Route Pattern Mining From Personal Trajectory Data*

MINGQI LV1, YINGLONG LI1, ZHENMING YUAN2 AND QIHUI WANG2

1College of Computer Science and Technology
Zhejiang University of Technology
Zhejiang, 310023 P.R. China
E-mail: mingqilv@zjut.edu.cn; liyinglong@zjut.edu.cn
2School of Information Science and Engineering
Hangzhou Normal University
Zhejiang, 311121 P.R. China
E-mail: zmyuan@hznu.edu.cn; wangqihui_xinyi@hotmail.com

The discovery of route patterns from trajectory data generated by moving objects is an essential problem for location-aware computing. However, the high degree of uncertainty of personal trajectory data significantly disturbs the existing route pattern mining approaches, and results in finding only short and incomplete patterns with high computational complexity. In this paper, we propose a personal trajectory data mining framework, which includes a group-and-partition trajectory abstraction technique and a frequent pattern mining algorithm called SCPM (Spatial Continuity based Pattern Mining). The group-and-partition technique can discover common sub-segments which are used to abstract the original trajectory data. The SCPM algorithm can efficiently derive longer and more complete route patterns from the abstracted personal trajectory data by tolerating various kinds of disturbances during the trips. Based on the real-world personal trajectory data, we conducted a number of experiments to evaluate the performance of our framework. The experimental results demonstrate that our framework is more efficient and effective as compared with the existing route pattern mining approaches, and the extracted route patterns can be effectively utilized to predict users’ future route.

Keywords: route pattern, personal trajectory data, data mining, future route prediction, trajectory abstraction

1. INTRODUCTION

The trajectory of a moving object can be represented by a sequence of spatial locations with timestamps. With the prevalence of the GPS-enabled devices, people can conveniently record their outdoor movements as GPS trajectory data. Since people’s daily routes usually show a high degree of temporal and spatial regularity [1], the problem of mining route patterns from trajectory data has attracted increasing interest recently. The discovered route patterns can be utilized by many kinds of location-aware applications, e.g. navigation system [2], recommendation system [3], transportation system [4], etc.

Many approaches have been proposed to solve the route pattern mining problem, and they generally consist of two essential steps, i.e. trajectory abstraction and frequent pattern mining. However, most existing approaches focused on analyzing the trajectory data of vehicles [4-10] rather than the trajectory data of individual people (i.e. personal trajectory data). The personal trajectory data are often of higher degree of uncertainty...

Received April 15, 2013; revised July 10 & August 14, 2013; accepted September 3, 2013.
Communicated by Wen-Chih Peng.
* This work was supported by the Natural Science Foundation of China (No. 61202282).
than that of vehicles due to various reasons, e.g. vehicles must travel over road network while people could take pathways which do not exist in the road network during a journey, people could temporarily interrupt a journey to visit some places, people could use various kinds of transportation during a single journey, etc. Therefore, mining techniques designed for vehicle trajectory data cannot adapt to the uncertainty of personal trajectory data, and applying them directly will result in finding only short and incomplete route patterns. We give two examples to illustrate the problems as follows.

**Example 1:** Space decomposition and segment clustering are the two most frequently used trajectory abstraction techniques. Consider the four trajectories in Fig. 1, space decomposition technique [11, 12] divides the area of interest into a number of regions, and abstracts the raw trajectories as region-id sequences. As shown in Fig. 1 (a), the four trajectories are abstracted as 

\[ T_1 = R_1R_3R_5R_7, \quad T_2 = R_2R_4R_6R_8, \quad T_3 = R_1R_3, \quad T_4 = R_2R_4. \]

However, the uncertainty of the personal trajectory data is likely to cause the answer loss problem, where the locations in two instances of a route pattern might not fall into the same region. For example, \( T_1 \) and \( T_2 \) share a large portion in common, but have totally different region-id sequences. On the other hand, Lee et al. [13] have proposed a segment clustering algorithm, which firstly partitions the raw trajectories into segments, and then clusters these segments to find common sub-trajectories. As shown in Fig. 1 (b), the four trajectories are abstracted as 

\[ T_1 = C_1C_2, \quad T_2 = C_1C_3, \quad T_3 = C_1C_5, \quad T_4 = C_1C_6. \]

Although the common sub-trajectory \( C_1 \) is discovered, it cannot well represent the movement behaviors of \( T_3 \) and \( T_4 \). A better solution is to further segment \( C_1 \) to three parts (as shown in Fig. 1 (c)), and thus the four trajectories could be abstracted as 

\[ T_1 = C_1C_2C_3C_4, \quad T_2 = C_1C_2C_3C_5, \quad T_3 = C_1C_5, \quad T_4 = C_1C_2C_6. \]

![Fig. 1. An illustration of the problems of the existing trajectory abstraction techniques.](image)

**Example 2:** Since a moving object should move contiguously in spatial space, existing approaches [14, 15] integrate an adjacency property (i.e. the adjacent two elements of a route pattern should also appear adjacent in the same order in the abstracted trajectories) into frequent pattern mining algorithms to extract reasonable route patterns. However, the adjacency property might not usually be satisfied due to the uncertainty of personal trajectory data. Given two abstracted trajectories in Fig. 2 (\( T_1 = C_1C_2C_3C_4C_5, \quad T_2 = C_1C_4C_5 \)), the longest route pattern that could be found by taking the adjacency property into account is \(<C_4, C_5>\) (if the minimum support of a route pattern is set to 2). However,
it can be clearly found from the figure that $C_2$ and $C_3$ are short pathways which temporarily deviate from the normal route. This temporary deviation could be caused by other disturbances in reality, e.g. obstacles on the road, temporarily visiting a place, GPS signal drift, etc. If we can relax the requirement of the adjacency property to tolerate these disturbances, a longer pattern $<C_1, C_4, C_5>$ should be discovered.

Aiming at these problems, this paper proposes a route pattern mining framework for personal trajectory data. The framework comprises two parts: trajectory abstraction and frequent pattern mining. It abstracts the raw personal GPS trajectory data by two steps. First, the GPS trajectory is preprocessed through clean, reconstruction and compression. Second, a group-and-partition approach is proposed to detect common sub-segments, which are then used to abstract the GPS trajectory. After trajectory abstraction, the framework extracts route patterns based on the SCPM algorithm, which takes into account the spatial continuity property of elements in route patterns to generate longer and more complete patterns. In summary, the contributions of this paper are as follows:

(1) We propose a route pattern mining framework, which is designed to adapt to the high degree of uncertainty of personal trajectory data.
(2) We present a trajectory abstraction technique, which uses a group-and-partition approach to detect common sub-segments.
(3) We present the SCPM algorithm, which considers the spatial continuity property of elements in route patterns to generate longer and more complete patterns.
(4) We evaluate the performance of our framework from three aspects (i.e. efficiency, completeness and usefulness) by using real datasets.

2. RELATED WORK

Our work is most related to the problem of pattern discovery from trajectory data. Although there are many different approaches to solve such problem, they generally consist of two essential parts, i.e. trajectory abstraction and frequent pattern mining.

According to whether a road network is available, current trajectory abstraction techniques can be categorized into two types, i.e. road network based approaches and geometric based approaches. Road network based approaches [4, 6, 16, 17] assume the journeys take place on city roads, and project trajectories onto road segments based on map matching techniques. However, road network based approaches are not suitable for personal trajectory data, because individual people usually take pathways which may not
exist in a road network. As geometric based approaches, the most commonly used techniques are space decomposition [11, 12, 18] and segment clustering [13]. Space decomposition divides the area of interest into a number of regions, and then a trajectory could be abstracted as a sequence of region-ids. However, the space decomposition technique may cause the answer loss problem, which means for two candidate instances of a pattern, the sampled locations may not fall into the same sequence of regions. As a result, we may miss some patterns. This problem can be alleviated by increasing the size of regions, but this will lead to missing the user movement information inside a region, and thus the extracted patterns will become coarser and less descriptive. On the other hand, segment clustering intends to detect similar portions of the trajectories. Since clustering trajectories as a whole could not detect the similar portions, the TRACLUS algorithm [13] partitions the trajectories into segments and groups these segments to find common sub-trajectories. However, the TRACLUS algorithm clusters segments as a whole, while different trajectories might share only part of segment in common.

Previous work on discovering frequent patterns from abstracted trajectories mostly applied association or sequential mining algorithms. For example, Bayir et al. [19] adopted association mining algorithm to find mobility patterns from abstracted cellular trajectory data. Gidófalvi et al. [18] proposed a database projection based method to extract long, sharable frequent patterns from multiple users’ trajectory data. However, association and sequential mining algorithms, which do not take into account the temporal and spatial continuity between adjacent elements of a route pattern, may generate candidate patterns which cannot exist as actual routes on one hand, and result in high computational complexity on the other hand. For this reason, Cao et al. [14] used a substring tree structure to extract route patterns which are represented by substrings. Lee et al. [15] proposed a graph-based mining algorithm which utilizes the adjacency property of route patterns to reduce the computational complexity. However, these approaches require that the adjacent two elements of a pattern should also appear strictly adjacent in the same order in the abstracted trajectory data. This requirement may often be disturbed by the high degree of uncertainty of personal trajectory data.

One of the most important applications of route pattern mining is to predict the future movement of a user. Most existing researches regarding future route prediction focused on vehicles. For example, Krumm and Horvitz [7] proposed a method called Predestination that leveraged a driver’s travel history to predict where the driver was going as a trip progressed. Karimi and Liu [17] used a digital map to abstract real trips, and predicted the subsequent movement of a user based on probabilities assigned to each road intersection. Simmons et al. [6] leveraged the road network and built a HMM to predict the destination and future route of a driver simultaneously. However, individual people’s current movement, which is likely to be more diverse than that of the vehicles, is more difficult to match with the historical movement regularities (i.e. route patterns).

3. TRAJECTORY ABSTRACTION

Since we cannot expect a person to visit exactly the same locations in every trajectory [14], it is almost impossible to find patterns from raw GPS trajectory data. For this problem, this section explains the personal GPS trajectory data abstraction procedure, which includes trajectory preprocessing and common sub-segment discovery.
3.1 Trajectory Preprocessing

The raw GPS trajectory data is preprocessed through three steps: clean (i.e. removing outliers), reconstruction (i.e. dividing the trajectory into discrete trips with definite origin and destination) and compression (i.e. compressing the point-based trips and transforming them to line-based trips).

For trajectory clean, a maximum speed based filtering method is adopted. The method checks each location with respect to its previous location of a trajectory, and calculates the corresponding instant speed. If it exceeds the maximum speed threshold $\lambda_{\text{speed}}$, this location is considered as an outlier and removed from the original trajectory.

For trajectory reconstruction, we divide the original trajectory in two conditions: the user stops at a place or the user turns off the data recording device. For the former condition, we use the visit point extraction algorithm proposed in [20] to detect visit points in a trajectory. Then, the trajectory is divided at each visit point (for a visit point, its first location becomes the destination of the last trip and its last location becomes the origin of the current trip). For the latter condition, we further divide an obtained trip when the time gap between two adjacent locations in the trip is greater than a threshold $\lambda_{\text{gap}}$.

For trajectory compression, we adopt a line simplification technique to convert the point-based trips to line-based trips. The purpose of trajectory compression is twofold. First, it is to reduce the data volume to facilitate the pattern mining procedure. Second, it is to make it possible to detect the common movement behaviors among different trips. As shown in Fig. 3 (a), the point-based trips contain a large number of non-repeated locations which make it almost impossible to discover any patterns. We employ the DP (Douglas-Peucker) algorithm for this problem. In brief, DP algorithm recursively detects point $p_k$ in a sequence of points $S$: $\{p_1, \ldots, p_n\}$ which is farthest from line $p_1p_n$, and decomposes $S$ into two point sequences $S_1$: $\{p_1, \ldots, p_k\}$ and $S_2$: $\{p_k, \ldots, p_n\}$, until that the perpendicular distance from every point in the decomposed point sequence $S_i$ to the line formed by the first and the last point of $S_i$ is less than a threshold $\lambda_{\text{distance}}$. The detected points are then used to simplify the original point sequence $S$.

The output of the trajectory preprocessing is a set of line-based trips, which are defined as LTSs (as shown in Fig. 3 (b)) as follow.

![Fig. 3. An illustration of the trajectory abstraction approach.](image)
**Definition 1:** A line temporal sequence (LTS) is a sequence of triples \((l_0, t_0^{\text{in}}, t_0^{\text{out}}), \ldots, (l_n, t_n^{\text{in}}, t_n^{\text{out}})\), in which \(l_i\) is a line used to simplify a subsequence of the point-based trip, \(t_i^{\text{in}}\) is the time when \(l_i\) starts to be visited, and \(t_i^{\text{out}}\) is the time when \(l_i\) ends to be visited (\(\forall 0 \leq k < n, t_k^{\text{in}} < t_k^{\text{out}} = t_{k+1}^{\text{in}} < t_{k+1}^{\text{out}}\)).

### 3.2 Common Sub-Segment Discovery

Since the lines also do not repeat themselves exactly in every LTS, we propose a group-and-partition approach to detect common movement behaviors which are represented by common sub-segments (i.e. sub-segment cluster) among all the LTSs. As discussed in Example 1, directly applying clustering algorithm to these lines may miss some common sub-segments. This problem is even serious when the historical trajectory data is not that adequate, since we may not find enough common movement behaviors for frequent pattern mining. Our alternative which goes through two phases (i.e. grouping phase and partitioning phase) intends to discover all the sub-segment clusters from the LTSs.

#### 3.2.1 Grouping phase

For the grouping phase, we propose a line clustering algorithm to identify all the candidate line clusters which can accommodate the common sub-segments. Before giving the details of the algorithm, we define the distance functions, including the perpendicular distance and the orientation distance, which are illustrated in Fig. 4.

![Fig. 4. An illustration of the distance functions.](image)

Given two lines \(L_1 = s_1e_1\) and \(L_2 = s_2e_2\), we assign a longer line to \(L_2\) and a shorter one to \(L_1\) without losing generality. The perpendicular distance between \(L_1\) and \(L_2\) is calculated using the distance function defined in [13] as Eq. (1), where \(d_{\perp}\) is the perpendicular distance from \(s_1\) to \(L_2\). The orientation distance between \(L_1\) and \(L_2\) is defined as their intersecting angle \(\theta\). We consider a line as a vector, and calculate the orientation distance using inner product operation as Eq. (2), where \(\|L_i\|\) is the length of \(L_i\).

\[
d_{\perp}(L_1, L_2) = \frac{d_{11}^2 + d_{12}^2}{d_{11} + d_{12}}
\]

\[
d_{\theta}(L_1, L_2) = \cos^{-1}\left(\frac{L_1 \cdot L_2}{\|L_1\| \|L_2\|}\right)
\]
Clustering algorithm is of high computational complexity, since all pairs of elements have to be compared multiple times. To expedite the process, we employ a heuristic line clustering algorithm, which is shown in Fig. 5. The algorithm firstly selects the longest line \( l_l \) in the line set \( LS \) as the initial seed. Then, several line clusters are formed through the filtering and clustering steps. The filtering step considers only the orientation distance, while the clustering step considers only the perpendicular distance. Next, the above mentioned process is repeated for the remaining lines that have not been put to any line clusters (lines 1-5). The reason of selecting the longest line as seed is that it can better capture users’ major movement behaviors. The filtering step could decompose the whole line set into a number of independent line sets to reduce the search space size for the clustering step. After filtering, we can apply existing clustering algorithms (e.g. DBSCAN, etc.) on each set of verified lines to obtain the line clusters (the parameter \( \lambda_{\text{perpendicular}} \) could be used as the clustering radius). As the example shown in Fig. 1 (b), five line clusters are formed after the grouping phase.

Algorithm 1: Heuristic Line Clustering

| INPUT: (1) A set of lines \( LS \); (2) Two parameters \( \lambda_{\text{perpendicular}} \) and \( \lambda_{\text{orientation}} \) |
| OUTPUT: A set of line clusters \( CS \) |
| 1. while \( LS \) is not empty do |
| 2. Find the longest line \( l_l \) in \( LS \) |
| 3. Verified lines \( VS = \text{FilterLines}(l_l, LS) \) |
| 4. Line clusters \( CCS = \text{Clustering}(VS, \lambda_{\text{perpendicular}}) \) |
| 5. Append all the line cluster of \( VCS \) to \( CS \) |

Procedure \( \text{FilterLines} \) (a line \( ll \), a set of lines \( LS \))

| 6. A set of lines \( VS = \emptyset \) |
| 7. for each line \( l_i \) in \( LS \) do |
| 8. if \( d_{\theta}(ll, l_i) < \lambda_{\text{orientation}} \) then |
| 9. Append \( l_i \) to \( VS \), and remove \( l_i \) from \( LS \) |
| 10. Return \( VS \) |

3.2.2 Partitioning phase

For the partitioning phase, we borrow the idea of sweep line approach [13] to generate sub-segment clusters. Fig. 6 gives an illustration of the approach. Given a line cluster with \( n \) lines \( LC = \{L_1, L_2, \ldots, L_n\} \), the partition algorithm works as follows.

(1) Compute the average direction vector \( V \) based on Eq. (3), where \( |V| \) is the cardinality of \( V \). This equation makes longer line contribute more to the average result.

(2) Generate vertical sweeping lines (i.e. \( l_1 \) to \( l_{10} \) in Fig. 6) in the direction of \( V \) at the starting and ending point of each line, and form a number of projection segments (i.e. \( s_1 \) to \( s_5 \) in Fig. 6) and projection points on \( V \).

(3) Detect the splitting points (i.e. \( p_1 \), \( p_2 \) and \( p_3 \) in Fig. 6) as follows: First, find all the major projection segments whose length is longer than a threshold \( \lambda_{\text{length}} \) (i.e. \( s_5 \) and \( s_7 \) in Fig. 6). Second, generate the midpoint of every two adjacent major projection segments on \( V \) (i.e. \( p_2 \) in Fig. 6). Finally, the splitting points include all the generated
midpoints, the first and the last projection points. 

(4) Generate an extra vertical sweeping line for each generated midpoint (i.e. \( ml_1 \) in Fig. 6), such that the sub-segments between the sweeping lines of two adjacent splitting points form a sub-segment cluster (i.e. sub-segment cluster 1 and sub-segment cluster 2 represented by the dotted rectangles in Fig. 6).

Fig. 6. An illustration of the sweep line approach for the partitioning phase.

After common sub-segment discovery procedure, the LTSs can be converted into a set of sub-segment cluster based trips, which are defined as CTSs (as shown in Fig. 3 (c)) as follow. In practice, a sub-segment cluster can be represented by its centroid line (i.e. the regression line of the starting and ending points of all the line segments within it).

**Definition 2:** A *cluster temporal sequence* (CTS) is a sequence of triples \(<(c_0, t_0^{\text{in}}, t_0^{\text{out}}), \ldots, (c_n, t_n^{\text{in}}, t_n^{\text{out}})>\), in which \( c_i \) is a sub-segment cluster used to abstract a subsequence of the original trip, \( t_i^{\text{in}} \) is the time when \( c_i \) starts to be visited, and \( t_i^{\text{out}} \) is the time when \( c_i \) (\( i = 0, \ldots, n \)) ends to be visited (\( \forall 0 \leq k < n, t_k^{\text{in}} < t_k^{\text{out}} \leq t_{k+1}^{\text{in}} < t_{k+1}^{\text{out}} \)).

\[
V = \frac{L_1 + L_2 + \cdots + L_n}{|\mathcal{P}|} \tag{3}
\]

### 4. FREQUENT PATTERN MINING

After trajectory abstraction, the original GPS trajectory data can be converted into a transactional database. In this database, each transaction is a sub-segment cluster, and a CTS is represented by a list of transactions ordered by time. Therefore, the frequent pattern mining can be viewed as a problem of finding ordered lists of transactions appearing with high frequency in the transactional database [21].

Unfortunately, frequent pattern mining techniques for transactional databases (e.g. association mining algorithms like Apriori [22], sequential mining algorithms like PrefixSpan [23], etc.) are not suitable for discovering patterns from CTSs. First, transactional pattern mining techniques do not take into account the temporal and spatial continuity between the elements in a route pattern, while actually a user should move contiguously in spatial space. Second, transactional pattern mining techniques tend to have high com-
putational complexity. To solve these problems, substring mining algorithm [14] or graph mining algorithm [15] which depends on the adjacency property of route patterns can be adopted. The adjacency property requires that the adjacent two elements of a route pattern should also appear adjacent in the same order in the original transactional databases, so that the search space size can be greatly reduced, and as a result, the computational complexity can be significantly reduced.

However, the adjacency property may not usually be satisfied due to the high degree of uncertainty of personal trajectory data (as shown in Example 2). Aiming at this problem, we define spatial continuity property instead of adjacency property for CTSs in Definition 3. Based on the spatial continuity property, a database projection algorithm SCPM which extends the PrefixSpan algorithm is proposed to allow the effective and efficient extraction of route patterns from the CTSs. As shown in Fig. 7, the algorithm firstly constructs a hash table to facilitate the sub-segment cluster query. The key of the hash table is the identifier of a given sub-segment cluster, and the value is a set of sub-segment clusters that satisfy the spatial continuity property with the given sub-segment cluster (line 1). Then, it calls the function ExtendProjection recursively to find all the prefixes based on the database projection approach. When the function is called for the first time, the parameter $cur_{projs}$ is constructed as the set of the original CTSs (lines 3-4). Finally, the route patterns are constructed based on the generated prefixes (lines 5-6).

**Definition 3:** Given two sub-segment clusters $C_1$ and $C_2$, we say that $C_2$ satisfies the spatial continuity property with $C_1$, if the distance between the ending point of $C_1$ (i.e. the ending point of the centroid line of $C_1$) and the starting point of $C_2$ (i.e. the starting point of the centroid line of $C_2$) is less than a threshold $\lambda_{radius}$.

**Algorithm 2: Spatial Continuity based Pattern Mining**

**INPUT:** (1) A set of cluster temporal sequences $CTSS$; (2) A set of all sub-segment clusters $CS$; (3) Two parameters $\lambda_{radius}$ and $min_{sup}$

**OUTPUT:** A set of route patterns $PS$

1. Construct a hash table $continuity\_map$ for $CS$ according to $\lambda_{radius}$
2. prefix $all\_prefixes = \emptyset$
3. Initialize the set of projections $init\_projs$ based on $CTSS$
4. $ExtendProjection(\emptyset, init\_projs)$
5. for each prefix $prefix_i$ in $all\_prefixes$ do
6. Generate route pattern $p_i$ based on $prefix_i$, and append $p_i$ to $PS$

Procedure $ExtendProjection$ (current prefix $cur\_prefix$, current projections $cur\_projs$)

7. frequent singular items $candidate\_items = \emptyset$
8. if $cur\_prefix = \emptyset$ then
9. for each cluster $c_i$ in $CS$ do
10. if $c_i$ appears in more than $min\_sup$ projections in $cur\_projs$ then
11. Append $c_i$ to $candidate\_items$
12. else
13. Get sub-segment cluster $sc$ corresponding to the last item in $cur\_prefix$
14. A set of sub-segment clusters $continuity\_clusters = continuity\_map[sc]$
15. for each cluster $c_i$ in $continuity\_clusters$ do
16. if $c_i$ appears in more than $min\_sup$ projections in $cur\_projs$ then
17. Append $c_i$ to $candidate\_items$
18. for each item itemi in candidate_items do
19. Concatenate cur_prefix and itemi to generate a new prefix new_prefix
20. Append new_prefix to all_prefixes
21. new projections new_projs = Ø
22. for each proj in cur_projs do
23. Get the subsequence of proj from itemi to the end as the new projection new_proj
24. Append new_proj to new_projs
25. if new_projs is not empty then
26. ExtendProjection(new_prefix, new_projs)

Fig. 7. The SCPM algorithm for route pattern mining.

In the ExtendProjection procedure, we firstly find all the frequent singular items (whose support is larger than min_sup) from the current projections. If the procedure is called for the first time, all the sub-segment clusters are checked as candidate items (lines 8-11). Otherwise, only the sub-segment clusters which satisfy the spatial continuity property with the sub-segment cluster corresponding to the last item in the current prefix are considered (lines 12-17), so the search space size can be significantly reduced. The algorithm is hereafter similar to PrefixSpan. The function calls itself recursively until no new projections are generated.

Apparently, the time complexity of the SCPM algorithm depends on the value of $\lambda_{radius}$. When $\lambda_{radius} \to 0$, the SCPM algorithm has the same time complexity as the substring mining algorithm (the time complexity becomes the best case). When $\lambda_{radius} \to +\infty$, the SCPM algorithm has the same time complexity as the PrefixSpan algorithm (the time complexity becomes the worst case).

In order to make the SCPM algorithm more understandable, we give an example execution over the two CTSs ($T_1 = C_1C_2C_3C_4C_5$ and $T_2 = C_1C_4C_5$) shown in Fig. 8 (the same case as Example 1). Let the minimum support of a route pattern be 2, the process of prefixes generation is partially illustrated in Table 1. Initially, we find three frequent singular items (i.e. $C_1$, $C_4$ and $C_5$) from $T_1$ and $T_2$. Taking $C_1$ as the initial prefix, there are three sub-segment clusters (i.e. $C_2$, $C_3$ and $C_4$) satisfy the spatial continuity property with it, because $p_1$, $p_2$ and $p_3$ (the starting points of the centroid lines of $C_2$, $C_3$ and $C_4$ respectively) are within the range of $\lambda_{radius}$ with respect to $p_0$ (the ending point of the centroid line of $C_1$). Among $C_2$, $C_3$ and $C_4$, only one frequent singular item (i.e. $C_4$) can be detected from the generated projections (i.e. $C_2C_3C_4C_5$ and $C_4C_5$). Thus, a new prefix $C_1C_4$ can be generated by concatenating the former
prefix $C_1$ and the frequent singular item $C_4$, and $C_4$ is treated as the last item of the new prefix. The above mentioned process is repeated and three prefixes (i.e. $C_1$, $C_1C_4$, and $C_1C_4C_5$) can be generated. These prefixes can be judged as route patterns.

Table 1. An example of the process of the SCPM algorithm.

<table>
<thead>
<tr>
<th>Current Prefix</th>
<th>Spatial Continuity Clusters</th>
<th>Projections</th>
<th>Frequent Singular Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>Null</td>
<td>$C_1C_2C_3C_4C_5$</td>
<td>$C_1$, $C_4$, $C_5$</td>
</tr>
<tr>
<td>$C_1$</td>
<td>$C_2$, $C_3$, $C_4$</td>
<td>$C_2C_3C_4C_5$</td>
<td>$C_4$</td>
</tr>
<tr>
<td>$C_1C_4$</td>
<td>$C_5$</td>
<td>$C_5$</td>
<td>$C_5$</td>
</tr>
<tr>
<td>$C_1C_4C_5$</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
</tr>
</tbody>
</table>

5. EXPERIMENT

In order to evaluate the performance of the proposed framework, we conducted a number of experiments based on real-world personal trajectory data collected from 10 participants. To facilitate the personal trajectory data collection, we chose mobile phones (Nokia N70 and Samsung GT-I9000 were used in our experiment) which are by far the most prevalent mobile devices as the data recording platforms. A program running on the mobile phone could connect to either the internal or external GPS receivers (HOLUX 1000 GPS), and record GPS points at 1Hz.

All participants were instructed to carry out the experiment in an open-ended way to make the recorded trajectory data reflect their daily lives as truly as possible, i.e. we did not set up any constraints on the data recording environment, and the participants could take the recording devices during their daily lives for arbitrary trips, e.g. going to work, going for shopping, going for a drive, etc. Finally, 695708 GPS points and 511 trips (after trajectory cleaning and segmentation) are collected from the 10 participants for more than one month.

5.1 Evaluation Approach

The performance of our personal trajectory data mining framework was evaluated from three aspects, i.e. efficiency, completeness, and usefulness. To carry out the evaluation, we proposed a set of metrics as follows.

(1) Efficiency

An algorithm is efficient if it runs quickly on a large amount of data. Therefore, we directly look at the execution time as evaluation metric.

(2) Completeness

We say that a route pattern mining algorithm is of high completeness if the route patterns it discovers can cover more parts of the original trips with fewer instances. Thus, we define the concept of Maximum Pattern Coverage (MPC) as follow. MPC reflects the relative length of the route patterns with respect to the real trips. We take it as evaluation
metric, and calculate the average MPC (i.e. AMPC) for all trips to evaluate the completeness of the route pattern mining algorithms.

**Definition 4:** Given a set of trips and the route patterns extracted from them, the MPC for a specific trip $T$ is defined as the ratio of the length of the longest route pattern (i.e. the sum of the length of the centroid lines of the involved sub-segment clusters) which is supported by $T$ over the length of $T$.

(3) Usefulness

One of the most important applications of route patterns is to predict the future movement of users. Thus, we take the ability of predicting future route based on the extracted route patterns as evaluation metric. We employed the algorithm in [12] as the future route prediction algorithm.

In order to evaluate the future route prediction ability, we measure the similarity of the predictive future route and the real future route based on a variant of the Hausdorff distance. As illustrated in Fig. 9 (a), the distance from $polyline_1$ to $polyline_2$ is calculated by averaging the shortest distances from all the points of $polyline_1$ to $polyline_2$ (the average distance of $d_1$ to $d_6$ in the example). The distance reflects the average deviation of $polyline_1$ from $polyline_2$, and a small distance indicates a high degree of similarity between $polyline_1$ and a subsequence of $polyline_2$. Apparently, this distance metric is asymmetric. In our experiment, the distance from predicted route to real route (we call it *precision distance*) and that from real route to predicted route (we call it *recall distance*) hold different meanings. Precision distance reflects how accurately the predicted route is aligned with the real route (i.e. prediction accuracy), and recall distance shows to what extent the real route could be accurately predicted (i.e. prediction completeness). Fig. 9 (b) gives an example that the predicted route is well aligned with the real route (i.e. the precision distance is small, and thus the prediction accuracy is high), but the predicted route is not long enough as compared with the real route. Fig. 9 (c) gives an opposite example that the real route is completely predicted (i.e. the recall distance is small, and thus the prediction completeness is high), but there exists a phenomenon of over prediction which result in partially incorrect future route.

All the experiments were performed on a PC with a Pentium Dual-Core CPU (E5300 @ 2.60 GHz, 2.59 GHz) and 2 GB main memory, running on Microsoft Windows XP Professional. All the methods were implemented in Python 3.2.2.

---

Fig. 9. An illustration of the evaluation metric for future route prediction: (a) a variant of the Hausdorff distance; (b) future route prediction with high precision accuracy; (c) future route prediction with high prediction completeness.
5.2 Efficiency Evaluation

The key parameter that influences the efficiency of the SCPM algorithm is the spatial continuity threshold $\lambda_{radius}$ (see Section 4). As the definition of $\lambda_{radius}$, with increasing $\lambda_{radius}$, more disturbances in the trips can be tolerated (so that longer and more complete route patterns could be extracted), while the search space size will expand (so that the SCPM algorithm requires more execution time to finish). We conducted experiments to make a counterbalance between algorithm completeness and efficiency by varying the value of $\lambda_{radius}$. While adjusting $\lambda_{radius}$, other parameters of the framework are set as their default values (i.e. $\lambda_{speed} = 30$m/s, $\lambda_{gap} = 1800$s, $\lambda_{distance} = 50$m, $\lambda_{perpendicular} = 50$m, $\lambda_{orientation} = \pi/12$, $\lambda_{length} = 100$m, $\min_supp = 3$).

Our route pattern mining framework is geared to the need of individual people. Therefore, we utilized the trajectory data of one of the experiment participants who collected data for the longest period of time (for more than three months) for efficiency evaluation. The dataset involves nearly 150 trips and each trip contains approximately 20 sub-segment clusters in average after trajectory abstraction. Fig. 10 shows the execution time and AMPC value of SCPM algorithm versus the spatial continuity threshold $\lambda_{radius}$. As $\lambda_{radius}$ increases, the execution time of the SCPM algorithm increases because the number of frequent singular items which satisfy the spatial continuity property with the last item of the previous prefix increases (so the number of recursion of the SCPM algorithm on new generated projections increases substantially). The execution time of SCPM algorithm remains under one second when $\lambda_{radius} \leq 100$ meters, and increases to over 100 seconds when $\lambda_{radius} \geq 300$ meters. On the other hand, the value of AMPC also increases with increasing $\lambda_{radius}$. When $\lambda_{radius}$ increases from 10 to 150 meters, the value of AMPC increases significantly (from 25.39% to 76.46%). However, the value of AMPC tends to become stable with further increase of $\lambda_{radius}$ (AMPC = 81.89% when $\lambda_{radius} = 300$m). Therefore, we can set $\lambda_{radius}$ to 150 meters to make a good counterbalance between algorithm efficiency and completeness (execution time is 1.41 seconds and AMPC is 76.46%) for a single participant.

![Fig. 10. Effect of the spatial continuity threshold on: (a) the execution time; (b) the AMPC value.](image)

5.3 Completeness Evaluation

We conducted experiments to evaluate the completeness of different trajectory ab-
straction techniques and frequent pattern mining algorithms. As summarized in Table 2, three kinds of trajectory abstraction techniques (i.e. GaP, LC and SP) and two kinds of frequent pattern mining algorithms (i.e. SPCM and SSM) were evaluated. The spatial continuity threshold $\lambda_{\text{radius}}$ is set to 150 meters in the experiments. First, we tried to evaluate the influence on the AMPC metric of different trajectory abstraction techniques, while using the same frequent pattern mining algorithm (i.e. SPCM). Thus, three compositions of approaches (i.e. GaP + SPCM, LC + SPCM and SP + SPCM) were evaluated. Fig. 11(a) shows the AMPC values of all the participants by using different compositions of approaches. From the figure, it can be found that the AMPC values of the line-based trajectory abstraction techniques (i.e. GaP and LC) are much higher than that of the region-based trajectory abstraction technique (i.e. SP). This result indicates that personal trips can be better abstracted for route pattern mining based on polylines rather than sequences of regions. Besides, the proposed trajectory abstraction technique (i.e. GaP) has the highest AMPC value (except for participant 9). This means that by creating subsegment clusters based on the group-and-partition approach, the SPCM algorithm can discover longer and more complete route patterns from the same personal trajectory data.

Second, we tried to evaluate the effect on the AMPC metric of different frequent pattern mining algorithms, while using the same trajectory abstraction technique (i.e. GaP). Therefore, two compositions of approaches (i.e. GaP + SPCM, GaP + SSM) were evaluated. The experimental result is presented in Fig. 11 (b). As expected, the SPCM algorithm obviously outperforms the substring mining algorithm. This is because that the SPCM algorithm can tolerate the disturbances in personal trajectory data to derive longer and more complete route patterns by adjusting $\lambda_{\text{radius}}$, while substring mining algorithm requires that the route patterns must satisfy the strict adjacency property.

![Fig. 11. The AMPC value of all participants by using: (a) different trajectory abstraction techniques; (b) different frequent pattern mining algorithms.](image-url)
5.4 Usefulness Evaluation

As mentioned in Section 5.1, we evaluated the usefulness of the extracted route patterns by exploring how well they could be utilized for future route prediction by using the prediction algorithm proposed in [12]. Given the extracted route patterns and the current trip, the prediction algorithm works in three steps, i.e., building prediction tree based on the extracted route patterns, selecting the candidate route patterns based on pattern matching, and predicting future route based on a probabilistic model given the candidate route patterns. One thing that needs to be elaborated is that the prediction algorithm in [12] uses a substring based route pattern matching method, i.e., both the extracted route patterns and the current trip are represented by sequences of region-ids, and the candidate route patterns are selected by matching the suffix of the current trip and the prefix of every route pattern. Thus, the prediction algorithm is executed over abstracted trips.

For each participant, we used 10-fold cross validation where the route patterns are extracted from 90% of the abstracted trips, and the other 10% is used for test. This procedure is repeated 10 times and the average performance is reported. Given the current trip, we tried to evaluate the capability of continuous future route prediction, i.e., predicting as long as possible the future route that may be traveled by the user. The current trip and predicted future route are both represented by a sequence of trajectory abstraction elements (e.g., sub-segment clusters, regions, etc.). To establish the test cases, each abstracted trip in the testing set was divided into two parts, i.e., the current trip and the real future route. For example, an abstracted trip \( T = C_1C_2C_3 \) can produce two test cases, because we can divide it into \( <C_1> \) (i.e., current trip) and \( <C_2, C_3> \) (i.e., real future route), or \( <C_1, C_2> \) and \( <C_3> \). Moreover, a test case is of less value if the real future route is too short (since it is inadequate for evaluation), so only the test cases the length of whose real future route accounts for over 60% of the length of the whole trip were used in the experiments. Thus, the test case \( <C_1> \) and \( <C_2, C_3> \) is remained in the example.

Fig. 12 (a) compares the precision distance of future route prediction based on the route patterns extracted by using the four different compositions of approaches (i.e., \( GaP + SCPM \), \( LC + SCPM \), \( GaP + SSM \) and \( SP + SCPM \)). It can be found from the figure that the \( SP + SCPM \) approach has much shorter precision distance (thus much higher performance on prediction accuracy) than that of the other three approaches. However, based on the analysis of the test cases, we found that a trip segment is decomposed into many fine-grained regions by using the space decomposition technique, and thus a number of test cases could be produced by splitting a single straight trip segment. Obviously, given even a short part of the trip segment as current trip, the rest part of the same trip segment without forked paths could be easily predicted with high accuracy. While on the other hand, line simplification based trajectory abstraction techniques often produce test cases by splitting trips at intersections (because a straight trip segment is always abstracted by a single sub-segment cluster which cannot be split). Moreover, the \( GaP + SCPM \) approach has slightly higher performance on prediction accuracy than the other two compositions of approaches (i.e., \( LC + SCPM \) and \( GaP + SSM \)).

No matter how accurately the future route is predicted, the result is useless if the predicted future route is too short. Therefore, we also evaluated the prediction completeness measured using the recall distance metric based on the route patterns extracted by using the above mentioned four compositions of approaches. As shown in Fig. 12 (b),
although the $SP + SPCM$ approach has the best performance on prediction accuracy, its capability to predict longer future route is much weaker (i.e. the recall distance is much longer). The proposed approach (i.e. $GaP + SPCM$) has much higher performance on prediction completeness as compared with the other three compositions of approaches. This is mainly because that the proposed approach can discover longer and more complete route patterns by tolerating the disturbances of personal trajectory data based on the group-and-partition trajectory abstraction technique (as compared with the space decomposition based trajectory abstraction technique and the simple line clustering based trajectory abstraction technique) and the spatial continuity based pattern mining algorithm (as compared with the substring pattern mining algorithm).

Fig. 12. Future route prediction performance of all participants by using different compositions of approaches: (a) precision distance; (b) recall distance.

6. CONCLUSIONS

In this paper, we propose a route pattern mining framework. The main advantage is that it can tolerate various kinds of disturbances in personal trajectory data. The experimental results demonstrated that the proposed framework is both efficient and effective.

REFERENCES

5. J. Froehlich and J. Krumm, “Route prediction from trip observations,” in *Society of*


Mingqi Lv (吕明琪) received the Ph.D. degree in Computer Science and Technology from Zhejiang University, China. He is currently a Lecturer in College of Computer Science and Technology, Zhejiang University of Technology, China. His research interests include ubiquitous computing and data mining.

Yinglong Li (李英龙) is currently a Lecturer in College of Computer Science and Technology, Zhejiang University of Technology, China. His research interest is data processing in sensor network.

Zhenming Yuan (袁贞明) received the Ph.D. degree in Computer Science and Technology from Zhejiang University, China. He is now a Professor in School of Information Science and Engineering, Hangzhou Normal University, Hangzhou, China. His research interests include multimedia computing, machine learning and information retrieval.

Qihui Wang (王琦晖) received the M.S. degree in School of Computer Science and Technology at Zhejiang University, China. He is a Ph.D. candidate in the School of Aeronautics and Astronautics at Zhejiang University. He is currently a Lecturer in Qianjiang College of Hangzhou Normal University. His main research interests include computer vision, computer graphics and image processing.