms and recovery mechanisms of different prediction models will affect the system performance. Prediction model works well if one can tolerate “small amount of errors” in predictions and “latency” in generating prediction models.

Our focus, in the prior studies [13][14][15], has been on developing strategies for reducing the energy consumption in object tracking operations. In [13], Y. Xu, et al. proposed the localized prediction paradigm for power-efficient object tracking sensor network. Localized prediction consists of a localize network architecture and a prediction mechanism called dual prediction, which can achieve power savings by allowing most of the sensor nodes to stay in sleep mode and by reducing the amount of long-range transmissions. The basic method for dual prediction is to have sensor nodes and their cluster heads both calculate the next states of tracked objects. The sensor nodes do not send an update of object movement to its cluster head unless it is different from the prediction. In addition, no prediction values need to be sent from cluster heads to sensor nodes. However, the saving of long distance transmissions between a sensor node and its cluster head comes with a small price, i.e., transfer of moving history from a current node to the destination node. As we will show later in the performance evaluation, this cost is well justified because it consumes less power for transmission to a neighbor sensor node and it occurs only when the tracked object moves into a new detection area.

In [14], Y. Xu, et al. proposed a prediction-based energy saving scheme, called PES, to reduce the energy consumption for object tracking under acceptable conditions. PES tries to approach to the ideal scheme by minimizing both of the sampling frequency and the number of nodes involved in object tracking, while balances off the overhead caused by missing the objects. PES consists of three parts: 1) a prediction model which anticipates the future movement of an object so only the sensor nodes expected to discover the object will be activated; 2) a wake up mechanism that, based on some heuristics taking both energy and performance into accounts, sets up which nodes and when they should be activated; 3) a recovery mechanism initiated only when the network loses the track of an object.

In [15], Y. Xu, et al. proposed the dual prediction reporting (DPR) mechanism, in which the sensor nodes make intelligent decisions about whether or not to send updates of objects movement states to the base station and thus save energy. DPR consists of two major components, i.e., location model and prediction model. The choice of a location model determines the granularity of the movement states of mobile objects. A prediction model decides how to estimate the future movement of objects from their movement history.

From papers review [13][14][15], this study differs from the prior works in two points. First, we consider entering and leaving of object in boundary of monitoring region. Second, we develop one prediction-based algorithm based on modified Viterbi algorithm to solve object tracking problem. A comparison among the PES, DPR, and POTA are listed in Table 2.5. The dynamic prediction algorithm, POTA, maintains *n*-1, *n*-2, and *n*-*h* speed and direction of the object at time interval *n*. The mechanism can improve accuracy of object predicted position.

Table 2.5. A comparison among the PES, DPR, and POTA.

Factors of consideration / Algorithms	Object tracking	Entering and leaving of object in monitoring region	Dynamic prediction	Researchers
Prediction-based energy saving (PES)	●			Xu, et al. [14]
Dual prediction-based reporting (DPR)	●			Xu, et al. [14]
Our work (POTA)	●	●	●	Lee, et al.

2.3 Lagrangean Relaxation Method

Many approaches had been proposed in 1970s [32][33][34], most of them used the divide-and-conquer technique to decompose a complicated problem into several plain sub-problems and solve them one by one. Lagrangean relaxation method is one of the popular approaches used for solving some mathematical problems, like integer programming problems [34]. Since it is flexible and provides excellent solutions for these problems, it has become one of the best tools for solving optimization problems, such as integer programming, linear programming combinatorial optimization, and non-linear programming problems. We briefly describe the Lagrangean relaxation method as follow.

First, we remove some complex constraints of the primal mathematical model to the objective function with corresponding multiplier, and then the original problem will be transformed into a new Lagrangean relaxation problem. Second, by relaxing the complicated constraints, we can divide the primal problem into several simple and easily solvable sub-problems. For each sub-problem, we can optimally solve it by some well-known algorithms.

By solving the Lagrangean relaxation problems, we can get a boundary value to the objective function of the original primal problem. The solution of the Lagrangean relaxation problem is always the lower bound of the original minimization problem. Then we use the decision variables and multipliers got from the Lagrangean relaxation problem to design a heuristic approach to get a primal feasible solution. Furthermore, in order to improve the solution quality by minimizing the gap between the primal problem and Lagrangean relaxation problem, we use the subgradient method to adjust the multipliers per iteration.

The major concept of Lagrangean relaxation method is shown in Figure 2.11, and the Lagrangean relaxation method procedure is shown in Figure 2.12.

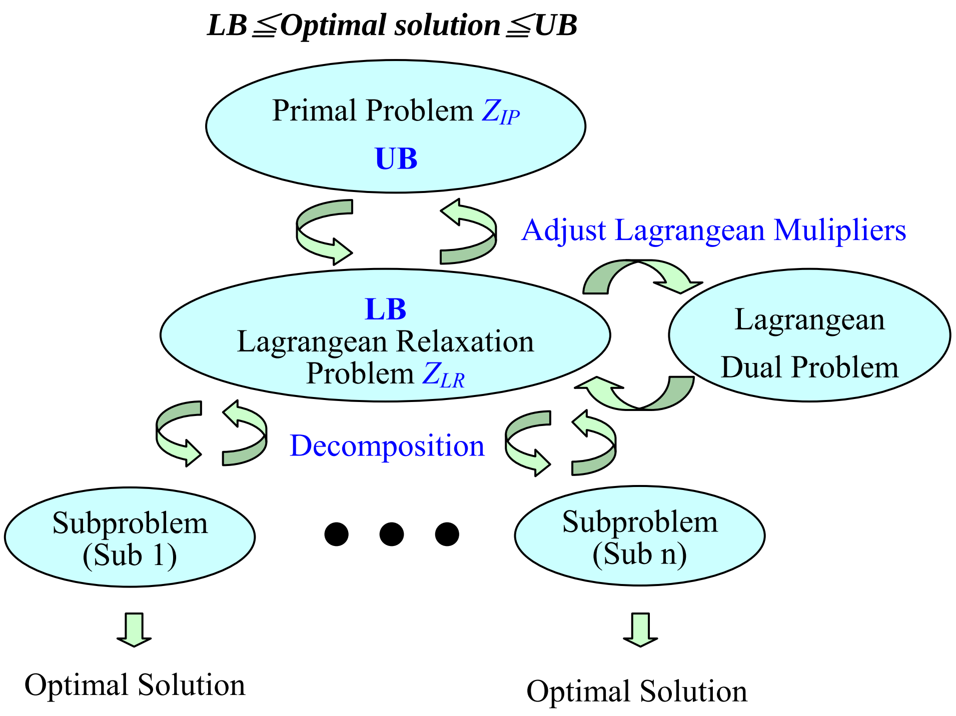


Figure 2.11. The major concept of Lagrangean relaxation method.

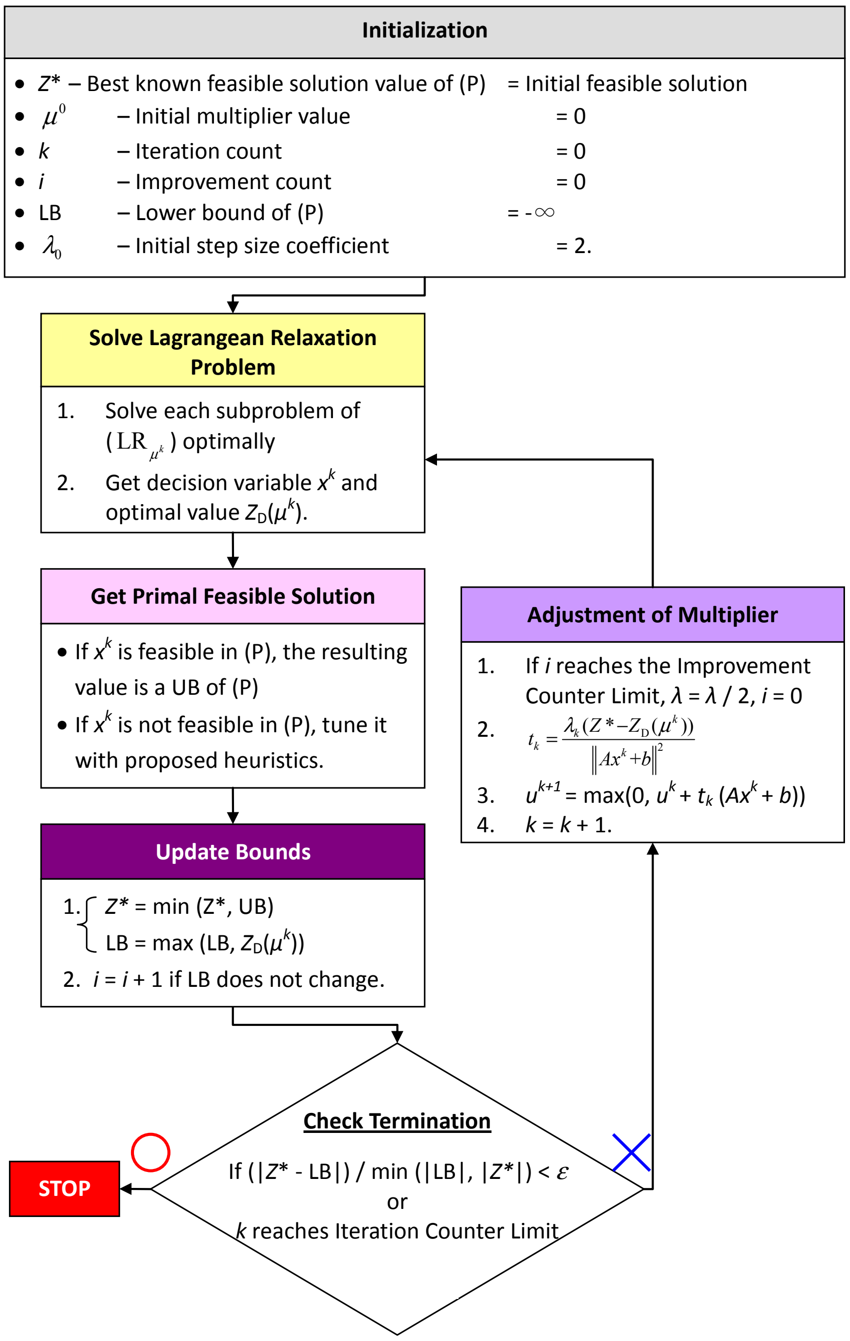

Initialization
Z^* – Best known feasible solution value of (P) = Initial feasible solution
μ^0 – Initial multiplier value = 0
k – Iteration count = 0
i – Improvement count = 0
LB – Lower bound of (P) = $-\infty$
λ_0 – Initial step size coefficient = 2.
Solve Lagrangean Relaxation Problem
1. Solve each subproblem of (LR_{μ^k}) optimally
2. Get decision variable x^k and optimal value $Z_D(\mu^k)$.
Get Primal Feasible Solution
If x^k is feasible in (P), the resulting value is a UB of (P)
If x^k is not feasible in (P), tune it with proposed heuristics.
Update Bounds
1. Z^* = min (Z^*, UB)
LB = max (LB, $Z_D(\mu^k)$)
2. $i = i + 1$ if LB does not change.
Adjustment of Multiplier
1. If i reaches the Improvement Counter Limit, $\lambda = \lambda / 2$, $i = 0$
2. $t_k = \frac{\lambda_k (Z^* - Z_D(\mu^k))}{\|Ax^k + b\|^2}$
3. $u^{k+1} = \max(0, u^k + t_k(Ax^k + b))$
4. $k = k + 1$.
Check Termination
If $(|Z^* - \mathrm{LB}|) / \min(|\mathrm{LB}|, |Z^*|) < \varepsilon$
or
k reaches Iteration Counter Limit
STOP

Figure 2.12. The procedure of Lagrangean relaxation method.

In reference [34], R.K. Ahuja et al. provide a guide to use Lagrangean relaxation and describe several applications in which Lagrangean relaxation method has been used to solve many well-known hard problems. We only list partial problems in Table 2.6.

Table 2.6. The applications of Lagrangean relaxation method.

Problems	Embedded network structure
Network with side constraints problem	● Minimum cost flows ● Shortest paths
Traveling salesman problem	● Assignment ● Minimum cost flows
Network design problem	● Shortest paths

2.4 Simulation Techniques

Operation research can be classified into two models: 1) deterministic model and 2) probabilistic model. The methods of deterministic model do not contain the element of probability. For example, linear programming, non-linear programming, and dynamic programming, etc. The methods of probabilistic model contain the element of probability. For example, Markovian decision processes, queueing theory, forecasting, reliability, and simulation techniques, etc. [53][54][71]

Operation researchers typically use simulation technique when the involved stochastic system is too complex to be analyzed satisfactorily by variety of analytical models [54].

In general, a simulation model is used in order to study real-life systems which do not currently exist. In particular, one is interested in quantifying the performance of a system under study for various values of its input parameters. Such quantified measures of performance can be very useful in the process of managerial decision. The basic steps of simulation are shown in Figure 2.13. [53]

First is to define the problem that we want to resolve. Second is to formulate model of simulation. Third is to write the simulator. Forth is to validate the model. Fifth is to run the simulator. Finally is to analyze the results.

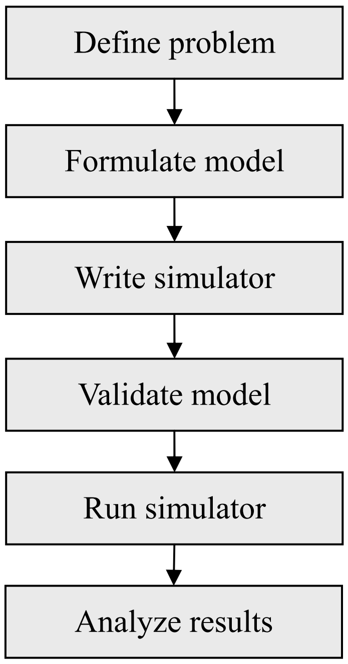


Figure 2.13. The procedure of simulation technique.

Chapter 3 Boundary Monitoring Algorithms

In this chapter, we propose two algorithms, BMAFS and BMAMS, to support boundary monitoring services. The BMAFS algorithm is to construct boundary monitoring for grouping capabilities. It tries to find the maximum *k* groups of sensors for boundary monitoring of the sensor field to prolong the system lifetime. The BMAMS algorithm is to address the problem of boundary nodes relocation. It can move previously deployed sensors to cover uncovered check points while the nodes failed or node battery exhausted. The mechanism can further prolong the system lifetime.

In this chapter, the boundary nodes grouping algorithms are described in Section 3.1 and boundary nodes mobility algorithm is presented in Section 3.2.

3.1 Boundary Monitoring Algorithms for Fixed Sensors

In this section, we develop three algorithms to construct efficient boundary monitoring for wireless sensor networks of grouping capabilities. We try to find the maximum *k* groups of sensors for boundary monitoring of the sensor field. The mechanism can prolong the system lifetime. This problem is formulated as a 0/1 integer-programming problem. Three algorithms are proposed for solving the optimization problem. The experiment results show that the proposed boundary monitoring algorithm (BMAFS) gets a near optimization in the efficient boundary monitoring for grouping capabilities.

The rest of this section is organized as follows. The overview is described in Section 3.1.1. The problem and mathematical models are described in Sections 3.1.2 and 3.1.3, respectively. In addition, the solution procedure is presented in Section 3.1.4. Furthermore, the computational results are discussed in Section 3.1.5, and conclusions are presented in Section 3.1.6.

3.1.1 Overview

In this section, we focus on the sensor grouping problem to support boundary monitoring services. First, we try to find the boundary nodes from the monitoring region. Second, we will deal with the problem of arrival and departure for the objects. Third, we want to find the maximum k groups of sensors to monitor a sensor field boundary. This mechanism can prolong the system lifetime of boundary monitoring.

We introduce the concept of check points. The check points are virtual points, which can check full coverage. Besides, it can save energy consumption because the concept can check full coverage more efficiently for arbitrary topology and disjoint monitoring regions. And further, we find the maximum k sets of sensors to support boundary monitoring services on the monitoring region. These sets can be joint or disjoint. Each of them, is called a group, can provide full coverage of the boundary of the sensor field. Each group is activated in turn to monitor the boundary of the monitoring region. Generally, the power consumption for inactive sensors can be neglected, and the system lifetime can be effectively prolonged up to k times. We present a mathematical model to describe the optimization problem and three heuristic-based algorithms are proposed to solve the problem.

We formulate the problem as a 0/1 integer programming problem where the objective function is the maximization of the system lifetime of the boundary of the monitoring region subject to the constraints full coverage, battery capacity, and integer variables. We construct three heuristic-based algorithms to solve the problem.

The problem is formulated as a linear optimization-based problem with three different decision variables: wakeup sensors, covered check points, and full coverage in the round r. Wakeup sensors are 1 if sensor s is awake in the round r, and 0 otherwise. Covered check points are 1 if check point a is covered by at least one awake sensor in the round r, and 0 otherwise. Full coverage is 1 if full coverage boundary check points in the round r, and 0 otherwise. In the further experiments, the proposed boundary monitoring for grouping capabilities algorithm is expected to be efficient and effective in dealing with the optimization problem.

3.1.2 Problem Description

3.1.2.1 Boundary Nodes Selection

In this section, we use the mathematical method to select boundary node. We particularly introduce novel concept of check points for full coverage check points. The monitoring region can be represented as a collection of 2D region. It includes check points and sensor nodes, as illustrated in Figure 3.1. The positioning resolution of application determines the granularity of check points and sensing range. We assume that sensors are randomly deployed in boundary of monitoring region.

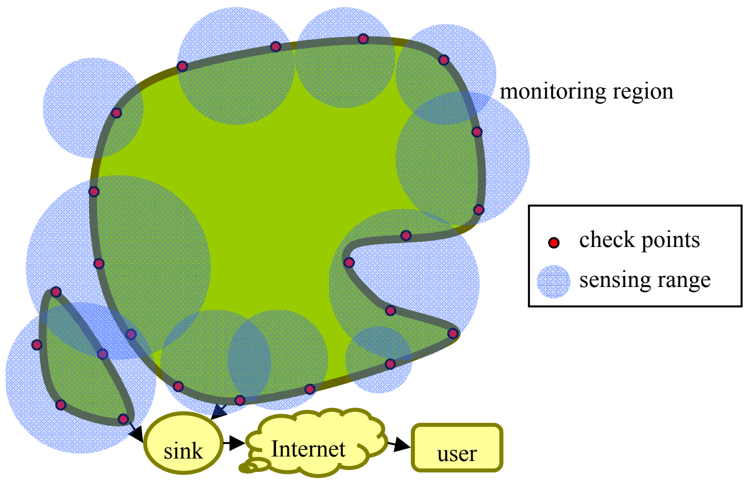


Figure 3.1. Check point-based boundary nodes selection.

The check points are virtual points. We assume that the distance of each neighboring check points is small or equal to the minimum size of monitoring object. The boundary of monitoring region is fully covered if all check points are covered by awaked sensors. It is a typical full coverage if check points are deployed in high density and check points are fully covered.

The proposed boundary nodes selection algorithm (BNSA) is shown in Figure 3.2.

We aimed at each sensor checking of the whole check points. If there is any radius of sensor covering the check point, then we should put the sensor into the set of boundary nodes.

Algorithm Boundary Nodes Selection

Input: Coordinates of check points and sensor nodes, and sensing radius of sensor nodes

Output: Boundary nodes (*BNSet*)

```
 1: begin
 2:       BNSet=$\varnothing$;                /* BNSet: the set of boundary nodes */
 3:       UncoverSet=$\varnothing$;           /* UncoverSet: the set of uncovered check points */
 4:       for a=1 to cp do          /* cp: number of check points */
 5:             $flag_a$=0;
 6:       for a=1 to cp do
 7:       begin
 8:             for s=1 to sn do    /* sn: number of sensor node*/
 9:             begin
10:             if check point a is covered by sensor node s
                                    /* $\sqrt{(x_s - x_a)^2 + (y_s - y_a)^2} \le r_s$ */
11:                   then BNSet←sensor node s and $flag_a$=1
12:             end
13:             if $flag_a$=0
14:                   then UncoverSet ←check point a
15:       end
16:       if Uncoverset$\neq\varnothing$
17:             then boundary of monitoring region is not fully covered
18:             else boundary of monitoring region is fully covered
                      and boundary nodes=BNSet
19: end
```

Figure 3.2. The boundary nodes selection algorithm.

In this algorithm, steps 2-5 set initialize values. Steps 6-15 are used to find boundary node set. Steps 16-18 check full coverage.

The computational complexity of the boundary nodes selection algorithm at steps 4-5 is $O(|A|)$, where $|A|$ is number of check points. From steps 6-15 is $O(|S||A|)$, where $|S|$ is number of sensor nodes. Therefore, the computational complexity is $O(|S||A|)$. Hence, the computational complexity of the boundary nodes selection algorithm should be $O(|S||A|)$.

We use above BNSA to find out boundary nodes and check full coverage of boundary.

3.1.2.2 Arrival and Departure of Objects

We assume that $r_c \geqq 2\max r_s + w$ and $w > 2\max r_s$, where r_c is communication radius, r_s is sensing radius, and w is minimum size of monitoring object, as shown in Figure 3.3. The assumption is in order to avoid unidentifiable arrival or departure of objects.

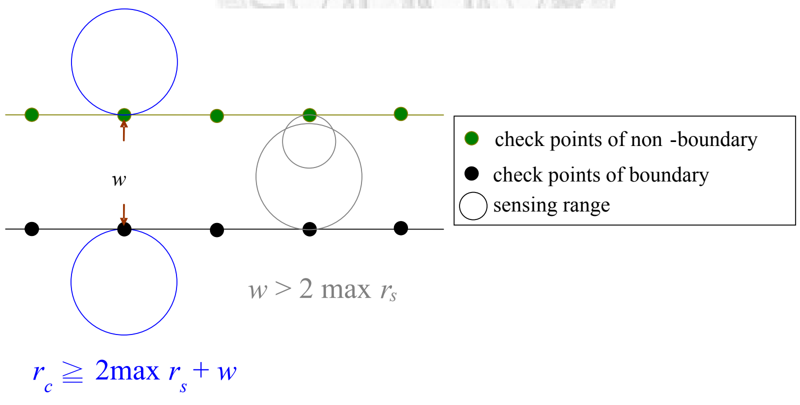


Figure 3.3. The communication and sensing radii for arrival and departure of objects.

We propose two algorithms, single ring algorithm (SRA) and double ring algorithm (DRA), to deal with the problem of arrival and departure of objects. In the single ring algorithm, an object is sensed by boundary nodes (*BNs*) while it touches the monitoring region, and *BNs* will wake up their neighboring non-boundary nodes (*non-BNs*). For the next moment, if *BNs* do not sense the object but neighboring *non-BNs* sense the object, the object is entering the monitoring region.

Similarly, the neighboring *non-BNs* of *BNs* detect the object. For the next moment, if *BNs* sense the object and soon after they do not sense the object, and neighboring *non-BNs* do not sense the object, the object is leaving the monitoring region, as shown in Figure 3.5.

The proposed single ring algorithm is shown in Figure 3.4.

Algorithm Single Ring

Input: Coordinate of boundary nodes and non-boundary sensor nodes, and sensing radius of sensor nodes

Output: Arrival or departure of objects

```
 1: begin
 2:       boundary nodes always wake up
 3:       for (;;)                    /* infinite loop */
 4:       begin
 5:             if BNs can sense the object
 6:                   then wake up its neighboring non-boundary nodes
 7:             if BNs do not sense the object and
                   neighboring non-BNs can sense the object for the next moment
 8:                   then the object is entering the monitoring region
 9:             else if BNs sense the object and soon after do not sense the object
                and neighboring non-BNs do not sense the object for the next moment
10:                         then the object is leaving the monitoring region
11:       end
12: end
```

Figure 3.4. The single ring algorithm.

In single ring algorithm, from steps 5-6 wake up its neighboring non-boundary nodes when boundary node senses the object. Steps 7-8 the object is entering the monitoring region when boundary node do not sense the object and neighboring non-boundary node sense the object for the next moment. Steps 9-10 the object is leaving the monitoring region when boundary node sense the object and soon after do not sense the object, and neighboring non-boundary node do not sense the object for the next moment.

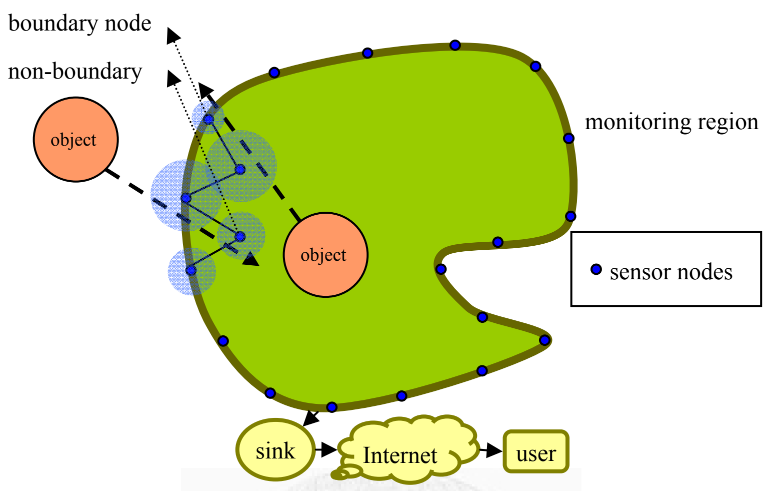


Figure 3.5. The single ring for arrival and departure objects.

In the double ring algorithm, an object is sensed by *BNs* of outer ring while the object touches the monitoring region. For the next moment, if outer ring *BNs* do not sense the object and inner ring *non-BNs* sense the object, the object is entering the monitoring region.

Similarly, the inner ring *non-BNs* detect the object. For the next moment, if outer ring *BNs* sense the object and presently do not sense the object, and inner ring *non-BNs* do not sense the object, then the object is leaving the monitoring region, as shown in Figure 3.7.

The proposed double ring algorithm is shown in Figure 3.6.

Algorithm Double Ring

Input: Coordinate of inner ring and outer ring sensor nodes, and sensing radius of sensor nodes

Output: Arrival or departure of object

```
1:  begin
2:          boundary nodes always wake up
```

```
3:      for (;;)                      /* infinite loop */
4:      begin
5:          if outer ring BNs sense the object, and for the next moment, outer ring
             BNs do not sense the object and inner ring non-BNs sense the object
6:              then the object is entering the monitoring region
7:          if inner ring BNs sense the object, and for the next moment,
             inner ring BNs do not sense the object and outer ring non-BNs
              sense the object and soon after do not sense the object
8:              then the object is leaving the monitoring region
9:      end
10: end
```

Figure 3.6. The double ring algorithm.

In double ring algorithm, from steps 5-6 the object is entering the monitoring region when outer ring boundary node sense the object, for the next moment, outer ring boundary node do not sense the object and inner ring non-boundary node sense the object. Steps 7-8 the object is leaving the monitoring region when inner ring boundary node sense the object, for the next moment, inner ring boundary node do not sense the object and outer ring non-boundary node sense the object and presently do not sense the object.

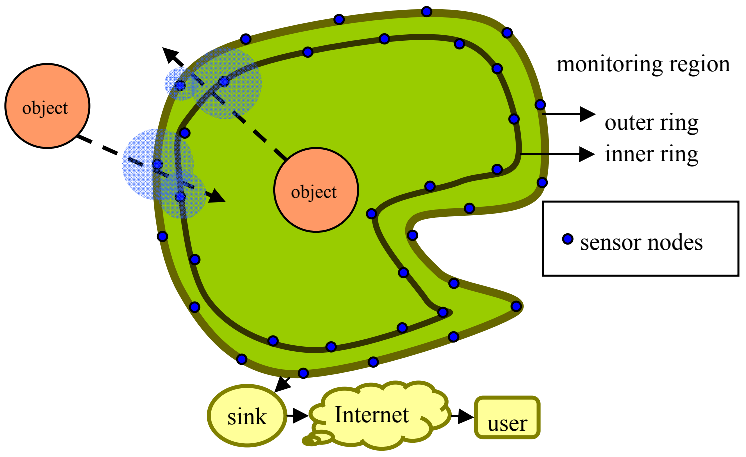


Figure 3.7. The double ring for arrival and departure objects.

3.1.2.3 Boundary Monitoring Algorithms for Grouping Capabilities

We try to find maximum *k* sets of sensors to support boundary monitoring services on the monitoring region, as shown in Figure 3.8. Each of them, is called a group, can provide full coverage of the field. Each group is activated in turn to monitor the boundary. Figure 3.9 shows the state transitions of the sensor network. From the network viewpoint, two operation states exist: the sleeping state and the active states. Only one group sensors are activated in turn to monitor the boundary, and the other group sensors are sleeping at one time. The system lifetime can be effectively prolonged up to *k* times. The detailed descriptions are shown in Table 3.1.

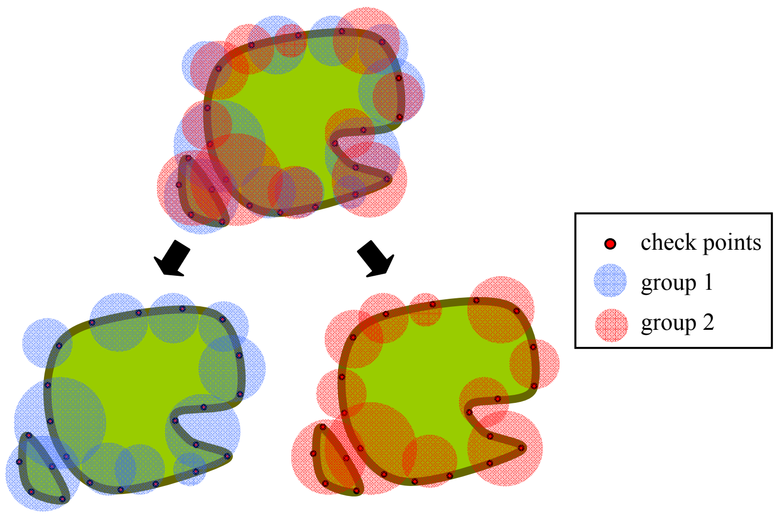


Figure 3.8. Boundary monitoring for grouping capabilities.

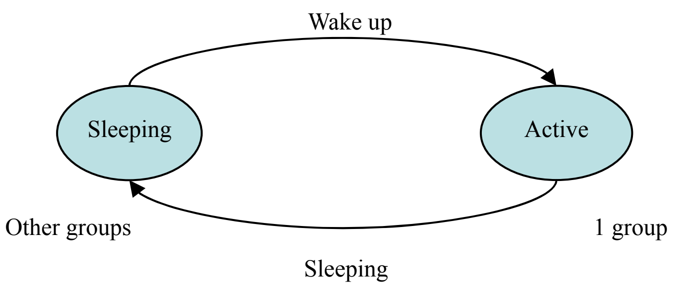


Figure 3.9. The state diagram of the sensor network.

Table 3.1. Problem description in boundary monitoring problem for fixed sensors.

Given	1. The set of check points. 2. The set of sensor nodes. 3. Initial energy level of each sensor node. 4. Energy consumption for sensor nodes to sense data in each round. 5. Detection radius of each sensor.
Objective	To maximize the boundary monitoring service lifetime.
Subject to	1. Full coverage of boundary check points in each round 2. Battery capacity.
To determine	To determine whether sensor *s* is awake or not in the round *r*.

3.1.3 Mathematical Model

In this section, we formulate the problem as a 0/1 integer programming problem where the objective function is the maximization of the amount of cover *k* required to full coverage under a given boundary of sensor networks. The problem is a variant of the set *k*-cover problem and thus is NP-complete [35].

The notations used to model the problem are listed in Table 3.2 and Table 3.3.

Table 3.2. Notations of the given parameters in boundary monitoring for fixed sensors problem.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
A	Index set of the service check points in the monitoring region boundary.
C_s	The initial energy level of each sensor node *s*.
E_s	The energy consumption for aware sensor node *s* to sense data in each round.
R	The upper bound number of rounds.
b_{sa}	The indicator function which is 1 if the check point *a* is in the sensing range of the sensor node *s*, and 0 otherwise.

Table 3.3. Notations of the decision variables in boundary monitoring for fixed sensors problem.

Decision Variables	
Notation	Description
π_{sr}	1 if sensor *s* is awake in the round *r*, and 0 otherwise.
y_{ar}	1 if check point *a* at least is covered by one awake sensor in the round *r*, and 0 otherwise.
z_r	1 if full coverage boundary check points in the round *r*, and 0 otherwise.

Problem (IP):

$$\max \sum_{r \in R} z_r \tag{IP}$$

subject to:

The full coverage boundary check points constraints

$$y_{ar} \le \sum_{s \in S} b_{sa} \pi_{sr} \qquad \forall a \in A, r \in R \tag{1}$$

$$z_r \le \frac{\sum_{a \in A} y_{ar}}{|A|} \qquad \forall r \in R \tag{2}$$

The battery capacity constraint

$$\sum_{r \in R} \pi_{sr} E_s \le C_s \qquad \forall s \in S \tag{3}$$

The integer constraints

$$\pi_{sr} = 0 \text{ or } 1 \qquad \forall s \in S, \ r \in R \tag{4}$$

$$y_{ar} = 0 \text{ or } 1 \qquad \forall a \in A, r \in R \tag{5}$$

$$z_r = 0 \text{ or } 1 \qquad \forall r \in R. \tag{6}$$

The objective function is to maximize the system lifetime of the monitoring region boundary. The lifetime is defined as the total number of rounds.

Constraints (1)-(2): Full coverage boundary check points constraints.

Constraint (3): For each sensor node *s*, the total sensing consumption can not exceed its initial energy level.

Constraints (4)-(6): The integer constraints for decision variables π_{sr}, y_{ar}, and z_r.

3.1.4 Solution Procedure

The parameters and decision variables used to model boundary monitoring algorithms in this section are listed in Table 3.4.

Table 3.4. The parameters and decision variables in algorithms of boundary monitoring problem.

Notation	Description
max_k	The upper bound of system lifetime.
cp	The number of check points.
sn	The number of sensor nodes.
cpc_no[*a*]	The number of covered rounds in each check point *a*.
cap	The initial energy level.
cs[*s*]	The energy level of sensor node *s*.
es[*s*]	The energy consumption for aware sensor node *s* to sense data in each round.
max_round	The system lifetime.
c_bsa[*a*]	The number of covered times in check point *a* by waked sensors.
count[*s*]	The number of covered check points by awaked sensor *s*.
c_s[*s*]	The number of covered check points under sensing range of sensor *s*.
t_cover	The number of full coverage in each iteration.
bsa[*s*][*a*]	The indicator function which is 1 if the check point *a* is in the sensing range of the sensor node *s* and 0 otherwise.
full_coverage[*r*]	The decision variable which is equal to *cp* if full coverage boundary check points in the round *r*, and less than *cp* otherwise.
p[*s*][*r*]	The decision variable which is 1 if sensor *s* is awake in the round *r*, and 0 otherwise.
cover[*a*][*r*]	The decision variable which is 1 if check point *a* is covered by at least one awake sensor in the round *r*, and 0 otherwise.

3.1.4.1 Upper Bound of the Maximum Rounds

In this section, we study the upper bound of maximum rounds in boundary monitoring.

We can calculate the upper bound (UB) of system lifetime by follow algorithm in Figure 3.10.

We use initial energy level of sensor node *s* divided by energy consumption for aware sensor node *s* to sense data in each round. The rounds can get in each sensor nodes. The upper bound of system lifetime is that we search for the minimum round for all sensor nodes.

Algorithm Upper Bound of the Maximum Rounds

Input: The initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round

Output: The upper bound of system lifetime (*max_k*)

```
 1:  begin
 2:      max_k=∞;
 3:      for a=1 to cp do
 4:          cpc_no[a]=0;
 5:      for s=1 to sn do
 6:          for a=1 to cp do
 7:              if (bsa[s][a]=1)
 8:                  then cpc_no[a]=cpc_no[a]+(cs[s]/es[s])
 9:      for a=1 to cp do
10:          if (cpc_no[a]<max_k)
11:              then max_k=cpc_no[a]
12:  end
```

Figure 3.10. The upper bound algorithm of system lifetime.

In this upper bound of the maximum rounds algorithm, steps 2-4 are setting initialize value, steps 5-8 are finding the maximum rounds value for each check point. Steps 9-11 are used to get system upper bound of the maximum rounds.

The computational complexity of the upper bound algorithm of system lifetime at steps 3-4 is $O(|A|)$, where $|A|$ is number of check points. At steps 5-8 is $O(|S||A|)$, where $|S|$ is number of sensor nodes. From steps 9-11 is $O(|A|)$. Therefore, the computational complexity is $O(|S||A|)$. Hence, the computational complexity of the upper bound algorithm of system lifetime should be $O(|S||A|)$.

3.1.4.2 Simple Algorithm 1

We compare the proposed iteration-based algorithm (boundary monitoring algorithm for fixed sensors) with non-iteration-based algorithms (simple algorithms 1 and 2) that use the concept of "cover" to determine whether sensor *s* is awake or not in the round *r*. The "cover" is 1 if the check point *a* is in the sensing range of the sensor node *s* and 0 otherwise.

In each round, we first find sensor *s* to cover check point *a*, and then sensor *s* is awake in the round *r*, and repeat the assignment process until all check points have been covered.

A simple algorithm 1 (SA1) is listed in Figure 3.11.

Algorithm Simple 1

Input: The initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round, and the upper bound of system lifetime *max_k*

Output: The maximum rounds (*max_round*)

```
 1: begin
 2:         max_round=0;
 3:         for r=1 to max_k do
 4:         begin
 5:                 full_coverage[r]=0;
 6:                 for s=1 to sn do
 7:                         p[s][r]=0;
 8:         end
 9:         for r=1 to max_k do
10:         begin
11:                 for s=1 to sn do
12:                         for a=1 to cp do
13:                                 if ((bsa[s][a]=1) and (cs[s]>=es[s]) and (cover[a][r]=0)) then
14:                                 begin
```

```
15:                     p[s][r]=1;
16:                     cs[s]=cs[s]-es[s];
17:                     for a=1 to cp do
18:                         if ((bsa[s][a]=1) and (cover[a][r]=0))
19:                             then cover[a][r]=1 and
                                    full_coverage[r]=full_coverage[r]+1;
20:                 end
21:     end
22:     for r=1 to max_k do
23:         if (full_coverage[r]=cp)
24:             then boundary of monitoring region in round r is fully covered
                    and max_round=max_round+1;;
25: end
```

Figure 3.11. The simple algorithm 1 in boundary monitoring problem.

In the simple algorithm 1, steps 2-8 are setting initialize value, steps 9-21 determine whether sensor *s* is awake or not in the round *r*. Steps 22-24 are used to get system maximum rounds.

The computational complexity of the simple algorithm 1 in boundary monitoring problem at steps 3-8 is $O(|R||S|)$, where $|R|$ is total number of rounds and $|S|$ is number of sensor nodes. At steps 9-21 is $O(|R||S||A|^2)$, where $|A|$ is number of check points. From steps 22-24 is $O(|R|)$. Therefore, the computational complexity is $O(|R||S||A|^2)$. Hence, the computational complexity of the simple algorithm 1 in boundary monitoring problem should be $O(|R||S||A|^2)$.

3.1.4.3 Simple Algorithm 2

Simple algorithm 1 wastes on energy consumption, because system has redundant awaked sensor nodes. Therefore, we propose simple algorithm 2 (SA2) to deal with the problem. For example, s_2 is redundant sensor node as shown in Figure 3.12.

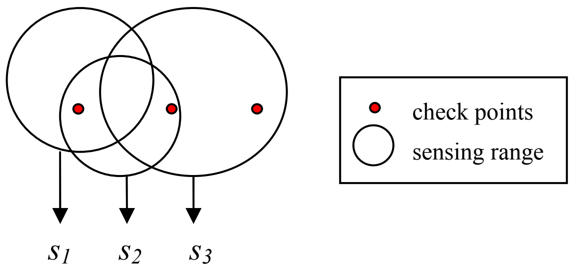


Figure 3.12. An example of deleting redundant awaked sensor node.

A simple algorithm 2 is listed in Figure 3.13.

Algorithm Simple 2

Input: The initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round, and the upper bound of system lifetime *max_k*

Output: The maximum rounds (*max_round*)

```
 1: begin
 2:      max_round=0;
 3:      for r=1 to max_k do
 4:      begin
 5:          full_coverage[r]=0;
 6:          for s=1 to sn do
 7:              p[s][r]=0;
 8:      end
 9:      for r=1 to max_k do
10:      begin
11:          for s=1 to sn do
12:              for a=1 to cp do
13:                  if ((bsa[s][a]=1) and (cs[s]>=es[s]) and (cover[a][r]=0)) then
14:                  begin
15:                      p[s][r]=1;
16:                      cs[s]=cs[s]-es[s];
17:                      for k=1 to cp do
18:                          if (bsa[s][a]=1)
19:                              then c_bsa[a]=c_bsa[a]+1;
20:                          if ((bsa[s][a]=1) and ( cover[a][r]=0))
21:                              then cover[a][r]=1 and
                                     full_coverage[r]=full_coverage[r]+1;
```

```
22:                 end
23:         if (full_coverage[r]=cp) then          /* delete redundant nodes */
24:             for s=1 to sn do
25:             begin
26:                 for a=1 to cp do
27:                 begin
28:                     if ((p[s][r]=1) and (bsa[s][a]=1) and (c_bsa[a]>=2))
29:                         then count[s]=count[s]+1;
30:                     if (count[s]=c_s[s])
31:                     begin
32:                         cs[s]=cs[s]+es[s];  /* energy recovery */
33:                         p[s][r]=0;
34:                         for a=1 to cp do
35:                             if (bsa[s][a]==1)
36:                                 then c_bsa[a]=c_bsa[a]-1;
37:                     end
38:                 end
39:             end
40:     end
41:     for r=1 to max_k do
42:         if (full_coverage[r]=cp)
43:             then boundary of monitoring region in round r is fully covered
                  and max_round=max_round+1;;
44: end
```

Figure 3.13. The simple algorithm 2 in boundary monitoring problem.

In the simple algorithm 2, steps 2-8 are setting initialize value, steps 9-22 determine whether sensor *s* is awake or not in the round *r*. Steps 23-40 are used to delete redundant awaked sensor nodes in the round *r*. Steps 41-43 are used to get system maximum rounds.

The computational complexity of the simple algorithm 2 in boundary monitoring problem at steps 3-8 is $O(|R||S|)$, where $|R|$ is total number of rounds and $|S|$ is number of sensor nodes. At steps 9-40 is $O(|R||S||A|^2)$, where $|A|$ is number of check points. From steps 41-43 is $O(|R|)$. Therefore, the computational complexity is $O(|R||S||A|^2)$. Hence, the computational complexity of the simple algorithm 2 in boundary monitoring problem should be $O(|R||S||A|^2)$.

3.1.4.4 Boundary monitoring algorithm for fixed sensors

In this section, we present a heuristic-based boundary monitoring algorithm for fixed sensors (BMAFS) to improve SA1 and SA2 algorithms.

To solve the original problem near-optimally, we use the *full_coverage*[*r*] to check full coverage in the round *r*. The decision variable is equal to *cp* if full coverage boundary check points are in the round *r*, and 0 otherwise. Then, in each round, we use different sensor node *id* to cover uncheck point *a* given minimum be cover check points and then sensor *s* is awake in the round *r*, and repeat the assignment process until all check points have been covered. For example, system prioritizes to select s_1 sensor node, because s_1 sensor node has not covered selected check points. If system can not find the s_1 sensor node, then second priority is s_2 sensor node, as shown in Figure 3.14.

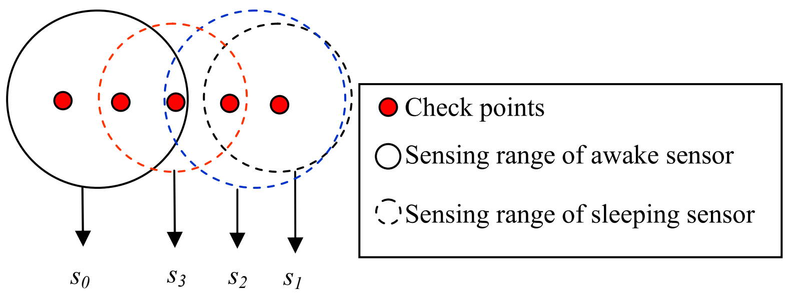

Check points
Sensing range of awake sensor
Sensing range of sleeping sensor
s_0
s_3
s_2
s_1

Figure 3.14. An example of greedy-based sensor node selection.

The procedure of boundary monitoring algorithm for fixed sensors is shown in Figure 3.15. First of all is to initialize. Second is to determine whether sensor *s* is awake or not in the round *r*. Third is to delete redundant awaked sensor nodes. Forth is to get system maximum rounds. Finally is to check whether it is a stop condition or not. If the answer is negative, go back to the first step.

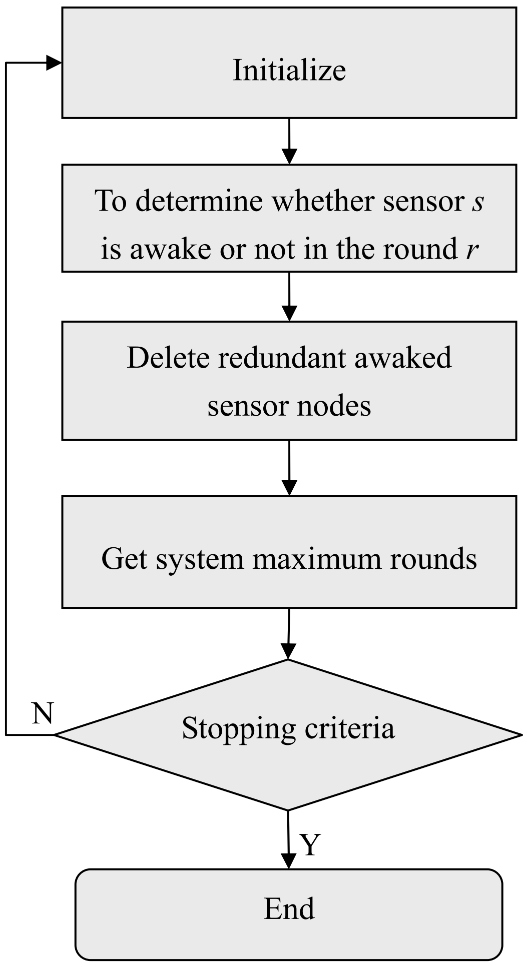

Initialize
To determine whether sensor s is awake or not in the round r
Delete redundant awaked sensor nodes
Get system maximum rounds
Stopping criteria
N
Y
End

Figure 3.15. The procedure of boundary monitoring algorithm for fixed sensors.

A boundary monitoring algorithm for fixed sensors is listed in Figure 3.16.

Algorithm Boundary Monitoring for Fixed Sensors

Input: The initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round, and *max_k*

Output: The maximum rounds (*max_round*)

```
 1: begin
 2:     for iteration=1 to sn do
 3:     begin
 4:         for r=1 to max_k do
 5:         begin
 6:             full_coverage[r]=0;
 7:             for s=1 to sn do
 8:                 p[s][r]=0 and cs[s]=cap;
 9:         end
10:         for r=1 to max_k do
11:         begin
```

```
12:          s=iteration;
13:          for i=1 to sn do
14:              for a=1 to cp do
15:                  if ((bsa[s][a]=1) and (cs[s]>=es[s]) and (cover[a][r]=0)) then
16:                      begin
17:                          p[s][r]=1;
18:                          cs[s]=cs[s]-es[s];
19:                          for k=1 to cp do
20 :                           if (bsa[s][a]=1)
21 :                               c_bsa[a]=c_bsa[a]+1 ;
22:                               if ((bsa[s][a]=1) and (cover[a][r]=0))
23:                                   then cover[a][r]=1 and
                                          full_coverage[r]=full_coverage[r]+1;
24:                          s=(s+1)%sn;
25:                      end
26:          if (full_coverage[r]=cp) then        /* delete redundant nodes */
27:              for s=1 to sn do
28:              begin
29 :                 for a=1 to cp do
30:                      if ((p[s][r]=1) and (bsa[s][a]=1) and (c_bsa[a]>=2))
31 :                         then count[s]=count[s]+1;
32:                      if (count[s]=c_s[s]) then
33:                      begin
34 :                         cs[s]=cs[s]+es[s] ;   /* energy recovery */
35 :                          p[s][r]=0 ;
36 :                          for a=1 to cp do
37 :                              if (bsa[s][a]=1)
38 :                                  c_bsa[a]=c_bsa[a]-1 ;
39 :                     end
40 :             end
41:      end
42:      t_cover=0;
43:      for r=1 to max_k do
44:          if (full_coverage[r]=cp)
45:              then boundary of monitoring region in round r
                     is fully covered and t_cover=t_cover+1;
46:      if (round< t_cover)
47:          then round= t_cover;
48:      if (max_round<round)
49:          then max_round=round;
50:      round=-∞;
```

```
51:         if ($max\_round=max\_k$)
52:             then break;
53:     end
54: end
```

Figure 3.16. The boundary monitoring algorithm for fixed sensors.

In the boundary monitoring algorithm, steps 2, 12, and 24 are iteratively to improve system maximum rounds. From steps 4-9 are to set initial values, steps 10-11, and 13-23 are to determine whether sensor *s* is awake or not in the round *r*. Steps 26-41 are used to delete redundant awaked sensor nodes. Steps 42-50 are used to get system maximum rounds.

The computational complexity of the boundary monitoring algorithm for fixed sensors at steps 4-9 is $O(|R|)$, where $|R|$ is total number of rounds. At steps 10-41 is $O(|R||S||A|^2)$, where $|S|$ is number of sensor nodes and $|A|$ is number of check points. From steps 43-45 is $O(|R|)$. Above steps from steps 2-53 run $|S|$ times. Therefore, the computational complexity is $O(|R||S|^2|A|^2)$. Hence, the computational complexity of the boundary monitoring algorithm for fixed sensors should be $O(|R||S|^2|A|^2)$. This makes the algorithm scalable to a large scale WSNs.

After solving the problem, a set of feasible solutions of the problem (IP) can be obtained. The feasible solution is a lower bound (*LB*) of the problem (IP), and the *max_k* is the upper bound (*UB*) of the problem (IP). We get the *UB* and the *LB*, respectively. The gap between *UB* and *LB*, computed by $|(UB-LB)/LB|*100\%$, illustrates the optimality of problem solution. The smaller the gap computed, the better the optimality.

3.1.4.5 Varieties of the model

We can extend the model to two different scenarios to fulfill more applications.

Scenario 1:

In some scenario, the lower energy of sensor results the decrease of sensing range. In such scenario, we can periodically run the BMAFS to ensure full coverage of the check points.

Scenario 2:

In some application, the coverage rate of check points does not need to be 100%. We only modify our model that adds the given parameter *v*. Table 3.5 shows the description of *v*.

Table 3.5. Notation descriptions for new given parameter *v*.

Given Parameter	
Notation	Description
v	The coverage rate.

The coverage rate constraints can be modified to our mathematical model as followings:

The coverage rate constraints

$$y_{ar} \le \sum_{s\in S} b_{sa}\pi_{sr} \qquad \forall a\in A, r\in R \qquad (1)$$

$$z_r \le \frac{\sum_{a\in A} y_{ar}/|A|}{v} \qquad \forall r\in R \qquad (2)$$

The scenarios described above are only different from the original model on simple mathematical calculation. Hence, we only consider the original problem in experiments, and the others can be easily inferred.

3.1.5 Computational Results

To evaluate the performance of the proposed algorithms, we conduct an experiment. The performance is assessed in terms of total number of rounds.

3.1.5.1 Scenario

The proposed algorithm is coded in C under a Dev C++ 4.9.9.2 development environment. All the experiments are performed on a Core 2 Duo 2.2GHz CPU running Microsoft Windows Vista. The algorithm is tested on a 2D sensor field. We distribute 100, 400, and 1600 sensor nodes and 36, 72, and 156 check points respectively in 2D sensor field. The radii of different sensors types s_a, s_b, s_c, and s_d are 1, 2, 3, and 4, respectively. The energy consumption of aware different sensor types s_a, s_b, s_c, and s_d are 1, 2, 3, and 4, respectively, with linear model $e_s = r_s$; 1, 4, 9, and 16, respectively, with quadratic model $e_s = r_s^2$; and 2, 8, 18, and 32, respectively, with quadratic model $e_s = 2r_s^2$ in each round. The initial energy level of each sensor node is 32.

3.1.5.2 Experiment results

Figure 3.17 shows an example of boundary monitoring.

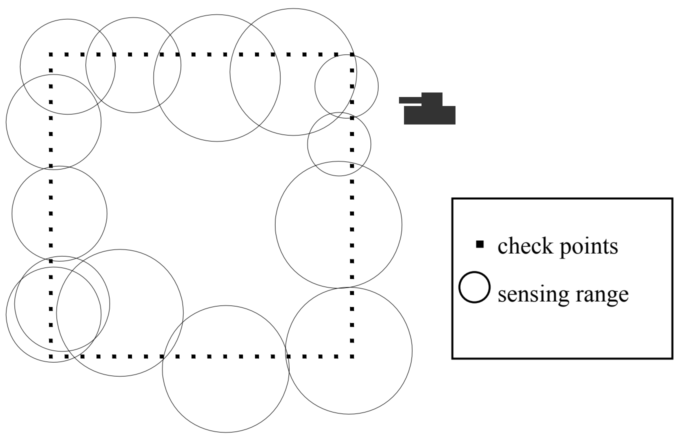


Figure 3.17. An example of boundary monitoring.

Table 3.6, Table 3.7, and Table 3.8 show the maximum total number of rounds calculated by different algorithms. We can see that the BMAFS outperforms the SA1 and SA2.

Table 3.6. Evaluation of the gap and improvement ratio with different number of nodes with the linear model $e_s = r_s$.

Number of Nodes (check points, sensor nodes)	Monitoring Region (m^2)	BMAFS	UB	Gap	SA1	Improvement Ratio to SA1	SA2	Improvement Ratio to SA2
(36, 100)	10×10	94	94	0	52	0.81	78	0.21
(76, 400)	20×20	86	86	0	58	0.48	76	0.13
(156, 1600)	40×40	60	60	0	50	0.20	52	0.15
(316, 6400)	80×80	50	50	0	34	0.47	42	0.19

Table 3.7. Evaluation of the gap and improvement ratio with different number of nodes with the quadratic model $e_s = r_s^2$.

Number of Nodes (check points, sensor nodes)	Monitoring Region (m^2)	BMAFS	UB	Gap	SA1	Improvement Ratio to SA1	SA2	Improvement Ratio to SA2
(36, 100)	10×10	29	29	0	18	0.61	21	0.38
(76, 400)	20×20	23	23	0	15	0.53	20	0.15
(156, 1600)	40×40	20	20	0	16	0.25	18	0.11
(316, 6400)	80×80	13	13	0	9	0.44	11	0.18

Table 3.8. Evaluation of the gap and improvement ratio with different number of nodes with the quadratic model $e_s = 2r_s^2$.

Number of Nodes (check points, sensor nodes)	Monitoring Region (m^2)	BMAFS	UB	Gap	SA1	Improvement Ratio to SA1	SA2	Improvement Ratio to SA2
(36, 100)	10×10	11	11	0	7	0.58	8	0.38
(76, 400)	20×20	10	10	0	7	0.43	9	0.11
(156, 1600)	40×40	9	9	0	8	0.13	8	0.13
(316, 6400)	80×80	6	6	0	4	0.50	5	0.20

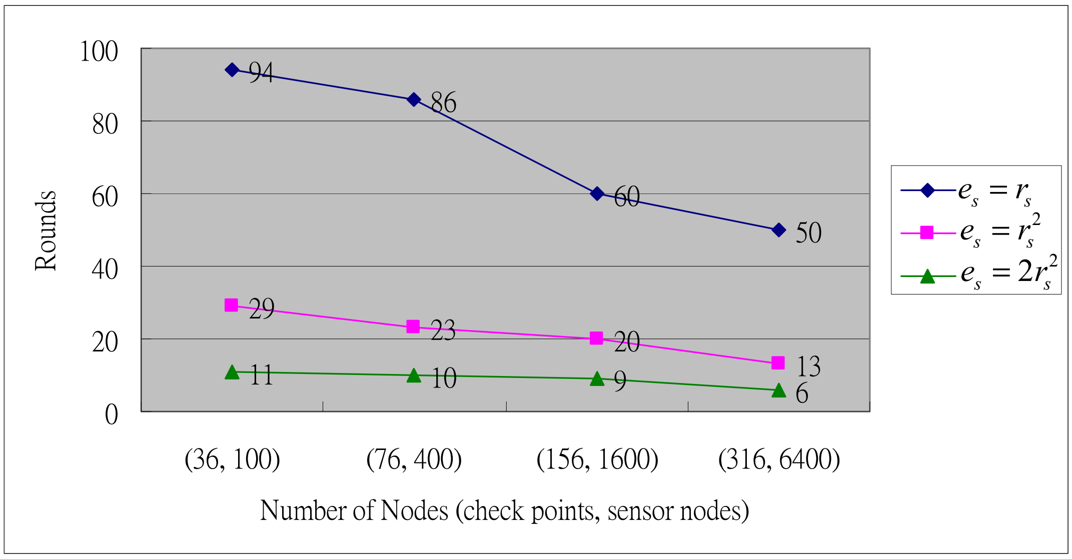


Figure 3.18. A comparison among the linear model $e_s = r_s$, quadratic model $e_s = r_s^2$, and quadratic model $e_s = 2r_s^2$.

3.1.5.3 Discussion

The experiment results show that the algorithm is not only better than the other heuristic algorithms, such as SA1 and SA2 algorithms, but the gap is also small. Compared with SA1 and SA2 algorithms, the proposed BMAFS algorithm can improve the percentage of energy consumption from 11% to 81%. In the test problems, BMAFS also achieves optimality since the gaps are 0%, as shown in Table 3.6, Table 3.7, and Table 3.8. Therefore, the results show that the proposed algorithm can achieve boundary monitoring for grouping capabilities. Furthermore, the algorithm is very efficient and scalable in terms of the running time. Besides, Total rounds of quadratic model, $e_s = r_s^2$, are exponential decrease than total rounds of linear model, $e_s = r_s$, as shown in Table 3.6, Table 3.7, and Figure 3.18. Total rounds of quadratic model, $e_s = 2r_s^2$, are approximately double decrease than total rounds of quadratic model, $e_s = r_s^2$, as shown in Table 3.7, Table 3.8, and Figure 3.18.

3.1.6 Concluding Remarks

This study proposes a boundary monitoring algorithm in wireless sensor networks. To our best knowledge, the proposed algorithm is truly novel and it has not been yet discussed in previous researches. This study first formulates the problem as a 0/1 integer programming problem, and then proposes a heuristic-based algorithm for solving the optimization problem. The proposed approach can prolong system lifetime for wireless sensor networks of grouping capabilities.

As to the next section, we describe to further investigate mobile capabilities model based on boundary monitoring application requirements.

3.2 Boundary Monitoring Algorithm for Mobile Sensors

In this section, we address the problem of boundary node relocation, i.e., moving previously deployed sensors to cover uncovered check points due to failure of other nodes or battery exhaustion of other nodes.

The rest of this section is organized as follows. The overview is described in Section 3.2.1. The problem and mathematical model are described in Sections 3.2.2 and 3.2.3, respectively. Additionally, the solution procedure is presented in Section 3.2.4. Furthermore, the computational results are discussed in Section 3.2.5, and conclusions are presented in Section 3.2.6.

3.2.1 Overview

We propose a BMAMS for relocating mobile sensors in a timely and efficient. In our framework, sensor relocation consists of two phases. First, we propose a solution to find the uncovered check points. Second, we propose a relocation solution to quickly locate the sensors with low message overhead. This problem is formulated as 0/1 integer-programming problem. The BMAMS is proposed for solving the optimization problem. Experiment results show that the proposed heuristic algorithm is very effective in reducing the relocation time and the energy consumption.

3.2.2 Problem Description

We also address the problem of boundary nodes relocation [36][37][46]. We can move previously deployed sensors to cover uncovered check points, when failure of other nodes or battery exhaustion of other nodes. The mechanism also can prolong the system lifetime. Figure 3.19 shows an example of boundary monitoring for mobile sensors. We assume that sensors are randomly deployed in boundary of monitoring region. Besides, we assume that global position system (GPS) is installed in each sensor node and sensor node is implemented by ground robot [78].

The purpose of this section is to study an energy-efficient sensors mobility algorithm for full coverage boundary in wireless sensor networks (WSNs). Such sensor network has to be designed to achieve full coverage boundary for given arbitrary topology of sensor network. We propose algorithms for sensors mobility for full coverage boundary. The experiment results show that the proposed algorithm can prolong the system lifetime than BMAFS.

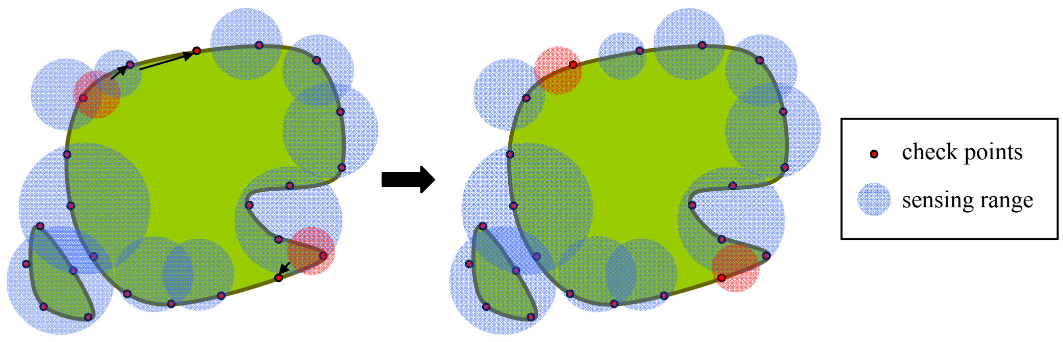


Figure 3.19. An example of boundary monitoring for mobile sensors.

We use boundary node selection algorithm to check full coverage boundary. And if the monitoring region boundary is not full coverage, we can move sensor nodes to achieve full coverage of the monitoring region boundary. The detailed descriptions are shown in Table 3.9.

Table 3.9. Problem description in boundary monitoring problem for mobile sensors.

Given	1. The set of check points. 2. The set of sensor nodes. 3. Residual energy level of each sensor node. 4. Energy consumption for sensor node to move one unit. 5. Energy consumption for sensor nodes to sense data in each round. 6. Detection radius of sensor.
Objective	To maximize the boundary monitoring services lifetime.

Subject to	1. Full coverage boundary check points in each round. 2. Battery capacity.
To determine	1. whether sensor *s* is awake or not in the round *r*. 2. whether sensor node *s* moves to cover check point *a* or not.

3.2.3 Mathematical Model

Table 3.10. Notations of the given parameters in boundary monitoring for mobile sensors problem.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
A	Index set of the service check points in the monitoring region boundary.
d_{sa}	Euclidean distance for sensor node *s* moves to cover uncovered service check point *a*, $s \in S$, $a \in A$.
$e(d_{sa})$	Energy consumption for sensor node *s* moves to cover uncovered service check point *a*.
E_s	The energy level of each sensor node *s*, $s \in S$.
E_m	The energy consumption for sensors node to sense data in each round.

Table 3.11. Notations of the indicator parameters in boundary monitoring for mobile sensors problem.

Indicator Parameters	
Notation	Description
ρ_{sa}	The indicator function which is 1 if the check point *a* is in the coverage of the non-moved sensor node *s* and 0 otherwise.
σ_{sa}	The indicator function which is 1 if the check point *a* is in the coverage of the moved sensor node *s* and 0 otherwise.

Table 3.12. Notations of the decision variables in boundary monitoring for mobile sensors problem.

Decision Variables	
Notation	Description
ι_s	1 if sensor node *s* does not move, and 0 otherwise. $s \in S$.
ξ_{sa}	1 if sensor node *s* moving to cover uncovered check point *a*, and 0 otherwise. $a \in A$.
π_{sr}	1 if sensor *s* is awake in the round *r*; otherwise is equal to 0.
y_{ar}	1 if check point *a* at least is covered by one awake sensor in the round *r*, and 0 otherwise.
z_r	1 if full coverage check points in the round *r*, and 0 otherwise.

Problem (IP):

$$\max \sum_{\forall r \in R} z_r \qquad \text{(IP)}$$

subject to:

The full coverage check points constraint

$$y_{ar} \le \sum_{s \in S}(\iota_s \pi_{sr} \rho_{sa} + \sum_{b \in A} \xi_{sa} \pi_{sr} \sigma_{sa}) \qquad \forall a \in A, r \in R, \qquad (1)$$

$$z_r \le \frac{\sum_{a \in A} y_{ar}}{|A|} \qquad \forall r \in R \qquad (2)$$

$$\sum_{b \in A} \xi_{sb} = 1 \qquad \forall s \in S \qquad (3)$$

The battery capacity constraints

$$\sum_{r \in R} \pi_{sr} E_m + \sum_{\forall a \in A} \xi_{sa} e(d_{sa}) \le E_s \qquad \forall s \in S \qquad (4)$$

The integer constraints

$$\iota_s = 0 \text{ or } 1 \qquad \forall s \in S \qquad (5)$$

$$\xi_{sa} = 0 \text{ or } 1 \qquad \forall a \in A,\ s \in S \qquad (6)$$

$$\pi_{sr} = 0 \text{ or } 1 \qquad \forall s \in S,\ r \in R \qquad (7)$$

$$y_{ar} = 0 \text{ or } 1 \qquad \forall a \in A,\ r \in R \qquad (8)$$

$$z_r = 0 \text{ or } 1 \qquad \forall r \in R. \qquad (9)$$

The objective function is to maximize the system lifetime of the sensor network given sensor network.

Constraints (1)-(2): Full coverage boundary check points constraint in each round *r*.
Constraint (3): Sensor node *s* only moving to one check point *a*.
Constraint (4): For each sensor node *s*, the moving power consumption and total sense data consumption can not exceed its energy level.
Constraints (5)-(9): The integer constraints for decision variables ι_s, ξ_{sa}, π_{sr}, y_{ar}, and z_r.

3.2.4 Solution Procedure

In this section, we propose a boundary monitoring algorithm for mobile sensors to solve the problem. The parameters and decision variables used to model our algorithms in this section are listed in Table 3.13.

Table 3.13. The parameters and decision variables in algorithms of boundary monitoring for mobile sensors problem.

Notation	Description
max_k	The upper bound of system lifetime.
cp	The number of check points.
sn	The number of sensor nodes.
BNSet	The boundary nodes set.
UncoverSet	The uncovered check points set.
$c[j]$	The number of cover for check point *j*.
$s[i]$	The sensor node *i*.
$a[j]$	The check point *j*.
$x[i][j]$	The decision variable which is 1 if a_j is covered by sensor s_i, and 0 otherwise.

The procedure of boundary monitoring algorithm for mobile sensors is shown in Figure 3.20. First is to initialize. Second is to find the uncovered check points in the round *r*. Third is to move sensor node to cover uncovered check points in the round *r*. Forth is to delete redundant awaked sensor nodes in the round *r*. Finally is to check whether it is stopping criteria or not. If the answer is negative, go back to the first step.

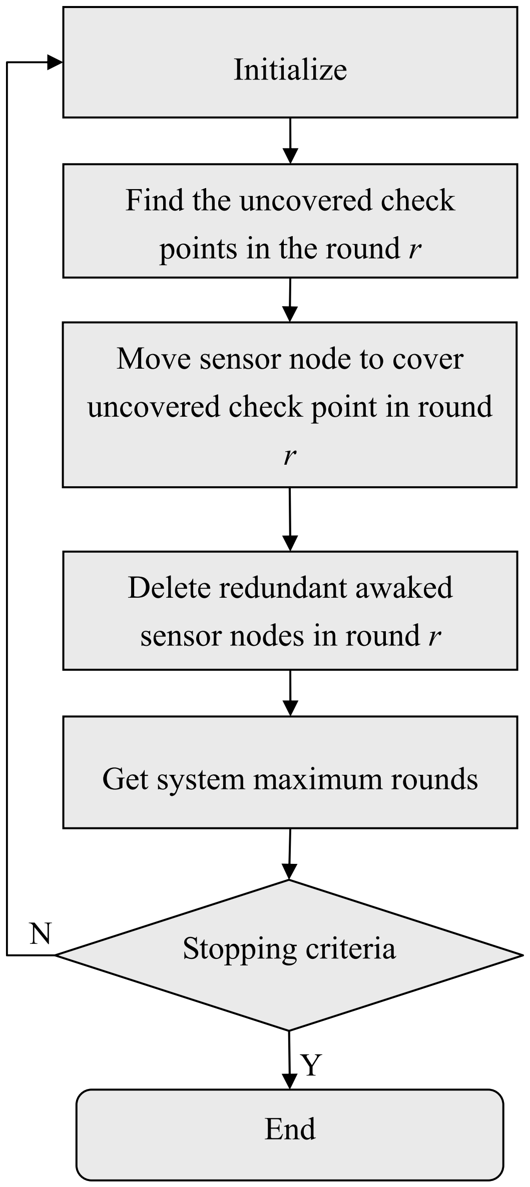

Initialize
Find the uncovered check points in the round *r*
Move sensor node to cover uncovered check point in round *r*
Delete redundant awaked sensor nodes in round *r*
Get system maximum rounds
N
Stopping criteria
Y
End

Figure 3.20. The procedure of boundary monitoring algorithm for mobile sensors.

The BMAMS includes two phases. First, the uncovered check points finding phase, we propose a heuristic algorithm for finding the uncovered check points. Second, the relocation phase, we propose a heuristic algorithm for relocating the sensor nodes with low message. The boundary monitoring algorithm for mobile sensors (BMAMS) is listed in Figure 3.21.

Algorithm Boundary Monitoring for Mobile Sensors

Input: The initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round, and *max_k*

Output: The maximum rounds (*max_round*)

```
 1: begin
 2:    for r=1 to max_k do
 3:    begin
 4:       uncovered check points finding phase();      /* phase 1 */
 5:       relocation phase();                          /* phase 2 */
 6:    end
 7: end
```

Figure 3.21. The boundary monitoring algorithm for mobile sensors.

In uncovered check points finding phase, each sensor shall check every check points. If there is any check point out of the radius of sensor, then we should put the check point into the uncovered check points set. An uncovered check point finding algorithm (UCPFA) is listed in Figure 3.22.

Algorithm Uncovered Check Point Finding

Input: Coordinate of check points and sensor nodes, and sensing radius of sensor nodes

Output: The uncovered check points

```
 1: begin
 2:    BNSet=∅
 3:    UncoverSet=∅
 4:    for j=1 to cp do
 5:    begin
 6:       c[j] = 0;
 7:       for i=1 to sn do
 8:          x[i][j] = 0;
 9:    end
10:    for j=1 to cp do
11:    begin
12:       for i=1 to sn do
13:       begin
```

```
14:            if a[j] is covered by sensor s[i]        /* $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \le r_i$ */
15:                then BNSet ← s[i], x[i][j]=1, and c[j] = c[j] +1
16:          end
17:         if c[j] =0
18:             then UncoverSet ← a[j]
19:     end
20:  end
```

Figure 3.22. The uncovered check points finding algorithm.

In the uncovered check point finding algorithm, steps 2-9 are setting initialize value, steps 10-19 find uncovered check points.

In relocation phase, we relocate previously deployed sensors to cover uncovered check points due to failure or battery exhaustion of other nodes. Let check points, *a*[*j*-1] or *a*[*j*+1], are neighbors of uncovered check points, *a*[*j*]. If either (*x*[*i*][*j*-1] = 1 and *c*[*j*-1] >=2) or (*x*[*i*][*j*+1] = 1 and *c*[*j*+1] >= 2) is satisfied for each neighbor check point, *a*[*j*-1] or *a*[*j*+1], which is covered by sensor *s*[*i*], then we move the *s*[*i*] to cover the check point *a*[*j*]. A relocation algorithm (RA) is listed in Figure 3.23.

Algorithm Relocation

Input: *c*[*j*] is number of cover for check point *j*, *x*[*i*][*j*] is 1 if *a*[*j*] is covered by sensor *s*[*i*]

Output: To determine 1) whether sensor *s* is awake or not in the round *r*, and 2) whether sensor node *s* moves to cover check point *a* or not

```
 1:  begin
 2:     for j=1 to cp do
 3:        if c[j] =0
 4:          begin
 5:            for i=1 to sn do
 6:               if (x[i][j-1] =1) and (c[j-1] >=2)
 7:                  then move s[i] to cover check point a[j]
 8:               else if (x[i][j+1] =1) and (c[j+1] >=2)
 9:                  then move s[i] to cover check point a[j]
10:               if system can not find ((x[i][j-1] =1) and (c[j-1] >=2))
```

```
                        or ((x[i][j+1] =1) and (c[j+1] >=2))
11:             then if (x[i][j-1] =1) and (c[j-1] =1)
12:                 then move s[i] to cover check point a[j]
13:         end
14: end
```

Figure 3.23. The relocation algorithm.

In the relocation algorithm, if either (x_{ij-1} = 1 and c_{j-1} >= 2) or (x_{ij+1} =1 and c_{j+1} >= 2) is satisfied, then steps 4-7 move the s_i to cover the check point a_j. Steps 8-10 move the s_i to the check point a_j, if (x_{ij-1} = 1) and (c_{j-1} = 1) and system can neither find ((x_{ij-1} = 1) and (c_{j-1} >= 2)) or ((x_{ij-1} = 1) and (c_{j-1} >= 2)).

The computational complexity of the boundary monitoring algorithm for mobile sensors in uncovered check points finding phase is $O(|S||A|)$, where $|S|$ is number of sensor nodes and $|A|$ is number of check points. In relocation phase is $O(|S||A|)$. Above steps from steps 3-6 run $|R|$ times, where $|R|$ is total number of rounds. Therefore, the computational complexity is $O(|R||S||A|)$. Hence, the computational complexity of the boundary monitoring algorithm for mobile sensors should be $O(|R||S||A|)$.

3.2.5 Computational Results

To evaluate the performance of the proposed algorithms, we conduct an experiment. The performance is assessed in terms of total number of rounds.

3.2.5.1 Scenario

The proposed algorithm is coded in C under a Dev C++ 4.9.9.2 development environment. All the experiments are performed on a Core 2 Duo 2.2GHz CPU running Microsoft Windows Vista. The algorithm is tested on a 2D sensor field. We distribute 100, 400, and 1600 sensor nodes and 36, 72, and 156 check points respectively in 2D sensor field. The radii of different sensors types s_a, s_b, s_c, and s_d are 1, 2, 3, and 4, respectively. The energy consumption of aware different sensor types s_a,

s_b, s_c, and s_d are 1, 2, 3, and 4, respectively, with linear model $e_s = r_s$; and 1, 4, 9, and 16, respectively, with quadratic model $e_s = r_s^2$. The initial energy level of each sensor node is 32. The energy consumption is 1 when sensor node moves one unit.

3.2.5.2 Experiment results

Table 3.14 and Table 3.15 show the maximum total number of rounds calculated by different algorithms. We can see that the BMAMS outperforms the BMAFS.

Table 3.14. Evaluation of improvement ratio with different number of nodes with the linear model.

Number of Nodes (check points, sensor nodes)	Monitoring Region (m^2)	BMAMS	BMAFS	Improvement Ratio to BMAFS
(36, 100)	10×10	100	94	0.06
(76, 400)	20×20	93	86	0.08
(156, 1600)	40×40	66	60	0.10

Table 3.15. Evaluation of improvement ratio with different number of nodes with the quadratic model.

Number of Nodes (check points, sensor nodes)	Monitoring Region (m^2)	BMAMS	BMAFS	Improvement Ratio to BMAFS
(36, 100)	10×10	39	29	0.35
(76, 400)	20×20	25	23	0.09
(156, 1600)	40×40	20	20	0

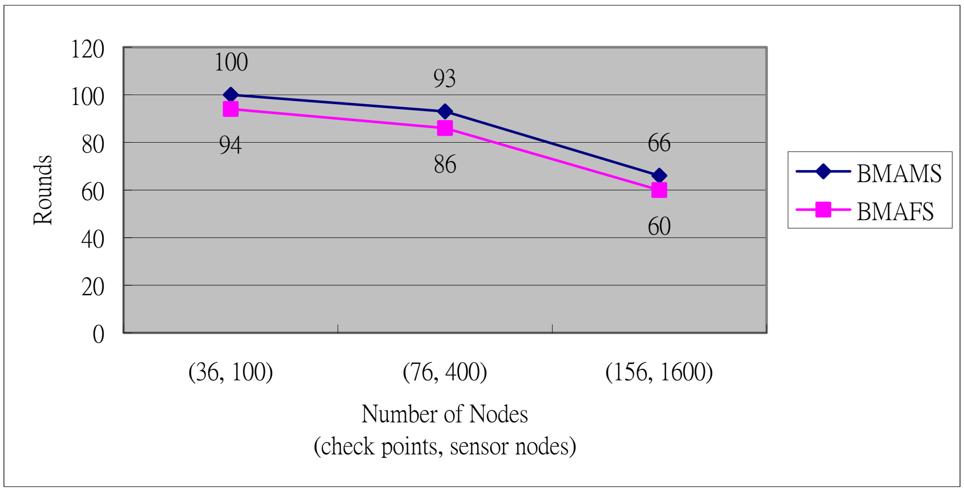


Figure 3.24. A comparison of the total number of rounds in BMAMS and BMAFS with the linear model.

3.2.5.3 Discussion

The experiment results show that the algorithm is better than the BMAFS. Compared with BMAFS, the proposed BMAMS can improve the lifetime of boundary monitoring services from 0% to 35%, as shown in Table 3.14, Table 3.15, and Figure 3.24. Therefore, the results show that the proposed algorithm can achieve boundary monitoring for mobile and grouping capabilities. Furthermore, the proposed approach can prolong system lifetime in boundary monitoring for mobile sensors. Besides, Total rounds of quadratic model $e_s = r_s^2$ are exponential decrease than total rounds of linear model $e_s = r_s$, as shown in Table 3.14 and Table 3.15.

3.2.6 Concluding Remarks

This study proposes a boundary monitoring algorithm for mobile sensors. To our best knowledge, the proposed algorithm is truly novel and it has not been yet discussed in previous researches. This study first formulates the problem as a 0/1 integer programming problem, and then proposes a heuristic-based algorithm for solving the optimization problem. The proposed approach can prolong system lifetime for wireless sensor networks of mobile and grouping capabilities. The proposed BMAMS can improve the lifetime of boundary monitoring services from 0% to 35% than BMAFS.

Chapter 4 In-Depth Defense Algorithms

In this chapter, we propose two algorithms, LDA and NLDA, to support in-depth defense services. The LDA is to construct layered defense for wireless sensor networks of grouping capabilities. It tries to find the maximum *k* groups of sensors for layered defense of the monitoring region to prolong the system lifetime. The NLDA is to construct non-layered defense of supporting different types of intruders for grouping capabilities, and it tries to find the maximum *k* groups of sensors for non-layered defense subject to the constraints of defense rate, early warning rate, battery capacity, intruder behaviors, and defender strategies. The NLDA can prolong the system lifetime and provide lead time alarms.

In this chapter, the layered defense algorithms are described in Section 4.1 and non-layered defense algorithm supporting different types of intruders is presented in Section 4.2.

4.1 Layered Defense Algorithms

In this section, we develop three algorithms to construct Layered Defense for wireless sensor networks of grouping capabilities. We try to find the maximum *k* groups of sensors for layered defense of the sensor field. The mechanism can prolong the system lifetime. This problem is formulated as a 0/1 integer-programming problem. Three heuristic-based algorithms are proposed for solving the optimization problem. The experiment results show that the proposed layered defense algorithm (LDA) gets a near optimization in the layered defense for grouping capabilities.

The rest of this chapter is organized as follows. The overview is described in Section 4.1.1. The problem and mathematical models are described in Sections 4.1.2 and 4.1.3, respectively. Additionally, the solution procedure is presented in Section

4.1.4. Furthermore, the computational results are discussed in Section 4.1.5, and conclusions are presented in Section 4.1.6.

4.1.1 Overview

In this section, we focus on the sensor grouping problem to support layered defense services. First, we try to find the nodes of each layer from the monitoring region. Second, we want to find the maximum *k* groups of sensors to monitor a layered defense in sensor networks. This mechanism can prolong the system lifetime of layered defense.

The problem is similar to that of Section 3.1. In Section 3.1, boundary monitoring algorithm for fixed sensors considers only one layer. However, in this section, layered defense algorithm considers multiple layers.

We formulate the problem as a 0/1 integer programming problem where the objective function is the maximization of the system lifetime of the layered defense subject to the constraints of defense rate, battery capacity, and integer variables.

The problem is formulated as a linear optimization-based problem with three different decision variables: wakeup sensors, covered check points, and satisfy defense rate in the round *r*. Wakeup sensor is 1 if sensor *s* is awake in the round *r*, and 0 otherwise. Covered check point is 1 if check point *a* is covered by at least one awake sensor in the layer *j* and round *r*, and 0 otherwise. Satisfy defense rate is 1 if defense rate is satisfied in the round *r*, and 0 otherwise. In the further experiments, the proposed layered defense algorithm is expected to be efficient and effective in dealing with the optimization problem.

4.1.2 Problem Description

4.1.2.1 Layered Nodes Selection Algorithm

In this section, we use the mathematical method to construct layered defense. We particularly introduce novel defense rate definition for layered defense. The monitoring region can be represented as a collection of two-dimensional check points in multiple layers, as illustrated in Figure 4.1. We assume that sensors are randomly deployed in boundary of each layer of layered defense region.

Definition 4.1 The defense rate of layer *j*: The number of covered check points (B_j) divided by the total number of check points (A_j) in layer *j*. Defense rate of layer *j* is B_j / A_j.

Definition 4.2 The defense rate of layered defense (*Q*) is $1-\prod_{j\in J}(1-\frac{B_j}{A_j})$, where *J* is total number of layers.

Definition 4.3 The early warning distance of layer *j* (m_j): The shortest distance from layer *j* to core, the protected area.

Definition 4.4 The detectability of layer *j*: The defense rate of layer *j* multiplied by early warning distance of layer *j*. The detectability of layer *j* is $\frac{B_j}{A_j}m_j$.

Definition 4.5 The detectability of layered defense (*P*) is $(\sum_{j\in J}\frac{B_j}{A_j}m_j)/|J|$, where $|J|$ is total number of layers.

The positioning resolution of application determines the granularity of check point and sensing range. The layered defense region illustrated in Figure 4.1 has 3 layers and 32 sensors are placed on the layers. For example, the defense rate of system is 1-[(1-0.8) *(1-0.9)*(1-0.95)] = 0.999 in Figure 4.1.

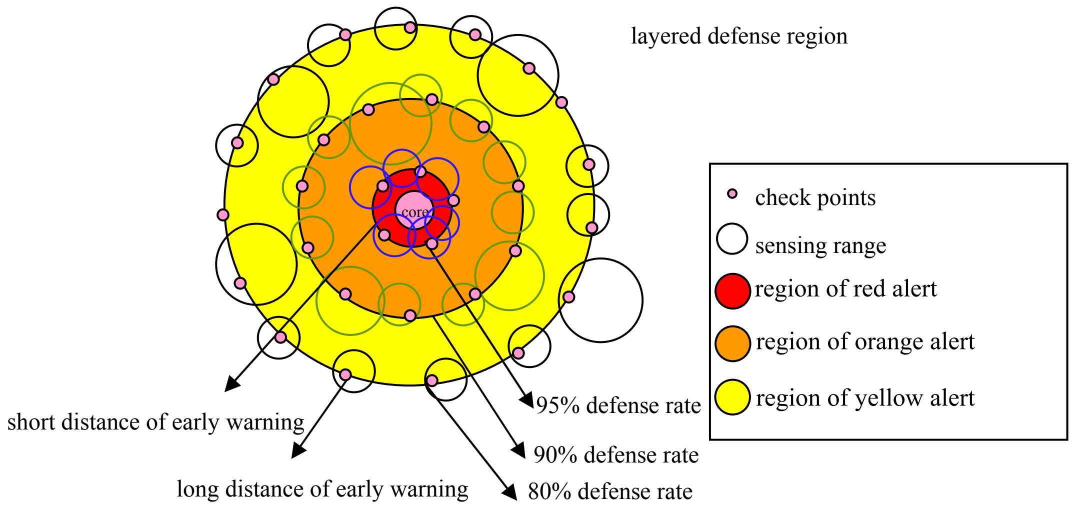


Figure 4.1. An example of layered defense.

The proposed layered nodes selection algorithm (LNSA) is shown in Figure 4.2. We aimed at each sensor checking of the whole check points. If there is any radius of sensor covered the check point, then we should put the sensor into the nodes set of layer *j*.

Algorithm Layered nodes selection

Input: Coordinates of check points and sensor nodes, and sensing radii of sensor nodes

Output: Nodes of each layer (*LNSet*[*j*])

```
 1:  begin
 2:      $LNSet[j]=\varnothing$;              /* $LNSet[j]$ is nodes set of layer $j$ */
 3:      $UncoverSet[j]=\varnothing$;          /* $UncoverSet[j]$ is uncovered check points
                                             set of layer $j$ */
 4:      for $j$ = 1 to $J$ do                /* $J$: the number of layers */
 5:          for $a$ = 1 to $cp$ do           /* $cp$: number of check points */
 6:              $flag_a^l = 0$;
 7:      for $j$ = 1 to $J$ do
 8:          for $a$ = 1 to $cp$ do
 9:          begin
10:              for $s$ = 1 to $sn$ do       /* $sn$: number of sensor node*/
11:              begin
12:                  if $check\_point_a^j$ is covered by sensor $s$
```

```
                                        /* $\sqrt{(x_s - x_a)^2 + (y_s - y_a)^2} \le r_s$ */
13:                         then $LNSet[j]$ ← $s$ and $flag_a^j = 1$
14:                 end
15:                 if $flag_a^j = 0$
16:                     then $UncoverSet[j]$ ← $check\_point_a^j$
17:             end
18:             if $1 - \prod_{j \in J}(1 - |LNSet_j| / |A_j|) \ge D$   /* $D$: The total defense rate */
19:                 then defense rate is satisfied and
                        nodes of layer $j = LNSet[j]$
20:                 else defense rate is not satisfied
21:   end
```

Figure 4.2. The layered nodes selection algorithm.

In this algorithm, from steps 2-6 set initialize values. Steps 7-17 are used to find node set of each layer. Steps 16-20 check defense rate is satisfied.

The computational complexity of the layered nodes selection algorithm at steps 4-6 is $O(|J||A|)$, where $|J|$ is the number of layers and $|A|$ is number of check points. From steps 7-20 is $O(|J||A||S|)$, where $|S|$ is number of sensor nodes. Therefore, the computational complexity is $O(|J||A||S|)$. Hence, the computational complexity of the layered nodes selection algorithm should be $O(|J||A||S|)$.

We use the above LNSA to find out layered nodes and check whether total defense rate is satisfied.

4.1.2.2 Layered defense Algorithms for Grouping Capabilities

We try to find maximum k sets of sensors to support layered defense services on layered defense region, as shown in Figure 4.3. Each of them, is called a group, can provide defense rate is satisfied of the layered defense region. Each group is activated in turn to monitor the layered defense region. Each group is activated in turn to monitor the monitoring region as illustrated in Figure 3.9 of Section 3.1. From the network viewpoint, two operation states exist: the sleeping state and the active state.

Only one group sensors are activated to monitor the layered defense region, and the other group sensors are sleeping at the same time. The system lifetime can be effectively prolonged to *k* times. The detailed descriptions are shown in Table 4.1.

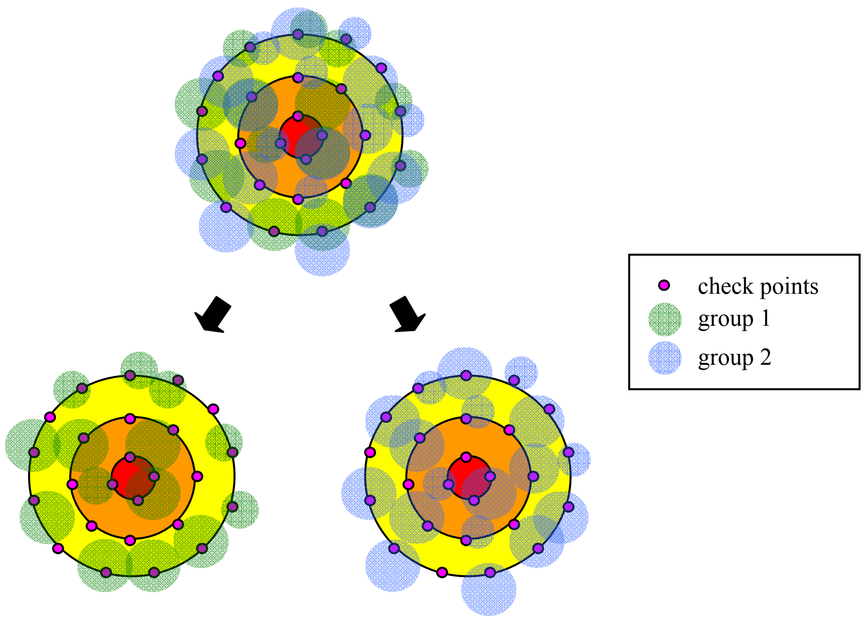


Figure 4.3. An example of layered defense for grouping capabilities.

Table 4.1. Problem description in layered defense problem.

Given	1. The set of check points. 2. The set of sensor nodes. 3. Initial energy level of sensor node. 4. Energy consumption for sensor nodes to sense data in each round. 5. Detection radius of sensor. 6. Total number of layers. 7. Total defense rate. 8. The detectability.
Objective	To maximize the layered defense services lifetime.
Subject to	1. Total defense rate. 2. The detectability of system. 3. Battery capacity.
To determine	To determine whether sensor *s* is awake or not in the round *r*.

4.1.3 Mathematical Model

In this section, we formulate the problem as a 0/1 integer programming problem where the objective function is the maximization of the amount of cover *k* required to satisfy defense rate under a given layered defense region. The problem is a variant of the set *k*-cover problem and thus is NP-complete [35].

The notations used to model the problem are listed in Table 4.2 and Table 4.3.

Table 4.2. Notations of the given parameters in layered defense problem.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
A_j	Index set of the service check points of layer *j* in the layered defense.
C_s	The initial energy level of sensor node *s*.
E_m	The energy consumption for sensor nodes to sense data.
b_{saj}	The indicator function which is 1 if the check point *a* is in the radius of the sensor node *s* on layer *j*, and 0 otherwise.
R	The upper bound number of rounds.
J	The total number of layers.
d_j	The defense rate of layer *j*.
Q	The total defense rate.
m_j	The distance of early warning of layer *j*.
P	The detectability of system.

Table 4.3. Notations of the decision variables in layered defense problem.

Decision Variables	
Notation	Description
π_{sr}	1 if sensor *s* is awake in the round *r*, and 0 otherwise.
y_{arj}	1 if check point *a* at least is covered by one awake sensor on layer *j* in the round *r*, and 0 otherwise.
z_r	1 if satisfy total defense rate in the round *r*, and 0 otherwise.

Problem (IP):

$$\max \sum_{\forall r \in R} z_r \tag{IP}$$

subject to:

The defense rate constraints

$$y_{arj} \le \sum_{s\in S} b_{saj}\pi_{sr} \qquad \forall a\in A, r\in R, j\in J \qquad (1)$$

$$1-\prod_{j\in J}\left(1-\frac{\sum_{a\in A} y_{arj}}{|A_j|}\right) \ge \mathrm{D} \qquad \forall r\in R \qquad (2)$$

$$z_r-\left(\left(1-\prod_{j\in J}\left(1-\frac{\sum_{a\in A} y_{arj}}{|A_j|}\right)\right)-D\right) \le 1 \qquad \forall r\in R \qquad (3)$$

The detectability constraint

$$z_r-\frac{\dfrac{\sum_{j\in J}\left(\dfrac{\sum_{a\in A} y_{arj}}{|A_j|}*m_j\right)}{J}-P}{\dfrac{\sum_{j\in J}\left(\dfrac{\sum_{a\in A} y_{arj}}{|A_l|}*m_j\right)}{J}+P} \le 1 \qquad \forall r\in R \qquad (4)$$

The battery capacity constraint

$$\sum_{r\in R}\pi_{sr}E_m \le C_s \qquad \forall s\in S \qquad (5)$$

The integer constraints

$$\pi_{sr} = 0 \text{ or } 1 \qquad \forall s\in S,\ r\in R \qquad (6)$$

$$y_{arj} = 0 \text{ or } 1 \qquad \forall a\in A, r\in R, j\in J \qquad (7)$$

$$z_r = 0 \text{ or } 1 \qquad \forall r\in R. \qquad (8)$$

The objective function is to maximize the system lifetime of the given sensor network. The lifetime is defined as the total number of rounds.

Constraints (1)-(3): If defense rate constraint is satisfied then enforce z_r=1.
Constraint (4): The detectability constraint.
Constraint (5): For each sensor node s, the total sensing consumption can not exceed its initial energy level.
Constraints (6)-(8): The integer constraints for decision variables π_{sr}, y_{arj}, and z_r.

4.1.4 Solution Procedure

The parameters and decision variables used to model layered defense algorithms in this section are listed in Table 4.4.

Table 4.4. The parameters and decision variables in algorithms of layered defense problem.

Notation	Description
max_k	The upper bound of system lifetime.
J	The total number of layers
D	The total defense rate.
P	The detectability of system.
m_j	The distance of early warning of layer *j*.
layer[*r*]	The number of layers that satisfy defense rate of layer in the round *r*.
cp[*j*]	The number of check points on layer *j*.
sn	The number of sensor nodes.
cpc_no[*a*]	The number of covered rounds in each check point *a.*
cs[*s*]	The initial energy level of sensor node *s.*
es[*s*]	The energy consumption for aware sensor node *s* to sense data in each round.
max_round	The system lifetime.
c_bsa[*a*]	The number of covered times in check point *a.*
count[*s*]	The number of covered check points by awaked sensor *i.*
c_s[*i*]	The number of covered check points under sensing range of sensor *i.*
sat_ldr[*r*]	The number of covered check points by awaked sensor in round *r*.
bsa[*s*][*a*][*j*]	The indicator function which is 1 if the check point *a* is in the sensing range of the sensor node *s* in layer *j*, and 0 otherwise.
sat_dr[*r*]	The decision variable which is 1 if total defense rate is satisfied in the round *r*, and 0 otherwise.
p[*s*][*r*]	The decision variable which is 1 if sensor *s* is awake in the round *r*, and 0 otherwise.
cover[*a*][*r*][*j*]	The decision variable which is 1 if layer *j* check point *a* is covered by at least one awake sensor in the round *r*, and 0 otherwise.

4.1.4.1 Simple Algorithm 1

We compare the proposed iteration-based algorithm with non-iteration-based algorithms (simple algorithm 1 and 2) that use the concept of “cover” to determine whether sensor *s* is awake or not in the round *r*. The “cover” is 1 if the check point *a* is in the sensing range of the sensor node *s*, and 0 otherwise.

In each round, we first find sensor *s* to cover check point *a*, and then sensor *s* is awake in the round *r*, and repeat the assignment process until total defense rate and detectability are satisfied in round *r*.

The simple algorithm 1 (SA1) is listed in Figure 4.4.

Algorithm Simple 1

Input: The initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round

Output: The maximum rounds (*max_round*)

```
 1: begin
 2:       max_round=0;
 3:       for r=1 to max_k do
 4:       begin
 5:             sat_dr[r]=0;
 6:             layer[r]=0;
 7:             for j=1 to J do
 8:                   sat_ldr[r][j]=0;
 9:             for s=1 to sn do
10:                   p[s][r]=0;
11:       end
12:       for r=1 to max_k do
13:       begin
14:             for s=1 to sn do
15:             begin
16:                   for j=1 to J do
17:                   begin
18:                         for a=1 to cp[j]do
19:                         begin
20:                               if ((bsa[s][a][j]=1) and (cs[s]>=es[s])
                                        and (cover[a][r][j]=0)) then
21:                               begin
```

```
22:                         if $p[s][r]=0$ then
23:                         begin
24:                             $p[s][r]=1$;
25:                             $cs[s]=cs[s]-es[s]$;
26:                         end
27:                         for $a=1$ to $cp[j]$ do
28:                             if (($bsa[s][a][j]=1$) and ($cover[a][r][j]=0$))
29:                                 then $cover[a][r][j]=1$ and
                                        $sat\_ldr[r][j]=sat\_ldr[r][j]+1$;
30:                     end
31:                 end
32:             end
33:             if (($1-\prod_{j\in J}(1-(sat\_ldr[r][j]/cp[j]))\ge D$) and ($\sum_{j\in J}(sat\_ldr[r][j]/cp[j])m_j/J\ge P$))
34:                 then $sat\_dr[r]=1$ and break;
35:         end
36:     for $r=1$ to $max\_k$ do
37:         if ($sat\_dr[r]=1$)
38:             then total defense rate is satisfied in round $r$ and
                    $max\_round=max\_round+1$;
39: end
```

Figure 4.4. The simple algorithm 1 of layered defense.

In the simple algorithm 1 of layered defense, steps 2-11 are setting initialize value, steps 12-32 determine whether sensor *s* is awake or not in the round *r*. Steps 33-34 are used to check whether total defense rate is satisfied in round *r*. Steps 36-38 are used to get system maximum rounds.

The computational complexity of the simple algorithm 1 in layered defense at steps 3-11 is $O(|R|)$, where $|R|$ is total number of rounds. At steps 12-32 is $O(|R||S||J||A|^2)$, where $|S|$ is number of sensor nodes, $|J|$ is number of layers and $|A|$ is number of check points. From steps 36-38 is $O(|R|)$. Therefore, the computational complexity is $O(|R||S||J||A|^2)$. Hence, the computational complexity of the simple algorithm 1 in layered defense problem should be $O(|R||S||J||A|^2)$.

4.1.4.2 Simple Algorithm 2

Simple algorithm 1 wastes on energy consumption, because system has redundant awaked sensor nodes. Therefore, we propose a simple algorithm 2 (SA2) to deal with the problem. An example of deleting redundant awaked sensor node as illustrated in Figure 3.12 of Section 3.1.

The simple algorithm 2 is listed in Figure 4.5.

Algorithm Simple 2

Input: The initial energy level of sensor node *s*, the energy consumption for awaked sensor node *s* to sense data in each round

Output: The maximum rounds (*max_round*)

```
 1: begin
 2:     max_round=0;
 3:     for r=1 to max_k do
 4:     begin
 5:         sat_dr[r]=0;
 6:         for s=1 to sn do
 7:             p[s][r]=0;
 8:     end
 9:     for r=1 to max_k do
10:     begin
11:         for s=1 to sn do
12:             for j=1 to J do
13:                 for a=1 to cp do
14:                     if ((bsa[s][a][j]=1) and (cs[s]>=es[s]) and (cover[a][r][j]=0))
                        then
15:                     begin
16:                         p[s][r]=1;
17:                         cs[s]=cs[s]-es[s];
18:                         for k=1 to cp do
19:                             if (bsa[s][a][j]=1)
20:                                 then c_bsa[a]=c_bsa[a]+1;
21:                             if ((bsa[s][a][j]=1) and ( cover[a][r][j]=0))
22:                                 then cover[a][r][j]=1 and
                                          sat_ldr[r][k]=sat_ldr[r][k]+1;
23:                     end
24:             for s=1 to sn do                    /* delete redundant nodes */
25:             begin
26:                 for a=1 to cp do
```

```
27:                     begin
28:                         if (($p[s][r]$=1) and ($bsa[s][a][j]$=1) and ($c\_bsa[a]$>=2))
29:                             then $count[s]$=$count[s]$+1;
30:                         if ($count[s]$=$c\_s[i]$)
31:                         begin
32:                             $cs[s]$=$cs[s]$+$es[s]$;  /* energy recovery */
33:                             $p[s][r]$=0;
34:                             for $a$=1 to $cp$ do
35:                                 if ($bsa[s][a][j]$==1)
36:                                     then $c\_bsa$[a]=$c\_bsa[a]$-1;
37:                         end
38:                     end
39:                   end
40:                 end
41:                 if (($1-\prod_{j\in J}(1-(sat\_ldr[r][j]/cp[j]))\geq D$) and ($\sum_{j\in J}(sat\_ldr[r][j]/cp[j])m_j/J\geq P$))
42:                     then $sat\_dr[r]$=1 and break;
43:         end
44:         for $r$=1 to $max\_k$ do
45:             if ($sat\_dr[r]$=1)
46:                 then total defense rate is satisfied in round $r$ and
                        $max\_round$=$max\_round$+1;
47:     end
```

Figure 4.5. The simple algorithm 2 of layered defense.

In the simple algorithm 2, steps 2-8 are setting initialize value, steps 9-23 determine whether sensor *s* is awake or not in the round *r*. Steps 24-40 are used to delete redundant awaked sensor nodes. Steps 41-42 are used to check whether total defense rate is satisfied in round *r*. Steps 44-46 are used to get system maximum rounds.

The computational complexity of the simple algorithm 2 in layered defense at steps 3-8 is $O(|R|)$, where $|R|$ is total number of rounds. At steps 9-43 is $O(|R||S|^2|J||A|^2)$, where $|S|$ is number of sensor nodes, $|J|$ is number of layers and $|A|$ is number of check points. From steps 44-46 is $O(|R|)$. Therefore, the computational complexity is $O(|R||S|^2|J||A|^2)$. Hence, the computational complexity of the simple algorithm 2 in layered defense problem should be $O(|R||S|^2|J||A|^2)$.

4.1.4.3 Layered Defense Algorithm

In this section, we present an iteration-based layered defense algorithm (LDA) to improve SA1 and SA2 algorithms.

To solve the original problem near-optimally. In each round, we first use different sensor node *id* to cover first check point *a* and then sensor *s* is awake in the round *r*, and repeat the assignment process until total defense rate and detectability are satisfied in round *r*. We improve the object function by solving the problem optimally and use the different sensor node *id* to improve the maximum rounds per iteration.

The procedure of layered defense algorithm is shown in Figure 4.6. First of all is to initialize. Second is to determine whether sensor *s* is awake or not in the round *r*. Third is to delete redundant awaked sensor nodes. Forth is to check whether defense rate is satisfied in round *r* or not. Fifth is to get system maximum rounds. Finally is to check whether it is a stop condition or not. If the answer is negative, go back to the first step.

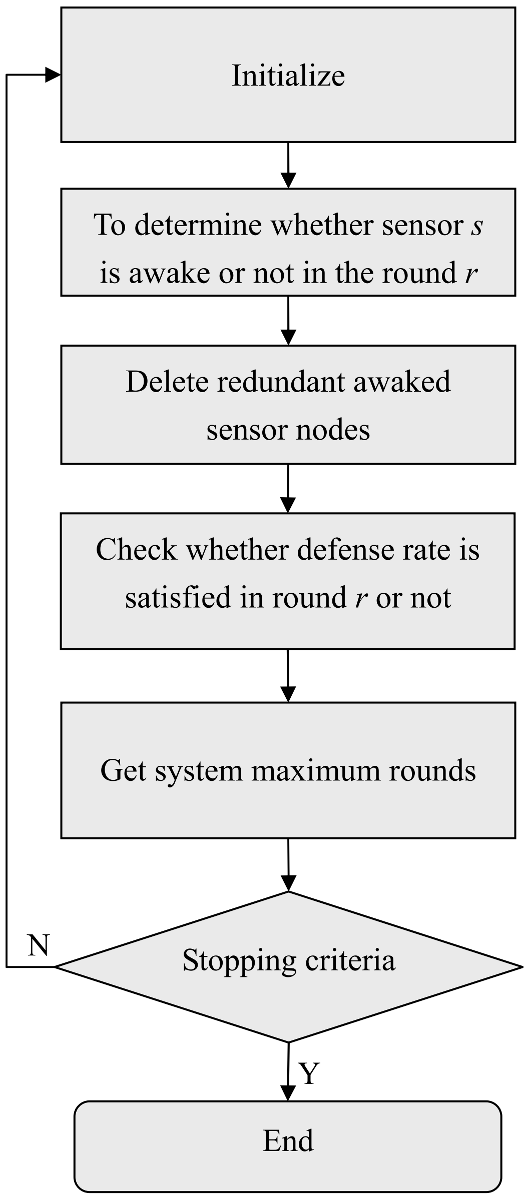

Initialize
To determine whether sensor *s* is awake or not in the round *r*
Delete redundant awaked sensor nodes
Check whether defense rate is satisfied in round *r* or not
Get system maximum rounds
Stopping criteria
N
Y
End

Figure 4.6 The procedure of layered defense algorithm.

The layered defense algorithm is listed in Figure 4.7.

Algorithm Layered Defense

Input: The initial energy level of sensor node *s*, the energy consumption for awaked sensor node *s* to sense data in each round

Output: The maximum rounds (*max_round*)

```
 1: begin
 2:         for iteration=1 to sn do
 3:         begin
 4:                 for r=1 to max_k do
 5:    begin
 6:        sat_dr[r]=0;
 7:        for s=1 to sn do
 8:            p[s][r]=0;
 9:    end
10:    for r=1 to max_k do
11:    begin
12:        s=iteration;
13:        for i=1 to sn do
14:          for j=1 to J do
15:            for a=1 to cp do
16:                if ((bsa[s][a][j]=1) and (cs[s]>=es[s]))
                       and (cover[a][r][j]=0)) then
17:                begin
18:                    p[s][r]=1;
19:                    cs[s]=cs[s]-es[s];
20:                    for k=1 to cp do
21:                      if (bsa[s][a][j]=1)
22:                          c_bsa[a]=c_bsa[a]+1;
23:                          if ((bsa[s][a][j]=1) and (cover[a][r][j]=0))
24:                              then cover[a][r][j]=1 and
                                     sat_dr[r]=sat_dr[r]+1;
25:                    s=(s+1)%sn;
26:                end
27:        if (sat_dr[r]=cp) then                /* delete redundant nodes */
28:            for s=1 to sn do
29:            begin
30:                for a=1 to cp do
31:                    if ((p[s][r]=1) and (bsa[s][a][j]=1)
                           and (c_bsa[a]>=2))
32:                      then count[s]=count[s]+1;
33:                    if (count=c_s[i]) then
34:                    begin
35:                        cs[s]=cs[s]+es[s];    /* energy recovery */
36:                         p[s][r]=0;
37:                         for a=1 to cp do
38:                             if (bsa[s][a][j]=1)
39:                                 c_bsa[a]=c_bsa[a]-1;
40 :                   end
41:              end
42:            end
```

```
43:             if $((1-\prod_{j\in J}(1-(sat\_ldr[r][j]/cp[j]))\ge D)$ and $(\sum_{j\in J}(sat\_ldr[r][j]/cp[j])m_j/J \ge P))$
44:                 then $sat\_dr[r]$=1 and break;
45:         end
46:         for r=1 to $max\_k$ do
47:             if ($sat\_dr[r]$=1)
48:                 then total defense rate is satisfied in round $r$ and
                    $max\_round$=$max\_round$+1;
49:     end
50: end
```

Figure 4.7. The layered defense algorithm.

In the layered defense algorithm, steps 2, 12, and 25 are iteratively to improve system maximum rounds. Steps 4-9 are to set initial values, steps 10-11 and 13-24 are to determine whether sensor *s* is awake or not in the round *r*. Steps 27-42 are used to delete redundant awaked sensor nodes. An example of deleting redundant awaked sensor node as illustrated in Figure 3.12 of Section 3.1. Steps 43-44 are used to check whether total defense rate is satisfied in round *r*. Steps 46-48 are used to get system maximum rounds.

The computational complexity of the layered defense algorithm at steps 4-9 is $O(|R|)$, where $|R|$ is total number of rounds. At steps 10-45 is $O(|R||S|^2|J||A|^2)$, where $|S|$ is number of sensor nodes, $|J|$ is the number of layers and $|A|$ is number of check points. From steps 46-48 is $O(|R|)$. Above steps from steps 2-49 run $|S|$ times. Therefore, the computational complexity is $O(|R||S|^3|J||A|^2)$. Hence, the computational complexity of the layered defense algorithm should be $O(|R||S|^3|J||A|^2)$. This makes the algorithm scalable to a large scale WSNs.

4.1.5 Computational Results

To evaluate the performance of the proposed algorithms, we conduct an experiment. The performance is assessed in terms of total number of rounds.

4.1.5.1 Scenario

The proposed algorithms are coded in C under a Dev C++ 4.9.9.2 development environment. All the experiments are performed on a Core 2 Duo 2.2GHz CPU running Microsoft Windows Vista. The algorithm is tested on a 2D sensor field. We distribute 1600 and 6400 sensor nodes and 720 and 1440 check points respectively in 2D sensor field.

4.1.5.2 Experiment results

Figure 4.8 and Figure 4.9 show the example of layered defense strategies. (Defense rate =0.8 and 0.999)

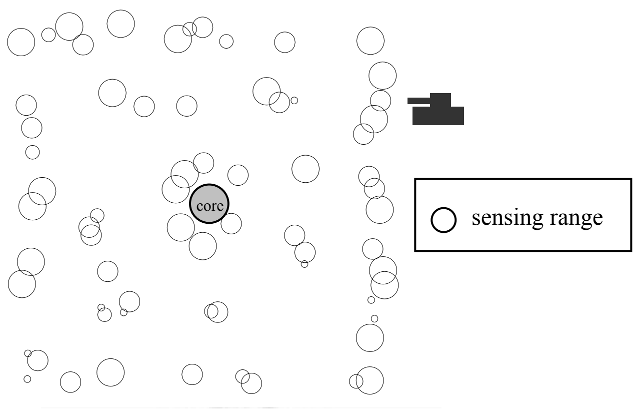


Figure 4.8. An example of layered defense. (defense rate = 0.8)

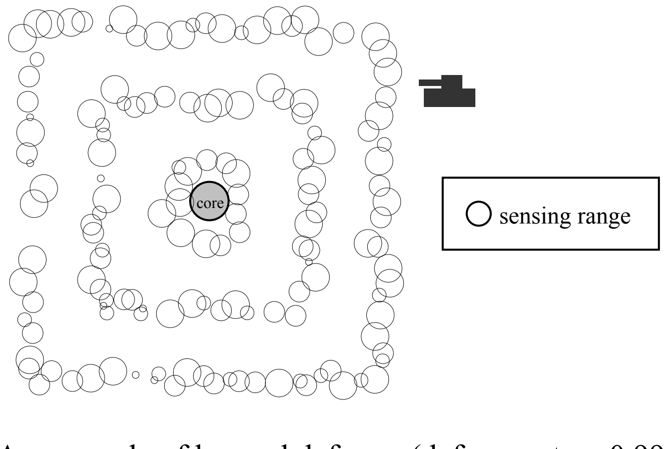


Figure 4.9. An example of layered defense. (defense rate = 0.999)

We set detectability = $\dfrac{\sum_{j\in J}((1-\sqrt[j]{(1-D)})*m_j)}{J}$. Table 4.5 and Table 4.6 show the maximum total number of rounds calculated by different algorithms. We can see that the LDA outperforms the SA1 and SA2 algorithms.

Table 4.5. Evaluation of the improvement ratio with the linear model. (Defense rate = 0.8)

Number of Nodes (check points, sensor nodes)	D	LDA	SA1	Improvement Ratio to SA1	SA2	Improvement Ratio to SA2
(720, 1600)	0.7	78	70	0.11	76	0.03
(720, 1600)	0.8	60	52	0.15	54	0.11
(720, 1600)	0.9	42	36	0.17	40	0.05
(720, 1600)	0.99	18	18	0	18	0

Table 4.6. Evaluation of the improvement ratio with quadratic model. (Defense rate = 0.8)

Number of Nodes (check points, sensor nodes)	D	LDA	SA1	Improvement Ratio to SA1	SA2	Improvement Ratio to SA2
(720, 1600)	0.7	27	25	0.08	26	0.04
(720, 1600)	0.8	19	17	0.12	18	0.06
(720, 1600)	0.9	13	11	0.18	12	0.08
(720, 1600)	0.99	5	5	0	5	0

4.1.5.3 Discussion

The experiment results show that the algorithm is better than the other heuristic algorithms, such as SA1 and SA2 algorithms. Compared with SA1 and SA2 algorithms, the proposed LDA can improve the percentage of energy consumption from 0% to 18%, as shown in Table 4.5 and Table 4.6. Therefore, the results show that the proposed algorithm can achieve layered defense for grouping capabilities. Furthermore, the algorithm is very efficient and scalable in terms of the running time. Besides, Total rounds of quadratic model $e_s = r_s^2$ are exponential decrease than total rounds of linear model $e_s = r_s$, as shown in Table 4.5 and Table 4.6.

4.1.6 Concluding Remarks

This study proposes a layered defense algorithm in wireless sensor networks. To our best knowledge, the proposed algorithm is truly novel and it has not been yet discussed in previous researches. This study first formulates the problem as a 0/1 integer programming problem, and then proposes a heuristic-based algorithm for solving the optimization problem.

4.2 Non-Layered Defense Algorithms

In this section, we focus on non-layered defense for wireless sensor networks of grouping capabilities. We try to find the maximum *k* groups of sensors for non-layered defense subject to defense rate, early warning rate, battery capacity, intruder behavior, and defender strategies constraints. The mechanism can prolong the system lifetime and provide lead time alarms [42]. The problem is modeled as a generic mathematical programming problem. A novel solution procedure of three phases, which well combines mathematical programming and simulation techniques, is proposed. The experiment results show that the proposed non-layered defense algorithm (NLDA) gets applicability and effectiveness in the non-layered defense for grouping capabilities.

The rest of this section is organized as follows. The overview is described in Section 4.2.1. The problem and mathematical models are described in Sections 4.2.2 and 4.2.3, respectively. In addition, the solution procedure is presented in Section 4.2.4. Furthermore, the computational results are discussed in Section 4.2.5, and conclusions are presented in Section 4.2.6.

4.2.1 Overview

In this section, we focus on the sensor grouping problem to support non-layered defense services. First, we try to find out the sensors nodes to cover the monitoring region for non-layered defense and early warning rate. Second, we will describe the behavior of intruders. Third, we want to describe the defender strategies. Forth, we want to find the maximum *k* groups of sensors for non-layered defense in sensor networks. This mechanism can prolong the system lifetime.

The problem is modeled as a generic mathematical programming problem, and a novel solution of three phases, which well combines mathematical programming and simulation techniques, is proposed. In the first phase, the “initial solution phase”, we propose an efficient heuristic algorithm for initial solution. In the second phase, the “objective function evaluation phase”, we propose efficient and effective simulations to evaluate the effectiveness of the current defense policy. In the third phase, the “add-and-drop phase”, we use an add-and-drop algorithm to improve and satisfy the defender strategies. From experiments in WSNs, applicability and effectiveness of the proposed framework and algorithms are clearly demonstrated.

In this section, we use the concept of check point, which can check full coverage and coverage rate of each layer. Besides, it can save energy consumption because the concept can check full coverage and coverage rate of each layer more efficiently for arbitrary topology. The concept of check points is introduced in Section 3.1.1. And further, we find the maximum k sets of sensors to support non-layered defense services on the monitoring region. These sets can be joint or disjoint. Each of them, is called a group, can provide full coverage of the boundary of the sensor field. Each group is activated in turn to monitor the each layer of non-layered defense regions. Generally, the power consumption for inactive sensors can be neglected, and the system lifetime can be effectively prolonged to k times. We present a mathematical model to describe the optimization problem and a heuristic-based algorithm is proposed to solve the problem.

To the best of our knowledge, this work is the first effort to model the non-layered defense with consideration of behaviors of intruders in wireless sensor networks. We formulate the problem as a generic mathematical programming problem where the objective function is the maximization of the system lifetime of non-layered defense subject to defense rate, early warning rate, battery capacity, intruder behavior, and defender strategies constraints. We construct a heuristic-based algorithm to solve the problem.

The problem is formulated as an optimization-based problem with two different main decision variables: wakeup sensor s in the round r and satisfying defense policies $\overline{F}$ in the round r. wakeup sensor s in the round r is 1 if sensor s is awake in the round r, and 0 otherwise. Satisfying defense policies $\vec{F}$ in the round r is 1 if total defense rate and early warning rate in the round r are satisfied, and 0 otherwise. In the further experiments, the proposed non-layered defense for grouping capabilities algorithm is expected to be efficient and effective in dealing with the optimization problem.

4.2.2 Problem Description

4.2.2.1 Non-Layered Defense for Grouping Capabilities

In this section, we describe the problem and propose the intruder and defense scenario with specific assumptions. The definitions use in the proposed non-layered defense algorithm, they are illustrated as follows:

Definition 4.6 The defense rate of non-layered defense (D): The number of detected intruders (G) divided by the total number of intruders (K). The defense rate of non-layered defense $D = G / K$.

Definition 4.7 The early warning rate (W): The number of detected intruders (H) satisfied early warning distance L divided by the total number of intruders (K). Early warning rate $W = H / K$.

For example, assume the defense rate is 0.9 and the early warning rate is 0.8. If defenders deploy the topology of sensor to satisfy the condition, then the strategies can prevent 90% intruders and satisfy 80% early warning. Defenders use the defense strategies to protect core field. Furthermore, the defense strategies can support object tracking and detect airborne intruders.

We try to find maximum k sets of sensors to support non-layered defense services, as shown in Figure 4.10. Each of them, called a group, can satisfy total defense rate and early warning rate of the monitoring region. Each group is activated in turn to monitor the monitoring region as illustrated in Figure 3.9 of Section 3.1. From the network viewpoint, two operation states exist: the sleeping state and the active state.

Only one group sensors are activated to monitor the monitoring region, and the other group sensors are sleeping at the same time. The system lifetime can be effectively prolonged to *k* times. We assume that sensors are randomly deployed in non-layered defense region.

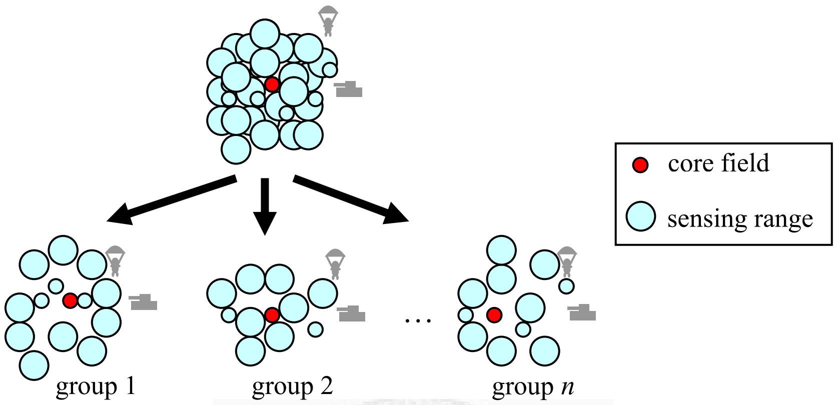


Figure 4.10. The non-layered defense model.

The objective of each intruder is to attack the core field in the given sensor network. The defender has perfect knowledge of the sensor network. The defender tries to find the maximum *k* groups of sensors for non-layered defense subject to defense rate, early warning rate, battery capacity, intruder behavior, and defender strategies constraints. However, the intruders are not aware that the defender has deployed topology in the sensor network; in other words, their knowledge of the network is imperfect. In addition, we assume that each intruder only has information about the core field location. The detailed descriptions are shown in Table 4.7.

Table 4.7. Problem description in non-layered defense problem.

Given	1. The set of sensor nodes. 2. Initial energy level of sensor node. 3. Energy consumption for sensor nodes to sense data in each round. 4. Detection radius of sensor. 5. The total evaluation number of times for all intruder

	categories in each round. 6. All possible defense strategies. 7. All possible intrusion strategies. 8. Total defense rate. 9. Distance of early warning. 10. Early warning rate. 11. Location of core field. 12. False positive rate. 13. False negative rate.
Objective	To maximize the non-layered defense services lifetime.
Subject to	1. Total defense rate. 2. Early warning rate. 3. Battery capacity.
To determine	To determine whether sensor *s* is awake or not in the round *r*.

As mentioned earlier, we classify intruders based on their attack behaviors. The behaviors are as follows:

4.2.2.2 Behaviors of Intruders

We describe the behaviors of intruders as follows.

1. Motion model

Gauss-Markov motion model is introduced in Section 2.1.4. We set the value of *α* by the modified logistic function, as shown in Figure 4.11 and Figure 4.12. Because most intruder attack path is near a straight-line. The modified logistic function is shown as follow:

$$\alpha = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}, \text{ where } 0 \le x \le 1 \text{ and } \lambda \ge 6$$

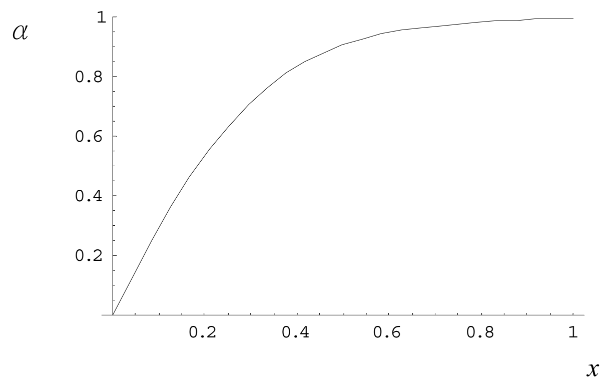


Figure 4.11. The curve of modified logistic function. (λ = 6)

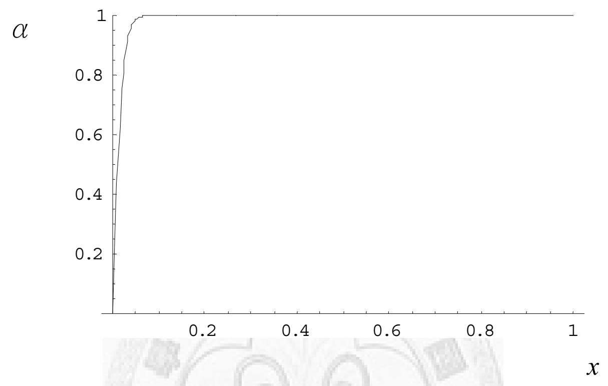


Figure 4.12. The curve of modified logistic function. (λ = 100)

2. Deviant angle and deviant range

Intruders eventually move to core field, because intruders know core field location. We propose deviant angle and deviant range to fulfill this assumption. Figure 4.13 shows the deviant angle and deviant range. The trajectory of intruders is shown in Figure 4.14.

Definition 4.8 Deviant angle d_a = $\arccos((a^2 + b^2 - c^2)/2ab)$, where a= $\sqrt{(x_{core} - x_{cur})^2 + (y_{core} - y_{cur})^2}$, b= $\sqrt{(x_{core} - x_{init})^2 + (y_{core} - y_{init})^2}$, and c= $\sqrt{(x_{cur} - x_{init})^2 + (y_{cur} - y_{init})^2}$, (x_{core}, y_{core}) is coordinate of core, (x_{cur}, y_{cur}) is current coordinate, (x_{init}, y_{init}) is coordinate of initial intruder, the deviant angle is found by *law of cosines*.

Definition 4.9 Deviant range (d_r): The controlled parameter. It is used to ensure that intruders eventually move to core field.

If deviant angle is greater than or equal to deviant range, we set the $x_{init} = x_n$, $y_{init} = y_n$, and $theta_n$=$\arctan((y_n - y_{core})/(x_n - x_{core})) + \pi$, where (x_{core}, y_{core}) is the coordinate of core, (x_n, y_n) is the current coordinate, (x_{init}, y_{init}) is the coordinate of the initial intruder, and $theta_n$ is the new intrusion angle. We set (x_n, y_n) is the coordinate of the initial intruder ($x_{init} = x_n$, $y_{init} = y_n$). The mechanism can ensure that intruders eventually move to core field.

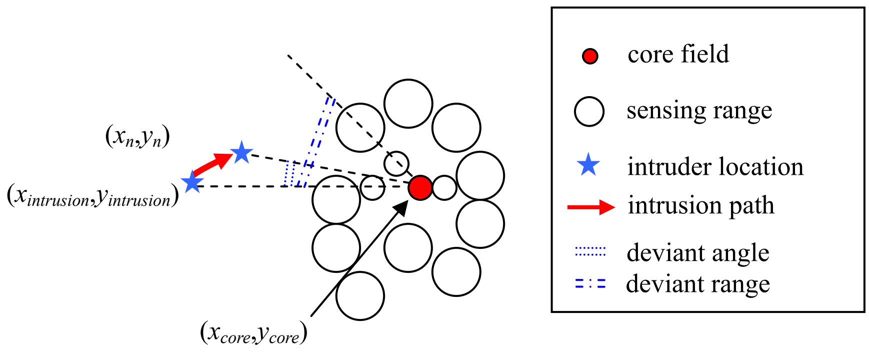


Figure 4.13. The deviant angle and deviant range.

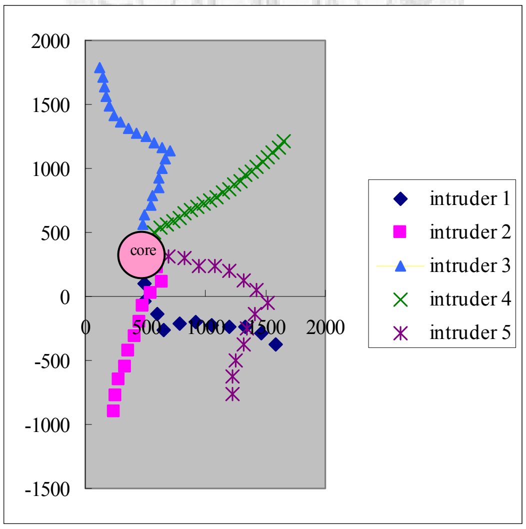


Figure 4.14. The trajectory of intruders. (deviant range = $\pi/6$)

3. Intrusive angle

An intrusive angle model uses one tuning parameter to vary the degree of randomness in the intrusive angle pattern by using the random distribution. The initial angle and location are as follows:

$$\text{angle of initial location } (\theta) = \text{randomize}(0\sim1) * \text{range of intrusive angle}$$
$$\text{initial location } x = x_{core} + r\cos\theta$$
$$\text{initial location } y = y_{core} + r\sin\theta$$

where randomize (0 ~ 1) is random number between 0 and 1, (x, y) and (x_{core}, y_{core}) are the x and y coordinates of the initial position of intruder, the x_{core} and y_{core} coordinates of the core position, and r is distance between core and initial position of intruder. Figure 4.15 shows the initial position of intruders.

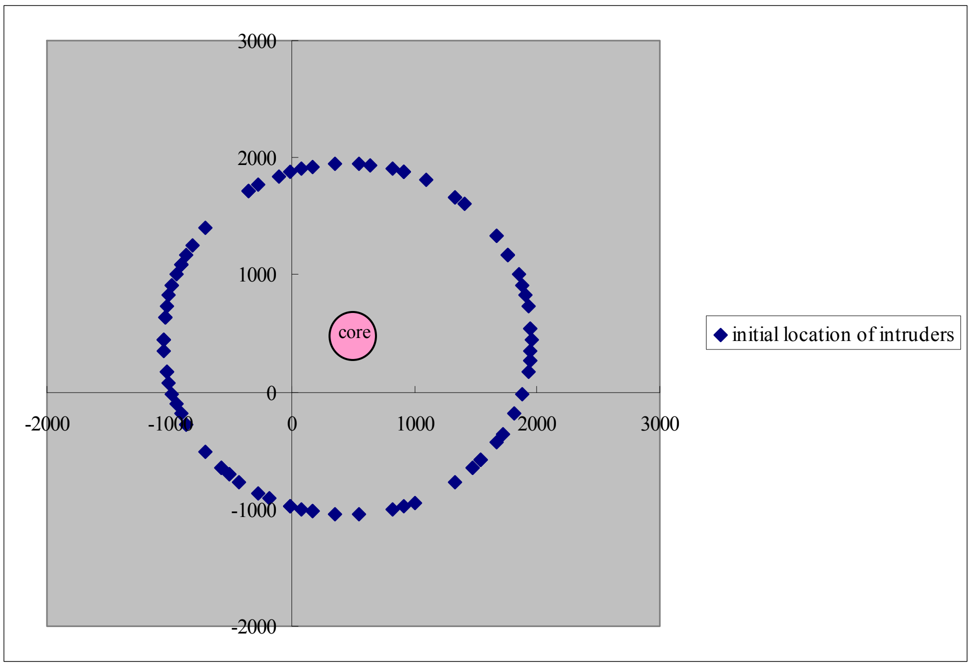


Figure 4.15. The initial positions of intruders given ranges of intrusive angle = 2π. (non-airborne intruders)

4. Airborne intruders

We use special airborne intruder to make intrusive behavior more general. The airborne rate is the ratio of number of airborne intruders to number of all intruders. The initial angle, radius, and location of airborne intruder are as follows:

$$\text{angle of initial location } (\theta) = \text{randomize}(0 \sim 1) * 2\pi$$
$$\text{initial location } r_{\text{airborne}} = (\text{randomize}(0 \sim \text{r}) \% \text{ r}) + 1$$
$$\text{initial location } x = x_{core} + r_{\text{airborne}} \cos\theta$$
$$\text{initial location } y = y_{core} + r_{\text{airborne}} \sin\theta$$

where (x, y) and (x_{core}, y_{core}) are the initial x and y coordinates of the airborne intruders, the x_{core} and y_{core} coordinates of the core position, r is distance between core and initial position of non-airborne intruder, and r_{airborne} is distance between core and initial position of airborne intruder. The airborne intruder ratio is the number of airborne intruders to the total number of intruders. Figure 4.16 shows the initial position of intruders.

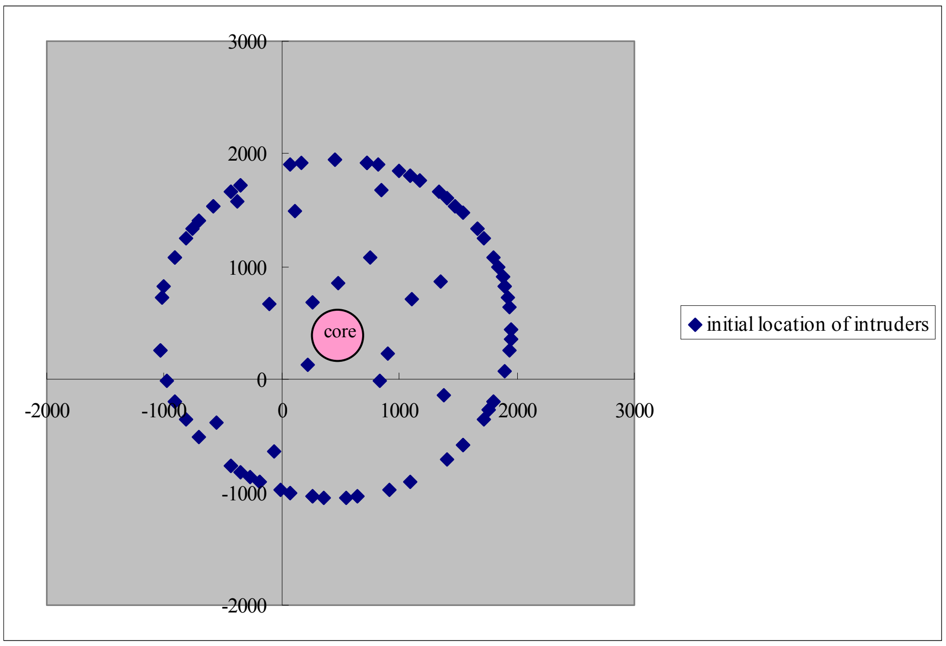


Figure 4.16. The initial positions of intruders include both airborne and non-airborne intruders.

The definitions of false positives (also called false alarm) and false negatives (also called miss) are illustrated as follows [76][77]:

Definition 4.10 False positive: the situation that alarm is raised without intrusion.

Definition 4.11 False negative: the situation that intrusion occurs without alarm.

4.2.3 Mathematical Model

The notations used to model the problem are listed as follows.

Table 4.8. Notation of the controlled parameters in layered defense strategy problem.

Controlled parameters	
Notation	Description
M_r	The total evaluation frequency for all intruder categories in round r.
η	False positive rate.
τ	False negative rate.

Table 4.9. Notation of the given parameters in layered defense strategy problem.

Given parameters	
Notation	Description
K	The total intruder categories.
T_{kr}	Total evaluation frequency of each intruder type in round r (where $k\in K$, $r\in R$).
F	All possible defense strategies.
$\vec{I}_k$	The strategies of an intruder, comprising his motion and intrusive angle.
$G_{kjr}(\vec{F},\vec{I}_k)$	1 if intruder j of the k^{th} intruder category has alarm raised under $\vec{F}$ defense strategies and $\vec{I}_k$ intruder strategies in round r without false positive and false negative, and 0 otherwise (where $k\in K$).
S	The set of all sensor nodes.
C_s	The initial energy level of sensor node s.
E_m	The energy consumption for sensor nodes to sense data.
R	The upper bound number of rounds.
D	The defense rate.
L	The distance of early warning.
W	The early warning rate.
C	Core field: $x_c^2 + y_c^2 \le h^2$, (x_c, y_c) is coordinate of core and h is radius of core.
N	The set of candidate location (x, y) if intruder be detected.

Table 4.10.Notation of the decision variables in layered defense strategy problem.

Decision Variables	
Notation	Description
π_{sr}	1 if sensor *s* is awake in the round *r*; and 0 otherwise.
z_r	1 if satisfy total defense rate and early warning rate in the round *r*, and 0 otherwise.
$\vec{F}$	The strategies of defender that sensor *s* is awake in the round *r*.
$u_{(x,y)}^{kjr}$	1 if the intruder *j* of the k^{th} intruder category that Euclidean distance between location (*x,y*) and *core* greater than or equal to *L* in round *r;* and 0 otherwise.

Problem (IP):

$$\max \sum_{\forall r \in R} z_r \qquad \text{(IP)}$$

subject to:

The defense rate constraint

$$z_r - \left(\frac{\sum_{k=1}^{K}\sum_{j=1}^{T_{kr}} G_{kjr}(\vec{F}, \vec{I_k})}{M_r} - D\right) \le 1 \qquad \forall r \in R \qquad (1)$$

The early warning rate constraints

$$u_{(x,y)}^{kjr} - \frac{\sqrt{x^2+y^2}-L}{\sqrt{x^2+y^2}+L} \le 1 \qquad \forall k \in K,\ j \in T_k,\ r \in R, (x,y) \in N \qquad (2)$$

$$z_r - \left(\frac{\sum_{k=1}^{K}\sum_{j=1}^{T_{kr}} b_{(x,y)}^{kjr}}{M_r} - W\right) \le 1 \qquad \forall r \in R, (x,y) \in N \qquad (3)$$

The battery capacity constraints

$$\sum_{r \in R} (\pi_{sr} E_m) \le C_s \qquad \forall s \in S \qquad (4)$$

The all possible defense strategies constraints

$$\vec{F} \in F \qquad (5)$$

The total evaluation frequency constraints

$$\sum_{k=1}^{K} T_{kr} = M_r \qquad \forall r \in R \qquad (6)$$

The integer constraints

$$\pi_{sr} = 0 \text{ or } 1 \qquad \forall s \in S,\ r \in R \qquad (7)$$

$$z_r = 0 \text{ or } 1 \qquad \forall r \in R \qquad (8)$$

$$u_{(x,y)}^{kjr} = 0 \text{ or } 1 \qquad \forall k \in K,\ j \in T_k,\ r \in R,\ (x,y) \in N\ . \qquad (9)$$

The objective function is to maximize the system lifetime of the given sensor network. The lifetime is defined as the total number of rounds.

Constraint (1): If defense rate constraint is satisfied then set z_r=1.
Constraints (2)-(3): The early warning rate constraints. If early warning rate constraints is satisfied then set z_r=1.
Constraint (4): For each sensor node s, the total sensing consumption can not exceed its initial energy level.
Constraint (5): The all possible defense strategies constraints.
Constraint (6): The total evaluation frequency constraints
Constraints (7)-(9): The integer constraints for decision variables π_{sr}, z_r, and $u_{(x,y)}^{kjr}$.

4.2.4 Solution Procedure

In this section, we propose a non-layered defense strategies algorithm to solve the problem. The algorithm includes three phases. First, the "initial solution phase", we propose a heuristic algorithm for initial defense policy. Second, the "objective function evaluation phase", we propose efficient and effective simulations to evaluate the effectiveness of the current defense policy. Third, the "add-and-drop phase", we use an add-and-drop algorithm to improve and satisfy the defender strategies.

The parameters and decision variables used to model non-layered defense algorithms in this section are listed in Table 4.11.

Table 4.11. The parameters and decision variables used in algorithms of non-layered defense problem.

Notation	Description
max_k	The upper bound of system lifetime.
no_improve_ub	The upper bound of no improving counter.
$counter_{no\_improve}$	No improve counter.
L	The distance of early warning.
D	The defense rate.
W	The early warning rate.
max_a_d (*X*)	The upper bound of times of add-and-drop.
M	The total evaluation frequency for all intruder categories in each round.
sn	The number of sensor nodes.
cp	The number of check points.
gap	The controlled parameter which is tolerant degree of defense rate and early warning rate.
s_no	The number of wake up sensor nodes for full coverage check points.
cs[*s*]	The initial energy level of sensor node *s*.
es[*s*]	The energy consumption for aware sensor node *s* to sense data in each round.
round	The system lifetime.
coverage_rate	The number of covered check points divided by the total number of check points.
c_bsa[*a*]	The number of covered times in check point *a*.
count[*s*]	The number of covered check points by waked sensor *s*.
c_s[*s*]	The number of covered check points under sensing range of sensor *s*.
air_yn	The controlled parameter which is 1 if monitoring region has airborne intruders, and 0 otherwise.
airborne_rate	The controlled parameter which is ratio of airborne intruders.
o_l	The initial location of intruder.
o_r	The initial distance between core and non-airborne intruder.
intrusion_theta	The initial angle between core and location of initial intruder.
max_s_s (*Y*)	The upper bound of step size.
s_n	The speed of the intruder at time interval *n*.

d_n	The direction of the intruder at time interval n.
(x_n, y_n)	The coordinate of intruder at time interval n.
(x_{now}, y_{now})	The coordinate of intruder at now.
(x_{core}, y_{core})	The coordinate of core.
$(x_{intrusion}, y_{inyrusion})$	The coordinate of initial intruder l.
$count_d$	The number of detected intruders in each round.
$count_l$	The number of satisfying distance of leader time in each round.
t_energy	The sum of all sensor energy.
threshold_e	The threshold of total remaining energy.
$bsa[s][a]$	The indicator function is 1 if the check point a is in the sensing range of the sensor node s, and 0 otherwise.
$p[s][r]$	The decision variable is 1 if sensor s is awake in the round r, and 0 otherwise.
sat_d$[r]$	The decision variable is 1 if round r satisfies defense rate and early warning rate, and 0 otherwise.
$cover[a][r]$	The decision variable is 1 if check point a at least is covered by one awake sensor in the round r, and 0 otherwise.

4.2.4.1 Non-Layered Defense Algorithm

We present a non-layered defense algorithm (NLDA) to solve the problem. For solving the original problem near-optimally, we use the *sat_d*$[r]$ to check defense rate and early warning rate in the round r. The decision variable is 1 if defense rate and early warning rate are satisfied, and 0 otherwise. Then, in each round, we first use set of sensor node to cover subset of check point in initial solution and then awake sensor s in the round r. Objective function evaluation is to check whether to satisfy defense rate and early warning rate or not. We use the add-and-drop phase to improve the objective function in each round. The procedure of non-layered defense algorithm is shown in Figure 4.17.

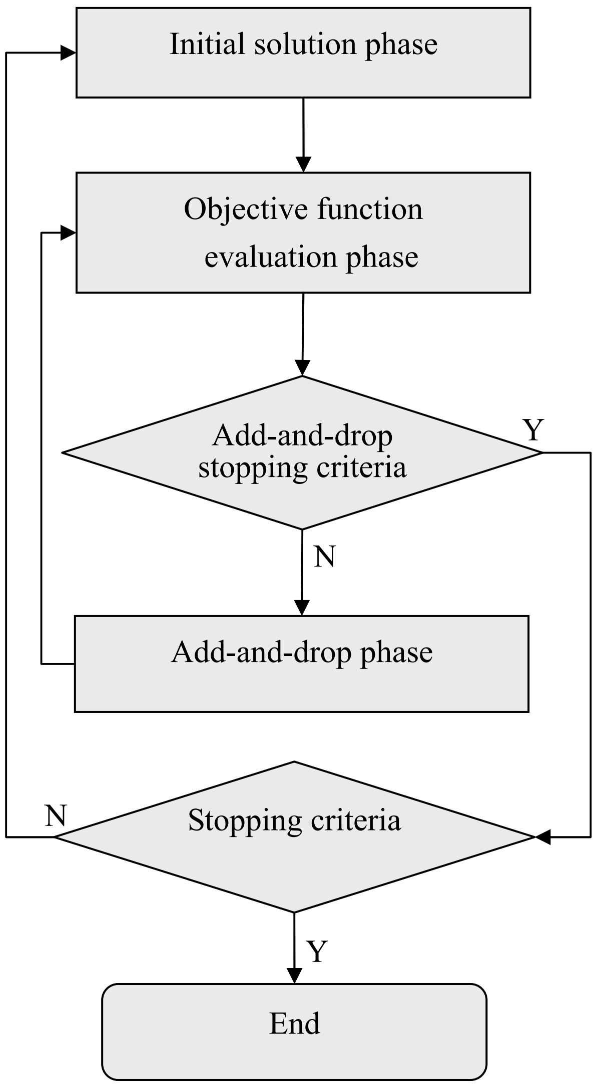

Initial solution phase
Objective function evaluation phase
Add-and-drop stopping criteria
Y
N
Add-and-drop phase
N
Stopping criteria
Y
End

Figure 4.17. The procedure of non-layered defense algorithm.

The non-layered defense algorithm is listed in Figure 4.18.

Algorithm Non-layered Defense

Input: Coordinate of check points and sensor nodes, and sensing radius of sensor nodes

Output: The defense strategies of defenders ($\overline{F}$)

```
1: begin
2:   for $r$=1 to $max\_k$ do
3:   begin
4:        initial solution phase();                    /* phase 1 */
5:        for $add\_drop$ =1 to $max\_a\_d$ do
```

```
 6:         begin
 7:              objective function evaluation phase();   /* phase 2 */
 8:              add-and-drop phase();                    /* phase 3 */
 9:         end
10:         if (($t\_energy<threshold\_e$) or ($counter_{no\_improve}=no\_improve\_ub$))
11:              then break;
12:   end
13: end
```

Figure 4.18. The non-layered defense algorithm.

1. *Initial solution phase*

To solve the original problem efficiently, we use the concept of "cover" to determine whether sensor *s* is awake or not in the round *r*. The "cover" is 1 if the check point *a* is in the sensing range of the sensor node *s*, and 0 otherwise.

The concept of check points is introduced in Section 3.1.1, which can check coverage rate. The coverage rate is the number of check points covered by awake sensors divided by the total number of checks points. Besides, check points can save energy consumption because they check the coverage rate more efficiently for arbitrary topology.

We first find sensor *s* to cover check point *a*, and then sensor *s* is awaken by this phase in the round *r*, and repeat the assignment process until this phase satisfies the coverage rate. In addition, we must turn off redundant awake sensor nodes in the phase.

The initial solution algorithm is listed in Figure 4.19.

Algorithm Initial solution

Input: The round *r*, the initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round, and coverage rate
Output: The initial solution (*p*[*s*][*r*])

```
 1:   begin
 2:         for $s$=1 to $sn$ do
 3:              $p[s][r]$=0;
 4:         while (coverage_rate is not satisfied) do
 5:         begin
 6:          for s=1 to sn do
 7:              for a=1 to cp do
 8:                  if ((bsa[s][a]=1) and (cs[s]>=es[s]) and (cover[a][r]=0))
                     then
 9:                  begin
10:                      p[s][r]=1;
11:                      cs[s]=cs[s]-es[s];
12:                      for k=1 to cp do
13:                          if (bsa[s][a]=1)
14:                              then c_bsa[a]=c_bsa[a]+1;
15:                          if ((bsa[s][a]=1) and ( cover[a][r]=0))
16:                              then cover[a][r]=1;
17:                  end
18:      end
19:      for s=1 to sn do                        /* delete redundant sensor nodes */
20:      begin
21:          for a=1 to cp do
22:          begin
23:              if ((p[s][r]=1) and (bsa[s][a]=1) and (c_bsa[a]>=2))
24:                  then count[s]=count[s]+1;
25:              if (count[s]=c_s[s])
26:              begin
27:                  cs[s]=cs[s]+es[s];  /* recovery energy */
28:                  p[s][r]=0;
29:                  for a=1 to cp do
30:                      if (bsa[s][a]=1)
31:                          then c_bsa[a]=c_bsa[a]-1;
32:              end
33:          end
34:      end
35:      for s=1 to sn do
36:          if (p[s][r]=1)
37:              then s_no=s_no+1;
38:  end
```

Figure 4.19. The initial solution algorithm.

In the algorithm, from steps 2-3 are used to set initial value, steps 4-18 are used to decide whether sensor *s* is awaken in the round *r*. Steps 19-34 are used to delete redundant awaked sensor nodes. An example of deleting redundant awaked sensor node as illustrated in Figure 3.12 of Section 3.1. Steps 35-37 are used to calculate number of wake up sensor nodes in initial phase.

2. *Objective function evaluation phase*

Since the scenario and environment are dynamic, it is difficult to solve the problem only by mathematical programming. The proposed evaluation process enables us to better describe the behavior of different intruders. In each intruder category, there is some randomness in the behavior of intruders, even intruders are the same type.

The number of total intruders is set to the same value as *M*, which is determined by experiment. First, we select an initial value, for example, 10000. Then, if the diagram shows a stable trend, it implies that the value of *M* is ideal. On the other hand, if the diagram shows an unstable result, it shows that *M* is too small; therefore, we set *M* to a larger number to run the test experiment. Figure 4.20 shows the experiment results, and *M* is set to 2000 intruders.

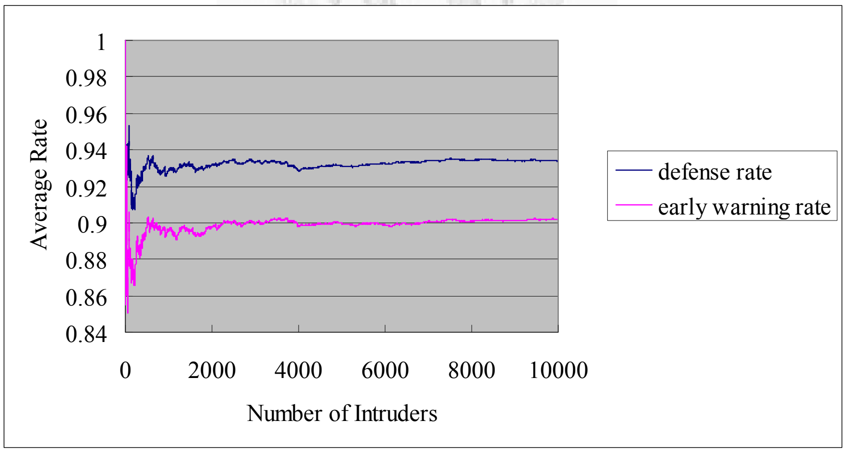


Figure 4.20. The experiment results: the number of total intruders. (D = 0.9, W = 0.9)

After deciding the value of *M* and initial solution configuration, we apply the evaluation process to simulate behavior of intruders. Based on this, we run the evaluation *M* times with different categories of intruders to attack the core field. Then, we divide this frequency by *M* to obtain the average defense rate and average early warning rate. We take this result as the benchmark to evaluate the performance of each round.

An objective function evaluation algorithm is listed in Figure 4.21.

Algorithm Objective Function Evaluation

```
Input: The round $r$, initial solution, and intruder behavior
Output: Defense rate and early warning rate
 1: begin
 2:   for intruder=1 to $M$ do                    /* simulation */
 3:   begin
 4:       if (((intruder%(1/airborne_rate))=0) and (air_yn=1)) then
 5:           o_l=(randomize(0~1)%o_r)+1;
 6:       else
 7:           o_l=o_r;
 8:       calculate $\alpha$ (using modified logistic function) and
                  intrusion_theta (using (randomize(0~1)*2π)+π)
 9:       for $n$=1 to max_s_s do
10:       begin
11:           calculate $s_n$, $d_n$, $x_n$, $y_n$,and deviant_theta
12:           if deviant_theta> deviant_range then
13:           begin
14:               $x_{intrusion}=x_n$;
15:               $y_{intrusion}=y_n$;
16:               $theta_n=\arctan((y_n - y_{core})/(x_n - x_{core})) + \pi$;
17:               $x_{n-1}=x_{intrusion}$;
18:               $y_{n-1}=y_{intrusion}$;
19:           end
20:           for $s$=1 to $sn$ do
21:           begin
22:               if $\sqrt{(x_s - x_n)^2 + (y_s - y_n)^2} \le r_s$ then
23:               begin
24:                   $count_d= count_d+1$;
25:                   if $\sqrt{(x_{core} - x_n)^2 + (y_{core} - y_n)^2} \ge L$ then
26:                       $count_l= count_l+1$;
27:                   break;
28:               end
29:           end
```

```
30:             if $\sqrt{(x_s - x_n)^2 + (y_s - y_n)^2} \le r_s$
31:                 then break;
32:             if $\sqrt{(x_{core} - x_n)^2 + (y_{core} - y_n)^2} \le r_{core}$
33:                 then attack success and break;
34:         end
35:     end
36:     $defense\_rate = count_d/M$;
37:     $early\_warning\_rate = count_l/M$;
38:     if (satisfy defense_rate and early_warning_rate) then
39:     begin
40:             $sat\_d[r]=1$;
41:             $round=round+1$;
42:             deleting redundant awaked nodes and break;
43:     end
44: end
```

Figure 4.21. The objective function evaluation algorithm.

In the algorithm, from steps 4-7 deal with airborne intruders, and steps 8-19 decide behavior of intruders. Steps 20-34 are used to check whether intruders are detected and distance of lead time is satisfied. Steps 36-43 are used to check whether defense rate and early warning rate are satisfied.

3. *add-and-drop phase*

In this phase, we improve the quality of the solution by removing wake up sensor nodes and adding sleep sensor nodes to wake up sensor nodes. Then, we run the evaluation another *M* times using the adjusted defense parameters and obtain the average defense rate and average early warning rate. Finally, we check whether one of the stopping criteria is satisfied. If it is, we terminate the procedure.

The stopping criteria can be divided into two concepts. The first is the total remaining energy, which we set to be no more than *threshold_e*. The value of threshold is decided by ratio of total sensor energy. If total remaining energy is below the *threshold_e*, then terminate the procedure. The second is that when the number of iteration reaches the *no_improve_ub*, then terminate the procedure.

An add-and-drop algorithm is listed in Figure 4.22.

Algorithm Add-and-drop

Input: Defense rate, early warning rate, and *s_no*

Output: Which sensor *s* is awaken in round *r* (*p*[*s*][*r*])

```
 1: begin
 2:  $k=\|\lfloor (defense\_rate-D)\times s\_no \rfloor\|$;
 3:  if (satisfy $D$ and $W$)
 4:      then
 5:      begin
 6:          if (add_flag=1)
 7:              then k= k/2;
 8:          drop_flag =1;
 9:          add_flag =0;
10:          for drop=1 to $\lfloor k \rfloor$ do
11:          begin
12:              drop the sensors in high priority whose radii
                 have not covered any intruder in previous simulation;
13:              cs[s]=cs[s]+es[s];          /* energy recovery */
14:              p[s][r]=0;
15:          end
16:      end
17:      else
18:      begin
19:          if (drop_flag =1)
20:              then k= k/2;
21:          add_flag =1;
22:          drop_flag =0;
23:          for add=1 to $\lceil k \rceil$ do
24:          begin
25:              add the sensors whose radii have covered the intruders and
                 keep sleeping in previous simulation;
26:              p[s][r]=1;
27:              cs[s]=cs[s]-es[s];
28:              for k=1 to cp do
29:                  if (bsa[s][a]=1)
30:                      then c_bsa[a]=c_bsa[a]+1;
31:                  if ((bsa[s][a]=1) and ( cover[a][r]=0))
32:                      then cover[a][r]=1;
33:          end
```

```
34:         end
35:         for s=1 to sn do                        /* delete redundant sensor nodes */
36:         begin
37:             for a=1 to cp do
38:             begin
39:                 if ((p[s][r]=1) and (bsa[s][a]=1) and (c_bsa[a]>=2))
40:                     then count[s]=count[s]+1;
41:                 if (count[s]=c_s[s])
42:                 begin
43:                     cs[s]=cs[s]+es[s];  /* energy recovery */
44:                     p[s][r]=0;
45:                     for a=1 to cp do
46:                         if (bsa[s][a]=1)
47:                             then c_bsa[a]=c_bsa[a]-1;
48:                 end
49:             end
50:         end
51: end
```

Figure 4.22. The add-and-drop algorithm.

In the algorithm, from steps 5-16 are to drop redundant awaked sensors. Steps 18-34 are to wake up sensors to satisfy defense rate and early warning rate. Steps 35-50 are used to delete redundant awaked sensor nodes. An example of deleting redundant awaked sensor node as illustrated in Figure 3.12 of Section 3.1.

The computational complexity of the non-layered defense algorithm in initial solution is $O(|S||A|^2)$, where $|S|$ is number of sensor nodes and $|A|$ is number of check points. In objective function evaluation phase is $O(|M||Y||S|)$, where $|M|$ is the total evaluation frequency for all intruder categories in each round and $|Y|$ is the upper bound of step size. In add-and-drop phase is $O(|S||A|)$. In non-layered defense algorithm, from steps 6-9 run $O|R||X|$ times, where $|R|$ is the upper bound of number of rounds and $|X|$ is the upper bound of times of add-and-drop. Therefore, the computational complexity is $O(|R||X||M||Y||S|)$. Hence, the computational complexity of the non-layered defense algorithm should be $O(|R||X||M||Y||S|)$.

4.2.4.2 Simple algorithm

We first find sensor *s* to cover check point *a*, and then sensor *s* is awaken by this phase in the round *r*, and repeat the assignment process until fully cover all check points (*coverage_rate* = 1). A simple algorithm is listed in Figure 4.23.

Algorithm Simple

Input: The initial energy level of sensor node *s*, the energy consumption for aware sensor node *s* to sense data in each round

Output: Which sensor *s* is awaken in round *r* (*p*[*s*][*r*])

```
 1: begin
 2:     for r=1 to max_k do
 3:     begin
 4:         for s=1 to sn do
 5:             p[s][r]=0;
 6:         while (coverage_rate is not 1) do
 7:         begin
 8:             for s=1 to sn do
 9:                 for a=1 to cp do
10:                     if ((bsa[s][a]=1) and (cs[s]>=es[s]) and (cover[a][r]=0))
                        then
11:                     begin
12:                         p[s][r]=1;
13:                         cs[s]=cs[s]-es[s];
14:                         for k=1 to cp do
15:                             if ((bsa[s][a]=1) and ( cover[a][r]=0))
16:                                 then cover[a][r]=1;
17:                     end
18:         end
19:         objective function evaluation ();
20:         if (defense_rate and early_warning_rate are not satisfied) then
21:         begin
22:             cs[s]=cs[s]+es[s];            /* recovery energy */
23:             p[s][r]=0;
24:         end
25:     end
26: end
```

Figure 4.23. The simple algorithm of non-layered defense.

In the algorithm, from steps 4-5 are used to set initial value, steps 6-18 are used to decide whether sensor *s* is awaken in the round *r*. Step 19 is used to check whether defense rate and early warning rate are satisfied. The procedure is same to objective function evaluation algorithm. Steps 20-24 are used to recovery energy for sensor *s* in the round *r* which defense rate and early warning rate are not satisfied.

The computational complexity of the simple algorithm of non-layered defense at steps 4-5 is $O(|S|)$, where $|S|$ is number of sensor nodes. From steps 8-17 is $O(|S||A|)$, where $|A|$ is number of check points. In objective function evaluation phase is $O(|M||Y||S|)$, where $|M|$ is the total evaluation frequency for all intruder categories in each round and $|Y|$ is the upper bound of step size. Above steps from steps 2-5 run $|R|$ times, where $|R|$ is the upper bound of number of rounds. Therefore, the computational complexity is $O(|R||M||Y||S|)$. Hence, the computational complexity of the simple algorithm of non-layered defense should be $O(|R||M||Y||S|)$.

4.2.5 Computational Results

We conduct an experiment to evaluate the performance of the proposed algorithm. The performance is assessed in terms of total number of rounds.

4.2.5.1 Experiment Environment

The proposed algorithm is coded in C under a Dev C++ 4.9.9.2 development environment. All the experiments are performed on a Core 2 Duo 2.2GHz CPU running Microsoft Windows Vista. The algorithm is tested on a 2D monitoring region. We distribute 400 and 1600 sensor nodes and 100 and 400 check points respectively in 2D monitoring region. The radius of different sensors types s_a and s_b is 100 and 200. The energy consumption of aware different sensor types s_a and s_b is 1 and 4 in each round.

Before the evaluation process, we need to determine the value of *M*. Therefore, we run a number of experiments to find the proper value for our scenario. The diagram in Figure 4.20 shows a stable trend in $M = 2000$. Hence, we set M as 2000. The important parameters and ratio of airborne intruders are listed in Table 4.12 and Table 4.13.

Table 4.12. The parameters of non-layered defense.

Parameters	Value
Battery capacity levels	5
Deviant range	$\pi/6$
Lambda (λ)	6
Number of sensor node (*sn*)	400 and 1600
Number of check point (*cp*)	100 and 400
Distance of early warning (L)	300 and 600
Monitoring Region (m^2)	1000×1000 and 2000×2000
False positive rate	0.02
False negative rate	0.05
Total number of intruders in one round (M)	2,000

Table 4.13. Ratio of airborne intruders.

Types of Intruder	Ratio
Airborne Intruders	20%
Non-airborne Intruders	80%

4.2.5.2 Experiment results

Figure 4.24 shows an example of non-layered defense with non-airborne intruders. And Figure 4.25 shows an example of non-layered defense with airborne intruders.

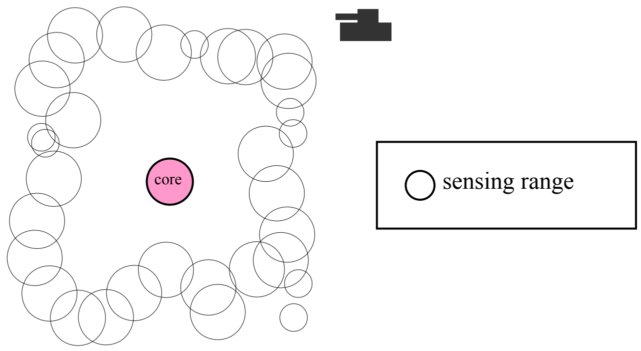


Figure 4.24. An example of non-layered defense with non-airborne intruders.

($D = 1.0$, $W = 0.9$)

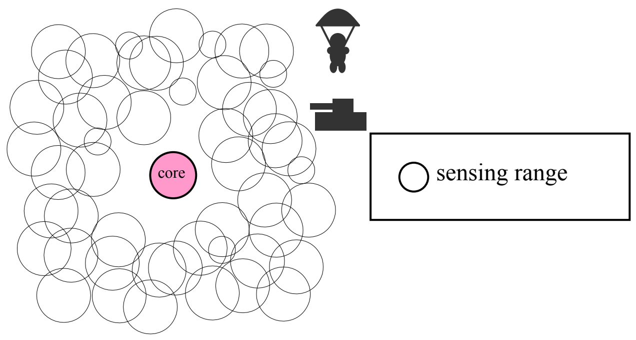


Figure 4.25. An example of non-layered defense with airborne intruders. (D = 1.0, W = 0.9)

Table 4.14 shows the maximum total number of rounds calculated by different scenarios. Figure 4.26 shows a comparison of the number of rounds in different nodes and different scenarios. Figure 4.27 shows a comparison of the number of rounds in airborne intruders. Figure 4.28 shows a comparison of the number of rounds in different nodes and defense rate given W=0.8. Figure 4.29 shows a comparison of the number of rounds in different nodes and Early warning rate given D=1.0. Table 4.15 shows the evaluation of the round with different λ value. Table 4.16 shows an evaluation of the round with false positives and false negative. Figure 4.30 shows a comparison of the number of rounds with false positives and false negative. Figure 4.31 shows an example of the false positive nodes. Figure 4.32 shows an example of the false negative node. Figure 4.33 shows a relationship between false negative rate and early warning distance.

Table 4.14. Evaluation of the round with different number of nodes and different scenarios without false positives and false negative.

Number of nodes (check points, sensor nodes)	Monitoring region (m^2)	Airborne intruders	D=0.8 W=0.8	D=0.9 W=0.8	D=0.9 W=0.9	D=1.0 W=0.8	D=1.0 W=0.9	D=1.0 W=0.99
(400, 1600)	2000×2000	no	87	80	71	63	57	47
(400, 1600)	2000×2000	yes	60	48	38	41	36	0
(100, 400)	1000×1000	no	51	50	41	38	37	30
(100, 400)	1000×1000	yes	43	42	27	31	28	0

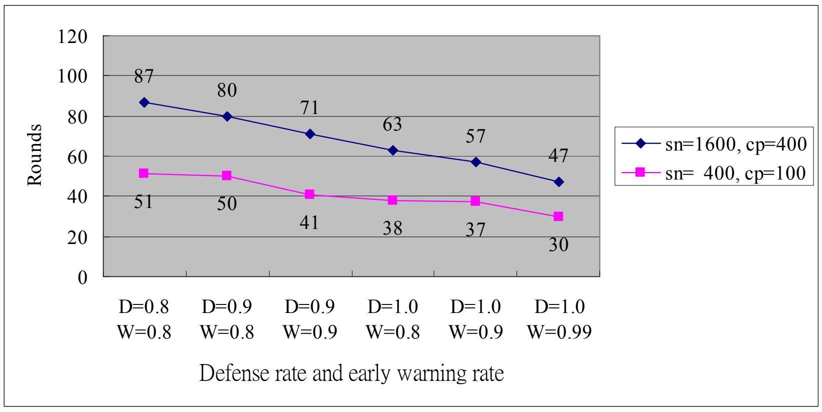


Figure 4.26. A comparison of the number of rounds in different nodes and different scenarios without false positives and false negative.

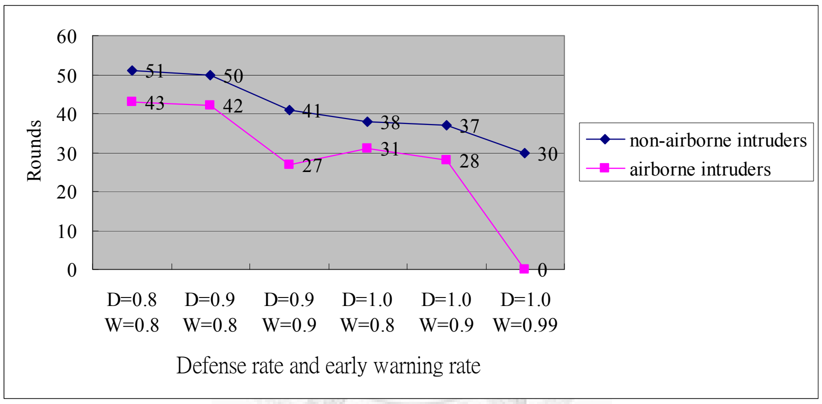


Figure 4.27. A comparison of the number of rounds in airborne intruders without false positives and false negative. (*sn* = 400, *cp* = 100, and *airborne ratio* = 0.2)

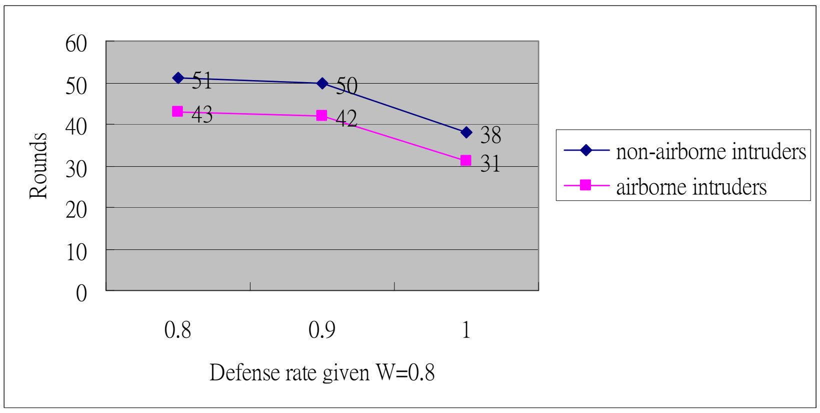


Figure 4.28. A relationship between number of rounds and defense rate without false positives and false negative given *W* = 0.8.

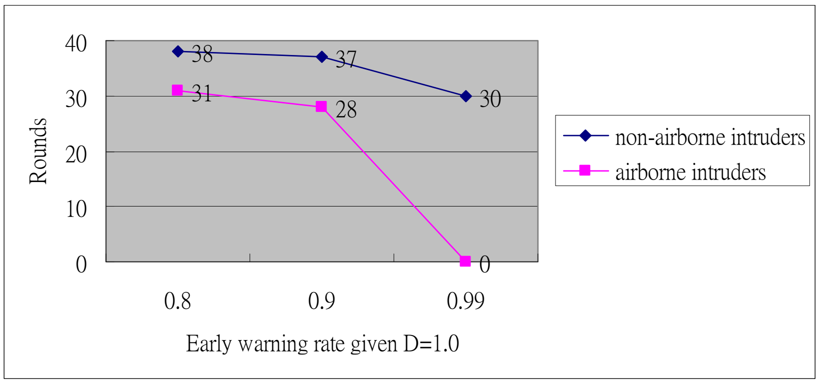


Figure 4.29. A relationship between number of rounds and early warning rate without false positives and false negative given $D = 1.0$.

Table 4.15. Evaluation of the round with different λ value without false positives and false negative.

Number of nodes (check points, sensor nodes)	Monitoring region (m^2)	λ	D=0.8 W=0.8	D=0.9 W=0.8	D=0.9 W=0.9	D=1.0 W=0.8	D=1.0 W=0.9	D=1.0 W=0.99
(100, 400)	1000×1000	6	51	50	41	38	37	30
(100, 400)	1000×1000	100	51	50	40	38	37	26

Table 4.16. Evaluation of the round with false positives and false negative.

Number of nodes (check points, sensor nodes)	False positives rate (FP) and false negative rate (FN)	Airborne intruders	D=0.8 W=0.8	D=0.9 W=0.8	D=0.9 W=0.9	D=1.0 W=0.8	D=1.0 W=0.9	D=1.0 W=0.99
(100, 400)	FP=0, FN=0	no	51	50	41	38	37	30
(100, 400)	FP=0.02, FN=0	no	47	49	39	37	35	36
(100, 400)	FP=0, FN=0.05	no	47	43	40	39	35	27
(100, 400)	FP=0.02, FN=0.05	no	46	44	35	34	34	23

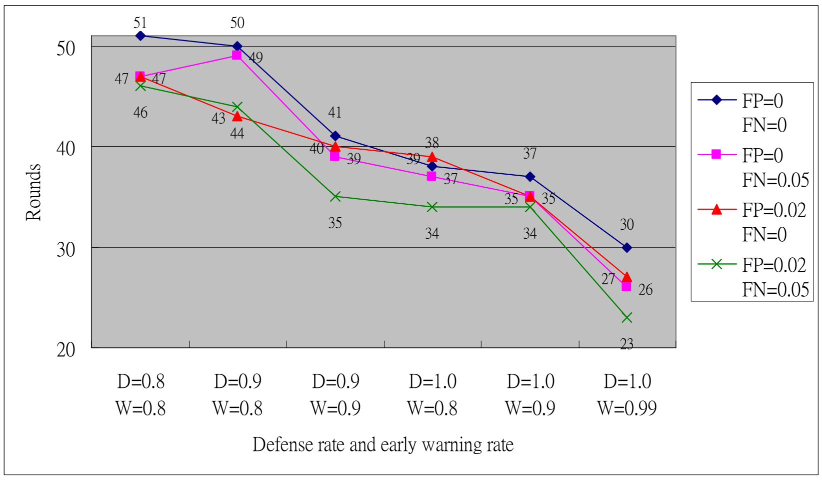


Figure 4.30. A comparison of the number of rounds with false positives and false negative.

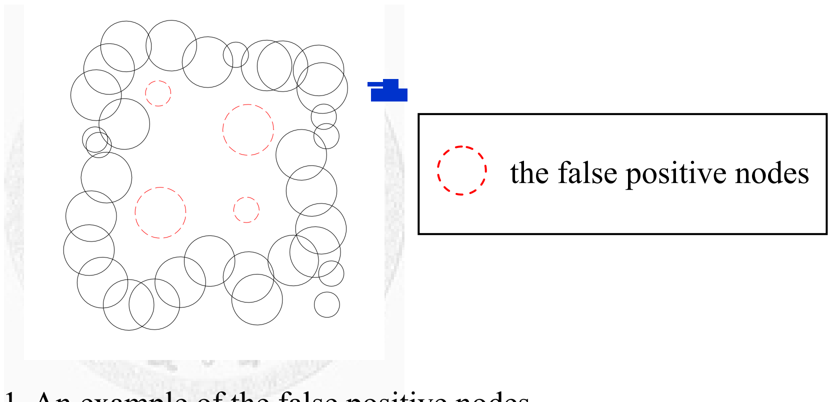


Figure 4.31. An example of the false positive nodes.

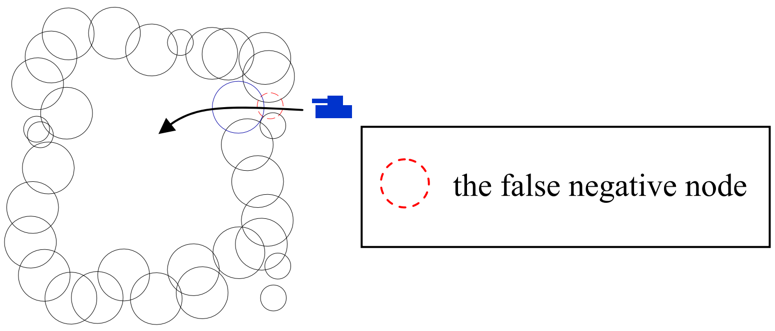


Figure 4.32. An example of the false negative node.

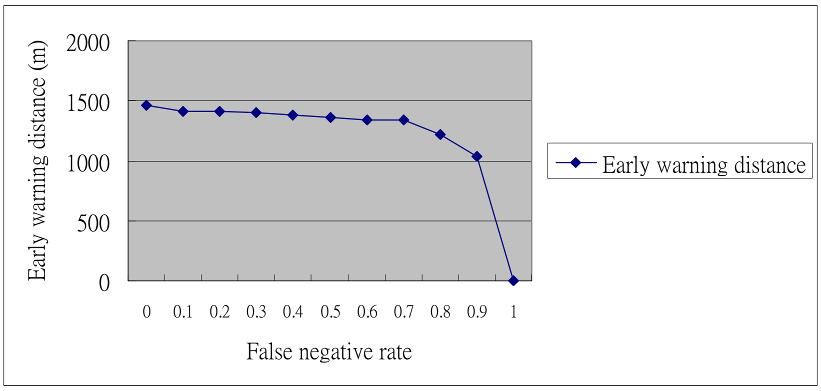


Figure 4.33. A relationship between false negative rate and early warning distance.

4.2.5.3 Discussion

The experiment results show that the large region of lower defense rate has higher rounds than that of small region in same deployment density, as shown in Figure 4.26, because large scale region has larger depth. Therefore, defenders can use lower density of sensors to cover monitoring region. The airborne intruder cases have lower rounds than that of non-airborne intruder cases. In addition, the rounds is 0 in $D = 1$ and $W = 0.99$, because airborne intruders drop randomly in the monitoring region. Therefore, the distance of early warning is not satisfied, as shown in Figure 4.27. The proposed approach can prolong system lifetime by lower defense rate and lower early warning rate, as shown in Figure 4.28 and Figure 4.29.

Without false positive and false negative cases have higher rounds than that with false positive and false negative cases, as shown in Table 4.16 and Figure 4.30. The reason is that false positive case turns on some redundant sensors and false negative case must turn on inner sensor to detect intruder, as shown in Figure 4.31 and Figure 4.32. In addition, system asks sensors two times to reduce false positive probability when alarms are raised. The high false negative rate leads to shorten the early warning distance as shown in Figure 4.33.

Table 4.17 shows the maximum total number of rounds calculated by different algorithms. We can see that the NLDA outperforms the simple algorithm.

Table 4.17. Evaluation of the improvement ratio with simple algorithm without false positives and false negative.

Number of nodes (sensor nodes, check points)	Monitoring Region (m^2)	NLDA (D=1, W=0.99)	Simple Algorithm (D=1, W=0.99)	Improvement Ratio to Simple Algorithm
(1600, 400)	2000×2000	52	22	136%
(400,100)	1000×1000	27	17	59%

The results show that the algorithm is better than the simple algorithm. The proposed NLDA can improve the percentage of energy consumption from 59% to 136%.

4.2.6 Concluding Remarks

This study proposes a non-layered defense algorithm for wireless sensor networks of grouping capabilities. To our best knowledge, the proposed algorithm is truly novel and it has not been yet discussed in previous researches. The study first formulates the problem as combining mathematical programming problem, and then proposes a heuristic-based algorithm for solving the optimization problem.

We find the maximum k groups of sensors for non-layered defense subject to defense rate, early warning rate, battery capacity, intruder behavior, and defender strategies constraints. The mechanism can prolong the system lifetime and provide lead time alarms. A novel three-phase solution procedure, which well combines mathematical programming and simulation techniques, is proposed. Compared with simple algorithm, the proposed NLDA can improve system lifetime since the improvement ratio is from 59% to 136%. Therefore, the experiment results show that the proposed non-layered defense algorithm gets applicability and effectiveness in the non-layered defense for grouping capabilities.

Our main contribution is that we combine mathematical programming with simulations and develop a novel approach to solve the problem with the imperfect knowledge property. This mechanism helps us prolong the system lifetime of non-layered defense in WSNs.

Chapter 5 Object Tracking Algorithms

In this chapter, we propose two algorithms, TOTA and POTA, to support object tracking services. The TOTA is to construct an object tracking tree for object tracking. Such tree-based algorithm can achieve energy-efficient object tracking for given arbitrary topology of sensor networks. The POTA is to construct a prediction-based algorithm for object tracking. Such prediction-based can minimize the number of nodes participating in the tracking activities, minimize out of tracking probability, and maximize the accuracy of object predicted position in the tracking activities. The POTA can prolong the system lifetime.

In this chapter, the tree-based object tracking algorithm is described in Section 5.1 and the prediction-based object tracking algorithm is presented in Section 5.2.

5.1 Tree-based Object Tracking Algorithm

In this section, we propose an energy-efficient tree-based object tracking algorithm (TOTA) in wireless sensor networks. Such sensor network has to be designed to achieve energy-efficient object tracking for given arbitrary topology of sensor networks. We particularly consider the bi-directed moving objects with given frequencies for each pair of sensor nodes and link transmission cost. This problem is formulated as a 0/1 integer-programming problem. A tree-based object tracking algorithm (TOTA) is proposed for solving the optimization problem. Experiment results show that the proposed algorithm gets a near optimization in the energy-efficient object tracking. Furthermore, the algorithm is very efficient and scalable in terms of the running time.

The rest of this section is organized as follows. The overview is described in Section 5.1.1. The problem and mathematical models are described in Sections 5.1.2 and 5.1.3, respectively. In addition, the solution procedure is presented in Section

5.1.4. Furthermore, the computational results are discussed in Section 5.1.5, and conclusions are presented in Section 5.1.6.

5.1.1 Overview

In this section, we formulate the problem as a 0/1 integer-programming problem where the objective function is to minimize the total communication cost subject to routing, tree, and variable-transformation constraints. To fulfill the timing and the quality requirements of the optimal decisions, the Lagrangean relaxation method is used. We use the LR-based heuristic algorithm to solve the problem and obtain a primal feasible solution. In the further experiments, the proposed object tracking algorithm is expected to be efficient and effective in dealing with the complicated optimization problem.

5.1.2 Problem Description

Our approach uses hierarchical object tracking tree to record information about presence of the object and keep this information up to date. Sensor nodes are required to detect and track the moving states of mobile objects. The information about presence of the detected objects is stored at communication nodes and each communication node particularly stores the set of objects that was detected jointly by its descendants. This set is called the detected set. For example, the detected set of a sensor at a leaf node consists of the objects within the detection range of sensor while the detected set of sink node contains all objects presented in the sensor field [10]. We assume that the moving frequencies of the sensor field are not uniformly distributed. For example, the moving frequencies of wild animals are not uniform in a wildlife protective zone, because animals usually move in their customary paths.

Figure 5.1 illustrates a scenario of object tracking. Sensor *u* will detect the object and deliver the object's location information to sink node when object enters the sensor filed, and sensor *v* will only forward the new location information to communication node *c* when object moves from sensor *u* to sensor *v*. This scenario can be performed through the entire sensor field. Finally, sensor *z* will forward the leaving information to sink node when object leaves sensor field from sensor *z*. The problem is solved in planning stage.

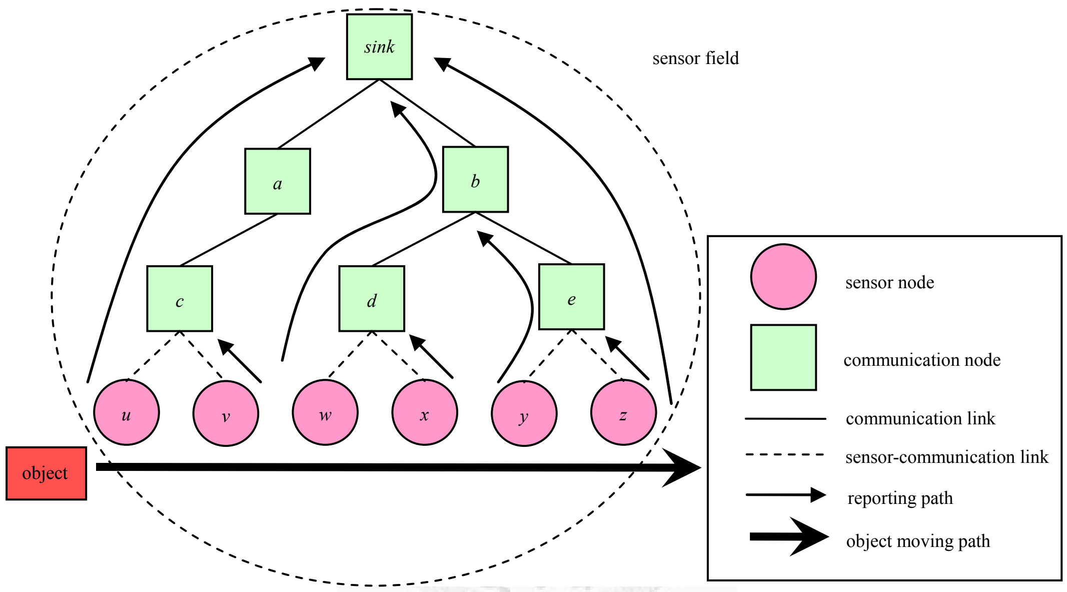


Figure 5.1. An example of object tracking.

The energy-efficient object tracking in WSNs problem is modeled as a graph, $G(V,L)$, where V is a set of communication nodes and sensor nodes randomly deployed in a 2D sensor field, and L is a set of links connect a pair of adjacent communication nodes or between a pair of a sensor node and a communication node.

For example, Figure 5.2 illustrates a 2D sensor field's routing sub-graph with each edge connecting a pair of adjacent communication nodes or between a pair of a sensor node and a communication node. Each weight of link represents link transmission cost. In [64], J. Cartigny, et al. define the energy consumption model of transmitting data which is measured as $r^{\alpha}+c$, where r is Euclidean distance between any two nodes, α is a signal attenuation constant, and c is a positive constant that represents signal processing. Table 5.1 presents power model for the MICAz hardware platform. As the table shows, transmission power and received power are different. To be more generic, we redefine the link transmission cost as the power consumption of transmission power and received power, which is measured as $r^{\alpha} + x + c$, where x is received power.

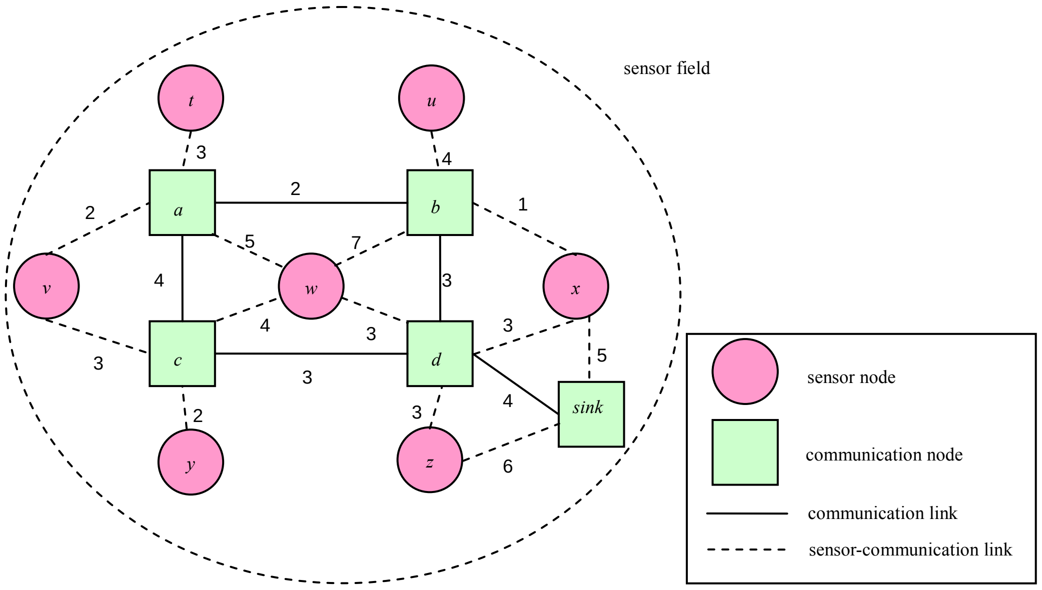


Figure 5.2. An example of 2D routing sub-graph.

The sensor sub-graph in Figure 5.3 illustrates a 2D sensor field with each edge connecting a pair of adjacent sensors. We use (*i*,*j*) to represent the weight of link which is the moving object frequency of a sensor node *i* and a senor node *j*. The link weight of artificial node is the moving frequency of object between sensor field and outside the sensor field.

Table 5.1. Power model of the MICAz.

Mode		Current
Radio	Rx	19.7 mA
	Tx(-10 dBm)	11 mA
	Tx(-5 dBm)	14 mA
	Tx(0 dBm)	17.4 mA

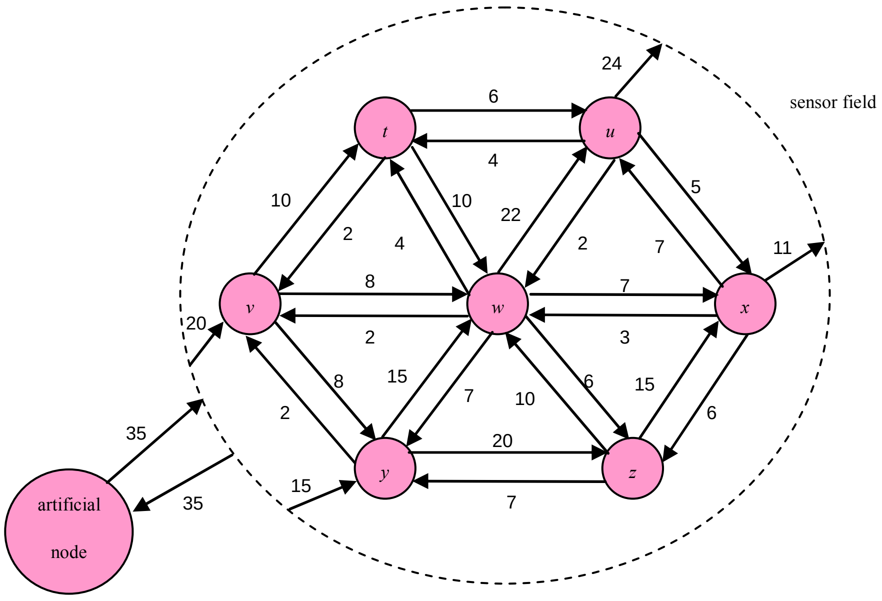


Figure 5.3. An example of 2D sensor sub-graph.

Figure 5.4 illustrates an object tracking tree of 2D sensor field with each edge connecting a pair of adjacent nodes. Each weight of link represents the link transmission cost between a pair of adjacent communication nodes, or between a pair of a sensor node and a communication node. The root is sink node.

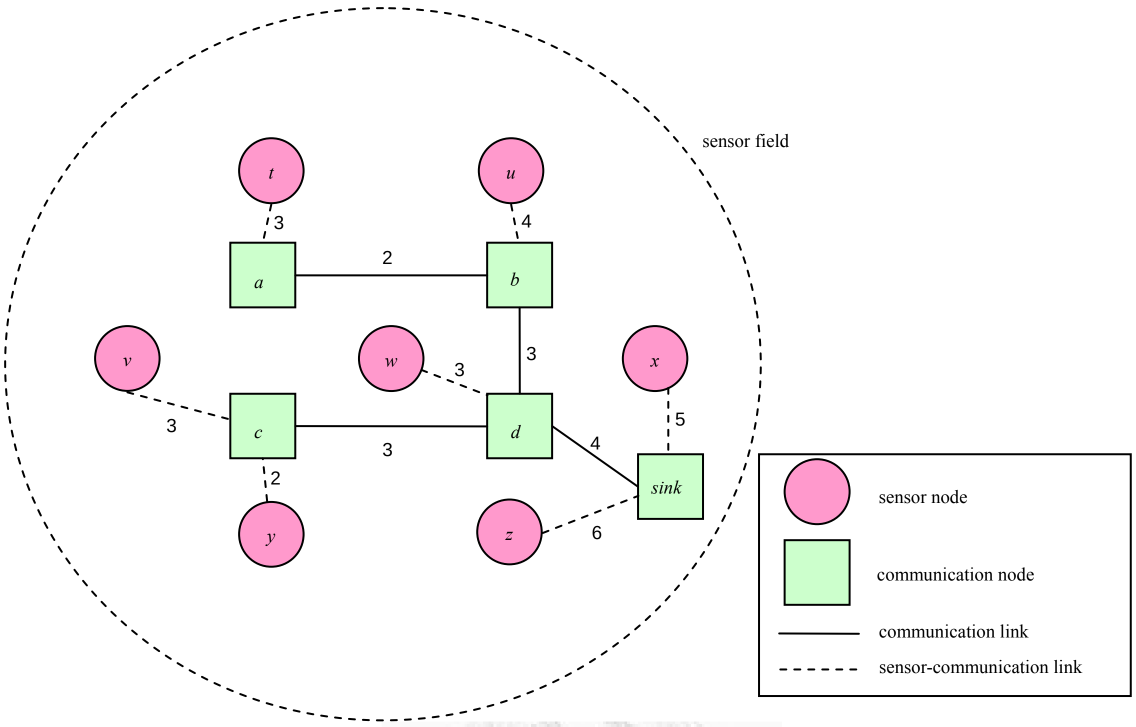


Figure 5.4. An example of 2D object tracking tree.

In this section, we consider a given arbitrary topology of sensor networks, bi-directed moving objects with given frequencies for each pair of sensor nodes, and link transmission cost. The sensor field consists of sensor nodes and communication nodes. We deploy hierarchical network topology architecture. All sensor nodes send data to upper layer communication nodes. Eventually, the sensing information is sent to sink node. We assume that *G* is connected. The location model is a sensor cell model constructed by voronoi diagram. For example, an object moves from sensor *x* to sensor *y* means that the object moves from voronoi cell of sensor *x* to voronoi cell of sensor *y* as shown in Figure 5.5.

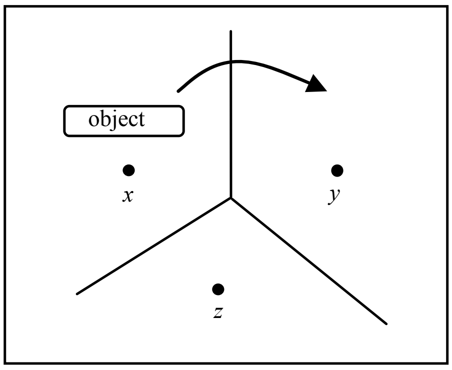


Figure 5.5. An example of an object moves from voronoi cell *x* to voronoi cell *y*.

A good tracking method is characterized by a low total communication cost [10]. Given a sensor graph, we can compute the total communication cost.

The calculating communication cost is different from that of prior studies [10][11][12]. First, we consider the bi-directed moving objects with given frequencies for each pair of sensor nodes because the round-trip traffic cost of each pair of sensor nodes is different. Second, we consider the link transmission cost since each link transmission cost is also different. Figure 5.6 illustrates an example of calculating communication cost. The weight of each solid link represents link transmission cost between a pair of adjacent communication nodes or between a pair of a sensor node and a communication node. The weight of each dash link represents the frequency of moving objects between a pair of adjacent sensors. When an object moves from sensor *x* to sensor *y*, sensor *y* needs to deliver the tracking information upward to the nearest common ancestor *p* via the tree links. We call the tree links as the tracking links [10]. For example, the link between communication node *p* and sensor node *y* is a tracking link.

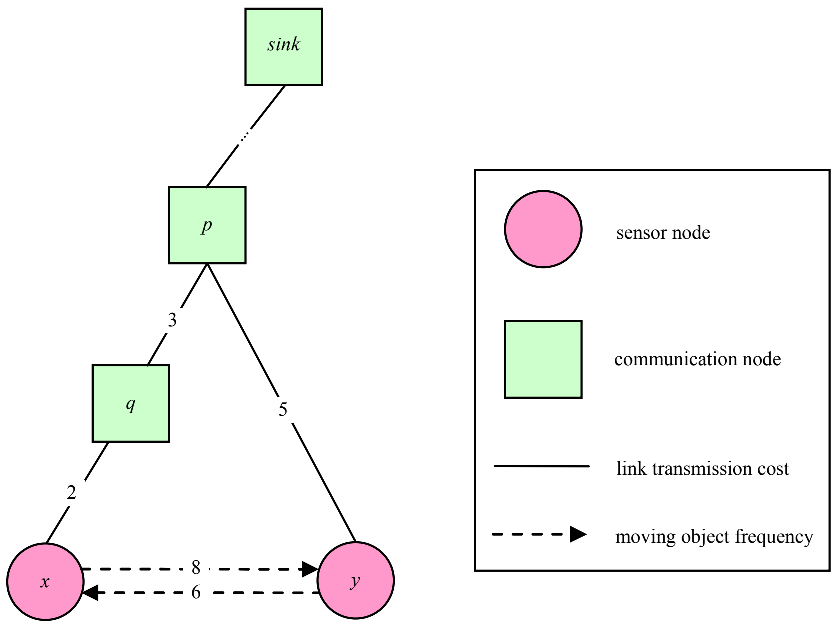


Figure 5.6. An example of calculating communication cost.

We define the communication cost of an object tracking tree *T* as the sum of the individual contributions of all pairs of sensors adjacent in *G*. Since the adjacent tree nodes may be physically in a distance, we define the costs of tree links used in the path to be Euclidean distances. Thus, the communication cost reflects the power consumption degree of required radio.

The communication cost of inside network, define as

$$(G,T)_{inside} = \sum_{(i,j)\in\Lambda} (1-\zeta^{x}_{(i,j)})\zeta^{y}_{(i,j)}\theta_{xy}\omega_{(i,j)} \quad \forall x, y \in S.$$

The communication cost of entering the sensor filed, define as

$$(G,T)_{enter} = \sum_{(i,j)\in\Lambda} \zeta^{s}_{(i,j)}\theta_{os}\omega_{(i,j)} \quad \forall s \in S.$$

The communication cost of leaving the sensor filed, define as

$$(G,T)_{leave} = \sum_{(i,j)\in\Lambda} \zeta^{s}_{(i,j)}\theta_{so}\omega_{(i,j)} \quad \forall s \in S.$$

Where *S* is the set of all sensor nodes and Λ is the set of all links. $\omega_{(i,j)}$ is the transmission cost associated with link (*i*,*j*). θ_{xy} is the frequency of moving object from *x* to *y*, θ_{os} is the frequency while object enters sensor field, θ_{so} is the frequency when object leaves sensor field, and the tree links, $\zeta^s_{(i,j)}$, are the links of object tracking tree. The decision variable $\zeta^s_{(i,j)} = 1$ if the sensor node *s* uses the tree link (*i*,*j*) to reach the sink node, and 0 otherwise.

For example, in Figure 6, communication cost is 5 × 8 = 40 when object moves from sensor *x* to sensor *y*, and communication cost is (3+2) × 6 = 30 when object moves from sensor *y* to sensor *x*.

Therefore, the total communication cost for tree *T* as the sum of counting the number of events transmitted in *G*:

Total Communication Cost (*G*,*T*) =

$$\sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda}(1-\zeta^x_{(i,j)})\zeta^y_{(i,j)}\theta_{xy}\omega_{(i,j)}+\sum_{s\in S}\sum_{(i,j)\in\Lambda}\zeta^s_{(i,j)}(\theta_{os}+\theta_{so})\omega_{(i,j)}$$

The detailed descriptions are shown in Table 5.2.

Table 5.2. Problem description in tree-based object tracking problem.

Given	1. The set of sensor nodes. 2. The communication nodes. 3. The set of the object moving frequency. 4. The set of transmission cost associated with link.
Objective	To minimize the total communication cost.
Subject to	1. Routing constraints. 2. Tree constraint. 3. Variable-transformation constraints.
To determine	Object tracking tree.

5.1.3 Mathematical Model

The notations used to model the problem are listed as follows.

Table 5.3. Notations of the given parameters in tree-based object tracking.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
Γ	The set of all communication nodes, including sink node.

o	Artificial node outside the sensor field.
Θ	The set of the object moving frequency from x to y, $\forall x, y \in S \cup \{o\}$, $x \neq y$.
Λ	The set of all links, $(i, j) \in \Lambda$, $i \neq j$.
Ω	The set of transmission costs $\omega_{(i,j)}$ associated with link (i, j).
Φ_s	The set of all candidate paths φ between a pair of nodes, *s and* sink, $\forall s \in S$.

Table 5.4. Notation of the indicate parameter in tree-based object tracking.

Indicate Parameter	
Notation	Description
$\delta_{\varphi(i,j)}$	The value of indicator function is 1 if link (i, j) is on path φ, and 0 otherwise.

Table 5.5. Notations of the decision variables in tree-based object tracking.

Decision Variables	
Notation	Description
$x_{s\varphi}$	1 if the sensor node *s* uses the path φ to reach the sink node, and 0 otherwise.
$\zeta^{s}_{(i,j)}$	1 if the sensor node *s* uses the link (i, j) to reach the sink node, and 0 otherwise.

Problem (IP1):

Objective function:

$$Z_{IP} = \min \sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in \Lambda} (1-\zeta^{x}_{(i,j)})\zeta^{y}_{(i,j)}\theta_{xy}\omega_{(i,j)} + \sum_{s \in S} \sum_{(i,j) \in \Lambda} \zeta^{s}_{(i,j)}(\theta_{os}+\theta_{so})\omega_{(i,j)} \qquad \text{(IP1)}$$

subject to:

$$\sum_{\varphi \in \Phi_s} x_{s\varphi} = 1 \qquad \forall s \in S \qquad (1.1)$$

$$\sum_{j \in \Gamma} \zeta^{s}_{(i,j)} = 1 \qquad \forall s \in S,\ i \in S \cup \Gamma - \{sink\},\ i \neq j \qquad (1.2)$$

$$\sum_{\varphi \in \Phi_s} x_{sp}\delta_{p(i,j)} \leq \zeta^{s}_{(i,j)} \qquad \forall s \in S,\ (i,j) \in \Lambda,\ i \neq j \qquad (1.3)$$

$$\sum_{(i,j)\in\Lambda}(1-\zeta^{x}_{(i,j)})\zeta^{y}_{(i,j)} \geq 1 \qquad \forall x,y\in S,\quad x\neq y \text{ and } i\neq j \qquad (1.4)$$

$$x_{s\varphi} = 0 \text{ or } 1 \qquad \forall s\in S,\ \varphi\in\Phi_s \qquad (1.5)$$

$$\zeta^{s}_{(i,j)} = 0 \text{ or } 1 \qquad \forall s\in S,\ (i,j)\in\Lambda, \text{and } i\neq j. \qquad (1.6)$$

The objective function (IP1) of this problem is to minimize the total communication cost subject to:

Constraint (1.1): Routing constraint which uses one path from sensor node *s* to *sink* node only.

Constraint (1.2): Tree constraint of avoiding cycle. Any outgoing link of a node to communication node is equal to 1 on the object tracking tree.

Constraint (1.3): Routing constraint. Once the path, $x_{s\varphi}$, is selected and the tree link (i,j) is on the path, the decision variable, $\zeta^{s}_{(i,j)}$, must set to be 1.

Constraint (1.4): Sensor *y* must use one or more tree links (*i,j*) to report location of object when object moves from sensor *x* to sensor *y*. Therefore, $\sum_{(i,j)\in\Lambda}(1-\zeta^{x}_{(i,j)})\zeta^{y}_{(i,j)}$ must be greater than or equal to 1.

Constraints (1.5)-(1.6): Decision variables $x_{s\varphi}$ and $\zeta^{s}_{(i,j)}$ equal to 0 or 1.

Problem (IP1) is hard to solve, since original objective function, $\min\sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda}(1-\zeta^{x}_{(i,j)})\zeta^{y}_{(i,j)}\theta_{xy}\omega_{(i,j)}+\sum_{s\in S}\sum_{(i,j)\in\Lambda}\zeta^{s}_{(i,j)}(\theta_{os}+\theta_{so})\omega_{(i,j)}$, and constraint (1.4) are nonlinear.

An auxiliary variable $\upsilon^{xy}_{(i,j)}$ is introduced. Tracking links, $\upsilon^{xy}_{(i,j)}$, are the links when object moves from sensor *x* to sensor *y*, and then sensor *y* delivers tracking information upward to the nearest common ancestor via the tracking links, where $\upsilon^{xy}_{(i,j)}=(1-\zeta^{x}_{(i,j)})\zeta^{y}_{(i,j)}$. Table 5.6 shows the truth table for variables $\zeta^{x}_{(i,j)},\zeta^{y}_{(i,j)}$, and $\upsilon^{xy}_{(i,j)}$.

Table 5.6. The truth table of variables $\zeta^{x}_{(i,j)}, \zeta^{y}_{(i,j)}$, and $\upsilon^{xy}_{(i,j)}$.

$\zeta^{x}_{(i,j)}$	$\zeta^{y}_{(i,j)}$	$\upsilon^{xy}_{(i,j)}$
0	0	0
0	1	1
1	0	0
1	1	0

We also add variable-transformation constraints (2.4 and 2.5) to fulfill the truth table. If $\zeta^{x}_{(i,j)} = 0 \cap \zeta^{y}_{(i,j)} = 1$, $\upsilon^{xy}_{(i,j)}$ must set to be 1, and 0 otherwise.

The constraints can transform nonlinear original objective function to a linear objective function, $\min \sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} \upsilon^{xy}_{(i,j)}\theta_{xy}\omega_{(i,j)} + \sum_{s\in S}\sum_{(i,j)\in\Lambda} \zeta^{s}_{(i,j)}(\theta_{os}+\theta_{so})\omega_{(i,j)}$, and linear constraint (2.6).

Therefore, we add the new decision variable, $\upsilon^{xy}_{(i,j)}$, to reformulate the problem as follows.

Table 5.7. Notation of the decision variable $\upsilon^{xy}_{(i,j)}$.

Decision Variables	
Notation	Description
$\upsilon^{xy}_{(i,j)}$	1 if $\zeta^{x}_{(i,j)} = 0 \cap \zeta^{y}_{(i,j)} = 1$ (reporting object's location uses the link (*i,j*) when object moves from sensor *x* to sensor *y*), and 0 otherwise, $x \neq y$.

Problem (IP2):

Objective function:

$$Z_{IP} = \min \sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} \upsilon^{xy}_{(i,j)}\theta_{xy}\omega_{(i,j)} + \sum_{s\in S}\sum_{(i,j)\in\Lambda} \zeta^{s}_{(i,j)}(\theta_{os}+\theta_{so})\omega_{(i,j)} \qquad \text{(IP2)}$$

subject to:

$$\sum_{\varphi\in\Phi_s} x_{s\varphi} = 1 \qquad \forall s\in S \qquad (2.1)$$

$$\sum_{j\in\Gamma}\zeta^{s}_{(i,j)} = 1 \qquad \forall s\in S,\ i\in S\bigcup\Gamma-\{sink\},\ i\neq j \tag{2.2}$$

$$\sum_{\varphi\in\Phi_s} x_{s\varphi}\delta_{\varphi(i,j)} \le \zeta^{s}_{(i,j)} \qquad \forall s\in S,\ (i,j)\in\Lambda,\ i\neq j \tag{2.3}$$

$$2\upsilon^{xy}_{(i,j)} \le \zeta^{y}_{(i,j)}-\zeta^{x}_{(i,j)}+1 \qquad \forall x,y\in S,\ (i,j)\in\Lambda,\ i\neq j \tag{2.4}$$

$$\zeta^{y}_{(i,j)}-\zeta^{x}_{(i,j)}+1 \le \upsilon^{xy}_{(i,j)}+1 \qquad \forall x,y\in S,\ (i,j)\in\Lambda,\ x\neq y \text{ and } i\neq j \tag{2.5}$$

$$\sum_{(i,j)\in\Lambda}\upsilon^{xy}_{(i,j)} \ge 1 \qquad \forall x,y\in S,\ x\neq y \text{ and } i\neq j \tag{2.6}$$

$$x_{s\varphi} = 0 \text{ or } 1 \qquad \forall s\in S,\ \varphi\in\Phi_s \tag{2.7}$$

$$\zeta^{s}_{(i,j)} = 0 \text{ or } 1 \qquad \forall s\in S,\ (i,j)\in\Lambda, \text{and } i\neq j \tag{2.8}$$

$$\upsilon^{xy}_{(i,j)} = 0 \text{ or } 1 \qquad \forall x,y\in S,\ (i,j)\in\Lambda,\ x\neq y \text{ and } i\neq j. \tag{2.9}$$

The objective function (IP2) of this problem is to minimize the total communication cost subject to:

Constraint (2.1): Routing constraint which uses one path from sensor node *s* to *sink* node only.

Constraint (2.2): Tree constraint of avoiding cycle. Any outgoing link of node to communication node is equal to 1 on the object tracking tree.

Constraint (2.3): Routing constraint. Once the path, $x_{s\varphi}$, is selected and the tree link (i,j) is on the path, the decision variable, $\zeta^{s}_{(i,j)}$, must set to be 1.

Constraint (2.4)-(2.5): There are variable-transformation constraints. If $\zeta^{x}_{(i,j)}=0 \bigcap \zeta^{y}_{(i,j)}=1$, reporting location of object will use the tracking link (i,j) when object moves from sensor *x* to sensor *y*, $\upsilon^{xy}_{(i,j)}$ must set to be 1, and 0 otherwise.

Constraint (2.6): Sensor *y* must use one or more tracking link (*i,j*) to report object's location when object moves from sensor *x* to sensor *y*. Therefore, $\sum_{(i,j)\in\Lambda}\upsilon^{xy}_{(i,j)}$ must be greater than or equal to 1.

Constraints (2.7)-(2.9): Decision variables $x_{s\varphi}$, $\zeta^{s}_{(i,j)}$, and $\upsilon^{xy}_{(i,j)}$ equal to 0 or 1.

5.1.4 Solution Procedure

5.1.4.1 Lagrangean Relaxation

Using the Lagrangean relaxation method successfully adopted to solve many famous NP-complete problems [32][33][34]. The overall procedure to solve the network planning problem is shown in Figure 5.7. The relaxation of the primal problem is developed first which provides lower bound (LB) on the optimal solutions. Since we relax three constraints of the problem (IP2), the boundary is used to design a heuristic approach to get a primal feasible solution. To solve the original problem near-optimally and minimize the gap between the primal problem and the Lagrangean dual problem, we improve the LB by solving the four sub-problems optimally and use the subgradient method to adjust the multipliers per iteration. Then, subgradient optimization procedure is used for further improving these solutions by updating the Lagrangean multipliers.

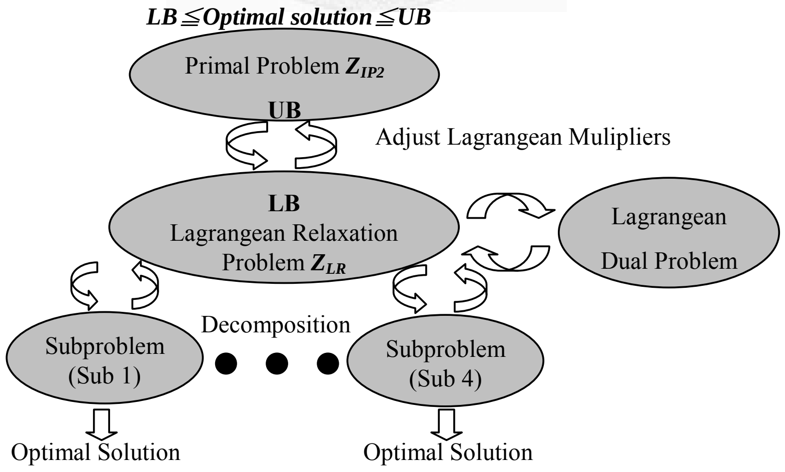


Figure 5.7. The procedure of Lagrangean relaxation.

We can transform the primal problem (IP2) into the following Lagrangean relaxation problem (LR) where constraints (2.3), (2.4), and (2.5) are relaxed. For a vector of non-negative Lagrangean multipliers, a Lagrangean relaxation problem of (IP2) is given by:

Problem (LR):

Objective function:

$$Z_{LR}(u^1_{s(i,j)}, u^2_{xy(i,j)}, u^3_{xy(i,j)}) = \min\{\sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} \upsilon^{xy}_{(i,j)}\theta_{xy}\omega_{(i,j)} + \sum_{s\in S}\sum_{(i,j)\in\Lambda} \zeta^s_{(i,j)}(\theta_{os}+\theta_{so})\omega_{(i,j)}$$

$$+\sum_{s\in S}\sum_{(i,j)\in\Lambda}\sum_{\varphi\in\Phi_s} u^1_{s(i,j)} x_\varphi \delta_{\varphi(i,j)} - \sum_{s\in S}\sum_{(i,j)\in\Lambda} u^1_{s(i,j)}\zeta^s_{(i,j)}$$

$$+\sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} u^2_{xy(i,j)} 2\upsilon^{xy}_{(i,j)} - \sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} u^2_{xy(i,j)}(\zeta^y_{(i,j)} - \zeta^x_{(i,j)} + 1) \qquad \text{(LR)}$$

$$+\sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} u^3_{xy(i,j)}(\zeta^y_{(i,j)} - \zeta^x_{(i,j)} + 1) - \sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} u^3_{xy(i,j)}(\upsilon^{xy}_{(i,j)} + 1)\}$$

subject to: (2.1), (2.2), (2.6), (2.7), (2.8), and (2.9).

Where $u^1_{s(i,j)}$, $u^2_{xy(i,j)}$, and $u^3_{xy(i,j)}$ are Lagrangean multipliers and $u^1_{s(i,j)}$, $u^2_{xy(i,j)}$, and $u^3_{xy(i,j)} \geq 0$. To solve (LR), we can decompose (LR) into the following four independent and easily solvable optimization sub-problems.

$$Z_{LR} = Z_{sub1} + Z_{sub2} + Z_{sub3} + Z_{sub4}$$

Sub-problem 1: (related to the decision variables $\upsilon^{xy}_{(i,j)}$)

Objective function:

$$Z_{sub1}(u^2_{xy(i,j)}, u^3_{xy(i,j)}) = \min\sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} \upsilon^{xy}_{(i,j)}(\theta_{xy}\omega_{(i,j)} + 2u^2_{xy(i,j)} - u^3_{xy(i,j)}) \qquad \text{(sub 1)}$$

subject to: (2.6) and (2.9).

This sub-problem is related to decision variable $\upsilon^{xy}_{(i,j)}$, which can be further decomposed into $|S|^2|\Lambda|$ sub-problems.

Constraint (2.6) is a redundant constraint used to reduce the duality gap. The duality gap is defined as the difference between the optimal primal objective value and the optimal dual objective value. The smaller duality gap computed, the better the optimality.

Two cases are listed below to determine the value of $v_{(i,j)}^{xy}$.

Let $\beta_{xy(i,j)}$ denote the weight of the object while moving from sensor *x* to sensor *y* using the tracking link (*i*, *j*), we get

$$\beta_{xy(i,j)} = (\theta_{xy}\omega_{(i,j)} + 2u_{xy(i,j)}^{2} - u_{xy(i,j)}^{3})$$

Case 1: If $\beta_{xy(i,j)} < 0$, then assign $v_{(i,j)}^{xy} = 1$

Case 2: If $\beta_{xy(i,j)} \geq 0$, then assign $v_{(i,j)}^{xy} = 0$.

If the sum of each pair of node $v_{(i,j)}^{xy}$ is zero, we enforce to select the minimum positive objective value $\beta_{xy(i,j)}$ and set $v_{(i,j)}^{xy}=1$ to fulfill the constraint (6).

Sub-problem 2: (related to the decision variables $x_{s\varphi}$)

Objective function:

$$Z_{sub2}(u_{s(i,j)}^{1}) \quad = \quad \min \sum_{s\in S} \sum_{(i,j)\in\Lambda} (u_{s(i,j)}^{1} \sum_{\varphi\in\Phi} x_{s\varphi}\delta_{\varphi(i,j)}) \qquad \text{(sub 2)}$$

subject to: (2.1) and (2.7).

The sub-problem 2 can be further decomposed into $|S|$ independent shortest path problems with nonnegative arc weight whose value is $u_{s(i,j)}^{1}$. The value of x_{sp} can be determined by the link cost, $u_{s(i,j)}^{1}$. This sub-problem is related to the decision variables x_{sp}, which can use the Dijkstra's algorithm to solve the single source shortest path problem. The time complexity of Dijkstra's algorithm is $O(|S|^{2})$. The time complexity of the sub-problem is $O(|S|^{3})$.

Sub-problem 3: (related to the decision variables $\zeta^{s}_{(i,j)}$)

Objective function:

$$Z_{sub3}(u^{1}_{s(i,j)}, u^{2}_{xy(i,j)}, u^{3}_{xy(i,j)}) =$$

$$\min \sum_{s\in S}\sum_{(i,j)\in\Lambda}[(\theta_{os}+\theta_{so})\omega_{(i,j)} - u^{1}_{s(i,j)} + \sum_{x\in S}(u^{3}_{xs(i,j)} - u^{2}_{xs(i,j)}) - \sum_{y\in S}(u^{3}_{sy(i,j)} - u^{2}_{sy(i,j)})]\zeta^{s}_{(i,j)} \quad \text{(sub 3)}$$

subject to: (2.2) and (2.8).

This sub-problem is related to the decision variables $\zeta^{s}_{(i,j)}$ which can be further decomposed into $|S|$ sub-problems.

Let $\psi_{s(i,j)}$ denote the weight of the sensor nodes s using the tree link (*i*, *j*), we get

$$\psi_{s(i,j)} = (\theta_{os}+\theta_{so})\omega_{(i,j)} - u^{1}_{s(i,j)} + \sum_{x\in S}(u^{3}_{xs(i,j)} - u^{2}_{xs(i,j)}) - \sum_{y\in S}(u^{3}_{sy(i,j)} - u^{2}_{sy(i,j)})$$

$\zeta^{s}_{(i,j)}$ must be enforced to 1 when choosing the minimum of $\psi_{s(i,j)}$ for each *s* and *i*_to fulfill the constraint (2.2). The time complexity of this sub-problem is $O(|S|^2|\Lambda|)$.

Sub-problem 4: (Constant Part)

Objective function:

$$Z_{sub4}(u^{2}_{xy(i,j)}) = -\sum_{x\in S}\sum_{y\in S}\sum_{(i,j)\in\Lambda} u^{2}_{xy(i,j)} \quad \text{(sub 4)}$$

The sub-problem 4 is constant part. The time complexity of the sub-problem is $O(|S|^2|\Lambda|)$.

According to the weak Lagrangean duality theorem [5, 6], $Z_D(u^{1}_{s(i,j)}, u^{2}_{xy(i,j)}, u^{3}_{xy(i,j)})$ is a lower bound (LB) on Z_{IP} when $u^{1}_{s(i,j)}$, $u^{2}_{xy(i,j)}$, and

$u^3_{xy(i,j)} \geq 0$. The following dual problem (D) is then constructed to calculate the tightest lower bound.

Dual Problem (D):

Objective function:

$$Z_D = \max Z_{LR}(u^1_{s(i,j)}, u^2_{xy(i,j)}, u^3_{xy(i,j)}) \qquad \text{(D)}$$

subject to:

$$u^1_{s(i,j)},\ u^2_{xy(i,j)},\ \text{and}\ u^3_{xy(i,j)} \geq 0 \qquad (2.10)$$

There are several methods for solving the dual-mode problem (D). One of the most popular approach is the subgradient method.

5.1.4.2. Getting Primal Feasible Solutions

After optimally solving the Lagrangean dual problem, we get a set of decision variables and develop tree-based heuristic algorithm to tune these decision variables. A set of feasible solutions of the primal problem (IP2) therefore can be obtained. The primal feasible solution is an upper bound (*UB*) of the primal problem (IP2), and the Lagrangean dual problem solution guarantees the lower bound (*LB*) of the primal problem (IP2). Iteratively, by solving primal feasible solution and Lagrangean dual problem, we get *UB* and *LB*, respectively.

The procedure of tree-based object tracking algorithm is shown in Figure 5.8.

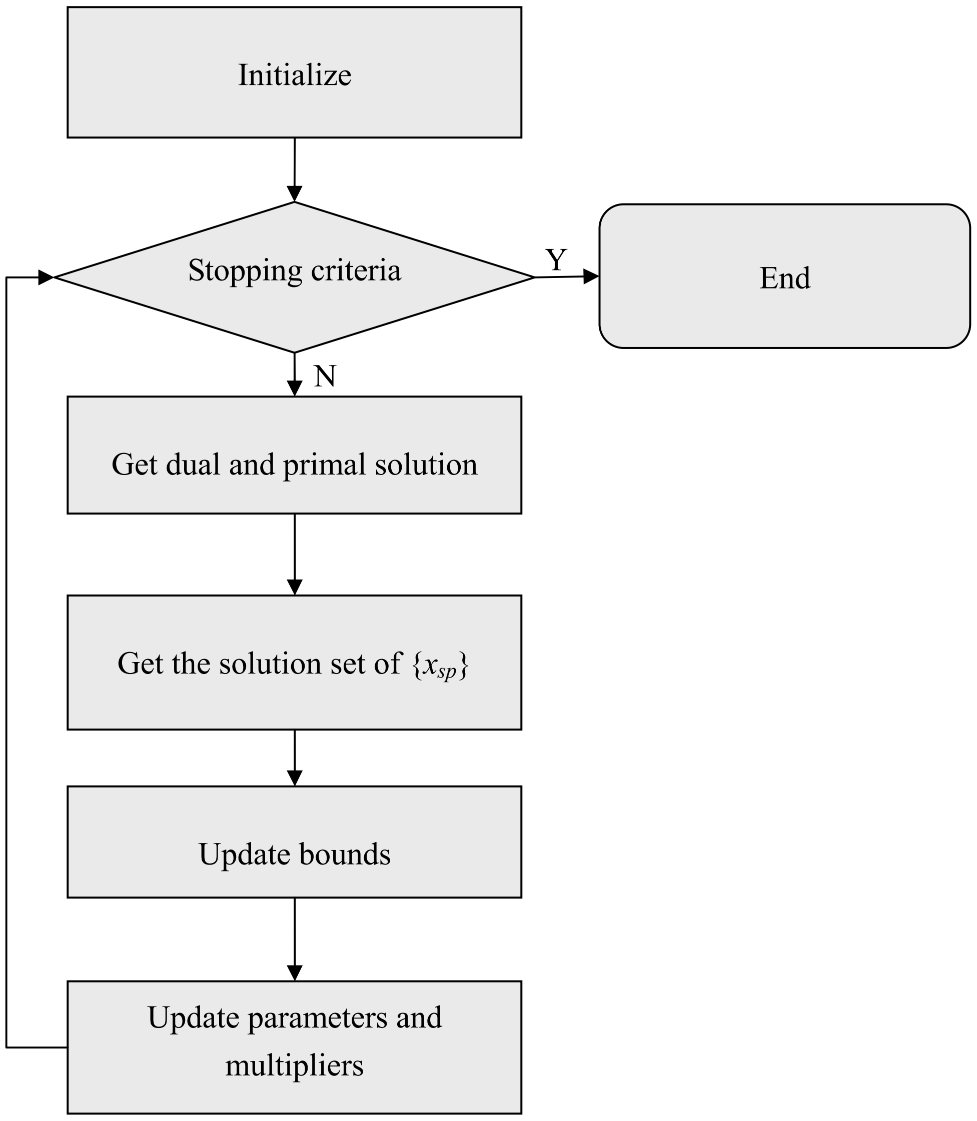

Initialize
Stopping criteria
Y
End
N
Get dual and primal solution
Get the solution set of $\{x_{sp}\}$
Update bounds
Update parameters and multipliers

Figure 5.8. The procedure of tree-based object tracking algorithm.

A tree-based object tracking algorithm is listed in Figure 5.9.

```
Algorithm Tree-based Object Tracking
Input: 2D routing and sensor sub-graphs
Output: Object tracking tree
1:  begin
2:      Initialize the Lagrangean multiplier vectors ($u^1, u^2, u^3$) to be zero vectors;
3:      UB:=total communication cost of shortest path tree; LB:=very small value;
4:      improve_counter:=0; step_size_coefficient:=2; improve_Threshold:=50;
5:      Using the shortest path tree algorithm (SPA) to find the initial primal value;
6:      for iteration:=1 to Max_Iteration_Number do
7:      begin
8:          run sub-problem(SUB1);
9:          run sub-problem(SUB2);
10:         run sub-problem(SUB3);
11:         run sub-problem(SUB4);
12:         calculate $Z_D$;
13:         if $Z_D$>LB then LB:= $Z_D$ and improve_counter:=0;
14:         else improve_counter:= improve_counter+1;
15:         if improve_counter= improve_Threshold then
16:             improve_counter:=0; $\alpha := \alpha / 2$;
17:         Adjust arc weight $c_{(i,j)} = \sum_{s \in S} u^1_{s(i,j)}$ for each link (i,j)
                and then run the Dijkstra algorithm to get the solution set of $\{x_{s\varphi}\}$;
18:         Once $\{x_{s\varphi}\}$ is determined, $\upsilon^{xy}_{(i,j)}$ and $\zeta^{s}_{(i,j)}$ are also determined;
19:         Get a new object tracking tree and calculate newly upper bound ub
20:         if ub<UB then UB:=ub;
21:         run updata-step-size;
22:         run updata-Lagrangean-multiplier;
23:     end;
24: end;
```

Figure 5.9. The tree-based object tracking algorithm.

In the algorithm, from steps 2-4 are setting initialize value, step 5 is finding the initial primal value. Steps 8-11 solve the sub-problems 1-4. Steps 12-16 and 20-22 update the parameters and multipliers. Steps 17-19 are used to get primal feasible solution.

5.1.5 Computational Results

To evaluate the performance of the proposed algorithm, we conduct an experiment. The performance is assessed in terms of the total communication cost.

5.1.5.1. Scenario

The proposed algorithm is coded in C++ under a Microsoft Visual C++ 6.0 development environment. All the experiments are performed on a Core 2 Duo-2.2 GHz PC with 4GB memory running Microsoft Windows VISTA. The algorithm is tested on a 2D sensor field. We distribute 12, 23, 36, 50, and 105 sensor and communication nodes, respectively, in a 2D sensor field.

The parameters listed in Table 5.8 are used for the all cases of experiments.

Table 5.8. Parameter of Lagrangean relaxation-based algorithm.

Parameter	Value
Number of nodes	12 ~ 105 (depend on each case)
Number of iterations	5,000
Improvement counter threshold	49
Initial upper bound	10^{10}
Initial upper bound	-10^{10}
Initial scalar of step size	2
Initial multiplier	0

5.1.5.2. Experiment results

In order to evaluate the proposed tree-based algorithm, we compare the algorithm with another heuristic algorithm, shortest path tree (SPT) algorithm. We also compare the proposed tree-based algorithm with the lower bound (*LB*) of the dual mode problem.

Figure 5.10 shows an example of LR-based object tracking tree.

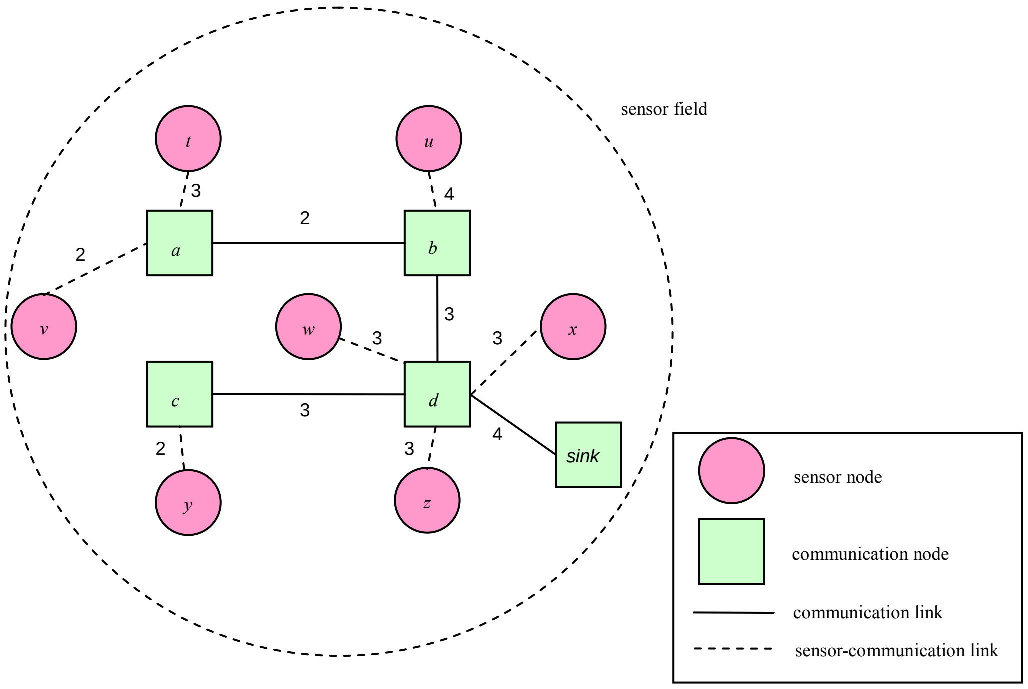


Figure 5.10. An example of LR-based object tracking tree.

Table 5.9 shows the total transmission cost calculated by different algorithms under the number of nodes 12, 23, 36, 50, and 105 respectively. We can see that the tree-based heuristic algorithm outperforms the SPT algorithm. We denote the dual solution as “Z_{du}” (*LB*), and tree-based heuristic solution as “Z_{IP2}” (*UB*). The gap between *UB* and *LB* is computed by $|(UB - LB)/LB| * 100\%$ which illustrates the optimality of problem solution.

Table 5.9. Evaluation of the gap and improvement ratio with different number of nodes.

Number of nodes		Z_{du}	Z_{IP2}	Gap	SPT	Improvement Ratio to SPT
12	Problem 1	2774	3127	0.13	3630	0.16
	Problem 2	3416	3906	0.14	4460	0.14
23	Problem 1	17850	20725	0.16	22491	0.09
	Problem 2	17385	20282	0.17	21839	0.08
36	Problem 1	42410	49970	0.18	57553	0.15
	Problem 2	42775	50411	0.18	57787	0.15
50	Problem 1	89824	78807	0.14	99639	0.11
	Problem 2	77905	88195	0.13	102796	0.17
105	Problem 1	326529	371438	0.14	508314	0.37
	Problem 2	328911	355546	0.08	511402	0.44

Figure 5.11 shows an example of the trend line for getting the primal problem solution values (*UB*) and dual mode problem values (*LB*). The *UB* curve tends to decrease to get the minimum feasible solution. In contrast, the *LB* curve tends to increase and converge rapidly to reach the optimal solution. The LR-based method ensures the optimization results between *UB* and *LB* so that we can keep the duality gap as small as possible in order to improve the quality of our solution and achieve near optimization.

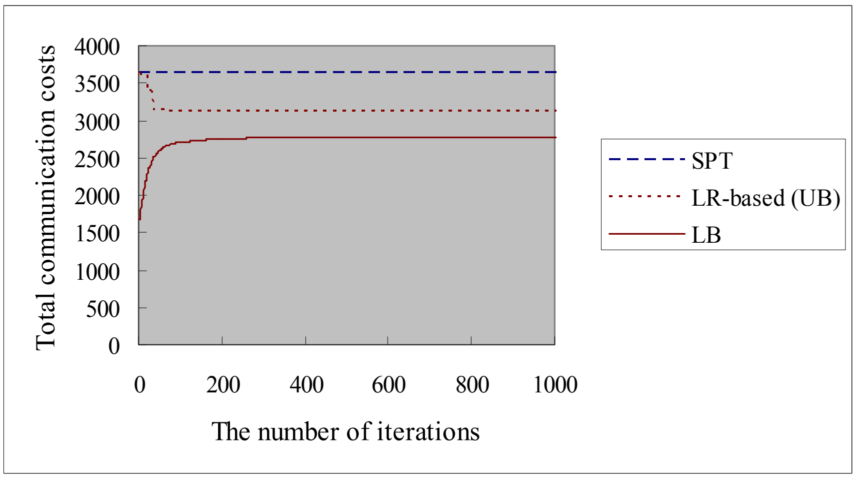


Figure 5.11. The execution results of LR-based algorithm with 12 nodes in the test problem 1.

Table 5.10 shows that the time complexity of our LR-based solution is dominated by Lagrangean dual problem. The Lagrangean dual problem has been solved by the above four sub-problems with the maximum number of iteration *I*.

Table 5.10. The time complexity of tree-based object tracking tree algorithm.

Problem	Time Complexity
Sub-problem (SUB1)	$O(\|S\|^2\|\Lambda\|)$
Sub-problem (SUB2)	$O(\|S\|^3)$
Sub-problem (SUB3)	$O(\|S\|^2\|\Lambda\|)$
Sub-problem (SUB4)	$O(\|S\|^2\|\Lambda\|)$
Getting primal feasible solutions	$O(\|S\|^2)$
Lagrangean dual problem	$O(I\|S\|^2\|\Lambda\|)^*$
*Parameter *I* means the maximum number of iterations	

5.1.5.3 Discussion

The experiment results show that the algorithm is better than the shortest path tree algorithm, and the gap is also small. In other words, when compared with SPT algorithm, the proposed TOTA can improve the percentage of energy consumption from 8% to 44%. It also achieves the near optimal solution since the gaps are only from 8% to 18%, as shown in Table 5.9. Therefore, the results show that the proposed tree-based algorithm can achieve energy-efficient object tracking. Furthermore, the algorithm is very efficient and scalable in terms of the running time.

5.1.6 Concluding Remarks

This study proposes an object tracking algorithm in wireless sensor networks. To our best knowledge, the proposed LR-based algorithm is truly novel and it has not been discussed in previous researches. This study first formulates the problem as a 0/1 integer programming problem, and then proposes a tree-based heuristic algorithm to solve the optimization problem.

We are planning to further investigate response time model based on object tracking application requirements and heuristic algorithms in the near future. In addition, we are looking into the tradeoff of total communication cost with various system issues, such as response time, report frequency, and number of sinks, etc.

5.2 Prediction-based Object Tracking Algorithm

This section is organized as follows. The overview is described in Section 5.2.1. The problem and prediction model are described in Sections 5.2.2 and 5.2.3, respectively. In addition, the solution procedure is presented in Section 5.2.4. Furthermore, the computational results are discussed in Section 5.2.5, and conclusions are presented in Section 5.2.6.

5.2.1 Overview

The prediction-based algorithm can minimize the number of nodes participating in the tracking. In addition, the mechanism can prolong the system lifetime since the cost of computation less than the cost of communication. The varieties of wake up mechanisms and recovery mechanisms will affect the system performance. The prediction model works well if it tolerates small number of errors and some latency. The basic method is that the sensors do not have to transmit the expected readings [13][14][15].

5.2.2 Problem Description

In the prediction-based object tracking model. There are three basic prediction models are as follows.

1. Linearly Prediction

Linearly prediction uses the previous two locations of an object to predict the third location linearly. It assumes that the object will stay in the current speed and direction.

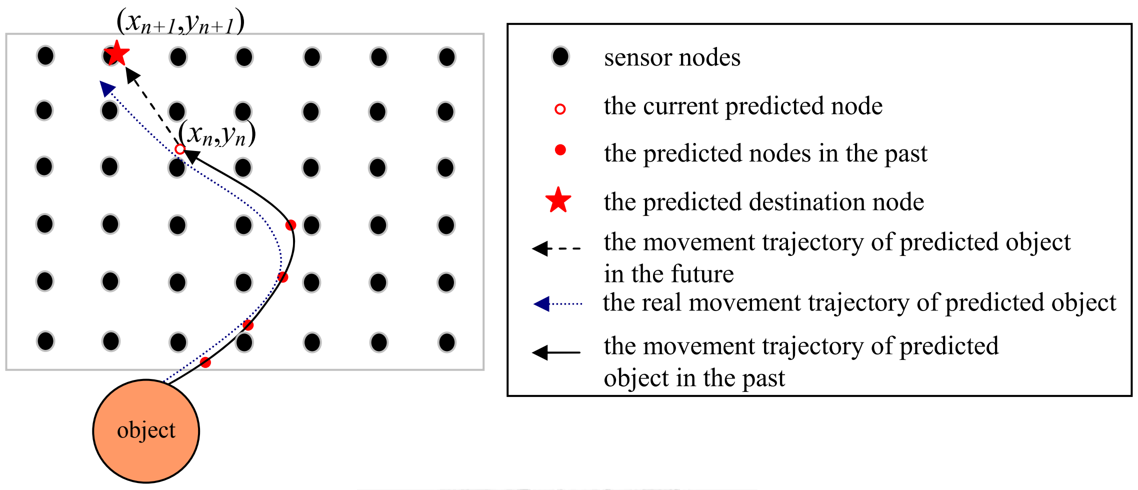
2. Averagely Prediction

Averagely prediction uses the average of the object's moving track history to derive the future speed and direction.

3. Dynamically Prediction

Dynamically prediction assigns different weights to the different stages of history.

Figure 5.12 illustrates a scenario of prediction-based object tracking. Prediction-based approach is used to predict the upcoming location of mobile object for energy saving. System uses the historical data to predict next location of mobile object.



(x_{n+1},y_{n+1})
(x_n,y_n)
object
sensor nodes
the current predicted node
the predicted nodes in the past
the predicted destination node
the movement trajectory of predicted object in the future
the real movement trajectory of predicted object
the movement trajectory of predicted object in the past

Figure 5.12. A scenario of prediction-based object tracking.

We assume that sensors are regularly deployed in tracking field. We develop a prediction-based algorithm based on dynamically prediction model to solve object tracking problem. The prediction-based approach can reduce the power consumption in wireless sensor networks by limiting the sensor active time. The tracking algorithm supports the sensor sleeping mechanism to save energy and prolong the system lifetime.

5.2.3 Prediction Model

In this section, we use the concept of Viterbi algorithm to calculate object location [38]. The system maintains n - 1, n - 2, and n - h speed and direction of the object at time interval n. The algorithm is called prediction-based object tracking algorithm (POTA). The detailed descriptions are shown in Table 5.11.

Table 5.11. Problem description in prediction-based object tracking problem.

Given	The set of sensor nodes.
Objective	1. To minimize the number of nodes participating in the object tracking. 2. To maximize the accuracy of object predicted position. 3. To minimize out of tracking probability.
To determine	The h value at time interval n.
To predict	The location of object at time interval n.

The model uses tuning parameters α and h to vary the degree of movement in the modified Viterbi algorithm. The parameter α is to vary the degree of randomness in mobility pattern in Section 2.1.4.

$$s_n = \beta s_{(n-1,n-2)} + (1-\beta) s_{(n-1,n-h)}$$

$$d_n = \beta d_{(n-1,n-2)} + (1-\beta) d_{(n-1,n-h)}$$

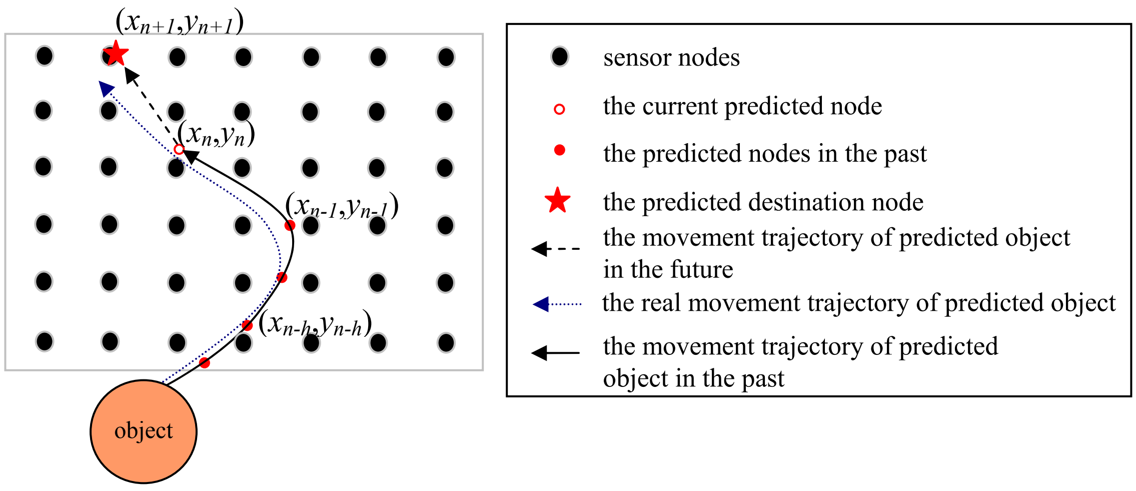
where s_n and d_n are the new estimative speed and direction of the object at time interval n; where $0 \le \alpha \le 1$, is the tuning parameter used to vary the object; $s_{(n-1,n-2)}$ and $d_{(n-1,n-2)}$ are speed and direction trends in short term, and $s_{(n-1,n-h)}$ and $d_{(n-1,n-h)}$ are speed and direction trends in long term. Totally short term is obtained by setting $\beta = 1$ and long term is obtained by setting $\beta = 0$. Intermediate levels of speed and direction are obtained by varying the value of a between 0 and 1.

At each time interval the next location is calculated based on the current location, speed, and direction of movement. Specifically, at time interval *n*, a position of object is given by the following equations:

$$x_{n+1} = x_n + s_n t \cos d_n$$
$$y_{n+1} = y_n + s_n t \sin d_n$$

where (x_n, y_n) and (x_{n+1}, y_{n+1}) are the *x* and *y* coordinates of the object's position at the n^{th} and $(n + 1)^{th}$ time intervals, respectively, *t* is time unit, and s_n and d_n are the speed and direction of the object, respectively, at the n^{th} time interval.

Figure 5.13 illustrates a scenario of POTA.



(x_{n+1},y_{n+1})
(x_n,y_n)
(x_{n-1},y_{n-1})
(x_{n-h},y_{n-h})
object
sensor nodes
the current predicted node
the predicted nodes in the past
the predicted destination node
the movement trajectory of predicted object in the future
the real movement trajectory of predicted object
the movement trajectory of predicted object in the past

Figure 5.13. A scenario of POTA.

System parameters, *α* and *h*, depend on object movement behavior.

5.2.4 Solution Procedure

We formulate the problem as a multiple criteria decision problem with 3 goals [62][65]:

$$Minimize\{U(p_i)\} = f\{U_1(p_i), U_2(p_i), U_3(p_i)\}$$

The combined objective function can be defined as

$$Minimize\{\sum_{j=1}^{3} w_j U_j(p_i)\}$$

where w_1, w_2, and w_3 are significant weights reflecting the relative importance of each goal.

Where $U_1(p_i)$ measures the energy consumption under the policy of prediction p_i, $U_2(p_i)$ measures the miss rate under the policy of prediction p_i, and $U_3(p_i)$ measures the maximum latency of one step under the policy of prediction p_i.

The relative importance of these utility functions is defined by the weights w_1, w_2, and w_3. Weights are used to assign different importance to the different performance metrics. For example, if the miss rate is a critical factor, then a high value should be assigned to w_2.

In the prediction-based object tracking algorithm, we propose 3 policies to deal with the problem.

The policy 1 includes two cases. First case is to predict object location and turn on predicted destination node and 1 hop neighbors without round advance. The meaning of round advance is to skip current round when object is out of tracking in current round. System turns on all sensor nodes on the next round if system is also out of tracking. Second case is to predict object location and turn on predicted destination node and 1 hop neighbors with round advance. The predicted procedure of the POTA of policy 1 is shown in Figure 5.14.

The policy 2 includes two cases. First case is to predict object location and turn on predicted destination node and 1 hop neighbors without round advance. It turns on 2 hop neighbors, if system can not find the object. Second case is to predict object location and turn on predicted destination node and 1 or 2 hops neighbors with round advance. The predicted procedure of the POTA of policy 2 is shown in Figure 5.15.

The policy 3 includes two cases. First case is to predict object location, and turn on predicted destination node, 1 hop and 2 hops neighbors without round advance. It turns on 1 hop neighbors, if system can find the object in the past 2*_hop_ub* times. And it turns on 1 hop and 2 hops neighbors, if system can not find the object in the past 1*_hop_ub* times. Second case is to predict object location, and turn on predicted destination node, 1 hop and 2 hops neighbors with round advance. The predicted procedure of the POTA of policy 3 is shown in Figure 5.16.

Table 5.12 illustrates a comparison among the policy 1, policy 2, and policy 3.

Besides, the *h* parameter adds 1 when $h < h\_ub$ and object is found. The *h* parameter subtracts 1 when deviant range > ± deviant angle, $h > h\_lb$, and object is not found.

Table 5.12. A comparison among the policy 1, policy 2, and policy 3.

	Policy 1	Policy 2	Policy 3
Prediction mechanism	Turn on predicted destination node and 1 hop neighbors	Turn on predicted destination node and 1 or 2 hops neighbors	Turn on predicted destination node and 1 or 2 hops neighbors
Turn on 2 hops neighbors	-	Turn on 2 hops neighbors if system misses the object when turning on predicted destination node and 1 hop neighbors	Turn on 2 hops neighbors if system can not find the object and continuous 1*_hop_ub* times turn on predicted destination node and 1 hop neighbors
Turn on 1 hop neighbors	-	-	Turn on 1 hop neighbors if system can find the object and continuous 2*_hop_ub* times turn on predicted destination node and 2 hops neighbors
Latency	2	3	2

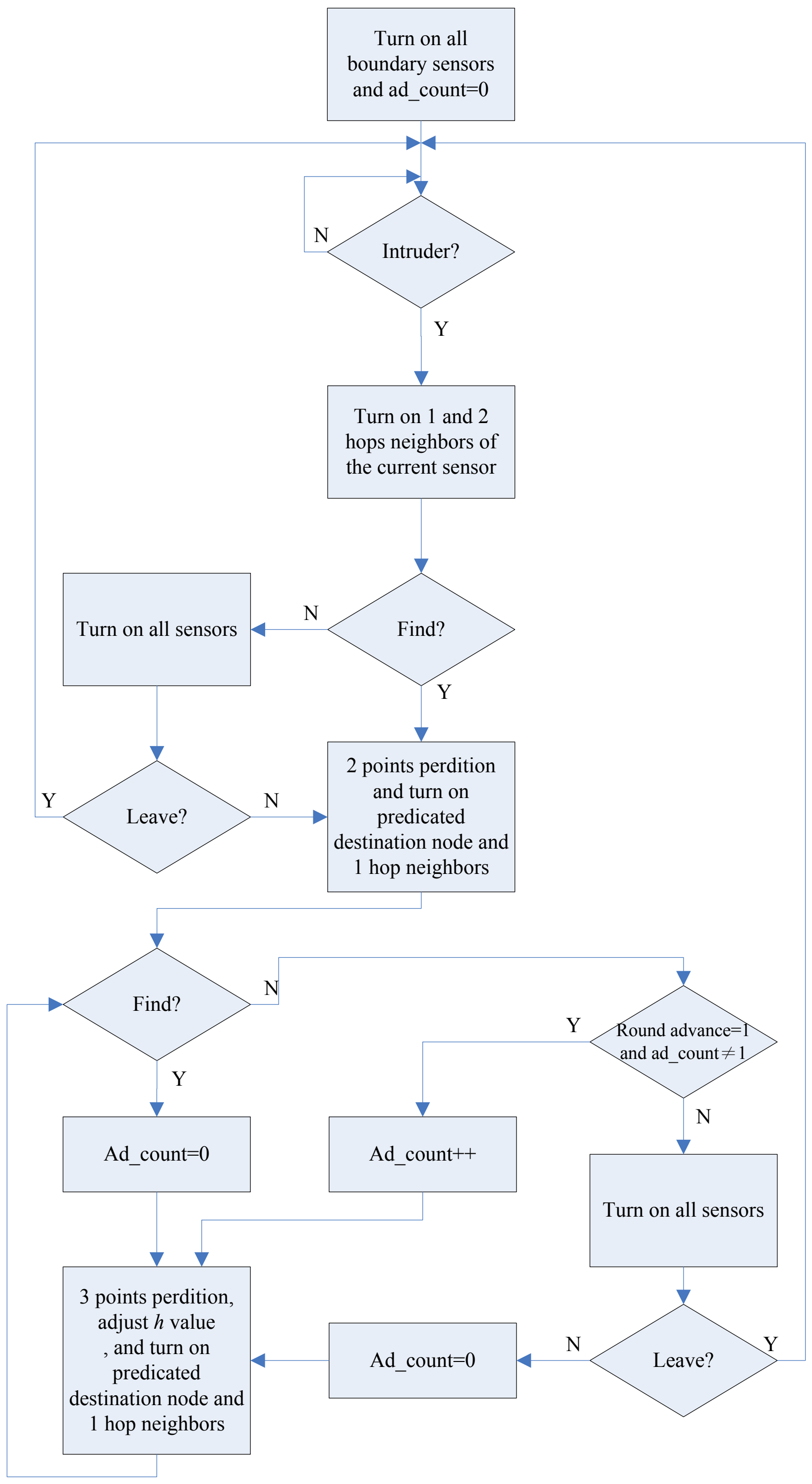

Turn on all boundary sensors and ad_count=0
Intruder?
N
Y
Turn on 1 and 2 hops neighbors of the current sensor
Find?
N
Y
Turn on all sensors
Leave?
Y
N
2 points perdition and turn on predicated destination node and 1 hop neighbors
Find?
N
Y
Round advance=1 and ad_count≠1
Y
N
Ad_count=0
Ad_count++
Turn on all sensors
3 points perdition, adjust h value , and turn on predicated destination node and 1 hop neighbors
Ad_count=0
Leave?
N
Y

Figure 5.14. The predicted procedure of the POTA of policy 1.

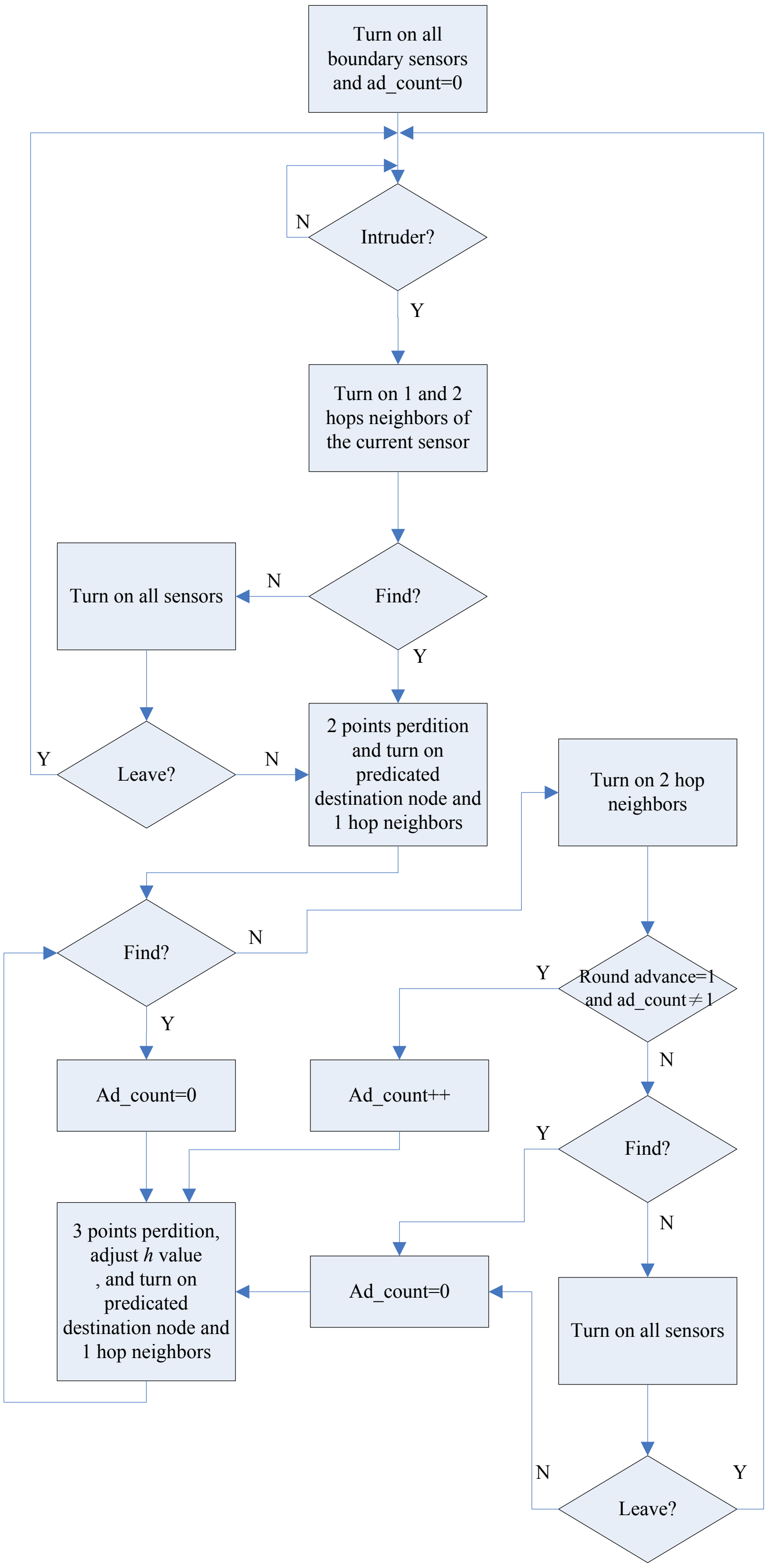

Turn on all boundary sensors and ad_count=0
N
Intruder?
Y
Turn on 1 and 2 hops neighbors of the current sensor
Turn on all sensors
N
Find?
Y
Y
Leave?
N
2 points perdition and turn on predicated destination node and 1 hop neighbors
Turn on 2 hop neighbors
Find?
N
Y
Round advance=1 and ad_count≠1
Y
Ad_count=0
Ad_count++
N
Y
Find?
3 points perdition, adjust h value , and turn on predicated destination node and 1 hop neighbors
Ad_count=0
N
Turn on all sensors
N
Y
Leave?

Figure 5.15. The predicted procedure of the POTA of policy 2.

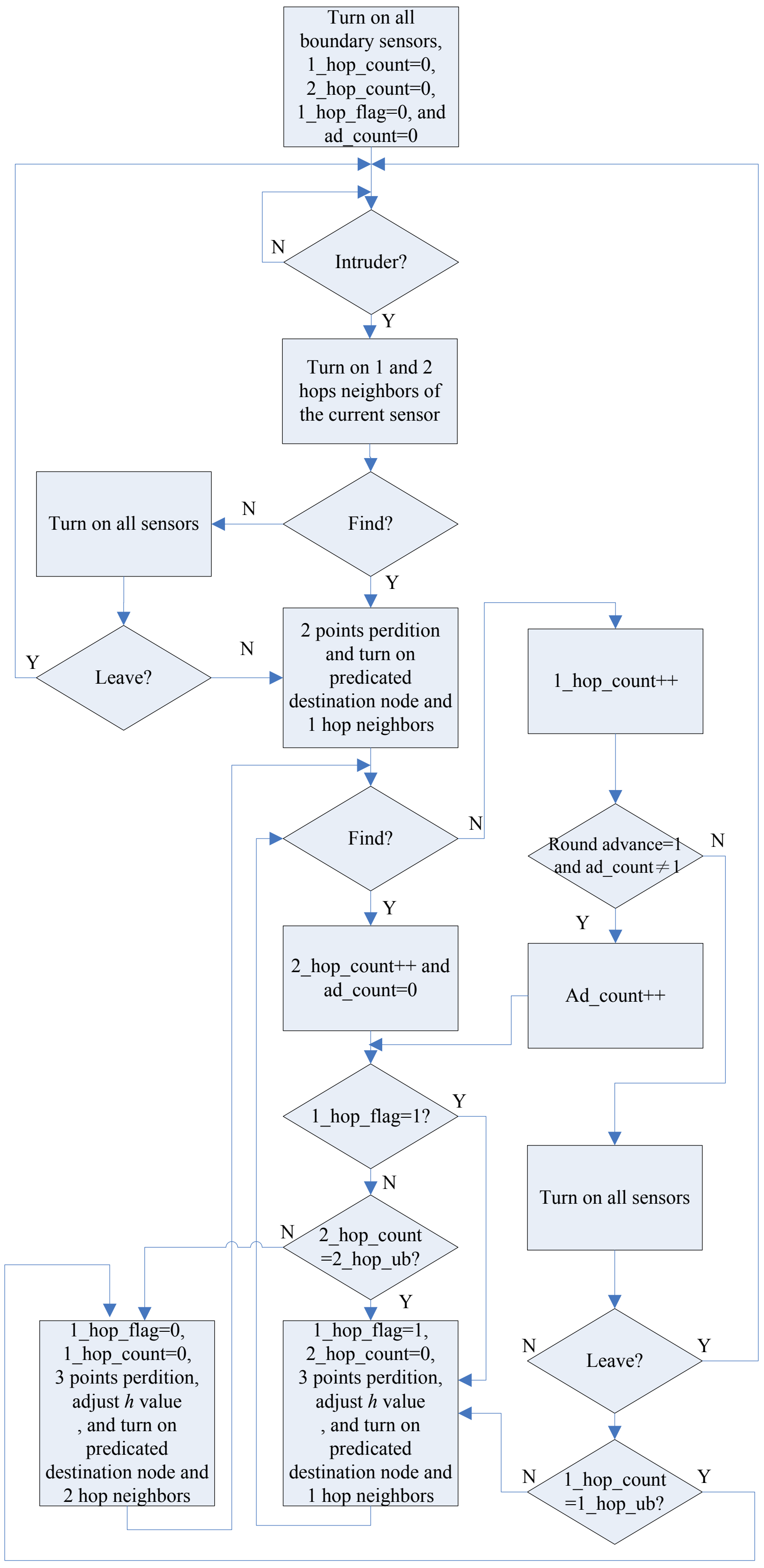

Turn on all boundary sensors, 1_hop_count=0, 2_hop_count=0, 1_hop_flag=0, and ad_count=0
Intruder?
N
Y
Turn on 1 and 2 hops neighbors of the current sensor
Find?
N
Y
Turn on all sensors
Leave?
Y
N
2 points perdition and turn on predicated destination node and 1 hop neighbors
Find?
N
Y
1_hop_count++
Round advance=1 and ad_count≠1
N
Y
Ad_count++
2_hop_count++ and ad_count=0
1_hop_flag=1?
Y
N
2_hop_count =2_hop_ub?
N
Y
Turn on all sensors
Leave?
N
Y
1_hop_count =1_hop_ub?
N
Y
1_hop_flag=0, 1_hop_count=0, 3 points perdition, adjust h value , and turn on predicated destination node and 2 hop neighbors
1_hop_flag=1, 2_hop_count=0, 3 points perdition, adjust h value , and turn on predicated destination node and 1 hop neighbors

Figure 5.16. The predicted procedure of the POTA of policy 3.

5.2.5 Computational Results

To evaluate the performance of the proposed algorithm, we conduct an experiment. The performance is assessed in terms of total number of rounds.

5.2.5.1 Experiment Environment

The proposed algorithm is coded in C under a Dev C++ 4.9.9.2 development environment. All the experiments are performed on a Core 2 Duo 2.2GHz CPU running Microsoft Windows Vista. The algorithm is tested on a 2D monitoring region. We distribute 10201 sensor nodes in 2D monitoring region. The radius of sensors is $\sqrt{25/2}$.

5.2.5.2 Experiment results

The parameters of POTA is listed in Table 5.13.

Table 5.13. The parameters of POTA.

Parameters	Problem1	Problem 2	Problem 3
	Value	Value	Value
β	0.7	0.9	0.9
h_lb	3	3	3
h_ub	7	7	5
Deviant angle	π/3	π/3	π/3
The radius of sensors	$\sqrt{25/2}$	$\sqrt{25/2}$	$\sqrt{25/2}$
Number of sensor node (*sn*)	10201	10201	10201
Monitoring region (m^2)	500 × 500	500 × 500	500 × 500

The evaluation of the performance metrics with different policies and α is listed in Table 5.14, Table 5.15 and Table 5.16.

Table 5.14. Evaluation of the performance metrics with different policies and α in the problem 1.

α	Performance metrics	Policy 1		Policy 2		Policy 3	
		Turn on 1 hop neighbors	Turn on 1 hop neighbors and round advance = 1	Turn on 1 or 2 hop neighbors	Turn on 1 or 2 hop neighbors and round advance = 1	Dynamic turn on 1 or 2 hop neighbors	Dynamic turn on 1 or 2 hop neighbors and round advance = 1
0.4	Total energy consumption	1087875	793834	951340	710406	961140	719751
	Miss rate	0.38	0.53	0.32	0.48	0.33	0.49
	Maximum latency of one step	2	2	3	3	2	2
0.5	Total energy consumption	612076	445461	485060	364227	485060	373586
	Miss rate	0.36	0.51	0.28	0.42	0.28	0.43
	Maximum latency of one step	2	2	3	3	2	2
0.6	Total energy consumption	450430	332825	440955	312280	440955	312280
	Miss rate	0.34	0.49	0.33	0.47	0.33	0.47
	Maximum latency of one step	2	2	3	3	2	2
0.7	Total energy consumption	297018	208816	209050	182724	238448	182706
	Miss rate	0.33	0.43	0.23	0.41	0.26	0.41
	Maximum latency of one step	2	2	3	3	2	2
0.8	Total energy consumption	265197	206378	206613	189280	216406	189259
	Miss rate	0.32	0.46	0.24	0.49	0.26	0.49
	Maximum latency of one step	2	2	3	3	2	2
0.9	Total energy consumption	200307	151289	171068	146310	180870	145906
	Miss rate	0.30	0.44	0.25	0.44	0.27	0.46
	Maximum latency of one step	2	2	3	3	2	2

Table 5.15. Evaluation of the performance metrics with different policies and α in the problem 2.

α	Performance metrics	Policy 1		Policy 2		Policy 3	
		Turn on 1 hop neighbors	Turn on 1 hop neighbors and round advance = 1	Turn on 1 or 2 hop neighbors	Turn on 1 or 2 hop neighbors and round advance = 1	Dynamic turn on 1 or 2 hop neighbors	Dynamic turn on 1 or 2 hop neighbors and round advance = 1
0.4	Total energy consumption	1029095	774249	843573	681829	872943	700559
	Miss rate	0.35	0.52	0.28	0.46	0.29	0.48
	Maximum latency of one step	2	2	3	3	2	2
0.5	Total energy consumption	543458	376839	338125	317230	416478	345319
	Miss rate	0.31	0.42	0.18	0.35	0.23	0.39
	Maximum latency of one step	2	2	3	3	2	2
0.6	Total energy consumption	421032	323006	362557	265256	362563	274633
	Miss rate	0.32	0.48	0.27	0.38	0.20	0.40
	Maximum latency of one step	2	2	3	3	2	2
0.7	Total energy consumption	238206	189220	169880	116927	218845	145078
	Miss rate	0.26	0.40	0.18	0.24	0.24	0.31
	Maximum latency of one step	2	2	3	3	2	2
0.8	Total energy consumption	225967	206393	138040	142272	177222	180235
	Miss rate	0.27	0.47	0.15	0.35	0.20	0.45
	Maximum latency of one step	2	2	3	3	2	2
0.9	Total energy consumption	161097	131693	73106	71120	122083	108685
	Miss rate	0.24	0.37	0.08	0.17	0.17	0.31
	Maximum latency of one step	2	2	3	3	2	2

Table 5.16. Evaluation of the performance metrics with different policies and α in the problem 3.

α	Performance metrics	Policy 1		Policy 2		Policy 3	
		Turn on 1 hop neighbors	Turn on 1 hop neighbors and round advance = 1	Turn on 1 or 2 hop neighbors	Turn on 1 or 2 hop neighbors and round advance = 1	Dynamic turn on 1 or 2 hop neighbors	Dynamic turn on 1 or 2 hop neighbors and round advance = 1
0.4	Total energy consumption	989877	695822	618256	540868	804378	606531
	Miss rate	0.34	0.45	0.20	0.35	0.27	0.40
	Maximum latency of one step	2	2	3	3	2	2
0.5	Total energy consumption	553273	386636	279337	279623	416486	316725
	Miss rate	0.32	0.42	0.14	0.30	0.23	0.36
	Maximum latency of one step	2	2	3	3	2	2
0.6	Total energy consumption	391615	303401	205832	199468	274383	218216
	Miss rate	0.29	0.44	0.13	0.27	0.19	0.30
	Maximum latency of one step	2	2	3	3	2	2
0.7	Total energy consumption	218595	189223	101310	107525	169870	173275
	Miss rate	0.24	0.40	0.09	0.21	0.18	0.39
	Maximum latency of one step	2	2	3	3	2	2
0.8	Total energy consumption	196575	167193	79270	85884	147824	152011
	Miss rate	0.23	0.38	0.07	0.19	0.16	0.36
	Maximum latency of one step	2	2	3	3	2	2
0.9	Total energy consumption	121880	102273	43733	42933	112281	89875
	Miss rate	0.17	0.27	0.03	0.07	0.15	0.24
	Maximum latency of one step	2	2	3	3	2	2

Figure 5.17 shows a comparison of the total energy consumption with different *α* and different policies in the problem 3. Figure 5.18 shows a comparison of the miss rate with different *α* and different policies in the problem 3. Figure 5.19 shows a comparison of the total energy consumption with different round advance and different policies in the problem 3. Figure 5.20 shows a comparison of the miss rate with different round advance and different policies in the problem 3. Figure 5.21 shows a comparison of the total energy consumption with different *α* and *h_ub*. Figure 5.22 shows a comparison of the miss rate with different *α* and *h_ub*.

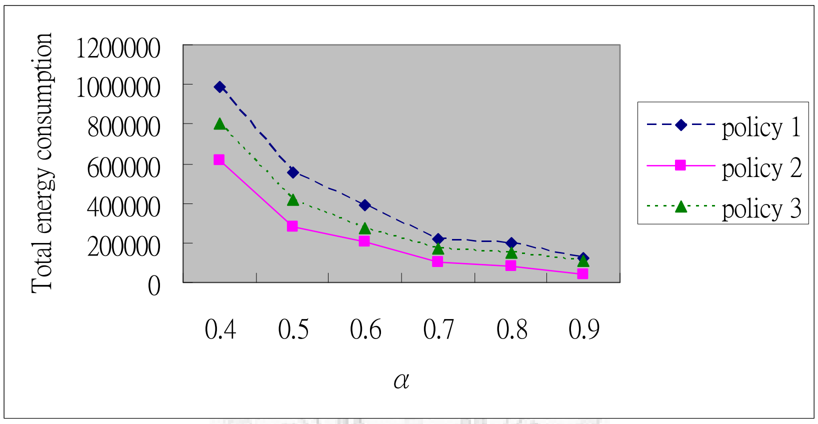


Figure 5.17. A comparison of the total energy consumption with different *α* and different policies without round advance in the problem 3.

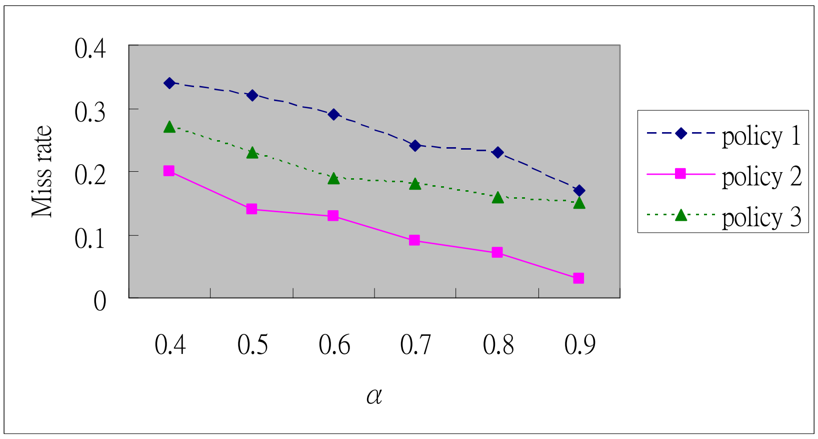


Figure 5.18. A comparison of the miss rate with different *α* and different policies without round advance in the problem 3.

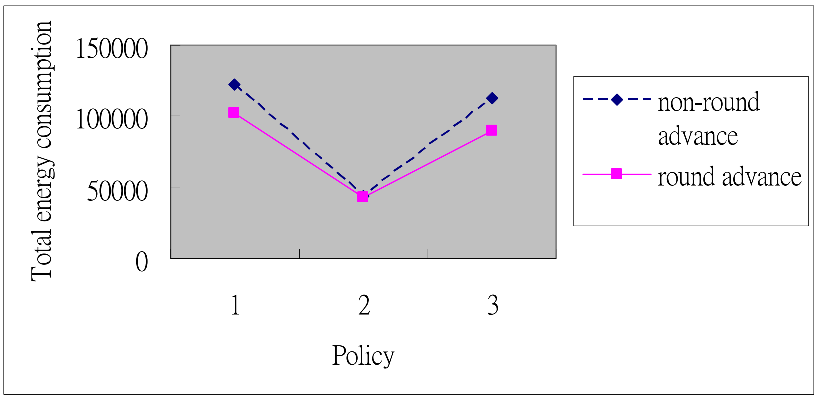


Figure 5.19. A comparison of the total energy consumption with different round advance and different policies in the problem 3. ($\alpha = 0.9$)

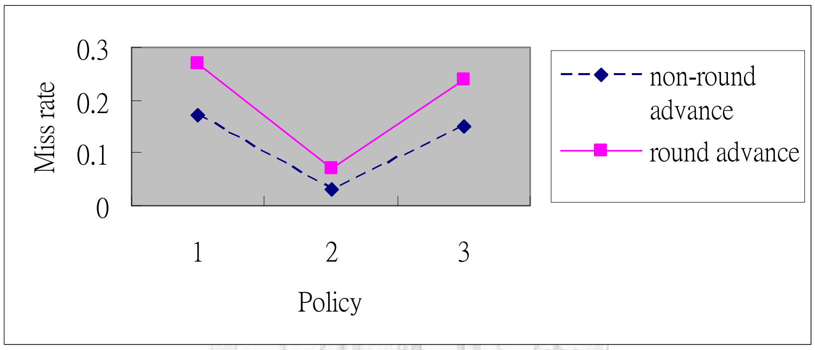


Figure 5.20. A comparison of the miss rate with different round advance and different policies in the problem 3. ($\alpha = 0.9$)

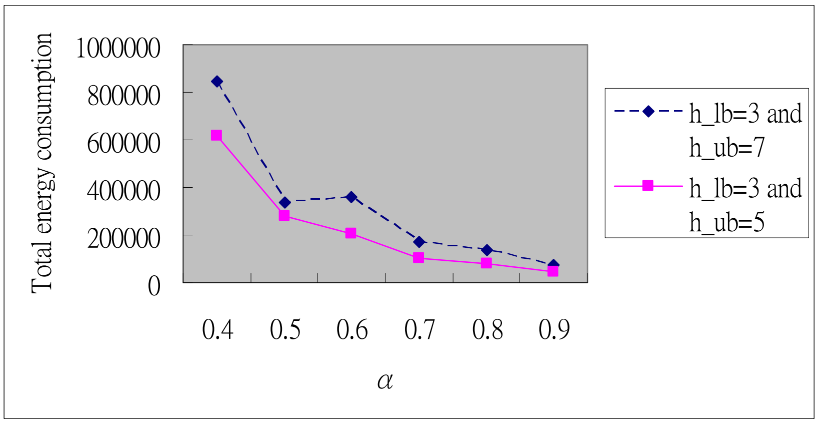


Figure 5.21. A comparison of the total energy consumption with different α and h_ub.

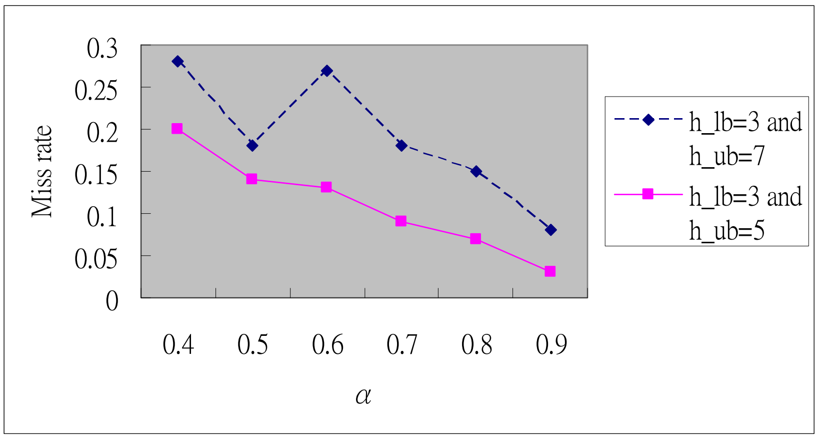


Figure 5.22. A comparison of the miss rate with different *α* and *h_ub*.

5.2.5.3 Discussion

The experiment results show that the policy 2 is better than policy 1 and policy 3, as shown in Figure 5.17 and Figure 5.18. Policy 2 is more energy saving and lower miss rate than policies 1 and 3. Because policy 2 eventually turns on 1 hop and 2 hops neighbor sensor nodes before turns on all sensor nodes. In addition, large *α* value has more energy saving and lower miss rate than small *α* value. Because large *α* value results the object moving near linearly. In the total energy consumption, we can see the round advance case is better than non-round advance case, as shown in Figure 5.19. In the miss rate, we can see that the non-round advance case is better than round advance case, as shown in Figure 5.20. Therefore, users choose round advance approach if the energy consumption is a critical concern. On the contrary, users choose non-round advance approach if the miss rate is a critical concern. The large *h_ub* is not suitable, because large *h_ub* results in high energy consumption and miss rate, as shown in Figure 5.21 and Figure 5.22.

5.2.6 Concluding Remarks

This study proposes a dynamic prediction-based algorithm in wireless sensor networks. To our best knowledge, the proposed modified Viterbi algorithm is truly novel and it has not been discussed in previous researches. This study first formulates the problem as a multiple criteria decision problem with 3 goals, energy consumption, miss rate, and latency, in the combined objective function.

We introduce round advance approach, which effectively help users to choose proper method according to the concern of energy consumption or miss rate.

Chapter 6 Conclusions and Future Work

6.1 Conclusions

In this dissertation, we have proposed five algorithms, BMAFS, BMAMS, LDA, NLDA, and TOTA, to solve optimization problems based on Lagrangean relaxation method, system simulation, and heuristic approaches. In addition, we develop one POTA based on modified Viterbi algorithm to solve prediction-based object tracking problem.

We propose two algorithms, BMAFS and BMAMS, to support boundary monitoring services. The BMAFS is to construct boundary monitoring for grouping capabilities. The experiment results show that the proposed BMAFS can improve the percentage of energy consumption from 11% to 81% while compared with SA1 and SA2. It also achieves the optimal solution since the gaps are 0% in the test problems. The BMAMS is to address the problem of boundary nodes relocation. It can move previously deployed sensors to cover uncovered check points due to nodes failed or nodes battery exhausted. The mechanism can further prolong the system lifetime. Compared with BMAFS, the proposed BMAMS can improve the lifetime of boundary monitoring services from 0% to 35%. The experiment results show that the proposed BMAMS gets effectiveness in the boundary monitoring services for grouping capabilities.

The proposed LDA and NLDA are to support in-depth defense services. The LDA is to construct layered defense for wireless sensor networks of grouping capabilities. The experiment results show that the proposed BMAFS can improve the percentage of energy consumption from 0% to 18% while compared with SA1 and SA2. The NLDA is to construct non-layered defense of supporting different types of intruders for grouping capabilities. The NLDA can prolong the system lifetime and provide lead time alarms. The experiment results show that the proposed NLDA can improve system lifetime since the improvement ratio is from 59% to 136% while compared with simple algorithm.

At last, in the TOTA and POTA, we use the TOTA to support tree-based object tracking services. The experiment results show that the proposed heuristic algorithm can improve the percentage of energy consumption from 8% to 44% while compared with shortest path tree algorithm. It also achieves the near optimal solution since the gaps are only from 8% to 18%. Furthermore, the algorithm is efficient and scalable in terms of the running time. The POTA is to construct a prediction-based algorithm for object tracking. Such prediction-based can minimize the number of nodes participating in the tracking activities, minimize out of tracking probability, and maximize the accuracy of object predicted position in the tracking activities. The POTA can prolong the system lifetime.

The experiment results show that all six algorithms can support object monitoring and tracking services efficiently. They also support quality of services and prolong the system lifetime in object monitoring and tracking of wireless sensor networks.

6.2 Future Work

In the future, the subsequent studies can be conducted as follows. The grouping algorithm by Voronoi diagram is described in Section 6.2.1. The multiple sinks for tree-based object tracking is presented in Section 6.2.2.

6.2.1 Grouping Algorithm by Voronoi Diagram

The V-points include intersection points of Voronoi diagram, intersection points of border, and corner points. The monitoring region is fully covered if all V-points are covered by awaked sensors.

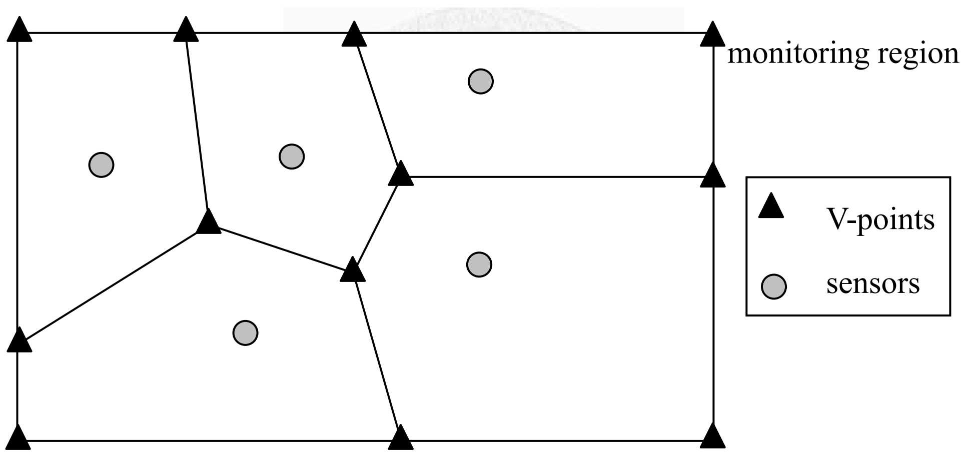


Figure 6.1. An example of V-points.

The notations used to model the problem are listed as follows.

Table 6.1. Notations of the given parameters in grouping algorithm by Voronoi diagram.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
V	The set of the all candidate v-points in the monitoring region.
C_s	The initial energy level of each sensor node *s*.

E_{m}	The energy consumption for sensor nodes to sense data.
R	The total number of rounds.
P_r	The set of all candidate v-points in the run *r*.

Table 6.2. Notation of the indicate parameter in grouping algorithm by Voronoi diagram.

Indicate Parameter	
Notation	Description
b_{sv}	The indicator function which is 1 if the v-point *v* is in the coverage of the sensor node *s* and 0 otherwise.

Table 6.3. Notations of the decision variables in grouping algorithm by Voronoi diagram.

Decision Variables	
Notation	Description
y_{vr}	1 if v-point *v* at least is covered by one awake sensor in the round *r*, and 0 otherwise.
z_r	1 if full coverage v-points in the round *r*, and 0 otherwise.
π_{sr}	1 if sensor *s* is awake in the run *r*; otherwise is equal to 0.

Problem (IP):

Objective function:

$$Z_{IP} = \max \sum_{\forall r \in R} z_r \qquad \text{(IP)}$$

subject to:

The full coverage v-points constraints

$$y_{vr} \le \sum_{s \in S} b_{sv} \pi_{sr} \qquad \forall v \in V, r \in R \qquad (1)$$

$$z_r \le \frac{\sum_{v \in V} y_{vr}}{|P_r|} \qquad \forall r \in R \qquad (2)$$

The battery capacity constraint

$$\sum_{r \in R} \pi_{sr} E_m \quad \le \quad C_s \qquad \forall s \in S \qquad (3)$$

The integer constraints

$$\pi_{sr} \quad = \quad 0 \text{ or } 1 \qquad \forall s \in S, r \in R \qquad (4)$$

$$y_{ar} \quad = \quad 0 \text{ or } 1 \qquad \forall a \in A, r \in R \qquad (5)$$

$$z_r \quad = \quad 0 \text{ or } 1 \qquad \forall r \in R. \qquad (6)$$

The objective function is to maximize the system lifetime of the given sensor network. The lifetime is defined as the total number of rounds.

Constraints (1)-(2): Full coverage V-points constraint in each round *r*.
Constraint (3): For each sensor node *s*, the total sensing consumption can not exceed its initial energy level.
Constraints (4)-(6): The integer constraints for decision variables π_{st}, y_{ar}, and z_r.

6.2.2 Multiple Sinks for Tree-based Object Tracking

It is planned to further take the load balancing and residual energy capacity into consideration to prevent the "hot spot" failing the object tracking tree. In addition, it is intended to extend the model to multiple sinks of object tracking tree in near future [44][45], since the multiple sinks can provide load balancing and fault tolerance.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Wireless Sensor Network,” *IEEE Communication Magazine*, pp. 102-114, August 2002.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Network: A Survey,” *Computer Networks*, vol. 38, pp. 393-422, 2002.

[3] Z. Guo, M.C. Zhou, and G. Jiang, “Adaptive Sensor Placement and Boundary Estimation for Monitoring Mass Objects,” *Part B, IEEE Transactions on Systems, Man, and Cybernetics,* vol. 38, pp. 222-232, 2008.

[4] C. Zhang, Y. Zhang, and Y. Fang, “Detecting Coverage Boundary Nodes in Wireless Sensor Networks,” *IEEE International Conference on Networking, Sensing and Control*, pp. 868-873, 2006.

[5] J. Lian, L. Chen, K. Naik, Y. Liu, and G.B. Agnew, “Gradient Boundary Detection for Time Series Snapshot Construction in Sensor Networks,” *IEEE Transactions on Parallel and Distributed Systems,* 2007.

[6] Y. Wang, X. Wang, B. Xie, D. Wang, and D.P. Agrawal, “Intrusion Detection in Homogeneous and Heterogeneous Wireless Sensor Networks,” *IEEE Transactions on Mobile Computing,* vol. 7, pp. 698-711, 2008.

[7] G. Li, J. He, and Y. Fu, “A Group-Based Intrusion Detection Scheme in Wireless Sensor Networks,” *The 3rd International Conference on Grid and Pervasive Computing Workshop*, pp. 286-291, 2008.

[8] G. Li, J. He, and Y. Fu, “A Distributed Intrusion Detection Scheme for Wireless Sensor Networks,” *28th International Conference on Distributed Computing Systems Workshops*, pp. 309-314, 2008.

[9] Y. Wang, Y.K. Leow, and J. Yin, “Is Straight-line Path Always the Best for

Intrusion Detection in Wireless Sensor Networks," *International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 564-571, 2009.

[10] H.T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," *IEEE Wireless Communications and Networking Conference*, pp. 1954–1961, 2003.

[11] C.Y. Lin and Y.C. Tseng, "Structures for In-network Moving Object Tracking in Wireless Sensor Networks," *IEEE First International Conference on Broadband Networks*," pp. 718–727, 2004.

[12] C.Y. Lin, W.C. Peng, and Y.C. Tseng, "Efficient In-Network Moving Object Tracking in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 1044-1056, 2006.

[13] Y. Xu and W.C. Lee, "On Localized Prediction for Power Efficient Object Tracking in Sensor Networks," *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops*, pp. 434–439, 2003.

[14] Y. Xu, J. Winter, and W.C. Lee, "Prediction-based Strategies for Energy Saving in Object Tracking Sensor Networks," *IEEE International Conference on Mobile Data Management*, pp. 346–357, 2004.

[15] Y. Xu, J. Winter, and W.C. Lee, "Dual Prediction-based Reporting for Object Tracking Sensor Networks," *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 154–163, 2004.

[16] Y.F. Wong, W.K.G. Seah, L.H. Ngoh, and W. C. Wong, "Sensor Traffic Patterns in Target Tracking Networks," *Wireless Communications and Networking Conference*, *WCNC*, pp. 4123-4126, 2007.

[17] Y.C. Wang and Y.C. Tseng, "Distributed Deployment Schemes for Mobile Wireless Sensor Networks to Ensure Multi-level Coverage", *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1280–1294, 2008.

[18] K. Mechitov, S. Sundresh, G. Agha and Y. Kwon. “Cooperative Tracking with Binary-Detection Sensor Networks,” *University of Illinois at Urbana-Champaign, Technical Report*, UIUCDCS-R-2003-2379, 2003.

[19] T. Camp, J. Boleng, and V. Davies, “A Survey of Mobility Models for Ad Hoc Network Research,” *Wireless Communication and Mobile Computing*, pp. 483-502, 2002.

[20] S. P. Manh Tran and T. A. Yang, “Evaluations of Target Tracking in Wireless Sensor Networks,” *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pp. 97-101, 2006.

[21] “The types of power consumption of MICAz 2.4GHz,” http://www.xbow.com/Products/productdetails.aspx?sid=164.

[22] S. P. M. Tran and T. A. Yang, “OCO: Optimized Communication & Organization for Target Tracking in Wireless Sensor Networks,” *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pp. 428-435, 2006.

[23] P. K. Sahoo, K.-Y. Hsieh, and J.-P. Sheu, “Boundary Node Selection and Target Detection in Wireless Sensor Network,” *IFIP International Conference on Wireless and Optical Communications Networks*, pp. 1-5, 2007.

[24] P.L. Chiu and F.Y.S. Lin, “Sensor Deployment Algorithms for Target Positioning Services,” *Doctoral Thesis*, *Graduate Institute of Information Management of National Taiwan Universit*y, 2007.

[25] S. Bhatti and J. Xu, “Survey of Target Tracking Protocols Using Wireless Sensor Network,” *Fifth International Conference on Wireless and Mobile Communications*, pp. 110–115, 2009.

[26] F.Y.S. Lin, H.H. Yen, and S.P. Lin, “A Novel Energy-Efficient MAC Aware Data Aggregation Routing in Wireless Sensor Networks,” *Sensors*, pp. 1295-2147, March, 2009.

[27] F.Y.S. Lin, H.H. Yen, and S.P. Lin, "Delay QoS and MAC Aware Energy-Efficient Data-Aggregation Routing in Wireless Sensor Networks," *Sensors*, vol. 9, pp. 7711-7732, October, 2009.

[28] Y.F. Wen, F.Y.S. Lin, and W.C. Kuo, "A Tree-based Energy-efficient Algorithm for Data-Centric Wireless Sensor Networks," *21st International Conference on Advanced Networking and Applications*, pp. 202–209, 2007.

[29] C.T. Lee and F.Y.S. Lin, "An Energy-Efficient Lagrangean Relaxation-based Object Tracking Algorithm in Wireless Sensor Networks," *20th International Conference on Information Management* (*ICIM*), 2009.

[30] C.T. Lee, F.Y.S. Lin, and Y.F. Wen, "An Efficient Object Tracking Algorithm in Wireless Sensor Networks," *9th Joint Conference on Information Sciences* (*JCIS*) *(9th International Conference on. Computer Science and Informatics)*, 2006.

[31] B.H. Liu, W.C. Ke, C.H. Tsai, and M.J. Tsai, "Constructing a Message-Pruning Tree with Minimum Cost for Tracking Moving Objects in Wireless Sensor Networks Is NP-Complete and an Enhanced Data Aggregation Structure," *IEEE Transactions on Computers*, vol. 57, pp. 849-863, 2008.

[32] M.L. Fisher, "The Lagrangean Relaxation Method for Solving Integer Programming Problem," *Management Science*, vol. 27, pp. 1-18, 1981.

[33] M.L. Fisher, "An Applications Oriented Guide to Lagrangian Relaxation," *Interfaces*, vol. 15, pp. 10-21, 1985.

[34] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, "Network Flows: Theory, Algorithm, and Applications," chapter 16, *Prentice Hall*, 1993.

[35] S. Meguerdichian and M. Potkonjak, "Low Power 0/1 Coverage and Scheduling Techniques in Sensor Networks," *UCLA Technical Reports* 030001, January 2003.

[36] G. Wang, G. Cao, T. L. Porta, and W. Zhang, "Sensor Relocation in Mobile

Sensor Networks," *IEEE INFOCOM*, vol. 4, pp. 2302-2312, March 2005.

[37] Y. Yoo and D.P. Agrawal, "Mobile Sensor Relocation to Prolong the Lifetime of Wireless Sensor Networks," *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pp. 193-197, 2008.

[38] R. Perry, A. Vaddiraju, K. Buckley, "Multitarget List Viterbi Tracking Algorithm," *Department of Electrical and Computer Engineering, Villanova University*.

[39] Y.F. Wong, et al., "Sensor Traffic Patterns in Target Tracking Networks," *Wireless Communications and Networking Conference*, WCNC, pp. 4123-4126 2007.

[40] Y. Zhu and A. Shareef, "Comparisons of Three Kalman Filter Tracking Algorithms in Sensor Network," *Proceedings of the 2006 International Workshop on Networking*, *Architecture*, *and Storages*, pp. 61-62, 2006.

[41] P. Zeng, et al., "Bounding the Lifetime of Target Tracking Sensor Networks," *IEEE International Conference on ICC*, pp. 3444-3449, 2006.

[42] O. Dousse, et al., "Delay of Intrusion Detection in Wireless Sensor Networks," *The Proceedings of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Florence, Italy, 2006.

[43] Y. Li and Y.H. Liu, "Energy Saving Target Tracking Using Mobile Sensor Networks," *IEEE International Conference on Robotics and Automation*, pp. 3653-3658, 2007.

[44] C.Y. Lin, Y.C. Tseng, and T.H. Lai, "Message-Efficient In-Network Location Management in a Multi-sink Wireless Sensor Network," *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pp. 496-505, 2006.

[45] F.Y.S. Lin and Y.F. Wen, "Multi-sink Data Aggregation Routing and Scheduling

with Dynamic Radii in WSNs," *IEEE Communications Letters*, vol. 10, pp. 692-694, 2006.

[46] Y. Yang and M. Cardei, "Movement-assisted Sensor Redeployment Scheme for Network Lifetime Increase," *The Proceedings of the 10th ACM Symposium on Modeling*, *Analysis*, *and Simulation of Wireless and Mobile Systems*, Chania, Crete Island, Greece, 2007.

[47] W. Liang and Y. Liu, "Online Data Gathering for Maximizing Network Lifetime in Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 2-11, 2007.

[48] C. Schurgers, et al., "Optimizing Sensor Networks in The Energy-latency-density Design Space," *IEEE Transactions on Mobile Computing*, vol. 1, pp. 70-80, 2002.

[49] E. L. Lloyd and G. Xue, "Relay Node Placement in Wireless Sensor Networks," *IEEE Transactions on Computers*, vol. 56, pp. 134-138, 2007.

[50] A.S. Chhetri, et al., "Sensor Resource Allocation for Tracking Using Outer Approximation," *IEEE Signal Processing Letters*, vol. 14, pp. 213-216, 2007.

[51] J. Li and X. Liu, "Survivability for Wireless Sensor Network Model, Evaluation and Experiment," *Fifth International Joint Conference on INC, IMS and IDC*, pp. 1513-1516, 2009.

[52] H. Yang and B. Sikdar, "A Protocol for Tracking Mobile Targets Using Sensor Networks," *Proceedings of IEEE Workshop on Sensor Network Protocols and Application*, pp. 71-81, 2003.

[53] H. Perros, "Computer Simulation Techniques: The Definitive Introduction," chapter 1, 2009.

[54] F.S. Hillier and G.J. Lieberman, "Introduction to Operations Research," fifth edition, chapter 23, *McGraw-Hill*, 1990.

[55] S. Megerian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, "Worst and Best-Case Coverage in Sensor Networks," *IEEE Transactions on Mobile Computing*, pp. 84-92, 2005.

[56] "The specification of RPS-326," http://migatron.com/products/rps-326/rps-326.pdf.

[57] M. Lu, J. Wu, M. Cardei, and M. Li, "Energy-efficient Connected Coverage of Discrete Targets in Wireless Sensor Networks," *Proceedings of 3rd International Conference Networking and Mobile Computing*, *ICCNMC*, *LNCS*, vol. 3619. Springer, Heidelberg, pp. 43-52, 2005.

[58] A. Dhawan, C.T. Vu, A. Zelikovsky, Y. Li, and S. K. Prasad, "Maximum Lifetime of Sensor Networks with Adjustable Sensing Range," *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing*, 2006.

[59] C.T. Lee and F.Y.S. Lin, "Boundary Monitoring Algorithms for Wireless Sensor Networks of Grouping Capabilities," *IEEE International Conference on Wireless Communications, Networking and Information Security* (*WCNIS*), Vol. 2, pp. 461-467, 2010.

[60] F.Y.S. Lin, C.T. Lee, and Y.Y. Hsu, "An Energy-Efficient Algorithm for Object Tracking in Wireless Sensor Networks," *IEEE International Conference on Wireless Communications, Networking and Information Security* (*WCNIS*), Vol. 2, pp. 424-430, 2010.

[61] C.T. Lee, F.Y.S. Lin, and Y.S. Wang, "Maximizing the Lifetime of Layered Defense in Wireless Sensor Networks," *IEEE Asia Pacific Wireless Communications Symposium* (*APWCS*), 2010.

[62] C. Mascolo and M. Musolesi, "SCAR: Contextaware Adaptive Routing in Delay Tolerant Mobile Sensor Networks," *Proceedings of the International Conference on Wireless Communications and Mobile Computing* (*IWCMC*), 2006.

[63] B. Amutha and M. Ponnavaikko, "Energy Efficient Hidden Markov Model Based Target Tracking Mechanism in Wireless Sensor Networks," *Journal of Computer Science*, vol. 5, no. 12, pp.1085-1093, 2009.

[64] J. Cartigny, D. Simplot, and I. Stojmenovic, "Localized Minimum-energy Broadcasting in Ad-hoc Networks," *IEEE Infocom*, pp. 2210-2217, 2003.

[65] R. L. Keeney and H. Raiffa, "Decisions with Multiple Objectives: Preference and Value Tradeoffs," *Wiley*, 1976.

[66] M. Vemula, M. F. Bugallo, and P. M. Djuric, "Target Tracking in a Two-tiered Hierarchical Sensor Network," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 969–972, 2006.

[67] J.B.D. Cabrera, C. Gutierrez, and R.K. Mehra, "Infrastructures and Algorithms for Distributed Anomaly-based Intrusion Detection in Mobile Ad-hoc Networks," *IEEE Conference on Military Communications* (*MILCOM*), pp. 1831-1837, 2005.

[68] W. Chen, J. C. Hou, and L. Sha, "Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks," *International Conference on Network Protocols*, pp. 284-294, 2003.

[69] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. "Coverage Problems in Wireless Ad-hoc Sensor Networks," *INFOCOM*, pp. 1380-1387, 2001.

[70] A. Savvides, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Location Discovery in Ad-hoc Wireless Sensor Networks," *unpublished*, UCLA EE and CS Departments.

[71] D. P. Bertsekas, "Dynamic Programming and Optimal Control," chapter 6, *Athena Scientific*, 2010.

[72] A. Wang, S-H. Cho, C.G. Sodini, and A.P. Chandrakasan, “Energy-efficient Modulation and MAC for Asymmetric Microsensor Systems,” *Proceedings of ISLPED*, pp 106-111, 2001.

[73] V. Tsiatsis, S. Zimbeck, and M. Srivastava, “Architectural Strategies for Energy Efficient Packet Forwarding in Wireless Sensor Networks,” *Proceedings of ISLPED*, pp. 92-95, 2001.

[74] I. Demirkol, C. Ersoy, F. Alagoz, “MAC Protocols for Wireless Sensor Networks: A Survey,” *IEEE Communication Magazine*, vol. 44, no. 4, pp. 115–121, 2006.

[75] J. N. Al-Karaki and A. E. Kamal. “Routing Techniques in Wireless Sensor Net works: a Survey,” *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 6-28, 2004.

[76] J. Cheng, M. Xie, R. Chen, and F. Roberts, “A Mobile Sensor Network for the Surveillance of Nuclear Materials in Metropolitan Areas,” *DIMACS Technical Report*, Rutgers University, 2009.

[77] R. C. Chen, C. F. Hsieh, Y. F. Huang, “An Isolation Intrusion Detection System for Hierarchical Wireless Sensor Networks,” *Journal of Network*s, vol 5, no 3, 2010.

[78] Y. C. Wang and Y. C. Tseng, “Intentional mobility in wireless sensor networks,” chapter 1, *Wireless Networks: Research, Technology, and Applications*, 2009.

Appendix A: List of Notations

A.1 Notations of Chapter 3

The notations used to model the boundary monitoring for fixed sensors problem are listed as follows.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
A	Index set of the service check points in the monitoring region boundary.
C_s	The initial energy level of each sensor node *s*.
E_s	The energy consumption for aware sensor node *s* to sense data in each round.
R	The upper bound number of rounds.
b_{sa}	The indicator function which is 1 if the check point *a* is in the sensing range of the sensor node *s*, and 0 otherwise.
Decision Variables	
Notation	Description
π_{sr}	1 if sensor *s* is awake in the round *r*, and 0 otherwise.
y_{ar}	1 if check point *a* at least is covered by one awake sensor in the round *r*, and 0 otherwise.
z_r	1 if full coverage boundary check points in the round *r*, and 0 otherwise.

The notations used to model the boundary monitoring for mobile sensors problem are listed as follows.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
A	Index set of the service check points in the monitoring region boundary.
d_{sa}	Euclidean distance for sensor node *s* moves to cover uncovered service check point *a*, $s \in S$, $a \in A$.
$e(d_{sa})$	Energy consumption for sensor node *s* moves to cover uncovered service check point *a*.
E_s	The energy level of each sensor node *s*, $s \in S$.

E_{m}	The energy consumption for sensors node to sense data in each round.
Indicator Parameters	
Notation	Description
ρ_{sa}	The indicator function which is 1 if the check point *a* is in the coverage of the non-moved sensor node *s* and 0 otherwise.
σ_{sa}	The indicator function which is 1 if the check point *a* is in the coverage of the moved sensor node *s* and 0 otherwise.
Decision Variables	
Notation	Description
ι_s	1 if sensor node *s* does not move, and 0 otherwise. $s \in S$.
ξ_{sa}	1 if sensor node *s* moving to cover uncovered check point *a*, and 0 otherwise. $a \in A$.
π_{sr}	1 if sensor *s* is awake in the round *r*; otherwise is equal to 0.
y_{ar}	1 if check point *a* at least is covered by one awake sensor in the round *r*, and 0 otherwise.
z_r	1 if full coverage check points in the round *r*, and 0 otherwise.

A.2 Notations of Chapter 4

The notations used to model the layered defense problem are listed as follows.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
A_j	Index set of the service check points of layer *j* in the layered defense.
C_s	The initial energy level of sensor node *s*.
E_m	The energy consumption for sensor nodes to sense data.
b_{saj}	The indicator function which is 1 if the check point *a* is in the radius of the sensor node *s* on layer *j*, and 0 otherwise.
R	The upper bound number of rounds.
J	The total number of layers.
d_j	The defense rate of layer *j*.
Q	The total defense rate.
m_j	The distance of early warning of layer *j*.
P	The detectability of system.
Decision Variables	
Notation	Description
π_{sr}	1 if sensor *s* is awake in the round *r*, and 0 otherwise.
y_{arj}	1 if check point *a* at least is covered by one awake sensor on layer *j* in the round *r*, and 0 otherwise.
z_r	1 if satisfy total defense rate in the round *r*, and 0 otherwise.

The notations used to model the non-layered defense problem are listed as follows.

Controlled parameters	
Notation	Description
M_r	The total evaluation frequency for all intruder categories in round r.
η	False positive rate.
τ	False negative rate.
Given Parameters	
Notation	Description
K	The total intruder categories.
T_{kr}	Total evaluation frequency of each intruder type in round r (where $k\in K$, $r\in R$).
F	All possible defense strategies.
$\vec{I}_k$	The strategies of an intruder, comprising his motion and intrusive angle.
$G_{kjr}(\vec{F},\vec{I}_k)$	1 if intruder j of the k^{th} intruder category has alarm raised under $\vec{F}$ defense strategies and $\vec{I}_k$ intruder strategies in round r with no false positive, and 0 otherwise (where $k\in K$).
$H_{kjr}(\vec{F},\vec{I}_k)$	1 if the intruder j of the k^{th} intruder category has not alarm raised under $\vec{F}$ defense strategies and $\vec{I}_k$ intruder strategies in round r with false negative, and 0 otherwise (where $k\in K$).
S	The set of all sensor nodes.
C_s	The initial energy level of sensor node s.
E_m	The energy consumption for sensor nodes to sense data.
R	The upper bound number of rounds.
D	The defense rate.
L	The distance of early warning.
W	The early warning rate.
C	Core field: $x_c^2 + y_c^2 \le h^2$, (x_c, y_c) is coordinate of core and h is radius of core.
N	The set of candidate location (x, y) if intruder be detected.
Decision Variables	
Notation	Description
π_{sr}	1 if sensor s is awake in the round r; and 0 otherwise.

z_r	1 if satisfy total defense rate and early warning rate in the round *r*, and 0 otherwise.
$\vec{F}$	The strategies of defender that sensor *s* is awake in the round *r*.
$u_{(x,y)}^{kjr}$	1 if the intruder *j* of the k^{th} intruder category that Euclidean distance between location (*x*,*y*) and *core* greater than or equal to *L* in round *r*; and 0 otherwise.

A.3 Notations of Chapter 5

The notations used to model the tree-based object tracking problem are listed as follows.

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
Γ	The set of all communication nodes, including sink node.
o	Artificial node outside the sensor field.
Θ	The set of the object moving frequency from *x* to *y*, $\forall x, y \in S \cup \{o\}$, $x \neq y$.
Λ	The set of all links, $(i, j) \in \Lambda$, *i*≠*j*.
Ω	The set of transmission costs $\omega_{(i,j)}$ associated with link (i, j).
Φ_s	The set of all candidate paths φ between a pair of nodes, *s and* sink, $\forall s \in S$.
Indicate Parameter	
Notation	Description
$\delta_{\varphi(i,j)}$	The value of indicator function is 1 if link (i, j) is on path φ, and 0 otherwise.
Decision Variables	
Notation	Description
$x_{s\varphi}$	1 if the sensor node *s* uses the path φ to reach the sink node, and 0 otherwise.
$\zeta_{(i,j)}^{s}$	1 if the sensor node *s* uses the link (i, j) to reach the sink node, and 0 otherwise.
$\upsilon_{(i,j)}^{xy}$	1 if $\zeta_{(i,j)}^{x} = 0 \cap \zeta_{(i,j)}^{y} = 1$ (reporting object's location uses the link (*i*,*j*) when object moves from sensor *x* to sensor *y*), and 0 otherwise, $x \neq y$.

Appendix B: Publications

Journal papers:

1. Y.F. Wen, F.Y.S. Lin, Y.C. Tzeng, and **C.T. Lee**, “Backhaul Assignment and Routing Algorithms with End-to-End QoS Constraints for Wireless Mesh Networks,” *Wireless Personal Communications*, Vol. 53, No. 2, pp. 211-233, April, 2010. (**SCI**/**EI**)

Book chapter:

1. **C.T. Lee** and F.Y.S. Lin, “An Efficient Circuit-Switched Broadcasting in Star Graph,” *Lecture Notes in Computer Science* (*LNCS*), No. 6082, pp.141-145, 2010. (**EI**/**DBLP**)

Conference papers:

1. **C.T. Lee** and F.Y.S. Lin, “Boundary Monitoring Algorithms for Wireless Sensor Networks of Grouping Capabilities,” *IEEE International Conference on Wireless Communications, Networking and Information Security* (*WCNIS*), Vol. 2, pp. 461-467, 2010. (**EI**/**ISTP**)
2. F.Y.S. Lin, **C.T. Lee**, and Y.Y. Hsu, “An Energy-Efficient Algorithm for Object Tracking in Wireless Sensor Networks,” *IEEE International Conference on Wireless Communications, Networking and Information Security* (*WCNIS*), Vol. 2, pp. 424-430, 2010. (**EI**/**ISTP**)
3. F.Y.S. Lin, **C.T. Lee**, and L.Y. Lin, “An Efficient Time Slot Allocation Algorithm in Wireless Networks,” *IEEE International Conference on Wireless Communications, Networking and Information Security* (*WCNIS*), Vol. 2, pp. 322-328, 2010. (**EI**/**ISTP**)

4. **<u>C.T. Lee</u>**, F.Y.S. Lin, and Y.S. Wang, "Maximizing the Lifetime of Layered Defense in Wireless Sensor Networks," *IEEE Asia Pacific Wireless Communications Symposium* (*APWCS*)*,* 2010. (**EI**)
5. **<u>C.T. Lee</u>** and F.Y.S. Lin, "An Efficient Circuit-Switched Broadcasting in Star Graph," *The 10th International Conference on Algorithms and Architectures for Parallel Processing* (*ICA3PP*), 2010. (**EI**/**DBLP**)
6. **<u>C.T. Lee</u>** and F.Y.S. Lin, "An Energy-Efficient Lagrangean Relaxation-based Object Tracking Algorithm in Wireless Sensor Networks," *The 20th International Conference on Information Management* (*ICIM*), pp. 282-291, 2009.
7. F.Y.S. Lin, L.Y. Lin, and **<u>C.T. Lee</u>**, "A Near-optimal Time Slot Allocation Algorithm for Wireless Networks that Support Multiple Classes of Traffic," *The 19th International Conference on Information Management* (*ICIM*), 2008.
8. **<u>C.T. Lee</u>**, F.Y.S. Lin, and Y.F. Wen, "An Efficient Object Tracking Algorithm in Wireless Sensor Networks," *The 9th Joint Conference on Information Sciences* (*JCIS*) *(The 9th International Conference on Computer Science and Informatics)*, pp. 619-625, 2006. (**EI**/**DBLP**)
9. C.F. Lin, F.Y.S. Lin, and **<u>C.T. Lee</u>**, "A Near-optimal Slot Assignment Algorithm for RFID Reader Networks," *The 9th Joint Conference on Information Sciences* (*JCIS*) *(The 9th International Conference on Computer Science and Informatics)*, pp. 856-859, 2006. (**EI**/**DBLP**)