國立交通大學

資訊工程學系

碩 士 論 文

利用迴歸分析於探勘使用者移動模式

Exploring Regression for Mining User Moving Patterns in a Mobile Computing System

研 究 生：洪智傑

指導教授：彭文志　教授

中 華 民 國 九 十 四 年 五 月

利用迴歸分析探勘使用者移動樣式於行動計算系統
Exploring Regression for Mining User Moving Patterns in a Mobile Computing System

研 究 生：洪智傑　　Student：Chih-Chieh Hung

指導教授：彭文志　　Advisor：Wen-Chih Peng

國 立 交 通 大 學

資 訊 工 程 學 系

碩 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Engineering

National Chiao Tung University

College of Eletrical Engineering and Computer Science

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

May 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年五月

博碩士論文授權書

本授權書所授權之論文為本人在 交通 大學(學院) 資訊工程 系所
 組 九十三 學年度第 二 學期取得 碩 士學位之論文。
論文名稱： 利用迴歸分析探勘使用者移動樣式於行動計算系統
指導教授： 彭文志

1.■同意　□不同意

本人具有著作財產權之上列論文全文(含摘要)資料，授予行政院國家科學委員會科學技術資料中心(或改制後之機構)，得不限地域、時間與次數以微縮、光碟或數位化等各種方式重製後散布發行或上載網路。

本論文為本人向經濟部智慧財產局申請專利(未申請者本條款請不予理會)的附件之一，申請文號為:_____，註明文號者請將全文資料延後半年再公開。

2. ■同意　□不同意

本人具有著作財產權之上列論文全文(含摘要)資料，授予教育部指定送繳之圖書館及國立交通大學圖書館，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會及學術研究之目的，教育部指定送繳之圖書館及國立交通大學圖書館得以紙本收錄、重製與利用；於著作權法合理使用範圍內，不限地域與時間，讀者得進行閱覽或列印。

本論文為本人向經濟部智慧財產局申請專利(未申請者本條款請不予理會)的附件之一，申請文號為:_____，註明文號者請將全文資料延後半年再公開。

3. ■同意　□不同意

本人具有著作財產權之上列論文全文(含摘要)，授予國立交通大學與台灣聯合大學系統圖書館，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會及學術研究之目的，國立交通大學圖書館及台灣聯合大學系統圖書館得不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間 –

本校及台灣聯合大學系統區域網路：　年　月　日公開

校外網際網路：　年　月　日公開

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未鉤選，本人同意視同授權。

研究生簽名：　　學號：9217582

（親筆正楷）　　（務必填寫）

日期：民國　94　年　5　月　24　日

1.本授權書請以黑筆撰寫並影印裝訂於書名頁之次頁。

國家圖書館博碩士論文電子檔案上網授權書

本授權書所授權之論文為本人在 交通 大學(學院) 資訊工程 系所
 組 九十三 學年度第 二 學期取得 碩 士學位之論文。
論文名稱： 利用迴歸分析探勘使用者移動樣式於行動計算系統

指導教授：彭文志

■同意　　□不同意

本人具有著作財產權之上列論文全文(含摘要)，以非專屬、無償授權國家圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未鉤選，本人同意視同授權。

研究生簽名：　　學號：9217582
（親筆正楷）　　（務必填寫）

日期：民國　94　年　5　月　24　日

---

1. 本授權書請以黑筆撰寫，並列印二份，其中一份影印裝訂於附錄三之一(博碩士論文授權書)之次頁；另一份於辦理離校時繳交給系所助理，由圖書館彙總寄交國家圖書館。

Ackowledgement

I spent a lot of time thinking about writing this part of the dissertation. There are many people I would like to thank and many of them have an influence on my life and this dissertation.

In order to reach the point of writing this dissertation, I conducted several research works under the supervision of my advisor Prof. Dr. Wen-Chih Peng. His overly enthusiasm and integral view on research has made a deep impression on me. I am also grateful for the interesting interactions and discussions we had during the two years I spent in National Chiao Tung University. With his constant guidance, timely encouragement and unwavering support, I have learned research capabilities and finished my dissertation. I always keep his advice: 'not pleased by external gains, not saddened by personal losses.' Besides of being an excellent advisor, he was as close as a relative and a good friend to me. I owe him lots of gratitude for having me shown this way of research and his care.

I also want to thank my committee members, Prof. Suh-Yin Lee, and Prof. Ming-Syan Chen, or all the comments. They made on this dissertation and also during my oral defence.

During the two years, I have also received encouragement from many friends and the members in Advanced Database System Laboratory. I want to thank my best friend Wan-Chun Yin. The solution procedures of this research work suddenly appeared in my brain when we had a talk. I will never forget the wonderful moment when I tasted the happiness of thinking. Also, the members in ADSLab gave me a pleasant atmosphere to do research, and I also indeed appreciate their friendship and cherish the time when we are together. At last, I would like to give particular thanks to Chun-Ling Lin for supporting me throughout this work, and in particular during the stressful writing-up period. Life would have been very difficult without you. Thanks for your consideration and thoughtfulness.

Finally, I could not reach the important milestone of my life without the support of my family. Thanks to my parents who have made it possible for me to reach where I am now standing. This dissertation is dedicated to them.

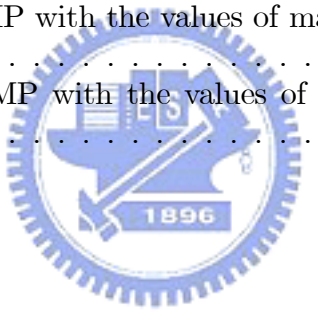
Abstract

In this thesis, by exploiting the log of call detail records, we present a solution procedure of mining user moving patterns in a mobile computing system. Specifically, we propose algorithm LS to accurately determine similar moving sequences from the log of call detail records so as to obtain moving behavior of users. By exploring the feature of spatial-temporal locality, which refers to the feature that if the time interval among consecutive calls of a mobile user is small, the mobile user is likely to move nearby, we develop algorithm TC to cluster those call detail records whose time intevals are very close. In light of the concept of regression, we devise algorithm MF to derive moving functions of moving behavior. Performance of the proposed solution procedure is analyzed and sensitivity analysis on several design parameters is conducted. It is shown by our simulation results that user moving patterns obtained by our solution procedure are of very high quality and in fact very close to real user moving behavior.

Index Terms — *user moving patterns, mobile computing, data mining, mobile database.*

Contents

List of Figures



List of Tables

Chapter 1

Introduction

Due to recent technology advances, an increasing number of users are accessing various information systems via wireless communication. Such information systems as stock trading, banking, wireless conferencing, are being provided by information services and application providers[3][5][9][11], and mobile users are able to access such information via wireless communication from anywhere at any time [16][22].

User moving patterns are referred to the areas where users frequently travel in a mobile computing environment. It is worth mentioning that user moving patterns are particularly important and are able to provide many benefits in mobile applications. A significant amount of research efforts has been elaborated upon issues of utilizing user moving patterns in developing location tracking schemes and data allocation methods [7][15]. We mention in passing that the authors in [7] developed a new location tracking strategy based on user moving behaviors. The authors in [15] devised data allocation schemes that are able to allocate data to the areas defined according to user moving patterns. Clearly, user moving patterns are beneficial on developing

Uid	Date	Time	Cellid
1	01/03/2004	03:30:21	A
1	01/03/2004	09:12:02	D
1	01/03/2004	20:30:21	G
1	01/03/2004	21:50:31	I

Table 1.1: An example of selected call detail records.

location management and querying strategy in a mobile computing system [7][15][18][20][21]. Thus, it has been recognized as an important issue to develop algorithms to mine user moving patterns so as to improve the performance of mobile computing systems.

The study in [15] explored the problem of mining user moving patterns with the moving log of mobile users given. Specifically, in order to capture user moving patterns, a moving log recording each movement of mobile users is needed. In practice, generating the moving log of all mobile users unavoidably leads to the increased storage cost and degraded performance of mobile computing systems. Consequently, in this paper, we address the problem of mining user moving patterns from the existing log of call detail records (referred to as CDR) of mobile computing systems. Generally, mobile computing systems generate one call detail record when a mobile user makes or receives a phone call. Table 1.1 shows an example of selected real call detail records where Uid is the identification of an individual user that makes or receives a phone call and Cellid indicates the corresponding base station that serves that mobile user. Thus, a mobile computing system produces daily a large amount of call detail records which contain hidden valuable information about the moving behaviors of mobile users. Unlike the moving log keeping track of the entire moving paths, the log of call detail records only reflects the fragmented moving behaviors of mobile users. However, such a fragmented moving behavior is of little interest in

a mobile computing environment where one would naturally like to know the complete moving behaviors of users. Thus, in this paper, with these fragmented moving behaviors hidden in the log of call detail records, we devise a solution procedure to mine user moving patterns. The problem we shall study can be best understood by the illustrative example in Figure 1.1 where the log of call detail records is given in Table 1.1. The dotted line in Figure 1.1 represents the real moving path of the mobile user and the cells with the symbol of a mobile phone are the areas where the mobile user made or received phone calls. Explicitly, there are four call detail records generated in the log of CDRs while the mobile user travels. The corresponding locations of these call detail records are scattered over the mobile computing environment, showing the limited information obtained from the log of CDRs when it comes to mining user moving patterns. Given these fragmented moving behaviors, we explore the technique of regression analysis to generate approximate user moving patterns (i.e., the solid line in Figure 1.1). As shown in Figure 1.1, the approximate user moving pattern (i.e., the solid line) is very close to the real moving behavior (i.e., the dotted line). In practice, approximate user moving patterns are able to provide sufficient user moving behaviors. For example, some mobile applications only require the moving trend of users. Furthermore, if approximate user moving patterns are close to the real moving paths, one can utilize approximate user moving patterns to predict the real moving behaviors of mobile users. Consequently, given the log of call detail records, we shall develop in this paper an efficient approach of mining user moving patterns close to real moving behaviors.

In this paper, we propose a regression-based solution procedure to mine user moving patterns. Regression analysis is widely applied in many scientific fields including statistics, economy, biological informatics and data mining. The main objective of regression analysis is that given

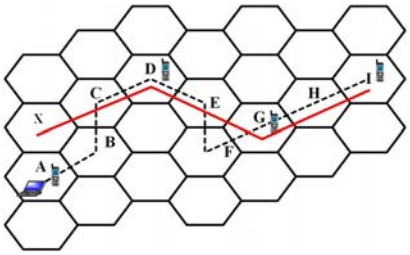


Figure 1.1: A moving path and an approximate user moving pattern of a mobile user

data points, a regression line is calculated with the purpose of minimizing the distance between the line derived and data points. Therefore, regression analysis is very suitable to mine user moving patterns with call detail records given. Compared to the moving log, call detail records reflect fragmented moving behavior of mobile users and thus call detail records are very precious for mining user moving patterns. However, the moving behavior of mobile users may scatter widely, making the traditional regression analysis not directly applicable to call detail records. To remedy this, three important issues, which we shall explicitly address and reflect in the design of a regression-based solution procedure for mining moving patterns, are as follows:

- **Extracting regular moving behavior**

  Note that call detail records not only contain the regularity of user moving behaviors but also have noise data accidentally generated. For example, a mobile user has some call detail records during his vacation. These call detail records are viewed as noise data in this paper. Since regression analysis is sensitive to noise data, we shall first rule out the noise data, i.e. those call detail recorded generated accidentally to increase the accuracy of regression analysis.

- **Exploiting spatial-temporal locality**

  Call detail records reflect the fragmented moving behavior of mobile users. If all call

detail records are put in regression analysis, it is likely that the regression line derived is not very close real moving behaviors of users. Note that the moving behavior of mobile users usually has spatial-temporal locality, which refers to the feature that if the time interval between two consecutive calls of a mobile user is small, the mobile user is likely to move nearby. In this paper, we will exploit spatial-temporal locality in our proposed algorithm.

- **Utilizing regression to generate moving patterns**

Regression analysis is able to derive the relationship among two or more random variables. User location is usually specified as 2-dimensional coordinates (i.e., x-axis and y-axis). Since x-axis and y-axis are not closely correlated in natural, we will properly divide user location into two dimensions and then utilize regression to derive moving behavior of mobile users.

Consequently, in this paper, we propose a solution procedure to mine approximate user moving patterns. Specifically, we shall first determine similar moving sequences from the log of call detail records and then these similar moving sequences are merged into one moving sequence (referred to as *aggregate moving sequence*). It is worth mentioning that to fully explore the feature of periodicity and utilize the limited amount of call detail records, algorithm LS (standing for Large Sequence) devised is able to accurately extract those similar moving sequences in the sense. By exploiting the feature of spatial-temporal locality, algorithm TC (standing for Time Clustering) developed should cluster those call detail records whose occurring time are close. For each cluster of call detail records, algorithm MF (standing for Moving Function), a regression-based method, devised is employed to derive moving functions of users so as to generate approximate user moving patterns. Performance of the proposed solution procedure is analyzed and sensitivity analysis on several design parameters is conducted. It is shown by our

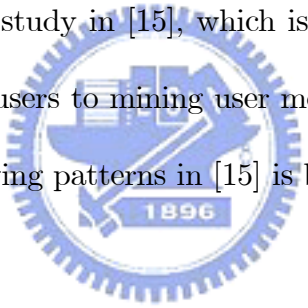
simulation results that approximate user moving patterns obtained by our proposed algorithms are of very high quality and in fact very close to real moving behaviors of users.

The rest of the thesis is organized as follows. Related works are described in Chapter 2. Algorithms for mining user moving patterns are devised in Chapter 3. Performance results are presented in Chapter 4. This thesis concludes with Chapter 5.

Chapter 2

Related Works

A significant amount of research works has been elaborated on mining user moving patterns. Among these research works, the study in [15], which is very related to the proposed method, exploited a moving log of mobile users to mining user moving patterns. Hence, in this chapter, the prior work of mining user moving patterns in [15] is briefly described.



2.1 Generation of Moving Log

In a mobile environment, each mobile user is associated with a home location database which maintains an up-to-date location data for the mobile user. The location management procedure for a mobile computing system considered in [15] is similar to the one in IS-41/GSM [4] [12], which is a two level standard and uses a two-tier system of home location register (HLR) and visitor location register (VLR) databases. Each mobile user is associated with an HLR. HLR databases maintain recent mobile users' records and current locations. A copy of the mobile user's record will be created in its local VLR while a mobile user moves out the area maintained

by its HLR. The record in the HLR is updated to reflect the movement of that user. The above procedure is so-called registration.

In order to capture user moving patterns, a movement log is needed. Each node in the network topology of a mobile computing system can be viewed as a VLR and each link is viewed as the connection between VLRs. Specifically speaking, a movement log contains a pair of (old VLR, new VLR) in the database when registration occurs. For each mobile user, we can obtain a moving sequence $\{(O_1, N_1), (O_2, N_2),...(O_n, N_n)\}$ from the movement log.

2.2 Incremental Mining for Moving Patterns in a Mobile Environment

Once the movement log is generated, we shall convert the log data into multiple subsequences, each of which represents a *maximal moving sequence*. After maximal moving sequences are obtained, we shall find frequent moving patterns among maximal moving sequences. A sequence of k movements is called a *large k-moving sequence* if there are a sufficient number (referred as support) of maximal moving sequences containing this k-moving sequence. After large moving sequences are determined, moving patterns can then be obtained in a straightforward manner. A moving pattern is a large moving sequence that is not contained in any other moving patterns. For example, let {AB, BC, AE, CG, GH} be the set of large 2-moving sequences and {ABC, CGH} be the set of large 3-moving sequences. We can obtain the user moving patterns {AE, ABC, CGH}. As we mentioned above, user moving patterns indicate the areas that users frequently travel in a mobile computing system.

The overall procedure for mining moving patterns is outlined as follows.

Procedure for incremental mining of moving patterns

Step 1. (Data collection phase) Employing algorithm MM to determine maximal moving sequences from a set of log data and also the occurrence count of moving pairs.

Step 2. (Incremental mining phase) Employing algorithm LM to determine large moving sequences for every w maximal moving sequence obtained in Step 1, where w is the retrospective factor which is an adjustable window size for the recent maximal moving sequences to be considered.

Step 3. (**Pattern generation phase**) Determine user moving patterns from large moving sequences obtained in Step 2, where user moving patterns are those frequent occurring consecutive subsequences among maximal moving sequences.

Note that in the data collection phase, the occurrence counts of moving pairs are updated on-line during registration procedure. Note that algorithm LM is executed to obtain new moving patterns in an incremental manner for every w maximal moving sequence generated, where the unit of w is the number of maximal moving sequences. As users travel, their moving patterns can be discovered incrementally to reflect the user moving behavior.

2.2.1 Finding Maximal Moving Sequences

Given a moving sequence $\{(O_1, N_1), (O_2, N_2), ...(O_n, N_n)\}$ of a user, we shall map it into multiple subsequences, each of which represents a maximal moving sequence. First, we can obtain a moving sequence $\{(O_1, N_1), (O_2, N_2), ...(O_n, N_n)\}$ for each mobile user from the movement log, where pairs of (O_i, N_i) are sorted by time. Then, algorithm MM (standing for maximal moving

Move	Maximal moving sequences output by algorithm MM
1	AB
2	ABC
3	ABCD
4	ABCDH
5	ABCDHG
6	ABCDHGH
7	ABCDHGHD
8	ABCDHGHDC
9	ABCDHGHDCB
10	ABCDHGHDCBA

Table 2.1: An illustrative example for algorithm MM

sequence), whose algorithmic form is given below, is applied to moving sequences of each mobile user to determine the maximal moving sequences of that user and update the occurrence count of moving pairs during registration procedure.

In algorithm MM, we use F to indicate if a node is revisited and Y to keep the current maximal moving sequence. D_F denotes the database to store all the resulting maximal moving sequences. S is the home location site of a mobile user. By the roundtrip model considered [10][17], the selection of S is either VLR or HLR whose geography area contains the homes of mobile users. Algorithm MM outputs a maximal moving sequence to D_F until the S is reached. In algorithm MM, moving sequences are scanned in line 2. A maximal moving sequence is output and a new maximal moving sequence will be explored (from line 14 to line 18) if MM finds that N_i in the moving pair (O_i, N_i) is the same as the starting site S. Otherwise, N_i is appended into Y (in line 12) and the occurrence count of (O_i,N_i) is updated on-line in the database (in line 14). An example execution scenario by algorithm MM is given in Table 2.1

Algorithm MM /* Algorithm MM for finding maximal moving sequences */
Input: A moving sequence $\{((O_1, N_1), (O_2, N_2), ...(O_n, N_n)\}$ of a mobile user.

Output: Maximal moving sequences of the mobile user.
begin
1. Set i to 1 and string Y to null , where Y is used to keep the current maximal moving sequence and S is the starting point.
2. **while** (not end of movings)
3. **begin**
4. Set $A = O_i$ and $B = N_i$;
5. **if** ($A == S$)
6. **begin**
7. Set Y=S;
8. Append B to Y;
9. **end**
10. **else**
11. **begin**
12. Append B to string Y;
13. Update the occurrence count of (A,B) in database D_F;
14. **if** ($B == S$)
15. **begin**
16. Output string Y to database D_F;
17. Set Y to null;
18. **end**
19. **end**
20. i++;
21. **end**

end

2.2.2 Finding Large Moving Sequences

As long as we obtain the maximal moving sequences, the large moving sequences are next to be determined. A large moving sequence can be determined from all maximal moving sequences of each individual user based on its occurrences in those maximal moving sequences. We define *intra-sequence count* to be the number of occurrences of a moving sequence within a maximal moving sequence, and *inter-sequence set* of a moving sequence to be the set of maximal moving sequences which contain that moving sequence. The count of a large moving sequence is the sum of intra-sequence counts from its inter-sequence set. For the example in Table 2.2, the

intra-sequence count of GB in {ABCGBCGBA} is 2 and that in {ABGBA} is 1. Also, the inter-sequence set of GB is {{ABCGBCGBA}, {ABGBA}}. Hence, the count of GB is the sum of intra-sequence counts in its inter-sequence set (i.e. 2 (i.e., intra-sequence count in ABCGBCGBA) +1 (i.e., intra-sequence count in ABGBA)=3). Algorithm LM (standing for large moving sequence) is then developed for the determination of large moving sequences. Let L_k represent the set of all large k-moving sequences and C_k be a set of candidate k-moving sequences.

Algorithm LM /* Algorithm for finding large moving sequences */
Input: A set of w maximal moving sequences of a mobile user.
Output: Large moving sequences of the mobile user.
begin
1. Determining L_2= {large 2-moving sequence} from moving pairs in C_2;
2. **for** $(k = 3; L_{k-1} \neq 0, k++)$
3. **begin**
4. $C_k = L_{k-1} * L_{k-1}$; /* Generating C_k from $L_{k-1} * L_{k-1}$ */
5. **for** w maximal moving sequence S
6. **begin**
/* Calculating the intra-sequence count of C_k within S */
7. intra-sequence =sub-sequence(C_k, S);
8. **if** (intra-sequence>0)
9. Including S into inter-sequence set;
/* sum of occurrence counts in a inter-sequence set */
10. **for** all candidate $c \in$ inter-sequence
11. c.count=c.count+c.intra-sequence;
12. **end**
13. L_k= $\{c \in C_k \mid$ c.count$\geqslant$support $\}$;
14. **end**
end

As pointed out in [14], the initial candidate set generation, especially for L_2, is the key issue to improve the performance of data mining. Since occurrence counts of moving pairs, i.e., C_2, were updated on-line in the data collection phase, L_2 can be determined by proper trimming on C_2 efficiently (line 1), showing the advantage of having on-line update in algorithm MM. Also,

C_2	Intra-sequence counts		Total count from the inter-sequence set
	ABCGBCGBA	ABGBA	
AB	1	1	2
BC	2	-	2
CG	2	-	2
BG	-	1	1
GB	2	1	3

Table 2.2: An example for counting the occurrences of 2-moving sequences

note that C_k can be simply generated from $L_{k-1} * L_{k-1}$(line 4). For example, with the set of L_2 being {AB, BK}, we have a C_3 as {ABK}. As explained above, the occurrence count of each k-moving sequence is the sum of intra-sequence counts (from line 5 to line 9 in algorithm LM) in its inter-sequence set (i.e., line 10 and line 11 in algorithm LM). Note that this step is very different from that in mining the path traversal patterns [1] where there are no loops in a moving sequence (i.e., the corresponding intra-sequence count is always zero). The occurrences of each k-moving sequence in C_k are determined for the identification of L_k. After the summation of the occurrence counts in the inter-sequence set from line 10 to line 11 in algorithm LM, those k-moving sequences with counts exceeding the support are qualified as L_k (line 13 of algorithm LM). Notice that those large k-moving sequences are obtained from w maximal moving sequences of that mobile user, showing the incremental mining capability of algorithm LM. For illustrative purposes, with the maximal moving sequences of a mobile user being $\{ABCGBCGBA, ABGBA\}$, Table 2.2 shows the corresponding counts of C_2.

As mentioned above, the main drawback of [15] is the generation of moving log for all mobile users. In practice, generating the moving log of all mobile users unavoidably leads to the increased storage cost and degraded performance of mobile computing systems. In order to reduce the effort

of generating moving log, we explore regression for mining user moving patterns from the existing log of call detail records (referred to as CDR).

Chapter 3

Mining User Moving Patterns

3.1 Preliminary

In this paper, assume that the moving behavior of mobile users have periodicity and consecutive movements of mobile users are not too far. Therefore, if the time interval of two consecutive CDRs is not too large, the mobile user is likely to move nearby. Two location models (i.e., geometric model and symbolic model) are available for the location identification techniques [2]. In geometric model, the location is specified as n-dimensional coordinates (typically n=2 or 3). For example, the location pair returned by global positioning system at time t is expressed by (X_t, Y_t) where X_t is the value of location in horizontal coordinate axis, whereas Y_t is the corresponding value of location in vertical coordinate axis. In symbolic model, the system uses logical entities to describe the location spaces. For example, in mobile computing systems, the base station identification is used to represent the location of mobile users. In our prior work [15], user moving patterns are represented in symbolic model (i.e., base station identification). In this

paper, we will take both two location models into consideration. To facilitate the presentation of this paper, a *moving section* is defined as a basic time unit. A *moving record* is a data structure that is able to accumulate the counting of base station identifications (henceforth referred to as *item*) appearing in call detail records whose occurring time are within the same moving section. Given a log of call detail records, we will first convert these CDR data into multiple moving sequences where a moving sequence is an ordered list of moving records and the length of the moving sequence is ε. The value of ε depends on the periodicity of mobile users and is able to obtain by proposed method in [6]. As a result, a moving sequence i is denoted by $<MR_i^1, MR_i^2, MR_i^3, ..., MR_i^\varepsilon>$, where MR_i^j is the jth moving record of moving sequence i. Assume the basic unit of a moving section is four hours and the value of ε is six. Given the log data in Table 1.1, we have the moving sequence $MS_1 = < \{A:1\}, \{\}, \{D:1\}, \{\}, \{\}, \{G:1, I:1\} >$. *Time projection sequence* of moving sequence MS_i is denoted as TP_{MS_i}, which is formulated as $TP_{MS_i} = < \alpha_1, ..., \alpha_n >$, where $MR_i^{\alpha_j} \neq \{\}$ and $\alpha_1 < ... < \alpha_n$. Explicitly, TP_{MS_i} is a sequence of numbers that are the identifications of moving sections in which the corresponding moving records are not empty. Given $MS_1 =< \{A:1\}, \{\}, \{D:1\}, \{\}, \{\}, \{G:1, I:1\} >$, one can verify that $TP_{MS_1} =< 1, 3, 6 >$. By utilizing the technique of sequential clustering, a time projection sequence TP_{MS_i} is able to divide into several groups in which time intervals among moving sections are close. For the brevity purpose, a clustered time projection sequence of TP_{MS_i}, denoted by $CTP(TP_{MS_i})$ is represented as $< CL_1, CL_2, ..., CL_x >$ where CL_i is the ith group and $i = [1, x]$. Note that the value of x is determined by our proposed method.

3.2 Procedure for Mining User Moving Patterns

In Section 3.1, we develop a solution procedure, which is composed of a sequence of algorithms in the corresponding phases, to mine approximate user moving patterns. Specifically, with the multiple moving sequences converted from the log of call detail records, we develop algorithm LS to identify those moving sequences beneficial to discover approximate user moving patterns in Section 3.2. Then, in Section 3.3, by exploring the spatial-temporal locality, we devise algorithm TC to cluster call detail records in time projection sequences. In Section 3.4, a regression-based algorithm MF is devised to mine approximate user moving patterns.

3.2.1 An Overview

The overall procedure for mining moving patterns is outlined as follows:

Procedure for Mining Approximate User Moving Patterns

Step 1. (Data Collection Phase) Employing algorithm LS to mine the regularity of moving sequences from original call detail records for every w moving sequence, where w is an adjustable window size for recent moving sequences to be considered.

Step 2. (Time Clustering Phase) Employing algorithm TC to cluster call detail records into groups and then generate a clustered time projection sequence.

Step 3. (Regression Phase) Employing algorithm MF to derive moving functions of mobile users from a clustered time projection sequence.

As mentioned before, once the log of call detail records is given, we shall covert the log data into multiple moving sequences, each of which is an ordered list of moving records. Generally speaking, call detail records not only contain the regularity of user moving behaviors but also

have noise data accidentally generated. For example, a mobile user has some call detail records during his vacation. These call detail records are viewed as noise data in this paper. As mentioned before, we explore the technique of regression analysis, which is very sensitive to noise data, to derive moving functions for mobile users. Thus, in data collection phase, algorithm LS is able to determine similar moving behaviors of mobile users for every w moving sequences. By exploring the feature of spatial-temporal locality, which refer to the feature that if the time interval between two consecutive calls of a mobile user is small, the mobile user is likely to move nearby, algorithm TC is employed to cluster call detailed records with spatial-temporal locality into several groups. After obtaining the clustering groups, we develop algorithm MF that takes both temporal and spatial data of similar moving records into consideration to determine approximate user moving patterns. The details of mining algorithms are described in the following subsections.

3.2.2 Data Collection Phase

As mentioned early, in this phase, we shall identify similar moving sequences from a set of w moving sequences obtained and then merge these similar moving sequences into one *aggregate moving sequence* (to be referred to as AMS). Algorithm LS is applied to moving sequences of each mobile user to determine the aggregate moving sequence that contains a sequence of large moving records denoted as LMR^i, where $i = [1, \varepsilon]$. Specifically, large moving record LMR^j is a set of items with their corresponding counting values if there are a sufficient number of MR_i^j of moving sequences containing these items. Such a threshold number is called $vertical\_min\_sup$ in this paper. Once the aggregate moving sequence is generated from these recent w moving sequences, we will then compare this aggregate moving sequence with these w moving sequences

so as to further accumulate the occurring counts of items appearing in each large moving record. The threshold to identify the similarity between moving sequences and the aggregate moving sequence is named by $match\_min\_sup$. The algorithmic form is given below.

Algorithm LS

input: w moving sequences with their length being ε,
two threshold:$vertical\_min\_sup$ and $match\_min\_sup$
output: Aggregate moving sequence AMS

```
1 begin
2   for j =1 to ε
3       for i=1 to w
4           LMR^j =large 1-itemset of MR_i^j;
                  (by vertical_min_sup)
5   for i = 1 to w
6     begin
7         match = 0;
8        for j = 1 to ε
9          begin
10           C(MR_i^j, LMR^j) = |x ∈ MR_i^j ∩ LMR^j| / |y ∈ MR_i^j ∪ LMR^j|;
11           match = match+|MR_i^j|*C(MR_i^j, LMR^j);
12         end
13       if match ≥ match_min_sup then
14         accumulate the occurring counts of
             items in the aggregate moving sequence;
15     end
16 end
```

In algorithm LS (from line 2 to line 4), we first calculate the appearing count of items in each moving sections of w moving sequences. If the count of an item among w moving sequence is larger than the value of $vertical\_min\_sup$, this item will be weaved into the corresponding large moving record. After obtaining all large moving records, AMS is then generated and is represented as $< LMR^1, LMR^2, ...LMR^{\varepsilon} >$, where the length of given moving sequence is ε. As mentioned before, large moving records contains frequent items with their corresponding counts. Once obtaining the aggregate moving sequence, we should in algorithm LS (from line 5 to line 12) compare this aggregate moving sequence with w moving sequences in order to identify those

similar moving sequences and then calculate the counts of each item in large moving records. Note that a moving sequence (respectively, AMS) consists of a sequence of moving records (respectively, large moving records). Thus, in order to quantity how similar between a moving sequence (e.g., MS_i) and AMS, we shall first measure the closeness between moving record MR_i^j and LMR^j, denoted by $C(MR_i^j, LMR^j)$. $C(MR_i^j, LMR^j)$ is formulated as $\frac{|\{x \in MR_i^j \cap LMR^j\}|}{|\{y \in MR_i^j \cup LMR^j\}|}$ that returns the normalized value in $[0, 1]$. The larger the value of $C(MR_i^j, LMR^j)$ is, the more closely MR_i^j resembles LMR^j. For example, we set large moving records $LMR^j = \{a, b, c, d\}$, $MR_x^j = \{b, e\}$ and $MR_y^j = \{a, b, c, d, e\}$. It can be verified that the value of $C(MR_x^j, LMR^j)$ is $\frac{1}{5}$ and the value of $C(MR_y^j, LMR^j)$ is $\frac{4}{5}$. Clearly, MR_y^j is more similar to LMR^j than MR_x^j is. Accordingly, the similarity measure of moving sequence MS_i and AMS is thus able to formulated as $sim(MS_i, AMS) = \sum_{i=1}^{\varepsilon} |MR_i^j| * C(MR_i^j, LMR^j)$. Given a threshold value $match\_min\_sup$, for each moving sequence MS_i, if $sim(MS_i, AMS) \geq match\_min\_sup$, moving sequence MS_i is identified as a similar moving sequence. In algorithm LS (from line 13 to line 14), for each item in large moving records, the occurring count is accumulated from the corresponding moving records of those similar moving sequences. Given an illustrative example in Table 3.1, we have $sim(MS_1, AMS) = 1*\frac{1}{2}+1*\frac{1}{1}+0+1*\frac{1}{2}+1*\frac{0}{1} = 2$. Consequently, we also have $sim(MS_2, AMS) = 3$, $sim(MS_3, AMS) = 2$, $sim(MS_4, AMS) = 3$, and $sim(MS_5, AMS) = \frac{1}{2}$. Assume that $match\_min\_sup$ is 2. Compared with AMS, MS_1, MS_2, MS_3 and MS_4 are then recognized as similar moving sequences. After identifying those similar moving sequences, algorithm LS is able to calculate the occurring count of each item in large moving records. Consider LMR^1 of AMS in Table 2 as an example. From those similar moving sequences, the occurring count of A in LMR^1 is calculated as the sum of the count of A in MR_1^1, that in MR_3^1 and that in MR_4^1

	1	2	3	4	5
MS_1	**A:14**	**A:2**		**F:1**	**I:2**
MS_2			**C:8**	**C:1, D:1, F:1**	**H:1, G:4**
MS_3	**A:1**	**C:1**		**D:1**	**H:1**
MS_4	**A:1, B:1**	**A:1**	**F:9**		
MS_5	**B:4**	**D:4**	**H:1**		**A:1, B:2**
AMS	**{A:16, B:1}**	**{A:3}**	ϕ	**{D:2, F:3}**	**{H:2}**

Table 3.1: An example of algorithm LS

(i.e., $14+1+1=16$). Following the same procedure, we could have $AMS <\{A:16, B:1\}, \{A:3\}, \phi, \{D:2, F:3\}, \{H:2\}>$ shown in Table 3.1.

It can be verified that algorithm LS is of polynomial time complexity. With w moving sequences and the length of a moving sequence being ε, the complexity of algorithm LS can be expressed by $O(\varepsilon\omega)$. Specifically, the complexity of calculating large moving records is $O(\varepsilon\omega)$ and that of extracting regular moving sequences is $\varepsilon * \omega * O(1) = O(\varepsilon\omega)$. As a result, the overall time complexity of algorithm LS is $O(\varepsilon\omega)$.

3.2.3 Time Clustering Phase

Recall that the time projection sequence of moving sequence MS_i is denoted as TP_{MS_i}, which is presented as $TP_{MS_i} =< \alpha_1, ..., \alpha_n >$, where $MR_i^{\alpha_j} \neq \{\}$ and $\alpha_1 < ... < \alpha_n$. Once obtaining AMS, we could easily determine TP_{AMS}. By exploring the feature of spatial-temporal locality, we will in this phase develop algorithm TC to generate a clustered time projection sequence of AMS (i.e., $CTP(TP_{AMS})$).

In algorithm TC, two threshold values (i.e., δ and σ^2) are given in clustering a time projection sequence. Explicitly, the value of δ is used to determine the density of clusters and σ^2 is utilized to

make sure that the spread of the time is bounded within σ^2. Algorithm TC is able to dynamically determine the number of groups in a time projection sequence.

Algorithm TC

input: Time projection sequence TP_{AMS},
threshold δ and σ^2
output: Clustered time projection sequence
$CTP(TP_{AMS})$

```
1   begin
2    group the numbers whose differences are within δ;
3    mark all clusters;
4    while there exist marked clusters and δ >= 1
5      for each marked clusters CL_i
6        if Var(CL_i) ≤ σ^2
7         unmark CL_i;
8         δ = δ − 1;
9       for all marked clusters CL_i
10        group the numbers whose differences are
          within δ in CL_i;
11    end while
12    if there exist marked clusters
13      for each marked cluster CL_i
14        k = 1;
15        repeat
16          k++;
17          divide evenly CL_i into k groups ;
18        until the spread degree of each group≤ σ^2;
19  end
```

By grouping those numbers together if the difference between two successive numbers is smaller than the threshold value δ, algorithm TC (from line 2 to line 3) first starts coarsely clustering TP_{AMS} into several marked clusters. As pointed out before, CL_i denotes the ith marked cluster. In order to make sure that quality of clusters, variance of CL_i, denoted as $Var(CL_i)$, is defined to measure the distribution of numbers in cluster CL_i. Specifically, $Var(CL_i)$ is the variance of a sequence of numbers. Hence, $Var(CL_i)$ is formulated as $\frac{1}{m}\sum_{k=1}^{m}(n_k - \frac{1}{m}\sum_{j=1}^{m} n_j)^2$, where n_k is the kth number in CL_i and m is the number of elements in CL_i. As can be seen from

line 5 to line 7 in algorithm TC, for each cluster CL_i, if $Var(CL_i)$ is smaller than σ^2, we unmark the cluster CL_i. Otherwise, we will decrease δ by 1 and with given the value of δ, algorithm TC (from line 8 to line 10) will re-cluster those numbers in unmark clusters. Algorithm TC partitions the numbers of TP_{AMS} iteratively with the objective of satisfying two threshold values, i.e., δ and σ^2, until there is no marked cluster or $\delta = 0$. If there is no marked clusters, $CTP(TP_{AMS})$ is thus generated. Note that, however, if there are still marked clusters with their variance values larger than σ^2, algorithm TC (from line 12 to line 18) will further finely partition these marked clusters so that the variance for every marked cluster is constrained by the threshold value of σ^2. If the threshold value of δ is 1, a marked cluster is usually a sequence of consecutive numbers in which the variance of this marked cluster is still larger than σ^2. To deal with this problem, we derive the following lemma:

Lemma 1: *Given a sequence of consecutive integers S_n with the length being n, the variance of S_n is $\frac{1}{12}(n^2-1)$.*

Proof:

Note that the variance of the sequence of consecutive integers with the same length is the same. For example, consider two sequences of consecutive integers: $\{1,2,3,4,5\}$ and $\{7,8,9,10,11\}$. It can be verified that $Var(\{1,2,3,4,5\}) = Var(\{7,8,9,10,11\})$. Without loss of generality, consider the variance $Var(\{1,2,3,...,n\})$.

Let $\bar{x}$ is the average of $\{1,2,...n\}$, $\bar{x} = (\frac{n(n+1)}{2})/n = \frac{n(n+1)}{2n} = \frac{1+n}{2}$. Then

$$\begin{aligned} Var(\{1,2,3,...,n\}) &= \tfrac{1}{n}(\sum_{x=1}^{n}(x-\bar{x})^2) \\ &= \tfrac{1}{n}(\sum_{x=1}^{n}x^2 - 2\bar{x}\sum_{x=1}^{n}x + \bar{x}\sum_{x=1}^{n}1) \\ &= \tfrac{(n+1)(2n+1)}{6} - (n+1)\bar{x} + \bar{x}^2 \end{aligned}$$

$$= \frac{(n+1)(2n+1)}{6} - \frac{(n+1)^2}{2} + (\frac{n+1}{2})^2$$

$$= \frac{(n+1)(2n+1)}{6} - \frac{(n+1)^2}{4}$$

$$= \frac{1}{12}(n+1)(4n+2-3n-3)$$

$$= \frac{1}{12}(n+1)(n-1)$$

$$= \frac{1}{12}(n^2-1) \qquad \square$$

Property: *Given a sequence of consecutive integers* $\{1,2,3,...,n\}$ *and a positive integer* k*, the optimal way of dividing* $\{1,2,3,...,n\}$ *into* k *clusters is to partition* $\{1,2,3,..,n\}$ *into* k *clusters with each cluster size being* $\lceil \frac{n}{k} \rceil$.

Proof:

Suppose $\{1,2,3,...,n\}$ is divided into $\{1,...,t_1\}\{t_1+1,...,t_2\},...,\{t_{k-1}+1,...,n\}$.

Let $t_0 = 1$, $t_k = n$, and $Var_i = Var(\{t_{i-1}, t_{i-1}+1,...,t_i\})$. Our goal is to find the point t_1, t_2, ..., and $t_{k-1}$with the purpose of minimizing $f = \sum_{i=1}^{k} Var_i$.

From lemma 1, $Var(\{1,2,...,n\}) = \frac{1}{12}(n^2-1)$, we have $f = \sum_{i=1}^{k} Var_i = \frac{1}{12}\sum_{i=1}^{k}((t_i - t_{i-1})^2 - 1)$.

To minimize $f = \sum_{i=1}^{k} Var_i$, the cutting points t_1, t_2, ..., and t_{k-1} are derived by letting first derivatives be zero.

$$\begin{cases} \frac{\partial f}{\partial t_1} = 4t_1 - 2t_2 - 2t_0 = 0 \\ \frac{\partial f}{\partial t_2} = 4t_2 - 2t_3 - 2t_1 = 0 \\ \qquad\qquad ... \\ \frac{\partial f}{\partial t_{k-1}} = 4t_{k-1} - 2t_k - 2t_{k-2} = 0 \end{cases}$$

Thus, we can have the following term:

$$\begin{cases} t_1 = \frac{t_{i0}+t_2}{2} \\ t_2 = \frac{t_1+t_3}{2} \\ \quad ... \\ t_{k-1} = \frac{t_{k-2}+t_k}{2} \end{cases}$$

By using substitution method, we could have

$$\begin{cases} t_1 = \frac{1}{2}t_2 \\ t_2 = \frac{2}{3}t_3 \\ \quad ... \\ t_{k-1} = \frac{k-1}{k}t_k \end{cases}$$

Therefore, we can get:

$$\begin{cases} t_1 = \frac{1}{k}n \\ t_2 = \frac{2}{k}n \\ \quad ... \\ t_{k-1} = \frac{k-1}{k}n \end{cases}$$

From the derivation above, the optimal way to divide $\{1, 2, 3, ..., n\}$ into k clusters is to divide $\{1, 2, 3, .., n\}$ into k clusters with each cluster size being $\lceil \frac{n}{k} \rceil$.

By the above property, given marked cluster CL_i, algorithm TC initially sets k to be 1. Then, marked cluster CL_i is evenly divided into k groups with each group size $\lceil \frac{n}{k} \rceil$. By increasing the value of k each run, algorithm TC is able to partition the marked cluster until the variance of each partition in the marked cluster CL_i satisfies σ^2.

Consider the execution scenario in Table 3.2 where the time projection sequence is TP_{AMS} = <1, 2, 3, 4, 5, 9, 10, 14, 17, 18, 20>. Given $\sigma^2 = 1.6$ and $\delta = 3$, algorithm TC first roughly partitions TP_{AMS} into three clusters. It can been verified in Table 3.2 that two marked clusters

Run	δ	σ^2	Clusters of a time projection sequence
0	3	1.6	$<\{1,2,3,4,5,9,10,14,17,18,20\}>$
1	3	1.6	$<\{1,2,3,4,5\}^*,\{9,10\},\{14,17,18,20\}^*>$
2	2	1.6	$<\{1,2,3,4,5\}^*,\{9,10\},\{14\},\{17,18,20\}>$
3	1	1.6	$<\{1,2,3,4,5\}^*,\{9,10\},\{14\},\{17,18,20\}>$
4	0	1.6	$<\{1,2,3,4,5\}^*,\{9,10\},\{14\},\{17,18,20\}>$
5	0	1.6	$<\{1,2,3\},\{4,5\},\{9,10\},\{14\},\{17,18,20\}>$

Table 3.2: An execution scenario under algorithm TC.

(i.e., $\{1,2,3,4,5\}$ with $Var(\{1,2,3,4,5\})$=2 and $\{14,17,18,20\}$ with $Var(\{14,17,18,20\})$=4.69) are determined due to that the variance values of these two clusters are larger than 1.6. Then, δ is reduced to 2, and these two marked clusters are examined again. Following the same procedure, algorithm TC partitions mark clusters until δ equals 1. As can been seen in Run 4 of Table 3.2, $\{1,2,3,4,5\}$ is still a marked cluster with $Var(\{1,2,3,4,5\})$=2. Therefore, algorithm TC finely partitions $\{1,2,3,4,5\}$. The value of k is initially set to be 1. Since $Var(\{1,2,3,4,5\})$=2.5 is larger than σ^2 (i.e., 1.6), k is increased to 2. Then, $\{1,2,3,4,5\}$ is divided into $\{1,2,3\}\{4,5\}$. Among these two clusters (i.e., $\{1,2,3\}$ and $\{4,5\}$), $\{1,2,3\}$ has the larger variance and thus $\{1,2,3\}$ is compared with the value of σ^2. Note that since the variance of $\{1,2,3\} = 0.67 < 1.6$, algorithm TC stops clustering. After the execution of algorithm TC, a $CTP(TP_{AMS})$ is generated as $<\{1,2,3\},\{4,5\},\{9,10\},\{14\},\{17,18,20\}>$.

The time complexity of algorithm TC is of polynomial time complexity. Explicitly, let TP_{AMS} have n numbers. In line 2 of algorithm TC, we have $O(n)$ to roughly divide sequence into t the clusters. Note that from line 4 to line 11 of algorithm TC, we have $O(\delta mt)$ to group the original sequence. From 13 to 19, assume that there are still t clusters with m numbers to be refined and then we have $t * m * (m) = O(m^2 t)$ to run the clustering process. Since algorithm TC is a

heuristic algorithm, we consider the worst case when estimating the time complexity of algorithm TC. Assume that the worst case is that $t = m = n$, and thus the overall time complexity of algorithm TC is at most $O(n^3)$.

3.2.4 Regression Phase

Given aggregate moving sequence AMS devised by algorithm LS with its clustered time projection sequence $CTP(\ TP_{AMS})$ generated by algorithm TC, in this phase, algorithm MF is able to derive a sequence of moving functions that are able to estimate moving behaviors of mobile users.

Assume that AMS is $<LMR^1, LMR^2, ..., LMR^{\varepsilon}>$ with its clustered time projection sequence $CTP(TP_{AMS}) = <\ CL_1,\ CL_2,\ ...,\ CL_k>$, where CL_i represents the ith cluster. For each cluster CL_i of $CTP(\ TP_{AMS})$, we will derive the estimated moving function of mobile users, expressed as $E_i(t) = (\ \hat{x}_i(t), \hat{y}_i(t),\ valid\_time\_interval\ )$, where $\hat{x}_i(t)$ (respectively, $\hat{y}_i(t)$) is a moving function in x-coordinate axis (respectively, in y-coordinate axis)) and the moving function is valid for the time interval indicated in $valid\_time\_interval$.

Without loss of generality, let CL_i be $\{t_1, t_2, ..., t_n\}$ where t_i is one of the moving section in CL_i. As described before, a moving record has the set of the items with their corresponding counts. Therefore, we could extract those large moving records from AMS to derive the estimated moving function for each cluster. In order to derive moving functions, the location of base stations should be represented in geometry model through a map table provided by tele-companies. Hence, given AMS and a cluster of $CTP(TP_{AMS})$, for each cluster of $CTP(TP_{AMS})$, we could have geometric coordinates of frequent items with their corresponding counts, which are able to

represent as (t_1,x_1,y_1,w_1), (t_2,x_2,y_2,w_2), ...(t_n,x_n,y_n,w_n). Accordingly, for each cluster of $CTP($ TP_{AMS}), regression analysis is able to derive the corresponding estimated moving function. By exploring the technique of regression analysis, the moving functions devised are able to generate the curves close to the data points and thus can be used to estimate users' moving behaviors.

The regression analysis fits equations of approximating curves to the raw field data [23]. For a given set of data, the fitting curves are generally not unique. Note that a curve with a minimal deviation from all data points is desired. Let e_i be the error between the ith data point and the estimated fitting curve. Given a set of data points, the best estimated curve is the one that has the minimal sum of least square errors (i.e., the minimal value of ϵ_x, where $\epsilon_x = \sum_{i=1}^{n} e_i^2$) [23]. Since the number of calls may be varied for each distinct (t_i, x_i, y_i), it is reasonable to derive moving functions by taking the weights into consideration. Therefore, the regression analysis with weighted least squares is then applied.

Given a cluster of data points (e.g., (t_1, x_1, y_1, w_1), (t_2, x_2, y_2, w_2), ..., (t_n,x_n,y_n,w_n)), we first consider the derivation of $\hat{x}(t)$. An m-degree polynomial function $\hat{x}(t) = a_0 + a_1 t + ... + a_m t^m$ will be derived to approximate moving behaviors in x-coordinate axis. Specifically, the regression coefficients $\{\alpha_0, \alpha_1, ...a_m\}$ are chosen to make the residual sum of squares $\epsilon_x = \sum_{i=1}^{n} w_i e_i^2$ minimal, where w_i is the weight of the data point (x_i, y_i) and $e_i = (x_i - (a_0 + a_1 t_i + a_2 (t_i)^2 ... + a_m (t_i)^m))$.The value of m is determined in accordance with the requirement of applications but m is usually smaller than the number of data points. To facilitate the presentation of our paper, we define the following terms:

$$\mathbf{H} = \begin{bmatrix} 1 & t_1 & (t_1)^2 & ... & (t_1)^m \\ 1 & t_2 & (t_2)^2 & ... & (t_2)^m \\ ... & ... & ... & ... & ... \\ 1 & t_n & (t_n)^2 & ... & (t_n)^m \end{bmatrix}, \mathbf{a}^* = \begin{bmatrix} a_0 \\ a_1 \\ ... \\ a_m \end{bmatrix}, \tilde{\mathbf{b}}_x = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}, \mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ ... \\ e_n \end{bmatrix}^T, \mathbf{W} = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & ... & \\ & & & w_n \end{bmatrix}.$$

It can be verified that the residual sum of squares (i.e., $\epsilon_x = \sum_{i=1}^{n} w_i e_i^2$) can be expressed as $\epsilon_x = \mathbf{e}^T\mathbf{W}\mathbf{e}$ in linear algebra manner. Note that $\mathbf{e}$ is able to be formulated as ($\tilde{\mathbf{b}}_x - \mathbf{H}\mathbf{a}^*$). Thus, we have:

$$\begin{aligned} \epsilon_x &= \mathbf{e}^T\mathbf{W}\mathbf{e} = (\tilde{\mathbf{b}}_x - \mathbf{H}\mathbf{a}^*)^T\mathbf{W}(\tilde{\mathbf{b}}_x - \mathbf{H}\mathbf{a}^*) = (\tilde{\mathbf{b}}_x - \mathbf{H}\mathbf{a}^*)^T\sqrt{\mathbf{W}}\sqrt{\mathbf{W}}(\tilde{\mathbf{b}}_x - \mathbf{H}\mathbf{a}^*) \quad [1] \\ &= (\sqrt{\mathbf{W}}\tilde{\mathbf{b}}_x - \sqrt{\mathbf{W}}\mathbf{H}\mathbf{a}^*)^T(\sqrt{\mathbf{W}}\tilde{\mathbf{b}}_x - \sqrt{\mathbf{W}}\mathbf{H}\mathbf{a}^*) \\ &= ||\sqrt{\mathbf{W}}\tilde{\mathbf{b}}_x - \sqrt{\mathbf{W}}\mathbf{H}\mathbf{a}^*|| \end{aligned}$$

let $\mathbf{A} = \sqrt{\mathbf{W}}\mathbf{H}$ and $\mathbf{B} = \sqrt{\mathbf{W}}\tilde{\mathbf{b}}_x$. It can be seen that the main objective is to minimize $\epsilon_x = ||\mathbf{B} - \mathbf{A}\mathbf{a}^*||$. According to the theorem of least squares, $\mathbf{B} - \mathbf{A}\mathbf{a}^*$ must be orthogonal to $\mathbf{A}\mathbf{a}^*$ so as to minimize ϵ_x[8]. For interest of brevity, the theorem of least squares is omitted in this paper. Consequently, we can have:

$$\mathbf{B} - \mathbf{A}\mathbf{a}^* \in R(\mathbf{A})^\perp = N(\mathbf{A}^T)$$

, where $R(\mathbf{A})^\perp$ represents the orthogonal complement of column space of $\mathbf{A}$

and $N(\mathbf{A}^T)$ represents the kernel space of $\mathbf{A}^T$

$$\Longrightarrow \quad \mathbf{A}^T(\mathbf{B} - \mathbf{A}\mathbf{a}^*) = 0$$

$$\Longrightarrow \quad \mathbf{A}^T\mathbf{A}\mathbf{a}^* = \mathbf{A}^T\mathbf{B}$$

$\mathbf{A}^T\mathbf{A}\mathbf{a}^* = \mathbf{A}^T\mathbf{B}$ is viewed as the normal equation [8]. By substituting $\mathbf{A} = \sqrt{\mathbf{W}}\mathbf{H}$ and $\mathbf{B} = \sqrt{\mathbf{W}}\tilde{\mathbf{b}}_x$, we could have $(\sqrt{\mathbf{W}}\mathbf{H})^T(\sqrt{\mathbf{W}}\mathbf{H})\mathbf{a}^* = (\sqrt{\mathbf{W}}\mathbf{H})^T\sqrt{\mathbf{W}}\tilde{\mathbf{b}}_x$. By solving the normal equation, we can a^* such that the value of ϵ_x is minimized. Therefore, $\hat{x}(t) = a_0 + a_1 t + ... + a_m t^m$

[1] Since $\mathbf{W}$ is diagonal and all elements are positive, we can decompose $\mathbf{W}$ into $\sqrt{\mathbf{W}}\sqrt{\mathbf{W}}$.

t_i	item	x_i	y_i	w_i
1	A	1	1	16
1	B	1	2	1
2	A	1	1	1
4	D	4	2	2
4	F	3	3	3
5	H	5	3	2

Table 3.3: Data points with their corresponding weights.

is obtained. Following the same procedure, we could derive $\hat{y}(t)$. As a result, for each cluster of $CTP(TP_{AMS})$, the estimated moving function $E_i(t) = (\hat{x}(t), \hat{y}(t), [t_1, t_n])$ of a mobile user is devised.

Consider an illustrative example in Table 3.1, where $AMS = <\{A : 16, B : 1\}, \{A : 3\}, \phi, \{D : 2, F : 3\}, \{H : 2\}>$. Assume that $CTP(TP_{AMS}) =< \{1, 2, 4, 5\} >$ and the coordinates of A, B, D, F and H are (1, 1), (1, 2), (4, 2), (3, 3) and (5,3), respectively. Given AMS and $CTP(TP_{AMS}) =< \{1, 2, 4, 5\} >$, we could have the data points with their weights shown in Table 3.3. By choosing m to be 3, the 3-degree polynomial $\hat{x}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ is derived. Due to that the coefficients a_0, a_1, a_2 and a_3 are unknown, we intend to have a regression curve with the purpose of minimizing the residual sum error. In other words, $\mathbf{a}^* = (\ a_0 \quad a_1 \quad a_2 \quad a_3\ )^T$ should be determined. Since there are five data points with their corresponding moving sections are 1, 2, 4, 4 and 5, $\mathbf{H} = \begin{bmatrix} 1 & 1 & (1)^2 & (1)^3 \\ 1 & 2 & (2)^2 & (2)^3 \\ 1 & 4 & (4)^2 & (4)^3 \\ 1 & 4 & (4)^2 & (4)^3 \\ 1 & 5 & (5)^2 & (5)^3 \end{bmatrix}$ is then obtained. Note that since the data pair (1, 1) appearing twice in Table 3.3, the weight of (1, 1) should be the

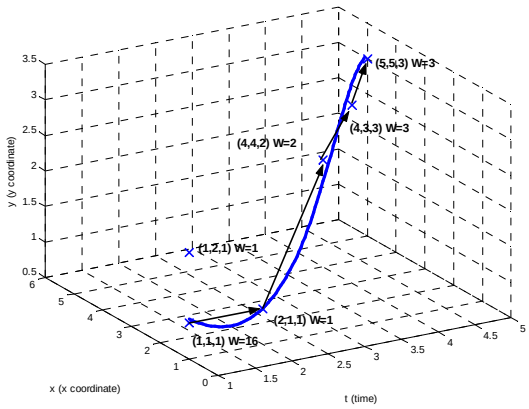


Figure 3.1: An illustrative example, where the arrow line is the real moving path and the solid line is estimated by moving functions obtained by algorithm MF.

sum of 16 and 1. The weights of data points are 17, 1, 2, 3 and 2, respectively. Hence, $\sqrt{\mathbf{W}}$ is a diagonal matrix with its diagonal entries to be $[\sqrt{17}, \sqrt{1}, \sqrt{2}, \sqrt{3}, \sqrt{2}]$. From Table 3.3, we can get $\tilde{\mathbf{b}}_x = (\ 1 \quad 1 \quad 4 \quad 3 \quad 5\ )^T$. By solving the equation $(\sqrt{\mathbf{W}}\mathbf{H})^T(\sqrt{\mathbf{W}}\mathbf{H})\mathbf{a}^* = (\sqrt{\mathbf{W}}\mathbf{H})^T\sqrt{\mathbf{W}}\tilde{\mathbf{b}}_x$, we can get $\mathbf{a}^* = (\ 2.333 \quad -2.133 \quad 0.867 \quad -0.066\ )^T$. Therefore $\hat{x}(t) = 2.333 - 2.133t + 0.867t^2 - 0.066t^3$ is devised to predict the x coordinate-axis of the mobile user from $t = 1$ to $t = 5$. Similarly, $\tilde{\mathbf{b}}_y = (\ 1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 3\ )^T$ is then determined from Table 3.3. By solving the normal equation $(\sqrt{\mathbf{W}}\mathbf{H})^T(\sqrt{\mathbf{W}}\mathbf{H})\mathbf{a}^* = (\sqrt{\mathbf{W}}\mathbf{H})^T\sqrt{\mathbf{W}}\tilde{\mathbf{b}}_y$, we can get $\mathbf{a}^* = (\ 2.529 \quad -2.386 \quad 1.021 \quad -0.105\ )^T$. Consequently, $\hat{y}(t) = 2.529 - 2.386t + 1.021t^2 - 0.105t^3$ is obtained. The estimated moving function is shown in Figure 3.1. It can be seen that the estimated moving function is very close to the real moving path, showing the advantage of utilizing regression in mining user moving patterns.

Algorithm MF
input: AMS and clustered time projection sequence
$CTP(TP_{AMS})$
output: A set of moving functions
$F(t) = \{E_1(t), U_1(t), E_2(t), ..., E_k(t), U_k(t)\}$
1 begin

```
2  initialize F(t)=empty;
3    for i= 1 to k-1
4      begin
5        doing regression on CL_i to generate E_i(t);
6        doing regression on CL_{i+1} to generate E_{i+1}(t);
7        t_1 =the last number in CL_i;
8        t_2 =the first number in CL_{i+1};
9         using inner interpolation to generate
            U_i(t) = (x̂_i(t), ŷ_i(t), (t_1, t_2));
10          insert E_i(t), U_i(t) and  E_{i+1}(t) in F(t);
11       end
12     if(1 ∉ CL_1)
13       generate U_0(t) and Insert U_0(t) into the head of F(t);
14     if(ε ∉ CL_k)
15       generate U_k(t) and Insert U_k(t) into the tail of F(t);
16   return F(t);
17 end
```

Given AMS and a cluster of $CTP(TP_{AMS}) = <CL_1, CL_2, ..., CL_k>$, algorithm MF is able to generate the whole estimated moving function, denoted as $F(t)$. $F(t)$ is represented as $\{U_0(t)$, $E_1(t)$, $U_1(t)$, $E_2(t)$, ..., $E_k(t)$, $U_k(t)\}$, where $E_i(t)$ is the estimated moving function in cluster i of $CTP(TP_{AMS})$ and $U_i(t)$ is the linkage moving function from $E_i(t)$ to $E_{i+1}(t)$. It is shown in algorithm MF (from line 5 to line 6) that for each cluster of $CTP(TP_{AMS})$, we could derive the corresponding estimated moving functions by the regression method mentioned above. Note that, however, it is possible that the first moving section is not in CL_1. If t_0 is the first number of CL_1 and $t_0 \neq 1$, the $U_0(t) = \{E_1(t_0), [1, t_0)\}$ is generated for the boundary condition. Otherwise, $U_0(t)$ will not be valid in $F(t)$. The situation of $U_k(t)$ is similar. The linkage moving function will be calculated by interpolation (in line 9 of algorithm MF). For example, assume that $CTP(TP_{AMS})$=<$\{1, 2, 4, 5\}\{7, 9, 10\}$>, $E_1(t)$ =$(2.333 - 2.133t + 0.867t^2 - 0.066t^3$, $2.529 - 2.386t + 1.021t^2 - 0.105t^3, [1, 5])$ and $E_2(t) = (10 - 2.17t + 0.17t^2, 32 - 6.33t + 0.33t^2, [7, 10])$. It can be verified that the first number of cluster $\{1, 2, 4, 5\}$ is 1. Thus $U_0(t)$ is invalid in $F(t)$.

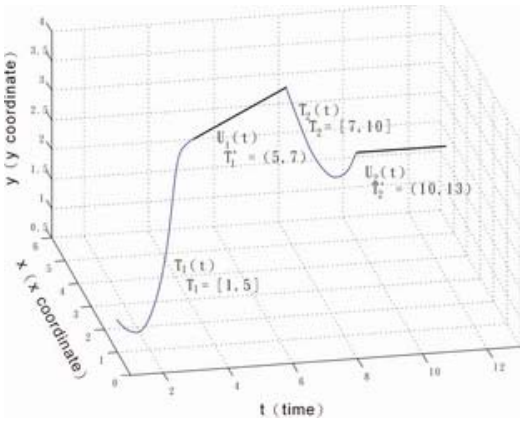


Figure 3.2: A snapshot of complete moving function $F(t)$

The last number of $\{1, 2, 4, 5\}$ is 5 and the first number of cluster $\{7, 9, 10\}$ is 7. Thus, a linkage moving function should be generated by inner interpolation. From $E_1(t)$, at time 5, we can have a data point ($x = 5.09$, $y = 3$). At time 7, a data point ($x = 3.14$, $y = 3.86$) is generated by applying $E_2(7)$. By inner interpolation, we could have $U_1(t) = (9.965 + \frac{3.14-5.09}{7-5}t,\ 0.85 + \frac{3.86-3}{7-5}t,$ (5,7)). Similarly, $U_2(t)$ can be produced. Thus, we could have $F(t) = \{E_1(t), U_1(t), E_2(t), U_2(t)\}$. The snapshot of F(t) is shown in Figure 3.2.

When using $F(t)$ to predict the location of a mobile user, we will only use the estimated moving function whose time interval includes the given time t. For the above example $F(t) = \{E_1(t), U_1(t), E_2(t), U_2(t)\}$, given the time to be 4, only $E_1(t)$ will be used to predict the location since the given time 4 is within the time interval of $E_1(t)$. Once the estimated moving function is obtained, it is straightforward to generate the approximate moving patterns in symbolic model. By utilizing the estimated moving function derived, the location of a mobile user is predicted as (x_t, y_t). Since each base station is aware of its location and converge area, as shown in Figure 3.3, we can obtain transform the geometric location (x_t, y_t) into base station D in symbolic model.

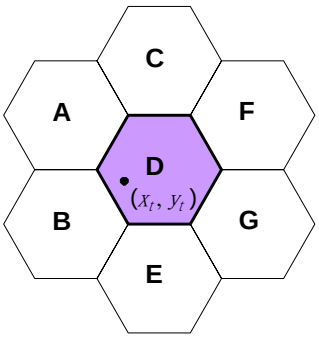


Figure 3.3: An illustrative example for transformation from geometric model to symbolic model

Note that the time complexity of algorithm MF is of polynomial time complexity. Specifically, with the maximal size in row/column being n, the time complexity of solving the normal equation by Strassen's algorithm is $\Theta(n^{\lg 7})$ [19]. Moreover, the interpolation by Lagrange's formula requires $\Theta(m^2)$, where m represents the number of points involved in the interpolation [19]. Since n is usually larger than m, the value of $\Theta(n^{\lg 7})$ is the dominating factor of the complexity of algorithm MF.

Chapter 4

Performance Study

In this section, the effectiveness of mining approximate user moving patterns by call detail records is evaluated empirically. The simulation model for the mobile system considered is described in Section 4.1. Section 4.2 is devoted to experimental results and comparison with the original algorithm of mining moving patterns [15]. Finally, sensitivity analysis of mining approximate user moving patterns is shown in Section 4.3.

4.1 Simulation Model for a Mobile System

To simulate base stations in a mobile computing system, we use an eight by eight mesh network, where each node represents one base station and there are hence 64 base stations in this model [13][15]. A moving path is a sequence of base stations travelled by a mobile user. The number of movements made by a mobile user during one moving section is modeled as a uniform distribution between mf-2 and mf+2. There are 10,000 users considered in our simulation model. According to Law of Large Number[23], we repeat each experiment for 20 times and every result presented

in the figures is the average performance of 20 experimental results. Explicitly, the larger the value of mf is, the more frequently a mobile user moves. To model user calling behavior, the calling frequency is employed to determine the number of calls during one moving section. If the value of cf is large, the number of calls for a mobile user will increase. Similar to [15], the mobile user moves to one of its neighboring base stations depending on a probabilistic model. To make sure the periodicity of moving behaviors, the probability that a mobile user moves to the base station where this user came from is modeled by P_{back} and the probability that the mobile user routes to the other base stations is determined by $(1\text{-}P_{back})/(n\text{-}1)$ where n is the number of possible base stations this mobile user can move to. We assign two P_{back} to each users to present the major and minor moving behavior respectively. As mentioned before, the method of mining moving patterns in [15], denoted as UMP, is implemented for the comparison purposes. For interest of brevity, our proposed solution procedure of mining user moving patterns is expressed by $AUMP$ (standing for approximate user moving patterns). The location is represented as the identifications of base stations. To measure the prediction accuracy, we use the hop count (denotes as hn), which is measured by the number of base stations, to represent the distance from the prediction location to the actual location of the mobile user. Intuitively, a smaller value of hn implies that the more accurate prediction is achieved. It is worth mentioning that the expected value of hop count per call, denoted by $E(hn/call)$, is $\frac{hn}{w*\varepsilon*cf/2}$ where $cf/2$ is the expected value of the number of CDR in a time unit. Thus a precise ratio is defined as $1 - \frac{E(hn/call)-1}{2n}$. Precise ratio is a measurement considering not only the distance between the moving patterns and real paths but also the ratio of the distance and the whole network size. Table 4.1 summarizes the definitions for some primary simulation parameters and the measurements of performance.

Notation	Definition	Value
w	retrospective factor	various value used
Ms	the number of moving sections in a moving sequence	various value used
Mf	moving frequency	various value used
Cf	call frequency	various value used
σ^2	variance threshold	various value used
$vertical\_min\_sup$	threshold of vertical minimal support	various value used
$match\_min\_sup$	threshold of match minimal support	various value used

Table 4.1: The parameters and measurements used in the simulation

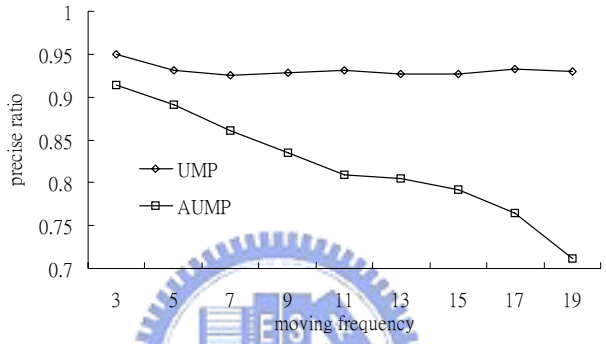


Figure 4.1: The precise ratio of UMP and AUMP with the value of mf varied.

4.2 Experiments of UMP and AUMP

To conduct the experiments to evaluate UMP and $AUMP$, we set the value of w to be 10, the value of cf to be 3 and the value of ε to be 12. The precise ratio of UMP and $AUMP$ with various values of mf are shown in Figure 4.1. It can be seen that by having a moving log, which contains the entire moving behaviors of users, the precise ratio of UMP is higher than that of $AUMP$.

As mentioned before, UMP is able to mine user moving patterns from a set of moving log in which every movement of mobile user is recorded. Note, however, that though performing

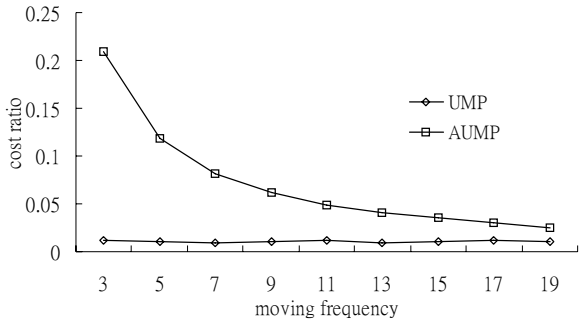


Figure 4.2: The cost ratios of AUMP and UMP with the moving frequency varied

better than $AUMP$ in the hop count, UMP incurs more amount of data in the moving log. In order to reduce the amount of data used in mining user moving patterns, $AUMP$ explores the log of call detail records. The cost ratio for a user, i.e., $\frac{precise\ ratio}{amount\ of\ log\ data}$, means the prediction accuracy gained by having the additional amount of log data. Figure 4.2 shows the cost ratios of UMP and $AUMP$. Notice that $AUMP$ has larger cost ratios than UMP, showing that $AUMP$ employs the amount of log data more cost-efficiently to increase the prediction accuracy.

4.3 Sensitivity Analysis of AUMP

The impact of varying the values of w for mining approximate moving patterns is next investigated. Without loss of generality, we set the value of ε to be 12, that of mf to be 3, and the values of cf to be 1, 3 and 5. Both $vertical\_min\_sup$ and $match\_min\_sup$ are set to 20% , the value of δ is set to be 3 , and σ^2 is set to be 0.25. With this setting, the experimental results are shown in Figure 4.3.

As can be seen from Figure 4.3, the precise ratio of AUMP increases as the value of w increases. This is due to that as the value of w increases, meaning that the number of moving

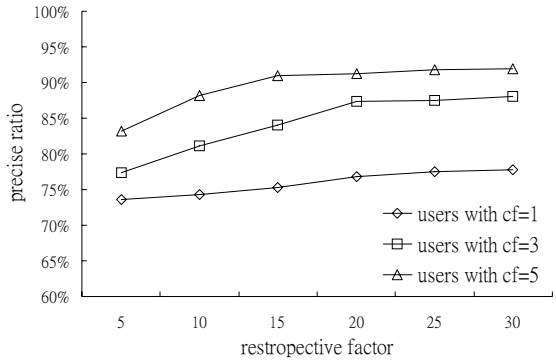


Figure 4.3: The performance of AUMP with the value of w varied.

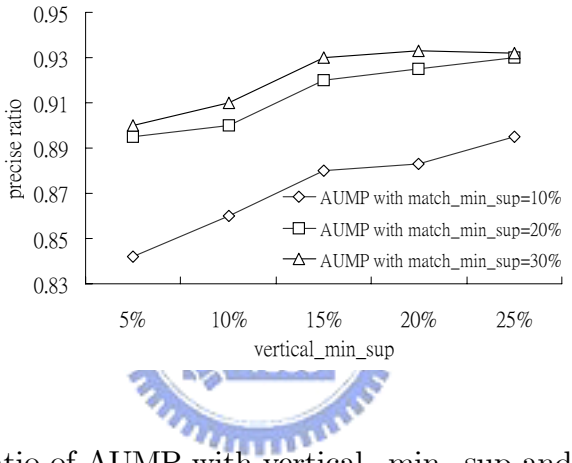


Figure 4.4: The precise ratio of AUMP with vertical_min_sup and match_min_sup varied.

sequences considered in AUMP increases, AUMP is able to effectively extract more information from the log of call detail records. Note that with a given the value of w, the precise ratio of AUMP with a larger value of cf is bigger, showing that the log of data has more information when the value of cf increases. Clearly, for mobile users having high call frequencies, the value of w is able to set smaller in order to quickly obtain moving patterns. However, for mobile users having low call frequencies, the value of w should be set larger so as to increase the accuracy of moving patterns mined by $AUMP$.

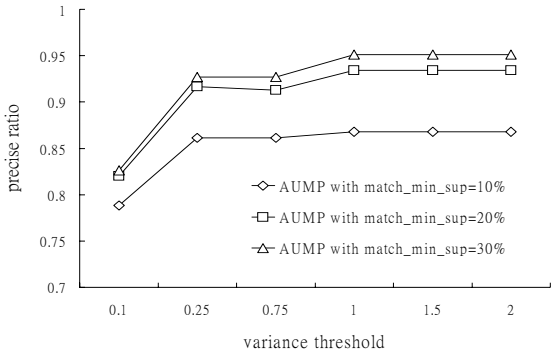


Figure 4.5: The precise ratio of AUMP with the values of match_min_sup and the variance threshold varied.

Now, the experiments of varying the values of *vertical_min_sup* and *match_min_sup* for algorithm LS are conducted where we set the value of cf to be 5, that of mf to be 1, that of ε to be 12 and that of σ^2 to be 0.25. The precise ratio of $AUMP$ with various values of *vertical_min_sup* and *match_min_sup* are shown in Figure 4.4, where it can be seen that the precise ratio of $AUMP$ with a given *vertical_min_sup* tends to increase as the value of *match_min_sup* increases. The reason is that increasing the *match_min_sup* is able to efficiently filter out call detail records that are viewed as noise data. As such, the precision of $AUMP$ with higher *match_min_sup* is larger. In addition, with a given *match_min_sup*, the precise ratio of $AUMP$ increases, as the value of *vertical_min_sup* increases. This is due to that as the value of *vertical_min_sup* increases, the more frequent set of base stations is determined from a set of call detail records. Thus, the frequent set of base stations, referring to areas that users more frequently travel, are very helpful to approximate user moving patterns.

As described before, the value of σ^2 for algorithm TC affects the precision of time clustering in $AUMP$. To conduct the experiments to evaluate $AUMP$ with the values of σ^2 varied, we set

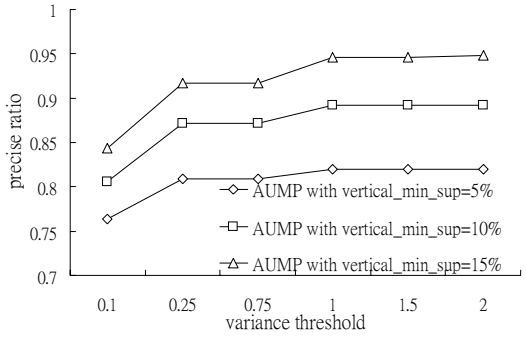


Figure 4.6: The precise ratio of AUMP with the values of vertical_min_sup and variance threshold varied.

the value of ε to be 12, that of mf to be 3, that of $vertical\_min\_sup$ to be 20%, and the values of $match\_min\_sup$ to be 1, 3 and 5, respectively. Figure 4.5 shows the precise ratio of $AUMP$ with the values of $match\_min\_sup$ and variance threshold σ^2 varied. As can been seen in Figure 4.5, the precise ratio of $AUMP$ tends to increase as the value of variance threshold σ^2 increases. This is mainly due to the reason that with larger values of $match\_min\_sup$, algorithm LS is likely to identify those similar moving sequences and thus the time projection of the aggregate moving sequence is very related to real moving behaviors of mobile users. Therefore, the precise ratio of $AUMP$ increases drastically when larger values of $match\_min\_sup$.

In addition, the hop counts of $AUMP$ with various values of $vertical\_min\_sup$ and variance threshold σ^2 are shown in Figure 4.6, where the $match\_min\_sup$ is set to be 20%. It can be seen that with a given value of variance threshold σ^2, the larger the value of $vertical\_min\_sup$, the larger the precise ratio of $AUMP$.

Chapter 5

Conclusions

In this paper, without increasing the overhead of generating the moving log, we presented a new mining method to mine user moving patterns from the existing log of call detail records of mobile computing systems. Specifically, we proposed algorithm LS to capture similar moving sequences from the log of call detail records and then these similar moving sequences are merged into the aggregate moving sequence. Algorithm LS devised is able to accurately extract those similar moving sequences in the sense that those similar moving sequences are determined by two adjustable threshold values (i.e., $vertical\_min\_sup$ and $match\_min\_sup$) when deriving the aggregate moving sequence from a set of call detail records. By exploring the feature of spatial-temporal locality, which refers to the feature that if the time intervals among consecutive calls of a mobile user is small, the mobile user is likely to move nearby, algorithm TC proposed is able to cluster those call detail records whose occurring time are very close from the aggregate moving sequence. For each cluster of the aggregate moving sequence, algorithm MF devised is employed to derive the estimated moving function, which is able to generate approximate user

moving patterns. Performance of the proposed algorithm was analyzed and sensitivity analysis on several design parameters was conducted. It is shown by our simulation results that the approximate user moving patterns achieved by our proposed algorithms are of very high quality and in fact very close to real user moving behaviors.

Bibliography

[1] M.-S. Chen, J.-S. Park, and P. S.Yu. Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, April 1998.

[2] B. Z. W.-C. L. Dik Lun Lee, Jianliang Xu. Data Management in Location-Dependent Information Services. In *Proceeding of IEEE Pervasive Computing*, pages 65–72, 2002.

[3] M. H. Dunham. Mobile Computing and Databases. *Tutorial of International Conference on Data Engineering*, February 1998.

[4] EIA/TIA. Cellular Radio Telecommunication Intersystem Operations. 1991.

[5] A. Elmagarmid, J. Jain, and T. Furukawa. Wireless Client/Server Computing for Personal Information Services and Applications. *ACM SIGMOD RECORD*, 24(4):16–21, December 1995.

[6] J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *Proceeding of the 15th International Conference on Data Engineering*, March 1999.

[7] J.-T. H. Hsiao-Kuang Wu, Ming-Hui Jin and C.-Y. Ke. Personal Paging Area Design Based On Mobile's Moving Behaviors. In *Proceeding of IEEE INFOCOM*, 2001.

[8] S. J.Leon. *Linear Algebra with Application*. Prentice-Hall International Inc., 2002.

[9] N. Krishnakumar and R. Jain. Escrow Techniques for Mobile Sales and Inventory Applications. *ACM Journal of Wireless Network*, 3(3):235–246, July 1997.

[10] D. Lam, D. C. Cox, and J. Widom. Teletrafic Modeling for Personal Communication Services. *IEEE Communications*, 35(2):79–87, February 1997.

[11] D. L. Lee. Data Management in a Wireless Environment. *Tutorial of International Conference on Database System for Advance Applications*, April 1999.

[12] Y.-B. Lin. GSM Network Signaling. *ACM Mobile Computing and Communications*, 1(2):11–16, 1997.

[13] Y.-B. Lin. Modeling Techniques for Large-Scale PCS Networks. *IEEE Communications Magazine*, 35(2):102–107, February 1997.

[14] J.-S. Park, M.-S. Chen, and P. S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):813–825, October 1997.

[15] W.-C. Peng and M.-S. Chen. Developing Data Allocation Schemes by Incremental Mining of User Moving Patterns in a Mobile Computing System. In *Proceeding of IEEE Transactions on Knowledge and Data Engineering, Volume 15*, pages 70–85, 2003.

[16] E. Pitoura and G. Samaras. Locating Objects in Mobile Computing. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):571–592, July/August 2001.

[17] N. Shivakumar, J. Jannink, and J. Widom. Per-User Profile Replication in Mobile Environments: Algorithms, Analysis and Simulation Results. *ACM Journal of Mobile Networks and Applications*, (2):129–140, 1997.

[18] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Proceeding of the 13th International Conference on Data Engineering*, pages 422–432, April 1997.

[19] R. L. R. Thomas H. Cormen, Charles E. Leiserson and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[20] S. Tseng and W. Chan. Efficiently Mining Complete User Moving Path in Mobile Systems. In *Proceeding of International Workshop on Database and Software Engineering (held with ICS)*, 2002.

[21] S. M. Tseng and C. F. Chiu. An Efficient Method for Mining Associated Service Patterns in Mobile Web Environments. In *Proceeding of ACM Symposium on Applied Computing*, March 2003.

[22] U. Varshney and R. Vetter. Emerging Mobile and Wireless Networks. *Communication of the ACM*, 43(6):73–81, June 2000.

[23] R. V.Hogg and E. A. Tanis. *Probability and Statistical Inference*. Prentice-Hall International Inc., 1997.