

國立臺灣大學資訊管理研究所

碩士論文

Graduate Institute of Information Management

National Taiwan University

Master Thesis

無線感測網路之低能耗物體追蹤樹建置演算法

An Energy-Efficient Algorithm for Constructing Object
Tracking Trees in Wireless Sensor Networks



指導教授：林永松 博士

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

中華民國九十八年七月

July, 2009



無線感測網路之低能耗物體追蹤樹建置演算法

An Energy-Efficient Algorithm for Constructing Object
Tracking Trees in Wireless Sensor Networks



研究生：許宴毅 撰

中華民國九十八年七月



國立臺灣大學碩士學位論文
口試委員會審定書

無線感測網路之低能耗物體追蹤樹建置演算法

An Energy-Efficient Algorithm for Constructing Object
Tracking Trees in Wireless Sensor Networks

本論文係 許宴毅 君(學號 R96725033)在國立臺灣大學資訊管理學系、所完成之碩士學位論文，於民國 98 年 7 月 25 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：

林永松

林冠廷

傅新彬

馬俊亨

鍾嘉德

所長：

許益坤代



謝誌

兩年的時間過去了，感謝在這段時間內陪伴我及幫助我的家人、朋友、師長們，多虧了大家的支持、鼓勵與教導，讓我有幸能完成這篇論文，完成了人生的另一個階段。

首先感謝我的父親許文彬先生及母親徐美齡女士，感謝您們在我從小到大的求學過程中，給我最大的支持與鼓勵，讓我能毫無後顧之憂的往前邁進。讓我能夠一步一步的往上爬，才能有今日的成就，未來的我會更加努力。在此向你們說聲謝謝。

在學業上最要感謝的就是林永松老師，當每次有學業上的問題時，老師總是會特地安排出時間，讓我們能夠面對面的與老師討論，在每一次的討論中，都能夠在老師身上吸收到不同的知識、激發出一些不同的想法。在待人處事及生活經驗上，老師也提供了我們不少人生上的寶貴經驗，讓我們深感受用無窮。在此衷心感謝林永松老師不辭辛勞的教導。

此外，要感謝實驗室的所有成員。首先要感謝政達學長在論文上給予我莫大的幫助，從一開始題目的訂定、到最後實驗的結果，都多虧了政達學長的教導與照顧，如果沒有政達學長一路上的陪伴與幫忙，此篇論文必定沒有辦法如此順利的誕生。永遠不會忘記，到學長家閉關修練的那兩天，感謝學長願意讓我到你家打擾，而且還辛苦的陪我研究了兩天、更不好意思的是還接受學長的招待，這段期間內，常常麻煩學長，在此真的要大聲的說聲：不好意思，麻煩了，謝謝學長！而一起度過兩年研究生生活的友仁、睿斌、猷順、冠璋、培維、竣章，在這兩年中，OP 七俠共同創造了許多歡笑、闖過了層層的難關，很高興最後大家能夠一起達到這個里程碑，這兩年的點點滴滴、那些歡笑聲，會一直存在我們心中的。

謝謝學弟永斌、怡緯、世昌、耀元在口試當天購買餐點、提供的協助，讓我們的口試得以順利完成，希望在明天的今天，你們也能夠快樂的一起畢業。

最後我要感謝在這兩年過程中，陪伴我的一些好友，家愷、忠翰、俊毅、瑋屏、芳瑩、泰均、緯霖、佩樺、子雋、皓霆等人，謝謝所有給我鼓勵、幫我打氣的人。特別要感謝的是瑋屏，當初沒有你的堅持，今天我也沒有機會進入台大的校門，在這兩年的過程中，所有遇到的難關、感受到的壓力，幸虧有你陪在我身邊、陪我一起度過。很高興我們能夠一起走向人生的另一段旅程，希望未來的我們，能夠一起發光發亮。

許宴毅 謹識

于台灣大學資訊管理研究所

民國九十七年七月



論文摘要

論文題目：無線感測網路之低能耗物體追蹤樹建置演算法

作者：許宴毅

指導教授：林永松 博士

近年來由於感測器的技術與無線通訊的蓬勃發展，使得無線感測網路(Wireless Sensor Networks)已經被廣泛的應用於各領域；但是在硬體上的限制與應用環境的影響，使得感測器在能源的消耗上有著高度的限制性，因此降低感測器於運作中所消耗之能源成了無線感測網路中最熱門的研究議題之一。

本篇論文研究的目的，是希望能夠在任意的網路拓撲中，能夠達到高效率節能(energy-efficient)的物體追蹤(object tracking)；物體追蹤有兩個主要的操作：更新與查詢，現有研究大多僅考慮更新成本，或者於第二階段以查詢成本做調整。本文希望以建立物體追蹤樹的方式，以最小化成本建立該樹，並於建立時同時考量更新成本與查詢成本，將此問題轉化成一個整數規劃問題，利用拉格蘭日鬆弛法，發展出一個啟發式法則的演算法，用以建立最小化成本之物體追蹤樹。

關鍵字：無線感測網路、物體追蹤、最佳化、高效率節能、拉格蘭日鬆弛法



THESIS ABSTRACT

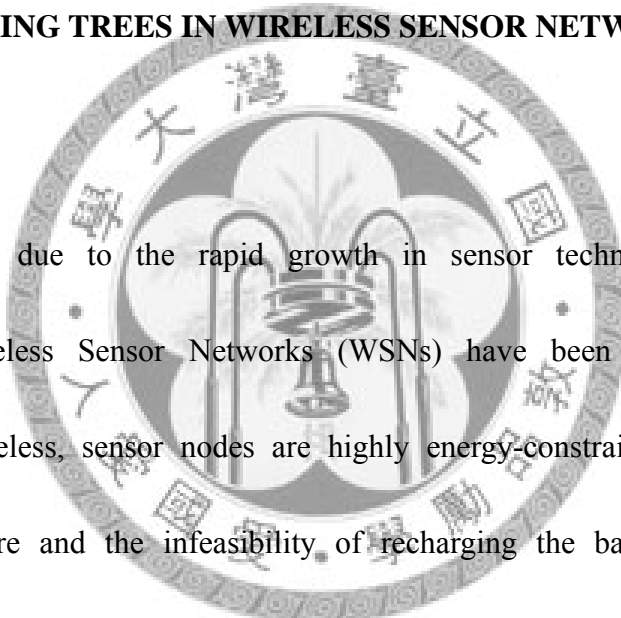
GRADUATE INSTITUTE OF INFORMATION MANAGEMENT

NATIONAL TAIWAN UNIVERSITY

NAME: YEN-YI HSU

ADVISOR: DR. YEONG-SUNG LIN

AN ENERGY-EFFICIENT ALGORITHM FOR CONSTRUCTING OBJECT TRACKING TREES IN WIRELESS SENSOR NETWORKS



In recent years, due to the rapid growth in sensor technology and wireless communication, Wireless Sensor Networks (WSNs) have been applied in various applications. Nevertheless, 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.

In this thesis, our purpose is to achieve energy-efficient object tracking for an arbitrary topology in WSNs. Object tracking typically contains two basic operations: update and query. Most research only considers the update cost during the design phase, or adjusts the structure by taking the query cost into consideration in the second round. We aim to construct an object tracking tree with minimum total cost including both the update and

query costs. This problem is formulated as an integer programming problem. We use the Lagrangean relaxation method to find an optimal solution and to develop a heuristic algorithm for constructing an object tracking tree with minimum cost.

Keywords: Wireless Sensor Networks (WSNs), Object Tracking, Optimization, Energy-Efficient, Lagrangean Relaxation (LR)



Table of Contents

論文摘要	I
THESIS ABSTRACT	III
TABLE OF CONTENTS	V
LIST OF TABLES	VII
LIST OF FIGURES	IX
CHAPTER 1 INTRODUCTION	1
1.1 Background	1
1.2 Motivation	3
1.3 Literature Survey	4
1.3.1 Object Tracking	4
1.3.2 Update Mechanism	8
1.3.3 Query Mechanism	12
CHAPTER 2 PROBLEM FORMULATION	15
2.1 Problem Description	15
2.2 Problem Notation	19
2.3 Problem Formulation	21
2.4 Varieties of the Model	23
CHAPTER 3 SOLUTION APPROACH	28
3.1 Introduction to Lagrange Relaxation Method	28
3.2 Lagrangean Relaxation (LR)	31
3.2.1 Subproblem 1 (related to decision variable $t_{(i,j)}^y$)	33
3.2.2 Subproblem 2 (related to decision variable x_{sp})	34
3.2.3 Subproblem 3 (related to decision variable $z_{(i,j)}^s$)	35
3.2.4 Subproblem 4 (constant part)	35
3.3 The Dual Problem and the Subgradient Method (IP)	36
CHAPTER 4 GETTING PRIMAL FEASIBLE SOLUTIONS	37
4.1 Lagrangean Relaxation Results	37
4.2 Getting Primal Feasible Solutions	37
4.3 Simple Algorithms	40
CHAPTER 5 COMPUTATIONAL EXPERIMENTS	41
5.1 Experiments Environment	41
5.2 Solution Quality	42
CHAPTER 6 CONCLUSION AND FUTURE WORK	49
6.1 Conclusion	49

6.2 Future Work 50
REFERENCES 51



List of Tables

Table 2-1	Problem Description	18
Table 2-2	Notation descriptions for given parameters (IP)	19
Table 2-3	Notation descriptions for indicate parameters (IP).....	20
Table 2-4	Notation descriptions for decision variables (IP).....	20
Table 2-5	Notation descriptions for decision variable $g_{(i,j)}^{xy}$	24
Table 2-6	Notation descriptions for new given parameter m_s	25
Table 4-1	The LR-based primal heuristic algorithm.....	38
Table 4-2	The object tracking tree algorithm.....	39
Table 5-1	Parameter of Lagrangean Relaxation based algorithm	41
Table 5-2	Evaluation of gap (%) and improvements ratio by given different number of nodes and different query rate	42





List of Figures

Figure 1-1.	A typical wireless sensor network.....	2
Figure 1-2.	A example of three-layer clustering architecture	5
Figure 1-3.	The events generated as object's moving.	9
Figure 1-4.	An example of object's moving	10
Figure 1-5.	An example of adding the shortcuts.....	10
Figure 1-6.	Example of calculating update cost.....	11
Figure 1-7.	Example of querying an object.....	12
Figure 1-8.	Example of 2D sensor sub-graph	13
Figure 2-1.	Example of object tracking.....	15
Figure 2-2.	Example of 2D tracking sub-graph	16
Figure 2-3.	Example of an update event	17
Figure 2-4.	Example of a query event	17
Figure 2-5.	An illustrated example of update	23
Figure 2-6.	An illustrated example of update and query event.....	25
Figure 2-7.	An illustrated example of update and query.....	26
Figure 3-1.	The Major Concepts of Lagrangean Relaxation Method.....	29
Figure 3-2.	Lagrangean Relaxation Method Procedure	30
Figure 5-1.	Evaluation of improve ratio to SA1 and SA2.....	44
Figure 5-2.	Evaluation of improve ratio to SA1 and SA2.....	44
Figure 5-3.	Evaluation of improve ratio to SA1 and SA2.....	45
Figure 5-4.	Evaluation of improve ratio to SA1 and SA2.....	45
Figure 5-5.	Computational time under different numbers of nodes	46
Figure 5-6.	The execution result of LR based algorithm.....	47
Figure 5-7.	Example of 2D tracking sub-graph.....	48
Figure 5-8.	Example of an object tracking tree (shortest path tree)	48
Figure 5-9.	Example of an object tracking tree ($T=0$)	48
Figure 5-10.	Example of an object tracking tree ($T=450$).....	48
Figure 5-11.	Example of an object tracking tree ($T=960$).....	48
Figure 5-12.	Example of an object tracking tree ($T=1920$).....	48



Chapter 1 Introduction

1.1 Background

In recent years, because the rapid growth in wireless communication and the inexpensive sensors capable of sensing environmental information, wireless sensor networks have been used in a wide range of applications. Such as Military intrusion detection, wildlife animal monitoring, civil applications.

As Figure 1-1, a lot of sensor nodes are deployed in a sensor field, and there is a special node so called sink node or base station. As an event occurs, such as the temperature rising over a threshold or the objective wildlife animal moving into a sensor node's sensing range, the sensor node collects the data and sends it back to the sink node for further processing. The scenario described above is called event-driven because the sensor nodes are appointed to monitor some event of interest. There are two other types of WSNs, called periodic and query-based WSNs. In the former, all sensor nodes periodically sense the data and send it back to the sink; in the latter, the user sends a query message to the sink to require information from some sensor nodes at any time.

There are many factors that must be taken into account when designing the WSNs, such as coverage, end-to-end delay, and lifetime. An important challenge in the design of WSNs is that the battery level is fixed and it is infeasible to recharge the battery. Sensor nodes are highly energy-limited due to the limitation of hardware and environment, therefore, more and more

research focus on the problem that how to prolong the lifetime, a lot of approaches were proposed, such as sleeping scheduling, data aggregation tree[10][19], and add some powerful nodes into the WSNs, etc..

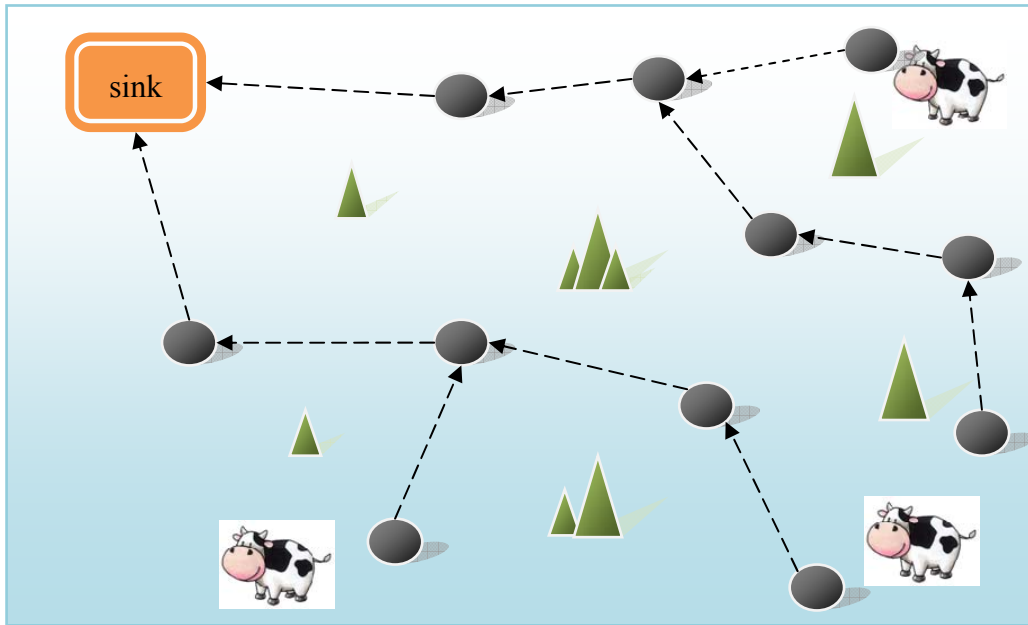


Figure 1-1. A typical wireless sensor network.

Object tracking is one of the key application issues of the WSNs. It can be used to track enemy vehicles, detect illegal border crossings, etc.. Sensor nodes are required to sense and track the movement of mobile objects then report to the special node, sink. Object tracking wireless sensor networks typically involve two basic operations to maintain and obtain the location of the target object [6].

The first is update. When object is moving from one sensor to another, the update of an object's location must be initiated to provide up-to-date information for the WSN. The cost caused by object moving is so called "update cost". The second is Query. In wireless networks, sink acts as a gateway between the wireless sensor network and outside network.

A query for object's location is usually sent from the outside network to the sink, and the sink forwards the query message to other nodes in the WSN to collect some information.

“Query cost” is defined as the total cost caused by transmitting the query message.

These two operations are interleaved during the entire process. In order to prolong the system's lifetime with the limitations, adopting an adequate method to minimize the total cost is necessary.

1.2 Motivation

There are many ways to maintain object's information while object move around in the WSN and query about the location of the target object. We can separate the way of storing data into two situations. The first situation, the data will be stored in different sensor nodes as a distributed database. A simple way to deliver the query message is to flood the entire network. However, great deals of query messages are wasted although there is no update message should be sent. Another aspect is to store all the information in one specific node, that is, the sink node. Once the sensor node senses the object is in its sensing range, the sensor node sends the update message back to the sink. Once the query arrives at the sink, there is no query message should be sent in the WSN. Even if the query cost is zero, the update cost, which caused by object moving, is still considerable when the frequency of object moving is high.

As described above, how to strike a balance between the update cost and query cost is

an important issue in object tracking wireless sensor network. We focus on the problem of building an energy-efficient wireless sensor network for object tracking by using object tracking tree with a given arbitrary topology. Accordingly, we are motivated to propose a heuristic algorithm for constructing an energy-efficient object tracking tree rooted at the sink so that the total communication cost can be computed and minimized.

1.3 Literature Survey

1.3.1 Object Tracking

There are many types of object tracking in design phase, such as cluster-based, prediction-based, tree-based architecture, etc.. In [8], H. Yang and B. Sikdar adopt the cluster based architecture in [18] to propose the Distributed Prediction Tracking (DPT) algorithm in order to address many challenges, such as scalable coordination, tracking accuracy, ad hoc deplorability, computation and communication cost, power constraints.

The cluster-based architecture in [18] is as Figure 1-2, wireless nodes are either switches or endpoints. Only switches can route packets, but both switches and endpoints can be the source or the destination of data. In the clustering scheme, it creates a set of cluster at each layer. All nodes are joined to the lowest layer (Layer 0). Node B, G, and K are the cluster representatives (or cluster head) of these clusters. These cluster head join the next layer immediate above, and placed these nodes (B, G, K) into one cluster. Then select the cluster heads of the clusters in Layer 1 and these nodes are also present in Layer 2.

To construct the hierarchical structure described above, it postulates some properties that should be present in the clustering mechanism that run at each layer, each cluster is connected, two cluster should have low overlap, clusters should be stable across node mobility, etc.. As the DPS's name suggest, this algorithms does not required any central control unit, making it robust against random node failures.

There are many different type of prediction models, Linearly Prediction [8], Kalman Filter[11], Gray Theory. The main idea of the prediction algorithm is based on an estimation of the target's present moving speed and direction to predict the target's next predicted location after a given period of time. DPS can also adopt higher order prediction, which predicts the n th location information using the previous $n-1$ actual location.

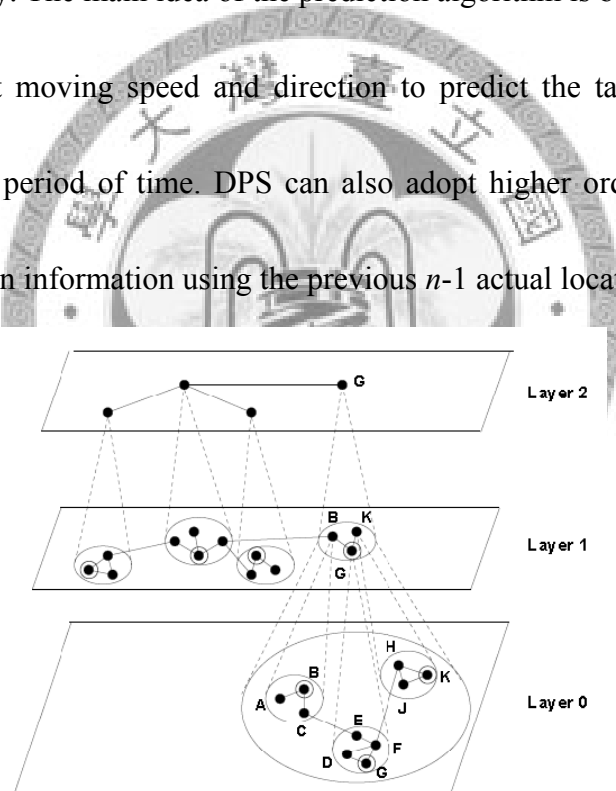


Figure 1-2. A example of three-layer clustering architecture [8].

Y. XU, Julian Water, and W.C. Lee[17] proposed another prediction based algorithm so called Prediction-based Energy Saving scheme (PES), which is used to minimize the number of nodes participating in the tracking process, while inactivate other nodes into

sleeping mode. PES is proposed to eliminate the total power consumption by sleeping scheduling in order to prolong the system lifetime.

“No matter what heuristic used in prediction mechanisms and wake up mechanisms, we are not able to guarantee 0% missing rate, except for waking up all the sensor nodes each time as the SM scheme does” quoted by [17]. If the object is not found by current nodes, it needs a recovery mechanism to relocate the object. DPS and PES both contains an appropriate recovery heuristic mechanism which is used to avoid system crashed due to wrong prediction.

The other type of object tracking similar to the cluster-based architecture is the tree-based structure. H.T. Kung and D. Vlah make two contributions in [1]. This paper proposed a tree-based structure so called STUN — Scalable Tracking Using Networked Sensors—a scalable tracking architecture called *hierarchy tree*. *“The tree is rooted at sink node, the leaves of the tree are sensors, and the other nodes are communication nodes, which are called intermediate nodes. Each intermediate node stores the set of object that were detected jointly by its descendants”*[1]. The set called *detected set* or *detected list*. For example, the detected set of at a sensor node consists of the objects within the sensor’s sensing range; while the root’s detected set contains all objects present in the entire region. In order to maintain the detected list up to date, update message should be sent back to the sink node from the sensor nodes. However, the message does not always need to reach the root; we will discuss the update mechanism in 1.3.2.

The second contribution of [1] is DAB (drain-and-balance), a method to construct STUN's hierarchical structure in a bottom-up fashion through a series of *DAB steps*. Within each DAB step, consisting of two phases, Draining and Balancing; a subset of the sensors is properly merged into balanced subtrees, and the high-rate subsets are merged first. *"In the first phase, sensors are partitioned using one or more event rate thresholds or called draining thresholds. Then, it adds those nodes into the DAB tree which have at least one incident edge whose weight is greater than or equal to the draining threshold. In the balancing phase, it repeatedly merge pairs of adjacent trees in T to form clusters"*[1].

In [6], it takes a two stage approach to construct a object tracking tree. The first stage aims at reducing the update cost while the second stage aims at further reducing the query cost. For the first stage, two solutions are proposed: Deviation-Avoidance Tree (DAT) and Zone-based Deviation-Avoidance Tree (Z-DAT). From the calculation of update cost, it makes three observations about the update cost to develop the DAT algorithm. It examine all links whether fit the conditions or not for each link, add into the tree if the link fit all conditions, otherwise, deviation may occur.

Z-DAT is based on the DAT and the locality concept, entire sensing range partitioned into α^2 square-like zone by adjusting two parameters α and δ . Then, it runs DAT on the sensors in each zone and merges subtrees in the above α^2 zones recursively until one single tree is obtained.

The above DAT and Z-DAT only consider the update cost. Query Cost Reduction

(QCR) is designed to reduce the update and query cost by adjusting the object tracking tree constructed by DAT or Z-DAT. In QCR, it examines the tree in a bottom-up manner and tries to adjust the structure by the following two operations.

“1. If a node v is not a leaf node, we can make it a leaf by cutting the links to its children and connecting each of its children to $p(v)$ ”

“2. If a node v is a leaf node, we can make $pevT$ closer to the sink by cutting v 's link to its current parent $p(v)$ and connect v to its grandparent $p(p(v))$.”[6]

The approaches described above are assumed there is only one sink, [7] extend the problem from single sink to multi sink. Two algorithms were proposed, MT-HW (Multi-tree construction with the weight-first property) and MT-EO (multi-tree construction with the edge-overlap-first property). These algorithm strike the tradeoff between the update and query costs, and the experiments verifies the benefits of a multi-sink WSN from different aspect.

1.3.2 Update Mechanism

The simply update approach is to sent the update message back to the sink when objects moving between sensors. The update cost is considerable if objects move frequently. Using appropriate update mechanism can successfully reduce the update cost. Figure 1-3 is an example of generating the update messages and sending it to the sink node when object moving.

Many research put the *detected list* into the nodes in the WSN try to reduce the number of update message transmitted. When an object o move from sensor u 's sensing range to sensor node v 's sensing range, a departure message $dep(o,u,v)$ will be reported by sensor u and a arrival message $arv(o,u,v)$ be reported by sensor v along the tree path to $lca(u,v)$.

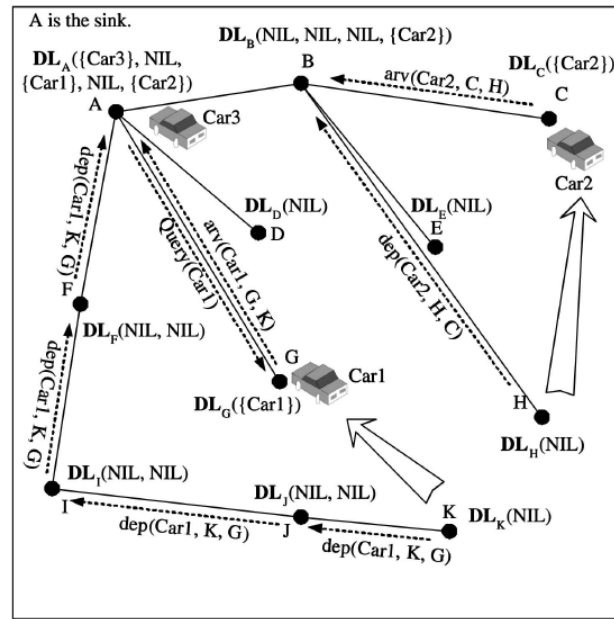


Figure 1-3. The events generated as object's moving[6].

$lca(u,v)$ means the lowest common ancestor of u and v in the tree T and denotes the root of the minimum subtree in T that contains both u and v . As shown in Figure 1-4 (a), when $Car1$ moves from g 's sensing range to h 's sensing range, the departure message reported by g and the arrival event reported from h update the relative information along the tree path to node b ($lca(g,h)$). The update message is not always sent to the sink, since the update cost can be reduced successfully.

In [2], it proposes a new data aggregation structure, a message-pruning tree with shortcuts, the edge ($d \rightarrow h$) in Figure 1-4 (b). When $Car1$ move from g to h , the update event

sent to node d from node g and the arrival event is transmitted from d to h . The addition of shortcuts should be observed the following two conditions:

$$1. dist_T(sink, u) + w_T(u \rightarrow v) \leq dist_T(sink, u)$$

$$2. (u, v) \in E_G - E_T$$

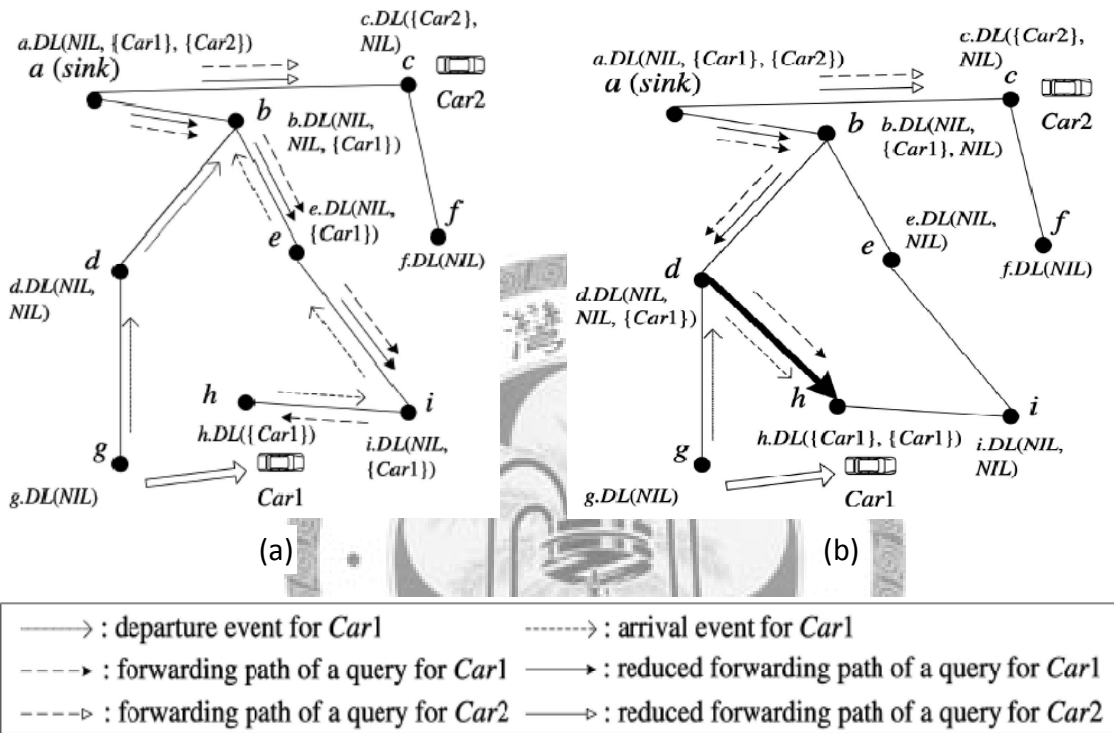


Figure 1-4. An example of object's moving [2].

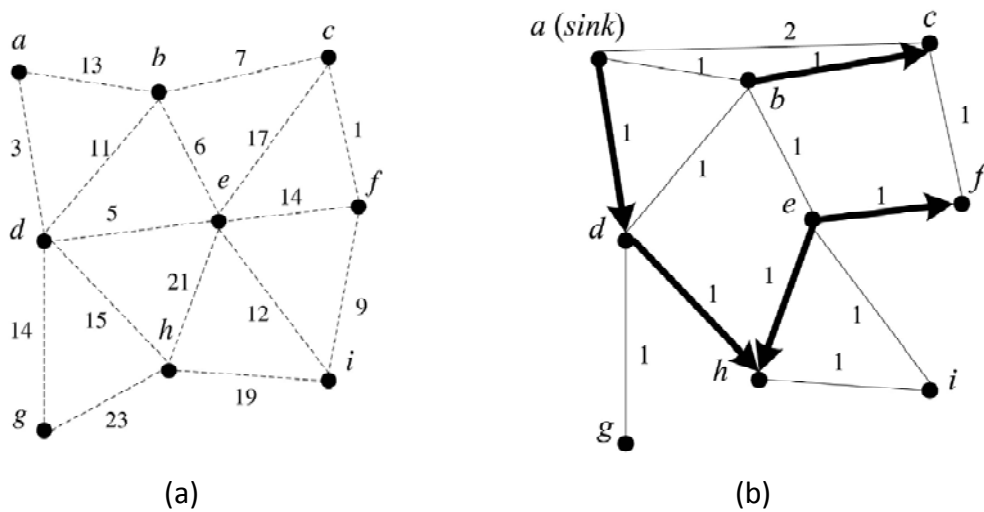


Figure 1-5. An example of adding the shortcuts[2].

As shown in Figure 1-5, (a) is the original graph G , (b) is a message-pruning tree with shortcuts.

Our model is an extension of [4], the update mechanism we adopted is shown as Figure 1-6. When object moves from sensor nodes x to sensor node y , we only transmit the update event from sensor y to the $lca(x,y)$. Although the detected list in communication node q is incorrect, it will not affect the correctness of the query result. Compare our method and the method without shortcuts, our method can save at least half of the update cost. Although the update cost in the structure with shortcuts is less than our mechanism, but it doesn't take the query cost into consider.

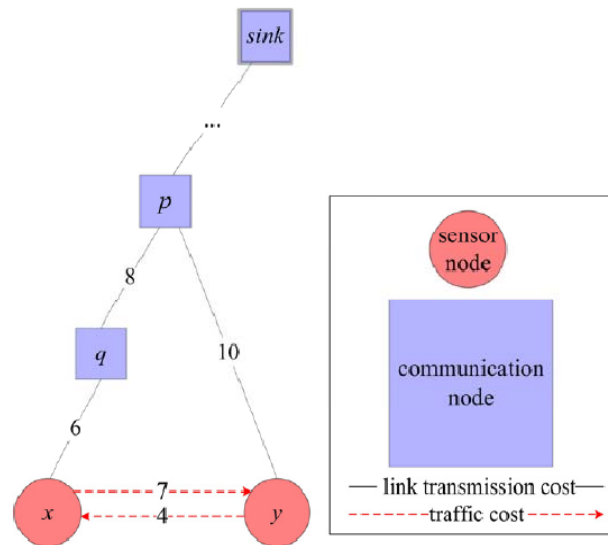


Figure 1-6. Example of calculating update cost[4].

1.3.3 Query Mechanism

We can find which sensor is the nearest to the target object by forwarding a query event along the tree path of the object tracking tree. If there is no detected list in the entire WSN, the query message should be sent from sink to the sensor node whose sensing range covers the target object. As Figure 1-7 shown, if outside network send a requirement to query the object's location, sink will transmit a query message forwarding to the sensor node x along the tree path after it receive the requirement. Some research join the communication node and detected list into the WSN [2][4][6][7], using the reduced forwarding path to transmit the query event to reduce the query cost. Figure 1-7 as an example, sink also sent the query event along the tree path. Due to the communication node r store a detected list and it know the descendant x is a leaf node, node r can confirm the object absolutely is in sensor node x , therefore, the query message have no need to be forwarded to the sensor node x .

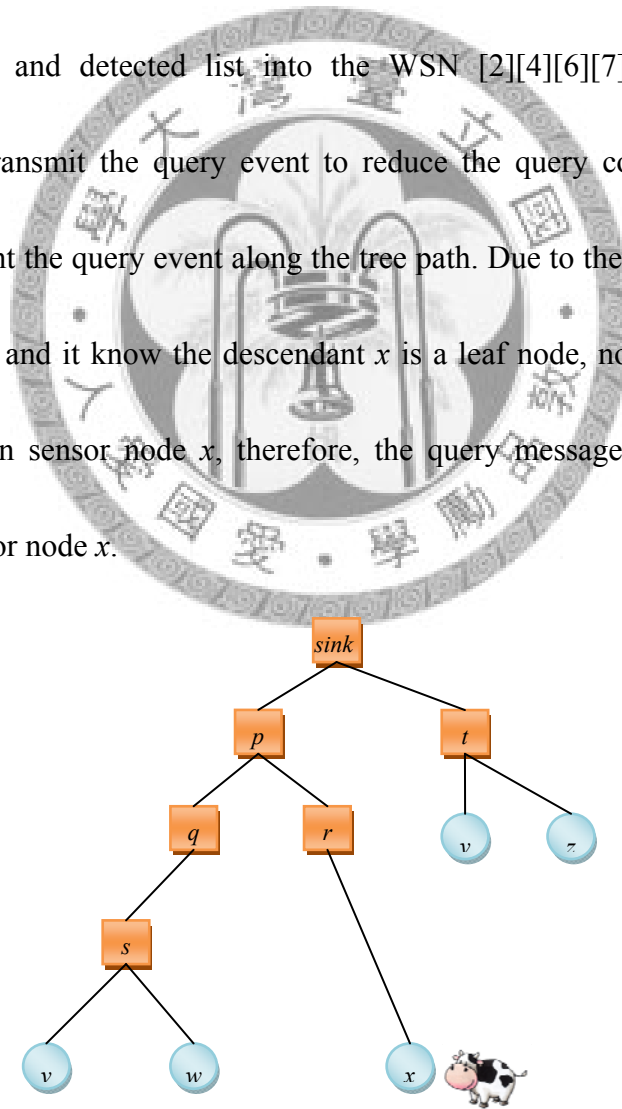


Figure 1-7. Example of querying an object.

Therefore, the cost of querying object is

$$2 \times \left(\sum_{s \in S} q(v) \times \text{dist}_T(\text{sink}, \text{parent}(s)) \right)$$

where $q(v)$ denotes the query rate of sensor v .

Since, the inadequate calculation of query rate may affect the total query cost, we use an approximate approach to calculate the query rate by using the Markov chain[14]. Figure 1-8 shows a sensor field with each edge connecting a pair of adjacent sensor nodes. The weight of each link represents the object moving frequency of each pair of sensors.

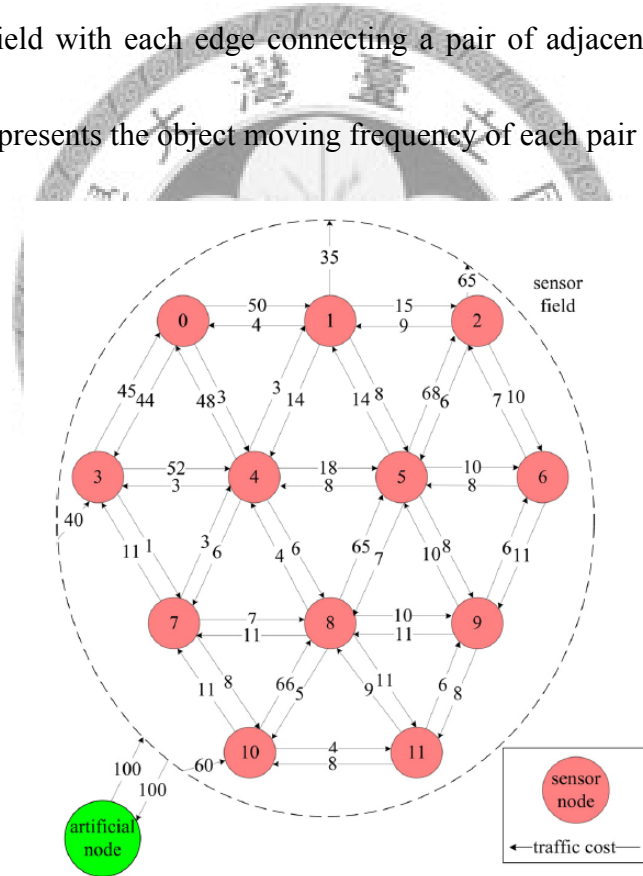


Figure 1-8. Example of 2D sensor sub-graph [4].

We model the object movements as a stochastic process, the following property will be observed in the steady-state.

$$S^{(0)}P = S^{(1)} \rightarrow S^{(0)}P^{n+1} = S^{(0)}P^n = S^{(n)} \rightarrow S^{(n)}P = S^{(n)} \rightarrow \pi P = \pi$$

$S^{(i)}$ denotes the network state at time i . For example, there are 5 sensors (sensor 1, 2, 3, 4, 5), only sensor 1 covers the target object at time 1, hence, $S^1=[1, 0, 0, 0, 0]$.

P is a 2×2 array, each elements P_{mn} in the array denotes a probability of objects moving from sensor m to sensor n .

We use π to indicate the network state at steady-state as equation (1), and the summation of every element in π should be equal to 1 as equation (2). Combine these two conditions as following:

$$\begin{cases} \pi P = \pi \\ \pi_1 + \pi_2 + \dots + \pi_n = 1 \end{cases} \quad (1)$$

$$\rightarrow [\pi_1 \ \pi_2 \ \dots \ \pi_n] \times \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} & 1 \\ p_{21} & \dots & \dots & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots & 1 \\ p_{m1} & \dots & \dots & p_{mn} & 1 \end{bmatrix} = [e_1 \ e_2 \ \dots \ e_n] \quad (3)$$

We use $\pi A = e$ to represent equation (3), then we try to find the result of π .

Step 1: $\pi A A' = e A'$

Step 2: $\pi A A' (A A')^{-1} = e A' (A A')^{-1}$

$A A' (A A')^{-1}$ is a identity matrix, hence we get $\pi = e A' (A A')^{-1}$, π_x as the probability of the object which is in the sensing range of sensor node x . We further let π_x multiplied by T as the query rate of node x at a given period of time, T is the total number of queries in a unit time.

Chapter 2 Problem Formulation

2.1 Problem Description

Our approach focus on construct an object tracking tree which is used to record object's information and maintain this information up to date. The sensor field consists of sensor nodes and communication nodes. Sensor nodes are appointed to sense and track the mobile object and send the information back to the sink. Communication nodes are required relay the update message, store and maintain a detected list. Figure 2-1 shows an example of object tracking.

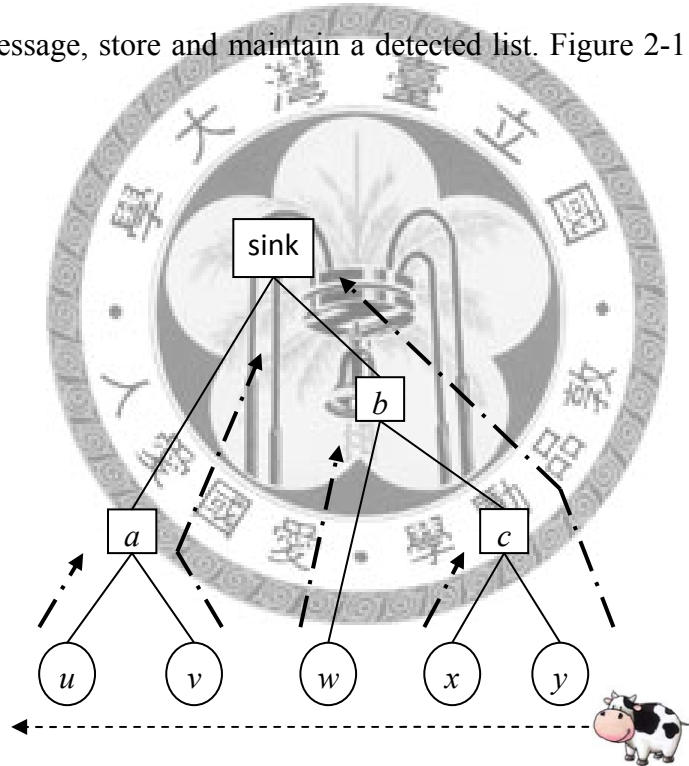


Figure 2-1. Example of object tracking.

The object tracking problem is modeled as a directed graph, $G(V, L)$, as shown in Figure 2-2, where V is a set of communication nodes and sensor nodes random deployed in a 2D sensor field and L is a set of links between adjacent sensor nodes or connected one

communication node and one sensor node. Each link weight represents the distance between sensor nodes. Since, upward links and downward links may have different transmission cost. Figure 2-3 shows an example of an update event and Figure 2-4 is a query event. This approach can keep a certain ratio between upward link cost and downward link cost. Therefore, we define the transmission cost as the power consumption of transmitting data which is measured as $r^\alpha + c$, where α is a signal attenuation constant(usually between 2 to 4) and c is a positive constant that represent signal processing and r is Euclidean distance between any nodes.

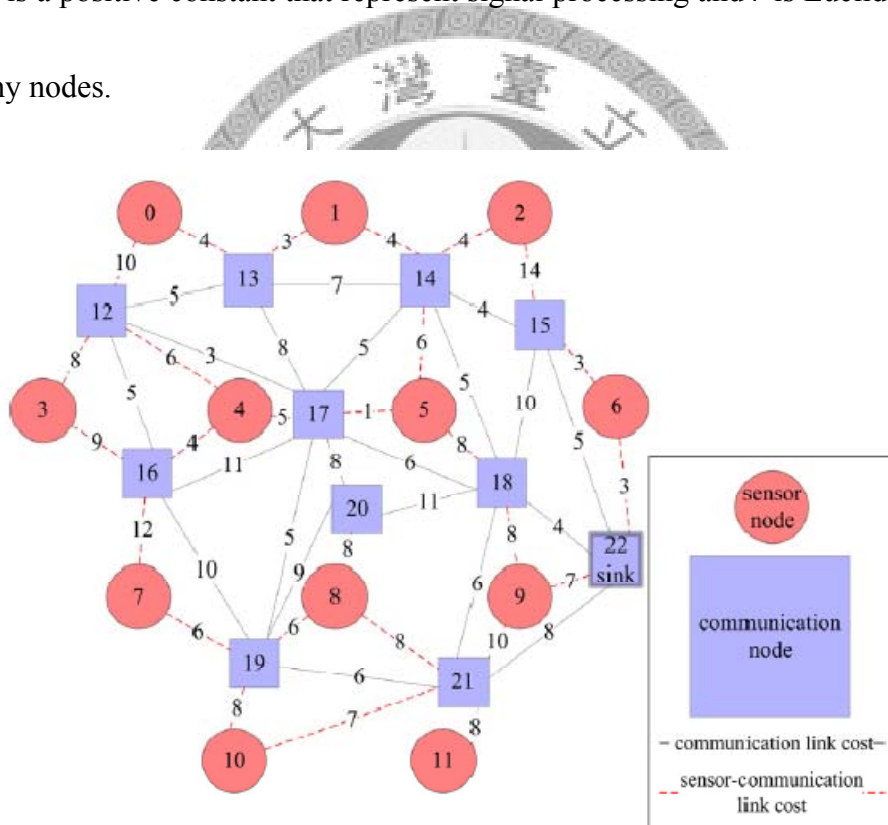


Figure 2-2. Example of 2D tracking sub-graph [4]

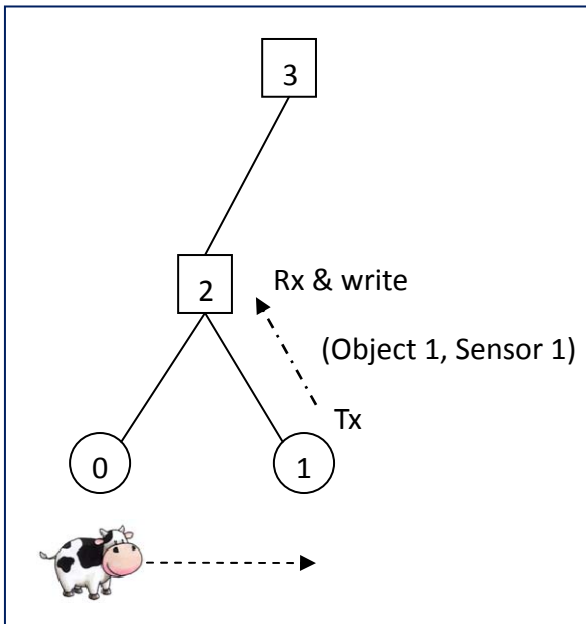


Figure 2-3. Example of an update event

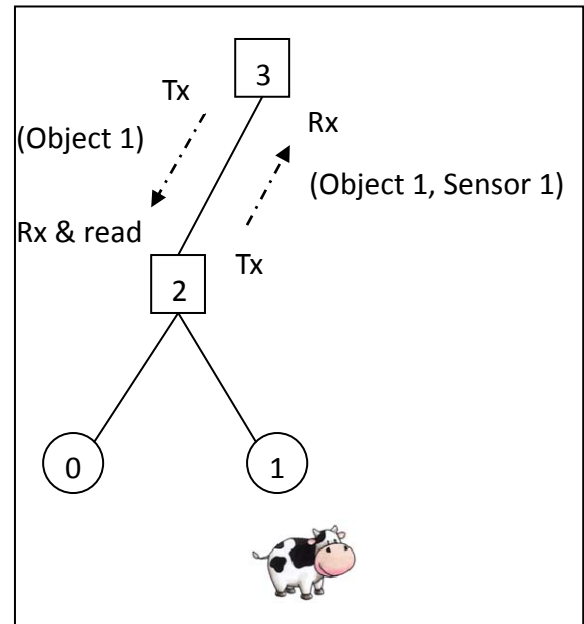


Figure 2-4. Example of a query event

In this study, we consider a given arbitrary sensor network topology as a directed graph, two-way object moving frequency of in-sensor field and incoming-outgoing sensor filed, two-way link transmission cost, and nodal processing cost. We deploy a tree-based architecture, sensor nodes as leaf nodes and send data to its ancestor which is adjacent communication node.

A good tracking method is characterized by a low total communication cost [9]. Given an arbitrary graph, we can compute the total communication cost. We define the total communication cost for graph G as following:

$$\text{Total communication cost } (G) = \text{total update cost} + \text{total query cost}.$$

We try to construct an object tracking tree with minimum total cost to achieve efficient object tracking in WSNs.

Table 1-1 Problem Description

Problem Assumption

- Fixed sensing range and fixed transmission range.
- All objects are identified.
- Two-way transmission cost.
- Two-way moving frequency.
- Object may exit and enter the sensing field.
- Transmission cost of each link with respect to energy consumption.

Given

- The network topology including node set and link set.
- The set of communication nodes, including sink node, and sensor nodes.
- Communication cost for each link.
- The event rate between each sensor nodes.
- The event rate of object enters or exits the sensing field.
- The set of all candidate paths from the sensor node to the sink node.

Objective

- To minimize the total cost of constructing the object tracking tree.

Subject to

- Routing constraint — each source node should only choose one routing path to send data back to the sink node.
- Tree constraint — the union of routing paths of each sensor nodes shall be a tree, namely, a object tracking tree.
- Variable-transformation constraint — the calculation of update cost and query cost must obey the mechanism describe above.

To determine

- Whether or not a link should be on the object tracking tree

2.2 Problem Notation

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

Table 1-2 Notation descriptions for given parameters (IP)

Given Parameters	
Notation	Description
S	The set of all sensor nodes.
C	The set of all communication nodes, including <i>sink</i> node.
R	The set of the frequency (r_{xy}) of object movement from x to y , $\forall x, y \in S \cup \{o\}, x \neq y$.
L	The set of all links, $(i, j) \in L, i \neq j$.
A	The set of transmission cost $a_{(i,j)}$, associated with link (i, j) .
P	The set of all candidate path P between any pair $(s, sink), \forall s \in S$.
Q_s	The probability of node s that object is in its sensing range, $\forall s \in S$.
T	Total number of the queries per unit time.
o	The artificial node outside the sensor field.
u	The coefficient of upward links.
v	The coefficient of downward links.
d_c^w	Nodal processing cost of a writing operation of the communication node c .
d_c^r	Nodal processing cost of a reading operation of the communication node c .

Table 1-3 Notation descriptions for indicate parameters (IP)

Indicate Parameter	
Notation	Description
$\delta_{p(i,j)}$	The indicator function which is 1 if link (i,j) is on path p and 0 otherwise.
$w_{(i,j)}$	1 if $i, j \in C$ 0 otherwise.

Table 1-4 Notation descriptions for decision variables (IP)

Decision Variables	
Notation	Description
x_{sp}	1 if the sensor nodes s uses the path p to reach the <i>sink</i> node and 0 otherwise, $\forall s \in S, p \in P$.
$z_{(i,j)}^s$	1 if the sensor node s uses the link (i,j) to reach the <i>sink</i> node and 0 otherwise.
$t_{(i,j)}^{xy}$	1 if $z_{(i,j)}^x = 0 \cap z_{(i,j)}^y = 1$ (reporting object's location uses the link (i,j) when object moves from sensor x to sensor y) and 0 otherwise, $x \neq y$

2.3 Problem Formulation

Objective Function

$$Z_{IP} = \text{Min} \sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in L} t_{(i,j)}^{xy} r_{xy} (ua_{(i,j)} + d_j^w) + \sum_{s \in S} \sum_{(i,j) \in L} z_{(i,j)}^s (r_{os} + r_{so}) (ua_{(i,j)} + d_j^w) \\ + \sum_{s \in S} \sum_{(i,j) \in L} [w_{(i,j)} z_{(i,j)}^s Q_s T(va_{(j,i)} + d_i^r) + w_{(i,j)} z_{(i,j)}^s Q_s T(ua_{(i,j)})]$$

Subject to:

$$\sum_{p \in P_s} x_{sp} = 1 \quad \forall s \in S \quad (\text{IP 1})$$

$$\sum_{j \in C} z_{(i,j)}^s \leq 1 \quad \forall s \in S, i \in S \cup C \quad (\text{IP 2})$$

$$\sum_{p \in P_s} x_p \delta_{p(i,j)} \leq z_{(i,j)}^s \quad \forall s \in S, (i,j) \in L \quad (\text{IP 3})$$

$$2t_{(i,j)}^{xy} \leq z_{(i,j)}^y - z_{(i,j)}^x + 1 \quad \forall x, y \in S, (i,j) \in L \quad (\text{IP 4})$$

$$z_{(i,j)}^y - z_{(i,j)}^x + 1 \leq t_{(i,j)}^{xy} + 1 \quad \forall x, y \in S, (i,j) \in L \quad (\text{IP 5})$$

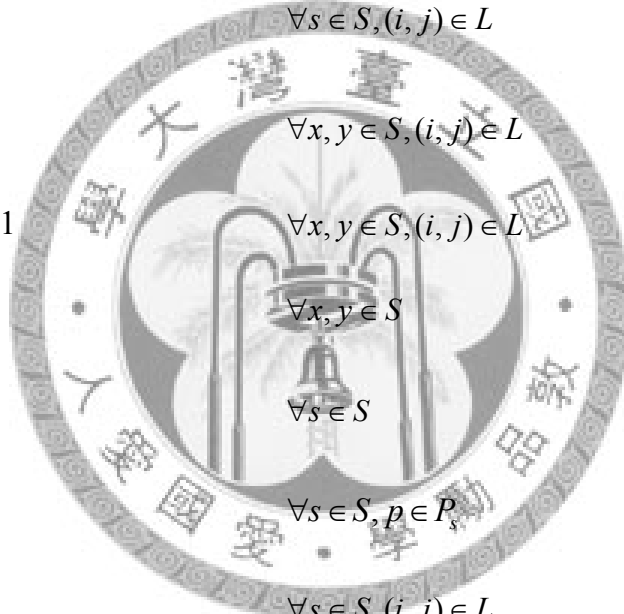
$$\sum_{j \in C} t_{(y,j)}^{xy} = 1 \quad \forall x, y \in S \quad (\text{IP 6})$$

$$\sum_{j \in C} z_{(s,j)}^s = 1 \quad \forall s \in S \quad (\text{IP 7})$$

$$x_{sp} = 0 \text{ or } 1 \quad \forall s \in S, p \in P_s \quad (\text{IP 8})$$

$$z_{(i,j)}^s = 0 \text{ or } 1 \quad \forall s \in S, (i,j) \in L \quad (\text{IP 9})$$

$$t_{(i,j)}^{xy} = 0 \text{ or } 1 \quad \forall x, y \in S, (i,j) \in L \quad (\text{IP 10})$$



The objective function is to minimize the total cost of constructing an object tracking tree. The total cost is defined as the combination of update and query costs.

Constraint (IP 1): Routing constraint: For each sensor nose s , it exactly exist one path only between the s and the *sink*.

Constraint (IP 2): To avoid cycle, we enforce that any nodes' outgoing link to

communication node should be equal to 1 on the object tracking tree, except the *sink* node.

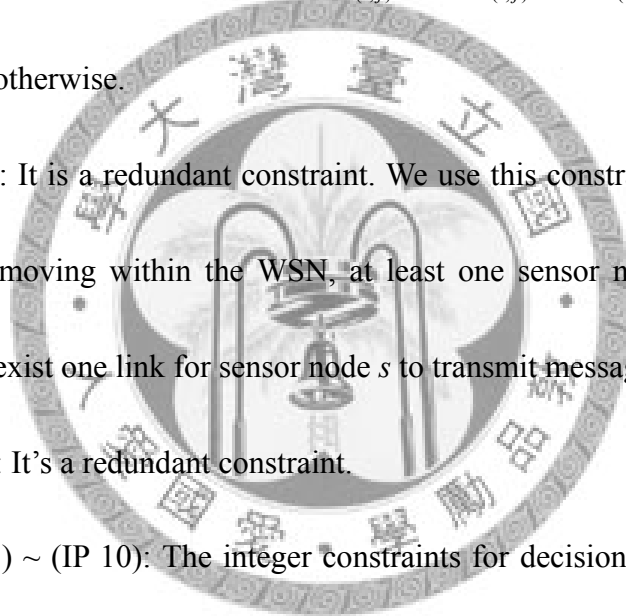
Constraint (IP 3): If the path x_p has been chosen, and the link (i, j) is on the path, the link (i, j) should be chosen, i.e. decision variable $z_{(i,j)}^s$ should be enforced to equal 1

Constraints (IP 4) and (IP 5): These two constraints are variable-transformation constraints. When object moves from sensor node x to sensor node y using the link (i, j) to report object's location to the sink, i.e. $z_{(i,j)}^x = 0 \cap z_{(i,j)}^y = 1$, $t_{(i,j)}^{xy}$ must be enforced to equal 1 and 0 otherwise.

Constraint (IP 6): It is a redundant constraint. We use this constraint to guarantee that when an object moving within the WSN, at least one sensor node s can detect the object and there exist one link for sensor node s to transmit message to *sink*.

Constraint (IP 7): It's a redundant constraint.

Constraints (IP 8) ~ (IP 10): The integer constraints for decision variables x_p , $z_{(i,j)}^s$, $t_{(i,j)}^{xy}$ must be equal 0 or 1.



2.4 Varieties of the Model

We can extend the model to several different scenarios to fulfill more application.

Scenario1:

We assume the mobile agent will move into any sensor nodes to get information, if we apply the original update mechanism described in 1.3.2, the mobile agent will get incorrect information. Since, we further update the detected list in all communication nodes. Figure 2-5 as an example when the object moves from sensor node x to sensor node w . We not only send a arrival message from sensor node w along the tree path to the lowest common ancestor, but also send a leaving message from sensor node x . Therefore, the detected list of communication node c can be corrected. In order to correct all communication node, we need to modify our model that replacing the decision variable $t_{(i,j)}^{xy}$ by $g_{(i,j)}^{xy}$. Table 1-5 shows the description of $g_{(i,j)}^{xy}$.

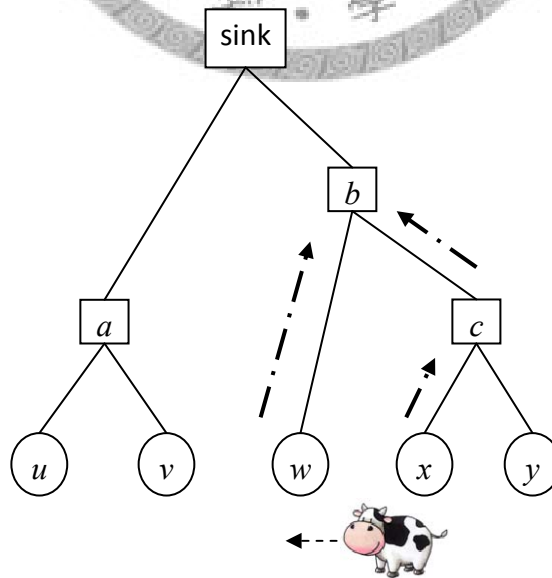


Figure 2-5. An illustrated example of update

Table 1-5 Notation descriptions for decision variable $g_{(i,j)}^{xy}$

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

Update cost should be reformulated as following:

$$\sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in S} (z_{(i,j)}^x + z_{(i,j)}^y - 2g_{(i,j)}^{xy}) r_{xy} (ua_{(i,j)} + d_j^w)$$

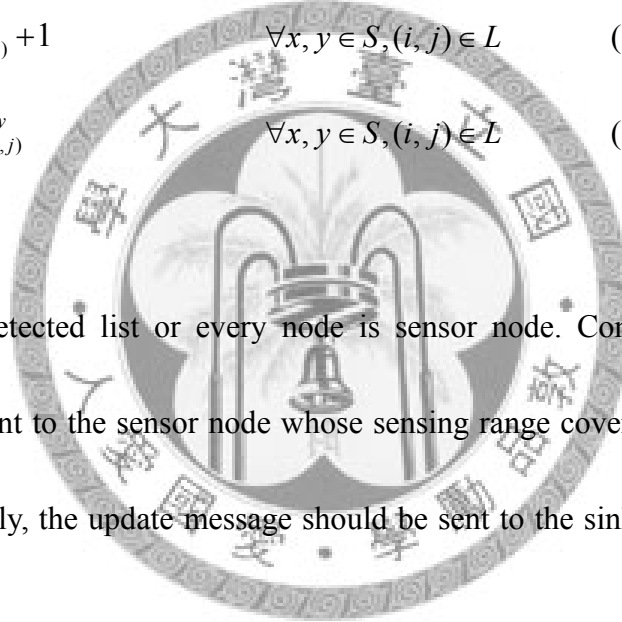
Furthermore, it replace (IP 4) and (IP 5) by (IP 4.1) and (IP 5.1):

$$z_{(i,j)}^x + z_{(i,j)}^y \leq g_{(i,j)}^{xy} + 1 \quad \forall x, y \in S, (i, j) \in L \quad (\text{IP4.1})$$

$$z_{(i,j)}^x + z_{(i,j)}^y \geq 2g_{(i,j)}^{xy} \quad \forall x, y \in S, (i, j) \in L \quad (\text{IP5.1})$$

Scenario 2:

If there is no detected list or every node is sensor node. Consequently, the query message should be sent to the sensor node whose sensing range covers the target object or the leaf node. Similarly, the update message should be sent to the sink node. Figure 2-6 as an example



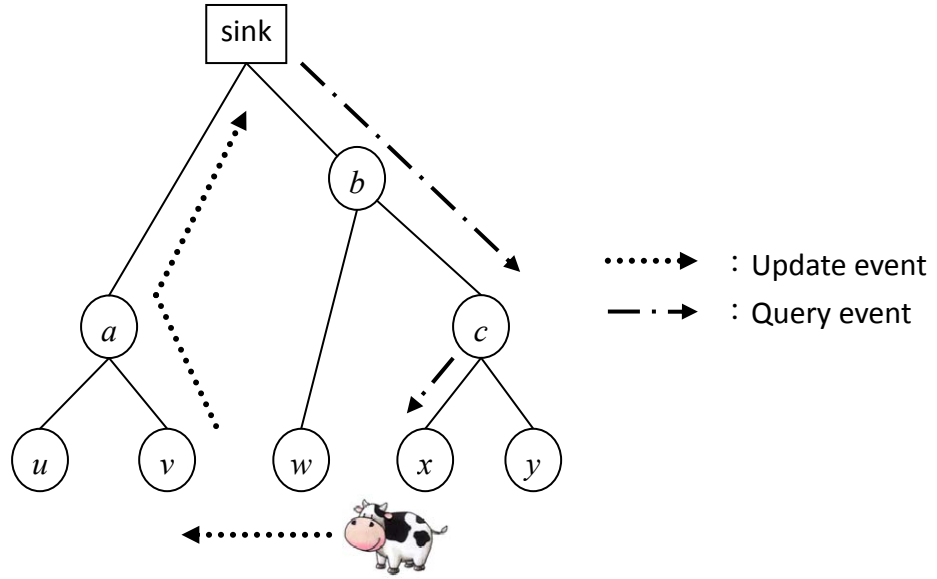


Figure 2-6. An illustrated example of update and query event

The query cost must be modified as

$$2 \sum_{s \in S} \sum_{(i,j) \in L} [w_{(i,j)} z_{(i,j)}^s Q T(va_{(j,i)} + d_i^r) + w_{(i,j)} z_{(i,j)}^s Q T(ua_{(i,j)})]$$

The update cost modified as

$$\sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in L} z_{(i,j)}^s r_{xy} (ua_{(i,j)} + d_j^w) + \sum_{s \in S} \sum_{(i,j) \in L} z_{(i,j)}^s (r_{os} + r_{so}) (ua_{(i,j)} + d_j^w)$$

Scenario 3:

We extend the task of the object tracking tree to send some message to the object. We add a given parameter m_s .

Table 1-6 Notation descriptions for new given parameter m_s

Notation	Description
m_s	the cost of the sensor node s send message to object

The query cost must be reformulated as following:

$$\sum_{s \in S} \sum_{(i,j) \in L} [w_{(i,j)} z_{(i,j)}^s Q T(va_{(j,i)} + d_i^r) + w_{(i,j)} z_{(i,j)}^s Q T(ua_{(i,j)})] + m_s$$

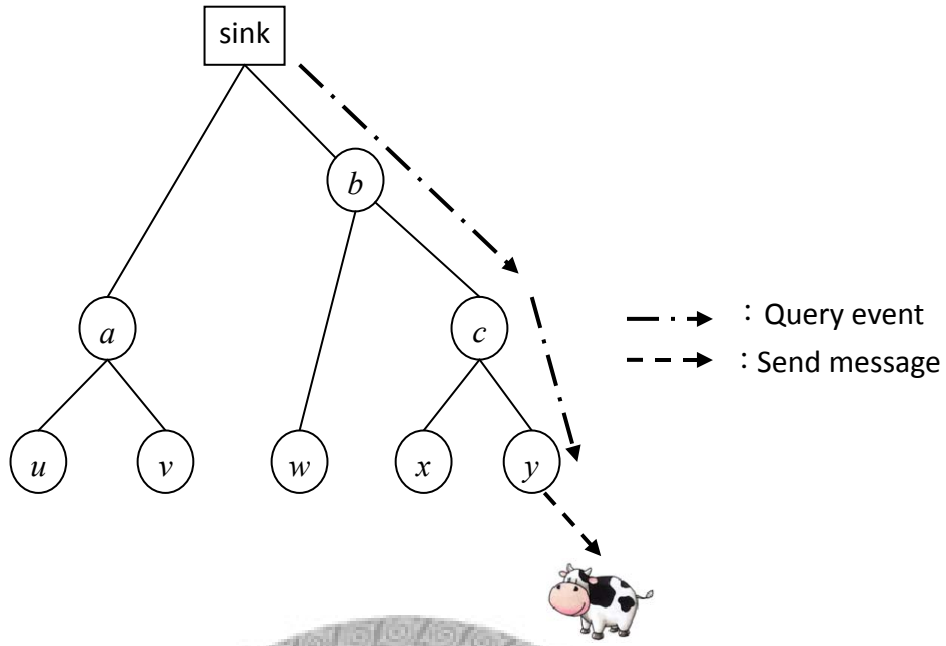


Figure 2-7. An illustrated example of update and query

Scenario 4:

In order to enhance the practical usefulness of our model, some constraints can be added to our mathematical model, such as followings:

1. hop constraint:

$$\sum_{p \in P_s} \sum_{(i,j) \in L} x_p \delta_{p(i,j)} \leq H_s \quad \forall c \in C$$

H_s means the restriction of the tree path of sensor node s .

2. Throughput constraint:

$$\sum_{x \in S} \sum_{y \in S} \sum_{i \in S+C} t_{(i,c)}^{xy} r_{xy} + \sum_{s \in S} \sum_{j \in C} (z_{(j,c)}^s + z_{(c,j)}^s) TQ_s \leq G_c \quad \forall c \in C$$

3. Battery capacity constraints:

$$\sum_{x \in S} \sum_{j \in C} t_{(s,j)}^{xs} E_s^{send} \leq E_s \quad \forall s \in S$$

$$\sum_{x \in S} \sum_{y \in S} \sum_{i \in S+C} (t_{(i,c)}^{xy} E_c^{receive} + t_{(c,i)}^{xy} E_c^{send}) + \sum_{s \in S} \sum_{i \in S+C} (z_{(i,c)}^s E_c^{receive} + z_{(c,i)}^s E_c^{send}) \leq E_c \quad \forall c \in C$$

These 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.



Chapter 3 Solution Approach

3.1 Introduction to Lagrange Relaxation Method

Many approaches had been proposed in 1970s [1], most of them used the divide-and-conquer technique to decompose a complicated problem into several plain sub-problems and solve them respectively. Lagrangean relaxation method is one of the popular approaches used for solving some mathematical problems, like integer programming problems [15] [16] Because 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.

First, we remove some complex constraints of the primal mathematical solution to the objective function with corresponding multiplier, and then the original problem will be transformed into a new Lagrangean relaxation problem in many different ways. 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 principle concept of the Lagrangean relaxation method has been shown in Figure 3-1, and a detailed flow chart of it in Figure 3-2.

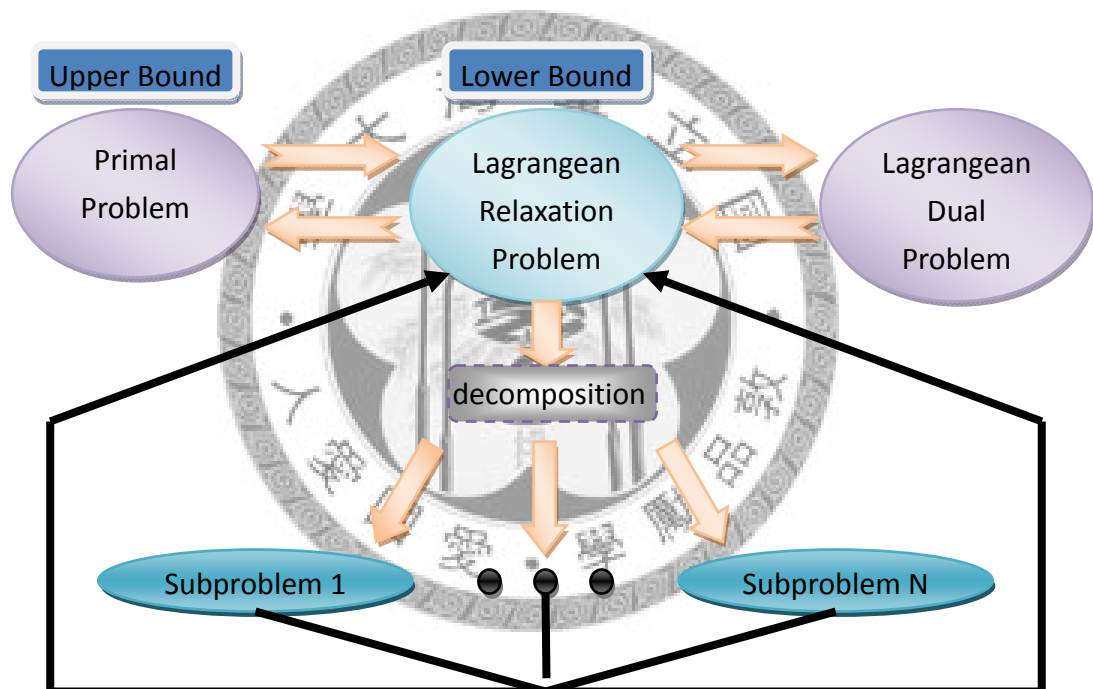


Figure 3-1. The Major Concepts of Lagrangean Relaxation Method

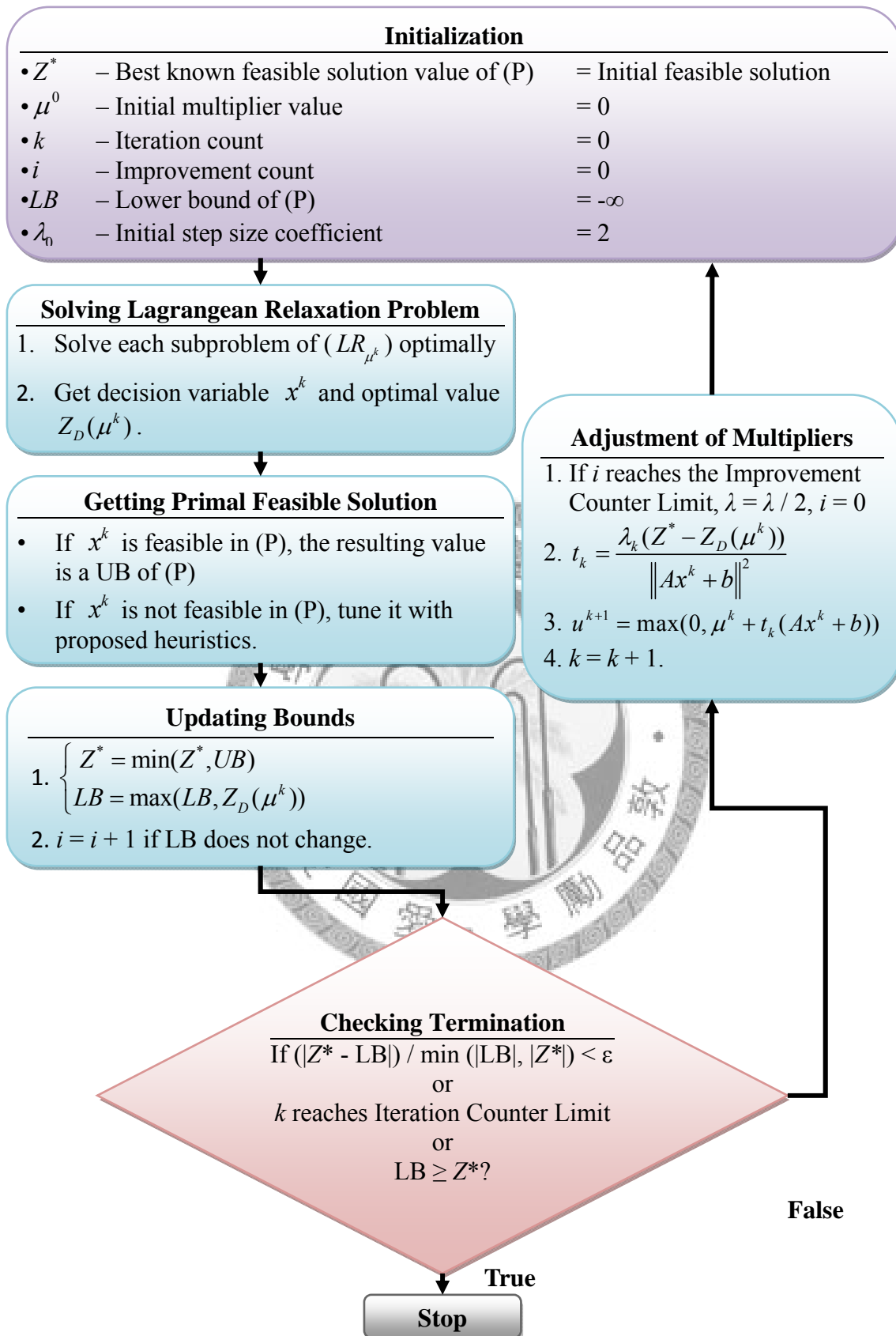


Figure 3-2. Lagrangean Relaxation Method Procedure

3.2 Lagrangean Relaxation (LR)

By adopting Lagrangean relaxation method, we can transform the primal problem into the following Lagrangean relaxation problem by relaxing the constraints (IP 3), (IP 4), (IP 5). For a vector of non-negative multipliers, we present the Lagrangean relaxation problem as bellow:

Objective function:

$$\begin{aligned}
 Z_{LR} &= (u_{s(i,j)}^1, u_{xy(i,j)}^2, u_{xy(i,j)}^3) \\
 \min & \sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in L} t_{(i,j)}^{xy} r_{xy} (ua_{(i,j)} + d_j^w) + \sum_{s \in S} \sum_{(i,j) \in L} z_{(i,j)}^s (r_{os} + r_{so}) (ua_{(i,j)} + d_j^w) \\
 & + \sum_{s \in S} \sum_{(i,j) \in L} [w_{(i,j)} z_{(i,j)}^s QT(va_{(j,i)} + d_i^r) + w_{(i,j)} z_{(i,j)}^s QT(ua_{(i,i)})] \\
 & + \sum_{s \in S} \sum_{(i,j) \in L} u_{s(i,j)}^1 \left(\sum_{p \in P_s} x_p \delta_{p(i,j)} - z_{(i,j)}^s \right) \\
 & + \sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in L} u_{xy(i,j)}^2 (2t_{(i,j)}^{xy} - z_{(i,j)}^y + z_{(i,j)}^x - 1) \\
 & + \sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in L} u_{xy(i,j)}^3 (z_{(i,j)}^y - z_{(i,j)}^x - t_{(i,j)}^{xy})
 \end{aligned} \tag{LR}$$

Subject to:

$$\sum_{p \in P} x_{sp} = 1 \quad \forall s \in S \tag{LR1}$$

$$\sum_{j \in C} z_{(i,j)}^s = 1 \quad \forall s \in S, i \in S \cup C \tag{LR2}$$

$$\sum_{j \in C} t_{(y,j)}^{xy} = 1 \quad \forall x, y \in S \tag{LR3}$$

$$\sum_{j \in C} z_{(s,j)}^s = 1 \quad \forall x, y \in S \tag{LR4}$$

$$x_{sp} = 0 \text{ or } 1 \quad \forall s \in S, p \in P \tag{LR5}$$

$$z_{(i,j)}^s = 0 \text{ or } 1 \quad \forall s \in S, (i,j) \in L \tag{LR6}$$

$$t_{(i,j)}^{xy} = 0 \text{ or } 1 \quad \forall s \in S, (i,j) \in L \tag{LR7}$$

We can further decompose this LR problem into following four independent subproblems according to different decision variables.

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



3.2.1 Subproblem 1 (related to decision variable $t_{(i,j)}^{xy}$)

$$Z_{sub1}(u_{xy(i,j)}^2, u_{xy(i,j)}^3) = \min \sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in L} t_{(i,j)}^{xy} [r_{xy}(ua_{(i,j)} + d_j^w) + 2u_{xy(i,j)}^2 - u_{xy(i,j)}^3]$$

Subject to:

$$\sum_{j \in C} t_{(y,j)}^{xy} = 1 \quad \forall x, y \in S \quad (\text{sub1 1})$$

$$t_{(i,j)}^{xy} = 0 \text{ or } 1 \quad \forall x, y \in S, (i,j) \in L \quad (\text{sub1 2})$$

This subproblem is related to decision variable $t_{(i,j)}^{xy}$, which can further decomposed into $|S|^2 |L|$ subproblems.

The proposed algorithm for solving (sub 1) is described as follows:

Step1. For each link (i, j) , we compute the weight of the a pair of sensor node x and y using the link (i, j) , denoted by $\theta_{xy(i,j)} = [r_{xy}(ua_{(i,j)} + d_j^w) + 2u_{xy(i,j)}^2 - u_{xy(i,j)}^3]$.

Step2. If $\theta_{xy(i,j)} < 0$, then we assign $t_{(i,j)}^{xy}$ to 1; otherwise, we set $t_{(i,j)}^{xy}$ to 0.

Step3. If the sum of each pair node $t_{(i,j)}^{xy}$ is 0, we enforce to select the minimum positive objective value $\theta_{xy(i,j)}$ and set $t_{(i,j)}^{xy} = 1$ to fulfill the constraint (sub1 1).

3.2.2 Subproblem 2 (related to decision variable x_{sp})

$$Z_{sub2}(u_{s(i,j)}^1) = \min \sum_{s \in S} \sum_{(i,j) \in L} (u_{s(i,j)}^1 \sum_{p \in P} x_{sp} \delta_{p(i,j)})$$

Subject to:

$$\sum_{p \in P} x_{sp} = 1 \quad \forall s \in S \quad (\text{sub2 1})$$

$$x_{sp} = 0 \text{ or } 1 \quad \forall s \in S, p \in P \quad (\text{sub2 2})$$

Z_{sub2} can be further decomposed into $|S|$ independent shortest path problems with nonnegative arc weight $u_{s(i,j)}^1$. For each shortest path problem it can be solved by the Dijkstra's algorithm.



3.2.3 Subproblem 3 (related to decision variable $z_{(i,j)}^s$)

$$\begin{aligned} Z_{sub3}(u_{s(i,j)}^1, u_{xy(i,j)}^2, u_{xy(i,j)}^3) = & \min \sum_{s \in S} \sum_{(i,j) \in L} [(r_{os} + r_{so})(ua_{(i,j)} + d_j^w) + w_{(i,j)} Q_s T(va_{(j,i)} + d_i^r) \\ & + w_{(i,j)} Q_s T(ua_{(i,j)}) - u_{s(i,j)}^1 + \sum_{x \in S} (u_{xs(i,j)}^3 - u_{xs(i,j)}^2) \\ & - \sum_{y \in S} (u_{sy(i,j)}^3 - u_{sy(i,j)}^2)] z_{(i,j)}^s \end{aligned}$$

Subject to:

$$\sum_{j \in C} z_{(i,j)}^s \leq 1 \quad \forall s \in S, i \in S \cup C - \{sink\} \quad (\text{sub3 1})$$

$$\sum_{j \in C} z_{(s,j)}^s = 1 \quad \forall s \in S \quad (\text{sub 3 2})$$

$$z_{(i,j)}^s = 0 \text{ or } 1 \quad \forall s \in S, (i,j) \in L \quad (\text{sub3 3})$$

The proposed algorithm for solving (sub 3) is described as follows:

Step1. For each link (i,j) , we compute the weight of the sensor node s using the link (i,j) ,

$$\begin{aligned} \text{denoted by } \pi_{xy(i,j)} = & [(r_{os} + r_{so})(a_{(i,j)} + D_j) + V_{(i,j)} Q_s T(a_{(i,j)} + a_{(j,i)}) \\ & - u_{s(i,j)}^1 + \sum_{x \in S} (u_{xs(i,j)}^3 - u_{xs(i,j)}^2) - \sum_{y \in S} (u_{sy(i,j)}^3 - u_{sy(i,j)}^2)] \end{aligned}$$

Step2. If $\pi_{xy(i,j)} < 0$, then we assign $z_{(i,j)}^s$ to 1; otherwise, we set $z_{(i,j)}^s$ to 0.

Step3. For each link (s,j) , we enforce to select minimum $\pi_{xy(s,j)}$ and set $z_{(s,j)}^s = 1$ to

fulfill constraint (sub3 2).

3.2.4 Subproblem 4 (constant part)

$$Z_{sub4}(u_{xy(i,j)}^2) = - \sum_{x \in S} \sum_{y \in S} \sum_{(i,j) \in L} u_{xy(i,j)}^2$$

3.3 The Dual Problem and the Subgradient Method (IP)

By solving the subproblems using the algorithms proposed above, we could solve the Lagrangean relaxation problem efficiently and optimally. According the weak Lagrangean duality theorem, $Z_D(u_1, u_2, u_3)$ generate a Lower Bound (LB) of the primal solution Z_{IP} . We construct the following dual problem (D1) for tightening the lower bound and solve the dual problem by using the subgradient method.

Dual Problem (D1)

$$Z_D = \max Z_D(u_1, u_2, u_3) \quad (D1)$$

Subject to: $u_1, u_2, u_3 \geq 0$.

Let the vector s be subgradient of $Z_D(u_1, u_2, u_3)$ at (u_1, u_2, u_3) . In iteration k of the subgradient optimization procedure, the multiplier vector $m^k = (u_1^k, u_2^k, u_3^k)$ is updated by $m^{k+1} = m^k + \alpha^k S^k$.

The step size α^k is determined by $\delta \frac{Z_{IP}^k - Z_D(m^k)}{\|S^k\|^2}$, where Z_{IP}^k is the best primal

objective function value found by iteration k (an upper bound on the optimal primal objective function value), and δ is a constant ($0 \leq \delta \leq 2$).

Chapter 4 Getting Primal Feasible Solutions

4.1 Lagrangean Relaxation Results

After optimally solving our primal problem by applying Lagrangean Relaxation method and subgradient method, we can obtain a set of decision variables and a theoretical lower bound of the primal problem. Because some constraints are relaxed by using Lagrangean Relaxation method, we cannot guarantee that the result of the Lagrangean Relaxation problem is feasible to the primal problem. If the solution of Lagrangean Relaxation problem is infeasible, we need to make some modifications to transform the infeasible solution into a feasible one.

4.2 Getting Primal Feasible Solutions

The heuristic algorithm for constructing object tracking trees based on the solution in Subproblem 3. However, the union of the shortest paths for each sensor nodes may not be a tree in Subproblem 3, since each sensor node may have a different arc weight, $u_{s(i,j)}^1$, which may result in having a cycle for the union of the shortest paths. Therefore, we set the arc weight of link (i,j) to be $\sum_{s \in S} u_{s(i,j)}^1$. This ensures that the union of the shortest paths shall be a tree. After constructing an shortest path tree by using Dijkstra's algorithm with the modified arc weight, we can obtain the shortest path for each sensor node. Once the x_{sp} is determined, we can also obtain the value of $t_{(i,j)}^{xy}$ and $Z_{(i,j)}^s$.

A LR-based primal heuristic algorithm is listed in Table 3-1 and the complete object tracking tree algorithm is listed in Table 3-2.

Table 3-1 The LR-based primal heuristic algorithm.

Algorithm Primal_Heuristic

Step 1 Using the shortest path tree algorithm (SPT) to find the initial primal value.

Step 2 We adjust arc weight $c_{s(i,j)} = \sum_{s \in S} u_{s(i,j)}^1$ for each $(i,j) \in L$ and then run the

Dijkstra algorithm to get the solution set of $\{x_{sp}\}$.

Step 3 Once $\{x_{sp}\}$ is determined, $t_{(i,j)}^{xy}$ and $z_{(i,j)}^s$ are also determined.

Step 4 We can have an object tracking tree now, and then iteratively execute the Step 2~3 with LR multipliers that can be updated from dual mode problem.

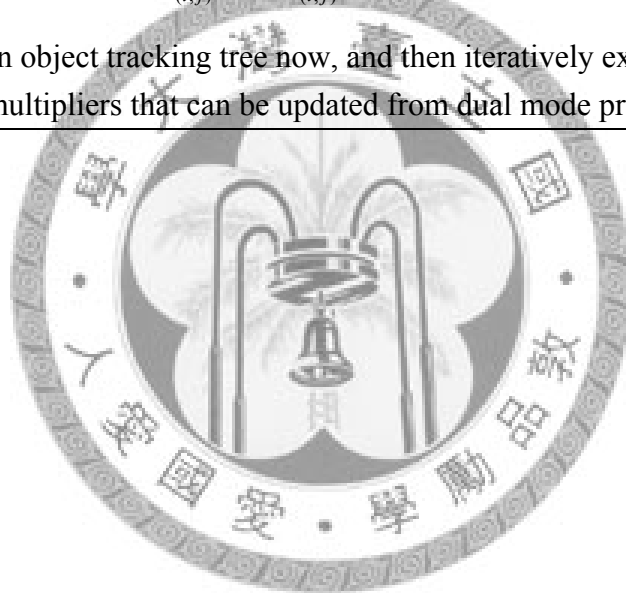


Table 3-2 The object tracking tree algorithm.

Algorithm Object_Tracking_Tree

begin

Initialize the Lagrangean multiplier vector (u_1, u_2, u_3) to be zero vectors;

UB:=total communication cost of shortest path tree;

LB:=very small number;

improve_counter:=0; step_size_coefficient:=2;

for *iteration:=1 to Max_Iteration_Number* **do**

begin

run sub-problem(SUB1);

run sub-problem(SUB2);

run sub-problem(SUB3);

run sub-problem(SUB4);

calculate Z_D ;

if $Z_D > LB$ **then** $LB := Z_D$ and *improve_counter:=0;*

else *improve_counter:=improve_counter+1;*

if *improve_counter=improve_Threshold*

then *improve_counter:=0; $\lambda := \lambda / 2$;*

run Primal_Heuristic Algorithm;

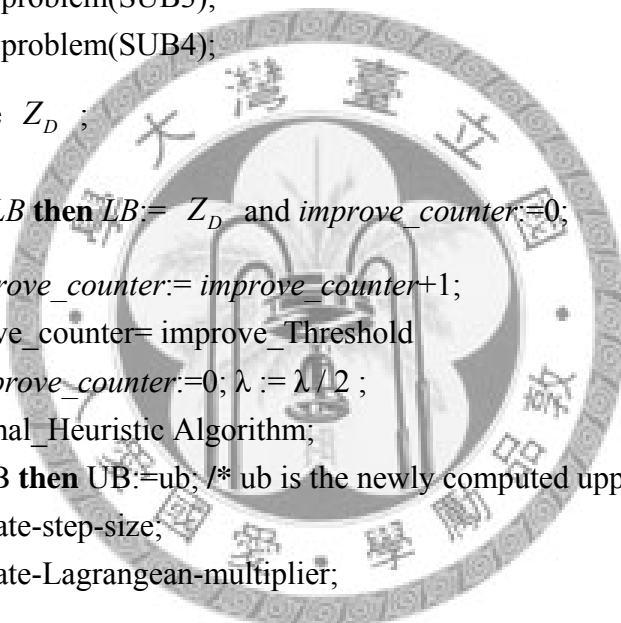
if $ub < UB$ **then** $UB := ub$; /* *ub is the newly computed upper bound* */

run update-step-size;

run update-Lagrangean-multiplier;

end;

end;



4.3 Simple Algorithms

In order to evaluate our proposed heuristic algorithm, we compare this one with other heuristic algorithm. We implement Dijkstra’s algorithm as Simple Algorithm 1 (SA1), and spanning tree-like algorithm as Simple Algorithm 2(SA2). Since the regular spanning tree algorithm, Kruskal, may incur the sensor node as a intermediate node. It violates our assumption. Therefore, we span all communication node included sink node by using Kruskal’s Algorithm, then find the shortest path to the spanning tree for each sensor node.

We will compare the solution quality of Lagrangean Relaxation based algorithm (LR) with these simple algorithms (SAs) in Chapter 5.



Chapter 5 Computational Experiments

5.1 Experiments Environment

In this chapter, we conduct several experiments with different parameters to evaluate the solution quality of our solution approach.

The proposed algorithm for constructing object tracking trees is coded in programming language C and executed on Windows XP and Visual C++ 6.0. The program is run on a notebook with Intel Core2 Duo 2.20G CPU and 2GB RAM.

The parameters listed in Table 4-1 are used for the all cases of experiments.

Table 4-1 Parameter of Lagrangean Relaxation based algorithm

Number of nodes	23~85 (depend on each case)
Number of iterations	10,000
Improvement counter	256
Initial upper bound	0
Initial scalar of step size	2
Initial Multiplier	0

According to different case, we randomly generate a number of communication nodes and sensor nodes in a square area. The power aware transmission cost, $a_{(i,j)}$, is define as the Euclidean distance.

We denote the dual solution as “ Zdu ”, and Lagrangean Relaxation based heuristic as “ ZIP ”. “ GAP ” is used to evaluate our solution quality. $GAP = \left| \frac{ZIP - Zdu}{Zdu} \right| * 100\%$.

5.2 Solution Quality

Table 4-2 shows the total transmission cost calculated by different algorithms under a different number of nodes, respectively. We can see that the heuristic proposed in Chapter 5 outperforms the other two simple algorithms. Although improvement ratio compared with simple algorithms 1 will reduce when the number of queries increase, the solution we found is absolutely better or equal than SA1. This occurs because when the number of queries is large, the total query cost will dominate the total update cos. Since the query mechanism is similar to the SPT, the gap between the LR-based algorithm and SA1 will reduce when the total number of queries increases. The reason why the gap will reduce will be illustrated later.

Table 4-2 Evaluation of gap (%) and improvements ratio by given different number of nodes and different query rate

Number of nodes		<i>Zdu</i>	<i>ZIP</i>	<i>Gap</i> (%)	SA1	Impr. Ratio to SA1 (%)	SA2	Impr. Ratio to SA2 (%)
23 nodes (U=960)	<i>T=0</i>	20255	23338	15.2	25546	9.46	25483	9.6
	<i>T=100</i>	23270	26425	13.5	28294	7.07	29960	12.4
	<i>T=200</i>	26284	29320	11.5	31043	5.88	34438	16.9
	<i>T=450</i>	33555	35802	6.7	37915	5.90	45632	27.5
	<i>T=960</i>	46481	48182	3.7	51933	7.78	68468	42.1
	<i>T=1400</i>	57039	58863	3.2	64027	8.77	92647	51.2
	<i>T=1800</i>	66631	68573	2.9	75022	9.40	106080	54.7
36 nodes (U=746)	<i>T=0</i>	7180	9816	36.7	12684	29.22	12436	26.7
	<i>T=100</i>	9718	12967	33.4	14936	15.19	18014	38.9
	<i>T=350</i>	15399	18607	20.8	20330	9.26	31959	71.8

	$T=746$	23487	27971	19.0	28873	3.23	54048	93.2
	$T=1100$	30397	35819	17.8	36511	1.93	78561	119.3
	$T=1400$	36058	42508	17.8	42983	1.12	90527	112.9
57 nodes ($U=2856$)	$T=0$	51057	68619	34.4	75030	9.34	84463	23.1
	$T=700$	79382	90725	14.2	97248	7.19	182191	100.8
	$T=1400$	106995	113410	6.0	119466	5.34	279920	146.8
	$T=2856$	152937	158661	3.7	165680	4.42	483195	204.5
	$T=4000$	186850	194327	3.3	201990	3.94	642911	230.8
	$T=6000$	251235	257897	2.7	265470	2.94	922135	257.5
87 nodes ($U=3726$)	$T=0$	58387	89410	53.1	94855	6.1	106002	18.56
	$T=1000$	96374	126349	31.1	131435	4.1	175987	39.29
	$T=2000$	128995	165047	27.9	168015	1.8	245977	49.03
	$T=3726$	178029	228062	28.1	231152	1.4	366766	60.82
	$T=5000$	209287	276362	32.0	277755	0.5	455929	64.97

In Table 4-2, we can find the duality gap is also small, which means we obtain near optimal solution in these cases. “ T ” means total query number and “ U ” means total update traffic. Since the heuristic algorithm we proposed is base on the SPT algorithm. When T becomes larger, the tree is much closer to the SPT; therefore, the duality gap will reduce when the “ T ” increases. Furthermore, the improve ratios shows that our algorithm is better than the other heuristics.

We summarize the above experiments results into diagrams and show them in Figure 5-1, Figure 5-2, Figure 5-3, and Figure 5-4.

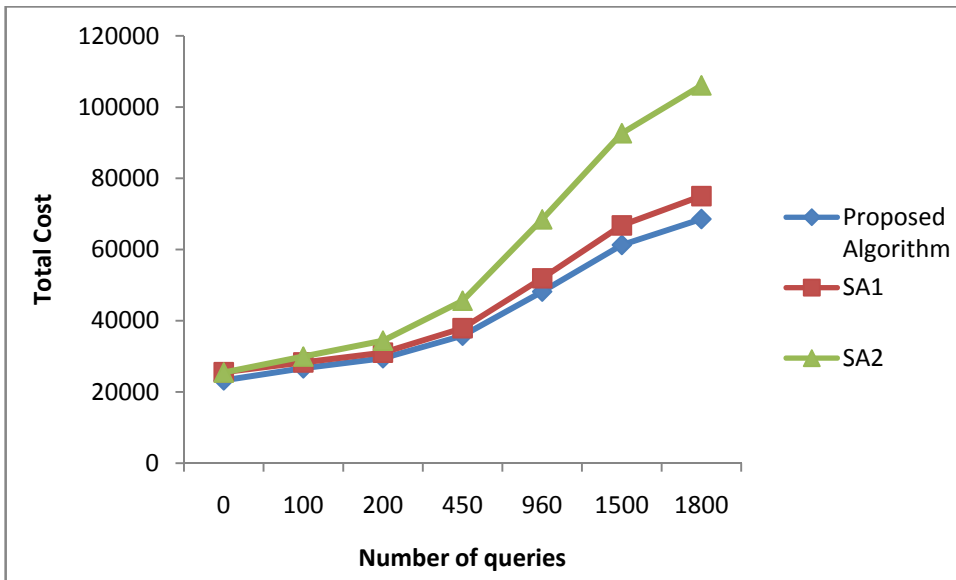


Figure 5-1. Evaluation of improve ratio to SA1 and SA2

(Number of Nodes = 23)

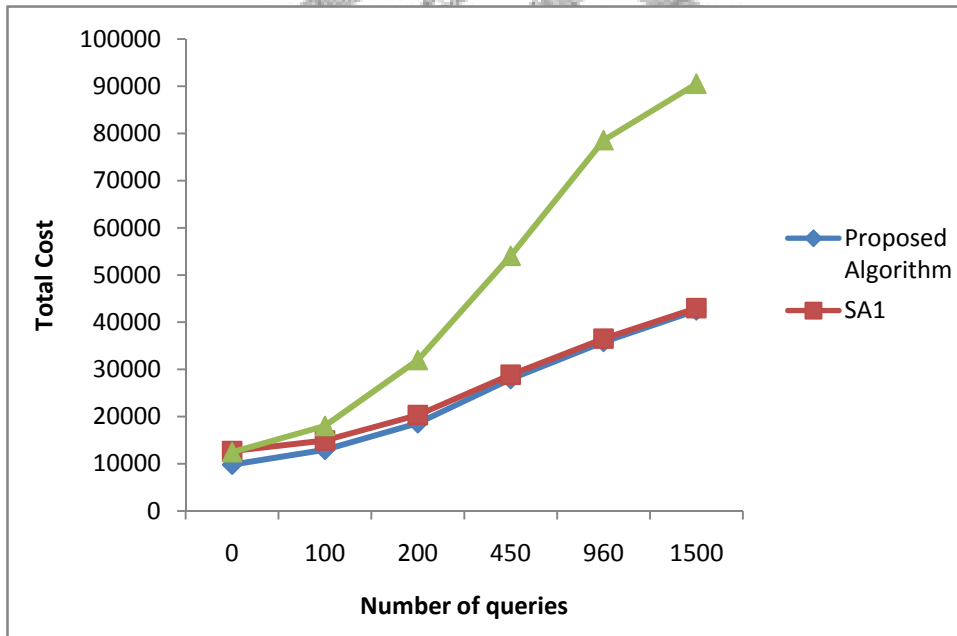


Figure 5-2. Evaluation of improve ratio to SA1 and SA2

(Number of Nodes = 36)

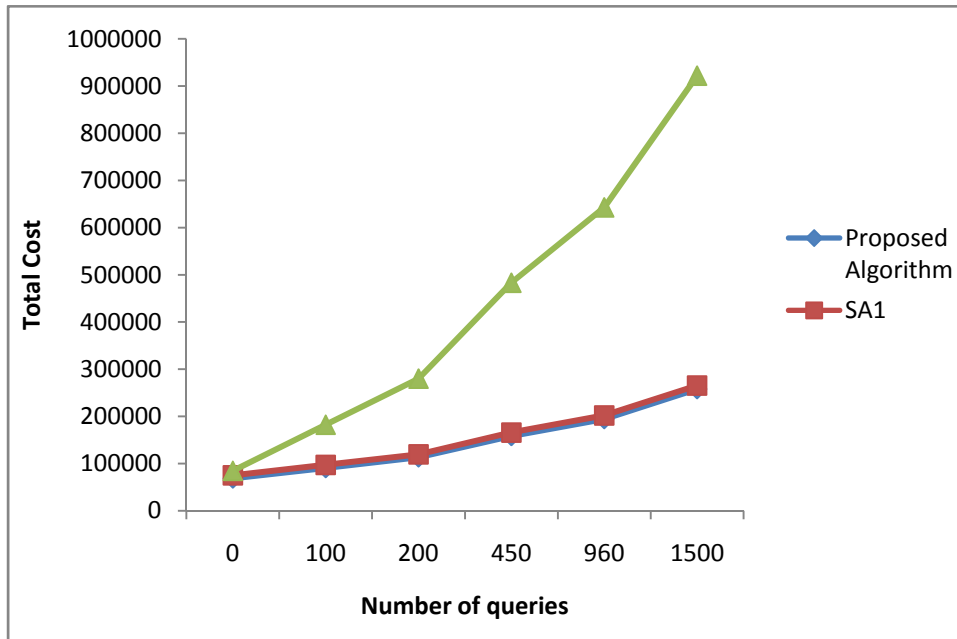


Figure 5-3. Evaluation of improve ratio to SA1 and SA2
(Number of Nodes = 57)

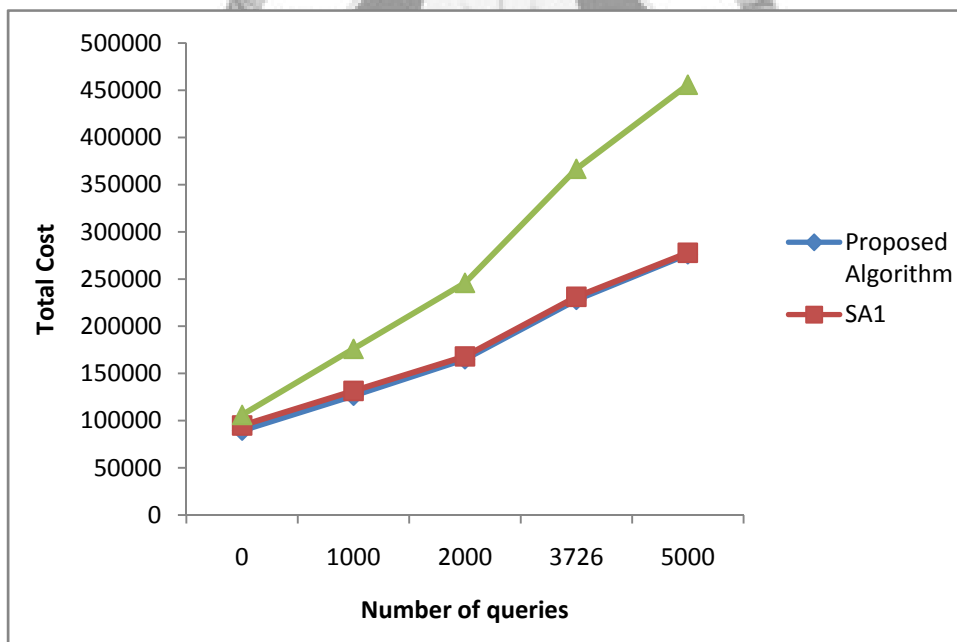


Figure 5-4. Evaluation of improve ratio to SA1 and SA2
(Number of Nodes = 87)

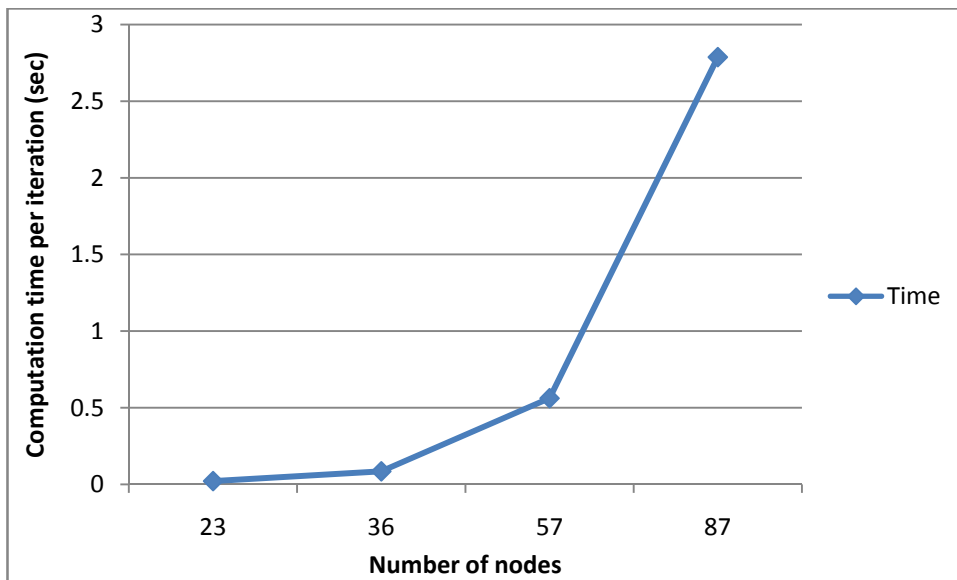


Figure 5-5. Computational time under different numbers of nodes

Figure 5-5 shows the computational time of the Lagrangean relaxation based algorithm per iteration.

Figure 5-6 shows an example of trend line for getting the primal problem solution values (UB) and dual mode problem values (LB). The UB curves tend to decrease to get the minimum feasible solution. In contrast, the LB curves tend 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 gaps as small as possible in order to improve our solution quality and achieve near optimization.

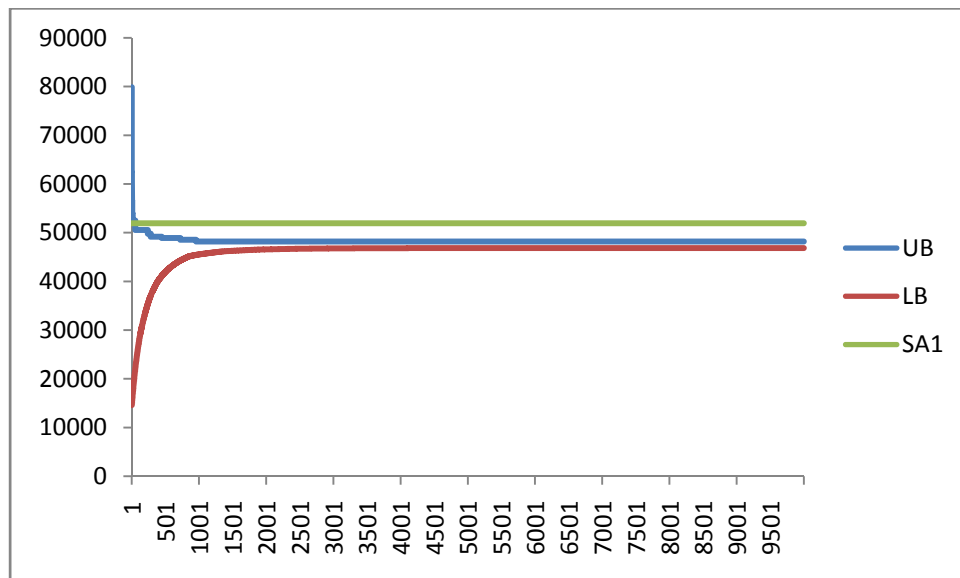


Figure 5-6. The execution result of LR based algorithm
(Number of Nodes = 23, Total Query Number = 960)

Figure 5-7 ~ Figure 5-12 are examples which the number of nodes is 23. Figure 5-7 is an original graph show the link status between any pair of nodes. Figure 5-8 is a shortest path tree which found by using simple algorithm 1, Dijkstra's algorithm. Figure 5-9 ~ Figure 5-12 show the object tracking trees found by using the proposed LR-based algorithm under different T .

Since most of the update usually happen on the link, which are farther from the sink; and most of the query happen contrarily on the links, which are closer to the sink. Yet, the tree structure formed by the latter links becomes much alike the tree structure formed by SPT, when T becomes larger. Thus, the larger the T , the smaller the improvement ratio.

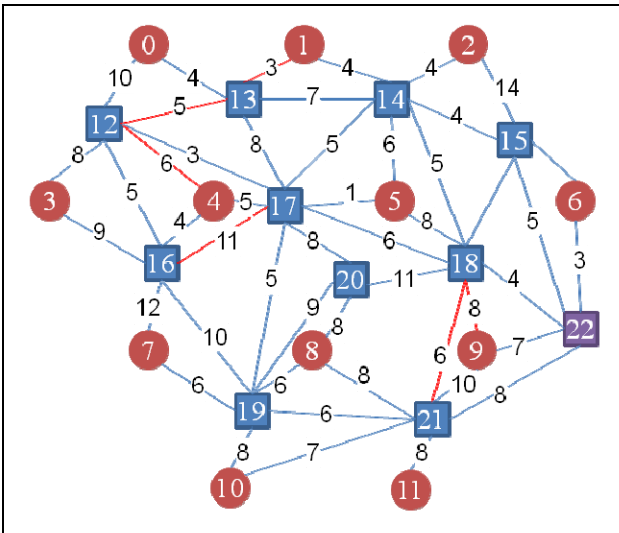


Figure 5-7. Example of 2D tracking sub-graph.

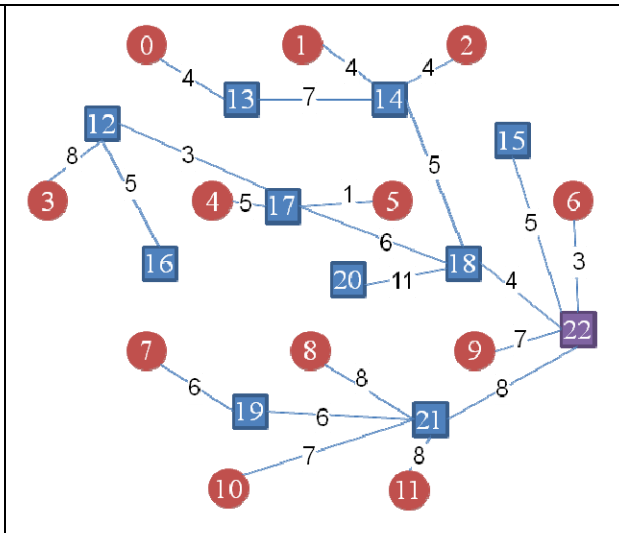


Figure 5-8. Example of an object tracking tree (shortest path tree)

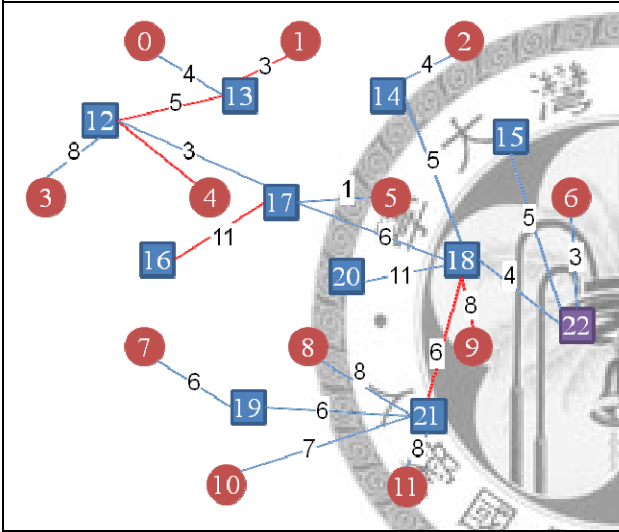


Figure 5-9. Example of an object tracking tree ($T=0$)

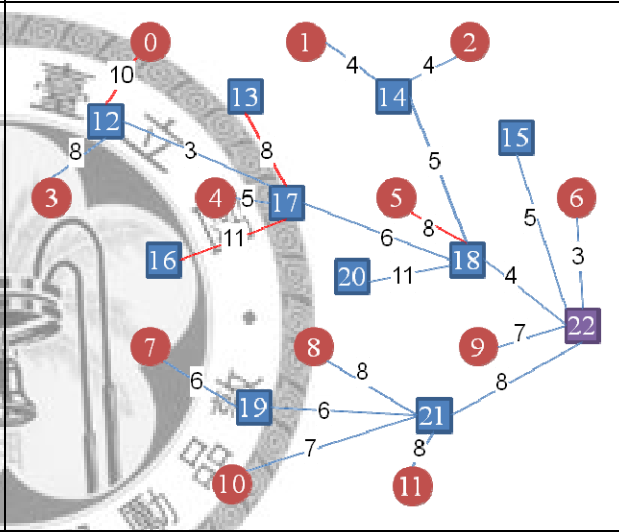


Figure 5-10. Example of an object tracking tree ($T=450$)

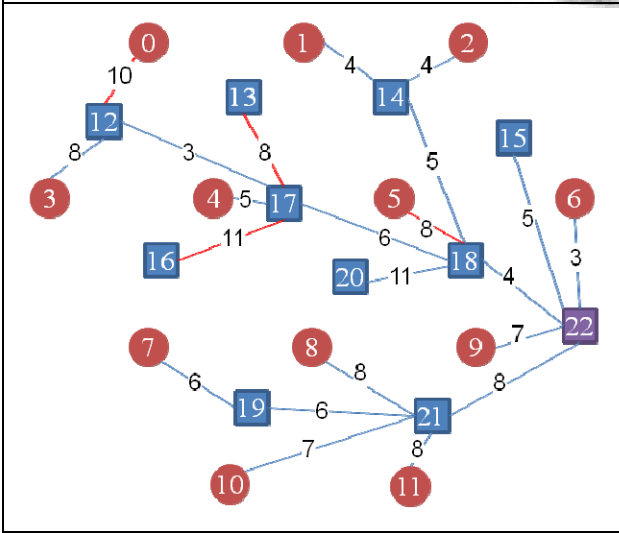


Figure 5-11. Example of an object tracking tree ($T=960$)

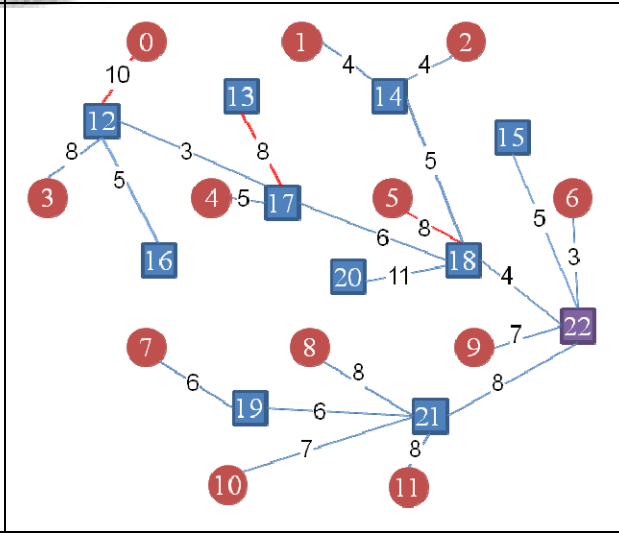


Figure 5-12. Example of an object tracking tree ($T=1920$)

Chapter 6 Conclusion and Future Work

6.1 Conclusion

Wireless sensor networks will become widespread due to the computation, sensing, and wireless communications capabilities. However, the energy awareness is an essential issue, since the battery level of the WSNs is fixed, limited and it is infeasible to recharge the battery, at least for now. By adopting detected list on object tracking tree, we can decrease the number of update messages and query messages and further reduce the total cost of the communication. We address the construction of an object tracking tree to maintain the detected list in order to track the objects efficiently. To solve this problem, we proposed a LR based algorithm to construct an object tracking tree for update and query with minimum cost in order to decrease the cost of communications and further prolong the system lifetime.

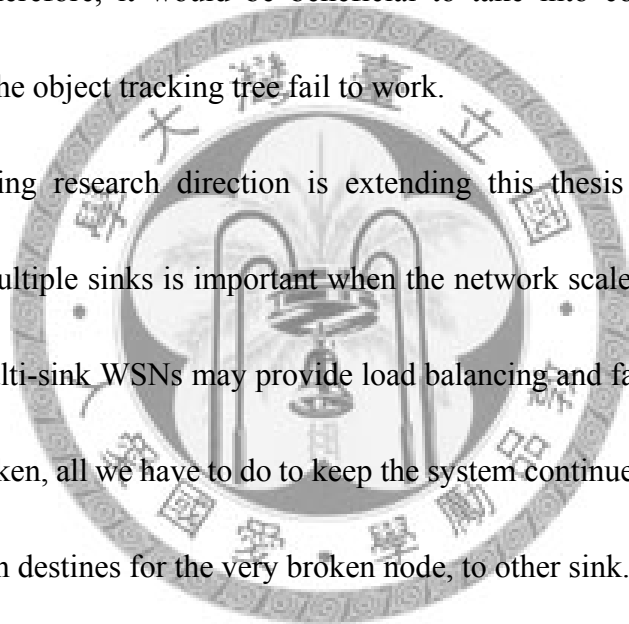
In this thesis, first, we propose a mathematical formulation to well model the object tracking tree construction problem as a 0/1 integer programming problem and apply Lagrangean Relaxation and subgradient method to solve it. Then, we design a heuristic approach to get feasible solution. Finally, we conduct several experiments in different cases. According to these experiments results, we can claim that our LR- based algorithm not only outperforms the other heuristics, such as SPT, but the duality gap is also small. The results shows that the proposed LR based algorithm can achieve energy-efficient object tracking.

6.2 Future Work

For the object tracking tree problem in wireless sensor networks, there are still several issues to be addressed.

In the thesis we consider the construction of an object tracking tree with several constraints. However, the communication nodes near the sink have higher probability to be used. As a result, these communication nodes may have less residual energy compared with the farther nodes. Therefore, it would be beneficial to take into consideration the load balancing to prevent the object tracking tree fail to work.

Another interesting research direction is extending this thesis from single-sink to multi-sink. Having multiple sinks is important when the network scale is large or when the query rate is high. Multi-sink WSNs may provide load balancing and failure tolerance. Once one of the sinks is broken, all we have to do to keep the system continue to work is to reroute the sensor node, which destines for the very broken node, to other sink.



References

- [1] A.M. Geoffrion, "Lagrangean Relaxation and its Use in Integer Programming," *Mathematical Programming Study*, vol. 2, pp. 82-114, 1974.
- [2] 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*, June 2008.
- [3] C.R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, July 1997.
- [4] C.T. Lee, F.Y.S. Lin, and Y.F. Wen, "An Efficient Object Tracking Algorithm in Wireless Sensor Networks," *Proc. JCIS'06*, 2006.
- [5] 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.
- [6] 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*, August 2006.
- [7] C.Y. Lin, Y.C. Tseng, T.H. Lai, and W.C. Peng, "Message-efficient In-network Location Management in a Multi-sink Wireless Sensor Network," *International*

Journal of Sensor Networks, 2008.

- [8] H. Yang and B. Sikdar, "A Protocol for Tracking Mobile Targets using Sensor Networks," *Proceedings of IEEE Workshop on Sensor Network Protocols and Application*, 2003.
- [9] H.T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," *Proceedings of 2003 IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.
- [10] H.H. Yen, F.Y.S. Lin, and S.P. Lin, "An Energy-Efficient Data-Central Routing Algorithm in Wireless Sensor Networks," *Proc. IEEE ICC*, 2005.
- [11] Henry Medeiros and Johnny Park, "Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks," *IEEE Journal of Selected Topics in Signal Processing*, 2008
- [12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: a Survey," *Elsevier Journal of Computer Networks*, vol. 38, pp. 393-422, March 2002.
- [13] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, pp. 102-114, August 2002.
- [14] L.H. Yen and C.C. Yang, "Mobility Profiling Using Markov Chains for Tree-Based Object Tracking in Wireless Sensor Networks," *Proc. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing - vol 2*

- *Workshops*, June 2006.

- [15] M.L. Fisher, "An Application Oriented Guide to Lagrangean Relaxation," *Interfaces*, vol. 15, no. 2, pp. 10-21, April 1985.
- [16] M.L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 27, no. 1, pp. 1-18, 1981.
- [17] J. Winter, Y. Xu, and W.C. Lee, "Prediction Based Strategies for Energy Saving in Object Tracking Sensor Networks," *Proceedings of IEEE International Conference on Mobile Data Management (MDM 04)*, Jan. 2004.
- [18] S. Banerjee and S. Khuller, "A clustering scheme for hierarchical control in multi-hop wireless networks," *Proceedings of IEEE INFOCOM*, April 2001.
- [19] Y.F. Wen, F.Y.S. Lin and W.C. Kuo, "A Tree-based Energy-efficient Algorithm for Data-Centric Wireless Sensor Networks," *Proc. IEEE AINA*, 2007.



簡 歷

姓 名：許 宴 毅

出 生 地：台 北 縣

出 生 日：中 華 民 國 七 十 四 年 三 月 十 二 日

學 歷：九 十 四 年 九 月 至 九 十 六 年 六 月
國 立 台 灣 科 技 大 學 資 訊 管 理 學 系

九 十 六 年 九 月 至 九 十 八 年 六 月
國 立 台 灣 大 學 資 訊 管 理 研 究 所

