Mining and clustering mobility evolution patterns from social media for urban informatics

Chien-Cheng CHEN
National Chiao Tung University

Meng-Fen CHIANG
Singapore Management University, mfchiang@smu.edu.sg

Wen-Chih PENG
National Chiao Tung University

Citation

Mining and clustering mobility evolution patterns from social media for urban informatics

Chien-Cheng Chen[1] · **Meng-Fen Chiang[2]** ·
Wen-Chih Peng[1]

Abstract In this paper, given a set of check-in data, we aim at discovering representative daily movement behavior of users in a city. For example, daily movement behavior on a weekday may show users moving from one to another spatial region associated with time information. Since check-in data contain both spatial and temporal information, we propose a mobility evolution pattern to capture the daily movement behavior of users in a city. Furthermore, given a set of daily mobility evolution patterns, we formulate their similarity distances and then discover representative mobility evolution patterns via the clustering process. Representative mobility evolution patterns are able to infer major movement behavior in a city, which could bring some valuable knowledge for urban planning. Specifically, mobility evolution patterns consist of segments with the spatial region distribution and the corresponding time interval. To measure good segmentation from a set of check-in data, we formulate the problem of mining evolution patterns as a compression problem. In particular, we compute the representation length of the patterns based on the Minimum Description Length principle. Since the number of daily mobility evolution patterns is huge, we further cluster the daily mobility evolution patterns into groups and discover representative patterns. Note that we use the concept of locality-sensitive hashing to accelerate the cluster performance. To evaluate our proposed algorithms, we conducted experiments on the Gowalla and Brightkite datasets, and the experimental results show the effectiveness and efficiency of our proposed algorithms.

Keywords Mobility pattern · Data mining · Pattern clustering · Urban planning

1 Introduction

With the development of mobile networks and the popularity of smart phones, it is easy for users to share geographic information in location-based services. For example, users could check in their visiting places and share them with their friends via Foursquare or Facebook. They can also upload photos with GPS information to Flickr. Note that check-in data and geo-related photos can reflect users' movement behavior. By analyzing these check-in data, one could understand how and when people move around in a city. An understanding of the moving behavior of users would be helpful in urban planning [18] and for mobile advertisement services. Moreover, a city government can understand better how the inhabitants move and how to plan city transportation.

With geographic data, investigating the mobility of users has been widely studied. For example, prior works in [14,15,21] have proposed some algorithms to mine user moving patterns. The authors in [12,28,30] proposed methods of location recommendation. Most of these works focus on exploring the movement patterns of users. Movement patterns refer to frequent sequences of regions where regions are spatial areas in which most users frequently stay. Note that those movement patterns only reflect spatial regions of users and sequential relationships among regions. We claim that mobility behaviors of users should indicate not only spatial regions but also time information. In addition, with the popularity of location-based social networks, a considerable amount of check-in data is generated every day. Thus, in this paper, we aim at mining mobility evolution patterns that capture daily movement behaviors of users in a city. Mobility evolution patterns indicate where and when users stay in a city and how users move around a city within a day. Moreover, given a set of daily mobility evolution patterns, we further cluster mobility evolution patterns to extract representative patterns. Such patterns are more concise and represent major daily mobility evolution patterns.

Since mobility evolution patterns consist of segments with spatial region distribution and the corresponding time interval, one naive way is to order the check-in records by their timestamp. Then, we can get a sequence of check-in records in the spatio-temporal domain. However, this approach is too detailed for mobility evolution patterns. The reason is that a check-in record is a point of place with the precise location and timestamp. Hence, we use the concept of feature region to transform the point level to the region level. The region level is to show people who come to a region rather to a specific place. To derive feature regions, we use the OPTICS algorithm [3] to cluster the check-in records with spatial proximity. The reason for using this OPTICS algorithm is that the cluster results can depict many check-ins in the cluster regions and represent the activities around an area. In Fig. 1, we have three feature regions A, B, and C in the left map. After extracting these regions, we can mark the check-in records to the feature regions. In Fig. 1, we represent the mobility evolution pattern of the day with feature regions of A, B, C, and D at the bottom of the graph. Thus, we get the basic daily mobility evolution patterns by ordering feature regions according to their timestamp.

After processing the spatial factor of the mobility evolution pattern, we need to determine the time factor of the pattern. Because the timestamp is too detailed to cluster daily mobility evolution patterns, we need to replace the timestamp of the patterns with time intervals. Intuitively, the easy way is to divide check-in records into a set of segments by a specific time interval. The mobility evolution patterns can now be recognized as a series of segments with feature regions. In order to represent the feature regions of the segments, we compute the distribution of the feature regions as the spatial model in each segment. Therefore, every
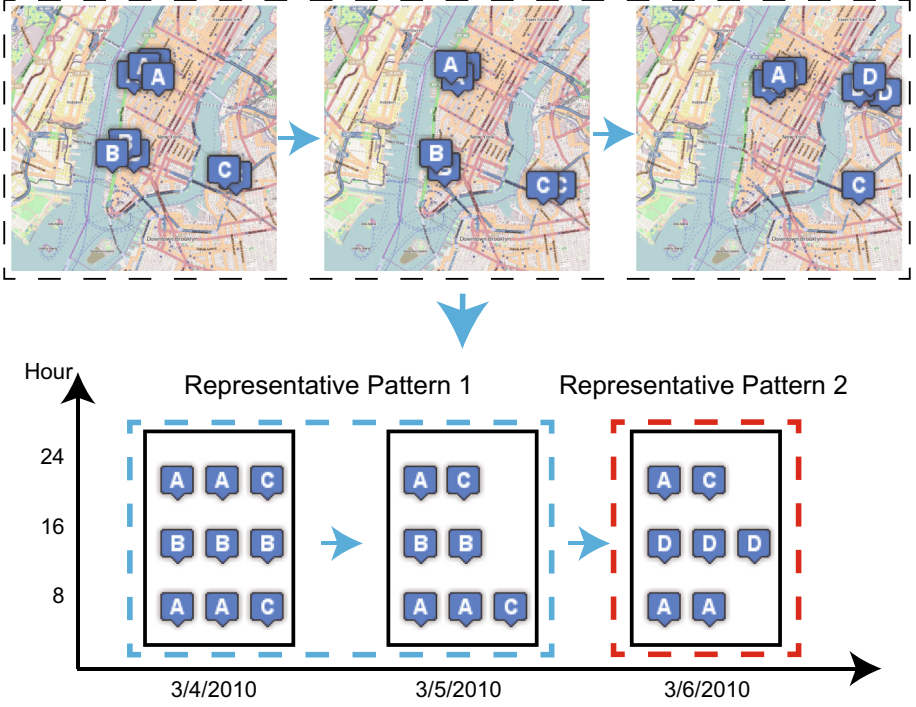
Fig. 1 An example of mobility evolution patterns

segment has its distribution of feature regions, and the mobility evolution pattern becomes a series of segments with feature region distribution.

With a large number of mobility evolution patterns, we may encounter the problem of loading a huge amount of mobility pattern data to memory for clustering the mobility evolution patterns. Therefore, we want to reduce the pattern size while still preserving mobility information. To solve this problem, we propose an algorithm GreedyMDL to compress the size of the mobility evolution pattern. Our algorithms combine the segments until we get good segmentation results. To measure good segmentation from a series of segments, we formulate the problem as a compression problem. We use the compression technique of the Minimum Description Length (MDL) principle [23], which has demonstrated its effectiveness in trading off accurate and concise data representations [25,26], for segmentation. By the MDL principle, we can merge segments with the shortest representative length. After segmentation, we would get the mobility evolution patterns of smaller size while preserving mobility for each day.

In light of the mobility evolution patterns of each day, we further cluster the daily mobility evolution patterns into groups, and for each group, we select one representative mobility evolution pattern. For example, in Fig. 1, we have two representative patterns from two clustering results. As can be seen in Fig. 1, the first 2 days have similar mobility evolution patterns so they would be grouped together. To cluster daily mobility evolution patterns, we derive a mobility evolution pattern distance function (MEPD) to measure the distance of the patterns. However, we may encounter performance issues in calculating all pair similarities in clustering when the number of days is huge. Therefore, we explore the locality-sensitive

hashing (LSH) [1,9,11] idea to speed up the cost of deriving similarities, which improves the clustering performance. Based on the features of mobility evolution patterns (i.e., the spatial and temporal information hidden in segments of the mobility evolution patterns), we propose the multiple level hash family. With the LSH method, the clustering process is efficient. When we cluster daily mobility evolution patterns to many groups, we can extract the representative mobility evolution patterns from each pattern group. Thus, we can have the representative mobility evolution patterns as the urban informatics around a city.

The main contributions of this paper are as follows:

- We formulate the problem of mining and clustering mobility evolution patterns from check-in data.
- We define mobility evolution patterns based on the MDL principle.
- We propose a distance function to measure the similarity in mobility evolution patterns and use this function for clustering mobility evolution patterns.
- We use the LSH concept to accelerate the performance of clustering mobility evolution patterns.
- We extract the representative mobility evolution patterns from the clustering result for the urban informatics in a city.
- We conduct experiments on real datasets to demonstrate the effectiveness and efficiency of our proposed algorithms.

The remainder of this paper is organized as follows: Sect. 2 reviews the related work. Section 3 presents the background information of our work. Section 4 describes the proposed methods of mining mobility evolution patterns. Section 5 illustrates the approach of clustering for representative mobility evolution patterns. Section 6 reports the performance of our algorithms. Section 7 concludes this paper.

2 Related works

Prior studies have elaborated on mining mobility patterns [19,22,29], mining trajectory patterns [4,7,10,13–15,21], recommending attractive locations [12,28,30] from collective GPS trajectories, and developing location searches [5,6,8,27].

In mining mobility patterns, the study in [22] extracted the movement features from check-in datasets to model the mobility pattern. Our method is different from this works in terms of handling the temporal factor. We use the MDL principle to determine the time variation. Moreover, the authors in [29] proposed fine-grained sequential patterns, and the [19] was to mine the periodic behaviors of moving objects. The authors in [7] studied the relation between human geographic movement, its temporal dynamics, and the ties of the social network. However, they did not handle the similarity issues of pattern clustering that proposed in our work.

In trajectory pattern mining, those studies on mining trajectory patterns focus on modeling mobility from GPS trajectories by exploring discrete Markov models [15] or spatio-temporal association rules [31]. The authors in [20] proposed mining periodic behaviors from moving trajectories. A periodic behavior is a statistical description of the periodic movement for one specific period. The study in [10] investigated the movement of objects from a sequence of spatio-temporal locations. However, the dataset of trajectory is high-sampling rate. Their methods cannot be applied to the low-sampling rate dataset of check-in records that we use.

Recently, location-representative information such as Travelogue has provided rich and useful user-generated content. The analysis of location-representative information enables

several application scenarios including destination recommendation and location summarization. For example, the authors in [30] proposed a location-aware recommendation framework that recommends interesting locations (e.g., Birds Nest) and possible activities (e.g., sightseeing) from geographic databases, GPS logs and Web data. The authors in [12] proposed a location-topic model that extracts location-representative knowledge from Travelogue. [28] proposed a location recommendation service that incorporates user preferences, social influence, and geographic influence which are integrated for Point of Interest (POI) recommendations in location-sharing services such as Foursquare, Facebook, etc. In these works, they focus on recommending location and are relevant to the feature regions we used in mobility evolution patterns. But, their purposes were not to extracting mobility patterns.

In summary, while the existing studies show promising results, the prior works did not address the problem of the clustering of mobility evolution patterns. Moreover, the methods of the prior works could not be applied to our work in pattern clustering for these three reasons. First, our mobility evolution patterns are different from the patterns of the previous works. For example, the work [22] did not consider the time factor, and the time factor of [7] is continuous. However, our patterns contain time factor, and the time is discrete. Second, previous works, such as [19], did not propose of similarity function for the patterning clustering. Third, dataset is not compatible because the trajectory data are high-sampling rate in some works such as [10]. In our work, we use the check-in dataset which is low-sampling rate. Therefore, we propose new mobility evolution patterns based on the MDL principle to solve the temporal issue, and this pattern can be apply to check-in records. In addition, for mobility evolution pattern clustering, we also propose a distance function to measure the similarity of the patterns.

3 Preliminary

In this section, we first describe our framework for mining and clustering mobility evolution patterns. Second, we define the mobility evolution patterns. Finally, we describe the objective function of mining mobility evolution patterns.

3.1 Framework

Our framework is illustrated in Fig. 2. The proposed framework consists of three components: feature region extraction, mobility evolution pattern extraction, and pattern clustering. In the

Fig. 2 Overview of the framework for mining and clustering mobility evolution patterns
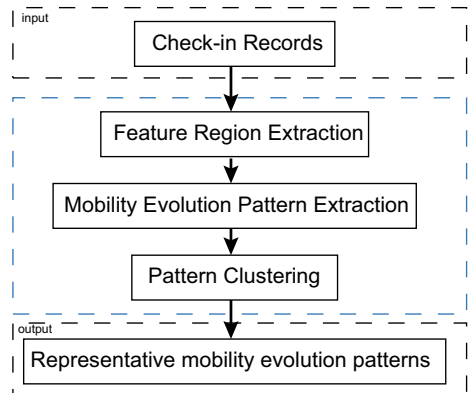
feature region extraction component, we group check-in records with location proximity to obtain the important areas in a city. Moreover, the mobility evolution pattern extraction is to mine the mobility pattern with the spatial and temporal factors, and it is used to reveal the daily movement in a city. For pattern clustering, we cluster the similar daily mobility evolution patterns to extract representative mobility evolution patterns. In the following sections, we describe the methods and definitions used in this framework.

3.2 Mobility evolution patterns and clusters

In this section, our goal is to define mobility evolution patterns. From a set of check-in data records, mobility evolution patterns depict the changes in moving behaviors with time. The following are some terms used in this paper.

Definition 1 (*Check-in records*) A **check-in record** $g_i \in G$ is a place with a timestamp and latitude and longitude location that is recorded by a user.

We first cluster check-in records to feature region set R by using the latitude and longitude attributes of the records.

Definition 2 (*Feature regions*) A **feature region** $r_i \in R$ is a cluster of locations derived by OPTICS [3]. It represents the location attribute of a check-in at the region level. Figure 1 shows four feature regions A, B, C, and D.

We mark these check-in records according to their feature region and divide the records by a time interval to get into a sequence of segments S.

Definition 3 (*Segments*) A **segment** $s_i \in S$ contains a set of check-in records in a time interval δ_t. For example, $s_i = \{g_1, g_2, \ldots, g_{|G_i|}\}$. Each check-in record $g_i \in s_i$ belongs to a feature region, where $f(g_i) = r_j$ and G_i is the check-in records in segment i. The $f(g)$ is the mapping function to get the check-in record's feature region, and r_i is one of the feature regions. For instance, Fig. 3 shows one segment with four check-in records, where two belong to the brown feature region, one belongs to the dark green feature region, and one is in the green feature region.

To formally describe the spatial aspect of a segment, we use a model to represent it.

Definition 4 (*Model*) A **model** $m_i \in M$ is the distribution of the feature regions in a segment which is represented as the probability sequence $p_i(r_1), \ldots, p_i(r_{|R|})$, where $p_i(r_1)$ indicates the occurring probability of r_1 and $|R|$ is the number of feature region types. In Fig. 3, the model m_1 is $\{p(brown) = 0.5, p(darkgreen) = 0.25, p(green) = 0.25\}$.

Definition 5 (*Change point*) In two consecutive segments, if their models are different, the dividing point is called the change point. In Fig. 3, one change point is between s_2 and s_3.

With the above definitions, we compress segments into mobility evolution patterns using the MDL principle.

Definition 6 (*Mobility evolution patterns*) A **mobility evolution pattern** $e_i \in E$ is a sequence of consecutive segments with models with the shortest representation length. We define the shortest representation length later. In Fig. 3, we can compress the twelve segments with three models. The three models form the mobility evolution patterns.
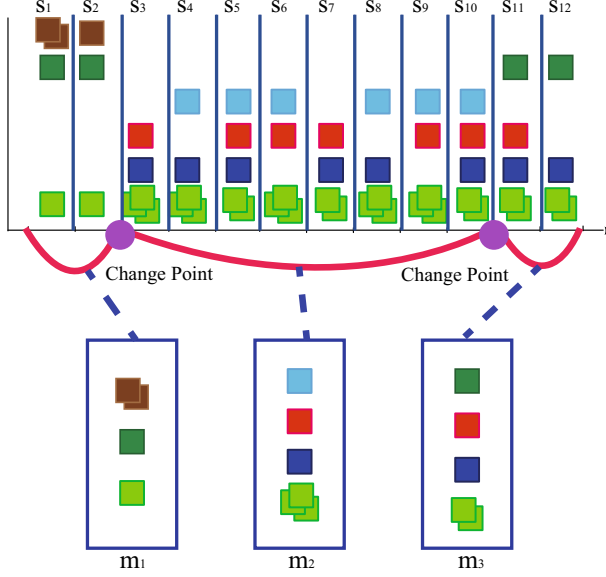
Fig. 3 Illustration of segments and models

Given a set of mobility evolution patterns of users, we can cluster the similar patterns together for urban planning.

Definition 7 (*Mobility evolution pattern clusters*) A **mobility evolution pattern cluster** is a set of similarity mobility evolution pattern. For example, Fig. 4a is shown the mobility evolution patterns of five users. In Fig. 4b, we can observe that two clusters (U_1, U_2) and (U_3, U_4, U_5) are extracted.

3.3 Representation length of the MDL

We have mentioned that the representation length is the way to determine the mobility evolution pattern. In this section, we introduce our objective representation length function based on the MDL principle [16,23,26]. To use the MDL principle, we need to introduce Kraft's inequality [24]. Using Kraft's inequality, we can identify the relationship between a code length and a probability. The relationship can be defined in the formula:

$$L(z) = \lceil -\log p(z) \rceil \tag{1}$$

where $p(z)$ is the probability of a random variable z, and $L(z)$ is the code length of $p(z)$. In our mobility evolution pattern, by the MDL principle, the representation length is divided into two parts. One is model length, which is used to measure the code length of each model. The other is the segment description length, which is the code length of the segment that is described by the models.

3.3.1 Model length

We now formulate the model length and the segment description length. Given the model set M, for each model $m_i \in M$, m_i is the probability sequence $p_i(r_1), \ldots, p_i(r_{|R|})$ according
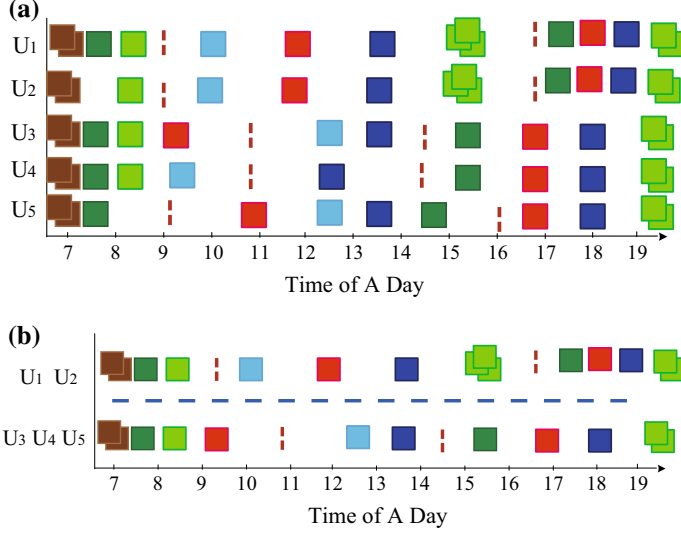
Fig. 4 Mobility evolution patterns and clusters. **a** Mobility evolution patterns. **b** Mobility evolution pattern clusters

to feature regions R. As a result, the model length of the model m_i is formulated as follows:

$$L(m_i) = \sum_{r_j \in R} -\log p_i(r_j). \tag{2}$$

We could further derive the model length of the model set M as follows:

$$L_{\text{model}}(M) = \sum_{m_i \in M} L(m_i) = \sum_{m_i \in M} \sum_{r_j \in |R|} -\log p_i(r_j). \tag{3}$$

3.3.2 The description length

Here, we introduce how to describe a segment using a model. Given one segment s_i and its corresponding model $m_i = (p_i(r_1), \dots, p_i(r_{|R|}))$, the description probability is as follows:

$$p(s_i|m_i) = \prod_{j=1}^{|R|} p_i(r_j)^{N(r_j, s_i)} \tag{4}$$

where $N(r_j, s_i)$ is the number of times that r_j occurs in s_i. Then, we can get the description length of m_i describing s_i below:

$$L(s_i|m_i) = -\log p(s_i|m_i) = \sum_{r_j \in |R|} -N(r_j, s_i) \log p_i(r_j). \tag{5}$$

Consequently, we acquire the segment description length of a sequence of segments S as follows:

$$L_{\text{segment}}(S|M) = \sum_{i=i}^{|S|} L(s_i|m_i) = \sum_{i=i}^{|S|} \sum_{r_j \in |R|} -N(r_j, s_i) \log p_i(r_j). \tag{6}$$

3.3.3 Representation length

Given model set M, segment set S, and feature region set R, the objective function designed to measure the representation length equals:

$$Q(M, S, R) = L_{\mathrm{model}}(M) + L_{\mathrm{segment}}(S|M). \tag{7}$$

4 Mining mobility evolution patterns

According to the objective function derived for mobility evolution patterns, we propose two greedy algorithms to efficiently discover the mobility evolution patterns.

4.1 Design concept

Given a set of check-in data, our first step is to extract the feature regions. As pointed out earlier, a feature region is a group of the check-in records with proximity. Once the feature regions have been extracted, the next step is to use the feature regions to discover the mobility evolution patterns for each day. The term evolution is used to describe how the mobility changes as time goes by. Moreover, the mobility evolution patterns are a series of segments where each segment has a sequence of the probability distribution of feature regions with a time interval. The naive way of discovering the mobility evolution patterns is to partition the check-in records with a specific time interval. For example, we can partition a day into twenty-four segments by a 1-h time interval. In this work, we use 1-h time interval according to the time feature of previous work [19]. The check-in records in each time interval would form a segment. Each segment would have a sequence of probability distribution of feature regions. The twenty-four segments form a mobility evolution pattern representing the mobility over time. However, with a big dataset, we may need a large amount of space to store every segment when using the naive method. We want to reduce the data size of mobility evolution patterns with fewer segments. In addition, when decreasing the size of the mobility evolution patterns, these patterns should still preserve the mobility information of the original patterns. To achieve this purpose, we borrow the concept of the Minimal Description Length (MDL) principle to compress the mobility evolution patterns. As we know, the MDL principle can be used to determine the regularity of the data. By this property we can compress mobility evolution patterns without losing much mobility information. To compress the mobility evolution patterns, we propose the representation length function Eq. 7 based on the MDL principle to merge segments. With the MDL principle, we can compress mobility evolution patterns with mobility information preserving. Thus, the compressed mobility evolution patterns would be small and would preserve the mobility information of the original data. According to the design concept, we propose two methods to mine mobility evolution patterns. One is the algorithm GreedyKL (standing for a greedy method based on KL divergence) and the other is GreedyMDL (standing for a greedy method based on MDL principle).

4.2 GreedyKL

Given a set of check-in records, we partition the records into a sequence of segments by a specific time interval, and we get a baseline mobility evolution pattern. Then, we use the GreedyKL algorithm to merge the similar segments until we reach the error threshold. The error threshold is the upper bound of error for compressing mobility evolution patterns. The

error is defined as follows:

$$\text{ERROR}(A, B) = \frac{1}{|T|} \sum_{i=0}^{|T|} H(A_{t_i}, B_{t_i}) \tag{8}$$

where A and B are models of mobility evolution patterns, T is the set of initial time intervals, and H is the Hellinger distance. The Hellinger distance is defined as follows:

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{9}$$

where P and Q are the probability distributions of the feature regions. The GreedyKL algorithm is to determine the segments that can be merged. To decide the segments to be merged, we need to formulate the similarity measurement among the segments. Each segment has its own model that depicts the probability distributions of feature regions. Thus, given two check-in segments, we use the Kullback-Leibler (abbreviated as KL) divergence as the similarity measurement. Without loss of generality, given two segments with their models P and Q, the distance between two probability distributions P and Q over feature regions is as follows:

$$\text{KL}(P \parallel R) = \sum_{i=1}^{|R|} p(R_i) \ln \frac{p(R_i)}{q(R_i)}. \tag{10}$$

Because the KL divergence is asymmetric, we define the distance for these two segments S_P and S_Q through the following equation:

$$D(S_P, S_Q) = \max \{\text{KL}(P \parallel Q), \text{KL}(Q \parallel P)\}. \tag{11}$$

The equation enables us to search for the most similar segment S_Q for the segment S_P. As we get the two most similar segments, we, then, merge the segments and execute the second take to compute the new representation length. When GreedyKL reaches a minimum representation length, it stops and returns the mobility evolution pattern.

Here, we formally introduce our pattern extraction algorithm, GreedyKL, to compress the mobility evolution patterns with error threshold. Specifically, given initial segments, GreedyKL computes the segments' initial model from the check-in records. Iteratively, GreedyKL selects the best pair of segments to merge according to the KL divergence Eq. 11. The merging of the best pair of segments in each iteration would have a new error rate from the original data. The algorithm would merge the segments until it reaches the error threshold. When reaching the error threshold, the algorithm terminates, and the current segment and model sets are returned as the mobility evolution result. Algorithm 1 summarizes the main idea of GreedyKL. First, GreedyKL assigns each segment with associated feature region distribution in its own segment (S^1, M^1) (Line 1). At each iteration of the algorithm, the most similar segments of KL divergence are selected and merged. Once the selected pair of segments have been merged, the segmentation result S^m and the set of models M^m at the m-th iteration are updated accordingly (Lines 15-16). If it reaches the error threshold, the segmentation result (S^m, M^m) is returned.

Time and space complexity: Let the number of models be k and the number of iterations be i. Given a sequence of n segments, it requires $O(n^2)$ space to store the spatial distances between each pair of segments in an iteration. On the other hand, it takes $O(n^2 * i)$ time in general as there are i iterations, each of which requires at most $O(n^2)$ operations to determine the best pair of segments.

Algorithm 1: GreedyKL

Input: S^1: initial segments;
    M^1: initial models;
    $error_{th}$: error threshold;
    $error_{cur}$: current error;

Output: (S^m, M^m): m-iteration segments and models;

1 $m = 1$;
2 $error_{cur} = 0$;
3 **while** $error_{cur} <= error_{th}$ **do**
4 $l_{min} = l_{cur}$;
5   Initialize $dist$ as zero;
6   **for** $j = 1$ *to* $|S^m| - 1$ **do**
7   **for** $k = j + 1$ *to* $|S^m|$ **do**
8   **if** $D(S_j^m, S_{j+1}^m) < dist$ **then**
9 $l = j$;
10 $r = k$;
11 $dist = D(S_j^m, S_{j+1}^m)$;
12   **end**
13   **end**
14   **end**
15   Update S^{m+1} by merging S_l^m, S_r^m ;
16   Update M^{m+1} by merging M_l^m, M_r^m;
17 $m = m + 1$;
18 $error_{cur} = ERROR(M^{m+1}, M^1)$;
19 **end**
20 **return** (S^m, M^m) ;

4.3 GreedyMDL

We introduce our algorithm GreedyMDL in this subsection. Unlike the compressing pattern with KL divergence of GreedyKL, the GreedyMDL method is to merge segments based on the MDL principle. The GreedyMDL method would greedily merge the segments if it gets the shortest representation length after merging. Specifically, given initial segments and an error threshold, the GreedyMDL computes the change points in a bottom-up fashion. Iteratively, GreedyMDL selects the best pair of segments to merge according to the objective function Eq. 7. The best pair of segments in each iteration is the one for which the merging has the shortest represented length. After merging, we get the error rate from the original data, and the current set of change points is returned. The algorithm would terminate if the error of the new pattern is larger than the error threshold. With the set of change points, we can easily get the segmentation results. Algorithm 2 summarizes the main idea of GreedyMDL. First, GreedyMDL assigns each segment with associated feature region distribution in its own segment (S^1, M^1). At the m-th iteration of the algorithm, the best pair of segments that satisfies the objective function is selected and merged. Once the selected pair of segments have been merged, the segmentation result S^m and the set of models M^m at the m-th iteration are updated accordingly (Lines 18–19). Then, it computes the new error rate from the original data. If the new error rate exceeds the error threshold, the segmentation result (S^m, M^m) based on the current set of split points is returned.

Time and space complexity: Let the number of models be k and the number of iterations be i. The complexity of GreedyMDL is analogous to the agglomerative clustering algorithm,

as GreedyMDL starts with initial segments and iteratively merges selected pairs of segments to merge until one or k segments are left. Therefore, for a sequence of n initial segments, it requires $O(n^2)$ space to store the spatial distances between each pair of segments. On the other hand, it takes $O(n^2 * i)$ time in general as there are i iterations, each of which requires at most $O(n^2)$ steps to update the spatial distance among segments and determine the best pair of segments.

Algorithm 2: GreedyMDL

Input: S^1: initial segments;
M^1: initial models;
$error_{th}$: error threshold;
$error_{cur}$: current error;

Output: (S^m, M^m): m-iteration segments and models;

2 $m = 1$;
3 $error_{cur} = 0$;
4 **while** $error_{cur} <= error_{th}$ **do**
5   Initialize $dist$ as zero;
6   **for** $j = 1$ *to* $|S^m| - 1$ **do**
7   **for** $k = j + 1$ *to* $|S^m|$ **do**
8   S': combine S_j^m and S_k^m in S^m;
9   M': combine M_j^m and M_k^m in M^m;
10   **if** $Q(M', S', R) < dist$ **then**
11     $l = j$;
12     $r = k$;
13     $dist = Q(M', S', R)$;
14   **end**
15   **end**
16 **end**
17 Update S^{m+1} by merging S_l^m, S_r^m ;
18 Update M^{m+1} by merging M_l^m, M_r^m;
19 $m = m + 1$;
20 $error_{cur} = ERROR(M^{m+1}, M^1)$;
21 **end**
22 **return** (S^m, M^m) ;

---

5 Clustering mobility evolution patterns

Given a set of daily check-in records, we already present two algorithms, GreedyKL and GreedyMDL, to extract the mobility evolution patterns of each day. Thus, the daily mobility evolution patterns from check-in records in an area are derived. In this section, we further cluster the daily mobility evolution patterns into groups, and for each group, one representative mobility evolution pattern is derived.

5.1 Hellinger distance and LSH

Before we describe the method of pattern clustering, we introduce the technique of the Hellinger distance and the locality-sensitive hash (LSH) that are used in our clustering method. The Hellinger distance is used to measure the similarity of the probability sets.

Equation 9 is the distance function of the Hellinger distance. Because the mobility evolution patterns are represented with the probability distribution of feature regions, we can use this distance function to measure the similarity of the patterns. Another advantage is that the Hellinger distance can be taken as the Euclidean norm [17]. Because an optimal locality-sensitive hash family exists within the Euclidean distance [1], it can be used to improve the similarity performance by eliminating the dissimilar patterns. Therefore, we can exploit LSH to improve the performance of the clustering algorithm.

5.2 Distance function of mobility evolution patterns

To perform pattern clustering, we need to measure the similarity of the daily mobility evolution patterns. To achieve this purpose, we propose the mobility evolution patterns distance function (MEPD) based on the concept of the Hellinger distance. The idea of MEPD is to measure the average Hellinger distance over all overlapping time intervals. Given two mobility evolution patterns $A = (a_1, a_2, \ldots, a_i)$ and $B = (b_1, b_2, \ldots, b_j)$, we have to determine the overlapping time intervals because different time intervals represent different movements.

The distance of mobility evolution patterns is the average Hellinger distance. We define the distance as follows:

$$\text{MEPD}(A, B) = \sum_{a_i \in A} \sum_{b_j \in B | O(a_i, b_j) > 0} H(a_i, b_j) \times \frac{O(a_i, b_j)}{TI} \tag{12}$$

where $O(a_i, b_j)$ is the overlap of time intervals, and TI is the time summation of all time intervals.

In general, the MEPD can be taken as the average Hellinger distance between the mobility evolution patterns. Thus, if two mobility evolution patterns are similar, the MEPD would be smaller. By using the MEPD function, we can then perform clustering on the mobility evolution patterns that are extracted from the check-in data.

5.3 Exploring LSH to speed up clustering

Based on the similarities function, we could perform existing clustering algorithms to cluster daily mobility evolution patterns. With GreedyMDL, we can merely load compressed patterns for clustering. This can handle the issue of memory size in clustering. Another major challenge in clustering is the performance of deriving all similarities among patterns. Thus, in this paper, we borrow the concept of the LSH (standing for locality-sensitive hashing) to prune the cost of deriving similarities among all patterns. In this paper, we adopt density-based clustering algorithms as the clustering algorithm. The reason for using the OPTICS algorithm is because we may not know how many clusters exist. Performing the OPTICS algorithm for pattern clustering according to their mobility patterns may result in performance issues when the data size is huge. In the OPTICS algorithm, we have to compute the MEPD of all patterns to get the pattern's neighbors. Because MEPD needs to compute the Hellinger distance of all models in the mobility patterns, the time complexity may grow to $O(n^3)$. As the data are big, it may not be feasible for computation. Hence, to accelerate the cluster efficiency, we use the concept of locality-sensitive hashing (LSH), which is used to determine the similar and dissimilar patterns, and guarantee the quality of the similarity results. When we can separate the similar and dissimilar patterns, we only have to compute the similarity of the similar partitions. By exploring the LSH, we can avoid the comparison of all mobility evolution patterns, and only compute the MEPD on the similar patterns.

Fig. 5 Multi-level LSH illustration
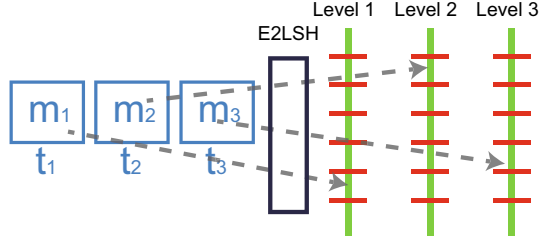
In the LSH algorithm, we use the existing method of E2LSH [2] for the pattern clustering on mobility evolution patterns. E2LSH is the near optimal LSH on the Euclidean norm. In addition, E2LSH can be used in clustering mobility evolution patterns. This is because the Hellinger distance in the MEPD function can be taken as the Euclidean norm [17]. Therefore, we can use the E2LSH method to solve the performance issue.

Because the time intervals in mobility evolution patterns vary, to perform E2LSH we need to add the following steps. First, we decompress the time intervals of the mobility evolution patterns into the initial time interval. Hence, we modify the MEPD function to LSH MEPD in Eq. 14. Assume we have the initial time interval set $T = (t_0, t_1, \ldots, t_k)$, which is the specific time interval for generating the initial segments. We use a time mapping function to get the corresponding pattern model. The time mapping function is below:

$$MT(t_i, A) = a_i \mid t_i \in I(a_i), \quad a_i \in A \tag{13}$$

where $I(a_i)$ is the time interval of the pattern model a_i.

$$\text{LMEPD}(A, B) = \sum_{i=1}^{k} H(MT(t_i, A), MT(t_i, B)) \times \frac{t_i}{|T|} \tag{14}$$

where $|T|$ is the time summation of all time intervals. Second, because the LMEPD function is the aggregation of the Hellinger distance, we create different level LSH buckets depending on the initial time interval set. Then, each model in the mobility evolution patterns is passed to its own level buckets via E2LSH. Figure 5 illustrates how models in a pattern hash to the different levels of buckets. For example, m_1, m_2, and m_3 are assigned to buckets at different levels. Thus, we can use the mobility patterns in the same level buckets for computing the MEPD to get the neighbors. By using the multi-level E2LSH method, the OPTICS algorithm on MEPD with LSH can reduce the time of comparison for selecting the neighbors and thus achieve better performance.

5.4 Extracting representative mobility evolution patterns

After the clustering process, we could have a set of daily mobility pattern groups and each group can form the representative mobility evolution patterns around a city. Although the daily mobility patterns in the same group are similar, each may still have different probability distributions of feature regions and time intervals. Therefore, we need a post processing step to integrate the patterns in the same group to extract representative mobility evolution patterns. For this purpose, we first decompress the patterns in the same group to the initial time interval, and each time interval of a pattern would have its own probability distribution of feature regions. Second, for each group, we compute the average probability distribution of feature regions of the patterns in the same initial time interval. After this step, we would have a new mobility evolution pattern with initial time intervals for each day. Finally, to acquire

a representative mobility evolution pattern, we run the GreedyMDL algorithm again to get a new mobility evolution pattern. Thus, every group can generate a representative mobility evolution pattern, and the representative mobility evolution patterns of a city are derived.

6 Performance evaluation

In this section, we first describe our experimental settings. We then analyze the quality and performance of clustering mobility evolution patterns.

6.1 Experimental settings

We set up our experiment with a real dataset, the Gowalla check-in dataset [7], in order to contend with real-world phenomena. Table 1 summarizes the details of the datasets that we use. It has 353,485 check-in records around New York city from February 2009 to October 2010 from Gowalla, and 567,472 records in American East from Brightkite.

We use different metrics for evaluating the quality and performance of clustering mobility evolution patterns. In mobility evolution patterns, we measure the pattern phenomenon by two metrics: *Segment Size* which is the number of segments used in mobility evolution patterns; and (ii) *Error* which is the average of the Hellinger distance from the original data. Intuitively, a small segment size means that we use less storage to store the patterns, and a higher error indicates that less original information is preserved from the ordinal check-in records. Moreover, in clustering mobility evolution patterns, we use *Mobility Loss Rate* to measure the effectiveness of the mobility evolution patterns and *Runtime* to measure the efficiency of the clustering algorithm.

Because we use lossy compression method to extract the mobility evolution patterns, we have the information loss compared to the original data. For example, in Fig. 6, U_1 and U_2 are similar in uncompressed patterns, but they are dissimilar after the algorithm GreedyMDL because of the more distinct region distribution (the dash square). Thus, to evaluate the effectiveness of the mobility evolution patterns in pattern clustering, we propose the *mobility loss rate* to measure the information loss compared to the uncompressed patterns. The mobility loss rate is defined as the number of users for incorrect cluster assignment divided to the total number of users. The correct cluster assignment is defined as the cluster results of uncompressed patterns because the uncompressed patterns would preserve all mobility information of users. For measure mobility loss rate, we use uncompressed patterns as the baseline method, which uses initial segments and models as the mobility evolution patterns. Thus, if the mobility evolution patterns we extracted have lower mobility loss rate, it depicts that the patterns could be less mobility information loss and more representativeness.

Table 1 Check-in datasets

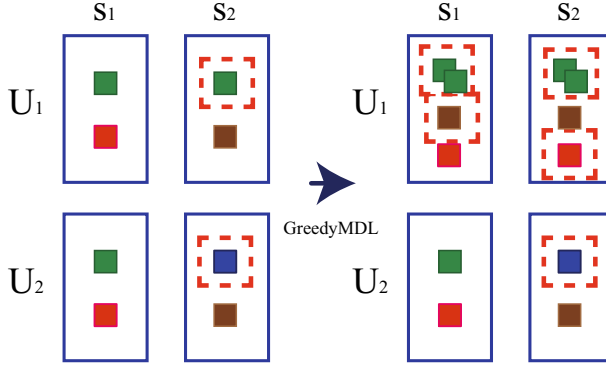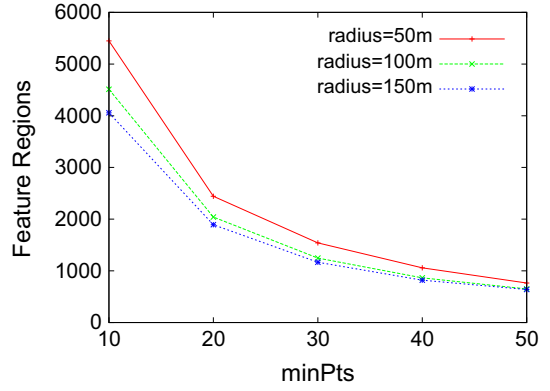|	Gowalla around New York	Brightkite in American East
Records	353,485	567,472
Duration	February 2009–October 2010	April 2008–October 2010

Fig. 6 Illustration of mobility loss rate

Fig. 7 Feature regions with different OPTICS settings

6.2 Performance study

6.2.1 Evaluation of mobility evolution patterns

Before we start the discussion of the quality of the mobility evolution patterns, we first investigate the relationship between the feature regions and parameters (radius ε and minimum points $minPts$) of the OPTICS algorithm. In Fig. 7, the number of feature regions decreases when the radius increases or the number of minimum points decreases. For example, there are 1059 feature regions when the radius is 50 m and $minPts$ is 40 points, and 22 regions when the radius is 300 m and $minPts$ is 40 points. This is because a larger radius would form a cluster with a large area, and smaller $minPts$ would easily form a cluster. The number of feature regions would influence the results of the mobility evolution patterns. Figure 8 illustrates how feature regions and error threshold influence the number of compressed segments in a mobility evolution pattern.

We would have fewer segments with a larger radius and smaller $minPts$. The reason is that there is higher probability of having two segments with similar region distribution when we have fewer feature regions. If two similar segments would be merged by the GreedyMDL algorithm, then we get a lower error rate in the mobility evolution patterns. Moreover, with a higher error threshold, we can compress the mobility evolution patterns to a fewer number of
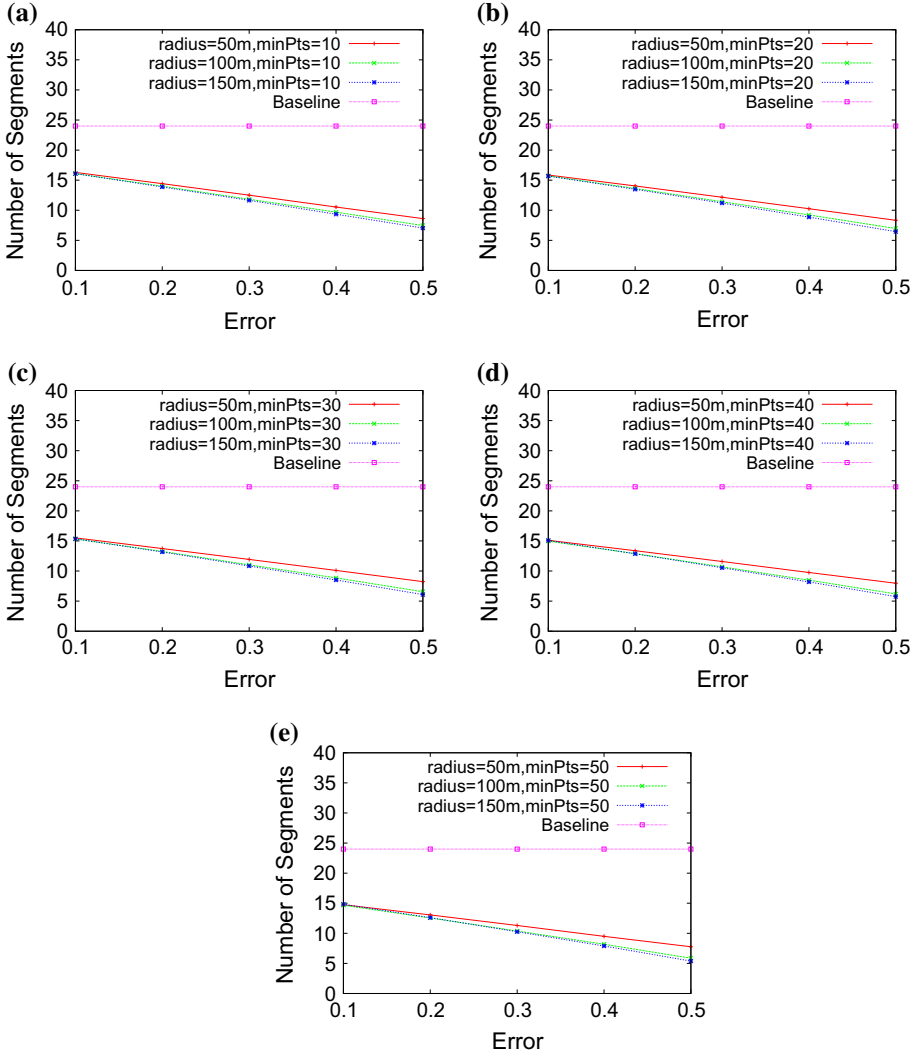
Fig. 8 Number of segments with the error threshold varied

segments. For example, in Fig. 8c, the number of segments of error threshold 0.5 is fewer than the number of segments of error threshold 0.1 in every feature regions setting. If we increase the error threshold, we can have good compression of the mobility evolution patterns with fewer segments. However, fewer segments would also lose some mobility information of the mobility evolution patterns. Thus, we use the result of the experiment to get the parameters of the OPTICS algorithm and error threshold. According to the discussion above, we have a concept of the relationship in the mobility evolution patterns, feature regions, and the parameters of OPTICS. Next, we discuss how to decide better OPTICS parameters and error threshold by pattern clustering and how to evaluate the quality of the representative mobility evolution patterns.

Fig. 9 Number of pattern clusters with different OPTICS settings
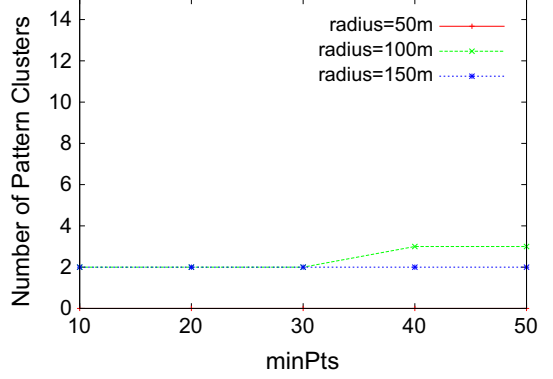
6.2.2 Performance of clustering mobility evolution patterns

In this section, we evaluate the quality and performance of pattern clustering for representative mobility evolution patterns. In the last section, we have left the issue of how to decide the proper OPTICS parameters. We would need proper parameters for the performance experiment. To decide the parameters, we use a parameter test to find the parameter which can extract the greatest number of clusters. If we have fewer feature regions, the daily mobility evolution patterns would be similar and form fewer clusters. And, if we have too many feature regions, the daily mobility evolution patterns would be dissimilar and no clusters may exist. For this reason, we run pattern clustering on the daily mobility evolution patterns of the baseline method. We choose the parameters of the OPTICS which have the largest number of clusters as our experiment setting of the OPTICS in the feature regions. In Fig. 9, we find that it extracts the largest number of clusters when using a radius equal to 100 m and minimum points equal to 50. Thus, we use this setting to evaluate the performance of the clustering for representative mobility evolution patterns.

After deriving the setting of the OPTICS in feature regions, the other issue is to determine the error threshold of the GreedyKL and GreedyMDL algorithm. For this issue, we want to have an error threshold that would have higher mobility preserving patterns after the compression of the GreedyMDL algorithm. To have higher mobility preserving patterns, we would choose an error threshold of the GreedyMDL algorithm that would have the lowest mobility loss rate for the clustering result compared with the baseline. The reason is that lower mobility loss rate means the mobility information is not lost after GreedyMDL compression. Thus, the clustering result would be mobility preserving. Figure 10 is the mobility loss rate of different error thresholds of the GreedyMDL algorithm. As Fig. 10 shows, the mobility loss rate dramatically decreases after the error threshold 0.6. This is because a higher error threshold leads to higher compression, and it also eliminates some mobility information. Furthermore, the mobility loss rate of GreedyKL decreases after the error threshold 0.5. This means the ability of mobility preserving of GreedyMDL would be better than that of GreedyKL. Based on the results of the mobility loss rate experiment, we choose error threshold 0.3 for GreedyKL and 0.5 for GreedyMDL as the better mobility preserving parameter for the GreedyMDL algorithm.

We have already determined the parameters of the feature regions and the GreedyMDL algorithm, and then we start to evaluate the effectiveness and efficiency of the clustering for representative mobility evolution patterns. For effectiveness, we compared the data size and

Fig. 10 The mobility loss rate of GreedyMDL and GreedyKL with the error threshold varied
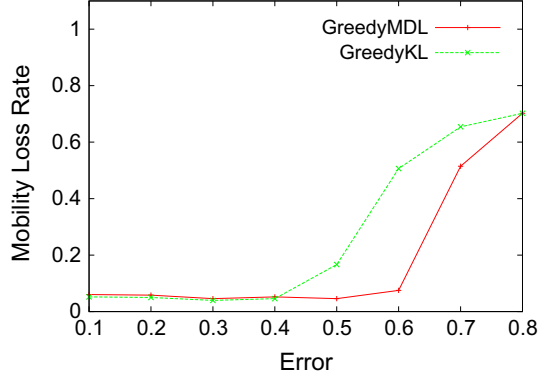

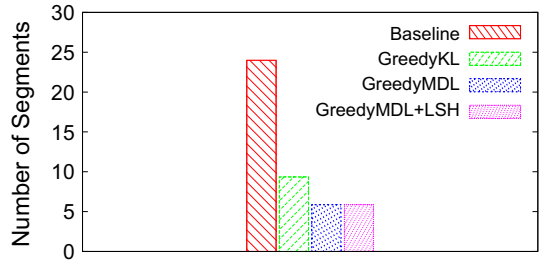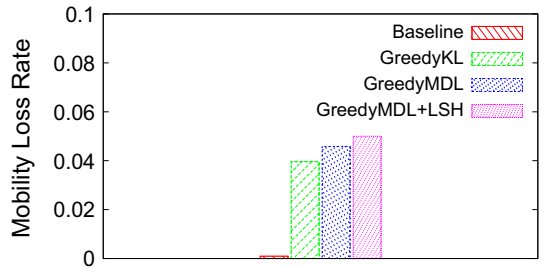**Fig. 11** Data size of GreedyKL, GreedyMDL, and GreedyMDL with LSH compared with Baseline

Fig. 12 Mobility Loss Rate of Baseline, GreedyKL, GreedyMDL, and GreedyMDL with LSH

mobility loss rate of the four methods: baseline, GreedyKL, GreedyMDL, and GreedyMDL with LSH. Here, the parameters of GreedyKL are radius of 100 m, minimum points of 50, and error of 0.3, and for GreedyMDL, and GreedyMDL with LSH, the settings are radius of 100 m, minimum points of 50, and error of 0.5. In Fig. 11, GreedyKL, GreedyMDL, and GreedyMDL with LSH have fewer segment to describe the daily mobility patterns than the baseline method, and GreedyMDL has fewest segments. Thus, the three algorithms store the patterns with less storage. In addition, GreedyMDL performs better compression than GreedyKL. Although GreedyKL, GreedyMDL, and GreedyMDL with LSH can save the space for storing data, they still have low mobility loss rate around 0.1 as shown in Fig. 12. That is, the mobility evolution patterns of our algorithms can be mobility preserving.

Finally, for efficiency, we compared the runtime of the four algorithms: baseline, GreedyKL, GreedyMDL, and GreedyMDL with LSH. Figure 13 shows that GreedyMDL is faster than baseline and GreedyKL, and GreedyMDL with LSH is the fastest. The reason is that GreedyMDL has fewer segments than baseline and GreedyKL, and fewer segments can reduce the times of comparison. In addition, GreedyMDL with LSH is the most efficient.

Fig. 13 Runtime comparison of Baseline, GreedyKL, GreedyMDL, and GreedyMDL with LSH of Gowalla
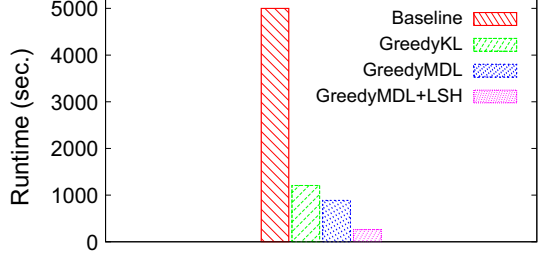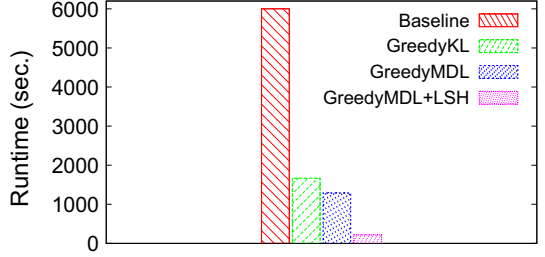

Fig. 14 Runtime comparison of Baseline, GreedyKL, GreedyMDL, and GreedyMDL with LSH of Brightkite

This is because it can avoid all pairs comparison in the clustering algorithm. Moreover, in Fig. 14, GreedyMDL is also faster than baseline and GreedyKL in a larger dataset. These results demonstrate that GreedyMDL and GreedyMDL with LSH are effective and efficient for extracting the representative mobility patterns.

7 Conclusion

In this paper, our goal is to mine and cluster daily mobility evolution patterns from check-in data. Via the clustering process, we intended to discover representative daily mobility evolution patterns. To achieve this, we addressed the problems: (1) how to model the mobility evolution patterns and (2) how to cluster the mobility evolution patterns. Thus, we propose two methods, GreedyKL and GreedyMDL, to extract the mobility evolution patterns for describing the movements. The mobility evolution pattern is formed from a sequence of segments, where each check-in segment is represented by a region distribution derived from the check-in records. To get a good mobility evolution pattern, our algorithms merge segments until we get the shortest representation length, where the representation length is defined based on the Minimum Description Length (MDL) principle. For grouping the users according to their mobility patterns, we built a mobility evolution pattern distance (MEPD), which is based on the Hellinger distance, to measure the similarity of the patterns. With this pattern distance function, we can run a cluster algorithm, which is the OPTICS, to group the users with similar mobility evolution patterns. In addition, to handle the performance issue on all pair comparisons, we used the locality-sensitive hashing (LSH) concept to put the users with similar patterns together. In this way, we only need to calculate the pattern distance with possibly similar patterns. In our experiments, we used the metrics of error and mobility loss rate to evaluate the quality of the mobility evolution patterns. The experimental results show that our algorithms can mine mobility evolution patterns with good quality. Moreover, for the clustering of mobility evolution patterns, our proposed scheme is able to speed up the

clustering process. In the mobility loss rate experiment, would prove the patterns we used in pattern clustering has lower mobility loss rate.

Compliance with ethical standards

Conflicts of interest The authors declare that they have no conflict of interest.

Ethical approval This chapter does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Andoni A, Indyk P (2006) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: 47th Annual IEEE symposium on foundations of computer science, 2006. FOCS'06. IEEE, pp 459–468
2. Andoni A, Indyk P (2004) E2lsh: exact euclidean locality-sensitive hashing. Implementation available at http://www.mit.edu/andoni/LSH/
3. Ankerst M, Breunig MM, Kriegel H-P, Sander J (1999) Optics: ordering points to identify the clustering structure. ACM SIGMOD Record 28(2):49–60
4. Bagrow JP, Lin Y-R (2012) Mesoscopic structure and social aspects of human mobility. PloS One 7(5):e37676
5. Cao X, Cong G, Jensen C (2010) Retrieving top-k prestige-based relevant spatial web objects. Proc VLDB Endow 3(1–2):373–384
6. Cao X, Cong G, Jensen C, Ooi B (2011) Collective spatial keyword querying. In: Proceedings of ACM international conference on management of data, pp 373–384
7. Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM international conference on knowledge discovery and data mining, pp 1082–1090
8. Cong G, Jensen C, Wu D (2009) Efficient retrieval of the top-k most relevant spatial web objects. Proc VLDB Endow 2(1):337–348
9. Datar M, Immorlica N, Indyk P, Mirrokni VS (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the 20th annual symposium on computational geometry. ACM, New York, pp 253–262
10. Giannotti F, Nanni M, Pinelli F, Pedreschi D (2007) Trajectory pattern mining. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 330–339
11. Gionis A, Indyk P, Motwani R et al (1999) Similarity search in high dimensions via hashing. In: 'VLDB', vol 99, pp 518–529
12. Hao Q, Cai R, Wang C, Xiao R, Yang J, Pang Y, Zhang L (2010) Equip tourists with knowledge mined from travelogues. In: Proceedings of the 19th international conference on World Wide Web, pp 401–410
13. Hsieh H-P, Li C-T, Lin S-D (2012) Exploiting large-scale check-in data to recommend time-sensitive routes. In: Proceedings of the ACM SIGKDD international workshop on urban computing. ACM, New York, pp 55–62
14. Jeung H, Liu Q, Shen H, Zhou X (2008) A hybrid prediction model for moving objects. In: Proceedings of the 24th international conference on data engineering, pp 70–79
15. Jeung H, Yiu M, Zhou X, Jensen C (2010) Path prediction and predictive range querying in road network databases. VLDB J 19(4):585–602
16. Koivisto M, Perola M, Varilo T, Hennah W, Ekelund J, Lukk M, Peltonen L, Ukkonen E, Mannila H (2003) An mdl method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. In: Pacific symposium on biocomputing, Vol 8, pp 502–513
17. Krstovski K, Smith DA, Wallach HM, McGregor A (2013) Efficient nearest-neighbor search in the probability simplex. In: Proceedings of the 2013 conference on the theory of information retrieval. ACM, New York, p 22
18. Lakshmanan V (2012) Automating the analysis of spatial grids: a practical guide to data mining geospatial images for human and environmental applications. Springer, New York

19. Li Z, Ding B, Han J, Kays R, Nye P (2010) Mining periodic behaviors for moving objects. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 1099–1108
20. Li Z, Han J, Ji M, Tang L, Yu Y, Ding B, Lee J, Kays R (2011) Movemine: mining moving object data for discovery of animal movement patterns. ACM Trans Intell Syst Technol 2(4):37
21. Monreale A, Pinelli F, Trasarti R, Giannotti F (2009) Wherenext: a location predictor on trajectory pattern mining. In: Proceedings of the 15th ACM international conference on knowledge discovery and data mining, pp 637–646
22. Noulas A, Scellato S, Lathia N, Mascolo C (2012) Mining user mobility features for next place prediction in location-based services. In: 2012 IEEE 12th international conference on data mining (ICDM), IEEE, pp 1038–1043
23. Rissanen J (1978) Modeling by shortest data description. Automatica 14(5):465–471
24. Rissanen JJ (1976) Generalized kraft inequality and arithmetic coding. IBM J Res Dev 20(3):198–203
25. Sun J, Faloutsos C, Papadimitriou S, Yu P (2007) Graphscope: parameter-free mining of large time-evolving graphs. In: Proceedings of the 13th ACM international conference on knowledge discovery and data mining, pp 687–696
26. Wang P, Wang H, Liu M, Wang W (2010) An algorithmic approach to event summarization. In: Proceedings of ACM international conference on management of data, pp 183–194
27. Wu D, Yiu ML, Cong G, Jensen CS (2012) Joint top-k spatial keyword query processing. IEEE Trans Know Data Eng 24(10):1889–1903
28. Ye M, Yin P, Lee W, Lee D (2011) Exploiting geographical influence for collaborative point-of-interest recommendation. In: Proceedings of the 34th ACM international conference on research and development in information retrieval
29. Zhang C, Han J, Shou L, Lu J, La Porta T (2014) Splitter: mining fine-grained sequential patterns in semantic trajectories. In: Proceedings of the VLDB endowment, vol 7, no 9, pp 769–780
30. Zheng V, Zheng Y, Xie X, Yang Q (2010) Collaborative location and activity recommendations with gps history data. In: Proceedings of the 19th international conference on World Wide Web, pp 1029–1038
31. Zheng Y, Liu L, Wang L, Xie X (2008) Learning transportation mode from raw gps data for geographic applications on the web. In: Proceedings of the 17th international conference on World Wide Web, pp 247–256

Chien-Cheng Chen received the BS degree and MS degree from the National Cheng Kung University, Tainan, Taiwan, in 2005 and 2007, respectively. He is currently a Ph.D. student at the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan. His research focus is on data mining, especially, trajectory pattern mining and heterogeneous social networks.
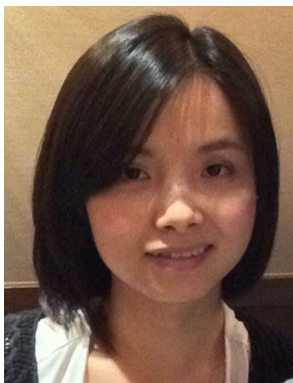
Meng-Fen Chiang received the B.E. and M.E. degrees from National ChengChi University in Taiwan, in 2004 and 2006, respectively, and the Ph.D. degree in computer science from National Chiao Tung University, Taiwan, in 2013. She was a software engineer at the Yahoo Taiwan from 2012–2014. In Feb. 2014, she joined Living Analytics Research Centre, Singapore Management University, as a research fellow. Her current research interests include big data analytics and spatio-temporal data mining.

Wen-Chih Peng received the BS and MS degrees from the National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in electrical engineering from the National Taiwan University, Taiwan, R.O.C, in 2001. Currently, he is a 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.