國立交通大學

資訊科學與工程研究所

碩 士 論 文

無線感測網路探勘群集式物體移動路徑機制

Group-Based Object Tracking Sensor Networks:
Exploiting Group Moving Patterns

研 究 生：黃正和

指導教授：彭文志　教授

中 華 民 國 九 十 六 年 六 月

無線感測網路探勘群集式物體移動路徑機制
Group-Based Object Tracking Sensor Networks: Exploiting Group Moving Patterns

研 究 生：黃正和　　Student：Cheng-Huom Huang

指導教授：彭文志　　Advisor：Wen-Chih Peng

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

摘　　要

在無線感測網路中利用預測技術來追蹤物體的移動路徑可以減少能源的耗費。在之前的研究中，我們透過探勘物體的移動模式來預測無線感測網路中物體的移動路徑並且發展階層式的架構來有效率地追蹤物體。感測節點本質上具有儲存空間的限制。然而，儲存每一個物體的移動樣式需要消耗感測節點大量的儲存空間，這會增加實際應用的難度。因此，在本篇論文我們利用群組移動模型的特色，提出了在具儲存空間限制的無線感測感網路中進行群集式物體追蹤的應用，簡稱 GBOT。首先，我們制定物體移動樣式之間的相似度，在此移動樣式被表示成散發樹。根據所制定的相似度，我們可以推知物體間的相似度關係。在給定物體間的相似度關係，我們進一步地提出分群演算法，將具有相似移動模式的物體分成一個群組。之後我們會為每一個群組選出最具代表性的散發樹並且使用這個散發樹來預測群組的移動路徑。此外，當物體的移動行為改變時，我們也設計了一個演算法來維持無線感測網路的預測準確率。實驗結果顯示 GBOT 應用於具儲存空間限制的無線感測網路中，不僅有效地減少儲存的成本並且有著極佳的預測準確率。

關鍵字：物體追蹤，無線感測網路，群組移動模型，分群。

Abstract

Predication-based techniques are able to reduce the energy consumption in object tracking sensor networks. Prior works exploit mining object moving patterns for prediction-based object tracking sensor network and developed a hierarchical architecture to efficiently track objects. Note that sensors are inherently storage-constrained. Clearly, mining and storing individual object moving patterns unavoidably need a considerable amount of storage spaces in sensor nodes, which is not of practical. Thus, in this paper, we propose a group-based object tracking sensor network (abbreviated as GBOT) which explores the feature of group mobility of objects for storage-constrained object tracking sensor networks. Specifically, we first formulate a dissimilarity function among object moving patterns, where object moving patterns are viewed as emission trees. In light of the dissimilarity function, the dissimilarity relationships among objects are derived. Given dissimilarity relationships among objects, we further propose two clustering schemes to discover group mobility patterns of objects. Furthermore, for each group, we judiciously select one representative emission tree and utilize this emission tree for prediction. In addition, a maintenance algorithm is derived to preserve the prediction accuracy when moving behaviors of objects vary. Experimental results show that GBOT not only effectively reduces storage cost but also has a good prediction accuracy in storage-constrained sensor networks.

Keywords — Object tracking, sensor networks, group mobility, clustering

致　　謝

經歷了多次反覆的討論與修改，本篇論文終於得以成形定稿。對於研究投注的心力，已經不是三言兩語能道盡。首先要感謝我的指導教授彭文志教授，從一開始教導我研究的方法，到後來論文的產生，一直親身參與。因為有他不時的提醒和督促，使得我在研究的過程中才不致鬆懈；同時我也從他的身上學習了人與人之間的相處之道。其次要感謝我的口試委員曾煜棋教授與曾新穆教授，在口試的時候，提供了許多寶貴的意見，使得本篇論文的部分缺失獲得改善的機會。

實驗室的日子，深深地影響了這兩年的研究生活。博士班學長洪智傑，在實驗室中是我們的大師兄，也是我們的開心果。感謝你兩年來的情義相挺，不僅傳授了有用的生活經驗，在研究的過程中亦不吝與我互相討論，對於本篇論文的貢獻良多。碩士班學長李志劭、張民憲、蕭向彥、楊慧友，在我剛進實驗室時，帶領我很快的習慣學校的大小事，感謝你們的提攜，還記得那些挑燈夜戰的日子，我總是很不好意思的第一個離開，你們對於研究的熱情令我印象深刻。碩士班同學游敦皓、李柏逸、周佳欣，總是一起修課、吃飯、打球，感謝你們的陪伴，讓我的研究所生活在歡樂的氣氛中渡過。碩士班學弟蔡尚樺、駱嘉濠、傳道揚、蔣易杉，很高興能認識你們，希望你們能一直保持積極的研究態度。感謝實驗室的大家，你們所帶給我快樂和溫暖林林總總。在此，預祝老師順利完成手邊的研究跟計畫、大師兄通過資格考、當兵的學長們平安退伍、還在努力的同學們口試成功以及學弟們有好的研究成果。

最後要感謝我的父母，始終站在我的背後支持我，給予我莫大的精神力量。因為有你們的付出，讓我能無後顧之憂的專注於研究，真的很感謝你們為我作的一切。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Object tracking is one of the killer applications for wireless sensor networks. In object tracking sensor networks (referred to as OTSN), a large number of static sensor nodes are deployed over a monitored region. There are access points (or called sinks) serving as interface for injecting queries and collecting tracking results (e.g., the location of objects). Sensor nodes obtain up-to-date location data of objects with a given sampling frequency. Data is reported to the sink via multi-hop communications according to a required reporting frequency. Typically sensor nodes use small batteries as their power source. As a result, energy conservation in object tracking sensor networks is a primary research issue to tackle.

Various energy conservation schemes for object tracking sensor networks have been extensively studied in the literature [20, 21, 12, 16, 17]. In particular, a predication-based OTSN is shown to be very energy-efficient for tracking objects. Explicitly, in predication-based OTSN only those sensor nodes whose sensing regions are likely to contain tracking objects are active, whereas the rest of sensor nodes are put in sleep mode to conserve energy. Clearly, predication-based object tracking sensor networks typically relies on certain prediction mechanisms to achieve energy saving. Prior works in [12] has proposed an in-network mining object moving patterns in a hierarchical tracking model (abbreviated as HTM) in which a large number of inexpensive sensor nodes perform sensing operations and a limited number of powerful sensor nodes (standing for cluster heads) offer data collection, queries and mining capabilities. Specifically, HTM exploits object moving patterns to predict the next positions of moving objects. Object moving patterns in HTM are represented as emission trees and stored at cluster heads. Though HTM has better prediction accuracy, storing emission trees for each object needs a considerable amount of storage space. Cluster heads are intrinsically storage-constraint sensors as well. Hence, how to exploit limited storage space of cluster heads efficiently is an important issue.

Generally speaking, many creatures usually have group movement behaviors [2, 6, 7, 14]. To deal with the problem mentioned above, in this paper, we propose the group-based OTSN framework (referred to as GBOT) in HTM. objects tend to move together in a gregarious fashion and consequently have similar emission trees. The main idea is that objects with similar moving behaviors (i.e., similar emission trees) form a group. For a group, we only keep an emission tree and the objects belonging to the same group share one representative emission tree. Therefore, the total number of emission trees to be stored decreases.

To perform GBOT, we first divide objects into groups in which objects would move similarly. To distinguish the moving behavior of objects, we adopt both the emission trees and up-to-date locations. According to the properties of the emission tree and up-to-date location, we define the dissimilarity and spatial proximity respectively. Regarding the problem of grouping objects, the grouping problem, we reduce the clique cover problem to the grouping problem by the dissimilarity measures. Then, two clustering schemes are formulated to group objects reactively or proactively. After grouping objects, we provide two metrics to judiciously select the representative emission tree for each group so that the storage cost can be further reduced and the prediction accuracy can be preserved. Note that the VMM model is originally designed for one object in HTM. In GBOT, we have to modify the VMM model to suit groups which are composed of multiple objects. Training the representative emission tree for a group is more complicated than for an object. If the VMM model is not well modified, multiple objects share one representative emission tree at the risk of lowering prediction accuracy. Lower prediction accuracy causes more recovery procedures and consumes more energy in the network. Therefore, we develop a group VMM model to train the representative emission trees well. In addition, it is necessary to observe the variation in moving behavior of intra-group and inter-group. An object needs to be split from the group to avoid wrong prediction when it escapes from its group. One group is also required to merge with another group to reduce the storage cost when they become similar in moving behavior. In this case, we propose a maintenance algorithm to periodically check the variation. To evaluate the performance of GBOT, a series of experiments are conducted to show that GBOT not only effectively reduce the storage cost but preserve the prediction accuracy.

A significant amount of researches [1, 3, 4, 5, 11, 12, 13, 15, 16, 17, 18, 20, 21, 22] have focused on the issue of using sensor networks to track objects. In [1], a simple tracking scheme based on binary sensor model is proposed. Using the minimalist sensors, each sensor's value is converted to only one bit of information. Prior works [3, 15, 18] use a cluster-based approach so that a cluster head can collect data from its slave sensors and generate the localization results. The authors in [4, 22] adopt a information-driven approach in which a leader sensor node determines which sensors should be selectively turned on. In [20, 21], a tree-based approach is presented to facilitate sensor nodes collaborating in detecting or tracking an object. Although prediction-based scheme has been refereed in [18, 20], the historical movements of an object is not taken into account carefully. In [16, 17], the authors propose a simple prediction model based on an observation that object movement usually remains constant for a certain period of time. In [12], the moving patterns of an object are mined by HTM to predict the future movements according to the historical movements of an object. In [13], the authors utilize the characteristic of the group movement of objects to achieve energy conservation in OTSN. Besides the tracking techniques, the trade-offs between power conservation and quality of surveillance are studied in [5, 11]. Different to previous works, we consider the objects with gregarious, periodic moving behaviors and propose the framework GBOT to perform group-based object tracking for HTM.

The rest of the paper is organized as follows. In Section 2, the overview of HTM, the storage overflow problem and the group-based object tracking sensor networks are presented. The details of the group-based object tracking sensor network are described from Section 3 to Section 6. Experimental results are shown in Section 7. Section 8 concludes with this paper.

Chapter 2

Preliminary

In this section, we first describe the overview of Heterogeneous Tracking Model (HTM) in Section 2.1. Then, the storage overload problem in HTM is presented in Section 2.2. Finally, an overview of group-based object tracking sensor network is proposed.

2.1 Overview of Heterogeneous Tracking Model

HTM consists of a large number of inexpensive sensor nodes that perform sensing operations and a limited number of powerful sensor nodes (standing for cluster heads) that offer data collection, queries and mining capabilities. Generally speaking, cluster heads have powerful computing capability and larger storage space. By exploring heterogeneous sensor nodes and the hierarchical feature, HTM not only provides in-network mining mechanism but also utilizes mining results for location predictions. Essentially, execution of HTM consists of two phases: (1)data collection and mining phase: In the data collection and mining phase, cluster heads collect the positions of objects and mines object moving patterns. In the beginning, low-end sensors and cluster heads turn on their power to monitor objects. At the mean time, cluster heads use current moving patterns to predict the next position of objects. (2)prediction phase: Once discovering that the prediction rate for objects is higher than a given threshold, cluster heads will be in the prediction phase and start to predict the next location of objects. This phase is basically to utilize predication-based OTSN to reduce energy consumption of sensors.

2.1.1 Data Collection and Mining Phase

To facilitate collaborative data collection processing in object tracking sensor networks, the hierarchical cluster architecture is exploited in which sensors are organized into clusters, where each cluster consists of a cluster head and low-end sensors. For simplicity, the monitored region is divided into grids and each low-end sensor is responsible for one grid. Figure 2.1 shows an example of three-level hierarchical cluster architecture with 4*4 grid structure. In Figure 2.1, there are one level-1 cluster head, four level-0 cluster heads and 16 low-end sensors in the monitored region. Each level-0 cluster head is responsible for corresponding subregion, i.e. one-fourth monitored region. Assume that low-end sensors and cluster heads have unique sensor identifications and these sensor nodes are well time-synchronized. Suppose that each low-end sensor is a logical representation of a set of sensor nodes which collaboratively detect an object. Given a sampling frequency, low-end sensors sense and report the sensing information to the sink. When a low-end sensor detects an object, this low-end sensor will inform the corresponding cluster head of the detected object identification, object arrival time and its sensor identification. In our work, object locations are represented as sensor identifications. Consequently, the movements of an object is viewed as a stream, which is composed of a series of sensor identifications.

Since the movements of objects have high dependence relationships, cluster heads in HTM thus adopt variable memory Markov model (referred to as VMM) to discover object moving behavior. In particular, given a stream of movements, a suffix tree, called *emission tree* [19], is proposed to mine moving behavior of objects. Specifically, each edge of an emission tree represents a moving record (i.e., sensor id) appearing in the moving path. A tree node of an emission tree is denoted as a concatenation of the edge

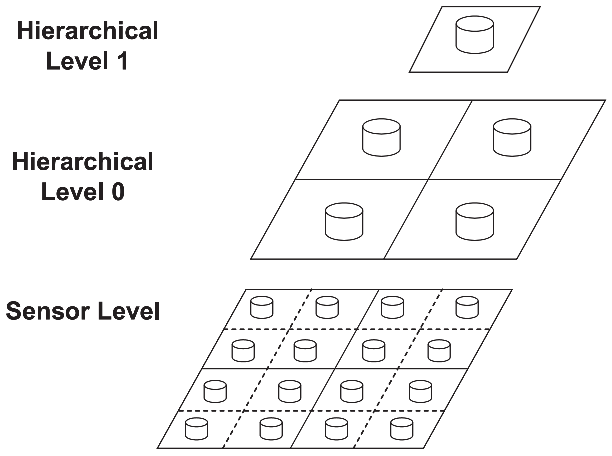


Figure 2.1: Architecture of Heterogeneous Tracking Model

labels from the node to the root. In other words, a tree node labeled as $r_k...r_2r_1$ can be reached from the traversal path from *root* $\rightarrow r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_k$. In an emission tree, each tree node will maintain a conditional table to record the appearing counts and the conditional probabilities of all appeared labels that follow its label. Initially, the emission tree has only a root node. Whenever a new moving record is generated, the corresponding label will be put into the buffer. According to the labels in the buffer, the conditional tables of tree nodes will be updated. If the appearing count of the label r_i is larger than the minimal support, one child node labeled as r_i will be inserted into the emission tree. Consider an illustrative example in Figure 2.2, where there are 16 low-end sensors deployed in the coverage region of cluster head CH and the object has 3 moving paths. The corresponding labels of moving records are put into the buffer. The label "*" means that there is no sensor node reporting the detection of this object. Figure 2.3 shows the evolution of the object's emission tree during the process of 15 moving records. In Figure 2.3(a), the emission tree has only a root node after receiving 5 moving records. In Figure 2.3(b), after receiving 11 moving records, node A and node B are inserted since the appearing counts of label A and label B are larger than minimal support (e.g., 2) according to the conditional table of root node. In Figure 2.3(c), node AB is inserted because the appearing count of label B is also larger than minimal support according to the conditional table of node A. Figure 2.3(d) shows the resulting emission

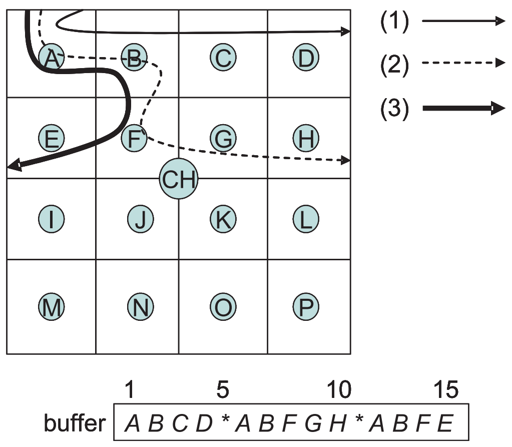


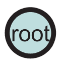
Figure 2.2: An illustrative example for moving record collection.

tree when the cluster head receives the 15th moving record.

2.1.2 Prediction Phase

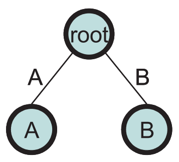
If an object is within the coverage region of level-0 cluster head, the corresponding cluster head should predict the next movement of the object. Note that there are two kinds of nodes: mature node and immature node. A mature node has more sufficient statistical information and is used for prediction. On the other hand, an immature node stills need more moving records to have more stable statistical information. According to the most recent moving records of the buffer received at this cluster head, the cluster head will traverse the emission tree to predict possible next movements. Consider an example in Figure 2.4, where nodes with dash circles are immature nodes and nodes with solid circles are mature nodes. Given a recent moving records (i.e., DCAEF), a cluster head traverses the emission tree and reaches the node AEF. Since node AEF is not a mature node, the cluster head should not use the conditional table of node AEF for prediction. The nearest mature node is EF and from the conditional table of node EF, the next movement is D since D has a higher probability.

(a) buffer A B C D *



Node root		
SID	***Count***	***C. Prob.***
A	1	0.25
B	1	0.25
C	1	0.25
D	1	0.25

(b) buffer A B C D * A B F G H *

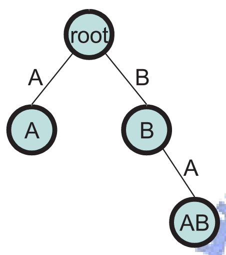


Node root		
SID	***Count***	***C. Prob.***
A	2	0.22
B	2	0.22
C	1	0.11
D	1	0.11
F	1	0.11
G	1	0.11
H	1	0.11

Node A		
SID	***Count***	***C. Prob.***
B	1	1

Node B		
SID	***Count***	***C. Prob.***
F	1	1

(c) buffer A B C D * A B F G H * A B



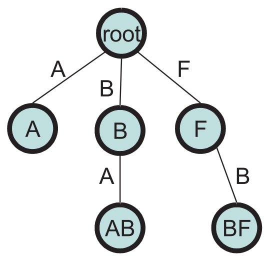
Node root		
SID	***Count***	***C. Prob.***
A	3	0.27
B	3	0.27
C	1	0.09
D	1	0.09
F	1	0.09
G	1	0.09
H	1	0.09

Node A		
SID	***Count***	***C. Prob.***
B	2	1

Node B		
SID	***Count***	***C. Prob.***
F	1	1

Node AB		
SID	***Count***	***C. Prob.***
-	-	-

(d) buffer A B C D * A B F G H * A B F E



Node root		
SID	***Count***	***C. Prob.***
A	3	0.23
B	3	0.23
C	1	0.07
D	1	0.07
E	1	0.07
F	2	0.15
G	1	0.07
H	1	0.07

Node A		
SID	***Count***	***C. Prob.***
B	2	1

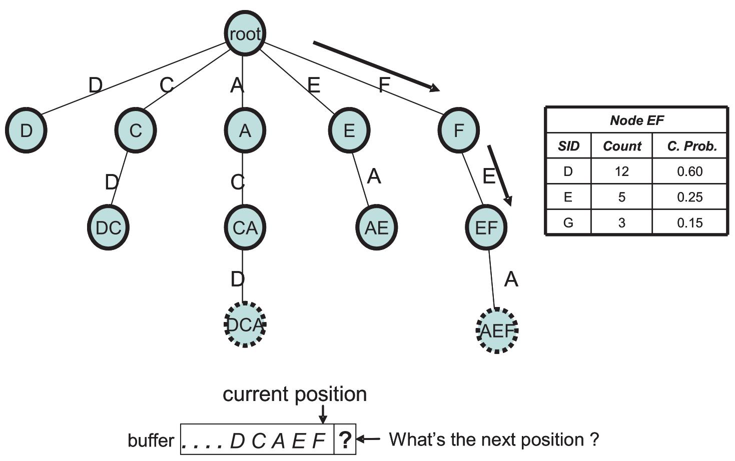
Node B		
SID	***Count***	***C. Prob.***
F	2	1

Node AB		
SID	***Count***	***C. Prob.***
F	2	1

Node F		
SID	***Count***	***C. Prob.***
E	1	1

Node BF		
SID	***Count***	***C. Prob.***
E	1	1

Figure 2.3: The evolution of an emission tree after 15 moving records.



Node EF		
SID	*Count*	*C. Prob.*
D	12	0.60
E	5	0.25
G	3	0.15

Figure 2.4: An example of utilizing emission trees for prediction, where nodes with dash circles are immature and nodes with solid lines are mature.

2.2 Storage Overflow Problem in HTM

Prior works have shown that exploring emission trees for mining object moving patterns are very effective in that emission trees fully captures dependencies of movements. The dependence relationship has a great impact on the location prediction. Through emission trees, one could traverse emission trees and utilize conditional tables to estimate the next movement of an object. Experimental results in [12] show that exploring emission trees for prediction significantly outperforms other heuristic prediction strategies in [17]. Although emission tree can good prediction accuracy, a considerable storage space is needed at cluster heads. To verify this claim, we simulate both periodic and random moving behaviors of one object, and then calculate the storage space required. Figure 2.5 shows the storage space of an object. As time goes by, a storage cost for one emission tree significantly increase. Once the number of objects tracked increases, a huge amount of storages are needed, thereby resulting in the overflow of storages in storage-constrained sensors. Though cluster heads are more powerful sensors, cluster heads are still constrained by storage spaces. Thus, it is important to reduce the storage cost of emission trees.

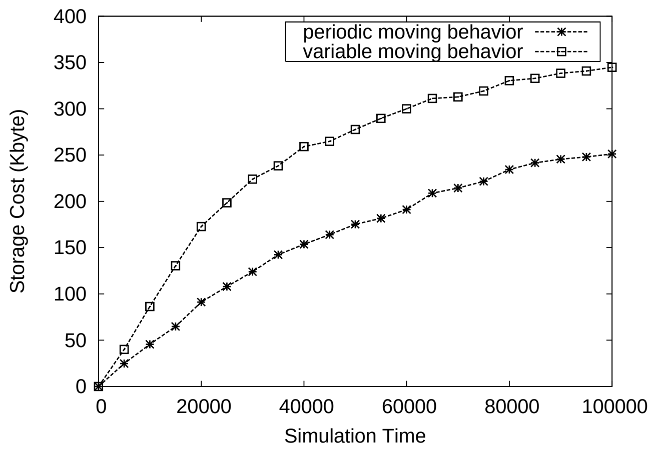


Figure 2.5: An experimental result of storage cost for emission trees.

2.3 Overview of Group-Based Object Tracking Sensor Networks

To deal with the storage overflow problem mentioned above, we propose group-based object tracking sensor networks (abbreviated as GBOT). As pointed out early, many creatures have group moving behaviors. Thus, objects with similar moving behaviors should be grouped together, and only one emission tree is maintained for each group. Clearly, by clustering objects with similar moving behaviors, storage costs of cluster heads are reduced. Specifically, in the beginning, GBOT performs data collection and mining phase function as usual in HTM. Then, GBOT performs the following steps:

Step 1: Clustering objects with similar moving behaviors

In this step, we first define the dissimilarity among emission trees. Then, in light of dissimilarity among emission tree, we develop two clustering schemes to group objects with similar moving behaviors.

Step 2: Selecting one representative emission tree for each group

Once objects are clustered into several groups, we should determine the representative emission tree for each group. By traveling representative trees, one could not only achieve the predication accuracy but also reduce the storage cost.

Step 3: Training representative emission trees

In GBOT, each group has its own representative emission tree and the cluster heads use representative emission trees to predict next movements of objects. At the same time, in order to continue mine object moving patterns for each object, we develop a mechanism to train representative emission trees with multiple buffers that are used for keep recent moving records of objects. Note that creatures may change their moving behaviors. Thus, we further propose a maintenance algorithm to dynamically adopt groups to reflect the changes of moving behaviors.

Chapter 3

Clustering Objects with Similar Moving Behaviors

In this section, we first formulate dissimilarity among emission trees in Section 3.1. Based on the dissimilarity derived, we propose two clustering schemes in Section 3.2.

3.1 Dissimilarity between Emission Trees

Each object has its own emission tree and in order to clustering objects with similar moving behaviors, we should first define dissimilarity measurements of emission trees. The following sections are two dissimilarity measurements used in GBOT.

3.1.1 Dissimilarity based on Label Set

In emission trees, labels of nodes represent spatial areas that objects frequently stay. Hence, the dissimilarity of emission trees is determined based on label sets of emission trees. Clearly, given two emission trees, a larger number of nodes that have the same labels, the more similar these two emission trees are. Therefore, each emission tree is transformed into one label set that includes labels of all nodes in an emission tree. For example, the label set of T_1 in Figure 3.2 is $\{A, B, AB\}$. Given two emission trees T$_i$

and T$_j$, the dissimilarity δ_{LS} between two emission trees, , is defined as follows:

$$\delta_{LS}(T_i, T_j) = \frac{|(LS_{T_i} \cup LS_{T_j}) - (LS_{T_i} \cap LS_{T_j})|}{|LS_{T_i} \cap LS_{T_j}|},$$

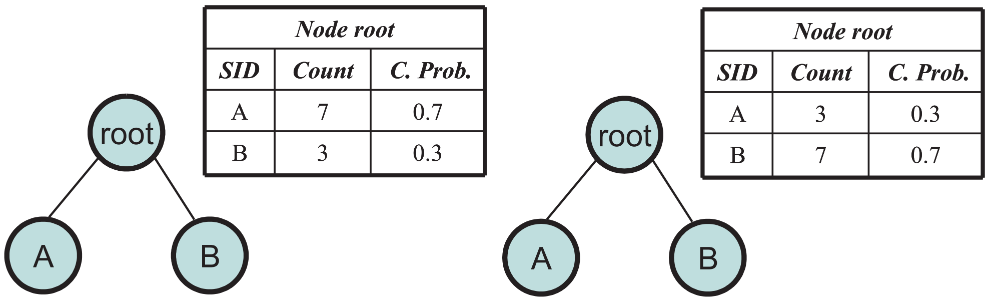
where LS_{T_i} denotes the label set of T_i.

Note that by exploring the intersection and the difference features, δ_{LS} is able to reflect spatial dissimilarity of two emission trees. Explicitly, in δ_{LS}, the intersection of two label sets represents how many nodes with same labels two emission trees have and the difference of two label sets demonstrates how many nodes with different labels two emission trees have. Clearly, when two emission trees have the same label sets, the dissimilarity of these two emission trees are zero, meaning that moving behaviors of the corresponding objects have exact the same moving patterns in terms of spatial areas. Consider emission trees T_1, T_2 and T_3 in Figure 3.2. It can be verified that $\delta_{LS}(T_1, T_2) = \frac{6-1}{1} = 5$, and $\delta_{LS}(T_1, T_3) = \frac{5-1}{1} = 4$, respectively. where LS_{T_1}, LS_{T_2} and LS_{T_3}. Thus, T_1 is more similar to T_3 than T_2, which agrees with the intuition observed from their emission trees.

δ_{LS} only determines dissimilarity measurement in terms of spatial information of emission trees. Note that though two emission trees have exact the same number of nodes with the same labels, these two emission trees may represent different moving behaviors in terms of temporal information. For example, Figure 3.1 shows that two emission trees with the same number of tree nodes having the same labels. However, these two emission trees reflect two different moving behaviors since conditional probabilities alone with trees nodes are not exact the same. In order to accurately formulate dissimilarity of moving behaviors, both spatial and temporal information hidden in emission trees should be taken into consideration. This is the call for the design of δ_{MSL}.

3.1.2 Dissimilarity based on Moving Sequence List

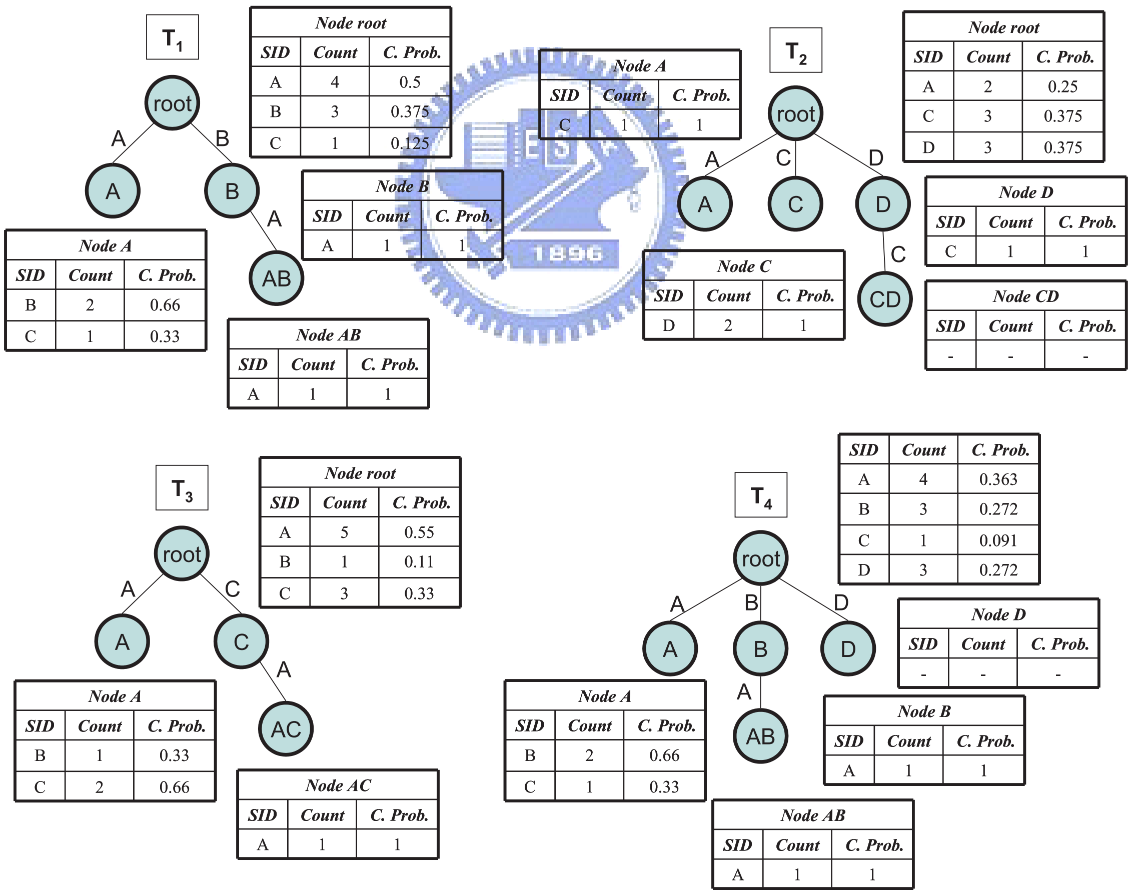
Note that emission trees reflect moving behaviors of objects. By traveling emission trees from the root node to leaf nodes, moving paths of objects are extracted. These mov-



Node root		
SID	*Count*	*C. Prob.*
A	7	0.7
B	3	0.3

Node root		
SID	*Count*	*C. Prob.*
A	3	0.3
B	7	0.7

Figure 3.1: An example of tree nodes with the same label but different conditional probability.



T_1:

Node root		
SID	*Count*	*C. Prob.*
A	4	0.5
B	3	0.375
C	1	0.125

Node A		
SID	*Count*	*C. Prob.*
B	2	0.66
C	1	0.33

Node B		
SID	*Count*	*C. Prob.*
A	1	1

Node AB		
SID	*Count*	*C. Prob.*
A	1	1

T_2:

Node root		
SID	*Count*	*C. Prob.*
A	2	0.25
C	3	0.375
D	3	0.375

Node A		
SID	*Count*	*C. Prob.*
C	1	1

Node C		
SID	*Count*	*C. Prob.*
D	2	1

Node D		
SID	*Count*	*C. Prob.*
C	1	1

Node CD		
SID	*Count*	*C. Prob.*
-	-	-

T_3:

Node root		
SID	*Count*	*C. Prob.*
A	5	0.55
B	1	0.11
C	3	0.33

Node A		
SID	*Count*	*C. Prob.*
B	1	0.33
C	2	0.66

Node AC		
SID	*Count*	*C. Prob.*
A	1	1

T_4:

SID	*Count*	*C. Prob.*
A	4	0.363
B	3	0.272
C	1	0.091
D	3	0.272

Node A		
SID	*Count*	*C. Prob.*
B	2	0.66
C	1	0.33

Node B		
SID	*Count*	*C. Prob.*
A	1	1

Node D		
SID	*Count*	*C. Prob.*
-	-	-

Node AB		
SID	*Count*	*C. Prob.*
A	1	1

Figure 3.2: An example profile of emission trees.

ing paths are referred to as frequent sequential moving paths of objects. Furthermore, the order of frequent sequential moving paths is distinguishable due to the construction processing of emission trees. Specifically, the frequent sequential moving paths in the left side appear more previously than those frequent sequential moving paths in the right side. In addition, since emission trees have statistical information stored in conditional tables of emission trees, each node in frequent moving paths extracted from emission trees is able to further determine their corresponding weights in terms of probabilities that indicate how frequent objects appear at the corresponding label. Consider emission tree T_1 in Figure 3.2 as an example. From the conditional table of emission tree T_1, we could obtain that the frequency that the object stays in area A is 0.5 and with probability 0.33 (i.e., $P(AB) = P(A) \times P(B|A)$), this object will frequently move along AB. Clearly, when formulating dissimilarity among emission trees, how frequent objects stay along which moving path provides more detailed moving behaviors of objects. Furthermore, tree nodes with more larger probabilities should be more important in dissimilarity measurements. In the above case, node A is more important than node AB since the object is more likely to appear A than AB. Consequently, we design a dissimilarity measurement in which the above features of emission trees are considered.

In order to capture structure similarity of emission trees, we transform an emission tree into a moving sequence list, where each element in a moving sequence list represents a moving path from the root node to a leaf node and elements are ordered from the left to the right of emission trees. To facilitate the presentation of this paper, a moving sequence list of emission tree T_i is denoted as MSL_i, and MSL_i={MS_i^1 , MS_i^2,..., MS_i^n}, where MS_i^j is the jth element in MSL_i, and the number of moving paths is n. Specifically, MS_i^j is a moving sequence of location pairs, denoted as $(TL_{i,j}^k, p(TL_{i,j}^k))$, where $TL_{i,j}^k$ is the kth tree node label from the root node to the leaf node of the jth moving path and $p(TL_{i,j}^k)$ is the probability that this object is likely to stay at area $TL_{i,j}^k$. Consequently, the weight of moving path MS_i^j is formulated as $w(MS_i^j) = \sum_{i=1}^{n} p(TL_{i,j}^k)$, where n is the number of elements in MS_i^j. For example, the moving sequence list in emission tree T_1 is $\langle[(A, 0.5)], [(B, 0.375)(AB, 0.33)]\rangle$ that includes two moving paths. The weight of moving path $[(A, 0.5)]$ is 0.5 and the weight of moving path $[(B, 0.375)(AB, 0.33)]$ is 0.705.

In light of some terms defined above, we now design a dissimilarity measurement that takes spatial and temporal information into account. Explicitly, given two emission trees T_i and T_j, we first transform these emission trees as two moving sequence lists as MSL_i and MSL_j. Similar to the editing problem in [9, 10], the dissimilarity between two emission trees (e.g., $\delta_{MSL}(T_i, T_j)$) is determined by the editing distance between two moving sequence lists, denoted by $dist(MSL_i, MSL_j)$. The distance between two moving sequence lists, MSL_i and MSL_j, is calculated as the minimal cost of transforming MSL_i into MSL_j through three operations: insertion, deletion and match operations. Three operations and the corresponding costs are described as follows:

- Insertion: Insert one moving sequence into MSL_i so as to transform MSL_i into MSL_j. Assume that one moving sequence MS_j^k is inserted into MSL_i. Thus, the cost is the weight of MS_j^k (i.e.,$w(MS_j^k)$).

- Deletion: Suppose that one moving sequence (i.e., MS_i^k) of MSL_i is deleted and and the corresponding cost is defined as $w(MS_i^k)$

- Match: Match two moving sequences from the first location pairs to the last location pairs. We begin to compare the location pairs of two moving sequences in order. If the tree labels of two location pairs are equivalent, we eliminate the first location pairs of two moving sequences and add the difference of probabilities of two location pairs as the corresponding cost. Then, continue to compare two moving sequences with the remaining location pairs until the moving sequences are matched completely. Otherwise, the match is over. By the suffix property of emission trees, we know the remaining tree labels of location pairs will be all different.

The probabilities of remaining location pairs are summed up as the corresponding cost.

According to the above three operations, we can derive recurrence relation for $dist(MSL_i, MSL_j)$. For the presentation of the recurrence relation, MSL_i (respectively, MSL_j) is represented as $MSL_i[1..m]$ (respectively, $MSL_j[1..n]$, where the numbers of elements in MSL_i and MSL_j are m and n, respectively. Thus, we could have

$$dist(MSL_i[1..m], MSL_j[1..n]) =$$
$$Min \begin{cases} dist(MSL_i[1..m], MSL_j[1..n-1]) + w(MS_j^n) \quad (insertion) \\ dist(MSL_i[1..m-1], MSL_j[1..n]) + w(MS_i^m) \quad (deletion) \\ dist(MSL_i[1..m-1], MSL_j[1..n-1]) + match(MS_i^m, MS_j^n) \quad (match) \end{cases}$$

, where $match(MS_i^m, MS_j^n) =$
$$\begin{cases} match(\{TL_{i,m}^2, ..., TL_{i,m}^k\}, \{TL_{i,m}^2, ..., TL_{i,m}^l\}) + |p(TL_{i,m}^1) - p(TL_{j,n}^1)|, \ if \ TL_{i,m}^1 = TL_{j,n}^1 \\ \sum_{c=1}^{k} p(TL_{i,m}^c) + \sum_{c=1}^{l} p(TL_{j,n}^c), \ otherwise \end{cases}$$

The boundary conditions are given as follows:
$dist(MSL_i[0], MSL_j[1..n]) = \sum_{c=1}^{n} w(MS_j^c)$,
$dist(MSL_i[1..m], MSL_j[0]) = \sum_{c=1}^{m} w(MS_i^c)$
for $MS_i^m = \{TL_{i,m}^1, TL_{i,m}^2, ..., TL_{i,m}^k\}$ $MS_j^n = \{TL_{j,n}^1, TL_{j,n}^2, ..., TL_{j,n}^l\}$

Generally speaking, by exploiting dynamic programming, the dissimilarity between two moving sequence lists is derived from the above recurrence relation. For example, the dissimilarity between emission tree T_1 and emission tree T_3 in Figure 3.2 is determined by calculating $dist(MSL_1[1..2], MSL_3[1..2])$. Table 3.1 shows the execution scenario of determining the dissimilarity of T_1 and T_3. Table entry, denoted as t[i,j], represents the costs of $dist(MSL_1[1..i], MSL_3[1..j])$. Since there are three operations performed, each table entry therefore have three costs (i.e., costs for insertion, deletion and match). For example, $dist(MSL_1[1..1], MSL_3[1..1])$ is calculated by insertion, deletion and match. The costs of these three operations are shown in t[1,1].

Among three costs in t[1, 1], only the minimal cost is selected to represent the value of $dist(MSL_1[1..1], MSL_3[1..1])$. Then, according to t[1,1], we could further determine $dist(MSL_1[1..1], MSL_3[1..2])$ (i.e., t[1, 2]). Specifically, the insertion cost is obtained by summing up the insertion of moving path ([C, 0.33), (AC, 0.36)] to the cost of t[1, 1]. Clearly, the insertion cost at t[1,2] is 0.74 (i.e., t[1, 1]+$w(MS_3^2) = 0.05 + 0.33 + 0.36 = 0.74$). Similar to the derivation of insertion, we could derive deletion and match costs at t[1,2]. Also, only the minimal cost is selected at t[1,2], showing the cost of transforming $MSL_1[1..1]$ to $MSL_3[1..2]$. Following the above procedure, the value of $dist(MSL_1[1..2], MSL_3[1..2])$ is easily calculated at t[2,2], thereby deriving dissimilarity of T_1 and T_3 in terms of the dissimilarity measurement of moving sequence lists.

		0	1	2
		MSL$_3$	[(A,0.55)]	[(C,0.33)(AC,0.36)]
0	MSL$_1$	0	0.55	1.24
1	[(A,0.5)]	0.5	$1.05(i)$ $1.05(d)$ $\underline{0.05(m)}$	$\underline{0.74(i)}$ $1.74(d)$ $1.74(m)$
2	[(B,0.375)(AB,0.33)]	1.205	$1.755(i)$ $\underline{0.755(d)}$ $1.755(m)$	$\underline{1.445(i)}$ $\underline{1.445(d)}$ $\underline{1.445(m)}$

Table 3.1: An execution scenario for the dissimilarity of T_1 and T_3.

By exploring dynamic programming, dissimilarity among emission trees are efficiently determined. To verify the correctness of our design of δ_{MSL}, we derive $\delta_{MSL}(T_1, T_4)$ is 0.602. Given three emission trees T_1, T_3 and T_4, by comparing tree structures of these three emission trees, it can be verified that emission tree T_1 is more similar to emission tree T_4 than emission tree T_3. We could justify the dissimilarity by assuming that the emission trees T_1, T_3 and T_4 are constructed for objects O_1, O_3 and O_4, respectively. By glancing through three emission trees, object O_1 and object O_4 move along the paths A and $A \rightarrow B$ frequently. However, object O_1 and object O_3 only move along the path A frequently. Thus, moving behavior of O_1 is more similar to O_3. This agrees to our dissimilarity measurement derived above (i.e, $\delta_{MSL}(T_1, T_4)$ is smaller than $\delta_{MSL}(T_1, T_3)$).

3.2 Clustering Schemes in GBOT

According to the above two dissimilarity measurements, dissimilarities among emission trees are derived. Based on dissimilarities derived, in this section, we develop two clustering schemes to cluster objects with similar moving behaviors.

3.2.1 Reactive Grouping Algorithm

As mentioned in Section 2.1, cluster heads at the lowest level are responsible for tracking objects in their monitored regions. Consequently, each cluster head maintains an emission tree for each object that ever stays in the corresponding monitored region. However, we could further classify objects into two kinds of objects. For each cluster head, those objects that are currently within the corresponding monitored regions are referred to as *active objects*. On the other hands, those objects that ever appeared in the monitored regions are referred to as *inactive objects*. For example, in Figure 3.3, there are five objects in the monitored regions of four cluster heads. Suppose that all five objects move around the entire monitored regions and thus each cluster head maintenances five emission trees for each object. In cluster head CH_3, there are five emission trees since all objects ever appear in the monitored region of CH_3. Among these five objects, it can be verified that objects O_3 and O_4 are active objects, whereas objects O_1, O_2 and O_5 are inactive. Though each cluster head has emission trees for objects appearing in the corresponding monitored region, not all objects should be clustered since each cluster head has two kinds of objects (i.e., active and inactive objects). Thus, two clustering schemes are proposed. Explicitly, one is to only cluster active objects (referred to as reactive grouping) and the other is cluster both active and inactive objects (referred to as

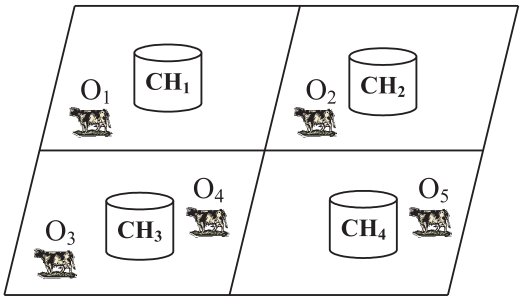


Figure 3.3: An example of active and inactive objects in HTM.

proactive grouping).

Reactive grouping scheme only clusters active objects. In light of dissimilarity measurements proposed, dissimilarity measurements of active objects are first determined. In addition to dissimilarity measurements, current locations of objects should be considered since objects are not always spatially close even if these objects have similar emission trees. To formulate the spatial proximity among objects, we define $\tau - close$ as follows:

Definition 1: ($\tau - close$)

Given two objects O_i and O_j, object O_i is $\tau - close$ to object O_j if the maximal distance of object O_i and object O_j is τ grid size.

For an object, its $\tau - close$ neighbors are those objects whose distance are within $\tau - close$ grid size.

Based on dissimilarity measurements, the definition of the spatial proximity, the general problem of clustering objects with similar moving behaviors is defined as follows:

Definition 2: (the grouping problem)

Given a set of objects S, the dissimilarity threshold δ_t and the spatial proximity threshold τ_t, objects are divided into groups such that the number of group is minimal and in each group, each pair of objects (e.g., O_i,O_j) should satisfy (i) $\delta(T_{O_i}, T_{O_j}) \leq \delta_t$ and (ii) $O_i\ is\ \tau_t - close\ to\ O_j$

From the definition of the group problem, the number of groups is expected to be minimal. Since each group will select one representative emission tree, the minimal number of groups means that only the minimal number of emission trees are preserved, thereby reducing storage cost at cluster heads. Furthermore, since the dissimilarity of each pair objects is within dissimilarity threshold and the spatial proximity is also required. Thus, objects within a group have similar moving behaviors. By judiciously selecting one emission tree for each group, movements of objects in a group are accurately predicted. Note that the grouping problem is able to reduced from the clique cover problem. The clique cover problem is that given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, the decision problem is that ”Are there k cliques of G covering all the vertices of G ?” We prove that the grouping problem is reduced from the clique cover problem. Moreover, since the clique cover problem is a NP-complete, the group problem is thus a NP-complete as well.

Theorem 1: The grouping problem is NP-complete.

Proof: Given an instance $I(G = (V, E), k)$ of the clique cover problem, we transform this instance into an instance $I'(S, \delta_t, \tau_t)$ of the grouping problem. Explicitly, the vertices of graph G is viewed as objects in S. If there exists an edge (i, j) between vertex i and vertex j, the corresponding objects (O_i, O_j) are $\tau_t - close$ and $\delta(T_{O_i}, T_{O_j}) \leq \delta_t$. Therefore, the instance $I(G = (V, E), k)$ can be transformed to the instance $I'(S, \delta_t, \tau_t)$. Notice that I' has a solution if I has a solution. Furthermore, if the solution for the grouping Problem exists, this can be verified in polynomial time.$\square$

According to the above proof, the grouping problem is a NP-Complete problem. Thus, we propose a heuristic algorithm for the grouping problem. Basically, the idea is to model the grouping problem as a clique cover problem. The reactive grouping algorithm (abbreviated as RG) is shown in Algorithm 1. Specifically, similar to the proof of Theorem 1, a graph is constructed, where each vertex represents an object and an edge between two objects is created if they are $\tau_t - close$ and the dissimilarity measurements of these two emission trees are smaller than the threshold δ_t. Once the graph G is constructed, algorithm RG is greedy in nature and select the largest cliques each run so as to minimize the number of cliques. Explicitly, selecting vertices with lager degrees implies larger cliques. Thus, from line 2 to line 3, we start to select vertex O_i with the highest degree in graph G. In line 4, those vertices adjacent to vertex O_i are put into list L. Then, the objects in list L form a graph together and recompute their node degrees (line 5). This step only to calculate the degrees of vertices in list L. We construct a group R_i that contains object O_i and repeatedly select object O_j with the highest node degree in list L and put object O_j into group R_i if object O_j has edges with all objects in R_i (from line 7 to line 12). Objects in group R_i are removed from graph G (line 14). Following the above operations, we are able to discover all cliques until all vertices are visited (i.e., G is empty).

Algorithm 1 : Reactive Grouping Algorithm

Input: S_{active}, set of active objects; δ_t, dissimilarity threshold; τ_t, spatial proximity threshold

Output: R_{active}, set of groups including active objects

```
1:  Transform the relation between the objects in $S_{active}$ into a graph $G$;
2:  while $G$ is not empty do
3:      select the object $O_i$ with the highest node degree in graph $G$;
4:      put those objects adjacent to object $O_i$ into list $L$;
5:      recompute the node degree of the objects in list $L$;
6:      construct a group $R_i$ which contains object $O_i$;
7:      while $L$ is not empty do
8:          select the object $O_j$ with the highest node degree in $L$ and remove $O_j$ from $L$;
9:          if $O_j$ is adjacent to all the objects in $R_i$ then
10:             put object $O_j$ into group $R_i$;
11:         end if
12:     end while
13:     insert group $R_i$ into set $R$;
14:     remove the objects in group $R_i$ from graph $G$;
15: end while
```

Note that in algorithm RG, we only cluster those active objects. In order to further reduce the storage cost, one could cluster both active and inactive objects. This is the call for the design of algorithm proactive grouping.

3.2.2 Proactive Grouping Algorithm

In algorithm PG, cluster heads group not only active objects but inactive objects. As a result, more storage cost is reduced. For active objects, algorithm PG performs the same clustering operations in algorithm RG. On the other hands, for inactive objects, cluster heads have no up-to-date locations of these objects. To maintenance recent up-to-date locations of these objects, in algorithm PG, cluster heads have $\tau_t - close$ neighbor lists for each object. Those objects $\tau_t - close$ to object O_i are kept in object O_i's $\tau_t - close$ neighbor list. Note that when object O_i moves into a monitored region of other cluster heads, $\tau_t - close$ neighbor list of O_i is updated. According to $\tau_t - close$ neighbor lists for inactive objects, algorithm PG is able to cluster objects with similar moving behaviors and objects are in $\tau_t - close$ neighbor lists. Clearly, by exploring $\tau_t - close$ neighbor lists, we approximately estimate the moving behaviors of objects. Hence, algorithm PG first clusters active objects and then further clusters inactive objects. For active objects, the clustering operations are the same as algorithm RG. For inactive objects, we also transform inactive objects into a graph, where each vertex represents an inactive object and an edge between objects are generated if inactive objects satisfy the dissimilarity threshold and both objects are in $\tau_t - close$ neighbor lists.

Algorithm 2 : Proactive Grouping Algorithm

```
Input: $S_{active}$, set of active objects; $S_{inactive}$, set of inactive objects; $\delta_t$, dissimilarity
    threshold; $\tau_t$, spatial proximity threshold
Output: $R_{active}$, set of groups for active objects; $R_{inactive}$, set of groups for inactive
    objects
    /* group active objects according to emission tree and up-to-date location */
 1: Perform algorithm PG for clustering active objects;
    /* group inactive objects according to emission tree and $\tau_t - close$ neighbor list */
 2: Transform the relation between the objects in $S_{inactive}$ into a graph $G$;
    /* line 3 to line 16 are the same as the line 2 to line 15 of reactive grouping algorithm
    */
```

For example in Figure 3.3, cluster head CH_3 can group inactive objects (O_1,O_2,O_5) given their emission trees and $\tau_t - close$ neighbor lists. Clearly, by grouping inactive objects, the number of emission trees is reduced. However, the grouping result of inactive objects may not fit in with real circumstances when the inactive objects become active. Assume that cluster head CH_3 has a grouping result as $\{(O_1O_2)(O_3O_4)(O_5)\}$. It can be seen that inactive objects O_1 and O_2 form a group together. Suppose that object O_1 moves from the monitored region of CH_1 to that of CH_3 and object O_2 stays in the monitored region of CH_2. Clearly, from the perspective of CH_3, object O_1 becomes active, whereas object O_2 remains inactive. Consequently, we should further perform maintenance operations, such as intra-group split or inter-group merge in Section 6, to adoptively adjust the grouping result. Hence, in algorithm PG, some maintenance overheads are required.

By grouping both active and inactive objects, algorithm PG can significantly reduce the number of emission trees compared to algorithm RG. However, some overheads are needed in algorithm PG due to various moving behaviors of inactive objects. The experimental results will show the properties of algorithms PG and RG later.

Chapter 4

Selecting One Representative Tree for Each Group

The objective of selecting one representative tree for each group is to decrease the total number of emission trees so that the original emission trees for objects can be deleted from cluster heads. Here, we select a representative emission tree from those emission trees whose corresponding objects are in a group. Considering a group R with k objects ($O_1..O_k$) and corresponding emission trees ($T_{O_1}..T_{O_k}$), instead of keeping one emission tree for each object, we select a representative emission tree among k emission trees. That is, k objects discard their original emission trees but share the representative one.

To select a representative emission tree, there are two factors to be considered: one is the storage cost of a representative emission tree and the other is the dissimilarity between emission trees. First, the smaller size of a representative emission tree is, the more storage cost reduction we can achieve. We can further reduce the storage cost, even though the total number of representative emission trees have been fixed after clustering objects. Second, we hope that the dissimilarity between representative emission tree and other emission trees in a group can be small as possible. So, the representative emission tree will be used to predict the movements of a group well. For simplicity, the size of emission tree (T_{O_i}) is represented as the number of tree nodes ($N(T_{O_i})$) and the error sum (ES) is used to quantify the dissimilarity between representative emission tree and other emission trees in a group. The error sum of emission tree T_{O_i} is defined as the sum of the dissimilarity between T_{O_i} and other $k-1$ emission trees, denoted by $ES(T_{O_i}) = \sum_{j=1}^{k} \delta(T_{O_i}, T_{O_j})$. Note that $\delta(T_{O_i}, T_{O_i}) = 0$. To select a representative emission tree from k emission trees, we define two metrics as follows:

- **Metric 1:** $Min\{N(T_{O_i}) \times ES(T_{O_i})\}$

  The emission tree with the minimum value of $N(T_{O_i}) \times ES(T_{O_i})$ is selected as the representative emission tree of group R. On the one hand, we can reduce more storage cost if the size of emission tree T_{O_i} is smaller. On the other hand, emission tree T_{O_i} is more similar to other emission trees in group R if the error sum of emission tree T_{O_i} is smaller. That means HTM can use emission tree T_{O_i} to predict the movements of group R more precisely than other emission trees. In order to take a balance between tree size and error sum, we select the emission tree with the minimal product of tree size and error sum among k emission trees.

- **Metric 2:** $Min\{\rho \times \frac{N(T_{O_i})}{\sum_{j=1}^{k} N(T_{O_j})} + (1-\rho) \times \frac{ES(T_{O_i})}{\sum_{j=1}^{k} ES(T_{O_j})}\}, 0 \le \rho \le 1$

  Metric 1 is not flexible enough for applications although it can easily take a balance between tree size and error sum. Suppose a user thinks that storage cost is more important than error sum and he desires to consider both storage cost and error sum. In such case, metric 1 fails to content the user. To improve this, we define the metric 2 which is more flexible than metric 1. We normalize tree size and error sum in Metric 2. Furthermore, a variable ρ is added for users to give different weights to tree size and error sum. Experimental results show that Metric 2 can obtain almost the same results as Metric 1 when $\rho = 0.5$, and it thus follows the flexibility of Metric 2.

In the following, we consider a group with three objects (O_1, O_2, O_3) and emission trees ($T_{O_1}, T_{O_2}, T_{O_3}$). The tree size and error sum of each emission tree is listed in

Table 4.1. To select a representative emission tree among $(T_{O_1}, T_{O_2}, T_{O_3})$, we adopt Metric 1 and Metric 2 with different values of the variable ρ. For emission tree T_{O_1}, the measurement of Metric 1 is 710 (100×7.1) and the measurement of Metric 2 with $\rho = 0.9$ is 0.350 $(0.9 \times \frac{100}{285} + (1-0.9) \times \frac{7.1}{20.4})$. Table 4.2 shows the results of representative tree selection. Using Metric 1, the emission tree T_{O_2} with the minimal product of tree size and error sum is selected as the representative emission tree. Similarly, the emission tree T_{O_2} is selected as the representative emission tree by using Metric 2 if we respect tree size and error sum equally and set the variable ρ to 0.5. In the case of Metric 2 with $\rho = 0.9$, the emission tree T_{O_3} is selected as the representative emission tree because its tree size is the smallest among three emission trees. Inversely, in the case of Metric 2 with $\rho = 0.1$, the emission tree T_{O_2} is selected as the representative emission tree because of the least error sum among three emission trees. Note that the emission tree T_{O_1} will never be selected due to the biggest tree size and the most error sum.

Table 4.1: Tree size and error sum of three objects

Tree	Size ($N(T_{O_i})$)	Dissimilarity ($\delta(T_{O_i}, T_{O_j})$)	Error Sum ($ES(T_{O_i})$)
T_{O_1}	100	$\delta(T_{O_1}, T_{O_2}) = 3.3$, $\delta(T_{O_1}, T_{O_3}) = 3.8$	7.1
T_{O_2}	95	$\delta(T_{O_2}, T_{O_1}) = 3.3$, $\delta(T_{O_2}, T_{O_3}) = 3.1$	6.4
T_{O_3}	90	$\delta(T_{O_3}, T_{O_1}) = 3.8$, $\delta(T_{O_3}, T_{O_2}) = 3.1$	6.9

Table 4.2: Example of selecting a representative emission tree.

Tree	Metric 1	Metric 2 ($\rho = 0.9$)	Metric 2 ($\rho = 0.5$)	Metric 2 ($\rho = 0.1$)
T_{O_1}	710	0.350	0.349	0.348
T_{O_2}	$\underline{608}$	0.331	$\underline{0.323}$	$\underline{0.315}$
T_{O_3}	621	$\underline{0.318}$	0.327	0.335

Chapter 5

Training Representative Emission Trees

In this section, we develop a group VMM model so that the representative emission tree can be trained for each group. In HTM, cluster heads originally construct an emission tree which is trained by VMM model for each object. Thus, it is necessary to modify VMM model to suit a group that is composed of multiple objects. The basic concept is to allow cluster heads to construct a VMM model with multiple buffers for a group. The buffers are used to hold the most recent moving records of objects in a group. There are two cases to illustrate the training process of a group VMM model.

- **Case 1:** group with one object

  In this case, an object forms a group individually. Cluster heads construct a VMM model and a buffer to hold the most recent moving records for the group. Then, VMM model mines the moving patterns according to the buffer and trains the representative emission tree. Regardless of predicting correctly or incorrectly, the labels which object visited are put into the buffer for moving pattern mining.

- **Case 2:** group with multiple objects

  In this case, multiple objects form a group together. Considering a group R with k

objects and a representative emission tree, cluster heads construct a VMM model and k buffers for the group with k objects. Then, VMM model will mine the moving patterns according to the k buffers and trains the representative emission tree. Different from case 1, not all labels which objects visited are put into the buffers. The rules to put labels into the buffers are:

Case 2.1: correct prediction

If cluster head predicts the movement of object correctly, we put the visited label into the buffer. The visited label would be worthy to put into the buffer for repetitive VMM model training when correct prediction happened.

Case 2.2: wrong prediction

If cluster head predicts the movement of object wrong and there exists no $0-close$ objects which are predicted correctly, we put the visited into the buffer. It means that cluster head can not correctly predict the next position of such object according to its current position by representative emission tree. Thus, the visited label should be put into the buffer in order to contribute to the moving pattern mining. Otherwise, we discard the visited label but put a character $\#$ into the buffer. The visited label would not be put into the buffer if the object is predicted incorrectly and there exists a $0-close$ object which is predicted correctly. We think that this visited label is unreliable and unworthy to put into the buffer for moving pattern mining because there exists another object which is predicted correctly according to the same current position. Instead, we put a character $\#$ into the buffer so that only the moving records behind character $\#$ can contribute to the moving pattern mining.

Figure 5.1 illustrates a example of case 2. Three objects $\{O_1, O_2, O_3\}$ with current positions $\{M, H, M\}$ form a group R together. Cluster head CH_1 predicts the positions of $\{O_1, O_2, O_3\}$, which are $\{N, I, N\}$. The actual next positions of $\{O_1, O_2, O_3\}$ are

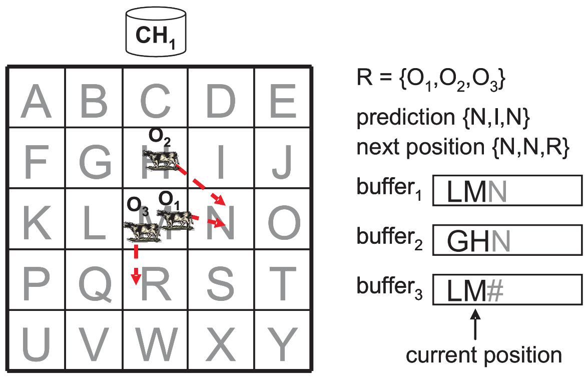


Figure 5.1: Different cases: put the visited labels into the buffers

$\{N, N, R\}$. Cluster head CH_1 constructs a VMM model with three buffers for group R. According to the principle of case 2.1, object O_1 is predicted correctly. Thus, the visited label N is put into the buffer. For object O_2, cluster head predicts its position incorrectly and there exists no $0-close$ objects which are predicted correctly. Hence, the visited label N is also put into the buffer. However, object O_3 is predicted incorrectly and there exists $0-close$ object O_1 which is predicted correctly according to the same current position M. Accordingly, we discard the label of current position R and put a character $\#$ into the buffer.

Chapter 6

Maintenance Algorithm in GBOT

In Section 6.1, we first introduce two conditions, split condition and merge condition. In Section 6.2, a maintenance algorithm is proposed to detect such two conditions and execute corresponding operations.

6.1 Split condition & Merge condition

To examine the variations in moving behavior of intra-group and inter-group, we focus on detecting the following two conditions.

Condition 1: (Split-Condition)

In this condition, for a group, there exists an object which left far away from other objects. The object which escaped from its group is called the spatial outlier. By detecting the split condition, the spatial outliers in a group are dug out. For example, Figure 6.1 illustrates the split condition. As seen, object O_3 is far away from other objects in Group1. Thus, object O_3 is a spatial outlier in Group1.

Condition 2: (Merge-Condition)

In this condition, there exists the overlap between two groups when they are spatially close. Figure 6.2 illustrates the merge condition that Group1 overlaps Group2. Object O_1 and object O_4 occupy the same grid M which is the overlap between Group1 and

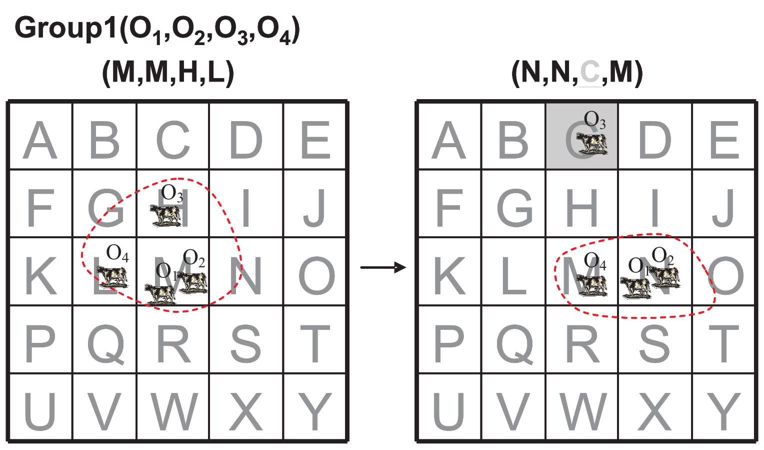


Figure 6.1: An illustrative example for Split-Condition

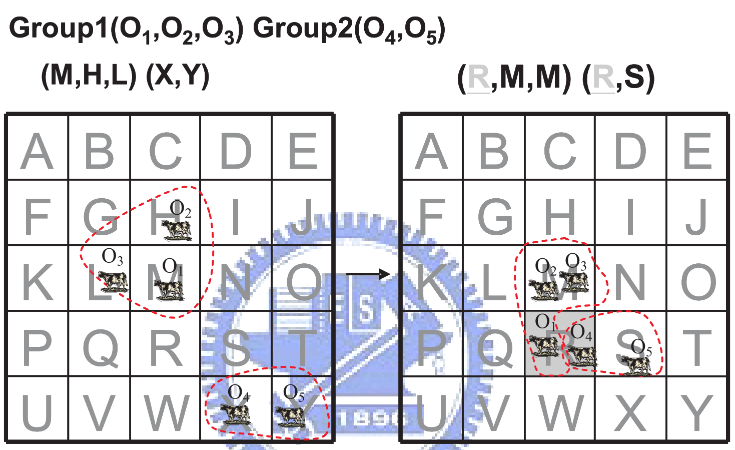

Group1(O1,O2,O3) Group2(O4,O5)
(M,H,L) (X,Y)
(R,M,M) (R,S)
O1 O2 O3 O4 O5
A B C D E F G H I J K L M N O P Q R S T U V W X Y

Figure 6.2: An illustrative example for Merge-Condition

Group2.

In the maintenance step, we execute the periodic maintenance operation to detect the split condition and merge condition. Then, the intra-group split operation and inter-group operation are executed to deal with these two conditions respectively. The three operations are described in the following:

- **Periodic Maintenance**

  A maintenance algorithm is proposed to periodically detect if there is any condition happened every T time units. If the split condition is detected, the maintenance algorithm calls the intra-group split operation. Similarly, the maintenance algorithm would call the inter-group merge operation if the merge condition is detected.

- **Intra-group Split**

  If the spatial outlier has been detected for consecutive x times, it would be split from original group to avoid wrong prediction. Then, the spatial outlier forms a group singly and obtains a copy of emission tree from original group. The parameter x is used to prevent frequent overhead for immediate split. It is possible that the object provisionally leaves the group and return to join the group soon. To avoid such event, we set up the parameter x to confirm that the object left for sure.

- **Inter-group Merge**

  When one group overlaps another group, we check whether the two groups should merge into one group or not. If the two groups have similar representative emission trees mutually and are geographically near each other. Then, two groups merge into one group for saving storage and execute the representative tree selection operation.

6.2 Maintenance Algorithm

In this section, we propose a maintenance algorithm to execute the periodic maintenance operation, intra-group split operation and inter-group merge operation. In order to detect the split condition and merge condition, each cluster head executes the periodic maintenance operation from line 1 to line 15 of Algorithm 3. For each group, we sort all objects by their number of $\tau_t - close$ neighbors and construct a empty list L (from line 2 to line 4). The object with more $\tau_t - close$ neighbors have higher priority to be selected. We start to select the object from high to low number of $\tau_t - close$ neighbors. For each object O_i, if it is $\tau_t - close$ to all objects in list L, we put the object O_i into list L (from line 5 to line 7). Or, we mark object O_i as the spatial outlier (from line 8 to line 9). Then, we search object $O_i's$ $0 - close$ neighbors. If there exists an object $O_j \in R_j$ which is

$0 - close$ to object O_i, the merge candidate pair (R_i, R_j) is generated (from line 11 to line 13). That is, group R_i overlaps group R_j. To execute the intra-group split operation, for each spatial outlier O_{sp}, we split it from original group if it has been detected for consecutive x times (from line 17 to line 21). To execute the inter-group merge, for each merge candidate pair (R_i, R_j), we merge R_i and R_i into a group if $\delta(T_{R_i}, T_{R_j}) \leq \delta_t$ and any two objects in $R_i \cup R_j$ are $\tau_t - close$.

Algorithm 3 : Maintenance Algorithm

Input: R_S, set of groups; δ_t, dissimilarity threshold; τ_t, spatial proximity threshold
Output: R'_S, set of groups after maintenance

```
1: Periodic Maintenance
2: for each group $R_i \in R_S$ and $|R_i| \geq 2$ do
3:     sort the objects in $R_i$ by the number of $\tau_t - close$ neighbors;
4:     construct a empty list $L$;
5:     for each object $O_i \in R_i$ do
6:         if object $O_i$ is $\tau_t - close$ to all objects in $L$ then
7:             put object $O_i$ into list $L$;
8:         else
9:             mark object $O_i$ as the spatial outlier;
10:        end if
11:        if there exists an object $O_j \in R_j$ which is $0 - close$ to $O_i$ then
12:            generate merge candidate pair $(R_i, R_j)$;
13:        end if
14:    end for
15: end for
16: Intra-group Split
17: for each spatial outlier $O_{sp}$ do
18:     if spatial outlier $O_{sp}$ has been detected for consecutive $x$ times then
19:         split spatial outlier $O_{sp}$ from its group;
20:     end if
21: end for
22: Inter-group Merge
23: for each merge candidate pair $(R_i, R_j)$ do
24:     if $\delta(T_{R_i}, T_{R_j}) \leq \delta_t$ and any two objects in $R_i \cup R_j$ are $\tau_t - close$ then
25:         merge $R_i$ and $R_j$ into a group;
26:     end if
27: end for
```

For example, the input of maintenance algorithm are $R_s = \{Group1(O_1, O_2, O_3, O_4)$ $\}$, δ_t, $\tau = 1$ in Figure 6.1. The intra-group split operation is executed because object O_3

is a spatial outlier. object O_3 would be split from $Group1$ if it has been detected for consecutive x times. Suppose that object is split from $Group1$ and form a group singly. The output of maintenance algorithm is $R'_s = \{Group1(O_1, O_2, O_4), Group2(O_3)\}$. Another example in Figure 6.2, the input of maintenance algorithm are $R_s = \{Group1(O_1, O_2, O_3), Group2(O_4, O_5)\}$, δ_t, $\tau = 1$. For object O_1, there exists an object $O_4 \in Group2$ which is $0 - close$ to object O_1. Thus, the inter-group merge operation is executed because the merge candidate pair $(Group1, Group2)$ is generated. $Group1$ and $Group2$ will merge into one group if $\delta(T_{Group1}, T_{Group2}) \leq \delta_t$ and any two objects are $\tau_t - close$ in $Group1 \cup Group2$. Suppose that $Group1$ and $Group2$ merge into one group. The output of maintenance algorithm is $R'_s = \{Group1(O_1, O_2, O_3, O_4, O_5)\}$.

Chapter 7

Performance Evaluation

In this section, experiments are conducted to evaluate the effectiveness and efficiency of the framework GBOT by simulation.

7.1 Simulation Model

To simulate GBOT, we adopt the heterogeneous tracking model and design a group mobility model based on city mobility model [8]. In our heterogeneous tracking model, there are three levels hierarchy and 10*10 low-end sensors in each level-0 cluster. Totally, there are one level-2 cluster head, 4 level-1 cluster heads, 16 level-0 cluster heads and 1600 low-end sensors. To simulate the objects' movement with gregarious property, we design a group mobility model based on city mobility model. In our group mobility model, a logical group pilot is first generated for a group and the objects in a group follow their group pilot according to two parameters: variation period and variation radius. For every variation period, an objet will choose a random direction to move the distance of variation radius away from the group pilot. Given a maximum variation radius max_{VR}, the variation radius of each objects is uniformly distributed from 0 to max_{VR}. For example, an objet with variation period = 2 and variation radius = 3 is 3-close to the group pilot every 2 time units. We simulate the moving behavior of the group pilot which makes a tour between several specified locations in monitor region. The group pilot will repeatedly visit those specified locations by turns. During the tour, the group pilot will leave the subregion of one level-i cluster head to another one with probability p_i and stay with probability 1 - p_i. The probability p_i is determined by an exponential probability $e^{-C \cdot 2^{i+2}}$, where C is a positive constant. A higher value C means higher locality. Also, the leaving probability is e^{-2C} when the group pilot moves from the location of one low-end sensor to another one.

In the following experiments, we generate 30 objects and one group pilot for objects to follow. Then, the group pilot visits four different positions which are randomly selected from four level-1 subregions. The variation period is set to 3 and the maximum variation radius is set to 4. The OTSN works for 10000 time units. For HTM, in the initial 3000 time units, the moving records of objects are collected for mining object moving patterns by cluster heads. After 3000 time units, the cluster heads will turn to be in prediction phase. For GBOT, in the beginning of prediction phase, the initialization step will be executed with the dissimilarity threshold δ_t = 2.5, the spatial proximity $\tau_t = 2$, dissimilarity δ_{MSL} and metric 1 for representative tree selection. During the prediction phase, the maintenance algorithm will be periodically executed every 500 time units.

7.2 Performance of GBOT

In this section, we study compare the performance of GBOT and HTM in terms of storage cost and prediction accuracy. We implement two clustering schemes of GBOT, reactive grouping (RG) and proactive grouping (PG). Two performance metrics are used to evaluate the performance of HTM, RG and PG. The first performance metric, average number of tree nodes which is defined as $\frac{Total\ Number\ of\ Tree\ Nodes}{Number\ of\ Objects}$, is used to measure the average storage cost for each objects. Total number of tree nodes represents the total number of nodes in all emission trees. The second performance metric, called hit rate of prediction = $\frac{Total\ Number\ of\ Correct\ Prediction}{Total\ Number\ of\ Prediction}$, is used to measure the prediction accuracy for all cluster heads. In prediction phase, we record the total number of prediction and the total number of correct prediction to calculate the hit rate of prediction.

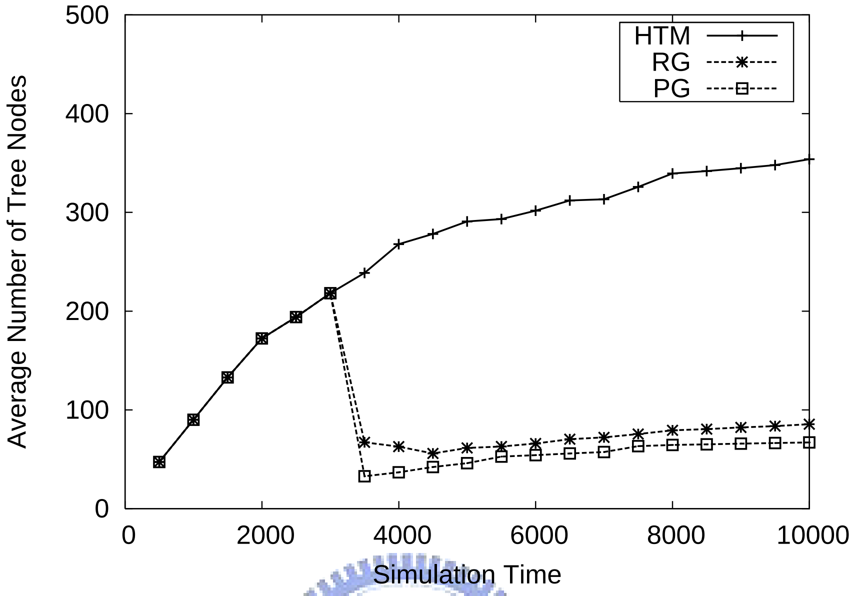
7.2.1 Comparison of HTM, RG and PG

The storage cost of HTM, RG and PG are shown in Figure 7.1(a). At the 3000th time unit, the average number of tree nodes of RG and PG largely decrease because cluster heads cluster objects and preserve only an emission tree for each group. Obviously, RG and PG have much smaller average number of tree nodes than HTM does. As mentioned before, the storage cost at cluster heads will increase gradually with time. We also observe that the average number of tree nodes of HTM increases faster than RG and PG with time. Thus, GBOT can reduce the storage cost at each cluster head effectively. In addition, PG reduces more storage cost than RG due to grouping both active and inactive objects. It can be also verified that RG reduces the storage cost by degrees from 3000 to 5000 time units.

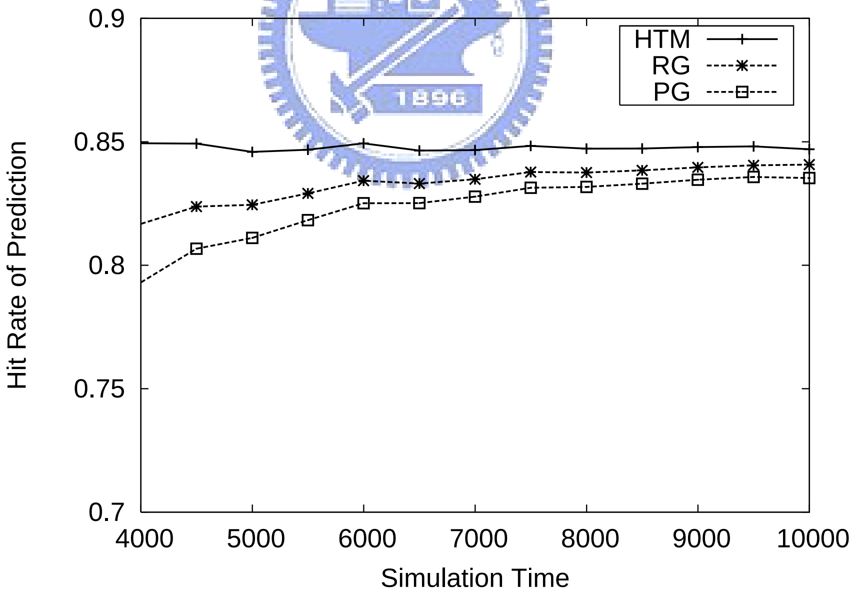
In Figure 7.1(b), we can see that RG and PG have slightly lower hit rate of prediction than HTM. As expected, we reduce the storage cost at cluster heads at the risk of lowering prediction accuracy. However, in the worst case, the prediction rate of HTM is only 5% higher than PG and RG. With time passing by, the prediction rates of RG and PG converge to 0.84 which is about 0.7% lower than HTM. In sum, a plenty of storage cost reduction outweighs the slight loss of prediction accuracy.

7.2.2 Dissimilarity δ_{MSL} & Dissimilarity δ_{LS}

In this experiment, we show the influence of dissimilarity δ_{MSL} and δ_{LS} on performance. To compare fairly dissimilarity δ_{MSL} with dissimilarity δ_{MSL}, we set the threshold δ_t for δ_{MSL} to 2.5 and the threshold δ_t for δ_{LS} to 0.3. Let dissimilarity δ_{MSL} and δ_{LS} have almost the same storage cost, we can see the impact on the hit rate of prediction in Fig-



(a) Average Number of Tree Nodes



(b) Hit Rate of Prediction

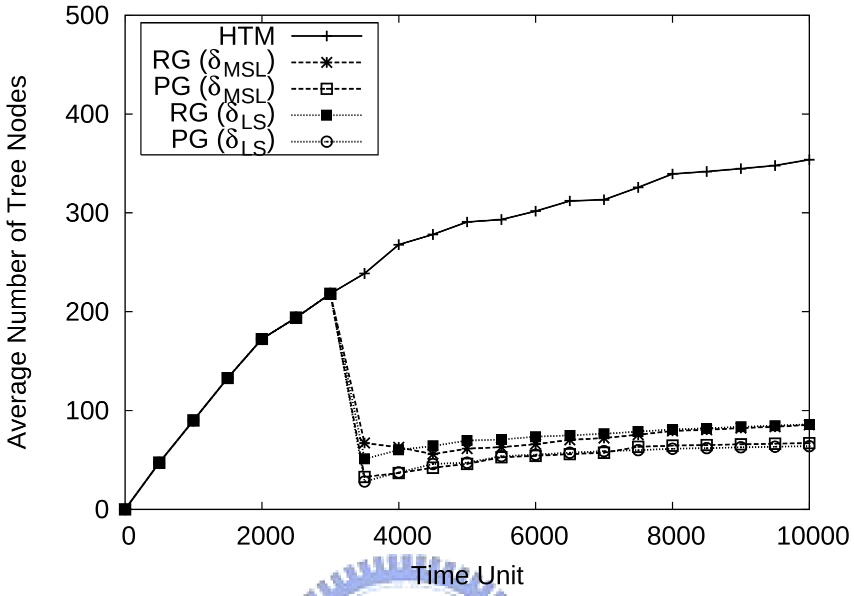
Figure 7.1: Comparison of HTM, RG and PG.

ure 7.2(b). As expected, dissimilarity δ_{MSL} outperforms dissimilarity δ_{LS} in prediction accuracy. For dissimilarity δ_{LS}, it only achieves about 76% hit rate of prediction. By thinking of the characteristics of an emission tree, dissimilarity δ_{MSL} can measure how two emission tree are dissimilar more precisely than dissimilarity δ_{LS}. Thus, dissimilarity δ_{MSL} improves the hit rate of prediction to about 83%.

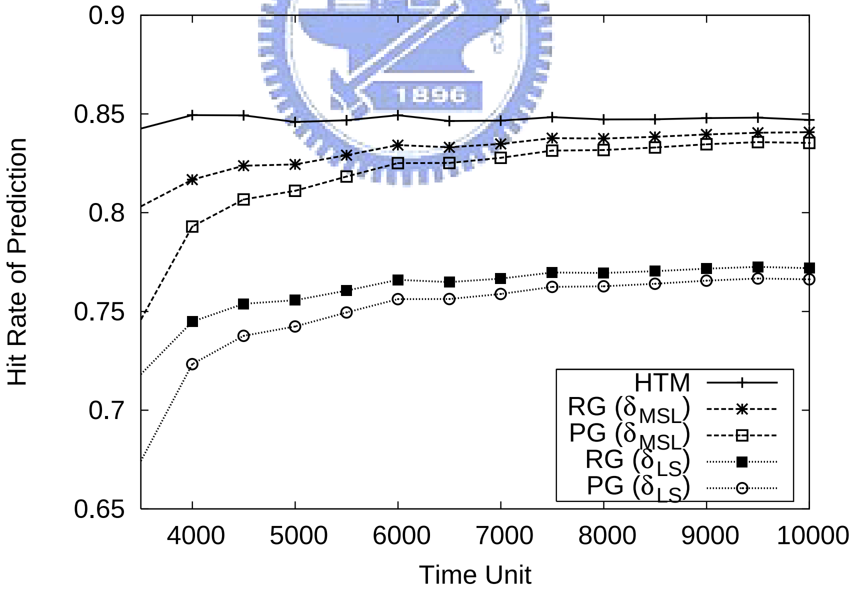
7.2.3 The Impact of Maintenance in GBOT

In this experiment, we examine the effectiveness of maintenance algorithm when the moving behaviors of intra-group and inter-group vary with time. By former experiment settings, we show the impact of maintenance algorithm in Figure 7.3. Clearly, GBOT (RG and PG) with maintenance reduces more storage cost than that without maintenance due to the inter-group merge operation. Additionally, the hit rate of prediction is almost the same whether the maintenance algorithm is executed or not. In this case, the moving behaviors of groups become more similar with time and consequently the inter-group merge operations are executed to further reduce the storage cost. Note that the intra-group split operation is scarcely executed in this case.

To show the effect of the intra-group split operation, we allow objects to alter their moving behaviors during the simulation time. Assuming that 30 objects move by following the group pilot during the simulation time [0,6000]. After 6000 time units, instead of following the group pilot, 15 objects are picked to randomly visit four specified locations. The spatial outlier would be split from its group if it has been detected for consecutive 3 times. Figure 7.4 shows the experimental results of this case. In Figure 7.4(a), GBOT with maintenance causes more storage cost after 6000 time units. This is because the intra-group split operation is executed when objects have become dissimilar in moving behaviors. We can observe that GBOT without maintenance worsens the prediction accuracy after 6000 time units. However, GBOT with maintenance can preserve the prediction accuracy because the spatial outliers are split from groups to avoid

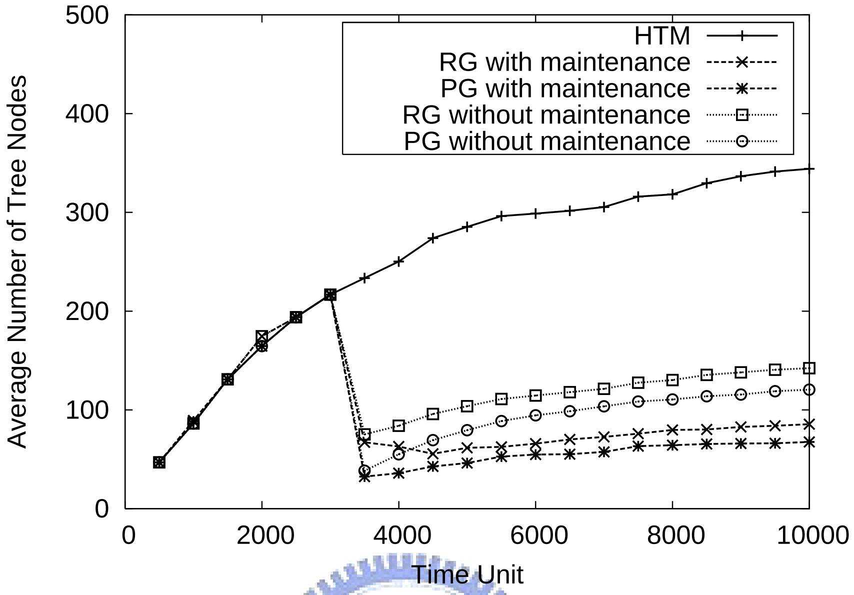


(a) Average Number of Tree Nodes

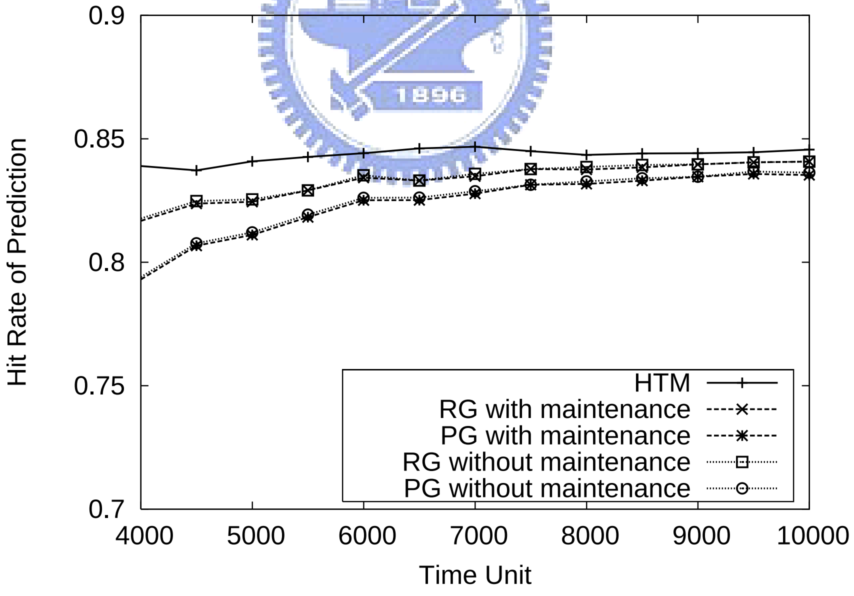


(b) Hit Rate of Prediction

Figure 7.2: The impact of dissimilarity δ_{MSL} & δ_{LS}.



(a) Average Number of Tree Nodes



(b) Hit Rate of Prediction

Figure 7.3: The impact of maintenance algorithm (merge condition).

wrong prediction. Therefore, with maintenance, GBOT can further reduce the storage cost and preserve the prediction accuracy if the moving behaviors of objects vary beyond expectation.

7.3 Sensitivity Analysis

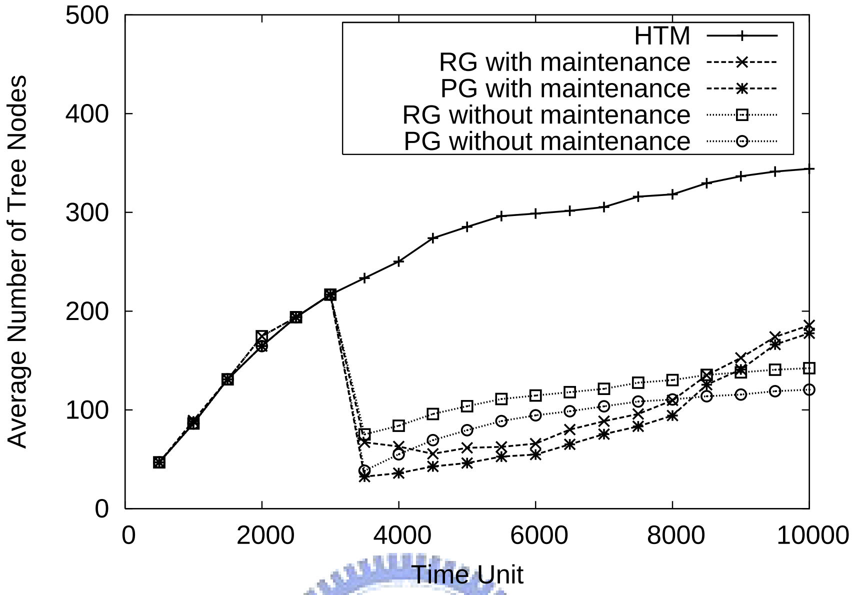
In this section, we analyze the impact of varying parameters of GBOT. In order to emphasize the storage cost reduction of GBOT, we introduce the reduction rate, which is defined as $\frac{Total\ Number\ of\ Tree\ Nodes_{HTM} - Total\ Number\ of\ Tree\ Nodes_{GBOT}}{Total\ Number\ of\ Tree\ Nodes_{HTM}}$, to make clear the storage cost reduction of different parameter settings. At the meantime, we also discuss hit rate of prediction effected by different parameter settings.

7.3.1 Number of Objects

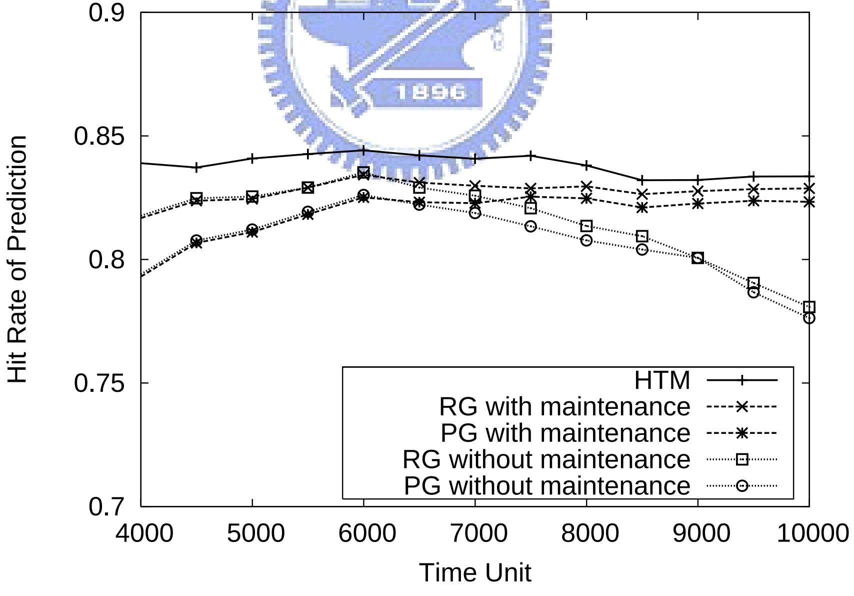
In this experiment, we show the impact of number of objects in Figure 7.5. As seen in Figure 7.5(a), the reduction rate increases while the number of objects increases. In other words, GBOT can achieve more storage cost reduction when more objects are involved in object tracking. Thus, the performance of GBOT exhibits good scalability. Note that the variations in number of objects do not influence the hit rate of prediction. The reason is that the dissimilarity threshold δ_t and the spatial proximity threshold τ_t generally dominate the prediction accuracy.

7.3.2 Dissimilarity Threshold & Spatial Proximity Threshold

In this experiment, we examine the impact of the dissimilarity threshold δ_t for dissimilarity δ_{MSL} and the spatial proximity threshold τ_t. In Figure 7.6, the reduction rate increases when the dissimilarity threshold δ_t or the spatial proximity threshold τ_t is enlarged. With larger dissimilarity threshold δ_t or the spatial proximity threshold τ_t, an objet is more likely to form a group with other objects Thus, more storage cost of emis-

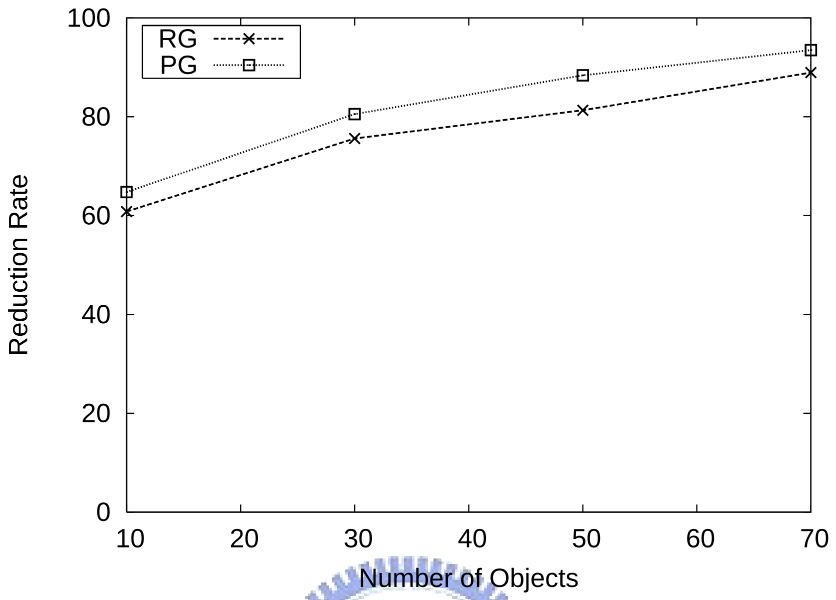


(a) Average Number of Tree Nodes

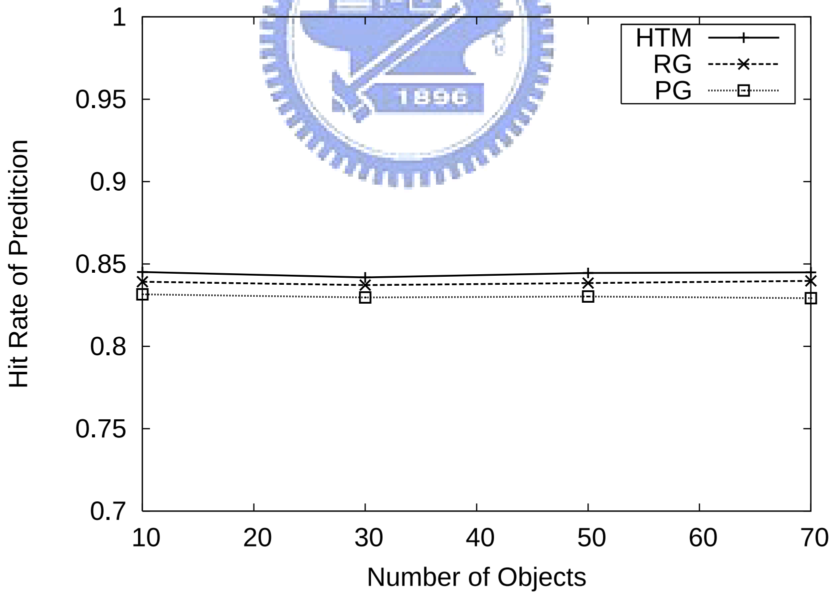


(b) Hit Rate of Prediction

Figure 7.4: The impact of maintenance algorithm (split condition).



(a) Reduction Rate



(b) Hit Rate of Prediction

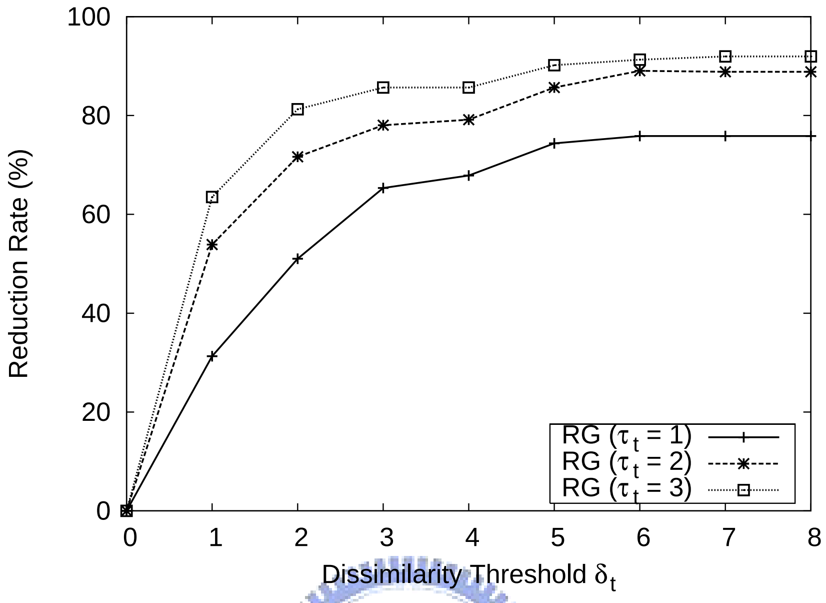
Figure 7.5: The impact of number of objects.

sion trees can be reduced. Another observation is that the reduction rate increases no more when the dissimilarity threshold δ_t is more than 6. It is intuitive that the dissimilarities between emission trees are all less than 6 in this case. Also, we can observe that the hit rate of prediction decreases when the dissimilarity threshold δ_t or the spatial proximity threshold τ_t is enlarged in Figure 7.7.

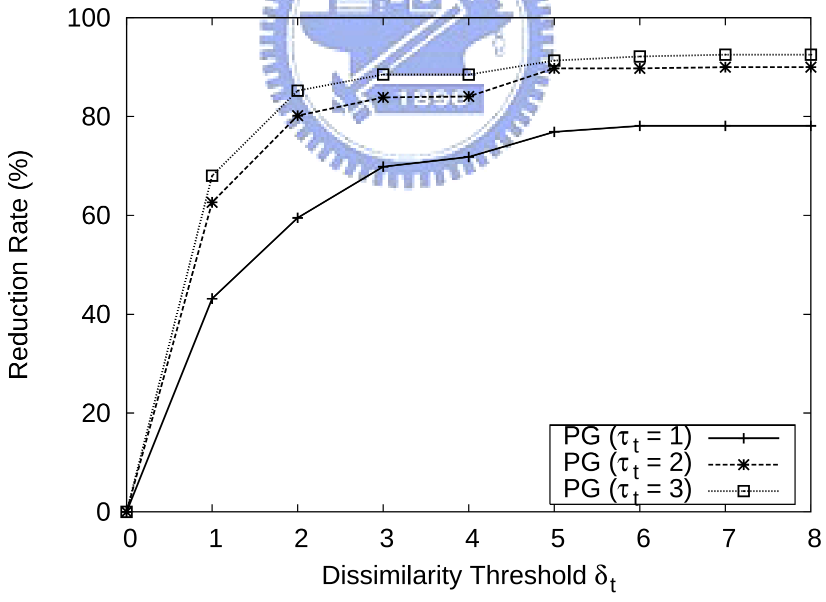
7.3.3 Variation Period & Maximum Variation Radius

In this experiment, the impact of variation period and maximum variation radius is investigated. Figure 7.8 shows the experimental results of variation period. As seen in Figure 7.8(a), the reduction rate increases when variation period is lengthened. The reason is that the gregarious property is more obvious when the objects less often move away from the group pilot. Consequently, more objects are able to form a group and more storage cost at cluster heads could be reduced due to gregarious property. In Figure 7.8(b), we can see that varying variation period does not affect the hit rate of prediction since the hit rate of prediction is primarily dominated by the dissimilarity threshold δ_t and the spatial proximity threshold τ_t.

Figure 7.9 shows the experimental results of maximum variation radius. Figure 7.9(a) shows that the reduction rate decreases consistently when max variation radius is enlarged. The reason is that the gregarious property become less obvious when objects move farther away from the group pilot. Consequently, fewer objects can form a group and less storage cost of emission trees can be reduced due to gregarious property. Note that the hit rate of prediction also decreases when max variation radius is enlarged. The objects are more likely to cross the boundary of the subregion because max variation radius is enlarged. Therefore, cluster heads make more wrong predictions and thus lowers the prediction accuracy.

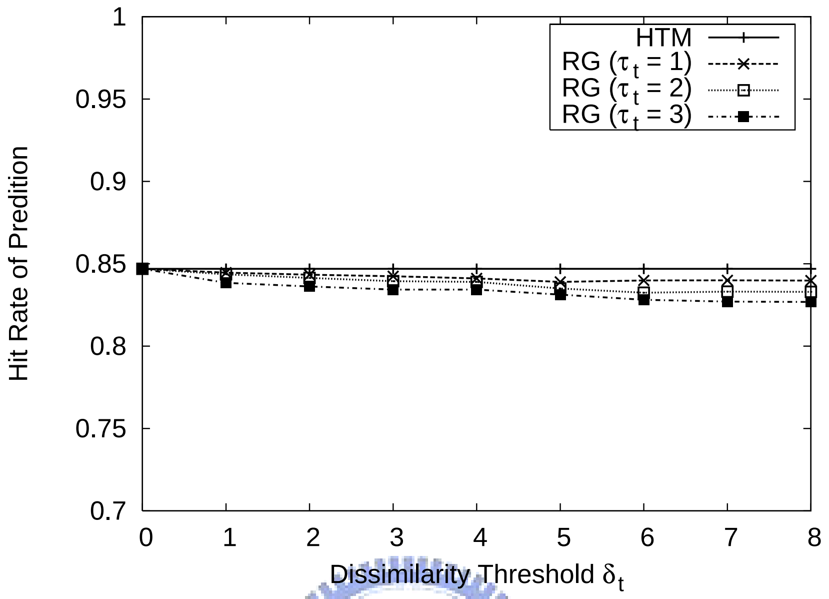


(a) Reactive Grouping

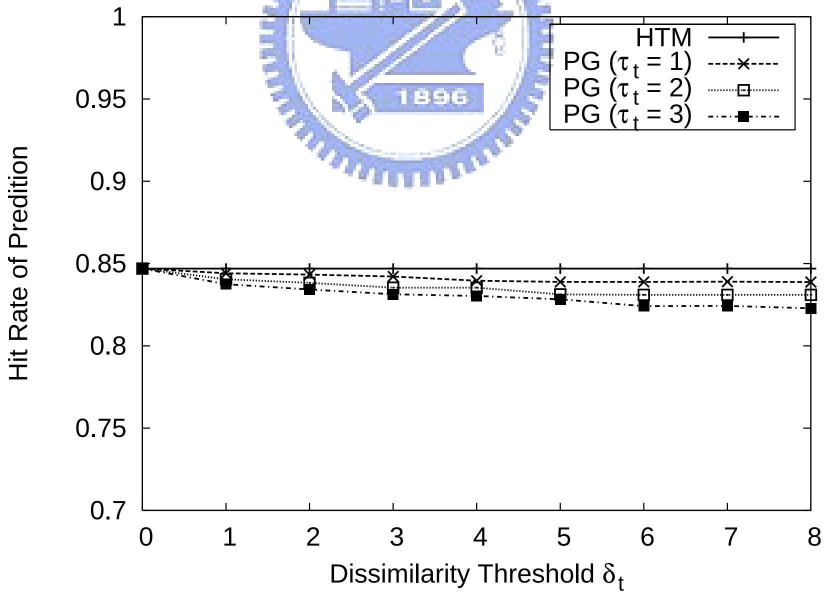


(b) Proactive Grouping

Figure 7.6: The impact of δ_t and τ_t on reduction rate.

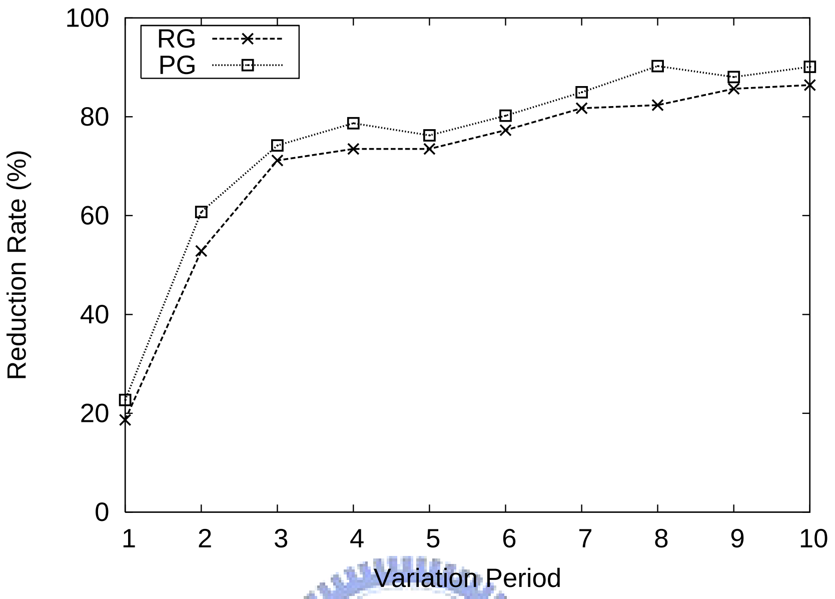


(a) Reactive Grouping

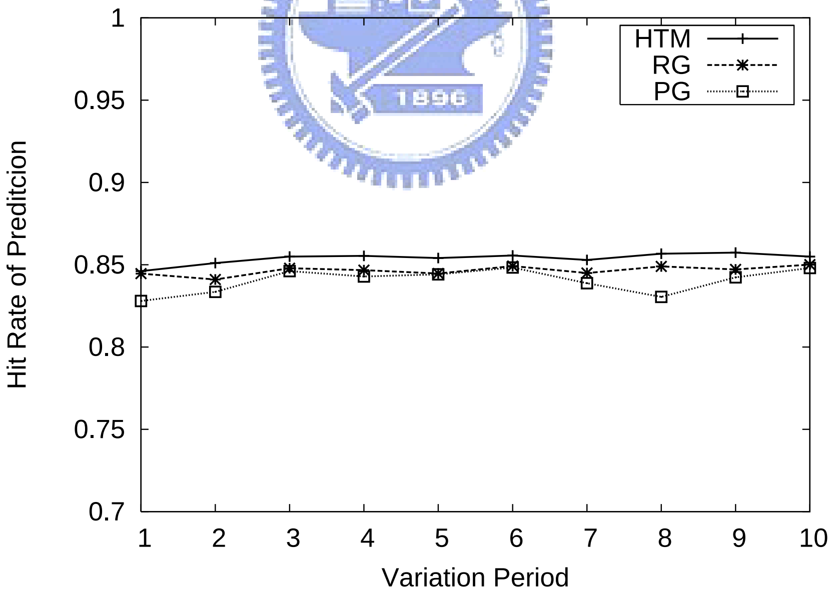


(b) Proactive Grouping

Figure 7.7: The impact of δ_t and τ_t on hit rate of prediction.

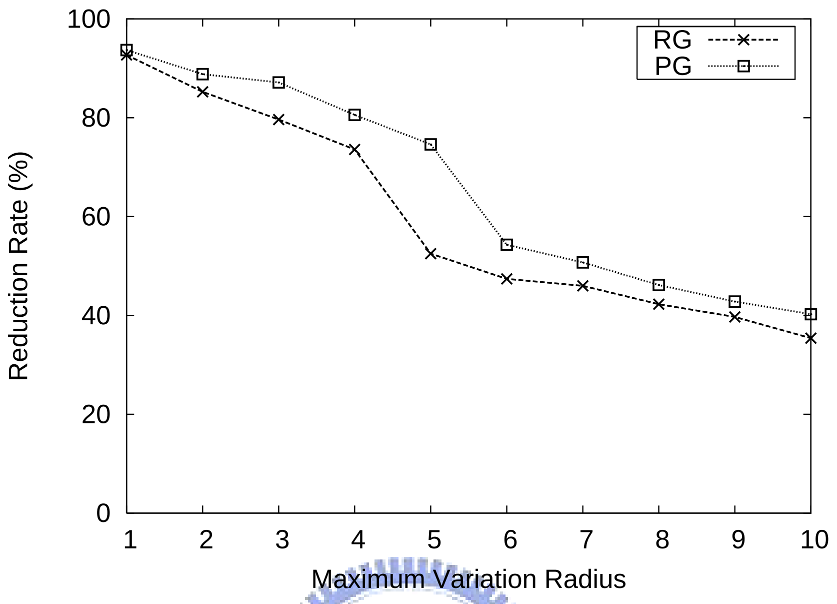


(a) Reduction Rate

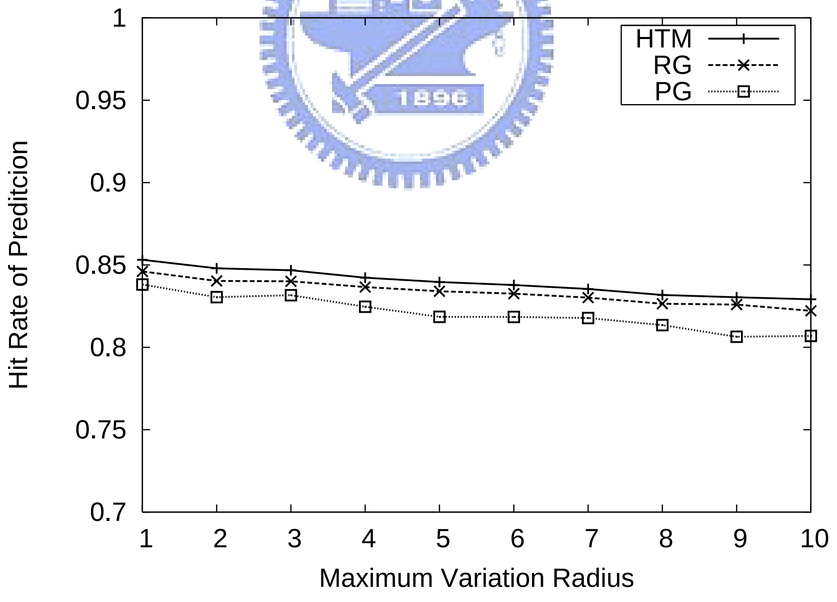


(b) Hit Rate of Prediction

Figure 7.8: The impact of variation period.



(a) Reduction Rate

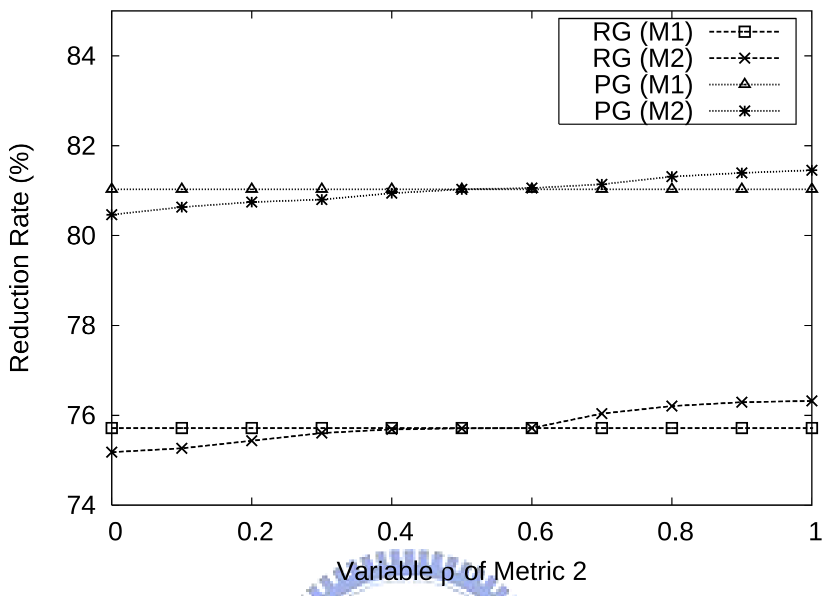


(b) Hit Rate of Prediction

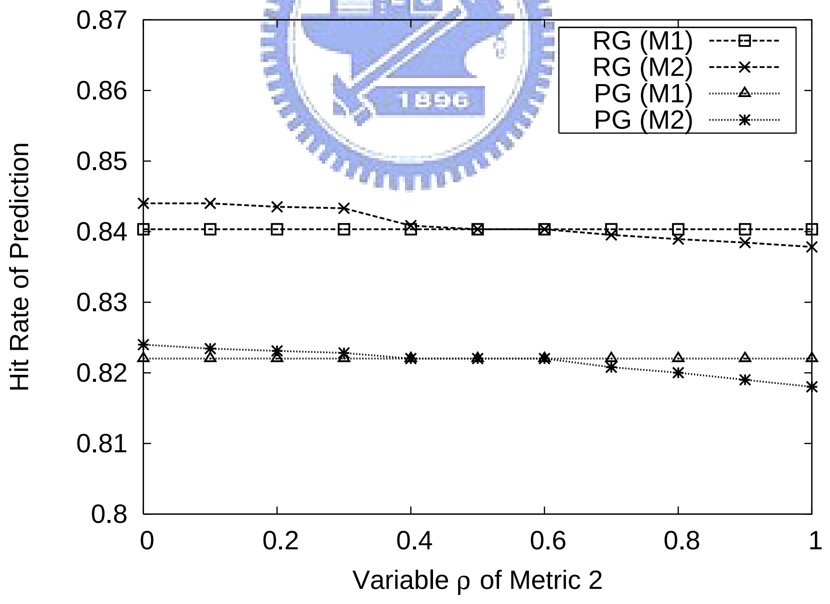
Figure 7.9: The impact of maximum variation radius.

7.3.4 Representative Tree Selection

In this experiment, we evaluate the performance of two metrics of representative tree selection and vary variable ρ between [0,1] to show the flexibility for applications. In Figure 7.10, the comparison between representative tree selection metrics and the effectiveness of variable ρ are presented. As seen, the reduction rate increases as variableρ increases. If we only favor the storage cost of emission tree and set variable ρ to be 1, we can obtain the reduction rate which is higher than others. Inversely, we can obtain the hit rate of prediction which is higher than others if the variable ρ is set to be 0. For metric 1, its performance is very close to the performance of metric 2 with $\rho = 0.5$. It supports that metric 2 really enjoy the flexibility between reduction rate and hit rate of prediction.



(a) Reduction Rate



(b) Hit Rate of Prediction

Figure 7.10: The impact of variable ρ.

Chapter 8

Conclusion

In this paper, we consider how to reduce the huge storage cost resulted from storing emission trees. For the sake of saving storage, we propose the framework GBOT to perform group-based object tracking for HTM. There are mainly three steps to be executed. To clustering objects with similar moving behaviors, we first define the dissimilarity among emission trees to distinguish the moving behaviors of objects. Based on such dissimilarity measures, we formulate two clustering schemes, reactive grouping and proactive grouping, to group objects reactively or proactively. In order to select the representative emission tree for a group, two metrics are provided to further reduce the storage cost and increase the prediction accuracy. Then, we develop a group VMM model to adequately train the representative emission trees. In addition, a maintenance algorithm is proposed to maintain the quality of groups. We also conduct several experiments to evaluate the performance of GBOT. The experimental results show GBOT not only effectively reduce the storage cost but preserve the prediction accuracy.

Bibliography

[1] J. A. Aslam, Z. J. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proceeding of ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 150–161, 2003.

[2] J.-C. Cano and P. Manzoni. Group mobility impact over tcp and cbr traffic in mobile ad hoc networks. In *Proceeding of Euromicro Workshop on Parallel, Distributed and Network-Based Processing (PDP)*, pages 382–389, 2004.

[3] W.-P. Chen, J. C. Hou, and L. Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. In *Proceeding of IEEE International Conference on Network Protocols (ICNP)*, pages 284–294, 2003.

[4] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal on High Performance Computing Applications*, 16(3), 2002.

[5] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceeding of ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 129–143, 2004.

[6] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad hoc wireless networks. In *Proceeding of ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 53–60, 1999.

[7] J.-L. Huang, M.-S. Chen, and W.-C. Peng. Exploring group mobility for replica data allocation in a mobile environment. In *Proceeding of ACM International Conference on Information and Knowledge Management (CIKM)*, pages 161–168, 2003.

[8] H. T. Kung and D. Vlah. Efficient location tracking using sensor networks. In *Proceeding of IEEE Wireless Communications and Networking Conference (WCNC)*, volume 3, pages 16–20, March 2003.

[9] K. Oflazer. Error-tolerant tree matching. In *Proceeding of International Conference on Computational Linguistics (COLING)*, pages 860–864, 1996.

[10] K. Oflazer. Error-tolerant retrieval of trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1376–1380, 1997.

[11] S. Pattem, S. Poduri, and B. Krishnamachari. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *Proceeding of IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 32–46, 2003.

[12] W.-C. Peng, Y.-Z. Ko, and W.-C. Lee. On mining moving patterns for object tracking sensor networks. In *Proceeding of International Conference on Mobile Data Management (MDM)*, page 41, 2006.

[13] H.-P. Tsai, D.-N. Yang, W.-C. Peng, and M.-S. Chen. Exploring group moving pattern for an energy-constrained object tracking sensor network. In *Proceeding of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2007.

[14] K. Wang and B. Li. Group mobility and partition prediction in wireless ad-hoc ntworks. In *Proceeding of IEEE International Conference on Communications (ICC)*, 2002.

[15] Q. Wang, W.-P. Chen, R. Zheng, K. Lee, and L. Sha. Acoustic target tracking using tiny wireless sensor devices. In *Proceeding of IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 642–657, 2003.

[16] Y. Xu, J. Winter, and W.-C. Lee. Dual prediction-based reporting for object tracking sensor networks. In *Proceeding of International Conference on Mobile and Ubiquitous Systems (MobiQuitous)*, pages 154–163, 2004.

[17] Y. Xu, J. Winter, and W.-C. Lee. Prediction-based strategies for energy saving in object tracking sensor networks. In *Proceeding of International Conference on Mobile Data Management (MDM)*, pages 346–357, 2004.

[18] H. Yang and B. Sikdar. A protocol for tracking mobile targets using sensor networks. In *Proceeding of IEEE International Workshop on Sensor Networks Protocols and Applications*, pages 71–81, 2003.

[19] J. Yang and W. Wang. Agile: A general approach to detect transitions in evovling data streams. In *Proceeding of IEEE International Conference on Data Mining (ICDM)*, pages 559–562, 2004.

[20] W. Zhang and G. Cao. Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transactions on Wireless Communications*, pages 1689–1701, 2004.

[21] W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *Proceeding of IEEE International Conference on Computer Communications (INFOCOM)*, pages 2434–2445, 2004.

[22] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 2002.