Mining Trajectory Profiles for Discovering User Communities

Chih-Chieh Hung
National Chiao Tung University
Hsinchu, Taiwan, ROC
smalloshin@gmail.com

Chih-Wen Chang
National Chiao Tung University
Hsinchu, Taiwan, ROC
chiwen17@gmail.com

Wen-Chih Peng
National Chiao Tung University
Hsinchu, Taiwan, ROC
wcpeng@gmail.com

ABSTRACT

With the rapid development of positioning techniques (e.g., GPS), users can easily collect their trajectories. Furthermore, with the growing of Web 2.0, some web sites allow users to share their own trajectories. In such web sites, users are able to search trajectories that are interested by users. To provide more insights into these trajectories, in this paper, we target at the problem of discovering communities among users, where users in the same community have similar moving behaviors. Note that moving behaviors are usually represented as trajectory patterns where a user frequently travels. In this paper, we propose a framework to discover communities of users. Explicitly, we adopt a probabilistic suffix tree (abbreviated as PST) as a trajectory profile which truly reflects user moving behavior of a user. In light of trajectory profiles, we further formulate a similarity measurement among trajectory profiles of users. Based on the similarity measurement, we develop algorithm CI (standing for Community Identification) to discover user communities. Furthermore, for the same community, one representative PST is selected. When a new user is added, one could simply derive the similarity measurement by comparing representative PSTs, which is able to efficiently determine which community this new user should join. To evaluate our proposed methods, we conduct experiments on the synthetic dataset generated from one real dataset. Experimental results show that the trajectory profile proposed can effectively reflect user moving behavior, and our proposed methods can accurately identify communities among users.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: Data mining

General Terms

Management

Keywords

Trajectory profile, community structure and location-based service

ACM LBSN'09, November 3, 2009. Seattle, WA, USA

1. INTRODUCTION

Web-based social networks have attracted more and more research efforts in recent years. In particular, community mining is one of the major directions in social network analysis where a community can be simply defined as a group of objects sharing some common properties. Nowadays, with the rapid development of positioning techniques (e.g., GPS), one can easily collect his/her trajectories. Many GPS community sites are then established and users can easily share their trajectories [1][2]. Furthermore, with a large amount of trajectories shared, users expect to have one trajectory search or recommendation mechanism to rank these trajectories interested. For example, one would like to find some friends who have the same traveling interests, or one may want to know some interesting traveling paths different from his habits. As such, mining communities among users is helpful for trajectory recommendation or search.

In this paper, we target at the problem of mining communities among users, where a community refers to a set of users who have similar moving behaviors. Prior works have elaborated on mining trajectory patterns that capture moving behaviors [9][6][3]. Note that trajectory patterns usually have a huge number of patterns such that are not easily to formulate distance measurements of users. In other words, trajectory patterns mined are not well-organized for mining communities, let alone deriving the distance measurement of trajectory patterns. Therefore, to find the community in a location-based social network, one important issue is to build a *trajectory profile* for a user, where a trajectory profile is referred to as a data structure which can organize trajectory patterns for a user.

To address the issues above, in this paper, we adopt a probabilistic suffix tree (abbreviated as PST) to represent a trajectory profile of a user, which is shown to be able to accurately capture user moving behavior [15][18][16]. In light of the tree structure of the proposed trajectory profiles, the distance measurement is formulated for discovering communities. Explicitly, our design of mining communities consists of three steps: 1.) constructing trajectory profiles of users, 2.) formulating distance measurements among trajectory profiles and 3.) clustering similar trajectory profiles of users into groups. Since new users may be added, to efficiently identify which community to include these new users, for each community, we further select one representative PST. By comparing similarities with representative PSTs, a new user is able to quickly find which community that he/her should join. To evaluate the performance of the proposed approaches, we conduct comprehensive experiments and ex-

perimental results show that the trajectory profile proposed is able to accurately reflect user moving behaviors of users and is very helpful to mine communities of users. From our experiments, communities mined are indeed referred those users who have similar moving behaviors.

The contributions of this paper are summarized as follows:

1. A trajectory profile, represented as a probability suffix tree (PST), is first proposed. Trajectory profiles proposed can not only capture moving behaviors but also are utilized for discovering communities.

2. A distance measurement between two PSTs is derived. By exploring the concept of editing distances, the distance measurement can truly reflect how closeness between two PSTs.

3. According to the distance among PSTs, one clustering algorithm is developed to identify communities in which similar users are put together.

4. To efficiently identify which community for a new user, a representative PST of a community is determined.

The rest of the paper is organized as follows. In Section 2, several related works are discussed. In Section 3, our approach for mining community in a location-based social network is presented. Experimental results are shown in Section 4. Section 5 concludes with this paper.

2. RELATED WORKS

2.1 Community Mining

With the growth of the web, community mining are of importance recently. Generally speaking, a community is defined as a group of objects that share or have some common interests. As such, the community mining can be usually formulated as a graph problems, where a vertex is viewed as one object and edges represent interactions among objects. According to the purpose of mining communities, the determination of edges are different. Once a graph is generated, prior works could utilize some graph algorithms to derive communities. We mention in passing that the authors in [11] explore the concept of a bipartite graph to find the core of the community first. Then, the core is then expanded to obtain the final community. The maximum-flow and minimum-cut framework is also applied on the community mining [4]. The authority-and-hub idea was also used in the community mining [7]. Note that the above research efforts focus on mining communities from some Blogspaces or publication databases. In such spaces, the interactions among users are very clarified. For example, the citation between papers clearly exist among papers. In this paper, we intend to find community structures that have similar moving behaviors of users. The challenge issue is that how to formulate implicit common interests among trajectories of users, which is the very problem dealt in this paper.

2.2 Trajectory Pattern Mining

As pointed out early, moving behaviors of users are usually represented as trajectory patterns. The problem of mining trajectory patterns has attracted a considerable amount of research efforts. Generally speaking, the flow of mining trajectory patterns is to first find frequent regions and then derive the relationship between these frequent regions into trajectory patterns. Trajectory patterns are usually represented in terms of spatio-temporal association rules. In [3], the authors claimed the fuzziness of locations in patterns and developed algorithms to discover spatio-temporal sequential patterns. Furthermore, the authors in [10] proposed a clustering-based approach to discover moving regions within time intervals. In [9], the authors developed a hybrid prediction model, consisting of vector-based and pattern-based model, to predict movements of users. In [6] and [5], the authors exploited temporal annotated sequences in which sequences are associated with time information (i.e., transition times between two movements). As mentioned above, only few works focus on deriving user similarity based on trajectory patterns in terms of a sequence of stay points [12]. By comparing the geographic regions and the sequence of these regions being visited, two users are viewed to be related to each other if there are the longer sequence matched between two users' stay points. However, prior works only focus on mining trajectory patterns of a user. In fact, the number of trajectory patterns are huge. Thus, given two sets of trajectory patterns of two users, it is not easily to formulate the similarity measurement. Consequently, we propose a new data structure to represent user trajectory profile and utilize this new data structure to derive similarity measurements.

3. THE FRAMEWORK OF MINING COMMUNITIES

In this section, we develop a framework to discover user communities. The design of our framework consists of three steps:

Step 1. Constructing Profiles: In this step, frequent regions of trajectories are derived. Based on these frequent regions, we propose a probability suffix tree (abbreviated as PST) as trajectory profiles for users.

Step 2. Formulating Distance of Profiles: In this step, we derive the distance measurement of users in terms of their profiles (i.e., their PSTs).

Step 3. Identifying Community: According to distance values derived in Step 2, we develop a clustering algorithm to identify communities of users, where users have similar trajectory profiles are in the same community.

After generating communities of users, for each community, we judiciously select one representative PST tree. As pointed our early, for a new user, we only need to compare his/her trajectory profiles with representative PSTs. Furthermore, in our performance study, for each group, we use representative PST to predict movements of users in the same group. The selection of representation PSTs for each group is finally presented in Section 3.4.

3.1 Constructing Profiles

In this step, by given trajectories of users, we would like to construct their profiles (i.e., PSTs). By given all trajectories of users, as the same as ordinary trajectory pattern mining approaches, the frequent regions of trajectories are first derived. Then, by representing each trajectory into a sequence of frequent regions, a PST is then constructed for each user.

Many previous works for determining frequent regions in trajectory patterns adopts the density-based approach [9][6]. In the density-based approach, a region is viewed as a frequent region if the number of trajectories passing by is larger than a pre-defined threshold. Furthermore, if nearby regions are also frequent regions, these regions could merge into one

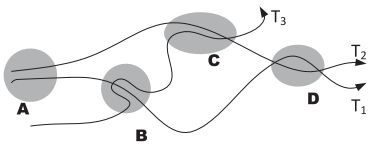

Figure 1: Frequent regions by the density-based approach.

larger region. For example, given three trajectories in Figure 1, four frequent regions A, B, C, and D are derived by the density-based approach in [9].

After deriving the frequent regions from trajectories of all users, trajectories can be transformed into a sequence of frequent region ids. For example, T_1 in Figure 1 can be represented as $< A, B, D >$. Such transformation can guarantee that the noise does not affect the procedure of trajectory pattern mining. Once transforming trajectories of each user, we can adopt a probabilistic suffix tree to capture moving behavior of each user. Specifically, each edge of a PST is labeled by a frequent region id that indicate one movement from one frequent region to the other one of a user. Each tree node is labeled by a string that shows a path from the node to the root. In other words, a *tree node* is labeled as $r_k...r_2r_1$ can be reached from the traversal path from $root \rightarrow r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_k$. In a PST, each tree node maintains a *conditional table* to record the appearing counts and the conditional probabilities of next frequent regions that follow the label of the tree node. For example, in Figure 2, the conditional table of tree node "AB" in T_1 shows that the conditional probability of next frequent region "A" after "AB" is 1. Clearly, tree nodes' labels show the frequently moving regions of a user and the corresponding conditional tables are used for predicting the next movements.

The construction of a PST is briefly described as follows. At the beginning, the PST has only one root node with the counts of each frequent region ids appearing in the trajectories. If the count of frequent region id r_i is larger than the predefined threshold (i.e., minimal support denoted as $MinSup$), one tree node labeled as r_i will be created as the child node of the root. Similarly, tree node r_i will maintain the occurrence count of r_i and the probability distribution table is also associated with the node to record the conditional probability of the frequent region id with the prefix segment r_i. For each frequent region sequence $r_1...r_{\ell-1}$, if a frequent region r_ℓ appears behind it, those statistical information of nodes labeled as all suffix of $r_1...r_{\ell-1}$ (i.e., counts and the conditional probabilities) should be updated accordingly. Interested readers are referred to [16][17] for the detailed procedure of constructing a PST.

3.2 Formulating Distance of Profiles

As mentioned above, a PST is a profile which represent trajectory patterns of each user. In this step, to determine distance of users in terms of their profile, we formulate a distance function to measure the distance of any two users in terms of their profiles.

To measure the distance between two PSTs, both the structure and the statistical information of a PST should be considered. Since the label of a node in a PST is the suffix of the label of its child nodes, a branch in a PST represents a series of frequent sequential patterns (represented as sequences of sensor ids). In other words, by extracting a branch in a PST, frequent sequential patterns with the same destination are determined. For example, the second branch in T_1 in Figure 2 is $root \rightarrow B \rightarrow AB$, which represents that the frequent sequential patterns with the destination as B are B and AB. Such frequent sequential patterns extracted from branches of a PST are referred to as structure information in a PST. As a result, the similarity of two PSTs should take the structure information of PSTs into consideration. For example, consider three PSTs T_1, T_2 and T_3 in Figure 2 as our example. It can be verified that the structure of T_1 is more similar to that of T_3 because T_1 and T_3 have the more common branches (i.e., $root \rightarrow A$ and $root \rightarrow B \rightarrow AB$) than T_2 does. Except the structure information of PSTs, we should further consider the statistical information of PSTs. Note that two users may have similar moving behaviors in terms of sequential patterns. However, their behaviors are different if the probabilities of these sequence patterns is considered. For example, in Figure 2, these two PSTs have the same branches but their probabilities are different. The left PST indicates that one user frequently stays A, whereas the right PST shows that one user frequently appears in area B. Therefore, the probabilities of nodes in a PST should also be considered. Through statistical information in PSTs, each node can determine its corresponding importance in terms of the probability that indicates how frequent one user travels along this frequent sequential pattern. Thus, when designing a distance function among PSTs, we should give higher weight to a node with higher probability. For example, consider a PST T_2 in Figure 2, we could obtain that the probability that the user stays in area C is 0.33. On the other hand, the probability that the user stays in area A and then stay in area C is $0.55 \times 0.66 \approx 0.36$ (i.e., $P(AC) = P(A) \times P(C|A)$). Thus, node AC will be given a higher weight than C when we design a distance function.

By exploiting PST tree structures and statistical information in conditional tables, a distance function δ_{MSL} is formulated. In order to capture structure similarity of PSTs, we transform a PST into a *moving sequence list*, in which each element is a moving sequence from the root node to a leaf node and elements are ordered from the left to the right of PSTs. A moving sequence of a PST is defined as follows:

Definition 1. Moving Sequence: Given a PST T_i, the j-th moving sequence is defined as $L_i^j = [TL_{i,j}^1{:}p(TL_{i,j}^1), TL_{i,j}^2{:}p(TL_{i,j}^2), ..., TL_{i,j}^\ell{:}p(TL_{i,j}^\ell)]$, where $TL_{i,j}^k$ denotes the k-th tree node traversing from the root in the j-th branch of the root node and $p(TL_{i,j}^k)$ is the corresponding probability.

For example, consider the second branch of a PST T_1 in Figure 2. Since the second branch of the root is $B \rightarrow AB$, we can obtain $L_1^2 = [B : 0.375, AB : 0.33]$. To measure the importance of a moving sequence, the weight of each moving sequence is defined as follows:

Definition 2. Weight of a Moving Sequence: The weight of a moving sequence L_i^j is formulated as $w(L_i^j) = \sum_{k=1}^{n} p(TL_{i,j}^k)$, where n is the number of elements in L_i^j.

For example, the weight of a moving sequence $[A : 0.5]$ is 0.5 and the weight of moving sequence $[B : 0.375, AB : 0.33]$

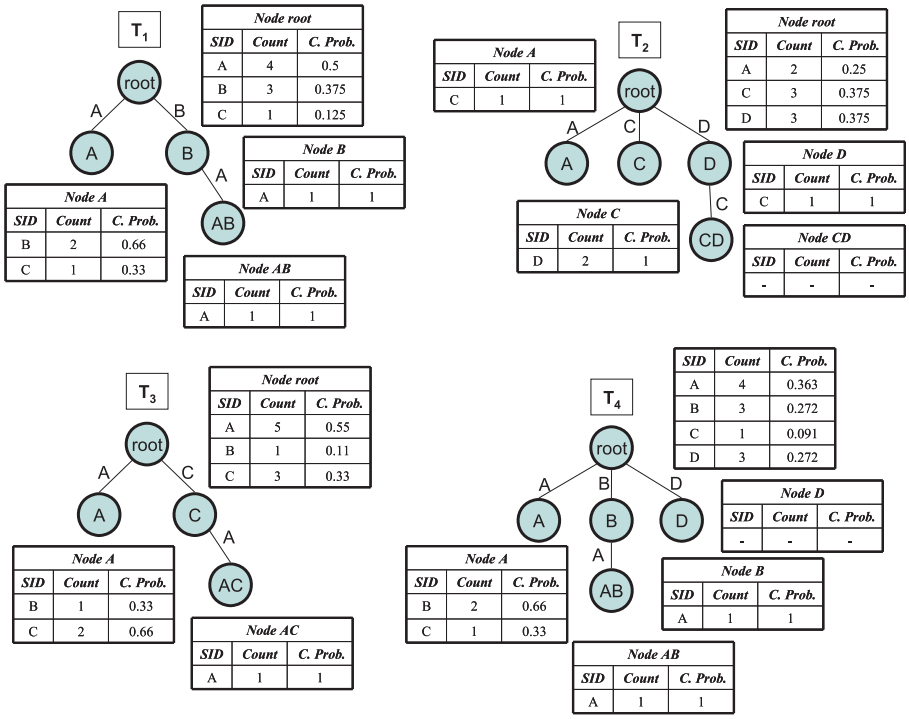

Figure 2: An example profile of PSTs.

is 0.705 (i.e., 0.375 + 0.33). Consequently, a moving sequence list is defined as follows:

Definition 3. Moving Sequence List: Given a PST T_i, the moving sequence list is defined as $L_i =< L_i^1, L_i^2, ..., L_i^n >$, where n denotes the number of moving sequences and L_i^j is the j-th moving sequence.

For example, the moving sequence list in T_1 is $< [A : 0.5], [B : 0.375, AB : 0.33] >$. Consequently, a moving sequence list derived from a PST is able to represent both the structure and the statistical information of a PST. As such, we propose a distance function δ_{MSL}. Given two PSTs T_i and T_j with their moving sequence lists L_i and L_j, the distance between two PSTs $\delta_{MSL}(T_i, T_j)$ is determined by the editing distance between L_i and L_j. The editing distance between L_i and L_j is determined as the minimal *cost* of transforming L_i into L_j via three editing operations (i.e., *insertion*, *deletion* and *replacement*). To facilitate the presentation, let $TL_{i,m}^{\ell}$ and $TL_{j,n}^{\ell}$ be a *pair*, $L_i^j[1..n]$ be L_i^j with n elements, and $L_i[1..m]$ be L_i with m moving sequences. Three operations and the corresponding costs are described as follows:

Insertion: We transform $L_i[1..m]$ to $L_j[1..n]$ by (1) transforming $L_i[1..m]$ to $L_j[1..n-1]$ and then (2) *inserting* one moving sequence L_j^n into $L_j[1..n-1]$. The corresponding cost is the sum of the cost of transforming $L_i[1..m]$ to $L_j[1..n-1]$ and $w(L_j^n)$.

Deletion: We transform $L_i[1..m]$ to $L_j[1..n]$ by (1) transforming $L_i[1..m-1]$ to $L_j[1..n]$ and (2) *deleting* one moving sequence L_i^m. The corresponding cost is the sum of the cost of transforming $L_i[1..m-1]$ to $L_j[1..n]$ and $w(L_i^m)$.

Replacement: We transform $L_i[1..m]$ to $L_j[1..n]$ by (1) transforming $L_i[1..m-1]$ to $L_j[1..n-1]$ and (2) *replacing* a moving sequence L_i^m to a moving sequence L_j^n. Specifically, to replace one moving sequence L_i^m as L_j^n, we compare nodes in two moving sequences pair by pair. If the labels of nodes in moving sequences are the same, the cost is estimated as the difference of their probability. Then, we keep comparing the next pair until the pairs of two moving sequences are not the same. The probabilities of remaining pairs are summed up as the cost. For example, suppose that two moving sequences $[C : 0.1, AC : 0.2, BAC : 0.3]$ and $[C : 0.3, BC : 0.5]$. Since the labels of the first pair are the same (i.e., C), the cost of this replacement is $|0.1-0.3| = 0.2$. On the other hand, since the labels of the second pairs are not the same (i.e., AC and BC), the cost for replacing $[AC : 0.2, BAC : 0.3]$ to $[BC : 0.5]$ is $0.2+0.3+0.5 = 1$ (i.e., AC:0.2, BAC:0.3 and BC:0.5). Therefore, the total cost for this replacement is $0.2 + 1 = 1.2$. Consequently, the cost is the sum of probabilities of all elements. The corresponding cost is the sum of the cost of transforming $L_i[1..m-1]$ to $L_j[1..n-1]$ and the cost for replacing L_i^m to L_j^n.

According to the above three operations, we can define the editing distance between two moving sequences lists. Assume that $R(\cdot)$ denotes the cost of replacement operation.

Definition 4. Editing Distance: Given two moving sequence lists L_i and L_j, the editing distance is defined as follows:

$$ed(L_i[1..m], L_j[1..n]) =$$
$$\min \begin{cases} ed(L_i[1..m], L_j[1..n-1]) + w(L_j^n) \\ ed(L_i[1..m-1], L_j[1..n]) + w(L_i^m) \\ ed(L_i[1..m-1], L_j[1..n-1]) + R(L_i^m, L_j^n) \end{cases}$$

where $R(L_i^m, L_j^n) =$

$$\begin{cases} R(L_i^m[2..k], L_j^n[2..\ell]) + |p(TL_{i,m}^1) - p(TL_{j,n}^1)|, \text{if } TL_{i,m}^1 = TL_{j,n}^1 \\ \sum_{c=1}^{k} p(TL_{i,m}^c) + \sum_{c=1}^{\ell} p(TL_{j,n}^c), \text{otherwise} \end{cases}$$

The boundary conditions are as follows:
$ed(L_i[0], L_j[1..n]) = \sum_{c=1}^{n} w(L_j^c)$,
and $ed(L_i[1..m], L_j[0]) = \sum_{c=1}^{m} w(L_i^c)$.

Thus, based on the editing distance function, given two PSTs T_i and T_j with their moving sequence list L_i and L_j, the distance function $\delta_{MSL}(T_i, T_j)$ is defined as $ed(L_i, L_j)$.

The distance function δ_{MSL} derives the distance of two PSTs by computing the editing distance between their moving sequence lists. By exploiting dynamic programming, Algorithm DP (**D**istance of **P**STs) is proposed to compute the editing distance between two moving sequence lists from the above recurrence relation in a bottom-up manner. Without loss of generality, given two PSTs T_1 and T_2, the table entry $c[i, j]$ represents the minimal cost of transforming $L_1[1..i]$ to $L_2[1..j]$. To compute the minimal cost in a bottom-up manner, the $c[0, j]$ and $c[i, 0]$ for all i and j are determined first and then compute the table entries row by row. For each table entry, three costs should be calculated according to three editing operations (i.e., insertion, deletion and replacement). Among three costs, each table entry only stores the minimal cost.

Algorithm 1 : Algorithm DP

Input: $L_i[1..m]$, and $L_j[1..n]$: two moving sequence lists
Output: $c[m, n]$, the distance of two moving sequence lists

```
1: for i = 0 to m do
2:    c[i, 0] ← Σ_{c=1}^{m} w(L_i^c);
3: for j = 0 to n do
4:    c[0, j] ← Σ_{c=1}^{n} w(L_j^c);
5: for i = 1 to m do
6:    for j = 1 to n do
7:       x = c[i, j − 1] + w(L_j^n);
8:       y = c[i − 1, j] + w(L_i^m);
9:       z = c[i − 1, j − 1] + R(L_i^m, L_j^n);
10:      c[i, j] ← min(x, y, z);
11:   end for
12: end for
```

For example, the distance between T_1 and T_3 can be determined by calculating $d(L_1[1..2], L_3[1..2])$. Note that our goal is to derive the minimal cost to transform $L_1[1..2]$ into $L_3[1..2]$ such that the value in $c[2, 2]$ is the distance $\delta_{MSL}(T_i, T_j)$. Table 3 shows the execution scenario for Algorithm DP. By taking $c[1, 2]$ as our example, we compare the costs of transforming $< [A : 0.5] >$ to $< [A : 0.55], [C : 0.33, AC : 0.36] >$ by insertion, deletion and replacement. Among these three costs, only the minimal one is kept in $c[1, 2]$. Explicitly, the cost for *insertion* is the sum of (1) $c[1, 1] = 0.05$: the cost transforming $< [A : 0.5] >$ to $< [A : 0.55] >$, and (2) the weight of L_2^2: 0.69 (i.e., 0.33 + 0.36). Thus, the total cost is 0.74. Then, the cost for *deletion* is the sum of (1) $c[0, 2] = 1.24$: the cost transforming an empty moving sequence $< [\,] >$ to $< [A : 0.55], [C : 0.33, AC : 0.36] >$, and (2) the weight of L_1^1: 0.5 (i.e., 0.33 + 0.36). Thus, the total cost is 1.74. Finally, the cost for replacement is the sum of (1) $c[0, 1] = 0.55$: the cost transforming an empty moving sequence $< [\,] >$ to $< [A : 0.55] >$, and (2) the cost of replacement of $[A : 0.5]$ and $[C : 0.33, AC : 0.36]$: 1.19 (i.e., 0.5 + 0.33 + 0.36). Thus, the total cost is 1.74. By taking the minimal cost between 0.74, 1.74 and 1.74, we can obtain the value of $c[1, 2]$ is 0.74. Following the same process, we can derive that $c[2.2]$ is 1.445. Therefore, $\delta_{MSL}(T_1, T_3) = 1.445$.

Consider three PSTs T_1, T_3 and T_4 in Figure 2 as our example. We derive that $\delta_{MSL}(T_1, T_3)$ is 1.445 and $\delta_{MSL}(T_1, T_4)$ is 0.602. By comparing tree structures of these three PSTs, it can be verified that T_1 is more similar to T_4 than T_3 since both T_1 and T_4 have sequences A and $A \rightarrow B$ where users frequently travel. However, T_1 and T_3 have only sequence A. Thus, T_1 is more similar to T_4, which is validated by $\delta_{MSL}(T_1, T_4) < \delta_{MSL}(T_1, T_3)$.

		[ ]	[(A:0.55)]	[(C:0.33), (AC:0.36)]
		0	1	2
[ ]	0	0	0.55	1.24
[(A:0.5)]	1	0.5	1.05 (i) 1.05 (d) 0.05 (r)	0.74 (i) 1.74 (d) 1.74 (r)
[(B:0.375), (AB:0.33)]	2	1.205	1.775 (i) 0.775 (d) 1.775 (r)	1.445 (i) 1.445 (d) 1.445 (r)

Figure 3: An execution scenario for the distance of T_1 and T_3. Only underlined values will be stored in this table.

3.3 Identifying Community

Once deriving the distance between profiles, the following task is to identify communities of users. The community identification problem can be formulated as follows:

Definition 5. Community Identification Problem: Let the distance function for PSTs be $\delta_{MSL}(\cdot)$. Given a set of users $\{U_1, ..., U_n\}$ with their profiles $\{T_1, ..., T_n\}$, the distance threshold δ, divide users into communities such that the number of communities is minimal and each U_i, U_j in the same community should satisfy $\delta(T_i, T_j) \leq \delta$.

To solve this problem, a graph can be first constructed where each vertex represents a user and there is an edge between two vertices if the distance of two PSTs are smaller than δ. Obviously, the community identification problem can be modeled as a minimal clique problem, where a graph is required to be decomposed into the minimal number of cliques. Algorithm CI (Community Identification) is then proposed. At first, vertices with lager degrees are selected to obtain larger cliques. Thus, from line 2 to line 3, we start to select vertex T_i with the highest degree in graph G. In line 4, those vertices adjacent to vertex T_i are put into list L. Then, the PSTs in list L form a graph together and recompute their node degrees (line 5). This step only calculates the degrees of vertices in list L. We construct a group C_i that contains the PST T_i and repeatedly select the PST T_j with the highest node degree in list L. Moreover, the PST T_j is included in group C_i if the PST T_j has edges with all PSTs in C_i (from line 7 to line 12). Finally, PSTs in group C_i are removed from graph G (line 14). Following the above operations, we could discover all cliques until all vertices are visited (i.e., G is empty).

3.4 Selecting Representative PSTs

After identifying communities, one representative PST, denoted as *r-PST*, is selected for each community. Then, we can use r-PST to represent the profile for all profiles in a community.

To select an r-PST, there are two factors to be considered: one is the size of an r-PST and the other is the distance be-

Algorithm 2 : Algorithm CI

Input: Users $\{U_1, ..., U_n\}$ with their profiles $\{T_1, ..., T_n\}$, and thresholds: δ
Output: C, communities of users

```
Construct a graph G by {T_1, ..., T_n} and δ;
while G is not empty do
    T_i ← the highest degree node in G;
    L ← PSTs adjacent to T_i;
    compute the node degree of the users in list L;
    construct a community C_i which contains T_i;
    while L is not empty do
        T_j ← the highest degree node in L and remove T_j from L;
        if T_j is adjacent to all the users in C_i then
            put T_j into group C_i;
        end if
    end while
    insert group C_i into set C;
    remove the PSTs in group C_i from graph G;
end while
```

Table 1: Tree size and error sum of three PSTs

PST	$N(T_i)$	Distance $\delta_{MST}(T_i, T_j)$	$ES(T_i)$
T_1	100	$\delta_{MST}(T_1, T_2) = 3.3$	7.1
		$\delta_{MST}(T_1, T_3) = 3.8$	
T_2	95	$\delta_{MST}(T_2, T_1) = 3.3$	6.4
		$\delta_{MST}(T_2, T_3) = 3.1$	
T_3	90	$\delta_{MST}(T_3, T_1) = 3.8$	6.9
		$\delta_{MST}(T_3, T_2) = 3.1$	

tween the selected r-PST and other PSTs. A r-PST with the smaller size can not reduce the overhead of storing profiles for a server but also provide better efficiency for query processing. For example, one can obtain the profile with different traveling behavior from him by accessing the profiles with larger distance values from the profile of his community. Thus, in this scenario, the smaller r-PSTs can speed up the computation for the distance values. Moreover, with a smaller distance between the r-PST and other PSTs in a group is, the more predication accuracy this r-PST could achieve. That is, this r-PST can best represent all other PSTs in the same community. Thus, a PST, which has a smaller tree size and is more similar to other PSTs in the same group, should be selected as r-PST.

For simplicity, the size of PST T_i is represented as the number of tree nodes, denoted as $N(T_i)$, and the error sum, denoted as ES, is used to quantify the distance between the r-PST and other PSTs in a group. Suppose that there are k PSTs in a group. The error sum of PST T_i is defined as the sum of the distance between T_i and other $k-1$ PSTs. In other words, $ES(T_i)$ is formulated as $ES(T_i) \sum_{j=1}^{k} \delta(T_i, T_j)$, where $\delta(T_i, T_i) = 0$. In order to take a balance between the tree size and the error sum, we can select the r-PST which can minimize $\alpha_i = \{\rho \times \frac{N(T_i)}{\sum_{j=1}^{k} N(T_j)} + (1-\rho) \times \frac{ES(T_i)}{\sum_{j=1}^{k} ES(T_j)}\}$, where k is the number of PSTs in a community and $0 \leq \rho \leq 1$

Obviously, a parameter ρ can be used to give different weights to tree size and error sum. For example, consider a community with three users (i.e., U_1, U_2 and U_3) and their PSTs (i.e., T_1, T_2 and T_3). The tree size and the error sum of each PST is listed in Table 1. ρ is set to 0.9 and we can compute that $\alpha_1 = 0.9 \times \frac{100}{285} + (1-0.9) \times \frac{7.1}{20.4}$. By the similar fashion, we can obtain the results in Table 2 when ρ varied. It can be seen that T_2 is selected as the r-PST if we respect tree size and the error sum equally and set the parameter ρ to 0.5. In the case of $\rho = 0.9$, T_3 is selected as the r-PST because its tree size is the smallest among three PSTs. On the contrary, in the case of $\rho = 0.1$, T_2 is selected as the r-PST because of the least error sum among three PSTs.

Table 2: Example of selecting an r-PST.

PST	$\rho = 0.9$	$\rho = 0.5$	$\rho = 0.1$
T_1	0.350	0.349	0.348
T_2	0.331	<u>0.323</u>	<u>0.315</u>
T_3	<u>0.318</u>	0.327	0.335

4. PERFORMANCE EVALUATION

In this section, extensive experiments are performed to evaluate the effectiveness and efficiency of our proposed methods.

4.1 Simulation Model

In our experiments, we use real trajectories from *CarWeb*, in which users record their own location every five seconds and upload their trajectories to CarWeb server [13]. Since CarWeb is our own platform, the ground truth of user moving behaviors is easily verified. To simulate the group mobility of users, for each group, one group pilot is generated and some followers in the same group will be simulated [8]. The movements of the followers in the same group are determined by three parameters: the variation probability p, the variation period ℓ and the variation radius r. For every variation period ℓ, a follower has the probability p to move away from the group pilot, where the moving distance is controlled by the variation radius r. The variation radius of each objects is uniformly distributed from 0 to max_{VR}. We select 24 real trajectories from CarWeb and to scale up the number of trajectories, each real trajectories is generated 10 times. Thus, there are total 240 trajectories, denoted as source trajectories. Each source trajectory is viewed as a group pilot and the number of followers is set to 10. As such, we could have 2400 trajectories for followers.

Once generating trajectories, we could further utilize DBSCAN algorithm to determine frequent hot regions. Then, each trajectory is transformed as a sequence of frequent hot regions. According to the transformed sequences, we implement our proposed approach of constructing trajectory profiles, denoted as PST. As pointed out early, for each group, we use representative PST to predict movements of users in the same group. For the comparison purpose, a general sequential pattern mining algorithm [14], denoted as GSP, is implemented for discovering sequential patterns. mining is implemented. In GSP, each user has a set of sequential patterns, which are viewed as trajectory patterns. Assume that the set of all sequential patterns is expressed by $sp_1, sp_2, ..., sp_n$, Clearly, the trajectory profile of a user is represented as a vector $\overrightarrow{v_i} = < x_1, ..., x_n >$ where $x_j = 1$ if $sp_j \in S_i$. Given a set of vectors, it is very common to exploit the cosine similarity measurement to derive the similarity matrix among users. According to the similarity ma-

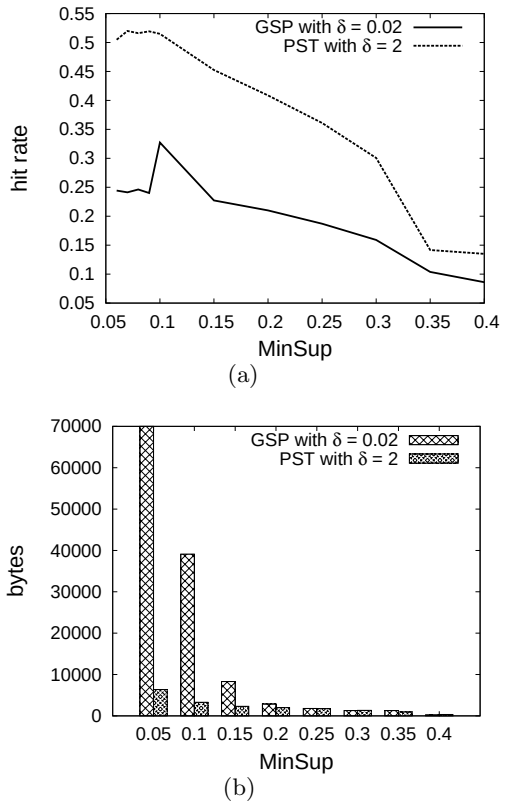

Figure 4: Comparison of GSP and our approach.

trix, we use the same clustering algorithm (i.e., algorithm CI) for extracting communities.

Four performance metrics (i.e., *hit rate*, *storage*, *entropy* and *purity*) are used. Since trajectory profiles capture moving behaviors of users, one could use trajectory profiles to predict the next movement of a user. Hence, the hit rate for a trajectory T is defined as $\frac{\text{total correct prediction}}{\text{number of points in T}}$. A higher hit rate represents that trajectory profiles can precisely capture the moving behaviors of users. The storage is the total size to store trajectory patterns. The storage is to compare how much storage space needed for two different trajectory profiles (i.e., PST and GSP). The entropy and purity are used measure the quality of clustering results, where the entropy is a function of the distribution of classes in the resulting clusters, and the purity is a function of the relative size of the largest class in the resulting clusters. Given a particular cluster, S_r, of size n_r, the entropy of this cluster is defined as $E(S_r) = -\frac{1}{\log(q)} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r}$, where q is the number of classes in the data set, and n_r^i is the number of PSTs of the ith class that were assigned to the rth cluster. The entropy of the entire clustering solution is then defined as $\sum_{r=1}^{k} \frac{n_r}{n} E(S_r)$. In general, the smaller the entropy value, the better the clustering solution is. Similarly, the purity of a cluster is defined as $P(S_r) = \frac{1}{n_r} \max_i(n_r^i)$. Thus, the overall purity of the clustering solution is defined as $\sum_{r=1}^{k} \frac{n_r}{n} P(S_r)$. The larger value of purity, the better the clustering solution is.

4.2 Impact of Trajectory Profiles

We first conduct experiments to show the advantage of our proposed trajectory profile (i.e., PST). For fair comparisons, the best parameters for GSP and PST are selected. Consequently, δ is set to 2 and 0.02 for PST and GSP, respectively.

Figure 4 shows the results with $MinSup$ varied, $MinSup$ is used to determine the frequentness in generating sequential patterns and probabilistic suffix trees. As can be seen in Figure 4(a), the hit rate of PST is much higher than that of GSP, which showing that PST could accurately capture moving behaviors of users. Interestingly, the hit rate of PST becomes a constant when $MinSup$ is smaller than 0.1. On the other hand, the hit rate of GSP has a peak when $MinSup = 0.1$ and decreases when $MinSup$ increases. The reason is that each source trajectory is used to generate 10 trajectories for a pilot such that the ground truth of $MinSup$ is almost 0.1. In Figure 4(b), it can be seen that the number of bytes of GSP are always larger than that of PST, especially in the case of $MinSup$ smaller than 0.15. As a result, PST could have better hit rates while using smaller storage size to store trajectory profiles, which demonstrates the well-organized structure in PST.

4.3 Sensitivity Analysis on Mining Communities

In this section, we examine the impact of the threshold $MinSup$ to the communities mined by our proposed methods. Figure 5 shows the experimental results with $MinSup$ and δ varied. First, consider the cases when the $MinSup$ is smaller than 0.15. Figure 5(a) shows that the hit rate is not sensitive to δ. However, as can be seen in Figure 5(b), a smaller δ results in larger storage spaces. This is because that the number of clusters when $\delta = 1$ are more than other cases, as shown in Figure 5(d). Note that the ground truth of the number of clusters should be 3 since there are only three kinds of pilots and followers. Even the number of clusters are more than the ground truth, from Figure 5(c), it can be seen that the entropy and purity are 0 and 1, respectively. That is, users that have similar trajectory profiles are divided into smaller clusters, where the users in the same cluster still have the similar moving behavior. On the other hand, consider the cases of $MinSup > 0.3$. Figure 5(a) shows that the hit rate significantly decreases when $MinSup$ is set to 0.3. Figure 5(d) shows the number of clusters is three when δ is 2, which is exactly the right result since we only have three groups in our trajectory dataset. However, Figure 5(c) shows that the quality of clusters decreases in terms of the entropy and the purity. From the above observations, we could conclude that the hit rate of PST decreases due to the quality of clusters is not good.

5. CONCLUSION

In this paper, we target at the problem of mining communities from trajectories of users. To achieve this goal, three main tasks should be done: 1.) constructing trajectory profiles of users, 2.) formulating the similarity measurements of users from their trajectory profiles, and 3.) clustering users with similar trajectory profiles. To deal with the above three tasks, we exploited a probabilistic suffix tree (abbreviated as PST) as a trajectory profile. Then, the similarity measurement of PSTs is judiciously formulated. Once we have the similarity measurement, one could derive the corresponding clustering algorithms for identifying communities. For

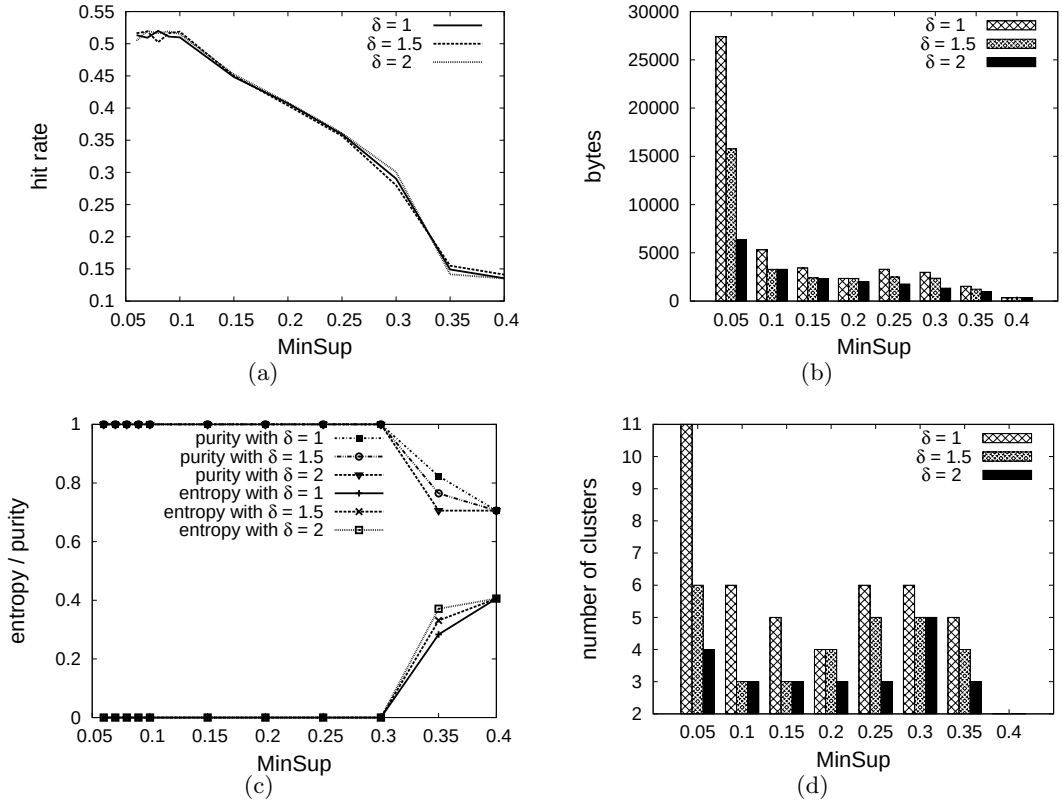

Figure 5: Sensitivity analysis when $MinSup$ varied.

each community, we further select one representative PST to represent the profile of trajectory patterns in the same cluster. Experimental results show that the trajectory profile proposed is able to effectively reflect user moving behavior. Moreover, according trajectory profiles, our proposed methods can accurately identify communities among users.

Acknowledgement

This work is supported in part by Microsoft Research Asia Internet Services Theme Research Program and by Taiwan MoE ATU Program.

6. REFERENCES

[1] EveryTrail - GPS Travel Community. *[available] http://www.everytrail.com/.*

[2] Run GPS Community Server. *[available] http://www.gps-sport.net/.*

[3] H. Cao, N. Mamoulis, and D. W. Cheung. Mining Frequent Spatio-Temporal Sequential Patterns. In *Proc. of ICDM*, 2005.

[4] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of Web communities. In *Proc. of KDD*, 2000.

[5] F. Giannotti, M. Nanni, and D. Pedreschi. Efficient Mining of Temporally Annotated Sequences. In *Proc. of SDM*, 2006.

[6] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory Pattern Mining. In *Proc. of KDD*, 2007.

[7] D. Gibson, J. M. Kleinberg, and P. Raghavan. Inferring Web Communities from Link Topology. In *Proc. of ACM Conference on Hypertext and Hypermedia*, 1998.

[8] J.-L. Huang, M.-S. Chen, and W.-C. Peng. Exploring Group Mobility for Replica Data Allocation in A Mobile Environment. In *Proc. of CIKM*, 2003.

[9] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A Hybrid Prediction Model for Moving Objects. In *Proc. of ICDE*, 2008.

[10] P. Kalnis, N. Mamoulis, and S. Bakiras. On Discovering Moving Clusters in Spatio-temporal Data. In *Proc. of SSTD*, 2005.

[11] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for Emerging Cyber-Communities. *Proc. of WWW*, 1999.

[12] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma. Mining User Similarity Based on Location History. In *Proc. of GIS*, 2008.

[13] C.-H. Lo, W.-C. Peng, C.-W. Chen, T.-Y. Lin, and C.-S. Lin. CarWeb: A Traffic Data Collection Platform. In *Proc. of MDM*, 2008.

[14] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In *Proc. of ICDE*, 2001.

[15] W.-C. Peng, Y.-Z. Ko, and W.-C. Lee. On Mining Moving Patterns for Object Tracking Sensor Networks. In *Proc. of MDM*, 2006.

[16] D. Ron, Y. Singer, and N. Tishby. The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length. *Machine Learning*, 25(2-3):117–149, 1996.

[17] H.-P. Tsai, D.-N. Yang, W.-C. Peng, and M.-S. Chen. Exploring Group Moving Pattern for an Energy-Constrained Object Tracking Sensor Network. In *Proc. of PAKDD*, 2007.

[18] J. Yang and W. Wang. Agile: A General Approach To Detect Transitions In Evovling Data Streams. In *Proc. of ICDM*, 2004.