Exploring Sequential Probability Tree for Movement-Based Community Discovery

Wen-Yuan Zhu, Wen-Chih Peng, *Member, IEEE*, Chih-Chieh Hung, Po-Ruey Lei, and Ling-Jyh Chen, *Senior Member, IEEE*

Abstract—In this paper, we tackle the problem of discovering movement-based communities of users, where users in the same community have similar movement behaviors. Note that the identification of movement-based communities is beneficial to location-based services and trajectory recommendation services. Specifically, we propose a framework to mine movement-based communities which consists of three phases: 1) constructing trajectory profiles of users, 2) deriving similarity between trajectory profiles, and 3) discovering movement-based communities. In the first phase, we design a data structure, called the Sequential Probability tree (SP-tree), as a user trajectory profile. SP-trees not only derive sequential patterns, but also indicate transition probabilities of movements. Moreover, we propose two algorithms: BF (standing for breadth-first) and DF (standing for depth-first) to construct SP-tree structures as user profiles. To measure the similarity values among users' trajectory profiles, we further develop a similarity function that takes SP-tree information into account. In light of the similarity values derived, we formulate an objective function to evaluate the quality of communities. According to the objective function derived, we propose a greedy algorithm Geo-Cluster to effectively derive communities. To evaluate our proposed algorithms, we have conducted comprehensive experiments on two real data sets. The experimental results show that our proposed framework can effectively discover movement-based user communities.

Index Terms—Trajectory profile, community structure, and trajectory pattern mining

1 Introduction

With the development of positioning techniques, many portable devices (e.g., smart phones) are equipped with location-aware sensors, meaning that the positions of users can be easily obtained. The sequences of these collected positions can be viewed as users' trajectories. Since trajectories represent the actual movements in users' real lives [33], many trajectory-sharing sites have been established, such as EveryTrail [1], Run.gps [2], and Running free [3].

Prior works [21], [30], [32] have elaborated on discovering user communities from user location history. Such a community is thus referred to as a movement-based community. In light of movement-based communities, many novel location-based services and application systems can be developed [32], [33]. Some application examples are listed below:

- *Trajectory ranking*: Most websites provide a query interface, depicting some query predicative requirements, to retrieve user trajectories. For example, in EveryTrail [1], one can issue a keyword (e.g., Taipei) to retrieve the trajectories with their information related to the keyword specified. Usually, the query may result in a mass of incoherent trajectories. If the ranking policy could take movement-based community structures into consideration, those trajectories shared by the same movement-based community are likely to provide more mobility information and specific recommendation to users.
- *Community-based traffic sharing services*: Recently, traffic sharing services have become very popular. For example, users can download the mobile application, Waze [4], to share traffic information and use the navigation service. Such a platform provides traffic information and navigation services collected from crowd power. However, one could easily obtain traffic information from movement-based community members who have similar trajectories. Moreover, better quality traffic information could be retrieved since traffic data shared by community members. The traffic information aggregated from movement community members will be intact and comprehensive.
- *Friend recommendation*: With the movement-based community, one could recommend possible users who are likely to have the same or similar movement preferences. For example, in some applications, users may record their own running paths for exercise purposes. By analyzing their running behavior from the trajectories shared by users, one could design more social favor services. For example, by recommending some users who have similar running behaviors, a group-based running activity could be easily initiated.

To discover movement-based communities, one should first formulate the similarity of users in terms of their trajectories. Clearly, since trajectories are time series data, one

- *W.-Y. Zhu and W.-C. Peng are with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan. E-mail: {wyzhu, wcpeng}@cs.nctu.edu.tw.*
- *C.-C. Hung is with the Rakuten Inc., Japan. E-mail: smalloshin@gmail.com.*
- *P.-R. Lei is with the Department of Electrical Engineering, ROC Naval Academy, Taiwan. E-mail: kdboy1225@gmail.com.*
- *L.-J. Chen is with the Institute of Information Science, Academia Sinica, Taipei, Taiwan. E-mail: cclljj@iis.sinica.edu.tw.*

Manuscript received 14 Sept. 2012; revised 13 Nov. 2013; accepted 17 Jan. 2014. Date of publication 3 Feb. 2014; date of current version 26 Sept. 2014. Recommended for acceptance by H. Zha.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2014.2304458

naive idea is to explore existing similarity functions, such as Euclidean distance, DTW, EDR, ERP and LCSS [9], to measure the similarity degree of users. However, since trajectories contain uncertainty and raw noise data, it is hard to accurately model the movement similarity of users via using existing similarity functions from user raw trajectories. Moreover, the computing cost is expensive if one would like to derive the similarity value among users from their raw trajectory data. To bridge the uncertainty and noise in trajectory data, without loss of generality, most prior works [15], [21] extract regions, where one user frequently visits. Those regions are likely to represent hot regions or popular regions from trajectories. Then, prior works [15], [21] explore the sequential relationships among the extracted regions. These sequential regions are able to be viewed as the determinable features to describe movement behavior. As in [11], [24], we utilize a grid-based approach to extract hot regions. As such, each trajectory could be represented by a sequence of hot regions (referred to as a transformed trajectory).

To capture movement behavior from a set of transformed trajectories, one could use sequential pattern mining methods to extract frequent sequential patterns. The frequent sequential patterns are the subsequences of the transformed trajectories if the number of transformed trajectories containing the subsequences is larger than a pre-defined threshold (i.e., the minimum support). Note that the number of frequent sequential patterns may result in the need for extra storage. To compare the movement behavior between users, such a set of large size sequential patterns may cause high complexity for deriving the similarity among them. Furthermore, the conventional sequential patterns are derived by the frequency of the sequences which occur in the trajectories and do not have transition probabilities among hot regions. Indeed, the same movement sequence may have different transition probabilities caused by different users. The transition probabilities among hot regions can be utilized as an identical feature for movement behavior comparison. Therefore, to compare the movement behaviors effectively, considering both the movement sequences and probability of mobility transition is necessary. Thus, in this paper, we first propose the concept of trajectory profiles to characterize users' movement behaviors, where the trajectory profile is a tree structure with frequent movement sequences and transition probabilities. Then, based on the free-based trajectory profile, we propose a new similarity function to model user similarity in terms of their trajectory profiles. Furthermore, we formulate the objective function of clustering and propose one algorithm to cluster users into groups. Note that each group represents one movement-based community. In our proposed framework, several technical issues are addressed:

1) *Constructing user trajectory profiles*: A trajectory profile should capture two kinds of movement information: a) the sequential patterns of hot regions, and b) the transition probabilities among hot regions. The sequential patterns among hot regions describe the ways in which trajectories frequently transit among these regions. Such patterns are well-studied and have many variations [6], [11]. In this paper, we propose a compact data structure, called Sequential Probability Tree (SP-tree), to mine both sequential patterns and transition probabilities. Explicitly, in an SP-tree, each node is labeled by tree edges traversed from the root node of the SP-tree, and labels represent sequential patterns. For each node, a conditional table that depicts the next movement from a tree node, is constructed. Instead of capturing all transition probabilities among hot regions, the conditional table of nodes only reflects the next movement probabilities. By traversing SP-trees, one could derive transition probabilities of hot regions. Furthermore, the proposed SP-tree is a data structure to compress the sequential patterns into a compact tree model, which indicates the sequential patterns and the transition probabilities.

2) *Deriving similarity between trajectory profiles*: To distinguish how close two trajectory profiles are, we need to develop a similarity function between two SP-trees. Since trajectory profiles are tree structures, we formulate some similarity functions, including two existing common similarity functions and one new similarity function that explores all of the information of the tree structures. Four details of the tree structures, such as the number of common tree nodes, the number of different tree nodes, the support values of the tree nodes and the conditional tables of the tree nodes, are considered in the proposed SP-tree based similarity function.

3) *Discovering movement-based communities*: Given a set of users with their similarity scores, a *geo-connection graph* is built for discovering movement-based communities. Each node of the geo-connection graph represents each user, and the weight edge depicts the similarity score between two users. A *geo-connection* can be constructed among users if their similarity scores are larger than a pre-defined threshold that could be viewed as a minimum similarity bound in movement-based communities. Given a geo-connection graph, our goal is to cluster vertices into groups, where each group represents one community. Thus, we formulate an objective function to measure the quality of the cluster results. In light of the objective function, we propose algorithm Geo-Cluster to cluster users in the geo-connection graph. In addition, since the number of user communities is usually not known in advance, the number of groups should not be assigned when partitioning users into communities. In this paper, we reveal a way to mine user communities without specifying the number of communities.

In summary, our major contributions are outlined as follows:

1. We propose the concept of trajectory profile as a tree structure SP-tree to characterize one user's movement behavior from trajectories, in terms of sequential patterns and transition probability to the next movement.
2. Based on the trajectory profiles, we design a similarity function that takes both tree nodes and conditional tables into account.
3. We formulate an objective function to evaluate the good quality cluster results and propose algorithm Geo-Cluster to effectively derive user movement-based communities.
4. We have conducted comprehensive experiments on real data sets, and the experimental results show that the proposed trajectory profile is able to accurately reflect user movement behavior and effectively discover user communities.

The remainder of this paper is organized as follows: Section 2 discusses several related works. Section 3 outlines the preliminaries. Section 4 presents the algorithms to construct trajectory profiles. Section 5 describes our proposed clustering algorithms to discover movement-based communities. The experimental results are shown in Section 6. Section 7 concludes this paper.

2 Related Works

In this paper, the goal is to discover user communities based on user movement behavior hidden in user trajectories. Our work is related to trajectory pattern mining in that we intend to mine user movement behavior from trajectories. Thus, we will present some existing works on trajectory pattern mining. Furthermore, we will describe the existing works on mining user community structures based on user trajectories.

2.1 Trajectory Pattern Mining

A considerable amount of research effort has already been put into mining movement behavior from trajectories. Given a set of trajectories, trajectory pattern mining aims to discover frequent sequential patterns that are sequential relationships among regions. Generally speaking, the flow of mining trajectory patterns is to find hot regions, and then discover sequential relationships among them. We mention in passing that the authors in [6] explored the fuzziness of locations in patterns and developed algorithms to discover spatial-temporal sequential patterns. The authors in [17] proposed a clustering-based approach to discover moving regions within time intervals. In [11] and [10], the authors exploited temporal annotated sequences in which the sequences are associated with time information (i.e., transition times between two movements). Furthermore, according to the size of the patterns, some previous works utilize data structures to organize them. In [15], the authors developed a hybrid prediction model, consisting of vector-based and pattern-based models, to predict the location of users at the specific time given. As for the pattern-based model, the authors utilized association rules to capture movement behavior, and the association rules are indexed by the *trajectory pattern tree*, which is a variant of the signature tree, for efficient retrieval of the association rules. In [24], the authors developed a decision tree, named the *T-pattern tree*, to organize trajectory patterns for predicting the next movement of objects. Note that prior works on trajectory patterns aimed to discover the movement behavior of users. In this paper, we intend not only to discover user movement behavior but also to utilize movement behavior to discover movement-based communities. Due to huge number of trajectories, the number of trajectory patterns is larger. In this paper, we propose a compact data structure to represent movement behavior in terms of sequential relationships and transition probabilities hidden in trajectories. Furthermore, we cluster users into different communities according to user trajectories. The work in [16] is to cluster trajectories with a similar shape into spatial and temporal domains, which targets clustering trajectories and not users as addressed in our proposed work.

2.2 User Movement-Based Community Mining

In [21], [32], the authors aim to find user similarity based on their trajectories. However, the proposed work in [21], [32] uses the sequential patterns to format user similarity. They extract similar sequences in two users' trajectories, and then calculate the similarity of each. There is a higher similarity score if there are many similar location sequences and similar time durations between two locations among their trajectories. After deriving the similarity of each similar sequence, the similarity between two users is the sum of all the similarities of each similar sequence in their trajectories. However, this work only uses similar sequences to represent users' movement behavior, and then it pays a large computation cost to find similar sequences between two users' trajectories. In [28], the authors aim to reduce the transmission cost in sensor networks. They propose a clustering method to aggregate objects with similar moving traces. They use PST to represent the movement profile of moving objects. To cluster the moving objects based on the PST, they calculate the difference of probability of all possible path in two PSTs, and then normalize and convert the value into similarity. Based on the similarity measurement, a distributed clustering method is proposed to cluster similar moving objects. However, they pay a large computation cost when selecting all possible paths to calculate the difference of probability from two PSTs for deriving the similarity of two moving objects. Therefore, these two works are not suitable for our problem. Furthermore, there are some works related to trajectory clustering. Given a set of trajectories, the goal is to cluster trajectories into groups, where trajectories in the same group have a certain degree of similarity. The state-of-the-art approach is TraClus [19]. The primary goal of this work is to discover common sub-trajectories from a set of trajectories. A key observation in this work is that clustering based on the whole trajectory could miss some common sub-trajectories, which are very useful in many applications, especially when regions of special interest are considered for analysis. Therefore, a partition-and-group framework for clustering sub-trajectories is proposed. This framework first decomposes a trajectory into a set of line segments, and then groups similar line segments together into a cluster. However, in this paper, the goal is to cluster users into groups according to their own set of trajectories. Therefore, the existing works of trajectory clustering cannot be directly applied to model user similarity.

3 Preliminaries

Without loss of generality, each trajectory is typically represented as a sequence $\langle \ell_1, \ell_2, \ldots, \ell_n \rangle$, where ℓ_i denotes the locations, $1 \leq i \leq n$. Since raw data points of trajectories may contain noise data, they may not accurately capture the movement behavior of users. Furthermore, calculating movement similarity from user raw trajectories incurs a higher computation cost because of a larger number of data points in the trajectories. Thus, instead of using data points of trajectories, we transform raw trajectories into sequences of hot regions, which refer to the regions that trajectories frequently pass through. We mention in passing that some prior works on trajectory pattern mining in [11], [15], [23], [24] have proposed several methods to determine hot regions, which could be viewed as trajectory features for trajectory pattern mining. The proposed methods of hot region determination are categorized into grid-based approaches [11], [24] and density-based approaches [15], [23]. Note that if the density-based approach is utilized for the generation of hot regions, the size of hot regions may vary. In such a case, hot regions may not capture fine user movement behavior. For example, in an urban area, many roads intersect with each other. If hot regions are merged into larger ones, they can easily form a cross shape that does not represent the movements of the trajectories precisely. Clearly, when many cross-shaped hot regions are found, it is not possible to infer user movement routes. In this paper, we adopt a grid-based approach [11], [24] to extract hot regions.

As such, each raw user trajectory is represented as a sequence of hot regions. To facilitate the presentation of this paper, a trajectory refers to a sequence of hot regions. Given a set of user trajectories, we intend to develop a trajectory profile to capture user movement behavior. Prior works [15], [21], [25] explore sequential patterns, which refer to frequent sequences of hot regions, as a user trajectory profile. Note that a huge number of sequential patterns may be generated, and transition probabilities of hot regions hidden in user trajectories cannot be captured from a set of sequential patterns. Therefore, we propose a compact tree structure, called sequential probability tree (abbreviated as SP-tree), to represent a user trajectory profile. Let $\mathcal{T}$ be the set of user trajectories, and Σ denotes the set of hot regions. A trajectory with length k in $\mathcal{T}$ is denoted by a string $\sigma_1\sigma_2\cdots\sigma_k$ for all σ_i in Σ. The formal definition of SP-tree is below.

Definition 1 (SP-tree). *An SP-tree SPT is a prefix tree which contains one root node root and a set of tree nodes, denoted as $\mathcal{S}$. Each tree edge indicates one hot region and each tree node s in an SP-tree is labeled by a string, represented by $\sigma_1\sigma_2\dots\sigma_k$, trace from root to node s, where k is the length from the root node to tree node s and $\sigma_i \in \Sigma$. Note that tree node s has the proportion of its appearing count, denoted as $support(s) \in [0,1]$ among a set of trajectories. The support value $support(i)$ of a normal node s is defined as $support(s) = |\{T_i|s$ which is a subsequence of $T_i, T_i \in \mathcal{T}\}|/|\mathcal{T}|$. Furthermore, tree node s contains a conditional table, in which each entry in the conditional table is a two-tuple (RID, C.Prob). The conditional table of tree node s captures the transition probability of user next movement under a given traversal sequence from the root to node s. In other words, if one user has the travel sequence s, this user has C.Prob probability of going to $RID \in R$.*

Given a set of trajectories of user U_1 in Fig. 1a, Fig. 1b is the corresponding SP-tree for user U_1. As can be seen in Fig. 1b, the SP-tree contains a root node and a set of tree nodes labeled as $\{A, B, C, AB, AC, BC, ABC\}$. As can be seen in Fig. 1b, the SP-tree contains a root node and a set of tree nodes labeled as $\{A, B, C, AB, AC, BC, ABC\}$. The nodes along with the leftmost branch is $root \rightarrow A \rightarrow AB \rightarrow ABC$, which represents the frequent sequential patterns (i.e., A, AB, and ABC). If the support threshold is set to 2, one could verify that A, AB, and ABC are indeed the frequent subsequences with their supports equal to 2. Moreover, the conditional table associated with each node represents the probabilities of the next possible movements of the sequential pattern of this node. For example, the conditional table of node A contains two entries: $(B, 0.67)$ and $(C, 0.33)$. This means that the user goes from the hot region A to the hot region B with a probability of 0.67 and to the hot region C with a probability of 0.33.

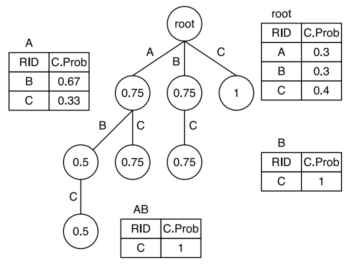
$T_1 : \langle A, B, C\rangle$

$T_2 : \langle A, B, C\rangle$

$T_3 : \langle B, C\rangle$

$T_4 : \langle A, C\rangle$

(a) User U_1's trajectories



(b) SPT_1

Fig. 1. User 1's trajectories and SP-tree SPT_1.

Note that SP-tree is able to not only capture sequential movement patterns but also indicate transition probabilities among hidden hot regions. Furthermore, SP-Tree is a compact structure in that nodes in the upper levels are shared with nodes in the lower levels. Usually, sequential patterns may have the same or similar prefix sequences. These prefix sequences will be represented as nodes in the upper levels of SP-trees, which is in fact a compact data structure. Since SP-tree should capture frequent user movement behavior, two minimal threshold values: $MinSup$ and $MinProb$ are used to guarantee that the support of each node should be larger or equal to $MinSup$, and the corresponding probability of the appearing count for each tree node should be larger than $MinProb$.

4 Algorithms for Constructing SP-Trees

Given a set of one user trajectory and two thresholds (i.e., $MinSup$ and $MinProb$), in this section, we propose two algorithms to construct an SP-tree as a user trajectory profile. The first algorithm explores a breadth first idea, called algorithm BF, to derive SP-trees in a level by level manner. On the other hand, the other algorithm, named DF, exploits the depth-first concept in deriving SP-trees. In addition, both time and space complexity analysis of algorithms BF and DF are presented.

4.1 Algorithm BF: A Breadth-First Approach

Algorithm BF is proposed to construct SP-trees in a level-by-level manner. Initially, SP-tree has only one root node with the conditional table which has the probabilities of hot regions being larger than a minimum probability threshold $MinProb$. When the support value of a hot region is larger than the minimal support threshold $MinSup$, a node is created in the second level. In each level, find frequent hot regions (line 7) and construct the conditional table of each node (line 8). Then, determine child nodes of each node (line 9 to line 13).

Fig. 2 shows the process of constructing the SP-tree of user U_1, where the settings are $MinSup = 0.4$ and $MinProb = 0.3$. In the beginning, the root node represents the empty sequential pattern. Thus, $S_0 = \{root\}$. Then, calculating the frequent locations and conditional table of each node in S_0, there is only one node, $root$, in S_0. The frequent hot regions of the projected trajectory set of $root$ are A, B and C since their support value are $3/4$, $3/4$ and $4/4$, respectively. Particularly, the projected trajectory set of $root$ is $\mathcal{T}$. As for the conditional table of $root$, there are three hot regions (i.e., A, B, and C) in U_1's trajectories. As can be seen in Fig. 2, the number of regions in U_1's trajectories is 10. Among these trajectories, A appears three times, B appears three times, and C appears four times. Thus, the $C.Prob$ of A, B, and C in the conditional table of $root$ are $3/10$, $3/10$, and $4/10$, respectively. Because the $C.Prob$ of all entries in the conditional table of $root$ are larger than $MinProb$, the conditional table of $root$ contains three entries (Fig. 2a). Then, we could derive the child nodes of $root$. For each frequent hot region, we inspect whether it is in the conditional table of $root$ or not. Since all frequent hot regions are in the conditional table of $root$, $root$ has three child nodes, and $S_1 = \{A, B, C\}$ (Fig. 2b). From S_1, we could further have S_2. As such, the frequent hot regions of node A are B and C, where the projected trajectory data set is $\{\langle B, C\rangle, \langle B, C\rangle, \langle C\rangle\}$. There are two entries, B and C, in the conditional table of node A. Thus, $S_2 = \{AB, AC\}$. After the same procedure for nodes B and C, $S_2 = \{AB, AC, BC\}$ (Fig. 2c). After finding frequent hot regions and creating the conditional table, we create child nodes of the nodes in S_1 (Fig. 2d). Following the above procedure, the final SP-tree is shown in Fig. 1b.

Algorithm 1 BF

Input: A set of transformed trajectories $\mathcal{T}$, a minimum probability threshold $MinProb$, a minimum support threshold $MinSup$

Output: An SP-tree SPT

```
1:  $SPT$ = a root node with empty entry;
2:  $S_0 = \{root\}$;
3:  $k = 0$;
4:  while $S_k \neq \emptyset$ do
5:      $S_{k+1} = \emptyset$
6:      for each node $s$ in $S_k$ do
7:          find frequent hot regions
8:          create conditional table of node $s$
9:          for each $\sigma$ in frequent hot regions do
10:             if $\sigma$ in conditional table of node $s$ then
11:                 create project transformed trajectory set of $s\sigma$
12:                 $s\sigma$ is a child of $s$
13:                 add node $s\sigma$ into $S_{k+1}$
14:             end if
15:         end for
16:     end for
17:     $k = k + 1$;
18: end while
```

Complexity analysis. As for the time complexity analysis, the algorithm runs from line 7 to line 15 $|S_k|$ times in the kth level. The time cost of line 7 to line 8 is $|\mathcal{T}|w$, where w is the average number of hot regions in the trajectories. Moreover, the time cost from line 10 to line 14 is $O(\mathcal{T}) + O(1) + O(1)$, and for each loop runs $O(|\Sigma|)$ times. Thus the time cost from line 7 to line 15 is $O(|\mathcal{T}|w + |\Sigma||\mathcal{T}|)$. Hence the time complexity of algorithm BF is $O(\sum_k |S_k|(|\mathcal{T}|w + |\Sigma||\mathcal{T}|))$. As for the space complexity analysis, the algorithm only needs $O((|S_k| + |S_{k+1}|)|\mathcal{T}|w)$ to store the projected trajectory set of nodes in the kth level. Thus, the space complexity of algorithm BF is $O(2 \cdot \max_k|S_k||\mathcal{T}|w) = O(\max_k|S_k||\mathcal{T}|w)$. Note that S_k will be affected by the two threshold values ($MinSup$ and $MinProb$). If $MinSup$ and $MinProb$ are smaller, the length of S_k becomes larger, and vice versa.

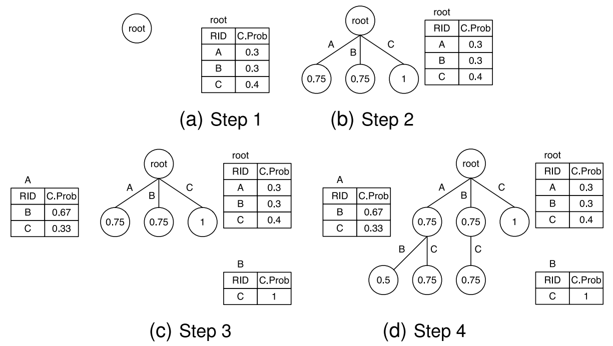


Fig. 2. An example of constructing SP-tree of user U_1 using algorithm BF.

4.2 Algorithm DF: A Depth-First Approach

Note that algorithm BF constructs the SP-tree level by level. To create child nodes for the next level, algorithm BF needs to have a buffer S_k to maintain processing nodes. Then, it iteratively selects one tree node from the buffer and determines possible candidate child nodes from the selected tree node until the buffer is empty. Those candidate child nodes still need to put in the buffer to explore more child nodes. Consequently, algorithm BF is likely to degrade its performance if the SP-tree is flat. The reason is that a considerable amount of memory space is for storing the projected trajectory set of a flat SP-tree. As such, we further propose algorithm DF, which is a depth-first approach, to derive the SP-trees. In the beginning, algorithm DF generates one root node in the SP-tree and its conditional table which has the probability of hot regions being larger than a minimum probability threshold $MinProb$ (line 1 to line 2). For each frequent hot region, algorithm DF will check whether it is in the conditional table or not. If it is in the conditional table with its probability larger than $MinProb$, algorithm DF will generate one tree edge labeled as the corresponding frequent hot region and one child tree node adjunct with the tree edge. For the new child tree node, algorithm DF will further derive the conditional table of the new child node. It will then further identify possible frequent hot regions from the conditional table. Note that algorithm DF is recursively performed (line 5 to line 7). Hence, a depth-first trace from the root node to the leaf tree node is derived.

For example, to construct SP-tree SPT_1 for user U_1, the setting of $MinSup$ and $MinProb$ is the same as the setting in 4.1. Fig. 3 shows the process of constructing an SP-tree using algorithm DF. In the beginning, DF($\mathcal{T}$, $root$, 0.3, 0.4) is called, and there is only one root node in SPT_1. Then, the frequent locations of transformed trajectories $\mathcal{T}'$ are A, B, and C (line 1), and there are three entries A, B, and C in the

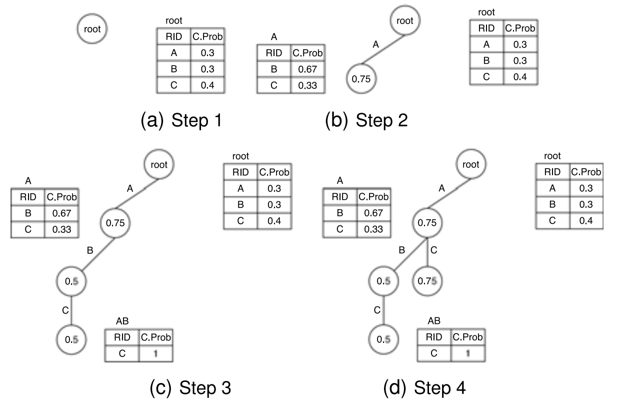


Fig. 3. An example of constructing the SP-tree of user U_1 using algorithm DF.

conditional table of $root$ (line 2, Fig. 3a). To create child nodes of $root$, the procedure checks whether each frequent location is in the conditional table (line 4). In this case, frequent locations, A, B, and C, are all in the conditional table. A is a child of $root$ (line 5). Thus, DF($\mathcal{T}'$, A, 0.3, 0.4) is called in line 7, where $\mathcal{T}'$ is $\langle B, C\rangle, \langle B, C\rangle, \langle C\rangle$ (Fig. 3b). Following the procedure, DF($\mathcal{T}'$, AB, 0.3, 0.4) calls DF($\mathcal{T}'$, ABC, 0.3, 0.4) (Fig. 3c), and DF($\mathcal{T}'$, A, 0.3, 0.4) calls DF($\mathcal{T}'$, AC, 0.3, 0.4) after calling DF($\mathcal{T}'$, AB, 0.3, 0.4) in the for each loop (line 3). The final SP-tree is shown in Fig. 1b.

Algorithm 2 DF

```
Input: A set of transformed trajectories $\mathcal{T}$, parent node $s$, a minimum probability threshold $MinProb$, a minimum support threshold $MinSup$
Output: An SP-tree $SPT$
1: find frequent locations
2: create conditional table of node $s$
3: for each $\sigma$ in frequent locations do
4:     if $\sigma$ in conditional table of node $s$ then
5:         $s\sigma$ is a child of $s$
6:         create projected transformed trajectory set $\mathcal{T}'$
7:         DF($\mathcal{T}'$, $s\sigma$, $MinProb$, $MinSup$)
8:     end if
9: end for
```

Complexity analysis. For the time complexity analysis of algorithm DF, from line 1 to line 2, for each trajectory, it needs to update the count and set the conditional table for each region in one trajectory. Thus, these two steps take $O(|\mathcal{T}|w)$ time, where w is the average number of hot regions in the trajectories. In line 5, it only takes $O(1)$ time, and in line 6, to generate the set of projected trajectories σ, it needs to scan the original set of trajectories $\mathcal{T}$. Thus, it costs $O(1) + O(|\mathcal{T}|)$ in the loop iterations. Furthermore, the loop from line 4 to line 8 runs $|\Sigma|$ times. Hence, the time complexity of algorithm DF from initial call is $O(\sum_{n_{call}}(|\mathcal{T}|w + |\Sigma||\mathcal{T}|))$, where n_call is the number of recursive calls of algorithm DF. For the space complexity analysis for algorithm DF, it only needs to store the trajectory set $\mathcal{T}$. Thus, the space complexity of algorithm DF from initial call is $O(n_{depth}|\mathcal{T}|w)$, where n_{depth} is the depth of the recursive call of algorithm DF.

5 Movement-Based Community Discovery

In this section, we present how to cluster users in terms of their trajectory profiles (i.e., SP-trees). Given a set of users with their SP-trees, our goal is to cluster users into groups where users in the same group have a certain degree of movement similarity. To achieve this goal, similarity functions based on SP-trees are presented. In light of the devised similarity degrees, we further propose one clustering algorithm to derive communities of users. The following sections show our approaches for calculating similarity and discovering communities in detail.

5.1 Similarity Function Based on SP-Tree Structures

As pointed out earlier, SP-trees of users capture user movement behavior. We thus design a new similarity function that explores the tree structures of SP-trees. Because an SP-tree contains both sequential patterns and transition probabilities of hot regions, the design of our proposed similarity function, denoted as Sim_{sp}, will consider the following tree structure information:

1. *The number of common tree nodes.* A node in an SP-tree is in fact a sequential pattern, which indicates a frequent sequence of hot regions from user trajectories. As such, the more common nodes that two SP-trees have, the more similar the movement behavior of the two users. For example, consider SP-trees SPT_1, SPT_2 and SPT_4 in Fig. 1b and Fig. 4. The

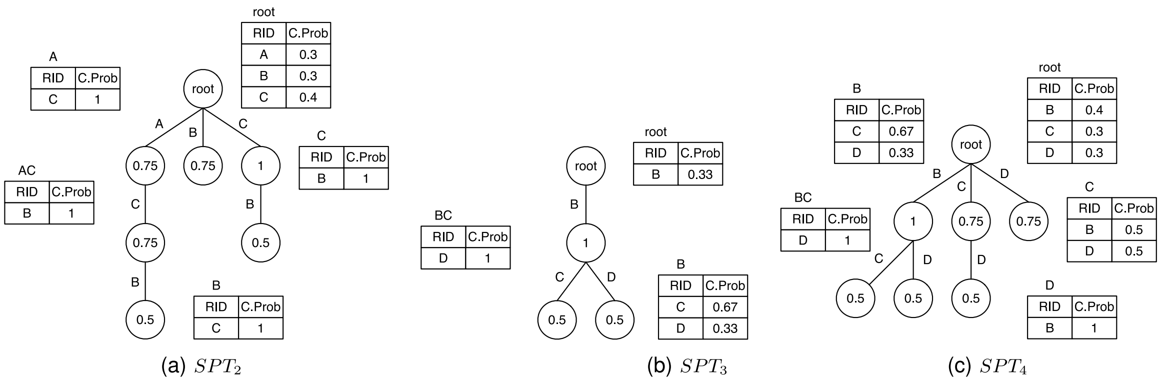


Fig. 4. SP-trees of user U_2, U_3 and U_4.

number of common nodes between SPT_1 and SPT_2 is 5 (i.e., $root$, A, B, C, and AC), while between SPT_1 and SPT_4 is 4 (i.e., $root$, B, C, and BC). Clearly, SPT_1 is more similar to SPT_2 than to SPT_4.

2. *The number of different tree nodes.* Due to the fact that the number of nodes in an SP-tree captures frequent user movement regions, the similarity function should consider the different number of nodes in SP-trees. The reason is that it is possible that two users may have common nodes in their SP-trees but their movement coverage ranges are different. For example, user U_i may have a variety of movement ranges and have an SP-tree with more number of nodes. On the other hand, if user U_j does not travel a lot and thus has a smaller movement coverage range and if these two users have some common nodes in their SP-trees, they are likely to have similar movement behavior. However, these two users have different movement behavior in terms of their movement coverage ranges. Thus, the number of different nodes in the SP-trees should be considered in the similarity function.
3. *Support values of tree nodes.* The support value of a node indicates how frequently the sequential pattern appears among user trajectories. Different support values of sequential patterns represent different movement behavior. Clearly, the closer the support values of two common nodes are, the more similar their movement behavior is. For example, SPT_3 and SPT_4 have the common nodes B and BC with the same support values (i.e., 1 and 0.5). In the case of SPT_1 and SPT_3, their common nodes B and BC have different support values. Consequently, the sequential patterns B and BC between SPT_3 and SPT_4 are more similar than those between SPT_1 and SPT_3. Furthermore, the support value of a sequential pattern also refers to the appearing frequency of this pattern. Obviously, the higher the support value is, the more important the sequential pattern is. Therefore, the support values of nodes in SP-trees should infer the importance of nodes in deriving the similarity scores.
4. *Conditional tables of tree nodes.* Each node in an SP-tree has the conditional table that demonstrates the next movements. From the conditional tables of the tree nodes, one could have the transition probability of the hot regions, which indicate more detailed movement behavior of users. Note that it is possible that two SP-trees have the same sequential pattern but the next movement probabilities are not the same. For example, consider SPT_1 and SPT_2. Although they have the common node A with the same support value 0.75, the conditional table of node A in SPT_1 is totally different from that in SPT_2. After staying at hot region A, User U_1 may go to hot regions B and C. However, User U_2 may go to only hot region C. By exploring the probabilities of next movements in the SP-trees, it is more accurate to evaluate how similar two SP-trees are by exploring the conditional tables of the tree nodes in the SP-trees.

To consider the above information of SP-trees in a similarity function, the main idea is to compute the similarity scores of each pair of common tree nodes in two SP-trees based on their support values and conditional tables of tree nodes. As such, there are two important similarity score functions: Sim_N and Sim_T. Then, the similarity function $Sim_{SP}(\cdot)$ is thus derived by summarizing the above scores and normalizing the scores based on the number of tree nodes.

Let two SP-trees be SPT_i and SPT_j, and nodes in SPT_i with the sequential pattern s are represented by N_i^s. Given two nodes N_i^s and N_j^s, the similarity score for these two nodes is defined as in Equation (1)

$$Sim_N(N_i^s, N_j^s) = \begin{cases} 1, \text{ if } s = root \\ \left(1 - \left|sup(N_i^s) - sup(N_j^s)\right|\right) \\ \times \dfrac{sup(N_i^s) + sup(N_j^s)}{2}, \text{ otherwise.} \end{cases} \tag{1}$$

For two nodes with a common sequential pattern, the first term considers the closeness of their support values. If these support values are similar to each other, the difference between them becomes smaller. Therefore, a larger value of the first term indicates that two support values are relatively close. The second term considers the weights of their support values. Tree nodes with larger support values should be more important in the similarity measurement. Therefore, the average of two support values shows how important the common sequential pattern is when we consider the similarity of these two nodes. For example, consider two nodes N_1^{BC} and N_3^{BC} in SPT_1 and SPT_3, respectively. We can derive $Sim_N(N_1^{BC}, N_3^{BC}) = (1 - |0.75 - 0.5|) \times \frac{0.75+0.5}{2} = 0.47$.

The conditional table of a node N_i^s is used to record the probabilities of next movements after the sequential pattern s. Clearly, the smaller difference of the two conditional tables of the two common nodes should be more similar in the similarity measurement. Phase 1 only adds the next movement for which the probability exceeds $MinProb$ into a conditional table. However, the sum of next movement probabilities is not equal to 1. In this case, if the sum of probabilities is p, we can still view the probability of a null next movement as $1 - p$. Thus, the conditional table of each node can be viewed as a probability distribution. To evaluate the similarity between two conditional tables, the well-known probability distribution distance can be adopted. Let the conditional table of N_i^s be C_i^s. Then, define the similarity score for the conditional tables using Equation (2)

$$Sim_T(C_i^s, C_j^s) = 1 - \frac{\sum_{\rho \in C_i^s \cup C_j^s} \left|Pr(C_i^s(\rho)) - Pr(C_j^s(\rho))\right|}{\left|C_i^s \cup C_j^s\right|}. \tag{2}$$

For example, consider the conditional tables C_3^B and C_4^B of the node B in SPT_3 and SPT_4. Because there are two common symbols (i.e., C and D), we can derive that $Sim_T(C_3^B, C_4^B) = 1 - \frac{|0.67-0.67|+|0.33-0.33|}{2} = 1$. Further, $Sim_T(C_2^B, {}_3^B) = 1 - \frac{|1-0.67|+|0-0.33|}{2} = 0.67$. The movement behavior of C_3^B and C_4^B shows that users may move to hot region C and hot region D after hot region B. Thus, these movement behavior is more similar than those of C_2^B and C_3^B. This can be reflected in that $Sim_T(C_3^B, C_4^B) > Sim_T\ (C_2^B, C_3^B)$.

Given the similarity scores Sim_N and Sim_C, the similarity measurement of two SP-trees, Sim_{SP}, can be derived. To consider the number of common nodes of two SP-trees, Sim_{SP} sums up the similarity scores of these nodes, and then normalizes them by the total number of tree nodes. Because the number of different nodes in two SP-trees equals the difference in the total number of nodes and the number of common nodes, considering the number of common nodes and the total number of tree nodes is sufficient to represent the number of different nodes in two SP-trees. Moreover, the total number of nodes may vary in different SP-trees. The greater the number of nodes in an SP-tree, the more likely it is that this SP-tree will have more nodes in common with other SP-trees. Meanwhile, it is much easier to have many different nodes from others as well. Therefore, normalizing the overall similarity based on the total number of nodes in two SP-trees can reflect this phenomenon

$$Sim_{SP}(SPT_i, SPT_j) = \frac{\sum_{s \in S_{i,j}} Sim_N(N_i^s, N_j^s) \times Sim_T(C_i^s, C_j^s)}{|SPT_i \cup SPT_j|}. \tag{3}$$

In SPT_1 and SPT_2, for example, $S_{1,2} = \{root, A, B, C, AC\}$. For each node, compute two similarity scores as follows: for the root node, we can obtain that $Sim_N(N_1^{root}, N_2^{root}) = 1$ and $Sim_T(C_1^{root}, C_2^{root}) = 1 - \frac{0+0+0}{3} = 1$; for node A, $Sim_N(N_1^A, N_2^A) = (1 - |0.75 - 0.75|) \times \frac{0.75+0.75}{2} = 0.75$ and $Sim_T(C_1^A, C_2^A) = 1 - \frac{0.67+0.67}{2} = 0.33$; for node B, $Sim_N\ (N_1^B, N_2^B) = (1 - |0.75 - 0.75|) \times \frac{0.75+0.75}{2} = 0.75$ and $Sim_T\ (C_1^B, C_2^B) = 1 - \frac{1-1}{1} = 1$; for node C, $Sim_N(N_1^C, N_2^C) = (1 - |1 - 1|) \times \frac{1+1}{2} = 1$ and $Sim_T(C_1^C, C_2^C) = 1 - \frac{1-0}{1} = 0$; for node AC, $Sim_N(N_1^{AC}, N_2^{AC}) = (1 - |0.75 - 0.75|) \times \frac{0.75+0.75}{2} = 0.75$ and $Sim_T(C_1^{AC}, C_2^{AC}) = 1 - \frac{1-0}{1} = 0$. To put them all together: because the number of nodes of $SPT_1 \cup SPT_3$ is 10, the similarity between nodes can be derived as $Sim_{SP}(SPT_1, SPT_2) = (1 \times 1 + 0.75 \times 0.33 + 0.75 \times 1 + 1 \times 0 + 0.75 \times 0)/ 10 = 0.20$. In a similar fashion, the similarity between SPT_1 and SPT_4 is $Sim_{SP}(SPT_1, SPT_4) = 0.14$. This result is consistent with the intuition that SPT_1 is more similar to SPT_2 than to SPT_4. In other words, it can be verified in Fig. 1a and Fig. 4a that user U_1 and user U_2 usually move in hot regions A, B, C, whereas User U_4 usually moves in hot regions B, C, and D.

Complexity analysis. To compute the similarity $Sim_{SP}(SPT_i, SPT_j)$, compute the similarity score of common nodes and their conditional tables first. For any common node in SPT_i and SPT_j, say N_i^s and N_j^s, the similarity score of the two nodes $Sim_N(N_i^s, N_j^s)$ is $O(1)$ because this value can be computed by their support values. On the other hand, the similarity score for the conditional table $Sim_T(C_i^s, C_j^s)$ is $|\Sigma|$ in the worst case because the conditional tables contain at most $|\Sigma|$ entries. Let the set of common nodes for these two trees be $S_{i,j}$. Computing Sim_{SP} (SPT_i, SPT_j) takes $|S_{ij}| \times O(1) \times O(|\Sigma|) = O(|S_{ij}||\Sigma|)$. On the other hand, computing the similarity score of each node in common nodes only needs $O(1)$ space. Thus, the space complexity of Sim_{SP} is $O(1)$.

5.2 Clustering Users Based on SP-Tree Similarities

According to the above similarity functions, the similarity scores of users are derived. Then, in light of these scores, we first explore a graph structure to represent users' movement similarity relationships. Based on the graph structure, we formulate an objective function to evaluate the quality of clustering results and thus propose one algorithm to derive clusters.

Two users have a *geo-connection* relationship if the similarity score of their trajectory profiles exceeds a threshold δ. The threshold δ is a pre-defined threshold that could be viewed as a minimum similarity bound in movement-based communities. According to the geo-connection relationships among users, a geo-connection graph is thus built as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents users and $E = \{(v_i, v_j) | Sim_{SP}(SPT_i, SPT_j) > \delta\}$ indicates the similarity scores between user U_i and user U_j. Given a geo-connection graph $G = (V, E)$, our goal is to derive a set of clusters (i.e., sub-graph components) where each sub-graph component represents a set of nodes that have a certain degree of movement similarity.

Prior works in [18], [20], [26] have elaborated on community discovery in social networks, where the social network is represented as a graph structure. Since a variety of community structures exist in a graph, for different purposes, prior works [18], [20], [26] would formally derive different objective functions to evaluate the quality of communities. As in the prior works [18], [20], [26], we formulate our objective function to evaluate the quality of the cluster results in the geo-connection graph. The main purpose of this function is to represent the whole geo-connection graph as perfect communities (i.e., disjoint cliques) by adding or removing the minimum number of edges. Hence, the objective function consists of two parts: the intra-cost score and the inter-cost score. Minimizing the objective function reveals the user communities with the minimum number of modified edges. For a community, use the *intra-cost* score, which refers to the minimum number of edges added into these nodes in this community, to make them a clique. Formally, the intra-cost score of a community $C_i = (V_i, E_i)$ is formulated as in Equation (4)

$$Cost_{intra}(C_i) = |K_{|V_i|}| - |E_i|, \text{ where } K_{|V_i|} \text{ is a } |V_i|\text{-clique.} \tag{4}$$

As mentioned previously, it is possible to infer whether two nodes are in a community based on the number of common neighboring nodes between them. The intra-cost score can reflect this phenomenon. For example, Fig. 5a shows two sets of nodes $C_1 = \{v_5, v_6, v_7, v_8\}$ and $C_2 = \{v_1, v_2, v_4, v_5\}$. C_1 is more likely to be a community than C_2 because every two nodes in C_1 have two common neighboring nodes, whereas v_4 is far from v_5. Thus,

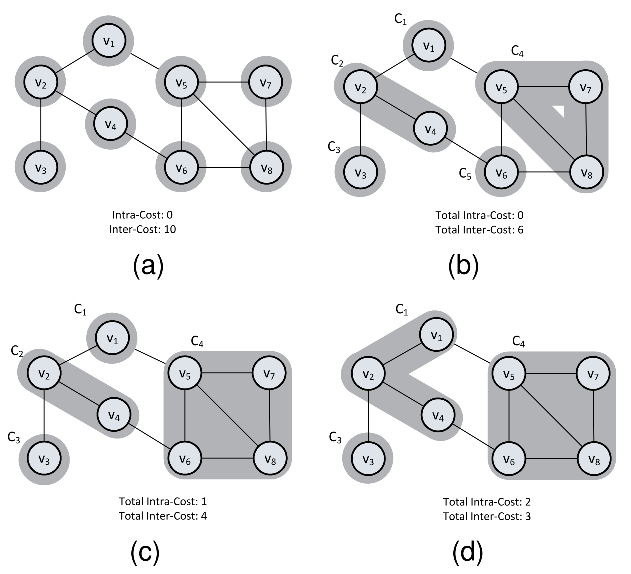


Fig. 5. An example of algorithm Geo-Cluster for mining user communities.

these nodes are hardly recognized as members of the same community. With communities in a geo-connection graph, the *inter-cost* represents the minimum number of edges removed from the geo-connection graph to make the communities disjointed from each other. Consequently, the inter-cost score of two communities $C_i = (V_i, E_i)$ and $C_j = (V_j, E_j)$ can be formulated as in Equation (5)

$$Cost_{inter}(C_i, C_j) = |\{(v_i, v_j)|v_i \in V_i, v_j \in V_j\}|. \tag{5}$$

To combine these two scores, the objective function for a set of user communities $C = \{C_1, C_2, \ldots, C_n\}$ can be derived as Equation (6)

$$\begin{aligned} Cost_{total}(C) = &\sum_{C_i \in C} Cost_{intra}(C_i) \\ &+ \sum_{C_i, C_j \in C} Cost_{inter}(C_i, C_j). \end{aligned} \tag{6}$$

Based on the derived objective function, algorithm Geo-Cluster, shown in algorithm 3, is proposed. Algorithm Geo-Cluster first constructs a geo-connection graph based on the similarities of trees and the threshold δ. In the beginning, each vertex is viewed as a community (line 2), and the total cost scores for merging any two communities are computed (line 10). Once the total cost can be reduced, two communities are set as the candidate communities (X_i, X_j) in line 13. After exiting the for loop in line 9 to line 15, the two communities that can minimize the total cost score are found, and they are merged into one. This algorithm terminates when the total cost score cannot be further reduced or none of the communities can be expanded.

Fig. 5 shows a running example of algorithm Geo-Cluster. Initially, Fig. 5a shows each node as a community. The total intra-cost score is 0 and the total inter-cost score is 10. Fig. 5b shows the results after three merges. Following the result in Fig. 5a, the $LastCost$ is 10. Geo-Cluster enumerates every pair from C and examines whether the merger of two communities can reduce the $LastCost$ or not. First, merging v_2 and v_4 into C_2 increases the total intra-cost by 0 and decreases the total inter-score by 1. Therefore, the value of $LastCost$ decreases by 1. Similarly, both merging v_5 and v_7 into a community $C^{'}$ and merging $C^{'}$ to v_8 into a community can reduce the $LastCost$ since the total intra-cost increases by 0 and the inter-cost increases by 1 and 2, respectively. After this iteration, there are five communities C_1, $C_2, \ldots, \text{and } C_5$. Fig. 5c shows an intermediate result in which these five communities have an intra-cost score of 0 and the inter-cost score is 6. Merging C_4 and C_5 increases the total intra-cost score by 1, while decreaseing the total inter-cost by 2. Thus, Fig. 5c shows that C_4 and C_5 are then merged and the total cost is 5 (i.e., $1 + 4$). Fig. 5d shows the final result with three communities C_1, C_3, and C_4.

Algorithm 3 : Geo-Cluster

```
Input: User profiles: ${SPT_1, ..., SPT_n}$, and thresholds: $\delta$
Output: $C$, communities of users
1:  Construct a geo-connection graph $G = (V, E)$ by ${SPT_1, ..., SPT_n}$ and $\delta$;
2:  $C \leftarrow V$;
3:  $LastCost \leftarrow Cost_{total}(C)$;
4:  $Flag \leftarrow True$;
5:  while $Flag = True$ do
6:      $Flag \leftarrow False$;
7:      $(X_i, X_j) \leftarrow (\phi, \phi)$;
8:      $MinCost \leftarrow LastCost$
9:      for each community $C_i, C_j \in C$ do
10:         $CurrCost \leftarrow LastCost + |C_i \times C_j| - 2 \cdot Cost_{inter}(C_i, C_j)$
11:         if $CurrCost \leq MinCost$ then
12:             $Flag \leftarrow True$;
13:             $(X_i, X_j) \leftarrow (C_i, C_j)$;
14:         end if
15:     end for
16:     if $Flag = True$ then
17:         Merge $X_i$ and $X_j$;
18:         $LastCost \leftarrow MinCost$;
19:     end if
20: end while
```

Complexity analysis. Since the algorithm utilizes $Cost_{total}$ to find communities, we start from analyzing the computational cost of $Cost_{total}$. Let a set of communities C be $C_1, C_2, \ldots, C_n$, i.e., in $|V| - n + 1$th round in algorithm Geo-Cluster. The time cost of $Cost_{total}$ consists of two components: the sum of $Cost_{intra}(C_i)$, where $C_i \in C$ and the sum of $Cost_{inter}(C_i, C_j)$, where $C_i, C_j \in C$. For a community C_i, it takes $O(1)$ to compute $Cost_{intra}(C_i)$. Totally, computing the first term takes $n \times O(1) = O(n)$. Given two communities C_i and C_j, it takes $O(|C_i| \times |C_j|)$ to compute $Cost_{inter}(C_i, C_j)$. Therefore, computing the second terms takes $O(C(n, 2)) = O(n^2)$.

TABLE 1
Data Set Statistics

	kstaxi		**everytrail**	
	all	used	all	top-10%
# of users	769	20	3,763	376
# of trajectories	N/A	5,707	16,142	8,218
# records	28,849,326	208,303	18,556,671	10,097,506

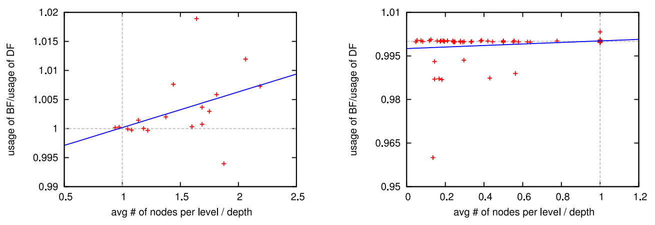
In line 1, constructing the network needs cost time $O(|\Sigma| \sum_{v_i,v_j \in V} |S_{i,j}|)$ to calculate the similarity of each pair of users. Since C is initialized as the whole vertex in line 2, line 3 takes $O(|V|^2)$-time to compute the initial $Cost_{total}(C)$. The while loop from line 5 to line 20 keeps executing when this iteration can make any reduction of $LastCost$. Therefore, in the worst case, the while loop will execute for $|V|$ iterations. Inside the while loop, the total time cost of the for loop from line 9 to line 15 is $O(|C|^2)$. Since C is initialized to be V, the computational cost from line 5 to line 20 is $O(|V|^2) + O((|V|-1)^2) + O((|V|-2)^2) + \cdots = O(|V|^3)$. Finally, the time complexity of algorithm Geo-Cluster is $(O(|\Sigma| \sum_{v_i,v_j \in V} |S_{i,j}| + |V|^3)$. On the other hand, in line 1, it needs $O(|V| \times |V|)$ space to store the connections between users. In lines 5 to 20, it needs $O(|V|)$ space to store the community label of each user.

6 Performance Evaluation

In this section, extensive experiments are conducted to evaluate the effectiveness and efficiency of our proposed framework. We implemented the proposed algorithms in Python.

6.1 Data Sets Description

To show the performance, there are two data sets, kstaxi and everytrail, in the experiment. To the best of our knowledge, no real data sets are labeled for mining user communities by their trajectories. The kstaxi data set is used in our experiments based on the observation that taxi drivers have their own preference for finding customers [22]. The kstaxi data set contains the recorded GPS trajectories of taxi drivers in Kaohsiung, Taiwan, during the period from September to December 2008. There are totally 28,849,326 records consisting of driver identifiers, dates and times, coordinates, and occupying status. To simulate user communities, we selected five drivers, denoted by $A, B, \ldots E$ in the experiments, and there are 208,303 records of the selected five drivers. Since the movement behavior may be different when a taxi is occupied or not occupied by customers, a raw trajectory is split whenever the occupying status is changed, giving a total of 5,707 trajectories. In order to test the effectiveness of detecting movement-based communities, we design a movement community scenario from real trajectory data. From a data set of taxi drivers' trajectories, we select five taxi drivers' trajectories. Those drivers are supposed to have individual movement behavior. Thus, a four-month individual user's trajectory data can be considered as a movement community including four synthetic users. Since trajectories are extracted from five drivers over four months, we have 20 users, denoted by $A_{09}, A_{10}, \ldots, A_{12}$, $B_{09}, B_{10}, \ldots$ and so on. Thus, they are taken as the ground truth in our experiments. Moreover, to show the results of our proposed methods for location-based social services, we also selected a data set, everytrail, from a location-based social service, EveryTrail [1]. We collected a total of 18,556,671 records from EveryTrail. Because there are many users with few trajectories and records, it is not enough to represent individual movement behavior. Therefore, we selected the top-10 percent of users that have the largest number of trajectories, and we focused on the records from Mountain View, California in everytrail. After the filtering out process, there were 10,097,506 records in everytrail. Table 1 shows the detailed statistics of these data sets. The default values of the parameters are listed as follows: for the kstaxi data set $MinSup = 0.1$, $MinProb = 0.1$, $\delta = 0.1$; for the everytrail data set $MinSup = 0.2$, $MinProb = 0.2$, $\delta = 0.2$.



(a) kstaxi (b) everytrail

Fig. 6. Comparison of memory usage of different approaches of profile construction.

6.2 Comparison of Algorithms for Construing SP-trees

In this section, we compare the proposed approaches for movement profile construction. Our proposed approaches, BF and DF, were designed to construct the SP-tree for two different movement behaviors. In our consideration, BF is adopted to save the memory storage for constructing the SP-tree for the user with long and coherent trajectories; on the contrary, DF was developed to construct the SP-tree for those users with short and sparse trajectories. To demonstrate our considerations, we use two factors to observe the relationship between the form of the SP-tree and the difference in memory usage of BF and DF on our data sets. The first factor is to represent whether the SP-tree is flat, so the average number of nodes per level over the depth of the SP-tree is selected. When the first factor is larger than 1, it means that the SP-tree is flat, and vice versa. The second factor is the ratio of memory usage of BF to memory usage of DF. If the second factor is larger than 1, it means that the memory usage of BF is smaller than the memory usage of DF.

Fig. 6 shows the relation between the two factors of each user profile (SP-tree) in our data sets. In Fig. 6a, most of the SP-trees in the kstaxi data set are flat, and the correlation coefficient is 0.42 which is larger than 0, reflecting that the memory usage of DF is smaller than the memory usage of BF. Moreover, Fig. 6b also shows the same results for the everytrail data set since the correlation coefficient is 0.15. However, these two figures show that most of the SP-trees in the kstaxi data set are flat while most of the SP-trees in the everytrail data set are deep. This reflects different user movement behavior in different data sets. On the other hand, Fig. 7 shows the total memory usage of BF and DF for

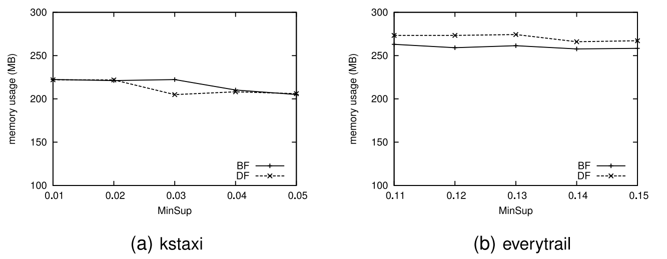


Fig. 7. Comparison of memory usage of different approaches on different data sets.

the two data sets. Fig. 7a shows that the memory usage of DF is efficient but not significant since most of the SP-trees are flat in the kstaxi data set. However, Fig. 7b shows that the memory usage of BF is efficient as most of the SP-trees are deep in the kstaxi data set. Indeed, the results show that the destinations of taxi drivers are decided by passengers. Thus the movement behavior of taxi drivers is spread out. Furthermore, users in the everytrail data set prefer to take similar trip routes such that most user SP-trees are deep. Finally, the results reflect that our proposed approaches are able to construct the SP-trees efficiently for different types of user behavior.

6.3 Comparison of Similarity Measurements

In this section, we compare our similarity function Sim_{SP} with the previously mentioned baseline methods, Cosine and LCSS. Furthermore, the similarity function, HGSM [21], [32], designed for comparing the similarity between two users by trajectory data is implemented for performance comparison. On the other hand, in order to show that our similarity function Sim_{SP} considers not only node similarity Sim_N (i.e., sequential patterns of locations) but also conditional table similarity Sim_T (i.e., probabilities of next movements), we modify the Sim_{SP} without Sim_T denoted by Sim_{SP} w/o Sim_T. Therefore, we have five similarity functions to compare the performance on the kstaxi data set.

To compare with different similarity measurements, we use the one nearest neighborhood (1NN) classifier [9], [29] to evaluate the performance by different similarity measurements. The closest user to another user is in the same community when using a better similarity measurement. Fig. 8a shows the results of the compariso. Sim_{SP}, Sim_{SP} w/o Sim_T and Cosine have the highest precision of 0.95 in different MinSup. LCSS shows that higher precision resulted by higher *MinProb* since it filters out the noise in users' trajectories with higher *MinProb*, and then the similar users have similar hot regions. HGSM has its profile construction method without *MinSup* and *MinProb*; thus the precision of HGSM is a constant for different *MinSup* and *MinProb*. Moreover, HGSM shows precision lower than 0.5 since the user profile in HGSM only contains one trajectory, and the idea of similarity in HGSM is similar to that of LCSS. Hence HGSM has lower precision in the experimental results.

In 1NN evaluation, the Sim_{SP}, Sim_{SP} w/o Sim_T and Cosine have the highest precision. Therefore, we apply these similarity functions in our proposed community detection algorithm to compare them. We first introduce two metrics, *entropy* and *purity*, in this series of experiments. The entropy

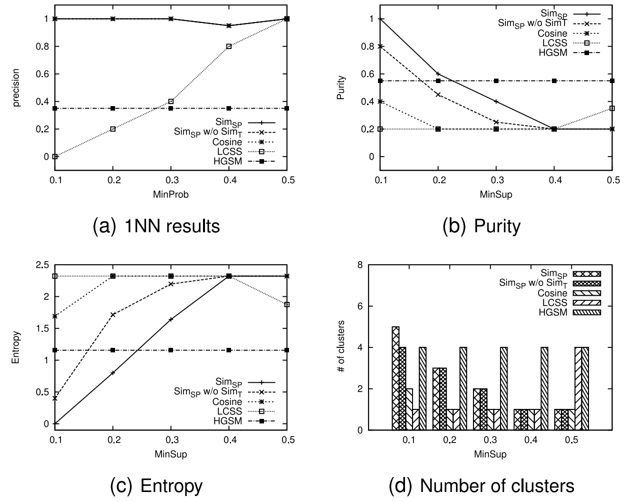


Fig. 8. Comparison of different similarity functions on kstaxi data set.

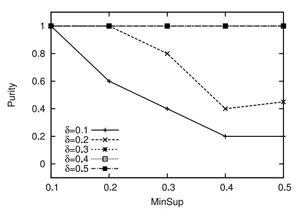
and purity metrics are used to measure the quality of the community results, where entropy is a function of the distribution of classes in the resulting communities, and purity is a function of the relative size of the largest class in the resulting communities [8]. The smaller the entropy value is, the better the derived communities are. The larger the purity value is, the better the derived communities are. However, for the everytrail data set, there is no ground truth. Therefore, we use the *silhouette coefficient*[1] [13] which is a popular measurement combining both cohesion and separation to show the results between communities in these two data sets. The value of this coefficient is between -1 and 1. The larger the value, the larger the similarity values in the same community (i.e., close to 1) and the smaller the values between communities (i.e., close to 0). Thus, we use these factors to evaluate the results after clustering.

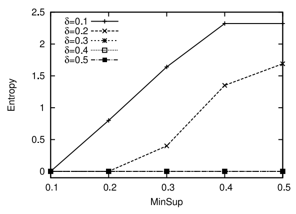
Fig. 8b and Fig. 8c show that Sim_{SP} has the best performance (the highest purity and the lowest entropy) in most cases while Sim_{SP} w/o Sim_T places second for the kstaxi data set. The performance of Sim_{SP} w/o Sim_T is better (higher purity and lower entropy) than that of Cosine, as Cosine only considers the current locations of two SP-trees. In other words, if only considering the current locations, Sim_{SP} w/o Sim_T can reflect the relation between two SP-trees better than Cosine can. However, LCSS has a bad performance on kstaxi. Fig. 8d reflects this phenomenon; thus, it is too difficult to detect geo-commentates when using LCSS as a similarity function. Finally, the results show that Sim_{SP} has better performance of similarity comparison on users' trajectories and therefore is a suitable similarity function for SP-trees in our experiments.

6.4 Impact of $MinSup$ and δ

In this section, the impact of the parameters $MinSup$ and δ are examined. Fig. 9 shows the impact of thresholds δ and $MinSup$ on kstaxi. Recall that $MinSup$ decides the minimum number of sequential patterns appearing in trajectories. A

1. Originally, the silhouette coefficient is defined in terms of distance function. Here, we use similarity scores so we rewrite the silhouette coefficient in this form.


(a) Purity


(b) Entropy

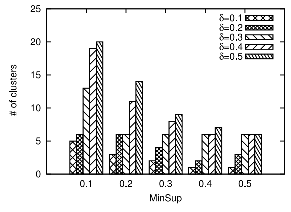
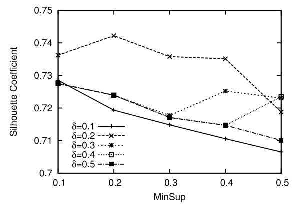

(c) Number of clusters

Fig. 9. Sensitivity analysis for $MinSup$ and δ for the kstaxi data set.

larger $MinSup$ tends to make SP-trees smaller, while δ decides the number of edges in a network. A larger δ tends to make the number of edges larger. First, Fig. 9a and Fig. 9b show the purity and entropy values with $MinSup$ and δ varied. First, in the case of δ being larger than 0.2, the purity and entropy values are 1 and 0, respectively. This indicates that our proposed method can successfully discover the correct communities. Because it is stricter to identify whether two users have geo-connections, $MinSup$ does not affect the results significantly. In the case of $\delta = 0.1$, a larger $MinSup$ leads to a decrease in purity and an increase in entropy. Finally, $\delta = 0.1$ makes too many social connections between users such that it is difficult to discover the correct communities in all cases. The results also show that with a larger δ, the influence of $MinSup$ is weak. From another point of view, Fig. 9c shows the number of communities in different settings for the kstaxi data set. Recall that the ground truth is that all users are divided into five communities. Interestingly, a large δ may divide users who belong to the same community in the ground truth into several smaller communities. For example, when $\delta = 0.3$, although the purity and entropy values are always 0 and 1, the number of communities is always larger than 5. In our experiments, $\delta = 0.1$ is the optimal setting with which our proposed method can discover the correct communities when $MinSup = 0.1$. In the everytrail data set, the silhouette coefficient for different values of $MinSup$ gently decrease with the same δ in Fig. 10a. The reason is that when there is a larger $MinSup$, the nodes of users' SP-tree are lower, then there is greater similarity between users, so it is difficult to discover distinct communities. Fig. 10b shows that the


(a) Silhouette Coefficient

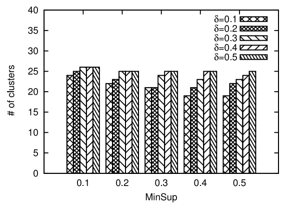
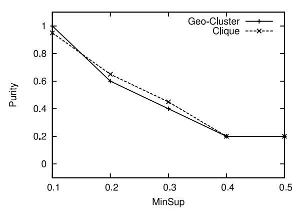
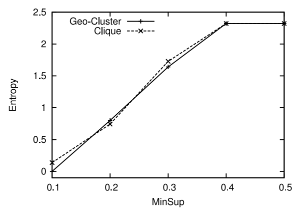

(b) Number of clusters

Fig. 10. Sensitivity analysis for $MinSup$ and δ for the everytrail data set.


(a) Purity for Clique


(b) Entropy for Clique

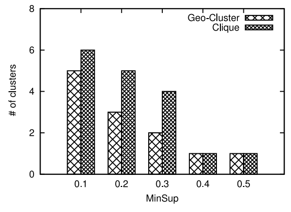

(c) Number of clusters for Clique

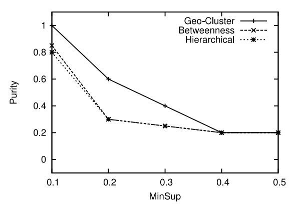
Fig. 11. Comparison of the Clique algorithm and our approach on the kstaxi data set.

number of clusters has the same trend for everytrail. Overall, δ is a more significant dominant factor of correctness, while $MinSup$ could also provide some enhancements in some cases for all data sets.

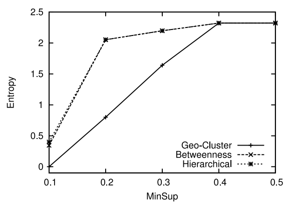
6.5 Comparison of Community Discovering Methods

In this section, we conduct experiments to compare the existing community methods with ours. For comparison different approaches for detecting communities, we implement three methods: the Clique method [14], the Hierarchical method [27], and the Betweenness method [12]. The Clique method is a straightforward approach which identifies a set of users with social connections to each other as a community. The Hierarchical method begins with each user being a community, and then iteratively merges two communities with the smallest average similarity score. We use the agglomerative hierarchical method which begins with each user being a community, and then iteratively merges two communities with the largest similarity score. The Betweenness method takes a different approach from the Hierarchical method. The concept is that the shortest path between two users in social networks reflects the path of a message from a user to another user. Thus it begins with all users as a single community, and then removes edges that belong to the shortest paths between most pairs of users.

In kstaxi data set, Figs. 11a and 11b show that the Clique method has mostly better quality community results than the Geo-Cluster method. However, Fig. 11c shows that the number of communities using the Clique method is always larger than the ground truth (i.e., 5). On the contrary, our proposed method can find the correct number of communities in most cases. This is because the Clique method is too strict in terms of identifying every two users in the same community as being linked to each other. It is also not desirable to separate users into communities that are too small. From this point of view, our proposed method is likely to strike a better result than the Clique method.



(a) Purity



(b) Entropy

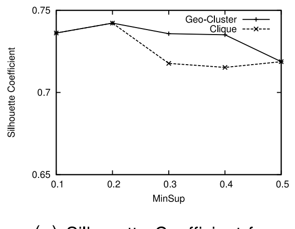
Fig. 12. Purity and entropy for community methods, Betweenness and Hierarchical, on the kstaxi data set.

Next, we compare our method with the Hierarchical method and the Betweenness method on kstaxi. These two methods need to pre-determine the number of communities. For a fair comparison, we assign the number of communities which were found by our proposed method. Figs. 12a and 12b show the results that our method could produce better purity and entropy values than the other two methods.

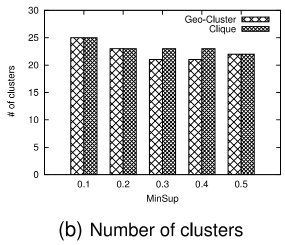
On everytrail, Fig. 13a shows that the coefficient of Clique decreases significantly when $MinSup$ is larger than 0.3. Furthermore, Fig. 13b shows that the trend of the number of clusters on everytrail is the same as the trend of the number of clusters on kstaxi. Moreover, Fig. 13c shows that Geo-Cluster has better performance than Betweenness and Hierarchical. The results show that Geo-Cluster also performs better than the comparative community detection algorithms mentioned above on the everytrail data set.

7 Conclusion

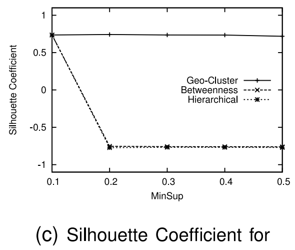
In this paper, we addressed the problem of mining movement-based communities from users' trajectories. We proposed a framework of mining movement-based communities, which consists of three phases: 1) constructing user profiles, 2) identifying the closeness of users by their profiles, and 3) discovering movement-based communities. In the first phase, for each user, we designed an SP-tree to capture not only the sequential patterns but also the transition probabilities of movements. In the second phase, we formulated some similarity functions, including two existing common similarity functions and one new similarity function that explores all of the information of the tree structures. In the third phase, we formulated an objective function to measure the quality of the cluster results. In light of the objective function, we proposed algorithm Geo-Cluster to cluster users. The experimental results demonstrate that our framework can effectively identify the movement-based communities of users. Due to more information and services being available in location-based social networks, prior works in [5], [7], [31] have proposed mining semantic meanings of regions. Therefore, in the future, we aim to extract hot regions with semantic meanings. Based on the semantics of hot regions, we will be able to mine communities that have users with similar movement behavior and activity behavior.



(a) Silhouette Coefficient for Clique



(b) Number of clusters



(c) Silhouette Coefficient for Betweenness and Hierarchical

Fig. 13. Comparison of the cluster algorithms and our approach on the everytrail data set.

Acknowledgments

Wen-Chih Peng was supported in part by the NSC, Project No. 102-2221-E-009-171-MY3 and 100-2218-E-009-016-MY3, by Academic Sinica Theme project, Project No. AS-102-TPA06, by Taiwan MoE ATU Program, and by Microsoft. Po-Ruey Lei was supported in part by the NSC, Project No. 102-2221-E-012-002. Ling-Jyh Chen was supported in part by the NSC, Project No. 102-2219-E-001-002.

References

[1] "Everytrail," http://www.everytrail.com/, 2014.
[2] "Run.Gps Community Server," http://www.gps-sport.net/, 2014.
[3] "Running free," http://www.runningfreeonline.com/, 2014.
[4] "Waze," http://www.waze.com/, 2014.
[5] L.O. Alvares, V. Bogorny, B. Kuijpers, J.A.F. de Macłdo, B. Moelans, and A.A. Vaisman, "A Model for Enriching Trajectories with Semantic Geographical Information," *Proc. 15th Ann. ACM Int'l Symp. Advances in Geographic Information Systems (GIS '07)*, Nov. 2007.
[6] H. Cao, N. Mamoulis, and D.W. Cheung, "Mining Frequent Spatio-Temporal Sequential Patterns," *Proc. Fifth IEEE Int'l Conf. Data Mining*, Nov. 2005.
[7] X. Cao, G. Cong, and C.S. Jensen, "Mining Significant Semantic Locations from GPS Data," *Proc. VLDB Endowment*, vol. 3, no. 1, pp. 1009-1020, Sept. 2010.
[8] J.G. Conrad, K. Al-Kofahi, Y. Zhao, and G. Karypis, "Effective Document Clustering for Large Heterogeneous Law Firm Collections," *Proc. 10th Int'l Joint Conf. Artificial Intelligence and Law*, June 2005.
[9] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures," *Proc. VLDB Endowment*, vol. 1, pp. 1542-1552, Aug. 2008.
[10] F. Giannotti, M. Nanni, and D. Pedreschi, "Efficient Mining of Temporally Annotated Sequences," *Proc. Sixth SIAM Int'l Conf. Data Mining (SDM '06)*, Apr. 2006.
[11] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory Pattern Mining," *Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '07)*, Aug. 2007.
[12] M. Girvan and M.E.J. Newman, "Community Structure in Social and Biological Networks," *Proc. Nat'l Academy of Sciences*, vol. 99, pp. 7821-7826, June 2002.
[13] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, third ed. Morgan Kaufmann, 2011.
[14] C.-C. Hung, C.-W. Chang, and W.-C. Peng, "Mining Trajectory Profiles for Discovering User Communities," *Proc. Int'l Workshop Location Based Social Networks*, Nov. 2009.
[15] H. Jeung, Q. Liu, H.T. Shen, and X. Zhou, "A Hybrid Prediction Model for Moving Objects," *Proc. IEEE Int'l Conf. Data Eng.*, Apr. 2008.

[16] H. Jeung, M.L. Yiu, X. Zhou, C.S. Jensen, and H.T. Shen, "Discovery of Convoys in Trajectory Databases," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 1068-1080, Aug. 2008.

[17] P. Kalnis, N. Mamoulis, and S. Bakiras, "On Discovering Moving Clusters in Spatio-Temporal Data," *Proc. Int'l Conf. Advances in Spatial and Temporal Databases (SSTD '05)*, Aug. 2005.

[18] A. Lancichinetti, F. Radicchi, J.J. Ramasco, and S. Fortunato, "Finding Statistically Significant Communities in Networks," *PLoS ONE*, vol. 6, no. 4, e18961, Apr. 2011.

[19] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory Clustering: A Partition-and-Group Framework," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '07)*, June 2007.

[20] I.X. Leung, P. Hui, P. Li, and J. Crowcroft, "Towards Real-Time Community Detection in Large Network," *Physical Review E*, vol. 79, no. 6, 066107, June 2009.

[21] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, "Mining User Similarity Based on Location History," *Proc. 16th ACM SIGSPATIAL Int'l Conf. Advances in Geographic Information Systems (GIS '08)*, Nov. 2008.

[22] L. Liu, C. Andris, and C. Ratti, "Uncovering Cabdrivers' Behavior Patterns from their Digital Traces," *Computers, Environment and Urban Systems*, vol. 34, no. 6, pp. 541-548, Nov. 2010.

[23] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D.W. Cheung, "Mining, Indexing, and Querying Historical Spatiotemporal Data," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '04)*, Aug. 2004.

[24] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: A Location Predictor on Trajectory Pattern Mining," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '09)*, June 2009.

[25] W.-C. Peng and M.-S. Chen, "Developing Data Allocation Schemes by Incremental Mining of User Moving Patterns in a Mobile Computing System," *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 1, pp. 70-85, Jan./Feb. 2003.

[26] G.-J. Qi, C.C. Aggarwal, and T.S. Huang, "Community Detection with Edge Content in Social Media Networks," *Proc. IEEE 28th Int'l Conf. Data Eng.*, Apr. 2012.

[27] J. Scott, *Social Network Analysis: A Handbook*, second ed. SAGE Publications, 2000.

[28] H.-P. Tsai, D.-N. Yang, and M.-S. Chen, "Mining Group Movement Patterns for Tracking Moving Objects Efficiently," *IEEE Trans. Knowledge and Data Eng.*, vol. 23, no. 2, pp. 266-281, Feb. 2011.

[29] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E.J. Keogh, "Experimental Comparison of Representation Methods and Distance Measures for Time Series Data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275-309, Mar. 2013.

[30] X. Xiao, Y. Zheng, Q. Luo, and X. Xie, "Finding Similar Users using Category-Based Location History," *Proc. ACM 18th SIGSPATIAL Int'l Conf. Advances in Geographic Information Systems (GIS '10)*, Nov. 2010.

[31] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer, "Semitri: A Framework for Semantic Annotation of Heterogeneous Trajectories," *Proc. 14th Int'l Conf. Extending Database Technology (EDBT '1)*, Mar. 2011.

[32] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W.-Y. Ma, "Recommending Friends and Locations Based on Individual Location History," *ACM Trans. Web*, vol. 5, no. 1, article 5, Feb. 2011.

[33] Y. Zheng and X. Zhou, *Computing with Spatial Trajectories*, first ed. Springer, 2011.

Wen-Yuan Zhu received the BS degree in computer and information science from Aletheia University, Taiwan, in 2007 and the MS degree from the Department of Computer Science and Information Engineering, National Taiwan Normal University, Taiwan, in 2009. He is currently working toward the PhD degree in computer science at National Chiao Tung University. His research interests include trajectory pattern mining, game evaluation, and social computing.

Wen-Chih Peng received the BS and MS degrees from the National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the PhD degree in electrical engineering from the National Taiwan University, Taiwan, R.O.C, in 2001. Currently, he is an associate professor in the Department of Computer Science, National Chiao Tung University, Taiwan. Prior to joining the Department of Computer Science, National Chiao Tung University, he was mainly involved in the projects related to mobile computing, data broadcasting, and network data management. He serves as PC members in several prestigious conferences, such as IEEE International Conference on Data Engineering (ICDE), ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD), IEEE International Conference on Data Mining (ICDM) and ACM International Conference on Information and Knowledge Management (ACM CIKM). He is a co-organizer of the Second International Workshop on Privacy-Aware Location-based Mobile Services (PALMS) and is a guest editor of *Signal Processing* (special issue on Information Processing and Data Management in Wireless Sensor Networks). His research interests include mobile data management and data mining. He is a member of the IEEE.

Chih-Chieh Hung received the PhD degree in computer science from National Chiao Tung University, Taiwan. During his PhD program, his research efforts focused on trajectory data mining and applications, spatial-temporal databases, and sensor data management. After getting the PhD degree, he starting working in the e-commerce industry. He served as a research engineer in the E-Commerce Department at Yahoo! Taiwan Ltd. Currently, he is a data scientist in the Analyzing and Optimization Group, Big Data Department at Rakuten Inc., Japan. He serves as one of the organizers of the PAKDD workshop of Big Data Science and Engineering on E-commerce, 2014. His research interests include big data science, forecasting system, personalization, and recommendation system.

Po-Ruey Lei received the MS degree from the University of Southern California in 2000, and the PhD degree in electrical engineering from the National Defense University, Taiwan, in 2012. Currently, he is an assistant professor in the Department of Electrical Engineering, ROC Naval Academy, Taiwan. His research interests include trajectory data analysis, mobile data management, and data mining.

Ling-Jyh Chen received the BEd degree in information and computer education from National Taiwan Normal University in 1998, and the MS and PhD degrees in computer science from the University of California, Los Angeles, in 2002 and 2005, respectively. He joined the Institute of Information Science of Academia Sinica as an assistant research fellow in 2005, and became an associate research fellow in 2011. His research interests are networked sensing systems, network measurements, and mobile data management. He is a senior member of the IEEE.