Data & Knowledge Engineering 70 (2011) 1–20

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

A regression-based approach for mining user movement patterns from random sample data

Chih-Chieh Hung, Wen-Chih Peng *

Department of Computer Science, National Chiao Tung University, Taiwan, ROC

ARTICLE INFO

Article history:
Received 6 July 2009
Received in revised form 13 July 2010
Accepted 13 July 2010
Available online 5 August 2010

Keywords:
User movement patterns
Data mining
Mobile data management

ABSTRACT

Mobile computing systems usually express a user movement trajectory as a sequence of areas that capture the user movement trace. Given a set of user movement trajectories, user movement patterns refer to the sequences of areas through which a user frequently travels. In an attempt to obtain user movement patterns for mobile applications, prior studies explore the problem of mining user movement patterns from the movement logs of mobile users. These movement logs generate a data record whenever a mobile user crosses base station coverage areas. However, this type of movement log does not exist in the system and thus generates extra overheads. By exploiting an existing log, namely, call detail records, this article proposes a Regression-based approach for mining User Movement Patterns (abbreviated as *RUMP*). This approach views call detail records as random sample trajectory data, and thus, user movement patterns are represented as movement functions in this article. We propose algorithm LS (standing for Large Sequence) to extract the call detail records that capture frequent user movement behaviors. By exploring the spatio-temporal locality of continuous movements (i.e., a mobile user is likely to be in nearby areas if the time interval between consecutive calls is small), we develop algorithm TC (standing for Time Clustering) to cluster call detail records. Then, by utilizing regression analysis, we develop algorithm MF (standing for Movement Function) to derive movement functions. Experimental studies involving both synthetic and real datasets show that *RUMP* is able to derive user movement functions close to the frequent movement behaviors of mobile users.

1. Introduction

Mobile services, such as navigation services, mobile search and location-aware services, are becoming very popular. These wireless communication systems enable users to access various kinds of information from anywhere at any time. A mobile computing system usually expresses a user movement trajectory as a sequence of areas in which the mobile user moves.[1] In this article, we aim at mining user movement patterns for a mobile user. Thus, given a user's set of movement trajectories, user movement patterns refer to the sequences of areas that this user frequently travels. Analysis of user trajectory data could provide some understandings and management of moving objects [1,2]. User movement patterns can be used to improve system performance, such as designing personal paging area [3], and developing data allocation strategies [4–6], querying strategies [7], and navigation services [8,9].

To discover user movement patterns in a mobile computing system, the methods proposed in previous studies require movement logs to record the movements of mobile users. For example, in [4,5], when a mobile user moves from the coverage area

* Corresponding author. No. 1001 University Road, Hsinchu, Taiwan 300, ROC. Tel.: +886 3 5731478.
E-mail address: wcpeng@cs.nctu.edu.tw (W.-C. Peng).

[1] This article defines a unit of a area as the coverage area of one base station. For ease of presentation, we simply use a base station identification to represent the corresponding coverage area.

0169-023X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2010.07.010

Table 1
An example of call detail records.

Uid	Date	Time	Cellid
1	Day 1	07:30	A
1	Day 1	09:32	D
1	Day 1	09:49	E
1	Day 1	13:50	H
1	Day 2	08:50	C
1	Day 2	09:50	E
1	Day 2	14:00	H
1	Day 3	07:15	A
1	Day 3	09:02	C
1	Day 3	09:30	D
1	Day 4	12:30	W
1	Day 4	12:52	X
1	Day 4	13:30	Y

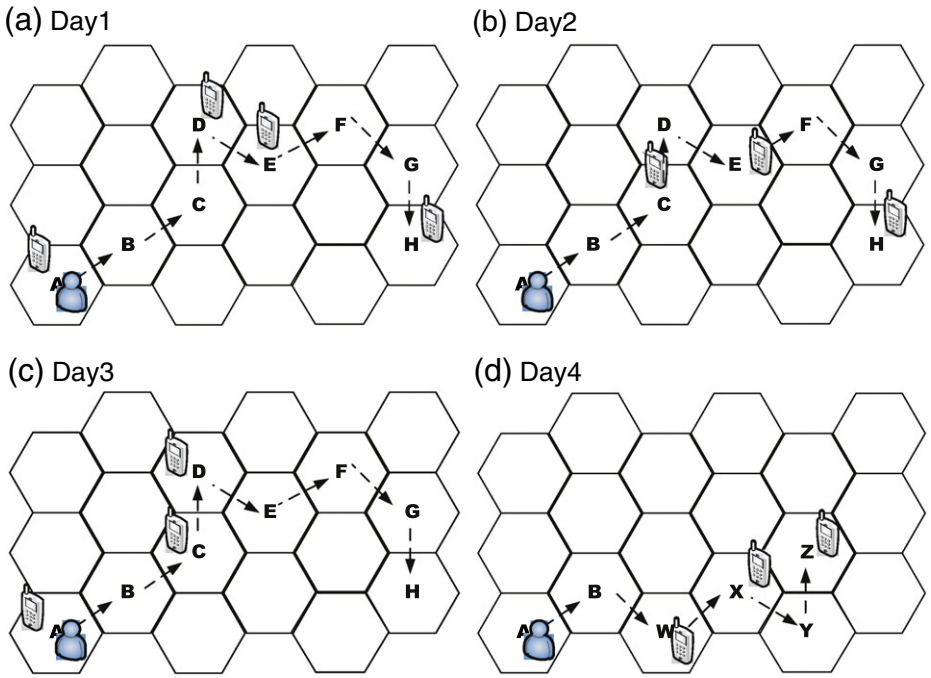

Fig. 1. An example of call detail records.

of base station *i* to the coverage area of base station *j*, a handoff procedure is performed to smoothly switch communication channels between base stations. Meanwhile, the movement log generates a movement pair $\langle i,j \rangle$. However, the movement log is not an existing log of mobile systems and needs some overheads to generate during handoff procedures. Hence, generating movement logs for all mobile users leads to increased storage costs and decreases the performance of mobile computing systems. Therefore, prior works are not practical for mobile computing systems due to the overhead of generating movement logs. In fact, mobile computing systems generate call detail records (abbreviated as CDR) when a mobile user makes or receives phone calls [10]. Table 1 shows an example of call detail records, where Uid represents the identification of a user making or receiving calls, and Cellid represents the corresponding base station that serves that user. Time information (i.e., date and time) is recorded in the CDR.[2] Table 1 shows that the CDRs of a mobile user contain both spatial (i.e., base station identification) and temporal information (i.e., date and time). Since CDRs reflect the movement behaviors of users, this article addresses the problem of mining user movement patterns from an existing log of CDRs, thereby reducing the overhead of generating a movement log.

Fig. 1 shows some trajectories of one user, where the dashed line represents one real trajectory of this user and the regions with mobile phones indicate that the user is receiving or making phone calls. This user's calling behavior is captured in the log of CDRs,

[2] The real call detail records analyzed in this study were provided by Taiwan mobile service providers, and we only extracted some useful attributes of call detail records to mine user movement patterns.

and Table 1 shows the CDR log. Fig. 1 shows that CDRs are data points that are randomly sampled from trajectories and the corresponding locations of these CDRs are scattered over the mobile computing environment. As a result, mining user movement behaviors from CDRs is a challenging task. Given these random sample data points, we aim to derive movement functions that are close to real user trajectories. We refer these movement functions as movement patterns due to that movement functions reflect the frequent movement behavior of users. Fig. 2 shows the movement function of a user for the example above.

This article proposes a novel approach, called *RUMP* (standing for Regression-based approach for User Movement Patterns), to mine user movement patterns from CDRs. Given a set of data points, the main objective of regression analysis is to derive a regression function that minimizes the sum of distances between the function derived and data points. In this approach, call detail records are viewed as data points, while the regression functions derived are regarded as movement functions. However, not all call detail records should be involved in mining user movement patterns. Without carefully selecting CDRs, user movement patterns cannot reflect the frequent movement behaviors of mobile users. On the other hand, CDRs should be fully utilized for mining user movement patterns since only limited information is available in the CDR logs. Thus, several issues remain to be addressed to efficiently utilize CDRs for mining user movement patterns.

1.1. Extracting frequent movement behaviors from CDRs

As mentioned before, user movement patterns refer to the frequent movement behaviors of mobile users. However, the CDR logs not only contain frequent user movement behaviors, but also include infrequent movement behaviors. For example, a user usually goes to his office and is back to his home every weekday (as Fig. 1(a), (b) and (c) shows), and occasionally takes a trip (as Fig. 1(d) shows). The frequent movement behavior is the trajectory from his home to his office, whereas a trip is an infrequent movement behavior. Since regression analysis is sensitive to these infrequent CDRs, they should be eliminated. In other words, the call detail records that capture the frequent movement behaviors of users should be extracted. To extract the frequent movement behaviors of mobile users, we develop algorithm LS (standing for Large Sequence) to extract base stations whose coverage areas are frequently visited by users.

1.2. Determining the number of regression functions

Once CDRs that capture the frequent movement behaviors have been extracted, it is necessary to determine how many regression functions are needed. If only one regression function is derived, it may not be very close to the frequent user movement behavior. Thus, given a set of call detail records of the frequent movement behavior, clustering techniques can be used to divide call detail records into several groups. The number of groups is viewed as the number of regression functions. The movement trajectories of mobile users generally follow spatio-temporal locality (i.e., if the time interval between two consecutive calls of a mobile user is small, the mobile user is likely to have moved nearby). Therefore, the feature of spatio-temporal locality in algorithm TC (standing for Time Clustering) can be used to group the call detail records with a close occurrence time.

1.3. Deriving movement functions

Location identification techniques typically use one of two location models: the geometric model and the symbolic models [11]. The geometric model specifies the location in n-dimensional coordinates (typically $n = 2$ or 3). The symbolic model, however, uses logical entities to describe the location. This article represents the location of mobile users in CDRs using the symbolic model (i.e., the base station identification). To derive movement functions of a mobile user, the location of the call detail records in the symbolic model must be transformed into the geometric model. Then, with the cluster results obtained, we develop algorithm MF (standing for Movement Function) for each cluster. This algorithm utilizes weighted regression analysis to derive the corresponding movement functions of a user.

The *RUMP* approach consists of a series of algorithms that tackle the various issues described above. This study evaluates *RUMP* performance using both synthetic and real datasets. Sensitivity analysis is conducted on several design parameters. Experimental results show that *RUMP* is able to efficiently and effectively derive user movement patterns that capture the frequent movement behaviors of mobile users.

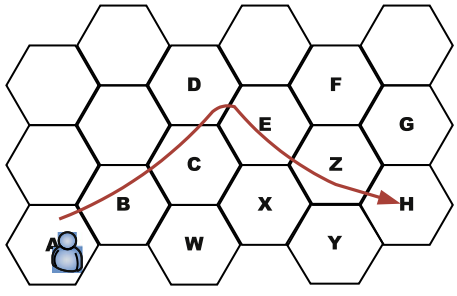

Fig. 2. A movement function of a user.

The rest of this article is organized as follows. Section 2 presents some related works. Section 3 then devises algorithms for mining user movement patterns. Section 4 presents performance results. Finally, Section 5 draws conclusions.

2. Related works

The problem of mining user movement patterns has attracted a considerable amount of research effort. Prior studies are generally classified into two categories based on their definitions of user movement patterns: spatial movement patterns and spatio-temporal movement patterns. In the first category, a user movement pattern refers to a sequence consisting of base station identifications or pre-defined regions. In the second category, user movement patterns represent the spatio-temporal associated relationships among base station identifications or pre-defined regions.

In the first category, the authors in [12] proposed an information–theoretical method to mine user movement patterns and represented them in a trie data structure. Moreover, the authors in [3] proposed a statistical approach to mine user movement patterns. The authors of [13] and [4] proposed a data mining approach for mining user movement patterns based on the movement logs of mobile users.

In the second category, user movement patterns are usually extracted from user trajectories, where trajectories are detailed user movements. A considerable amount of research efforts focuses on mining spatio-temporal association rules [14–19]. The authors in [20] explored the fuzziness of locations in patterns and developed algorithms to discover spatio-temporal sequential patterns. Furthermore, the authors in [21] proposed a clustering-based approach to discover movement regions within time intervals. In [22], the authors developed a hybrid prediction model, consisting of vector-based and pattern-based models, to predict user movements. In [23] and [24], the authors exploited temporal annotated sequences in which sequences are associated with time information (i.e., transition times between two movements).

To the best of our knowledge, this study proposes a new way of mining user movement patterns from random sample data points. Though the main theme of this article is to mine user movement patterns from call detail records, the proposed approach can be used for other log data with randomly sampled data features. Existing studies neither fully utilize fragmented spatio-temporal information (e.g., call detail records) for mining user movement patterns nor explore regression techniques for deriving movement functions. These features distinguish this article from others. Our preliminary work was presented in [25]. The current article extends this preliminary work with more detailed complexity and theoretical analysis. Furthermore, we conduct an extensive performance analysis on both synthetic and real datasets. Finally, this study investigates the sensitivity of several parameters, such as the calling behavior and thresholds for each algorithm.

3. A regression-based approach for mining user movement patterns

This section develops a regression-based approach (i.e., *RUMP*) consisting of a sequence of algorithms to mine user movement patterns. First, Section 3.1 provides an overview of *RUMP*, and the following sections present details of Algorithm LS, TC, and MF.

3.1. An overview

Given a log of CDRs, the goal of this article is to derive movement functions that closely reflect the frequent movement behaviors of mobile users. Due to that CDRs are random samples, the timestamps of CDRs are not likely to be the same even if a user follows the same movement behavior. Consequently, a basic time slot is defined as a time interval. For example, if call detail records whose occurrence time is within the time interval of one time slot, these CDRs are associated with the same time slot. Therefore, these CDRs are further put in a movement record defined as follows:

3.1.1. Definition: movement record

A movement record is defined as a set of pairs (BS_i:N_i), where BS_i is a base station and N_i is the number of occurrence counts of BS_i in call detail records whose occurring times are within the same time slot.

Table 2
Notations used in our algorithms.

Definition	Notation
Number of movement sequences	w
Movement sequence i	MS_i
Movement record at time slot j in MS_i	$MR_{i,j}$
A large movement sequence	LMS
Large movement record at time slot i	LMR_i
An aggregation movement sequence	AMS
Aggregated movement record at time slot i	AMR_i
A time projection sequence of AMS	TP_{AMS}
A clustered time projection sequence of AMS	CTP_{AMS}

Assume that one time unit has its time interval of 6:00 am to 10:00 am. From Table 1, in Day 1, we could have one movement record that includes {A:1, D:1, E:1} since the occurrence time of three call detail records (i.e., A, D, and E) is within the time interval (i.e., 6:00 am to 10:00 am). With the definition of movement records, a movement sequence is defined as follows:

3.1.2. Definition: movement sequence

A *movement sequence* MS_i, denoted by $<MR_{i,1}, MR_{i,2}, MR_{i,3}, ..., MR_{i,\varepsilon}>$, is an ordered sequence of ε movement records, where $MR_{i,j}$ is the movement record at time slot j in MS_i and ε is an adjustable parameter.

The length of a time slot determines the granularity of user movement patterns in terms of time. Same as in [22], the value of ε indicates that a movement pattern may re-appear. Thus, the value of ε depends on the periodicity of a user. Table 2 are notations used in our article. The overall procedure for mining movement patterns is outlined as follows:

3.1.2.1. Execution steps in RUMP

Step 1. (Extracting the aggregation movement sequence) In this step, call detail records are converted into w movement sequences, where w is an adjustable window size for the recent movement sequences being considered. *Algorithm LS* discovers an aggregation movement sequence, in which each movement record contains frequent areas that a user appears.

Step 2. (Clustering movement records) According to the aggregation movement sequence derived, we further develop *algorithm TC* to cluster movement records whose time slots are close.

Step 3. (Deriving movement functions) We then use regression techniques to derive the corresponding movement functions for each group in Step 2.

CDRs only reflect the fragmented movement behaviors of mobile users. Thus, the *RUMP* approach uses regression techniques to derive movement functions which are close to the frequent movement behaviors of mobile users. Due to the nature of regression techniques, without the proper determination of call detail records, user movement functions derived cannot capture the frequent movement behaviors of mobile users. On the other hand, call detail records should be fully utilized to mine user movement patterns since only limited information is available in CDRs. In the following subsections, each algorithm is presented in detail.

3.2. Algorithm LS: extracting the aggregation movement sequence

In this article, a user movement trajectory is represented as a sequence of base station identifications (hereafter, we use "base station" for short). Hence, call detail records are converted into movement sequences. With a set of movement sequences, algorithm LS determines an Aggregation Movement Sequence (abbreviated as *AMS*) and uses it to represent the frequent movement behaviors of a user. Intuitively, AMS is a sequence of movement records that have frequent base station and their corresponding counts at each time slot. At each time slot, a frequent base station in this article refers to a base station which a user appears more than *min_freq* times among movement sequences. The *min_freq* is given to quantify frequent base stations. As pointed out early, counts for frequent base stations should also be determined. Thus, before deriving AMS, a large movement sequence (abbreviated as LMS) is a sequence of frequent base stations and we use LMS to compute the similarity between LMS and each movement sequence. In light of similarity measurements obtained, we are able to identify those movement sequences capturing the frequent moving behavior of users and aggregate them as *AMS*.

3.2.1. Definition: large movement record

Given a set of movement sequences MS_1,MS_2...,MS_w and a threshold *min_freq*, a large movement record at time slot t is denoted as LMR_t and LMR_t contains a set of base stations whose occurrence count in the set of movement records at time slot t (i.e., $MR_{1,t}$, $MR_{2,t}$,...,$MR_{w,t}$) is larger or equal to *min_freq*.

Given five movement sequences in Table 3, if *min_freq* is set to 2, LMR_4 is {D,F} since both D and F have their occurrence count equal to *min_freq*. Large movement records demonstrate the frequent movement behavior of a user at each specific time slot. After obtaining large movement records at each time slot, a large movement sequence *LMS* is thus a sequence of large movement records, which is denoted as $LMS = <LMR_1,LMR_2,...,LMR_\varepsilon>$. Consequently, LMS indicates the frequent moving behavior of users.

Table 3
An example of algorithm LS.

	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
LMS	{A, B}	{A}		{D, F}	{H, I}
AMS	{A:16, B:1}	{A:3}	ϕ	{D:2, F:3}	{H:2}

Once a large movement sequence *LMS* is determined, we should further formulate the similarity between movement sequences and *LMS* to identify whether a movement sequence is the frequent movement behavior of a user or not. The conventional similarity measurements, such as Cosine similarity [26] and extended Jaccard coefficient [27], cannot be applied for the similarity measurement because they can only deal with scalar vectors with no missing values. Movement sequences and LMS are sequences of sets of base stations, not sequences of scalar values. Moreover, empty sets are allowed in movement sequences and LMS. As such, we formulate the similarity between a movement sequence (e.g., MS_i) and *LMS* as the closeness between movement records $MR_{i,j}$ and LMR_j, denoted by $C(MR_{i,j},LMR_j)$. $C(MR_{i,j},LMR_j)$ compares the set of base stations in $MR_{i,j}$ with the frequent base stations in LMR_j. $C(MR_{i,j},LMR_j)$ is formulated as $\frac{|\{x \in MR_{i,j} \cap LMR_j\}|}{|\{y \in MR_{i,j} \cup LMR_j\}|}$, and returns the normalized value in $[0, 1]$. The larger the value of $C(MR_{i,j},LMR_j)$, the more closely $MR_{i,j}$ resembles LMR_j. For example, assume that $LMRj = \{a,b,c,d\}$, $MR_{x,j} = \{b,e\}$ 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 $\frac{4}{5}$. Therefore, $MR_{y,j}$ is more similar to LMR_j. Based on the similarity between movement records and large movement records, the similarity measure of movement sequences MS_i and *LMS* is formulated as $sim(MS_i,LMS) = \sum_{j=1}^{\varepsilon} |MR_{i,j}| \; C(MR_{i,j}, LMR_j)$. Given a threshold value *min_sim*, for each movement sequence MS_i, if $sim(MS_i,LMS) \geq min\_sim$, the movement sequence MS_i is identified as a similar movement sequence. Consider the example in Table 3. It can be verified that $sim(MS_1, LMS) = 1\frac{1}{2} + 1\frac{1}{1} + 0 + 1\frac{1}{2} + 1\frac{0}{1} = 2$. Further, $sim(MS_2,LMS) = 3$, $sim(MS_3,LMS) = 2$, $sim(MS_4,LMS) = 3$, and $sim(MS_5, LMS) = \frac{1}{2}$. Assuming that *min _sim* is 2, MS_1, MS_2, MS_3 and MS_4 are recognized as similar movement sequences.

After identifying similar movement sequences, these similar movement sequences are aggregated as one *AMS* in which frequent base stations and their associated counts are determined. An aggregation movement sequence is defined as follows:

3.2.2. Definition: aggregation movement sequence

The aggregation movement sequence is denoted as $AMS = <AMR_1, AMR_2, ..., AMR_\varepsilon>$, where AMR_j is an aggregated movement record that contains frequent base stations, which are the same in large movement record LMR_j and their occurring counts accumulated from movement records at time slot *j* of similar movement sequences.

Consider the AMR_1 of *AMS* in Table 3 as an example. from those similar movement sequences, the occurrence count of *A* in AMR_1 is calculated as the sum of the count of *A* in $MR_{1,1}$, that in $MR_{3,1}$ and that in $MR_{4,1}$ (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.

3.2.3. Time complexity analysis

Given *w* movement sequences with ε time slots, the complexity of algorithm LS can be expressed as $O(\varepsilon\omega)$. The complexity involved in calculating large movement records is $O(\varepsilon\omega)$, while that of extracting frequent movement sequences is $\varepsilon * \omega * O(1) = O(\varepsilon\omega)$. As a result, the overall time complexity of algorithm LS is $O(\varepsilon\omega)$. Thus, algorithm LS is of polynomial time complexity.

Algorithm 1. Algorithm LS

Input: *w* movement sequences with length ε,two threshold:*min freq* and *min sim*
Output: aggregation movement sequence *AMS*

```
1:    begin
2:    for j ← 1 to ε do
3:      for i ← 1 to w do
4:        begin
5:        LMR_j ← frequent 1-itemset of MRi,j ; //by min freq
6:        end
7:    for i ← 1 to w do
8:      begin
9:      match ← 0;
10:     for j ← 1 to ε do
11:       begin
12:       C(MR_i,j, LMR_j) ← |x ∈ MRi,j ∩ LMR_j| / |y ∈ MR_i,j ∪ LMR_j|;
13:       match ← match + |MR_i,j| C(MR_i,j,LMR_j);
14:       end
15:       if match ≥ min sim then
16:         accumulate the occurring counts of items in the aggregation movement sequence;
17:   end
```

3.3. Algorithm TC: clustering aggregation movement records

As pointed out early, the movement trajectories of mobile users generally follow spatio-temporal locality (i.e., if the time interval between two consecutive calls of a mobile user is small, the mobile user is likely to have moved nearby). Accordingly, aggregation movement records in AMS could be clustered into several groups if their corresponding time slots are close. To

facilitate the presentation of this paper, only time information (i.e., time slots) is extracted from AMS. Thus, time projection sequence of AMS is defined as follows:

3.3.1. Definition: time projection sequence

A time projection sequence of *AMS* is expressed as $TP_{AMS}=<\alpha_1,...,\alpha_n>$, where $AMR_{\alpha_j}\neq\{\}$ and $\alpha_1<...<\alpha_n$.

A time projection sequence is a sequence of time slots in which the corresponding movement records are not empty. Algorithm TC then uses the time projection sequence to cluster close time slots. The cluster result of algorithm TC is represented as a clustered time projection sequence defined as follows:

3.3.2. Definition: clustered time projection sequence

A *clustered time projection sequence* of TP_{AMS}, denoted by CTP_{AMS}, is represented as $<CL_1,CL_2,...,CL_k>$, where the i-th group CL_i is the time slots of the clustered movement records, and k is an integer such that $1\leq k\leq\varepsilon$.

Given AMS obtained in Step 1, TP_{AMS} is then easily determined. By exploring the feature of spatial-temporal locality, algorithm TC generates a clustered time projection sequence of AMS (i.e., CTP_{AMS}). Each cluster in CTP_{AMS} contains close time slots. Those movement records with their time slots being clustered preserve the feature of spatio-temporal locality. Therefore, the objective of clustering is to bound the variance of time slots in each group with a given threshold (i.e., *min_var*).

The variance of a group CL_i is defined as $Var(CL_i)=\frac{1}{m}\sum_{k=1}^{m}\left(n_{i,k}-\frac{1}{m}\sum_{j=1}^{m}n_{i,j}\right)^2$, where $n_{i,j}$ represents the j-th time slots of movement records in CL_i and the total number of movement records in CL_i is m. Algorithm TC generates a clustered time projection sequence CTP_{AMS} such that $Var(CL_i)\leq min\_var$ for all clusters CL_i.

To achieve the objective of clustering, algorithm TC first starts coarsely clustering TP_{AMS} into several marked clusters using a value δ. The initial value of δ is set to ε and δ then decreases by one for each round. Thus, in the beginning, there is only one cluster. Dividing clusters with a variance larger than *min_var* increases the number of clusters. In algorithm TC, unmarked clusters refer to clusters that do not need to be refined, whereas marked clusters refer to clusters that should be further partitioned. For each cluster CL_i, if $Var(CL_i)$ is smaller than *min_var*, the cluster CL_i is unmarked. Otherwise, δ decreases by 1 and algorithm TC re-clusters the time slots in unmarked clusters with the updated value of δ. Algorithm TC partitions TP_{AMS} iteratively until no marked cluster remain or until $\delta=1$. If there are no marked clusters, CTP_{AMS} is generated. Otherwise, if there are still marked clusters with their variance values larger than *min_var*, algorithm TC continues to finely partition these marked clusters so that the variance for every marked cluster is constrained by the threshold value of *min_var*.

When the value of δ is 1, the time slots of movement records in a marked cluster generally follow a sequence of consecutive integers such that the variance of marked clusters is still larger than *min_var*. This situation results in loss of spatio-temporal locality. For example, given movement records with a sequence of consecutive time slots 1,2,3,4,5,6, and 7, though the differences of consecutive time slots are small, the location of a user at time slot 1 and that at time slot 7 are probably far from each other. To deal with this problem, this cluster must be further partitioned into smaller clusters. The variance of each refined cluster should be smaller than *min_var*. Moreover, to guarantee that no time slots of each refined clusters are as close as possible, the total variance of the refined clusters should be minimized. To derive the optimal method for further partitioning, the following Lemma is derived:

Lemma. *Given a cluster that has a sequence of consecutive integers* $1,2,3,...,n$ *and a positive integer* k*, the optimal method to minimize the sum of variance in each cluster and divide this cluster into* k *clusters is to partition it into* k *sub-clusters each with a size of* $\frac{n}{k}$.

Proof. Suppose that $<1,2,3,...,n>$ is divided into k sub-clusters: $<1,...,t_1>,<t_1+1,...,t_2>,...,<t_{k-1}+1,...,n>$. Let $t_0=0$, $t_k=n$, and $Var_i=Var(<t_{i-1}+1,...,t_i>)$. Our goal is to find the cutting points (i.e., t_1, t_2, ..., and t_{k-1}) to minimize $f=\sum_{i=1}^{k}Var_i$.

The variance remains the same constant for a sequence of consecutive integers with the same length. For example, consider two clusters with two sequences of consecutive time slots: $<1,2,3,4,5>$ and $<7,8,9,10,11>$. It can be verified that $Var(<1,2,3,4,5>)=Var(<7,8,9,10,11>)$. Since $Var(<1,2,...,n>)=\frac{1}{12}(n^2-1)$, we have $f=\sum_{i=1}^{k}Var_i=\frac{1}{12}\sum_{i=1}^{k}\left((t_i-t_{i-1})^2-1\right)$.

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

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

Thus, we can have the following terms:

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

Table 4
An execution scenario of algorithm TC.

Run	δ	*min_var*	Clusters
0	20	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>

Using substitution, we have

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

Therefore, we can get:

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

From the derivation above, the optimal way to divide $<1,2,3,...,n>$ into k sub-clusters is to divide $<1,2,3,...,n>$ into k sub-clusters each with size of $\frac{n}{k}$. □

This Lemma provides a guideline for partitioning a marked cluster that has a sequence of consecutive time slots into smaller clusters. Since the value of k is not known in advance, the value of k is initially set 2, and then increases in each iteration. In each iteration, a marked cluster is evenly divided into k sub-clusters, each with size of $\frac{n}{k}$, and the variance of each sub-cluster is tested. If the variance of a sub-cluster is smaller than *min _var*, the procedure terminates. Otherwise, the value of k is increased by 1 and the marked cluster will be further refined into smaller sub-clusters.

Consider the execution scenario in Table 4, where the time projection sequence is $TP_{AMS}=<1,2,3,4,5,9,10,14,17,18,20>$. The initial cluster is $<1,2,3,4,5,9,10,14,17,18,20>$. Given $min\_var=1.6$, algorithm TC first roughly partitions TP_{AMS} into three clusters. Table 4 shows that two marked clusters (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 because the variance values of these two clusters are larger than 1.6. Then, δ is reduced to 2, and these two marked clusters are re-examined. In the following run, the previous cluster $<14,17,18,20>$ is divided into two clusters, i.e., $<14>$ and $<17,18,20>$ in this run. Since $Var(<14>)=0<1.6$ and $Var(<17,18,20>)=1.56<1.6$, these two clusters remain unmarked. Following the same procedure, algorithm TC partitions marked clusters until δ equals 1. Run 4 in Table 4 shows that $<1,2,3,4,5>$ 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 at 1. Since $Var(<1,2,3,4,5>)=2.5$ is larger than *min _var* (i.e., 1.6), k increases to 2. Then, $<1,2,3,4,5>$ is divided into $<1,2,3><4,5>$. Of these two clusters (i.e., $<1,2,3>$ and $<4,5>$), the $<1,2,3>$ cluster has the larger variance and thus $<1,2,3>$ is compared with the value of *min_var*. Since the $Var(<1,2,3>)=0.67<1.6$, algorithm TC stops the clustering process. Finally, a CTP_{AMS} is generated as $<1,2,3>,<4,5>,<9,10>,<14>,<17,18,20>$.

3.3.3. Time complexity analysis

Algorithm TC is of polynomial time complexity. Let TP_{AMS} have n numbers. Algorithm TC needs $O(n)$ to divide TP_{AMS} into clusters from lines 5 to 15. From lines 17 to line 25, assume that there are still s clusters with m numbers to be refined

Algorithm 2. Algorithm TC

Input: time projection sequence:TP_{AMS}, thresholds: *min var*
Output: clustered time projection sequence:CTP_{AMS}
1: **begin**
2: $\delta \leftarrow \varepsilon$;
3: $CL_1 \leftarrow TP_{AMS}$;
4: Mark CL_1;
5: **while** there exist marked clusters and $\delta \geq 2$ **do**

Algorithm 2 (*continued*)

```
6:      begin
7:      for each marked clusters CL_i do
8:        if Var(CL_i) ≤ min var then
9:          begin
10:         unmark CL_i;
11:         end
12:     δ ← δ − 1;
13:     for all marked clusters CL_i do
14:       group the numbers whose differences are within δ in CL_i;
15:     end
16:
17:   if there are marked clusters then
18:     begin
19:     for each marked cluster CL_i do
20:       k = 2;
21:       repeat
22:         k ← k + 1;
23:         divide CL_i into k groups with equal sizes;
24:       until the variance of each group ≤ min var
25:       end
26:   end
```

Since k is at most m, we have $s\ O(m)$ to run the clustering process. The worst case occurs when estimating the time complexity of algorithm TC. In the worst case (i.e., $m = n$), the overall time complexity of algorithm TC is at most $O(n)$.

3.4. Algorithm MF: deriving movement functions

Given the aggregation movement sequence *AMS* devised by algorithm LS and its clustered time projection sequence CTP_{AMS} generated by algorithm TC, algorithm MF is able to derive a sequence of movement functions able to estimate the frequent movement behaviors of mobile users. For each cluster, we need to derive *confidence movement functions*. Then, *linkage movement functions* are determined to link confidence movement functions among clusters. Finally, a movement function $F(t)$ is derived and 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 confidence movement function in cluster CL_i of CTP_{AMS} and $U_i(t)$ is the linkage movement function from $E_i(t)$ to $E_{i+1}(t)$.

3.4.1. Deriving confidence movement functions

For each cluster CL_i of CTP_{AMS}, the confidence movement function of a mobile user, expressed as $E_i(t) = (\hat{x}_i(t), \hat{y}_i(t), TI_i)$, is derived. In this case, $\hat{x}_i(t)$ (respectively, $\hat{y}_i(t)$) is a movement function in x-coordinate axis (respectively, in y-coordinate axis) and the confidence movement function is valid for the time interval indicated in TI_i.

Without loss of generality, let CL_i be $<t_1, t_2,..., t_n>$, where t_j denotes one of the time slots in CL_i for $j = 1,2,...,n$. AMR^i contains frequent base stations with their corresponding counts in the i-th time slot of *AMS*. To derive movement functions, the location of base stations should be converted from the symbolic model into the geometric model through a map table that indicates the coordinates of base stations and is provided by telecompanies. Hence, given *AMS* and CTP_{AMS}, for each cluster of CTP_{AMS}, the geometric coordinates of frequent base stations can be derived along with their corresponding counts and represented as $(t_1,x_1,y_1,w_1),(t_2,x_2,y_2,w_2),...,(t_n,x_n,y_n,w_n)$ where t_i is the corresponding time slot, x_i (respectively, y_i) is the x-coordinate (respectively, y-coordinate) of the base station, and w_i is the number of phone calls a mobile user has made at this base station. Accordingly, for each cluster of CTP_{AMS}, a weighted regression analysis is able to derive the corresponding confidence movement function.

Given a set of data points, the goal of regression analysis is to derive the best estimated curve with the minimal sum of least square errors [28]. One aggregation movement sequence is generated in Step 1, which calculates the appearance counts of base stations. Thus, based on the appearance counts of base stations, we can derive curves closer to those base stations with larger appearance counts. This is because the more calls a user makes at a base station, the more confidence we have that this mobile user frequently appears in the coverage area of this base station. Another advantage of utilizing weighted regression analysis is that in a real scenario of mobile computing systems, the base station that serves to a user is not always the nearest base station. This is because other base stations nearby will cover the nearest base station when it becomes overloaded. However, the scenario above does not always happen. The appearing counts of other base stations will be fewer than that of the nearest base station. Therefore, weighted regression analysis makes it possible to derive curves close to base stations with higher appearance counts.

Given data points within a cluster, this article considers the derivation of the $\hat{x}(t)$. An m-degree polynomial function $\hat{x}(t) = a_0 + a_1 t + ... + a_m t^m$ is derived to approximate the movement behavior along x-coordinate axis. Given the data points (t_1,x_1,y_1,w_1), (t_2,x_2,y_2,w_2) ,..., (t_n,x_n,y_n,w_n), the regression coefficients $\{\alpha_0, \alpha_1, ...a_m\}$ are then selected to minimize the residual sum of squares $\epsilon_x = \sum_{i=1}^{n} w_i e_i^2$, where $e_i = (x_i - (a_0 + a_1 t_i + a_2 (t_i)^2 ... + a_m (t_i)^m))$. The value of m is application dependent, and must be smaller than the number of data points. The value of m is proportional to the precision of the fitting curve. Since $\hat{x}(t)$ is obtained by matrix operations, the matrix size is thus the dominant factor in regression performance. However, the impact of weighted regression analysis on execution time is not significant in this article since the maximal value of m is usually small. When

the value of m is small, the execution time of regression analysis is acceptable. Therefore, according to the number of data points available, the value of m should be set as large as possible.

For ease of presentation, the following terms are defined:

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

By solving the equation $\left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\left(\sqrt{\mathbf{W}}\mathbf{H}\right)\mathbf{a}^* = \left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\sqrt{\mathbf{W}}\,\tilde{\mathbf{b}}_x$, a^* can be derived such that the value of ϵ_x is minimized.[3] This leads to $\hat{x}(t) = a_0 + a_1 t + \ldots + a_m t^m$. $\hat{y}(t)$ can be derived following the same procedure. As a result, for each cluster of CTP_{AMS}, the confidence movement function $E_i(t) = (\hat{x}(t), \hat{y}(t), [t_1, t_n])$ of a mobile user can be devised.

For example, let $AMS = <\{A:16,B:1\},\{A:1\},\phi,\{D:2,F:3\},\{H:2\}>$ and the coordinates of A, B, D, F and H be (1, 1), (1, 2), (4, 2), (3, 3) and (5,3), respectively. Given AMS and $CTP_{AMS} = <1,2,4,5>$, it is possible to obtain data points with their weights, as Table 5 shows. By setting m to 3, the 3-degree polynomial $\hat{x}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ is derived. The coefficients a_0, a_1, a_2 and a_3 are determined by a regression curve that minimize the residual sum error. That is, $\mathbf{a}^* = (\mathrm{a}_0\ \mathrm{a}_1\ \mathrm{a}_2\ \mathrm{a}_3\ )^T$ must be determined. Since there are six data points with their corresponding time slots of 1, 1, 2, 4, 4 and 5, $\mathbf{H} = \begin{bmatrix} 1 & 1 & (1)^2 & (1)^3 \\ 1 & 1 & (1)^2 & (1)^3 \\ 1 & 2 & (2)^2 & (2)^3 \\ 1 & 4 & (4)^2 & (4)^3 \\ 1 & 4 & (4)^2 & (4)^3 \\ 1 & 5 & (5)^2 & (5)^3 \end{bmatrix}$ is then calculated. The weights of the data points are 16, 1, 1, 2, 3 and 2, respectively. Hence, $\sqrt{\mathbf{W}}$ is a diagonal matrix with the diagonal entries of $[\sqrt{16}, \sqrt{1}, \sqrt{1}, \sqrt{2}, \sqrt{3}, \sqrt{2}]$. Table 5 shows that $\tilde{\mathrm{b}}_x = (1\ 1\ 1\ 4\ 3\ 5\ )^T$. By solving the equation $\left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\left(\sqrt{\mathbf{W}}\mathbf{H}\right)\mathbf{a}^* = \left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\sqrt{\mathbf{W}}\,\tilde{\mathrm{b}}_x$, we can get $\mathbf{a} = (\ 2.333\ -2.133\ 0.867\ -0.066)^T$. 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, $\tilde{\mathrm{b}}_y = (\ 1\ 2\ 1\ 2\ 3\ 3\ )^T$ is then determined from Table 5. By solving the normal equation $\left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\left(\sqrt{\mathbf{W}}\mathbf{H}\right)\mathbf{a}^* = \left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\sqrt{\mathbf{W}}\,\tilde{\mathbf{b}}_y$, we can get $\mathbf{a}^* = (\ 2.529\ -2.386\ 1.021\ -0.105\ )^T$. We can obtain $\hat{y}(t) = 2.529 - 2.386t + 1.021t^2 - 0.105t^3$. Fig. 3 shows that the confidence movement functions, where the circle point indicates the location of a base station with its corresponding weight and the solid line is the curve derived by algorithm MF. The confidence movement function closely resembles actual movement behavior, demonstrating the advantage of utilizing regression analysis to mine user movement patterns.

Algorithm 3. Algorithm MF

Input: AMS and clustered time projection sequence $CTPAMS$
Output: a list of movement functions $F(t) = <E1(t), U1(t), E2(t), \ldots, Ek(t), Uk(t)>$

```
1:   begin
2:   F(t) = <>;
3:   for i = 1 to k − 1 do
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 time slot in CL_i;
8:     t_2 = the first time slot 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 then
13:    generate U_0(t) and Insert U_0(t) into the head of F(t);
14:  if ε ∉ CL_k then
15:    generate U_k(t) and Insert U_k(t) into the tail of F(t);
16:  end
```

3.4.2. Deriving linkage movement functions

Given the AMS and a cluster of $CTP_{AMS} = <CL_1, CL_2, \ldots, CL_k>$, algorithm MF generates the whole confidence movement function, denoted as $F(t)$. $F(t)$ is represented as $<U_0(t), E_1(t), U_1(t), E_2(t), \ldots, E_k(t), U_k(t)>$, where $E_i(t)$ is the confidence movement function in cluster CL_i of CTP_{AMS} and $U_i(t)$ is the linkage movement function from $E_i(t)$ to $E_{i+1}(t)$. Algorithm MF (from lines 5 to 6) shows that for each cluster of CTP_{AMS}, the corresponding confidence movement functions are derived using the regression method above. However, the first time slot may not be included in CL_1. If t_0 is the first time slot 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 same is true for $U_k(t)$. The linkage movement function is calculated by interpolation (in line 9 of algorithm MF).

[3] For the proof, see Appendix A.

Table 5
Data points with their corresponding weights.

t_i	ID	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
7	K	6	3	4
9	F	3	3	10
10	E	4	3	1

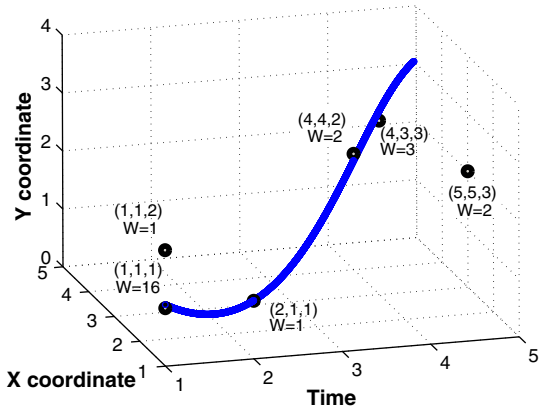

Fig. 3. An illustrative example of deriving confident movement functions.

For example, assume that CTP_{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 $E_2(t) = (6 + 1.17t - 0.16t^2, 3 + 0t + 0t^2, [7,10])$. It can be verified that the first time slot of cluster <1,2,4,5> is 1. The last time slot of <1,2,4,5> is 5 and the first time slot of cluster <7,9,10> is 7. Thus, a linkage movement function should be generated by inner interpolation. From $E_1(t)$, at the 5th time slot, we can have a data point ($x = 5.09$, $y = 3$). At the 7th time slot, a data point ($x = 6.35$, $y = 3$) is generated by applying $E_2(7)$. By inner interpolation, we could have $U_1(t) = \left(1.94 + \frac{6.35-5.09}{7-5}t, 3 + \frac{3-3}{7-5}t, (5,7)\right)$. Similarly, $U_2(t)$ can be determined. After obtaining the confidence and linkage functions, the $F(t)$ = <$E_1(t), U_1(t), E_2(t), U_2(t)$> can be derived. Fig. 4 shows the snapshot of $F(t)$. When using $F(t)$ to predict the location of mobile users, we will only use the confidence movement function whose time interval includes the given time t. For $F(t)$ = <$E_1(t), U_1(t), E_2(t), U_2(t)$>, when the time is 4, only $E_1(t)$ will be used to predict the location since the given time 4 is within the time interval of $E_1(t)$.

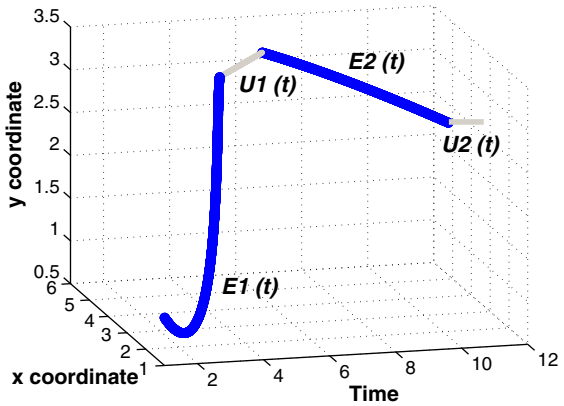

Fig. 4. A snapshot of a complete movement function $F(t)$.

3.4.3. Time complexity analysis

Algorithm MF is of polynomial time complexity. When the maximal size in row/column is n, the time complexity used to solve the normal equation by Strassen's algorithm is $\Theta(n^{lg7})$ [29]. Moreover, the interpolation by Lagrange's formula requires $\Theta(m^2)$, where m represents the number of points involved in the interpolation [29]. Since n is usually larger than m, the value of $\Theta(n^{lg7})$ dominates the complexity of algorithm MF.

3.5. Estimating a user's location based on a movement function

For many applications, it is necessary to estimate a user's location in the symbolic model. In this case, $F(t)$ represents the movement behavior of mobile users. Thus, once movement functions $F(t)$ have be obtained, the location of mobile users can be predicted as (x_t, y_t), which denotes the coordinates of applying the movement function at time t. Through the estimated coordinate (x_t, y_t), this coordinate can be transformed into a symbol which contains (x_t, y_t). In our example, since each base station is aware of its location and coverage area, it is easy to transform the geometric location (x_t, y_t) into the identification of the base station in the symbolic model.

4. Performance evaluation

This section evaluates the effectiveness and efficiency of mining user movement patterns from call detail records. Section 4.1 presents the models for user behaviors, including movement behavior and calling behavior. Section 4.1 also describes both the synthetic dataset and the real dataset. Section 4.2 presents experimental results. Finally, the *RUMP* sensitivity analysis is given in Section 4.3.

4.1. Modeling user behaviors

User behaviors in a mobile computing environment include movement behaviors and calling behaviors. This section first describes the synthetic dataset used in this study, in which user movement behaviors are derived according to pre-defined parameters. To simulate a mobile computing environment, we use a 16×16 mesh network, in which each node represents a base station. Thus, the simulation model contains 256 base stations [4,30]. Moreover, our simulation considers 10,000 users. As in [31], this simulation considers three movement trajectories. For each user, we randomly select one movement trajectory as his/her own movement pattern. Then, a user mostly follows his/her own movement pattern. However, users may have some movements that do not follow their movement patterns. These movements are viewed as biased movements. To prevent users from diverging too far from their movement patterns due to biased movements, we borrowed the concept in [17] that allows users to move back to their movement patterns. The number of movements made by mobile users in one time slot is modeled as a uniform distribution between $mf-2$ and $mf+2$. The larger the value of mf is, the more frequently mobile users move. We used the design above to generate user movements.

However, for a real dataset, it is difficult to obtain real call detail records from mobile service providers due to the privacy issue of customers. Moreover, the *RUMP* approach requires the location information of base stations, which is business-related information for mobile service providers. Thus, for real datasets, we use real movement logs from a GPS-based testbed, *CarWeb* [32], and generate simulated CDRs along with real movements. In the *CarWeb* platform, users can obtain their locations from a GPS device every five seconds and upload their locations to *CarWeb* servers. Fig. 5 shows one frequent movement behavior, where every red flag represents a user-uploaded location. By collecting user movement behaviors for four months, we produce roughly 200 movement trajectories for each user. In the *CarWeb* dataset, the ground truth is known, which is useful to validate our mining results.[4] In the *CarWeb* dataset, a user has frequent and infrequent movements. To simulate the coverage area of a base station, we divided the whole space into grids and viewed each grid as the coverage area of one base station. Fig. 5 shows the grids in the *CarWeb* datasets, where the frequent movement behaviors of this user occurred within or around 16 girds. Furthermore, since the traveling times of movement sequences in the *CarWeb* dataset are not exactly the same, the traveling time for each trajectory is thus normalized to 24 hours. In both datasets, the time slot is set to 2 hours and the value of ε is 12.

Once user movements have been determined, calling behaviors can be modeled for each user's movements. According to [30], calling behavior can be modeled as a Poisson distribution. Moreover, a Zeta distribution is used to model burst calling behavior in this article. In a Poisson distribution, the probability that a user has x calls in a time slot is determined by $P(x) = \frac{e^{-\lambda}\lambda^x}{x!}$, where x is the number of calls and λ is the expected number of calls in a time slot. Three time slots are grouped and then each time slot is divided into three subsections, producing a total of 9 subsections in each group. For each user, the probability of having phone calls in the x-th subsection of a group is $Z(x) = \frac{x^{-\lambda}}{\sum_{n=1}^{\infty}\frac{1}{n^{\lambda}}}$, where x indicates the subsection order in a group (i.e., the x-th subsection in a group) and λ is the value of the exponent feature for a Zeta distribution. In the beginning of subsections in a group, a user will have more phone calls, but the number of phone calls decays exponentially in the remaining subsections of a group. The speed of decay is determined by the parameter λ; the larger this parameter is, the faster the decay is. For brevity, $CDR(\rho,\lambda)$ indicates that the

[4] Due to customer privacy issues, it is impossible to get the ground truth of user movement behaviors even if mobile service providers were to release call detail records.

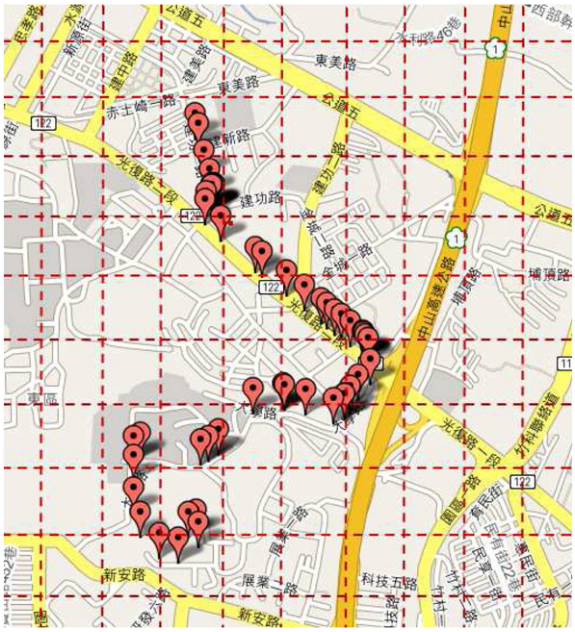

Fig. 5. The frequent movement behavior in CarWeb dataset.

calling behavior is modeled as ρ distribution with parameter λ, where the value of ρ is set to P (respectively, Z) if a Poisson (respectively, Zeta) distribution is used. For example, $CDR(P,2)$ represents the calling behavior of a user under a Poisson distribution with $\lambda = 2$.

For comparison purposes, we also implemented the method of mining movement patterns in [4], denoted by *UMP*. To validate the quality of movement patterns mined by *UMP* and *RUMP*, we could utilize movement patterns to predict next movements of users. The accuracy of prediction indicates the quality of movement patterns mined. Hence, the hop count (referred to as *hn*) represents the number of base stations between the prediction location and the actual location of the mobile user. Intuitively, the smaller the value of the hop count, the closer the current location and the derived location. Thus, the expected value of hop counts per call $E(hn/call)$ is defined as $\frac{total\_hop\_counts}{number\_of\_calls}$, where the *total _hop _counts* is the sum of hop counts per call and *number _of _calls* is the total number of calls per user. To evaluate the quality of user movement patterns mined by *UMP* and *RUMP*, the *precision ratio* is derived and defined as $1-\frac{E(hn/call)-1}{2n}$, where the size of network is $n \times n$ and $E(hn/call)$ is the expected value of hop counts per call. The precision ratio represents the percentage of the average hop counts from the derived cell to the current cell a mobile user with respect to the network size. Table 6 summarizes the definitions of some primary simulation parameters. In this table, the default values are optimal values based on following experiments in our experimental environment. Each experimental result was obtained by an average of twenty experiments.

4.2. Experiments of UMP and RUMP

We first evaluated *UMP* and *RUMP* in terms of the data amount, the precision ratio, and the execution time. The data amount is the number of records stored in a movement log and a CDR log. Fig. 6(a) shows that the data amount of *UMP* increases with the value of *mf*. This is because with a larger *mf*, a user tends to move frequently, producing a greater amount of data of the movement log. On the contrary, the data amount in *RUMP* remains almost constant. Fig. 6(b) shows that the precision ratio of *RUMP* is smaller than that of *UMP*. However, with CDR(P, 4), the precision ratio of *RUMP* is not far below *UMP*. Note, however,

Table 6
The parameters used in experiments.

Notation	Definition	Default value
w	Number of movement sequences	50
mf	Movement frequency	3
min_freq	Threshold used in algorithm LS	0.3
min_sim	Threshold used in algorithm LS	0.5
min_var	Threshold used in algorithm TC	0.75

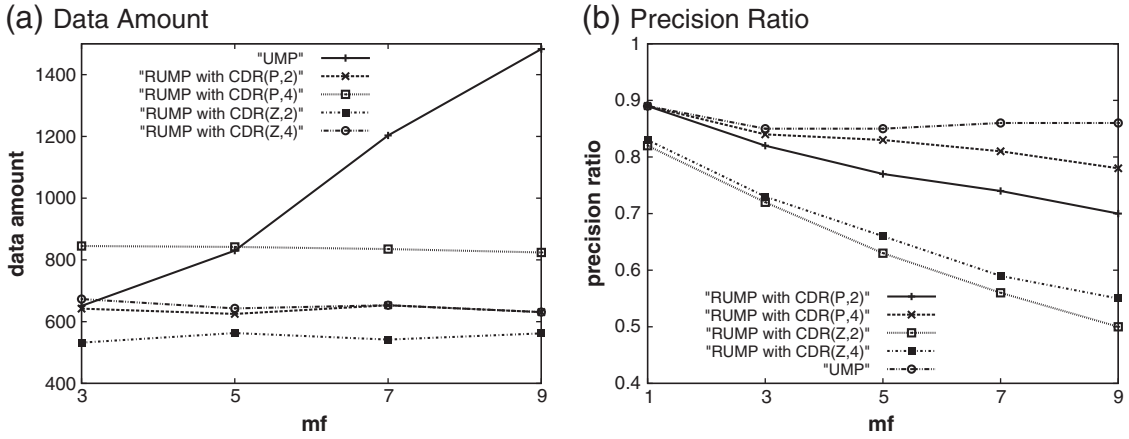

Fig. 6. Performance comparisons of *UMP* and *RUMP* on the synthetic dataset.

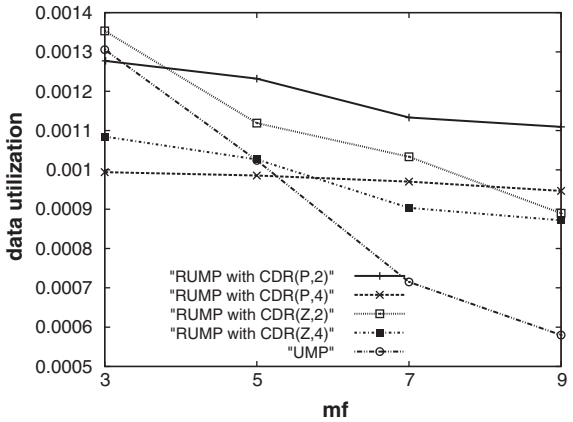

Fig. 7. Data utilization of *UMP* and *RUMP* on the synthetic dataset.

that though *UMP* performs better than *RUMP* in terms of the precision ratio, it also incurs a larger amount of data in a movement log. To investigate the precision ratio gained by having the additional amount of log data, this study defines *data utilization* as the ratio between the precision ratio and the amount of log data. Fig. 7 shows the data utilization of *UMP* and *RUMP*. With a

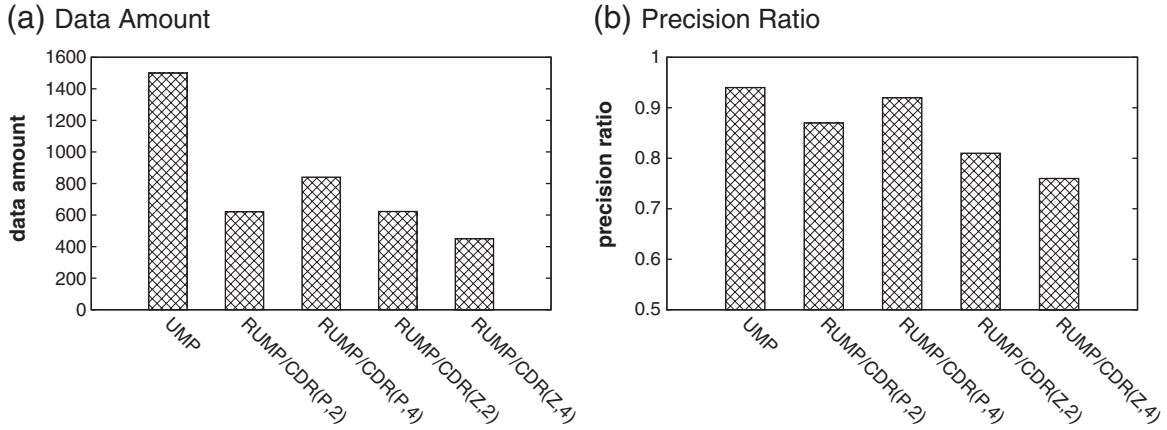

Fig. 8. Performance comparisons of *UMP* and *RUMP* on the *CarWeb* dataset.

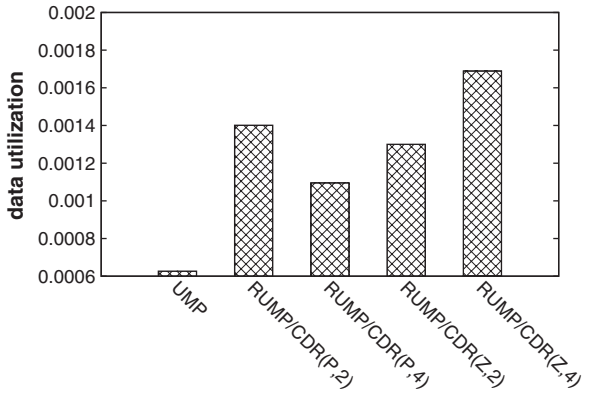

Fig. 9. Data utilization of *UMP* and *RUMP* on the *CarWeb* dataset.

higher *mf*, the data utilization of *UMP* drastically decreases. This is because the amount of data in the movement log increases dramatically as users move frequently. If the value of *mf* is smaller, the data utilization of *RUMP* with a Zeta distribution is larger than that of *RUMP* with a Poisson distribution. On the other hand, when the value of *mf* increases, the data utilization of *RUMP* with a Poisson distribution is larger than that of *RUMP* with a Zeta distribution. It is primarily because when *mf* is large, it is better to have more uniform calling behaviors to allow the call detail records fully reflect user movement behaviors. These experimental results show that *RUMP* has a higher data utilization than UMP. By exploring CDRs, *RUMP* is more cost-effective in mining user movement patterns.

Fig. 8(a) shows the data amount of *UMP* and *RUMP* with various calling behaviors under the *CarWeb* dataset. Fig. 8(a) shows that the data amount of *RUMP* is much smaller than that of *UMP*. Furthermore, Fig. 8(b) shows that the precision ratios of *UMP* and *RUMP*, indicating that the difference between *UMP* and *RUMP* is not large. This suggests that *RUMP* is able to achieve acceptable precision ratios when using a smaller amount of data. However, through performing better than *RUMP* in terms of the precision ratio, *UMP* incurs more amounts of data in the movement log. In Fig. 9, the data utilization of *UMP* is much smaller than that of *RUMP*, showing that with a smaller amount of log data, *RUMP* can still achieve an acceptable precision ratio.

Fig. 10 shows the execution time of *UMP* and *RUMP* under the synthetic dataset. Fig. 9 shows that the *RUMP* execution time is smaller than that of *UMP* in both the synthetic dataset and the *CarWeb* dataset. With a larger number of movement sequences, the *UMP* execution time significantly increases. With a higher *mf*, the execution time of *RUMP* becomes much slower than that of *UMP*. Further, *RUMP* has better scalability than *UMP*. In addition, Fig. 9 shows the execution time of *UMP* and *RUMP* on the *CarWeb* dataset. Similar to the results in the synthetic dataset, the *RUMP* execution time is much smaller than that of *UMP*. As the number of movement sequences increases, *UMP* takes longer to discover user movement patterns. On the other hand, the *RUMP* performance is determined by the data amount generated by calling. Since the data amount generated by calling is usually fewer than that by movements, *RUMP* incurs a smaller execution time.

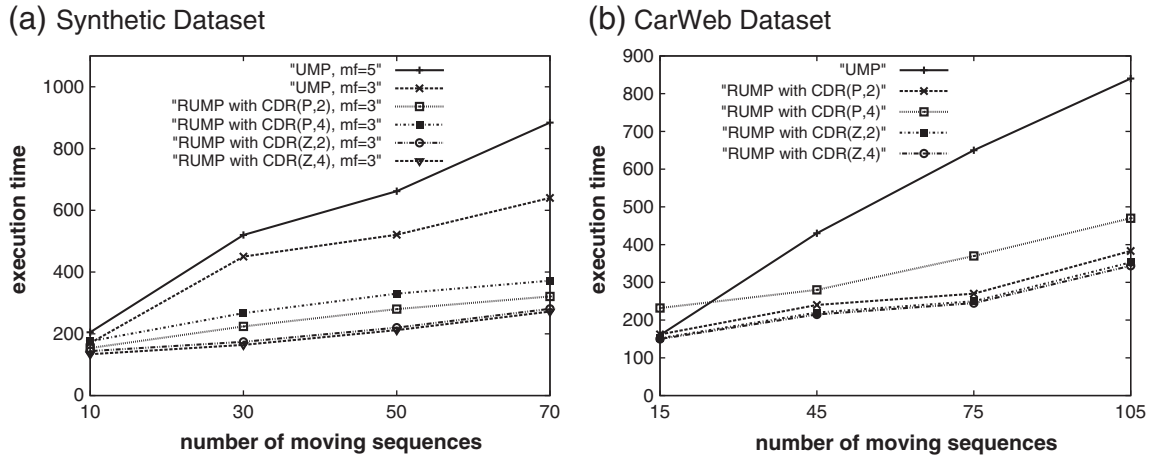

Fig. 10. Execution time for various numbers of movement sequences.

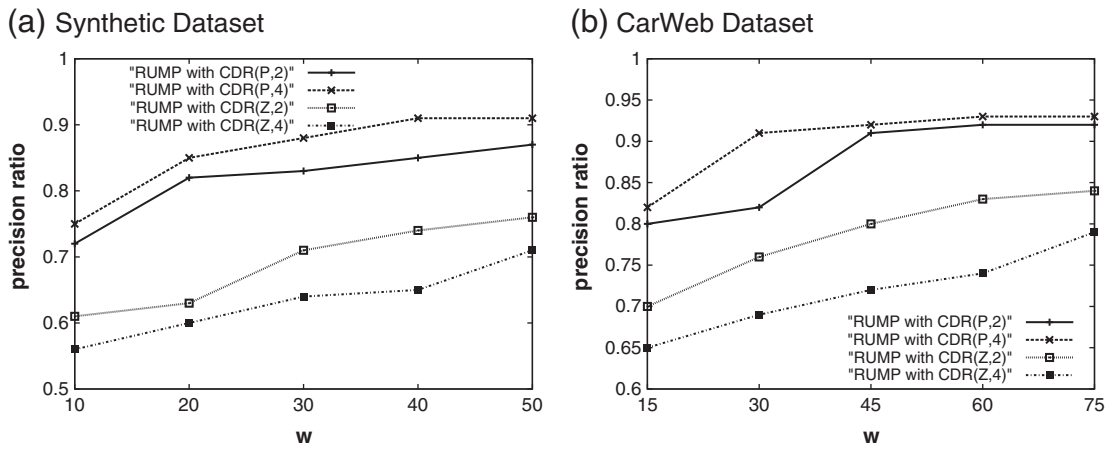

Fig. 11. Precision ratio of *RUMP* with varying *w*.

4.3. Sensitivity analysis of RUMP

This section further investigates the parameters used in *RUMP*. First, the impact of *w* is presented. Then, we examine the impact of thresholds on the mining results.

4.3.1. Impact of w

Fig. 11 shows the experiments of varying *w* values for *RUMP* under both the synthetic dataset and the *CarWeb* dataset. This figure indicates that the *RUMP* precision ratio increases as the value of *w* increases in both datasets. This is because as the value of *w* increases, the number of movement sequences considered in *RUMP* increases as the value of *w* increases. In this case, *RUMP* can use more calls to discover user movement patterns. The *RUMP* precision ratio with a Poisson distribution is larger than that of *RUMP* with a Zeta distribution. This is because the calling behavior in a Poisson distribution is much more evenly across user movements. Thus, *RUMP* is able to fully capture user movement behaviors when the calling behavior follows a Poisson distribution. In a Poisson distribution, with a larger value of λ, the precision ratio of *RUMP* is larger. For a larger value of λ, the amount of call detail records tends to increase, thereby reflecting the complete movement behaviors of users. For users with a larger number of calls and non-burst calling behavior, the value of *w* can be set smaller to quickly obtain movement patterns. In contrast, for users with a smaller number of calls or burst calling behavior, the value of *w* should be set larger to improve the precision ratio of the movement patterns mined by *RUMP*.

4.3.2. Impact of thresholds in algorithm LS

This section examines the impact of *min _freq* and *min _sim* on the *RUMP* performance. Algorithm LS uses *min _freq* and *min _sim* thresholds to extract CDRs representing frequent movement behaviors. Figs. 12 and 13 show *RUMP* experiments with

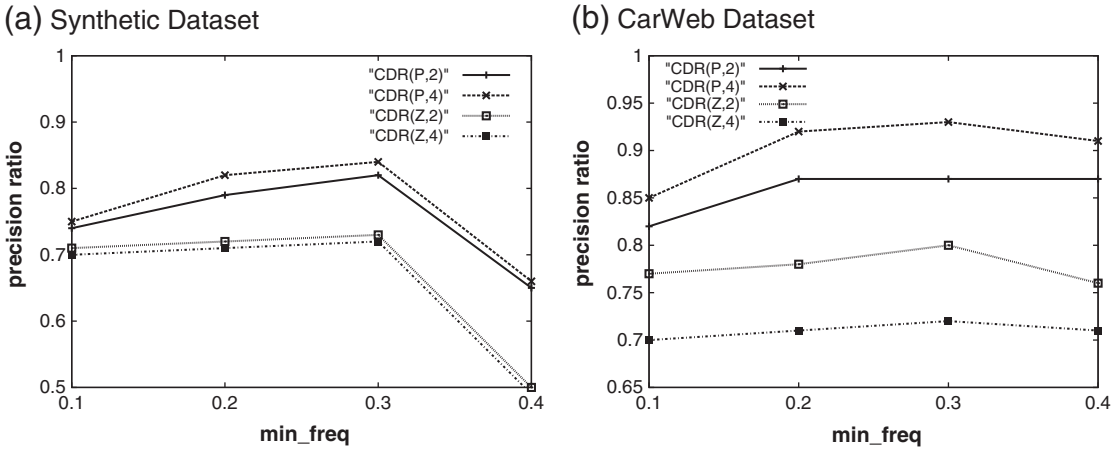

Fig. 12. Precision ratio of *RUMP* with *min_freq* varied.

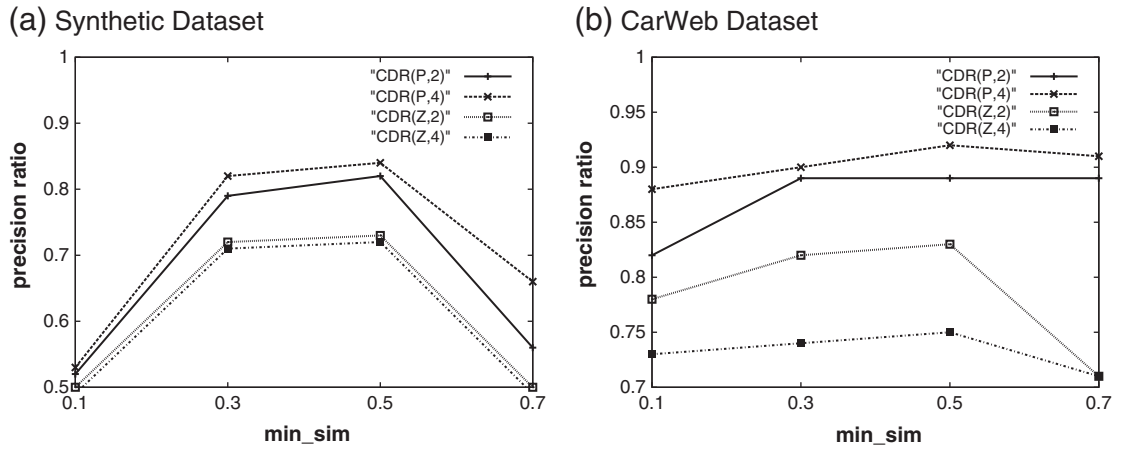

Fig. 13. Precision ratio of *RUMP* with *min_sim* varied.

varying values of *min _freq* and *min _sim*. Fig. 12(a) shows the result of using *RUMP* on the synthetic dataset. This figure indicates that the *RUMP* precision ratio tends to increase when the value of *min _freq* increases from 0.1 to 0.3. This figure also shows that the *RUMP* precision ratio decreases when *min _freq* exceeds than 0.3. This is because increasing *min _freq* filters out areas through which users do not frequently move are filtered out. However, a larger *min _freq* is too strict for identifying what areas are frequent and decreases the precision ratio. Fig. 12(b) shows that the same phenomenon for the *CarWeb* dataset. Selecting the value of *min _freq* should be determined empirically. For example, in this experiment, we set *min _freq* at 0.3.

Fig. 13 shows the *RUMP* precision ratio with various values of *min _sim*. In both datasets, the *RUMP* precision ratio tends to increase when *min _sim* increases from 0.1 to 0.5. However, when the value of *min _sim* exceeds than 0.5, the *RUMP* precision ratio decreases. The *min _sim* threshold is set to identify whether or not a movement sequence is similar to the frequent movement behavior. With a larger value of *min _sim*, only a few movement sequences are identified as being similar to frequent user movement behaviors. This, in this turn, decreases the *RUMP* precision ratio. Therefore, the value of *min _sim* should be carefully set. Experimental results shows that *min _freq* should be set to 0.3 and *min _sim* should be set to be 0.5 to achieve the best precision ratio performance.

4.3.3. Impact of thresholds in algorithm TC

As described above, the value of *min _var* for algorithm TC affects the accuracy of the *RUMP* time clustering results. We conducted experiments to examine the impact of *min _var*. For the synthetic dataset, Fig. 14(a) shows that the precision ratio of *RUMP* with the values of threshold *min _var* varied. This figure indicates the *RUMP* precision ratio significantly increases when *min _var* is 0.25. However, the precision ratio of *RUMP* decreases when *min _var* exceeds than 0.75. This is because excessively

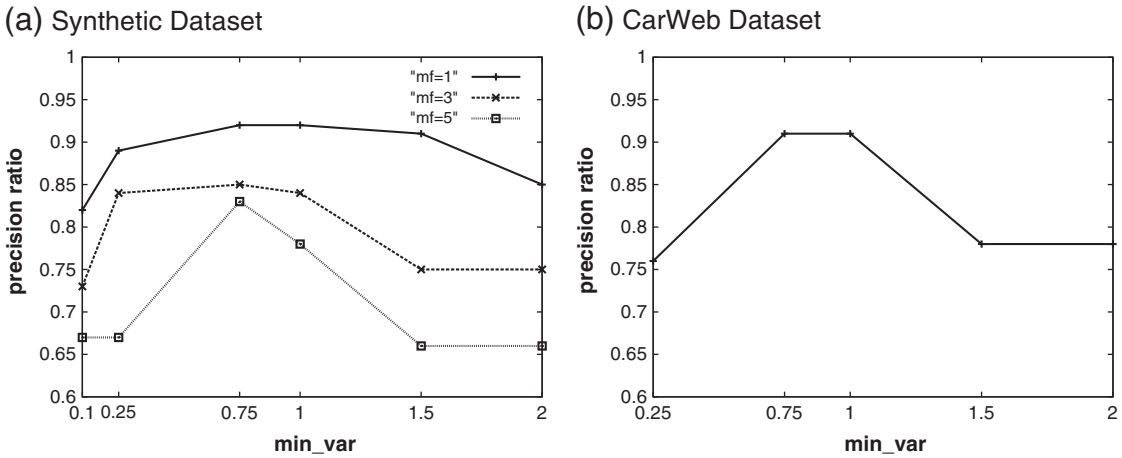

Fig. 14. Precision ratio with min_var varied.

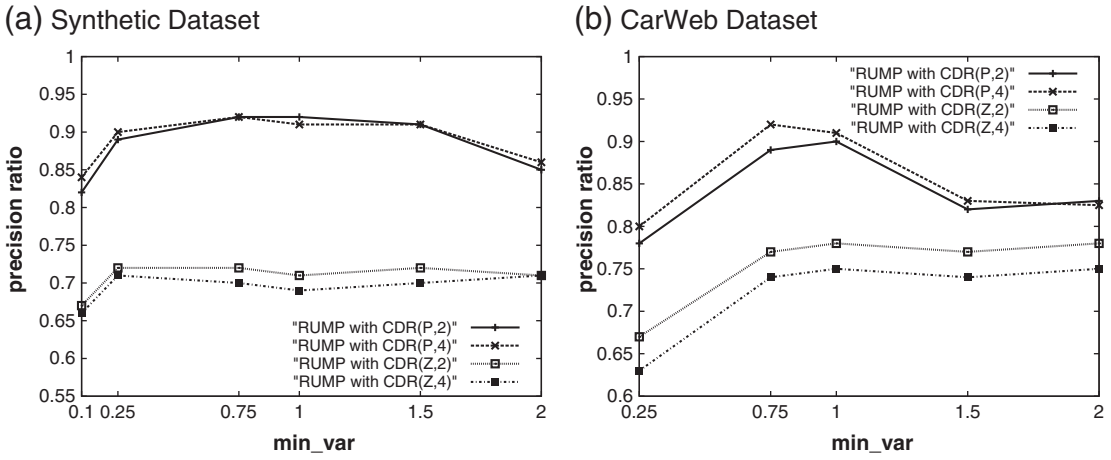

Fig. 15. Precision ratio of *RUMP* with min_sim varied.

large values of *min _var* result in most of the call detail records being grouped in the same cluster. Hence, the number of movement functions is not enough to capture user movement behaviors. Furthermore, with a larger *mf*, the *RUMP* precision ratio is smaller and significantly decreases when *min _var* is larger. For the *CarWeb* dataset, Fig. 14(b) shows the similar experimental results. These results indicate that *min _var* should be set to be a smaller value for users who move frequently. The value of *min _var*, which can be determined empirically, should not set too larger. For example, in Fig. 14(a), *min _var* should set to 0.75 because the *RUMP* precision ratio is the highest.

Fig. 15 depicts the *RUMP* precision ratio with various calling behaviors. In Fig. 15(a), the results of CDR(P,2) and CDR(P,4) are similar to the results above. However, it is interesting to note that the precision ratios of CDR(Z,2) and CDR(Z,4) do not decrease when the value of *min _var* exceeds than 0.75. Since burst calls happen in the beginning of every three time slots, most of the call detail records in these three time slots can be grouped into one cluster. Fig. 15 shows the similar results in the *CarWeb* dataset. Thus, we can set *min _var* as 0.75 to obtain the highest *RUMP* precision ratio.

5. Conclusions

User movement patterns can provide a lot of benefits in many mobile design schemes and applications, including designing a paging area, developing data allocation schemes, conducting querying strategies, or offering navigation services. This article proposes a regression-based approach called *RUMP* for mining user movement patterns from call detail records. To fully exploit the fragmented spatio-temporal information hidden in such trajectories, the proposed regression-based solution discovers user movement patterns. The *RUMP* approach uses three algorithms. First, algorithm LS extracts CDRs that reflect the frequent movement behaviors of mobile users. By capturing similar movement sequences from call detail records, an aggregation movement sequence is computed to represent the frequent movement behaviors of mobile users in each time slot. The feature of spatio-temporal locality states that if the time interval between consecutive calls is small, the mobile user is likely to have moved nearby. By exploring this feature, algorithm TC is able to determine the number of regression functions properly by clustering those movement records whose time of occurrence are very close from an aggregation movement sequence. For each cluster of the aggregation movement sequence, algorithm MF generates the movement functions representing user movement patterns of mobile users. This article evaluates the performance of the proposed algorithms and conducts sensitivity analysis on several design parameters. Experimental results indicate that *RUMP* can efficiently and effectively derive user movement patterns that capture the frequent movement behaviors of mobile users.

Appendix A. Proof of minimizing residual error sum

Following the same notation in Section 3.4, the residual sum of squares (i.e., $\epsilon_x = \sum_{i=1}^{n} w_i e_i^2$) can be expressed as $\epsilon_x = \mathbf{e}^T\mathbf{We}$ in a linear algebra manner. All elements in $\mathbf{W}$ are positive and $\mathbf{e}$ is able to be formulated as $\left(\tilde{\mathbf{b}}_x - \mathbf{Ha}^*\right)$. Thus, we have:

$$\epsilon_x = \mathbf{e}^T\mathbf{We} = \left(\tilde{\mathbf{b}}_x - \mathbf{Ha}^*\right)^T\mathbf{W}\left(\tilde{\mathbf{b}}_x - \mathbf{Ha}^*\right) = \left(\tilde{\mathbf{b}}_x - \mathbf{Ha}^*\right)^T\sqrt{\mathbf{W}}\sqrt{\mathbf{W}}\left(\tilde{\mathbf{b}}_x - \mathbf{Ha}^*\right) = \left(\sqrt{\mathbf{W}}\,\tilde{b}_x - \sqrt{\mathbf{W}}Ha^*\right)^T\left(\sqrt{\mathbf{W}}\,\tilde{\mathbf{b}}x - \sqrt{\mathbf{W}}\mathbf{Ha}^*\right)$$
$$= \left\|\sqrt{\mathbf{W}}\,\tilde{\mathbf{b}}_x - \sqrt{\mathbf{W}}\mathbf{Ha}^*\right\|$$

let $\mathbf{A} = \sqrt{\mathbf{W}}\mathbf{H}$ and $\mathbf{B} = \sqrt{\mathbf{W}}\,\tilde{\mathbf{b}}_x$. The main objective is to minimize $\epsilon_x = ||\mathbf{B}-\mathbf{A}\mathbf{a}^*||$. According to the least squares theorem, $\mathbf{B}-\mathbf{A}\mathbf{a}^*$ must be orthogonal to $\mathbf{A}\mathbf{a}^*$ so as to minimize ε_x. For the sake of brevity, the description of least squares theorem is omitted in this article. We can have:

$$\begin{aligned}
&\mathbf{B}-\mathbf{A}\mathbf{a}^* \in R(\mathbf{A})^{\perp} = N\left(\mathbf{A}^T\right),\\
&\quad \text{where } R(\mathbf{A})^{\perp} \text{ represents the orthogonal complement of column space of } \mathbf{A}\\
&\quad \text{and } N\left(\mathbf{A}^T\right) \text{represents the kernel space of } \mathbf{A}^T\\
\Rightarrow\ & \mathbf{A}^T(\mathbf{B}-\mathbf{A}\mathbf{a}^*) = 0\\
\Rightarrow\ & \mathbf{A}^T\mathbf{A}\mathbf{a}^* = \mathbf{A}^T\mathbf{B}
\end{aligned}$$

$\mathbf{A}^T\mathbf{A}\mathbf{a}^* = \mathbf{A}^T\mathbf{B}$ is viewed as the normal equation. By substituting $\mathbf{A} = \sqrt{\mathbf{W}}\mathbf{H}$ and $\mathbf{B} = \sqrt{\mathbf{W}}\,\tilde{\mathbf{b}}_x$, we have $\left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\left(\sqrt{\mathbf{W}}\mathbf{H}\right)\mathbf{a}^* = \left(\sqrt{\mathbf{W}}\mathbf{H}\right)^T\sqrt{\mathbf{W}}\,\tilde{\mathbf{b}}_x$. By solving the normal equation, a can be derived to minimize the value of ϵ_x. □

References

[1] S. Spaccapietra, C. Parent, M.L. Damiani, J.A.F. Macêdo, F. Porto, C. Vangenot, A conceptual view on trajectories, Data and Knowledge Engineering 65 (1) (2008) 126–146.
[2] C.-C. Hung, W.-C. Peng, Clustering object moving patterns for prediction-based object tracking sensor networks, Proc. of ACM Conference on Information and Knowledge Management (CIKM), 2009.
[3] H.-K. Wu, M.-H. Jin, J.-T. Horng, Personal paging area design based on mobiles moving behaviors, Proc. of IEEE Conference on Computer Communications (INFOCOM), 2001, pp. 21–30.
[4] W.-C. Peng, M.-S. Chen, Developing data allocation schemes by incremental mining of user moving patterns in a mobile computing system, IEEE Transactions on Knowledge and Data Engineering 15 (6) (2003).
[5] W.-C. Peng, M.-S. Chen, Shared data allocation in a mobile computing system: exploring local and global optimization, IEEE Transactions on Parallel and Distributed Systems 16 (4) (2005) 374–384.
[6] S.-M. Tseng, C.-F. Tsui, An efficient method for mining associated service patterns in mobile web environments, Proc. of ACM Symposium on Applied Computing (SAC), 2003, pp. 455–459.
[7] M.-H. Jin, J.-T. Horng, M.-F. Tsai, E.H.-K. Wu, Location query based on moving behaviors, Information Systems 32 (3) (2007) 385–401.
[8] H. Gonzalez, J. Han, X. Li, M. Myslinska, J.P. Sondag, Adaptive fastest path computation on a road network: a traffic mining approach, Proc. of International Conference on Very Large Data Bases (VLDB), 2007, pp. 794–805.
[9] E. Kanoulas, Y. Du, T. Xia, D. Zhang, Finding fastest paths on a road network with speed patterns, Proc. of IEEE International Conference on Data Engineering (ICDE), 2006, pp. 10–19.
[10] Y.-B. Lin, A.-C. Pang, Wireless and Mobile All-IP Core Networks, John Wiley, 2005.
[11] D.-L. Lee, J. Xu, B. Zheng, W.-C. Lee, Data management in location-dependent information services: challenges and issues, IEEE Pervasive Computing 1 (3) (2002) 65–72.
[12] A. Bhattacharya, S.K. Das, Lezi-update: an information-theoretic framework for personal mobility tracking in PCS networks, Wireless Networks 8 (2–3) (2002) 121–135.
[13] S. Ma, S. Tang, D. Yang, T. Wang, C. Yang, Incremental maintenance of discovered mobile user maximal moving sequential patterns, Proc. of International Conference on Database Systems for Advanced Applications (DASFAA), 2004, pp. 824–830.
[14] A.J.T. Lee, H.-W. Wu, T.-Y. Lee, Y.-H. Liu, K.-T. Chen, Mining closed patterns in multi-sequence time-series databases, Data and Knowledge Engineering 68 (10) (2009) 1071–1090.
[15] Y. Wang, E.-P. Lim, S.-Y. Hwang, Efficient mining of group patterns from user movement data, Data and Knowledge Engineering 57 (3) (2006) 240–282.
[16] H. Jeung, H.-T. Shen, X. Zhou, Mining trajectory patterns using hidden Markov models, Proc. of International Conference on Data Warehousing and Knowledge Discovery (DaWaK), 2007, pp. 470–480.
[17] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, D.W. Cheung, Mining, indexing, and querying historical spatiotemporal data, Proc. of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2004, pp. 236–245.
[18] F. Verhein, K-stars: sequences of spatio-temporal association rules, Proc. of ICDM Workshop on Spatial and Spatio-temporal Data Mining, 2006, pp. 387–394.
[19] S. Tseng, C. Tsui, Mining multilevel and location-aware service patterns in mobile web environments, IEEE Transactions on Systems, Man, and Cybernetics, Part B 34 (6) (2004) 2480–2485.
[20] H. Cao, N. Mamoulis, D.W. Cheung, Mining frequent spatiotemporal sequential patterns, Proc. of IEEE International Conference on Data Mining (ICDM), 2005, pp. 82–89.
[21] P. Kalnis, N. Mamoulis, S. Bakiras, On discovering moving clusters in spatiotemporal data, Proc. of International Symposium on Spatial and Temporal Databases (SSTD), 2005, pp. 364–381.
[22] H. Jeung, Q. Liu, H.-T. Shen, X. Zhou, A hybrid prediction model for moving objects, Proc. of IEEE International Conference on Data Engineering (ICDE), 2008.
[23] F. Giannotti, M. Nanni, F. Pinelli, D. Pedreschi, Trajectory pattern mining, Proc. of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2007, pp. 330–339.
[24] F. Giannotti, M. Nanni, D. Pedreschi, Efficient mining of temporally annotated sequences, Proc. of SIAM Conference on Data Mining (SDM), 2006.
[25] C.-C. Hung, W.-C. Peng, J.-L. Huang, Exploring regression for mining user moving patterns in a mobile computing system, Proc. of International Conference on High Performance Computing and Communications (HPCC), 2005, pp. 878–887.
[26] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, 2005, ISBN 0-321-32136-7.
[27] Paul Jaccard, Etude comparative de la distribution florale dans une portion des Alpes et des Jura, Bulletin del la Societe Vaudoise des Sciences Naturelles 37 (1901) 547–579.
[28] R.V. Hogg, E.A. Tanis, Probability and statistical inference, Prentice-Hall International Inc., 1997
[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, Second Edition. The MIT Press and McGraw-Hill Book Company, 2001.
[30] Y.-B. Lin, Modeling techniques for large-scale PCS networks, IEEE Communications Magazine 35 (2) (1997) 102–107.
[31] Y. Xu, W.-C. Lee, TTC: delay-tolerant trajectory compression for object tracking sensor networks, Proc. of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2006, pp. 436–445.
[32] C.-H. Lo, W.-C. Peng, C.-W. Chen, T.-Y. Lin, C.-S. Lin, CarWeb: a traffic data collection platform, Proc. of International Conference on Mobile Data Management (MDM), 2008, [available]http://carweb.cs.nctu.edu.tw/carweb/.

Wen-Chih Peng was born in Hsinchu, Taiwan, R.O.C in 1973. He received the BS and MS degrees from the National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in Electrical Engineering from the National Taiwan University, Taiwan, R.O.C in 2001. Currently, he is an associate professor at the department of Computer Science, National Chiao Tung University, Taiwan. Prior to joining the department of Computer Science and Information Engineering, National Chiao Tung University, he was mainly involved in the projects related to mobile computing, data broadcasting and network data management. Dr. Peng serves as PC members in several prestigious conferences, such as IEEE International Conference on Data Engineering (ICDE), Pacific Asia Knowledge Discovering and Mining (PAKDD) and Mobile Data Management (MDM). His research interests include mobile computing, network data management and data mining. He is a member of IEEE.

Chih-Chieh Hung was born in Taipei, Taiwan in 1977. He received the MS degrees from National Chiao Tung University, Taiwan in 2005, respectively. Currently, he is a Ph.D. candidate in the department of Computer Science, National Chiao Tung University, Taiwan. During his Ph.D. program, he was mainly involved in the projects related to mobile computing, wireless sensor data management, and location-based services. He received the best paper award in ACM workshop on Location-Based Social Network in 2009. He was also selected as a full-time intern of Microsoft Research Asia in 2010. His research interests include trajectory pattern mining and applications, location-based social network, and mobile data management on sensor networks.