The VLDB Journal
DOI 10.1007/s00778-011-0262-6

SPECIAL ISSUE PAPER

Clustering and aggregating clues of trajectories for mining trajectory patterns and routes

Chih-Chieh Hung · Wen-Chih Peng · Wang-Chien Lee

Received: 24 September 2010 / Revised: 28 September 2011 / Accepted: 1 November 2011

Abstract In this paper, we propose a new trajectory pattern mining framework, namely *Clustering and Aggregating Clues of Trajectories (CACT)*, for discovering *trajectory routes* that represent the frequent movement behaviors of a user. In addition to spatial and temporal biases, we observe that trajectories contain *silent durations*, i.e., the time durations when no data points are available to describe the movements of users, which bring many challenging issues to trajectory pattern mining. We claim that a movement behavior would leave some *clues* in its various sampled/observed trajectories. These clues may be extracted from spatially and temporally co-located data points from the observed trajectories. Based on this observation, we propose *clue-aware trajectory similarity* to measure the clues between two trajectories. Accordingly, we further propose the *clue-aware trajectory clustering* algorithm to cluster similar trajectories into groups to capture the movement behaviors of the user. Finally, we devise the *clue-aware trajectory aggregation* algorithm to aggregate trajectories in the same group to derive the corresponding trajectory pattern and route. We validate our ideas and evaluate the proposed CACT framework by experiments using both synthetic and real datasets. The experimental results show that CACT is more effective in discovering trajectory patterns than the state-of-the-art techniques for mining trajectory patterns.

C.-C. Hung · W.-C. Peng (✉)
National Chiao Tung University, Hsinchu, Taiwan
e-mail: wcpeng@cs.nctu.edu.tw

C.-C. Hung
e-mail: oshin@yahoo-inc.com

W.-C. Lee
The Pennsylvania State University,
State College, PA 16801, USA
e-mail: wlee@cse.psu.edu

Keywords Trajectory pattern mining · Trajectory similarity · Trajectory clustering

1 Introduction

Owing to the pervasiveness of GPS-equipped mobile devices today, the locations of users can be easily determined and shared with friends via social networking web sites such as Google Latitude and Foursquare. Notice that movements of mobile users can be captured as *trajectories*. For example, time-stamped and geotagged photographs shared by a user on Flickr may provide the trajectory of her photo-shooting trip. Moreover, some time-sorted check-in records of a user on Foursquare also form a trajectory. Furthermore, a number of web sites have been hosting the sharing of user-generated trajectory data [1]. Obviously, user trajectories provide very valuable information that can be useful for various location-based services such as trip planning, personalized navigation routing services, mobile commerce, and location-based recommendation services. In these application domains, techniques for mining trajectory patterns and frequent trajectory routes are very important. In this paper, we study data mining techniques for discovering the frequent trajectory patterns and routes of individual users.

Mining trajectory patterns and routes are very challenging due to inherent noises and the limitations of trajectory acquisition technology. Generally speaking, a trajectory consists of sequential data points recording the locations and associated occurrence time sampled from the movements of a user. Given the logged historic trajectories of the user, we not only aim to identify the sequential relationships, also termed as the *movement behavior*, among regions where the user frequently passes by but also to construct detailed trajectory routes that represent these movement behaviors.

Published online: 20 November 2011

Fig. 1 An example of trajectories from the CarWeb dataset. **a** T_1. **b** T_2. **c** T_3. **d** Time representation

The prior work in [8] defines general movement behavior that consists of movement features, such as velocity, acceleration, and spatial regions. In this paper, we study the movement behavior in terms of spatial regions. Nevertheless, as reported in many prior works, trajectories exhibit certain spatial/temporal biases and temporal shifts. In other words, the locations and occurrence time of data points in two trajectories are usually not the same, even if these two trajectories capture the same movement behavior of the user. We use Fig. 1 as an example to illustrate the spatial/temporal biases, temporal shifts, and a phenomenon called *silent duration* that frequently appears in trajectories. Figure 1a–c show three trajectories T_1, T_2 and T_3, respectively, collected from our CarWeb service.[1] These three trajectories actually logged the same movement behavior of a user. Notice that the roman numbers associated with the data points represent their sampling order in the corresponding trajectories. Figure 1d shows the timestamps of all data points (and their corresponding trajectories) in the time line. It can be seen in Fig. 1 that data points of T_1 and T_3 are not exactly aligned in either spatial or temporal dimensions, even though these two trajectories capture the same movement behavior of the user. Compared to T_1 and T_3, T_2 is delayed for approximately 2 time slots. This is referred to as the temporal shift of trajectories in this paper. Consider the data points collected at time slot 7 in T_1 and T_3. While these two trajectories do not have time shifts, there is a spatial bias since the location of data point III in Fig. 1a is clearly different from that of data point IV in Fig. 1c. Additionally, trajectory data may also exhibit the silent duration phenomenon that, to the best of our knowledge, has not been explicitly addressed previously in the literature. The silent duration denotes a time duration when there is no data point presence due to data loss or sampling strategies employed in forming a trajectory. As shown in Fig. 1d, data points are not available at all the time slots. Thus, detailed information between two data points of the trajectories is missing. For example, the route information between data point I and data point II of trajectory T_1 shown in Fig. 1a is not available. Clearly, the movement paths at silent durations are uncertain. While many research effort has been put forth to address the issues of uncertain data management [26], it is out of the scope of our work. Here, we aim to develop a framework to discover trajectory patterns and routes. By clustering and aggregating trajectories, our proposed framework infers movement paths at silent durations, which is very challenging due to the unique characteristics of trajectories (i.e., spatial and temporal bias, temporal shifts and silent durations).

As mentioned earlier, given the trajectories of a user, we aim to determine the frequent trajectory routes of this user in addition to his/her trajectory patterns. Consider again the three trajectories in Fig. 1 that capture the same movement behavior of a user. It is intuitively challenging to identify the actual routing path behind these three trajectories. While a considerable amount of research efforts on trajectory pattern mining have been reported in the literature [2,11,14,23], they mostly focus on discovering frequent sequences of "hot regions" where the user frequently appears, rather than piecing together the routing paths among hot regions. Even worse, due to the existence of silent durations, they are not able to generate a sufficient number of hot regions, not to mention capturing the trajectory routes among the hot regions in the discovered trajectory patterns. Furthermore, a user usually has more than one movement behaviors hidden in the logged trajectories. As a result, the hot regions identified from the whole collection of a user's trajectories may be too general to precisely represent her movement behaviors. To address this issue, an idea is to first cluster the trajectories of a user into several groups. Each group of similar trajectories is supposed to represent one movement behavior of the user. Thus, the hot regions (as well as the trajectory route) derived from trajectories in the same group are more representative of the particular movement behavior.

[1] Some trajectories of this datasets are available in http://www.cs.nctu.edu.tw/hungcc/CarWeb.htm.

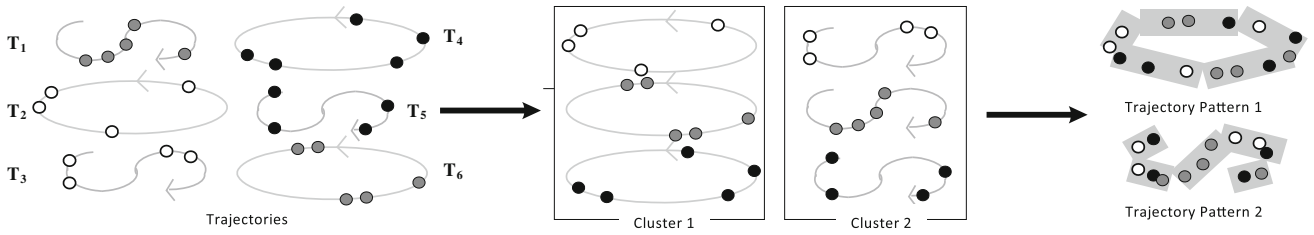

Fig. 2 Clustering and aggregating for deriving frequent trajectory routes

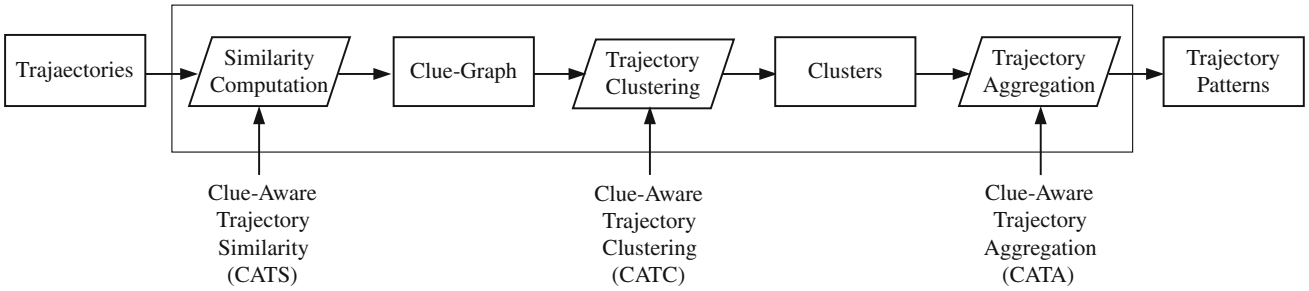

Fig. 3 Overview of our proposed framework CACT

To illustrate the research tasks and issues faced to achieve our goal, we consider Fig. 2, where six trajectories representing two movement behaviors (i.e., circle and *S*-shape movements) are given. Our goal is to derive trajectory patterns and routes as shown in the right-hand side of Fig. 2. With the presence of silent durations in these trajectories, it is very difficult to determine whether two given trajectories are similar, i.e., capturing the same movement behavior or not. Therefore, there is a need to define a new similarity measure for comparing a pair of trajectories with silent durations. Based on the similarity among pairs of trajectories, one could cluster similar trajectories into groups (as depicted in Fig. 2). Finally, data points of trajectories in the same cluster could be aggregated to infer the trajectory pattern and route. Based on the above observations, three major challenges to be addressed in this paper are summarized as follows:

- Define a new similarity measure for trajectories with silent durations.
- Design a new clustering technique that exploits the proposed new similarity measure to group trajectories with silent durations.
- Generate trajectory patterns and routes for the trajectory clusters to represent the underlying movement behavior.

In this paper, we propose a new trajectory pattern mining framework, called *Clustering and Aggregating Clues of Trajectories (CACT)*, for discovering trajectory patterns and deriving trajectory routes that represent the movement behaviors of a user. As suggested by its name, the framework is built upon an important notion of *clues*. We observe that trajectories obtained by sampling the same movement behavior are likely to consist of some spatially and temporally close data points that capture certain common partial movement behaviors of the user. These spatially and temporally close data points provide important clues hinting at the underlying movement behavior. Thus, trajectories with the close clues should be clustered together. However, even if we are able to cluster together the trajectories sampled from the same movement behavior, the trajectories in a cluster may only represent a partial trajectory route. Hence, we need to further merge several partial trajectory routes into a more complete trajectory route. To do so, one may identify clues among clusters for possible merges. Consider T_1 and T_2 in Fig. 1a, b. The data point II and data point III in T_2 are close to data points II and IV in T_3. Thus, these two trajectories are likely to refer to the same movement behavior.

Figure 3 shows the proposed framework CACT that consists of three primary components, including (i) *clue-aware trajectory similarity (CATS)*, (ii) *clue-aware trajectory clustering (CATC)*, and (iii) *clue-aware trajectory aggregation (CATA)*, corresponding to the research issues discussed earlier. Notice that CATS identifies clues among trajectories first and then uses them to measure the pair-wise similarity among trajectories. In light of CATS, CATC exploits clues between trajectories to group similar trajectories into clusters such that each cluster represents one movement behavior of the user. Specifically, given a set of trajectories with pair-wise similarity measures, a *clue-graph* is constructed to capture the similarity relationship among trajectory pairs. By deriving cliques in the clue-graph, CATC is able to obtain clusters of trajectories that have high similarity and further merge similar

clusters into a larger cluster based on the notion of clues. Finally, by aggregating trajectories in the same cluster, CATA derives the corresponding trajectory pattern and route. Notice that the proposed trajectory pattern mining framework is not only able to discover trajectory patterns but also trajectory routes to capture the frequent movement behaviors of a user. Extensive experiments using both real and synthetic datasets are conducted for performance evaluation. We compare the proposed algorithms with existing similarity measures, clustering algorithms and trajectory pattern mining algorithms. The results show that CACT can discover trajectory patterns effectively, even if trajectories with silent durations do not fully capture the complete movement route of a user.

The primary contributions made in this research study are fivefold:

- To the best of our knowledge, this is the first work that addresses the issue of silent durations in frequent trajectory pattern mining.
- A new similarity measure based on the notion of clues in trajectories of the same movement behavior is proposed.
- Based on the proposed clue-aware similarity, a clue-aware trajectory clustering algorithm is developed.
- New trajectory patterns, represented as trajectory routes, based on spatial-temporal hot regions derived in our framework are discovered.
- A comprehensive performance evaluation is conducted. Experimental results show the superiority of our proposed methods over existing works.

The rest of the paper is organized as follows. Related works are discussed in Sect. 2. Our research problem is formulated in Sect. 3. The three components of the CACT framework, i.e., CATS, CATC and CATA, are detailed in Sects. 4, 5 and 6, respectively. The experimental results are shown in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Related works

In this section, we first review some existing similarity measures for a pair of trajectories and then survey the trajectory clustering techniques. Finally, we discuss related works on trajectory pattern mining.

2.1 Similarity measures

A considerable amount of research efforts have been put forth on similarity measures for both trajectory data (e.g., LCSS [31], ERP [3], EDR [4], and RTSD [29]) and time series data (e.g., DTW [18,33], SpADe [5], and wDF[6]). Since our goal is to identify similar movement behaviors from trajectories, we compare the proposed similarity with LCSS, DTW, EDR and wDF. Table 1 summaries these existing similarity measures and their capabilities or schemes in dealing with the following five issues: (a) *temporal shifts*; (b) *time sensitivity*, (c) *space quantization*, (d) *mapping of data points*, and (e) *empty mapping*. The issue of temporal shifts, which has been discussed earlier, is handled well by all the compared similarity measures. On the other hand, time sensitivity indicates the capability of identifying two trajectories that have the same spatial feature but the occurrence time of data points in these two trajectories are different. Next, the issue of space quantization concerns how to assign weights to overcome spatial bias. Existing similarity measures assign different weights to data points from different trajectories based on their distance. The weights are then aggregated as a score as the similarity measure of two trajectories. There are two kinds of space quantization: one is the discrete scheme and the other is the continuous scheme. For the discrete scheme, if the distance between two data points is larger than a spatial threshold, the weight for these two data points is set to 0. Otherwise, the weight is set to 1. For the continuous scheme, pairs of data points with distances smaller than the spatial threshold are assigned different weights. Given two trajectories (e.g., T_i and T_j), a *mapping scheme* represents how to map one data point of trajectory T_i to data points of trajectory T_j when computing the similarity value. Mapping scheme $1-n$ denotes that a data point of T_i may be mapped to n data points of T_j; scheme $n-1$ denotes that n data points of T_i could be mapped to one data point of T_j; and scheme $1-1$ denotes that one data point of T_i can be mapped to only one data point of T_j and each data point of T_j could only be mapped to from one data point in T_i. For example, DTW has two mapping schemes $n-1$ and $1-n$. On the other hand, LCSS adopts only the mapping scheme $1-1$ because a point is only mapped to the other one point. Finally, empty mapping denotes whether a data point could be skipped from mapping to a data point.

Table 1 Comparison of similarity measurements

Functions	Temporal shifts	Time sensitive	Space quantization	Mapping scheme	Empty mapping
DTW	√		None	$1-n/n-1$	
LCSS	√	√	Discrete	$1-1$	√
EDR	√		Discrete	$1-1$	√
wDF	√	√	None	$1-1$	
CATS	√	√	Continuous	$n-1$	√

Note that RTSD and SpADe are not applicable for trajectory pattern mining. Explicitly, RTSD considers two trajectories to be similar if one can be transformed into another by translations and rotations. Thus, a hurricane moving from South to South-East is considered to be similar to another hurricane moving from West to South-West. Unfortunately,

the trajectory patterns considered in this paper are based on "physical" regions. Therefore, rotations are not allowed. As for SpADe, it uses "shapes" to distinguish whether two time series are similar or not. SpADe first finds local patterns and then calculates the distance between two time series by calculating the shortest path between matched local patterns. To recognize whether two trajectories have the same shape, SpADe tolerates certain amplitude-scaling and amplitude-shifting. However, this may not be desirable for trajectory pattern mining because two trajectories of the same shape do not represent the same movement behavior if they are not close enough in both spatial and temporal dimensions. Moreover, due to silent durations, local patterns may not always be recognized due to the lack of information in silent durations. Thus, we only consider LCSS, DTW, EDR and wDF for performance comparisons (see Sect. 7). Since the existing similarity measures are not designed to handle the silent durations of trajectories, our proposed similarity measure is uniquely different from others.

2.2 Trajectory clustering

The existing work on trajectory clustering could be classified into the following two categories: (i) clustering on the entire trajectories; and (ii) clustering on sub-trajectories. For the first category, a model-based approach for trajectory clustering is proposed in [9], which assumes that each trajectory is smooth such that each data point could be estimated by a probability density function. In this work, each cluster is assumed to have a density probability function to "generate" trajectories. The probability that a trajectory belongs to some clusters can be modeled as the mixture of these density probability functions. Therefore, each trajectory is first represented by a regression mixture model, which is used to determine its cluster memberships based on maximum-likelihood principle. However, a major deficiency of this technique is that the probability density function of each cluster and the number of clusters need to be given a priori. In [25], a density-based clustering algorithm is adapted to the trajectory data based on a simple notion of distance between trajectories. In this work, the Euclidean distance between location points at each time slot is calculated. Then, the average Euclidean distance over all time slots is obtained. Accordingly, a density-based clustering algorithm, OPTICS, is used to cluster trajectories. Moreover, an empirical comparison with several traditional k-means and hierarchical algorithms showed that OPTICS is the most suitable clustering algorithm for clustering trajectories. An advantage of this work is that the number of clusters does not need to be specified in advance. However, data points at every time slot need to be available (or be well-approximated) to compute the proposed distance function. Thus, this approach may not work very well when silent durations appear in trajectories.

For the second category, the state-of-art approach is TraClus [19]. The primary goal of this work is to discover common sub-trajectories from a set of trajectories. A key observation in this work is that clustering based on the whole trajectory could miss some common sub-trajectories, which are very useful in many applications, especially when regions of special interest are considered for analysis. Therefore, a partition-and-group framework for clustering sub-trajectories is proposed. This framework first decomposes a trajectory into a set of line segments and then groups similar line segments together into a cluster. In our work, we cluster the whole trajectories instead of sub-trajectories. For comparison, we adapt TraClus to find common sub-trajectories among trajectories and then formulate the similarity of trajectories using these common sub-trajectories. We compare our proposed clustering algorithm with this modified TraClus and find that it does not work so well when trajectories cannot fully represent the movement behaviors of a user, i.e., when the silent durations appear in the trajectories. The comparison results are shown later in Sect. 7.

2.3 Trajectory pattern mining

The problem of trajectory pattern mining has attracted a considerable amount of research efforts. Generally speaking, typical trajectory pattern mining frameworks first find hot regions and then derive the sequential relationship among these hot regions. The authors in [15,23,30] have studied the mining of spatio-temporal association rules of hot regions from a set of trajectories. In [2], the authors propose exploiting the fuzziness of locations in patterns and developing algorithms to discover spatio-temporal sequential patterns. Furthermore, the authors in [17] propose a clustering-based approach to discover moving regions within time intervals. The authors in [19,21] propose a framework to cluster
sub-trajectories, which first partition trajectories into small segments and then cluster them based on their geometric properties (i.e, angle, distance, length). In [14], the authors develop a hybrid prediction model, consisting of a vector-based and a pattern-based model, to predict the movements of users. In [11] and [10], the authors exploit temporal-annotated sequences in which sequences are associated with time information (i.e., transition times between two movements).

We mention in passing some works that deal with the trajectory convoy problem and the moving clusters problem. In the trajectory convoy problem [16], a convoy represents a group of objects that travel together for more than some minimum time duration. For example, in Fig. 4a, given the minimum time duration as 3, these five objects belong to one convoy because they travel together for a sufficient long time duration. Clearly, our trajectory patterns are different from the trajectory convoys in that a trajectory pattern

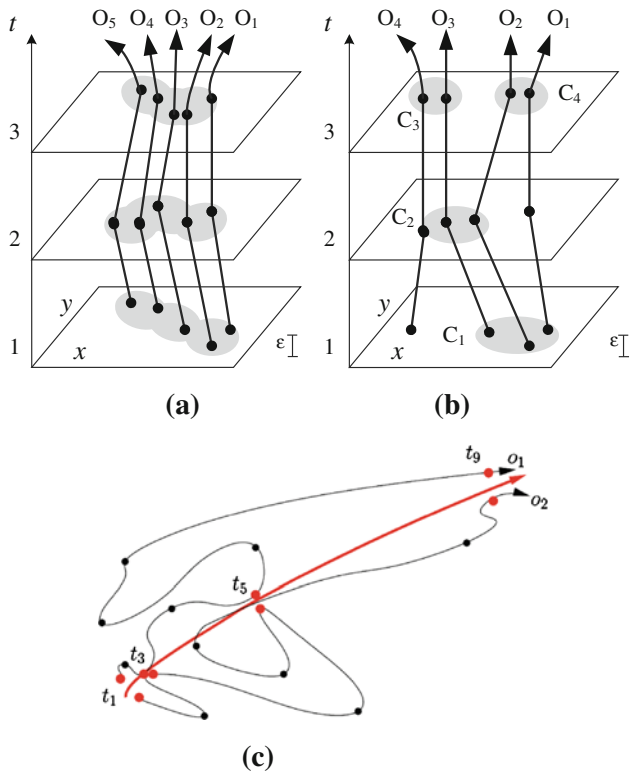

Fig. 4 Examples for trajectory convoy (**a**), moving clusters (**b**) and SWARM (**c**)

indicates sequential relationships among hot regions, whereas a convoy indicates a set of objects that have spatial relationships at some time slots. The objectives of these two problems are significantly different. Moreover, the distance between objects in a cluster at a time slot may be larger. For example, the distance between O_1 and O_5 is large, reflecting the same problem existing in hot regions determined by density-based approaches. If trajectories have silent durations, the trajectory convoy will not discover good clusters at each time slot. As for the moving clusters problem, the goal is to derive a sequence of spatial clusters to a sequence of spatial clusters at consecutive time slots such that the intersection of spatial clusters contains a sufficient number of objects. In other words, the number of objects in the intersection of spatial clusters should be larger than a specified threshold [17]. For example, in Fig. 4b, if the threshold for the portion of common objects is $1/2$, C_1 and C_2 become moving clusters. Notice that moving clusters refer to the spatial group of objects over time. However, our trajectory pattern mining is to discover frequent movement behavior. The moving clusters problem only considers the number of common objects among spatial clusters, and the objects in the set of moving clusters are not always the same. Hence, moving clusters cannot directly reflect frequent trajectory patterns. A variation of the moving clusters problem is to find a swarm [20], where a swarm is a group of objects that are in the same cluster for at least min_t timestamp snapshots. To illustrate these concepts, we borrow Figure 1 from [20]. As can be seen in Fig. 4c, the routes of O_1 and O_2 are very close at timestamps $t1, t3, t5$ and $t9$. Assume that the min_t is set to 2. Both O_1 and O_2 are in the same swarm. However, both O_1 and O_2 are different in terms of their trajectory route. Thus, the work in [20] cannot discover trajectory routes. All of the above works assume that trajectories contain detailed movement information. The most challenging issue in our paper is that trajectories may have some silent durations and thus do not always capture movement behaviors in detail. Moreover, previous works do not cluster trajectories to determine the hot regions. For trajectory clustering, the issues of dealing with trajectories that reflect only partial movement behaviors are not addressed. By fully exploiting trajectory clues, our proposed framework is not only able to accurately discover trajectory patterns, but also to identify trajectory routes that capture movement behavior very well. These features distinguish our works from others.

3 Problem formulation

A trajectory T_i of a user is a time-ordered sequence of data points, expressed as $T_i =< p_{i,1}, p_{i,2}, \ldots, p_{i,n} >$, where $p_{i,j} = (\ell_{i,j}, t_{i,j})$ represents the location of the user (i.e., $\ell_{i,j}$) at the time $t_{i,j}$, $t_{i,j} < t_{i,j+1}$ and n is the length of trajectory T_i. The location $\ell_{i,j}$ is usually a two-dimensional or three-dimensional data point. Given a set of trajectories, we intend to mine sequential relationships among hot regions, which is extended from the notion of "regions" in [2]. Explicitly, the hot region defined in this paper specifies not only a spatial area but also a temporal time interval where and when a user appears. The definition of a *hot region*, serving as the basic unit of trajectory patterns, is as follows:

Definition 1 Hot region: Given a spatial threshold ϵ and a temporal threshold τ, a hot region is a spatial-temporal prism structure r_i that satisfies the following two criteria: (1) the time interval of hot region r_i, denoted as $[r_i.S, r_i.E]$ where $r_i.S$ and $r_i.E$ are the start and end time of r_i, respectively, is required to satisfy $r_i.E - r_i.S \geq \tau$; (2) the projection of r_i in the two-dimensional XY-plane is a rectangle such that for each data point x in this rectangle, the distance between data point x and the representative line of the rectangle $\overline{L_i}$ is smaller than ϵ. The representative line of the rectangle is the regression line of the points.[2]

For example, three trajectories are shown in Fig. 5, where data points of trajectories are marked by black points at the corresponding time slots. Given $\tau = 2$ and ϵ, two hot regions r_1 and r_2, represented as two spatio-temporal prisms, are generated. The hot region r_1 is emphasized in the right of this

[2] The criterion (2) can be easily generalized to three-dimensional space.

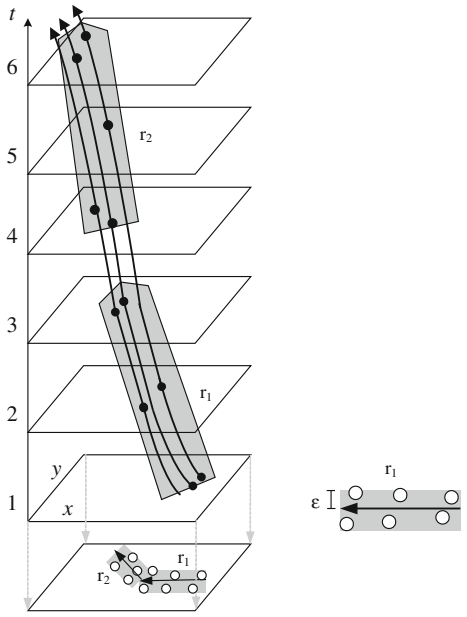

Fig. 5 An example to illustrate hot regions and trajectory patterns

figure. The line in the center of this rectangle is the representative line of this hot region. These two hot regions r_1 and r_2 have their time duration as 2 (i.e., $3-1 \leq 2$ and $6-4 \leq 2$, respectively), and the spatial projections of r_1 and r_2 are rectangles with their representative lines.

Definition 2 Trajectory patterns: A trajectory pattern TP (i.e., the frequent trajectory route) is represented as an ordered sequence of hot regions TP=$r_1 r_2 \ldots r_k$, where k is the length of TP.

From the above definitions, a trajectory pattern is represented by a sequence of representative lines, and thereby, the sequence of representative lines is viewed as a *trajectory route*. Since our goal is to derive *frequent* trajectory routes, each trajectory pattern TP is derived from a set of supporting trajectories where the number of supporting trajectories should exceed a predefined threshold min_sup. In other words, the derived trajectory pattern infers one frequent route from its supporting trajectories. Specifically, the supporting trajectories for trajectory pattern TP is defined as follows:

Definition 3 Supporting trajectory: Given a trajectory pattern $TP = r_1 r_2 \ldots r_k$, for each hot region r_i of TP, a supporting trajectory with respect to TP has at least a data point p such that the occurrence time of p is within $r_i.S$ and $r_i.E$, and $\text{dist}(p, \overline{L_i}) < \epsilon$, where $\text{dist}(\cdot,\cdot)$ is the Euclidean distance.

According to the definition of the trajectory patterns and supporting trajectories, the problem in this paper can be formulated as follows.

Problem formulation: Given a set of trajectories and four thresholds, min_sup, ϵ, τ and λ, where min_sup is for frequency, ϵ is for tolerating some spatial bias, τ is for the time duration of a hot region, and λ is for similarity values, the problem is to discover trajectory patterns in which (1) each trajectory pattern is derived from more than min_sup supporting trajectories; and (2) the similarity value between any two supporting trajectories in the same trajectory patterns is larger than or equal to λ. Moreover, the trajectory patterns are further used to infer the trajectory routes.

In summary, given a set of trajectories of a user, we target at discovering trajectory patterns that in turn are used to infer the frequent trajectory routes for the user. As discussed earlier, a user may have multiple movement behaviors. To judiciously infer the frequent trajectory routes of a user, we choose to first perform trajectory clustering to derive a set of clusters that represent different movement behaviors. Each cluster represents one trajectory pattern. Since trajectories have silent durations, we derive hot regions for each cluster by aggregating trajectories in the cluster. As pointed out earlier, the proposed Clustering and Aggregating Clues of Trajectories (CACT) framework consists of three major components, including Clue-Aware Trajectory Similarity (CATS), Clue-Aware Trajectory Clustering (CATC) and Clue-Aware Trajectory Aggregation (CATA). In the following sections, we will present the algorithmic details of each component.

4 Clue-aware trajectory similarity

In this section, we first analyze the unique characteristics of trajectories and then present our design of similarity measure. Finally, we discuss some interesting properties of the proposed clue-based similarity measure.

4.1 Characteristics of trajectories

To cluster "similar" trajectories together, it is essential to formulate a similarity measure between two trajectories. Before we proceed to present the clue-based similarity measure, we summarize some characteristics of trajectories as follows:

– **Spatial and temporal bias:** In practice, trajectories are obtained by trajectory acquisition devices or schemes, which unfortunately may introduce spatial and temporal bias to data points of trajectories. For example, the position accuracy of GPS (Global-Position System) has inherent spatial bias. Moreover, the occurrence times of data points sampled from exactly the same movement behavior are not always the same. Consider a worker who goes to his office from his home at 8:00 am every day. Data points of trajectories recording this movement behavior usually do not have the same occurrence time. One reason

is that the positioning device takes some time to determine the position. Thus, even if this user leaves his home at 8:00 am every day, data points of trajectories may have some temporal bias.
- **Temporal shifts:** Due to varied speeds and delays of user movements, trajectories may have temporal shifts. For example, although a user follows the same movement path to his office every day, some sub-trajectories have shifted occurrence times.
- **Noise:** Positioning devices can be easily affected by environmental factors, such as buildings, shelters, and weather. Hence, data points of trajectories usually have some *noises*. Noises represent some abnormal sampled locations that do not make sense to describe the user movements.
- **Silent duration:** The length of a trajectory is mainly decided by the time and the sampling rate. For the same movement path, even if the same sampling rate is used, trajectories collected may still have different lengths due to environmental factors (e.g., the weather) and the limitation of position devices (i.e., the capability of positioning devices in computing and networking). In this paper, a silent duration refers to a time duration when there are no logged data points about user movements.

4.2 Design of the clue-aware trajectory similarity

Trajectories that capture the same movement behavior of a user are likely to have some "clues" referring to those spatially and temporally co-located data points among trajectories. Since trajectories may have silent durations, these spatially and temporally co-located data points should be carefully identified and utilized to infer trajectory routes. Thus, given two trajectories, the Clue-Aware Trajectory Similarity (CATS) aims to couple as many spatially and temporally co-located data points between two trajectories as possible. CATS overcomes the impact resulting from the aforementioned characteristics of trajectories via a spatial decaying function, clue scores of data points, and a new mapping scheme.

To identify whether two trajectories come from the same movement behavior, we could observe whether there are many spatially and temporally co-located data points between two trajectories. The concept of clues is to evaluate how many such co-located points exist between two trajectories. To achieve this goal, some technical issues should be dealt with. First, for any two points in the different trajectories, the closer two points are, the stronger clues they reveal. That is, the movement routes of two trajectories tend to pass a similar area. Moreover, two co-located points reveal some clues only if their occurrence times are close enough. Due to the nature of temporal shifting, two points may appear in a similar area with a certain time delay. Thus, we should tolerate such temporal shifting when identifying the clues between two trajectories. To conclude, by tolerating certain temporal shifting, the strength of clues between two trajectories is decided not only by the number of spatial co-located points but also by how close they are. Thus, each component of CATS in the following is designed to capture the strength of clues between two trajectories.

Since trajectories usually contain spatial bias, we first use a spatial decaying function to measure the degree of bias as follows.

Definition 4 Spatial Decaying Function: Given a spatial threshold ϵ and two data points $p_{i,\ell} = (l_{i,\ell}, t_{i,\ell})$ and $p_{j,k} = (l_{j,k}, t_{j,k})$ from two trajectories (i.e., T_i and T_j), a spatial decaying function for two points $p_{i,\ell}$ and $p_{j,k}$ is defined as

$$f_\epsilon(p_{i,\ell}, p_{j,k}) = \begin{cases} 0 & \text{, if } \mathrm{dist}(p_{i,\ell}, p_{j,k}) > \epsilon \\ 1 - \frac{\mathrm{dist}(p_{i,\ell}, p_{j,k})}{\epsilon} & \text{, otherwise} \end{cases}$$

where $\mathrm{dist}(\cdot, \cdot)$ denotes Euclidean distance between two data points.

The value of spatial decaying function ranges from 0 to 1. Obviously, the closer the two data points, the larger the value is. If the locations of two data points are exactly the same, the value is 1. On the other hand, if the distance between two points is greater than ϵ, the value is 0. For example, Fig. 6 shows two trajectories T_1 and T_2, where the underlying gray lines are actual movements and the circles represent the data points of T_1 and T_2. The underlined number of each data point is the occurrence time of this data point. Consider $\epsilon = 4$ and Euclidean distance as the distance function. Given two points $p_{1,2} = (2, 2, 3)$ and $p_{2,1} = (3, 3, 3)$, it can be derived that $f_4(p_{1,1}, p_{2,4}) = 1 - \frac{\sqrt{2}}{4} = 0.65$. On the other hand, given $p_{1,4} = (7, 4, 9)$ and $p_{2,1} = (3, 3, 3)$, we can derive that $f_4(p_{1,4}, p_{2,1}) = 0$ since $\mathrm{dist}(p_{1,4}, p_{2,1}) > \epsilon = 4$.

In the spatial decaying function, a parameter ϵ is given to tolerate the spatial bias and shifting of data points.[3] Basically, the spatial decaying function performs a continuous space quantization (i.e., from 0 to 1), which reflects the closeness between two data points, in contrast to the discrete space quantization employed in LCSS and EDR (i.e., 0 or 1). For example, if the ϵ is set as 10 m, consider two cases: (i) two points with a distance of 1 m, and (ii) two points with a distance of 9 m. As aforementioned, the closer two points are, the stronger clues they have since they are highly likely to co-locate at the nearby area. Therefore, in this setting, case (i) reveals a stronger clue than case (ii). LCSS and EDR do not distinguish these two cases since the distances in both cases are smaller than 10 m.

According to the spatial and temporal information of data points, we give a score for data points with respect to a

[3] Since the parameter ϵ will decide the size of hot regions in trajectory patterns, this parameter should be set according to application requirements (the desirable size of hot regions).

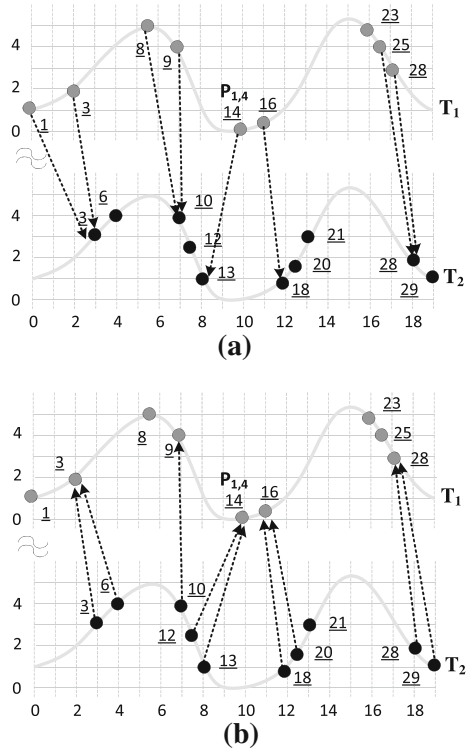

Fig. 6 An illustrative example for clue-aware similarity of T_1 and T_2. **a** $CATS_{4,4}(T_1, T_2)$. **b** $CATS_{4,4}(T_2, T_1)$

reference trajectory. For a data point, there are many possible way to evaluate the clues between this point and the reference trajectory. An obvious way is to map this data point to the closet point on the reference trajectory by tolerating some temporal shifting. That is, a data point is aligned to the point that can reveal the strongest clues in the reference trajectory. To realize this idea, we define the clue score of data points with respect to the reference trajectory. Specifically, given a data point $p_{i,\ell} \in T_i$ and a reference trajectory T_j, the clue score of data point $p_{i,\ell}$ with respect to trajectory T_j is used to identify the best mapping point on T_j for $p_{i,\ell}$ in spatial and temporal dimensionality.

Definition 5 Clue score of data points: Given a point $p_{i,\ell}$, a reference trajectory T_j, a spatial threshold ϵ, and a temporal threshold τ, the clue score of data point $p_{i,\ell}$ to trajectory T_j is defined as $score_{\epsilon,\tau}(p_{i,\ell}, T_j) = \max\{f_\epsilon(p_{i,\ell}, p_{j,k}) | p_{j,k} \in T_j \text{ and } t_{j,k} \in [t_{i,\ell} - \tau, t_{i,\ell} + \tau]\}$.

Figure 6 illustrates the computation of clue scores. Assume that $\epsilon = 4$, $\tau = 4$. Figure 6a shows two trajectories, T_1 and T_2 where the underlying gray line is the real movement. Consider a data point $p_{1,5}$ of T_1 as an example. The clue score of data point $p_{1,5}$ with respect to trajectory T_2 finds the best mapping data points of T_2 within a time interval between $14 - 4$ and $14 + 4$. As shown in Fig. 6a, four data points, $p_{2,3}$, $p_{2,4}$, $p_{2,5}$, and $p_{2,6}$ of T_2, are possible mapping data points for $p_{1,5}$ since their respective times are within $14 - 4$ to $14+4$. Since $f_4(p_{1,5}, p_{2,5}) = 1 - \frac{\sqrt{5}}{4} = 0.44$ is the largest value among that of all other points, the clue score of $p_{1,5}$ with respect to trajectory T_2 is thus $score_{\epsilon=4,\tau=4}(p_{1,5}, T_2) = 0.44$.

In the definition of clue scores, a temporal parameter τ is used to retrieve data points of trajectories whose occurrence times are within a particular time interval. Using this parameter, our proposed CATS can deal with temporal shifting. Since the times of data points to be mapped are constrained by the temporal parameter, the clue score is sensitive to time. Likewise, the clue score is also sensitive to locations. However, it still tolerates spatial and temporal biases within the degree specified by spatial and temporal thresholds.

In light of the clue score defined for data points, we define the clue-aware similarity between two trajectories as follows.

Definition 6 Clue-aware trajectory similarity: Given a spatial threshold ϵ and a temporal threshold τ, the clue-aware trajectory similarity from T_i to T_j is defined below:

$$CATS_{\epsilon,\tau}(T_i, T_j) = \frac{1}{|T_i|} \times \sum_{p_{i,\ell} \in T_i} score_{\epsilon,\tau}(p_{i,\ell}, T_j).$$

For example, let $\epsilon = 4$ and $\tau = 4$. The arrows in Fig. 6a show the mapping relationships from each data point of T_1 to data points of T_2. Consequently, the clue-based similarity measurement from T_1 to T_2 is derived as $CATS_{4,4}(T_1, T_2) = \frac{1}{9} \times (score(p_{1,1}, T_2) + score(p_{1,2}, T_2) + \cdots + score(p_{1,9}, T_2)) = 0.58$. The algorithmic form for CATS is listed in Algorithm 1.

Algorithm 1: CATS: Clue-Aware Trajectory Similarity Algorithm

Input : Trajectories: T_i, T_j; Thresholds: ϵ, τ
Output: CATS value: $CATS_{\epsilon,\tau}(T_i, T_j)$

```
TotalScore ← 0;
foreach p_{i,k} ∈ T_i do
    foreach p_{j,ℓ} ∈ T_j do
        ClueScore ← 0;
        if |t_{i,k} − t_{j,ℓ}| ≤ τ then
            ClueScore ← max(ClueScore, f_ε(p_{i,k}, p_{j,ℓ}));
    TotalScore ← TotalScore + ClueScore;
return TotalScore / length(T_i);
```

Time complexity: Let $|T_i|$ and $|T_j|$ be the number of data points in two trajectories T_i and T_j. For every point in T_i, it is necessary to compute the ClueScore with all the possible matching points in T_j. Therefore, in the worst case, the time complexity for computing the CACT value for these two trajectories is $O(|T_i||T_j|)$.

4.3 Properties of clue-based similarity measurements

From the definition of clue scores, both ϵ and τ thresholds are used to overcome spatial and temporal biases. Moreover, these two thresholds could deal with both the spatial and temporal shiftings. Since noisy data result in a larger distance value, our spatial decay function can easily filter out noises. For the mapping scheme, our clue-based similarity measurement allows many data points to map to the same data point on the reference trajectory. Consider the clue-based similarity from T_i to T_j as an example. It is possible that some data points of trajectory T_i map to the same data point on T_j if the mapped data point of T_j is not filtered by both the spatial and temporal thresholds ϵ and τ. This mapping scheme is referred to as n-to-1 mapping (abbreviated as $n-1$). If data points on T_i have higher clue scores to T_j than vice versa, the data points of T_i in fact provide more detailed information about the movement behavior than T_j does. Thus, these data points are very helpful for dealing with the silent duration in T_j. Furthermore, via the spatial and temporal thresholds, data points of T_i may not get mapped to any data point on T_j. In that case, the clue score is 0. Given two trajectories of different lengths that have some clues, our CATS may still derive a high clue score for these two trajectories, showing that CATS is able to overcome silent durations of trajectories.

Note that CATS has an asymmetry property. For example, in Fig. 6, $CATS_{4,4}(T_1, T_2) = 0.57$ is not equal to $CATS_{4,4}(T_2, T_1) = 0.65$. In Fig. 7, it can be seen that T_3 provides only a limited number of clues for the movement behaviors. On the other hand, with a large CATS values, trajectory T_4 provides more detailed movement information for T_3. Hence, this asymmetry property of CATS is helpful for identifying relationships between two trajectories. Based on our observations, there are two relationships between two trajectories: the first one is that both of the two trajectories can provide sufficient clues to each other. The second case is that from the perspective of T_i, trajectory T_i is likely to provide some detailed sub-trajectories to trajectory T_j, but from the perspective of T_j, T_j does not have a similar movement behavior with T_i. This asymmetry property reflects two kinds of relationships between two trajectories. The first relationship usually happens when the silent durations of two trajectories are distributed in a similar way and the two trajectories have only some spatial and temporal shifting. For the first case, these two trajectories can be recognized as having the same movement behavior. Figure 6 illustrates such a case. Figure 6a shows that $CATS_{4,4}(T_1, T_2) = 0.58$ and Fig. 6b shows that $CATS_{4,4}(T_2, T_1) = 0.65$. These two values show that both T_1 and T_2 have almost the same number of clues to each other. Thus, the two trajectories are very likely to follow the same movement behavior. For the second relationship, one trajectory may be part of the other trajectory. For example, in Fig. 7, the actual movements of T_3 and T_4 are the same (i.e., the underlying gray lines). Figure 7a shows that the mapping data points of T_4 have clues to T_3. However, in Fig. 7b, most data points of T_3 have no clues to T_4. In this case, we can see that by compensating T_3 with the data points of T_4, the movement behavior can be revealed in more detail.

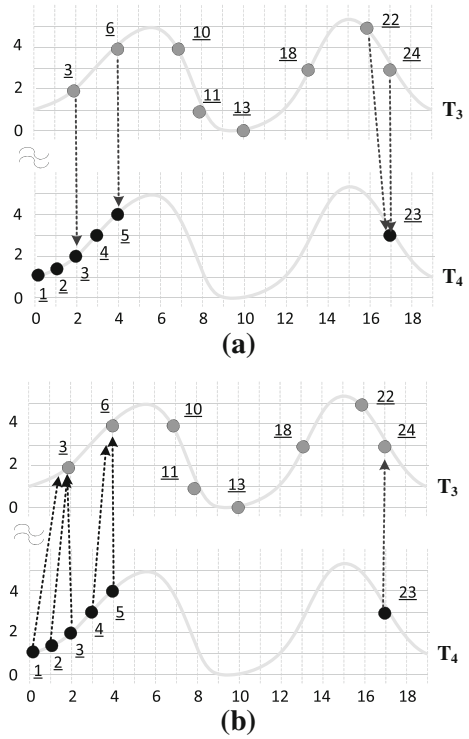

Fig. 7 An example to show the asymmetric property of CATS. **a** $CACT_{4,4}(T_3, T_4)$. **b** $CACT_{4,4}(T_4, T_3)$

With the design of clue-aware trajectory similarity, we are able to compute the pair-wise clue-aware similarity values for a set of trajectories. Due to silent durations of trajectories, though trajectories have strong clues, trajectories are likely to represent some partial movement behavior. One should fully utilize clues among trajectories to infer the complete trajectory route. Thus, in the next section, we propose a clue-aware trajectory clustering algorithm to cluster similar trajectories into groups, where each group represents one frequent movement behavior.

5 Clue-aware trajectory clustering algorithm

In this section, we describe the Clue-Aware Trajectory Clustering (CATC) algorithm for clustering trajectories based on CATS (clue-aware trajectory similarity) values.

5.1 Design of clue-aware trajectory clustering algorithm

Algorithm CATC consists of three phases: (1) clue-graph generation phase, (2) core set identification phase, and (3)

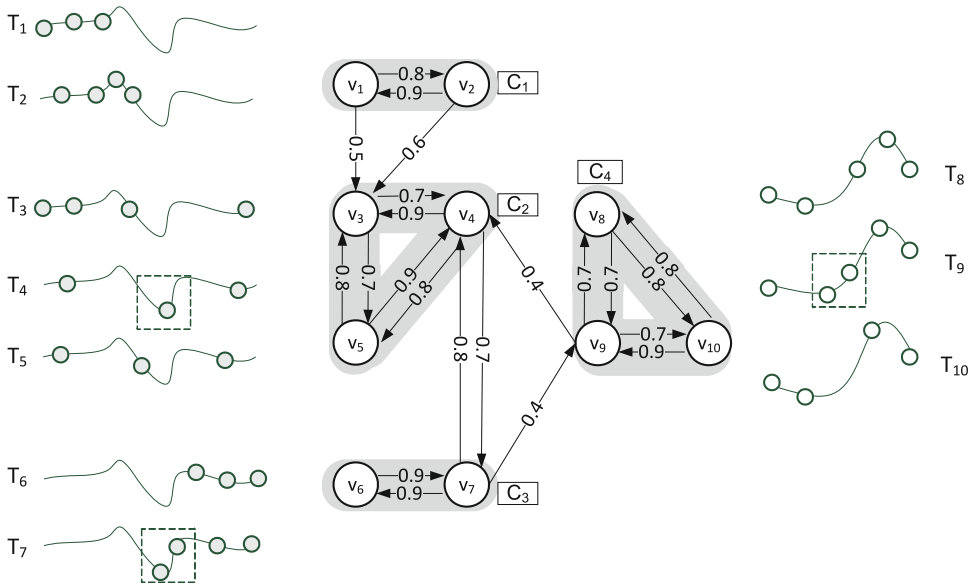

Fig. 8 An illustrative example for CATC

cluster discovery phase. Explicitly, in the clue-graph generation phase, a graph structure is used to represent CATS values among every pair of trajectories. In the core set identification phase, we cluster trajectories that have sufficiently large CATS values to each other as core sets. According to the core sets derived, in the cluster discovery phase, the core sets are merged as clusters. As such, each cluster represents one movement behavior. Clusters that have a sufficient number of supporting trajectories are identified as frequent movement behaviors. The details of the algorithm are described as follows:

Phase 1: Clue-graph generation phase

Given a set of trajectories, the CATS values between any two trajectories can be computed pairwisely. Then, a graph structure can be used to represent their clue-aware trajectory similarities. The definition of a clue-graph is given below:

Definition 7 Clue-graph: Given a set of trajectories $T = \{T_1, T_2, \ldots, T_n\}$ and a threshold λ, a clue-graph is a weighted directed graph $G = (V, E)$. In the clue-graph G, a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of all trajectories and a set of edges is defined as $E = \{(v_i, v_j) | CATS_{\epsilon,\tau}(v_i, v_j) \geq \lambda\}$ where an edge (v_i, v_j) is weighted as $CATS_{\epsilon,\tau}(v_i, v_j)$.

Figure 8 shows a clue-graph with $\lambda = 0.4$. A clue-graph is used to represent the strength of clues between trajectories. There is no edge between two vertices if the CATS values between them is smaller than a threshold λ. Once two vertices have no edge in the clue-graph, these two trajectories do not have enough clues (i.e., the spatially and temporally co-located points) to show they follow the same movement behavior. On the other hand, once the CATS values of two vertices exceed the threshold λ, the corresponding trajectories can be viewed as having enough clues to show that they follow the same movement behavior. In this case, two trajectories are called to have strong clues between them. In the clue-graph, we could further define a directly clue-reachable relationship among trajectories as follows.

Definition 8 Directly clue-reachable: A vertex v_i is directly clue-reachable to a vertex v_j, denoted as $v_i \rightsquigarrow v_j$, if $(v_i, v_j) \in E$.

Phase 2: Core set identification phase

In light of the clue-graph and Definition 8, in this phase, we aim to derive core sets in which trajectories in the core set have strong clues to each other. Note that with our clue-aware similarity measurement, if the CATS values of two trajectories exceed a threshold in both directions, these two trajectories likely infer the same movement behavior. In other words, these two trajectories are directly clue-reachable to each other in the clue-graph, i.e., they are one-hop neighbors. To capture trajectories with strong clue correlations, we define the notion of *core set* as follows.

Definition 9 Core set: Given a clue-graph $G = (V, E)$, a core set is a directed complete subgraph of G, where any two vertices v_i and v_j in a core set are directly clue-reachable to each other.

Each vertex in the core set has high clue values with respect to other vertices, indicating that these vertices in this core set capture the same movement behavior. Clearly, these core sets are viewed as seeds and these seeds could further merge with nearby seeds in the clue-graph. The detailed merging procedure among core sets is presented later. According to Definition 9, core sets could be perceived as cliques in the clue-graph. Thus, we may adopt a clique-covering algorithm to derive core sets. Since most clique-covering algorithms are

executed in an undirected graph [12], to facilitate the generation of core sets by using existing clique-covering algorithms, a Strong Clue-Graph (abbreviated as SC-G) from clue-graph G is defined as follows.

Definition 10 Strong clue-graph: Given a clue-graph $G = (V, E)$, a Strong Clue-Graph, denoted as SC-G, is an undirected graph, represented as SC-G$= (V, E')$, where $(v_i, v_j) \in E'$ if both $(v_i, v_j) \in E$ and $(v_j, v_i) \in E$.

By performing an existing clique-covering algorithm in SC-G, a set of core sets is derived. Then, vertices in the same core set are labeled in the clue-graph as well. Figure 8 shows an example of trajectories with their CATS values and a set of cliques $\{C_1, C_2, C_3, C_4\}$, where vertices in the same shaded region belong to a clique. As such, these cliques are viewed as core sets. The trajectories of each core set intend to represent similar movement behaviors because they have many co-located points to each other. For example, in the core set C_1, all of the data points of T_1 and T_2 are at the beginning of the movement route (in gray underline). Therefore, in the same core set, these two trajectories are likely to represent the same movement behavior.

Phase 3: Cluster discovery phase

Although each core set refers to one movement behavior, the movement behaviors of some core sets may be merged into a more complete movement behavior. For example, consider the core sets C_1 and C_3 in Fig. 8. The core set C_1 indicates that this user moves at the beginning of the route (i.e., most of the data points of these trajectories in C_1 are at the beginning of the route), whereas the core set C_3 represents the movement behavior that this user tends to exhibit at the end of the route. In this case, once we could merge the movement behaviors of C_1 and C_3, we can derive one movement behavior that is much closer to the actual route. Therefore, in this phase, the main goal is to use clues between core sets to infer which core sets can be merged into a cluster that represents the same movement behavior.

Core sets may be merged with other core sets as candidate clusters if two core sets have some reachable relationships, where candidate clusters are likely to infer one complete moving behavior. Since a core set may have reachable relationships with more than one other core set, one should judiciously decide which core sets are selected for merging. To infer whether two core sets capture the same movement behavior or not, the number and the weights of edges between two core sets should be considered. Intuitively, if both the number of edges among the vertices of two core sets and the edge weights are large, these two core sets are likely to reflect the same movement behavior. Furthermore, two core sets may still have clue-reachable relationships via other core sets between them. To define the clue-reachable relationship among two core sets (referred to as *clue-connected*), we should define a clue-reachable relationship between two vertices as follows.

Definition 11 Clue-reachable: A vertex u is clue-reachable to a vertex v, denoted as $u \rightsquigarrow^* v$, if there exists a chain of vertices $v = v_1, v_2, \ldots, v_n = u$ such that $v_i \rightsquigarrow v_{i+1}$ for all $i = 1, 2, \ldots, n-1$.

With the definition of clue-reachable, we could define the clue-connected relationship between two core sets as follows.

Definition 12 Clue-connected: Given two core sets C_u and C_v, C_u can clue-connect to C_v, denoted as $C_u \Rightarrow C_v$, if there exists a core set C_w, possibly coinciding with some of C_u or C_v, such that $x \rightsquigarrow^* y$ for all $x \in C_u$ and for some $y \in C_w$, and $y' \rightsquigarrow^* z$ for all $y' \in C_w$ and for some $z \in C_v$.

For example, in Fig. 8, C_1 can clue-connect to C_2. Let $C_v = C_1$, $C_w = C_1$, and $C_u = C_2$. We have all of the vertices in C_1 clue-reachable to some of the vertices in C_1 since C_1 is a core set, and $v_1 \rightsquigarrow^* v_3$ and $v_2 \rightsquigarrow^* v_3$. Two clue-connected core sets demonstrate the same movement behavior if these two core sets have connected core sets between these two core sets. Clearly, if two core sets have a clue-connected relationship, they should be grouped in the same cluster.

Thus, the result of algorithm CATC is a set of clusters consisting of core sets and the number of vertices in each cluster is larger than min_sup. To facilitate our presentation, a candidate cluster is used to represent our merging results of core sets. Initially, each core set is viewed as one candidate cluster. We iteratively merge candidate clusters until no further merge operation is needed. One criterion for stopping this merge operation is to measure the quality of cluster results. Traditional clustering algorithms use *cohesion* and *separation* to evaluate the quality of clusters. However, since our clustering algorithm is performed on the clue-graph, we develop clue-cohesion and clue-separation as follows.

Definition 13 Clue-cohesion: Given a candidate cluster K, the clue-cohesion of K, denoted by $CCOH(K)$, is defined as the minimum weight that for every core set $C_i \in K$, there exists a core set $C_j \in K$ such that $C_i \Rightarrow C_j$.

Definition 14 Clue-separation: Given two candidate clusters K_m and K_n, the clue-separation from K_m to K_n, denoted $CSEP(K_m, K_n)$, is defined as the total weight of all edges from K_m to K_n.

For example, consider C_2 and C_4 in Fig. 8. If we want to make $C_4 \Rightarrow C_2$, two extra edges (e.g., (v_8, v_3) and (v_{10}, v_5)) with the total weight $2 \times \lambda = 0.8$ should be added since there already exists an edge (v_9, v_4) from C_4 to C_2. Thus, if a cluster K contains C_2 and C_4, the clue-cohesion of this cluster can be derived as $CCOH(K) = 0.8$. The clue-separation value is $CSEP(C_4, C_2) = 0.4$ since there is one edge

(v_9, v_4) from C_4 to C_2 with the total weight $\lambda = 0.4$. Note that the clue-cohesion $CCOH(K_m, K_n)$ is zero if K_m can clue-connect to K_n. As such, if two core sets from different candidate clusters have a clue-connected relationship, these two candidate clusters are likely to be merged. By merging these two candidate clusters, a larger candidate cluster has more trajectories for inferring the whole frequent trajectory route.

With the definitions of clue-cohesion and clue-separation, we intend to derive a set of clusters such that the cluster result should have smaller clue-cohesions and clue-separations among clusters. In other words, trajectories within the same cluster have as many clues as possible and trajectories from different clusters have as few clues as possible. Therefore, the desired cluster result is defined as follows:

Definition 15 Objective function for clusters: Given a clue-graph $G = (V, E)$, the clustering algorithm aims to derive a set of clusters $\mathcal{K} = \{K_1, K_2, \ldots, K_m\}$ such that (1) K_i contains a set of core sets, (2) minimize $(\sum_{K_i \in \mathcal{K}} CCOH(K_i) + \sum_{K_i, K_j \in \mathcal{K}} CSEP(K_i, K_j))$, and (3) $|K_i| \geq min\_sup$ for all $K_i \in K$.

Based on the above objective function for clustering, we design a benefit function to evaluate whether merging two candidate clusters is able to reduce the value of the objective function or not. The benefit function is formulated as follows.

Definition 16 Benefit function: Given two candidate clusters K_m and K_n, the benefit function is defined as $Benefit(K_m, K_n) = DesCSEP(K_m, K_n) - IncCCOH(K_m, K_n)$. $DesCSEP(K_m, K_n) = (CSEP(K_m, K_n) + CSEP(K_n, K_m))/2$ and $IncCCOH(K_m, K_n) = CCOH(K_m) + CCOH(K_n) + \sum_{C_i \in K_m} \min_{C_j \in K_n}\{I(C_i, C_j) \times \lambda\}$, where $I(C_i, C_j)$ denotes the number of vertices that have no edge from core set C_i to C_j and λ is the threshold used in the clue-graph.

Generally speaking, merging two candidate clusters will increase the clue-cohesion while decreasing the clue-separation. Intuitively, if merging two candidate clusters could lead to a greater decrease in clue-separation than increase in clue-cohesion, the merging operation is able to minimize the objective function, which brings the benefit of achieving a better clustering result. Consequently, the benefit function is to evaluate whether merging two candidate clusters is able to reduce the value of the objective function (i.e., the sum of the clue-cohesion and the clue-separation after merging two candidate clusters). Assume that we intend to derive the benefit of merging two candidate clusters K_m and K_n. The first term of the benefit function (i.e., DesCSEP) represents how much clue-separation could be reduced by merging K_m and K_n. The average of the clue-separations between two candidate clusters aims to prevent the scenario in which only one cluster has many edges to the other, but there is no edge in reverse. For example, in Fig. 8, assume that four core sets are candidate clusters and a candidate cluster set, denoted as $\mathcal{K}$, $\mathcal{K} = \{K_1 = C_1, K_2 = C_2, \ldots, K_4 = C_4\}$. Then, we illustrate how to derive $Benefit(K_2, K_3)$. It can be derived that $CSEP(K_2, K_3) = 0.7$ since the only edge which crosses from K_2 to K_3 is (v_4, v_7) with weight 0.7. Similarly, we can obtain that $CSEP(K_3, K_2) = 0.8$. Thus, after merging K_2 and K_3, two edges (v_4, v_7) and (v_7, v_4) are in the same cluster such that the clue-separation is decreased by $0.7+0.8 = 1.5$. Thus, the value of $DesCSEP(K_2, K_3)$ can be derived as $1.5/2 = 0.75$. The second term of the benefit function (i.e., IncCCOH) is to evaluate the amount of increase in clue-cohesion by merging K_m and K_n. For the two candidate clusters K_m and K_n, they need $CCOH(K_m)$ and $CCOH(K_n)$ to make their core sets clue-connected. The last term of IncCCOH refers to the minimum weight that every core set in K_m can clue-connect to some core set in K_n. Specifically, if a core set C_i is required to clue-connect to the other core set C_j, every vertex in C_i should have an edge to some vertex in C_j. One could imagine that the cost of adding an edge is λ since there is an edge between two vertices if the CATS value between them is at least λ. Therefore, to make C_i clue-connect to C_j, the total cost can be derived by multiplying λ and the number of vertices that have no edge from core set C_i to C_j. Recall the example above where two candidate clusters $K_2 = \{C_2\}$ and $K_3 = \{C_3\}$ are given. Since these two candidate clusters contain only one core sets, we can obtain that $CCOH(K_2) = 0$ and $CCOH(K_3) = 0$. If we consider the case that K_2 needs to clue-connect to K_3, extra two edges from C_2 to C_3 need to be added (e.g., adding $(v_5, v_6$ and $(v_3, v_6))$. Therefore, the total increase in the clue-cohesion value is $IncCCOH(K_2, K_3) = 0+0+2\times\lambda = 0.8$. Finally, $Benefit(K_2, K_3) = 0.75 - 0.8 = -0.05$, which shows that by merging K_2 and K_3, there is no benefit to minimizing the objective function.

By exploiting this benefit function, algorithm CATC iteratively selects two candidate clusters with the maximum benefit value until the value of the benefit function is smaller than zero. Once the merge operation is finished, candidate clusters that have more than min_sup vertices will become final clusters. The reason for having at least min_sup vertices is that each cluster represents one frequent movement behavior. Note that the computation of the benefit function could be implemented by dynamic programming. Since candidate clusters are expanded in a bottom-up fashion, the clue-cohesions and clue-separations of any two candidate clusters are computed in the previous rounds. Therefore, we explore a dynamic programming strategy in algorithm CATC. The algorithmic form of CATC is shown in Algorithm 2. The time complexity of CATC is analyzed as follows:

Time complexity: Let N be the number of trajectories (vertices). In line 1, CATC constructs a clue-graph that

Algorithm 2: CATC: Clue-Aware Trajectory Clustering Algorithm

Input : Trajectories: $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$; Thresholds: λ, min_sup
Output: Set of clusters: $\mathcal{K}$

1 Construct a clue-graph $G = (V, E)$ by $\mathcal{T}$ and λ; //*
2 Step 1
3 Construct a Strong clue-graph SC-$G = (V, E')$ from G; //*
4 Step 2 $C = \{C_1, C_2, \ldots, C_\ell\} \leftarrow$ a clique cover of SC-G;
5 $\mathcal{K} \leftarrow C$;
6 **foreach** $(K_m, K_n) \in \mathcal{K} \times \mathcal{K}$ **do**
7 $\quad CCOH[K_m] \leftarrow 0$;
8 $\quad CSEP[K_m, K_n] \leftarrow$ total weights from K_m to K_n;
9 $\quad$ Compute $I(K_m, K_n)$;
//Step 3
10 **repeat**
11 $\quad Benefit \leftarrow \infty$;
12 $\quad$ **foreach** $(K_m, K_n) \in \mathcal{K} \times \mathcal{K}$ **do**
13 $\quad\quad r \leftarrow -1$;
14 $\quad\quad Des \leftarrow (CSEP[K_m, K_n] + CSEP[K_n, K_m])/2$;
15 $\quad\quad Inc \leftarrow 0$;
16 $\quad\quad$ **foreach** $C_i \in K_m$ **do**
17 $\quad\quad\quad q \leftarrow \infty$;
18 $\quad\quad\quad$ **foreach** $C_j \in K_n$ **do**
19 $\quad\quad\quad\quad q \leftarrow min(q, I(C_i, C_j))$;
20 $\quad\quad\quad Inc \leftarrow Inc + q$;
21 $\quad\quad Inc \leftarrow CCOH[K_m] + CCOH[K_n] + q$;
22 $\quad\quad Benefit \leftarrow Des - Inc$;
23 $\quad\quad$ **if** $Benefit > r$ **then**
24 $\quad\quad\quad (S, T) \leftarrow (K_m, K_n)$;
25 $\quad\quad\quad Benefit \leftarrow r$;
26 $\quad$ **if** $Benefit > 0$ **then**
27 $\quad\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{S \cup T\} - S - T$;
28 $\quad\quad$ Update $CCOH$ and $CSEP$;
29 **until** $Benefit < 0$;
30 Keep $K_m \in \mathcal{K}$ if K_m contains more than min_sup vertices;

requires $O(N^2)$. In line 2 to line 3, a clique-covering algorithm is performed in a clue-graph, which takes $O(N^2)$ [12]. In line 5 to line 8, initializing table CCOH, CSEP, and I costs $O(N^2)$. The outer loop from line 9 to line 27 depends on the benefit values. The worst case is that every single vertex is a clique such that this loop executes at most $N - 1$ times. From line 11 to line 24, there is a nested loop. From line 14 to line 18, all pairs of core sets are enumerated such that there are totally $C(\ell, 2)$ combinations. From line 24 to line 26, updating the CCOH table needs $O(1)$-time, and updating the CSEP table requires $O(\ell)$-time. Therefore, the outer loop from line 7 to line 27 totally takes at most $C(N-1, 2) + C(N-2, 2) + \cdots + C(2, 2) = O(N^3)$. To sum up, the time complexity of CATC is at most $O(N^3)$.

5.2 Running example for cluster discovery phase

The execution of CATC can be best illustrated by Fig. 8. In this figure, T_1 to T_7 capture the first movement behavior and T_8 to T_{10} record the second movement behavior. Note that there are edges from v_4 to v_9, and from v_7 to v_9. The reason is that they have some nearby points as shown in the dashed square such that there are some clues between them. At the beginning, CATC finds four core sets $C = \{C_1, C_2, C_3, C_4\}$ and the vertices in the same shaded area are in the same clique. Each clique is a candidate cluster in $\mathcal{K}$, i.e., $\mathcal{K} = \{K_1 = C_1, K_2 = C_2, \ldots, K_4 = C_4\}$. To evaluate whether two candidate clusters can be merged, we first compute the benefit values for each pair of candidate clusters. For example, it could be computed that $Benefit(K_1, K_2) = (0.5 + 0.6)/2 + 0 = 0.55$ (since all vertices in K_1 are clue-reachable to v_3), $Benefit(K_3, K_2) = (0.7 + 0.8)/2 - 0.4 = 0.35$ (since v_6 in K_3 is not clue-reachable to any vertex in K_2), and $Benefit(K_4, K_2) = (0.4 + 0.4)/2 - 2 \times 0.4 = -0.4$ (since v_8 and v_{10} in K_4 are not clue-reachable to any vertex in K_2). Since the $Benefit(K_1, K_2)$ is the maximum, two candidate clusters K_1 and K_2 are merged into one larger candidate cluster $K'_1 = \{K_1, K_2\}$. Next, CATC continues to compute the benefit values between all pairs of candidate clusters. Specifically, if we intend to merge K_3 to K'_1, we could first derive $DesCSEP(K_3, K'_1) = (0.7 + 0.8)/2 = 0.75$. Then, $C_3 \in K_3$ needs the cost $1 \times 0.4 = 0.4$ to clue-connect to $C_2 \in K'_1$. Thus, $IncCCOH(K_3, K'_1) = CCOH(K_3) + CCOH(K'_1) + 0.4 = 0 + 0 + 0.4 = 0.4$. Since $Benefit(K_3, K'_1)$ is the maximum, CATC merges K_3 and K'_1 into a new candidate cluster $K'_1 = \{C_1, C_2, C_3\}$. Finally, since all the benefit values are negative, CATC terminates. Assume that the threshold min_sup is 4, then K_4 is deleted since the number of trajectories in K_4 is smaller than 4. Finally, one cluster $K'_1 = \{C_1, C_2, C_3\}$ is found.

6 Clue-aware trajectory aggregation algorithm

For each given cluster, we intend to derive hot regions in order to compose trajectory patterns. As mentioned earlier, each hot region has its representative line segment. In this section, we propose a Clue-Aware Trajectory Aggregation algorithm (abbreviated as CATA) to derive a set of representative line segments. Accordingly, a representative trajectory is determined for each core set. Representative trajectories from different core sets will then be aggregated as a sequence of representative line segments.

6.1 Determine a representative trajectory of a core set

Trajectories in the same core set have mutually high clues to show that these trajectories capture the same movement behavior. As such, for each core set, the corresponding representative trajectory (i.e., a sequence of representative data points) will be determined. To determine the data points of a representative trajectory, for core set C_i, a set of data points

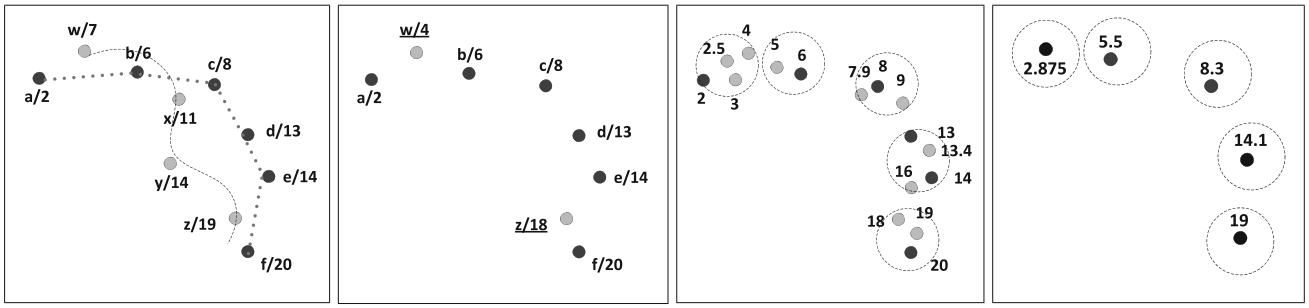

Fig. 9 An example of generating candidate line segments in a core set. **a** Two trajectories of core set C_i. **b** PS_i after merging two trajectories. **c** An example of point-clusters in PS_i. **d** Representative trajectory of core set C_i

derived by aggregating the data points of the trajectories within C_i is denoted as PS_i. Among the trajectories in C_i, a base trajectory is first selected and all of its points are then added into PS_i. The remaining trajectories contribute their data points to PS_i if these data points from other trajectories satisfy both spatial and temporal constraints. Finally, all points in PS_i are utilized to derive the representative trajectory.

The base trajectory has the most clues from the other remaining trajectories compared to the other trajectories in a core set. Therefore, in a core set C_i, the trajectory with the maximum sum of incoming edge weights is selected as a base trajectory. After selecting the base trajectory, data points of the other trajectories are considered for inclusion into the point set PS_i. To identify whether or not a point p from another trajectory should be included in PS_i, the location of p should be close to one of the line segments derived by two consecutive data points of the base trajectory. Specifically, let two consecutive data points s and e be from the base trajectory where $t_s > t_e$, and $\overline{se}$ is the line connecting (x_s, y_s) and (x_e, y_e).[4] Given a data point p and its closest line $\overline{se}$ in the base trajectory, if $\text{dist}(p, \overline{se}) < \epsilon$, then p satisfies the spatial constraint and its occurrence time will be further investigated. If $t_p \in [t_s - \tau, t_e + \tau]$, the data point p will be included in PS_i and the occurrence time of data point p is revised as $t_p = t_s + (t_e - t_s) \times \frac{\text{dist}(s,p)}{\text{dist}(s,p)+\text{dist}(p,e)}$. If not, the point p is discarded. Consider the example in Fig. 9a, where the dotted line is a base trajectory and the data points of the other trajectories are marked as gray points associated with their occurrence time. As illustrated in Fig. 9a, gray point y is eliminated, as $\text{dist}(y, \overline{cd}) > \epsilon$. Gray points w, x, and z are close to the base trajectory. We should further investigate their occurrence time. Suppose that $\tau = 2$ and the distance between a and w equals that between b and w. As the time of w is between $[2-2, 6+2]$, w is included, and the time of w is revised as $2+(6-2) \times \frac{\text{dist}(a,w)}{\text{dist}(a,w)+\text{dist}(w,b)} = 4$. On the other hand, although data point x is close to the base trajectory, the time of x is not within the interval $[6-2, 8+2]$. Thus, data point x is discarded. Given two trajectories in Fig. 9a, the point set PS_i is shown in Fig. 9b.

After examining the data points of the trajectories in core set C_i, we obtain our final point set PS_i. The next step is to determine a representative trajectory from PS_i. The determination of a representative trajectory consists of two steps: (1) to group spatially nearby points as clusters (referred to as point-clusters), and (2) in each point-cluster, the representative data point with its time stamp is determined. The first step is achieved by DBSCAN with decomposition proposed in [15], where DBSCAN is first used to find point-clusters. The point-clusters that are too large in size will then be partitioned into smaller clusters. The DBSCAN with decomposition is able to derive groups with spatially nearby points. Each point-cluster determines a representative point with its location and time stamp as the average of the coordinates and the time stamps of all points in this point-cluster. The representative trajectory is represented as $< rp_1, rp_2, \ldots, rp_n >$, where rp_k is the representative point of the k-th point-cluster. For example, Fig. 9c shows an example of point-clusters in the point set PS_i. In the first cluster, the time stamp of the representative point can be calculated as $(2+2.5+3+4)/4 = 2.875$. The representative trajectory is derived similarly, as illustrated in Fig. 9d.

6.2 Generate a trajectory pattern from a cluster

After generating the representative trajectories of core sets, the next task is to aggregate them into one final representative trajectory. Since our representative trajectory is viewed as a sequence of lines, given this final representative trajectory, we adopt DP* [24][5], a line simplification algorithm, to derive representative line segments.

[4] To facilitate our presentation, a data point p can be represented as a triple (x_p, y_p, t_p) where (x_p, y_p) is the coordinate at time t_p.

[5] Note that algorithm DP* in [24] is developed to derive a set of lines from a given set of spatial-temporal data points.

Since a cluster contains a set of core sets, and each core set has its own representative trajectory, these representative trajectories should be aggregated into one aggregated trajectory. Our idea is to first select one representative trajectory of a core set as a base trajectory. Then, we iteratively select representative trajectories of the other core sets and use them to compensate this base trajectory. Explicitly, the base trajectory is from the core set that has clue relationships with the most core sets. Once the base trajectory is determined, we then select other core sets one by one to aggregate their representative trajectories. Note that the order of aggregating trajectories is the same as the order of merging candidate clusters in algorithm CATC. Recall that the merging order is guided by the benefit function. For two candidate clusters K_i and K_j, the maximum $benefit(K_i, K_j)$ indicates that K_i can add the fewest number of edges to clue-connect to K_j. In other words, K_i is the candidate cluster that can provide the most detailed movement information to K_j. Given two candidate clusters K_i and K_j, let the representative trajectories of K_i and K_j be RT_i and RT_j. To aggregate two representative trajectories of K_i and K_j, the representative trajectory K_i should be added to the representative trajectory of K_j. For example, consider the cluster $K_1' = \{C_1, C_2, C_3\}$ in Fig. 8. As per the example in Sect. 5.2, in the first round, C_1 first merges to C_2 such that a candidate cluster $K_1' = \{C_1, C_2\}$ is obtained. In this round, the representative trajectory RT_1' is generated by using RT_2 as the base trajectory and aggregating points in RT_1 into RT_2. In the second round, C_3 merges to K_1' such that $K_1' = \{C_1, C_2, C_3\}$. Thus, RT_1' is the base trajectory and RT_3 will be aggregated into RT_1'. The final representative trajectory RT_1' is used to derive hot regions for the trajectory pattern in this cluster. Note that for each round, we need to compute the benefit values for two candidate clusters. This order list could be determined in algorithm CATC and by referring to the order list, algorithm CATA could quickly perform the aggregation of the representative trajectories.

Once the final representative trajectory is obtained, a spatiotemporal compression technique (i.e., DP* [24]) is used to determine the representative lines. Given a sequence of spatio-temporal points, DP* compresses these points into a sequence of lines such that the distance between each point to the compressed line is not farther than ϵ. Given the final representative trajectory $(p_1, p_2, \ldots, p_n)$, by exploring DP* in [24], a sequence of k representative lines $CL = \{(\ell_1, TI_1), \ldots, (\ell_k, TI_k)\}$ is derived. Each representative line is viewed as a hot region, and the trajectory pattern of a cluster is then discovered. The details of algorithm CATA are presented in Algorithm 3, and the time complexity of CATA is discussed below.

Algorithm 3: Clue-Aware Trajectory Aggregation algorithm

```
   Input  : A set of clusters: 𝒦
   Output: Trajectory pattern: R
 1 for each cluster K_ℓ ∈ 𝒦 do
 2     for each core set C_i ∈ K_ℓ do
 3         BS ← the trajectory with highest sum of incoming edge
           weights in C_i;
 4         Add BS into PS_i;
 5         for each other trajectory T in C_i do
 6             Eliminate points of T with intolerable location and
               time stamp to T_r;
 7             Add remaining points of T into PS_i;
 8         end
 9         PC ← point-clusters by execute DBSCAN with
           decomposition on PS_i;
10         for each point-cluster in PC do
11             p ← representative points;
12             Add p into rp_i;
13         end
14     end
15     RP ← ⋃_i rp_i;
16     while |RP| > 1 do
17         Remove rp_i and rp_j from RP according to the merging
           order of candidate clusters;
18         rp' ← Aggregate rp_i into rp_j;
19         Add rp' into RP;
20     end
21     Execute DP* for the representative trajectory in RP;
22 end
```

Time complexity: Let $|T|$ be the maximum trajectory length among all trajectories. The loop from line 2 to line 13 executes $|K_\ell|$ times, where $|K_\ell|$ is the number of core sets in K_ℓ. In this loop, line 3 needs a linear scan to find the base trajectory BS, which costs $O(|C_i|)$ time. The inner loop from line 5 to line 8 needs a linear scan for each trajectory for adjusting the location and time stamp, which spends $O(|C_i||T|)$-time. Line 9 needs $O(|PS_i|log|PS_i|) = O(|C_i||T|log(|C_i||T|))$ to execute DBSCAN with decomposition. Line 10 to line 13 computes the average locations and time stamps for selecting representative points. Thus, it needs at most $O(|PS_i|)$. Therefore, from line 2 to line 13, it spends $O(|C_i||T|log(|C_i||T|))$. Note that the size of RP is $|K_\ell|$. Therefore, line 16 to line 19 takes $O(|K_\ell||T|)$. Finally, the time executing DP* algorithm needs $O(|T|log|T|)$. Therefore, the total time complexity should be $\sum_\ell O(|K_\ell||T| + |K_\ell||T|lg(|K_\ell||T|)) = O(|K||T|log|K||T|)$, where $|K|$ is the number of clusters.

7 Performance evaluation

A series of experiments have been conducted to evaluate the proposed algorithms using both real and synthetic datasets. In Sect. 7.1, the experimental settings are presented. Performance comparisons of our clue-aware similarity measurement and clue-aware clustering algorithm with previous

works are described in Sects. 7.2 and 7.3, respectively. Moreover, trajectory patterns mined by our CACT and other works are analyzed in both qualitative and visualized manners in Sect. 7.4. Finally, the performance of CACT is investigated in Sect. 7.5.

7.1 Experimental environment

To validate and evaluate the techniques proposed for our trajectory pattern mining framework, both real and synthetic datasets are used. The experimental results are mainly obtained using the real datasets, whereas the sensitivity testing of the proposed algorithms also uses the synthetic datasets. The real dataset consists of trajectories obtained from the *CarWeb* platform [22]. Through the CarWeb dataset, we have access to the ground truth of user movement behaviors, which makes the quality evaluation of the discovered trajectory patterns feasible. We select trajectories capturing four types of frequent movement behaviors (as shown in Fig. 10). Note that both Type 2 and Type 4 have similar movement paths, but their times are different. Trajectories of Type 2 present a movement behavior happening on an early morning, whereas trajectories of Type 4 capture another movement behavior in the late afternoon. Table 2 summarizes some statistic information of the selected trajectories, including the number of trajectories, the length of the trajectories, and the time duration of the trajectories. While some existing trajectory datasets, e.g., hurricane and animal datasets used in [19], are available, evaluating the quality of mining results using them is difficult due to the lack of domain knowledge and ground truth in these datasets. Moreover, some trajectory datasets, e.g., the Australian Sign Language dataset that captures hand-writing trajectories [4,31], are not generated by position acquisition devices (e.g., GPS loggers) and thus are not suitable for our experiments.

To test the impact of the trajectory characteristics on the proposed algorithms, we use a synthetic trajectory dataset, generated from a given set of seed trajectories. Each seed trajectory has 3,000 to 6,000 data points, which is much longer than the length of real trajectories in CarWeb dataset. Rather than the shorter trajectories in CarWeb dataset, the longer seed trajectories can provide an other scenario to examine the effectiveness and the efficiency of the compared approaches. For each seed trajectory, we simulate a number of synthetic trajectories with spatial and temporal bias. Specifically, for data point p from a seed trajectory, a data point will have a probability of being a bias data point. The probability value is set to P_{bias}. If the data point generated is a bias data point, the location of this bias data point will be shifted from the location of p for at most r meters and its occurrence time is shifted for at most t units of time (minutes in our dataset) from the occurrence time of p. Accordingly, we randomly generate synthetic trajectories with spatial and temporal bias from a seed trajectory.

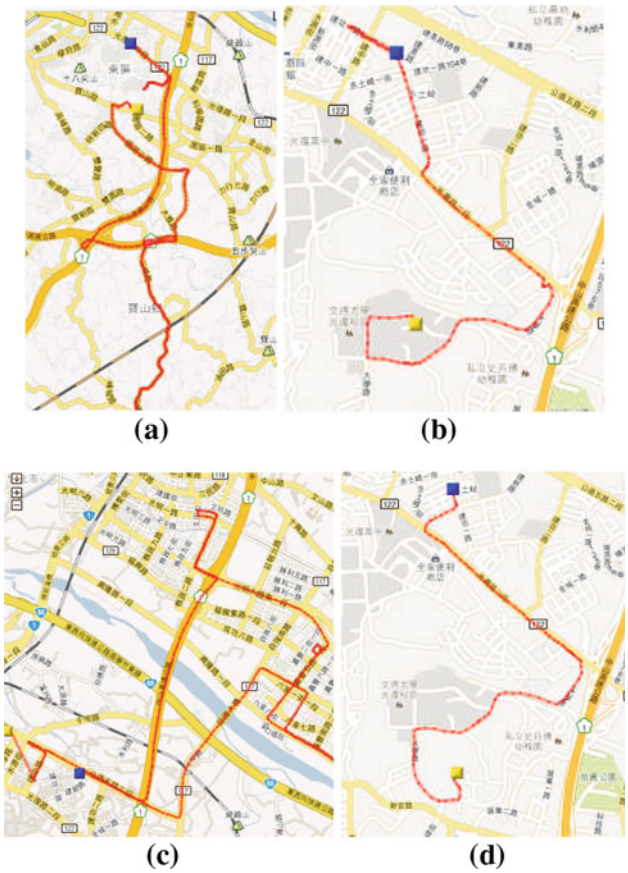

Fig. 10 Trajectories in *CarWeb* dataset. **a** Type 1. **b** Type 2. **c** Type 3. **d** Type 4

Table 2 Statistical information about the selected trajectories in CarWeb dataset

	Type 1	Type 2	Type 3	Type 4
Number of trajectories	14	13	5	13
Average length	634	128	341	142
Standard deviation in lengths	130.14	28.39	185.95	30.21
Average time duration	62	12	49	14
Standard deviation in time durations	9.27	1.98	42.26	2.1

To investigate the effect of silent durations, two parameters are used to control the distribution of silent durations: the sampling interval SI and the clue probability P_{clue}. Explicitly, a seed trajectory contains a detailed movement path of a user. To generate trajectories with silent durations, we select some data points from a seed trajectory and eliminate other data points. For each data point selected, this data point has P_{clue} to be included in the synthetic trajectory. The generation of synthetic trajectories could be best understood by Fig. 11. Figure 11a shows a seed trajectory, where the number associated with each point is its time stamp. Figure 11b

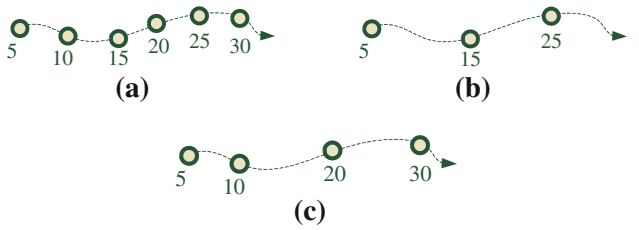

Fig. 11 Generation of trajectories with silent durations. **a** Seed. **b** $SI = 10$. **c** $P_{\text{clue}} = 0.7$

shows a trajectory generated by setting SI as 10 s, i.e., a data point is selected from the seed trajectory for every 10 s. A larger SI results in a trajectory with a longer silent durations. Figure 11c shows the synthetic trajectory generated by setting $P_{\text{clue}} = 0.7$ and $SI = 10$, where $P_{\text{clue}} = 0.7$ controls the percentage of data points selected from the seed trajectory to be 70%. The default setting of additional parameters used in our experiments are $\lambda = 0.3$, $\tau = 300$ s (which is decided by averaging the length of silent durations of all trajectories), $\epsilon = 100$ m, $min\_sup = 0.1$, $r = 300$ m, and $t = 300$ s. All the experimental results are reported by averaging the results of 50 trails that vary in SI and P_{clue}.

7.2 Performance comparison of similarity measures

In this section, we evaluate the performance of our clue-aware trajectory similarity (denoted as CATS in the figures) in comparison with a number of well-known similarity measures, including LCSS, DTW, wDF, and EDR. The spatial threshold is set as 100 m and temporal threshold threshold is set as 300 s (as needed in all compared similarity measures). To compare the performance of distance functions, the same as in [7,28], we utilize one Nearest Neighbor (abbreviated as 1NN) classifier [7]. Explicitly, given a training dataset in which data is labeled, 1NN classifier tries to predict the label as that of its nearest neighbor in the training set. Since different distance functions may have a different scale range in their similarity values, 1NN classifier could avoid this scale range problem. Moreover, in our real dataset, each trajectory is labeled as one of four types of trajectories. Thus, by exploring 1NN classifier, given one distance function and a trajectory, the label of a given trajectory is predicted by the label of its nearest neighboring trajectory. Therefore, the effectiveness of the distance functions can be reflected by the prediction accuracy for the label of the given trajectories. Due to the asymmetric property of CATS, it is necessary to define the nearest neighbor for CATS. Assume that we want to find the nearest neighbor of a trajectory T_i. All other trajectories would be mapped in a two-dimensional (2D) plane. Explicitly, a trajectory T_j could be represented as a point $(CATS(T_i, T_j), CATS(T_j, T_i))$ in this 2D plane. Then, the nearest neighbor of a trajectory T_i in CATS is the skyline point T_j with the maximum value of $(CATS(v_i, v_j) + CATS(v_j, v_i))$. The reason is that among the skyline points, we choose the trajectories that have the strongest clues with T_i as the closest neighbor of T_i. It is possible that multiple skyline points exist. In this case, we randomly select one skyline point and use its label for prediction.

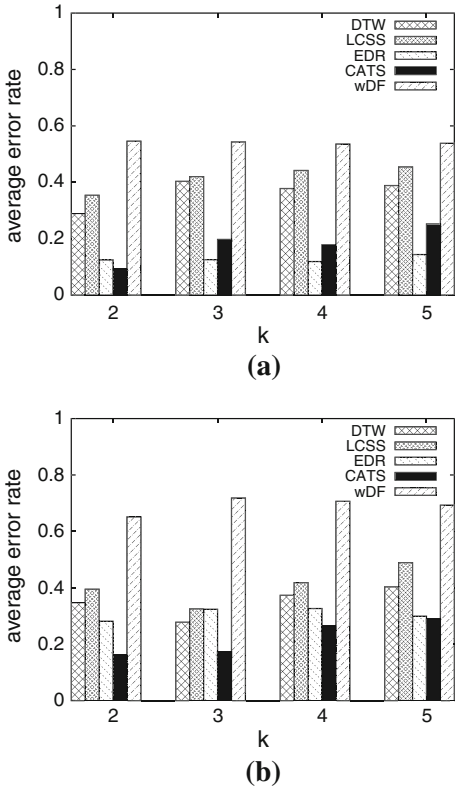

Fig. 12 Average error rates with k varied. **a** 3 types. **b** All types

The *classification error rate* is defined as the ratio between the number of incorrect predictions and the total number of trajectories. Because the performance of the 1NN classifier is highly sensitive to the given similarity measurement, the classification error rate could directly reflect the effectiveness of similarity measurements. Based on the 1NN classifier, we adopt the cross-validation approach in [7] in which training data are randomly divided into k subsets. Then, one subset is selected for testing and the other of $k - 1$ subsets are used as training sets. Finally, the average error rate of the 1NN classifier over the k cross-validation is reported.

Figure 12 shows the experimental results based on different similarity measures. This experiment tests the average error rate by (1) three types of trajectories (Type 1 to Type 3) and (2) all types of trajectories. To examine the impact of the time sensitivity of the distance function, we should include Type 2 and Type 4 trajectories that follow the same movement behavior with different times. Figure 12a depicts the experimental results with three types of trajectories (i.e., Type 1, Type 2, and Type 3). This figure shows that CATS and EDR have lower average error rates than others. Specifically,

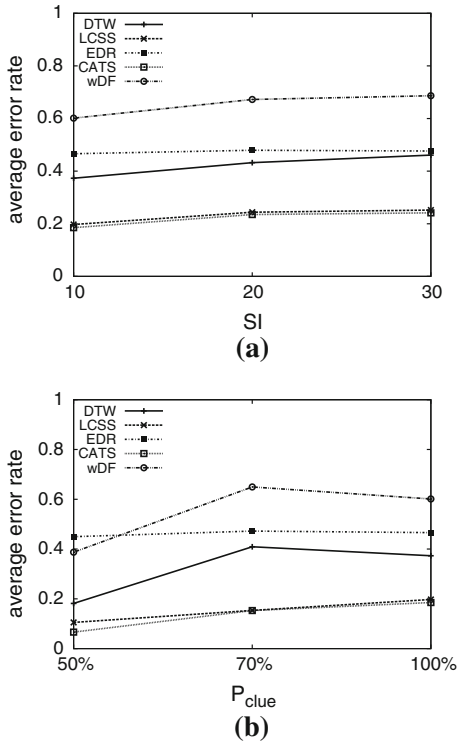

Fig. 13 Average error rates with varying the distribution of silent periods

wDF and LCSS fail to predict the correct labels because most trajectories pass many nearby locations. DTW does not perform effectively due to the large standard deviations of Type 1 and Type 3 trajectories. Because EDR accounts for not only common sub-trajectories but also gaps between trajectories, EDR can clearly distinguish trajectories with different types. Moreover, the average error rates of EDR are slightly smaller than CATS. If Type 4 trajectories are included, CATS can outperform EDR. Note that Type 2 and Type 4 trajectories have similar movements with different occurring times. Due to time sensitivity, CATS can easily distinguish Type 2 and Type 4 trajectories compared with EDR. This experiment also shows the advantage of CATA in which movement behaviors with different times are distinguished.

Next, the effects of silent durations are investigated. First, we study the scenario where the silent durations are evenly distributed in trajectories. Figure 13a shows the results when the SI is set to 10, 20, and 30 s. In all cases, CATS and LCSS outperform the other approaches. Except for wDF and DTW, the average error rates of the other approaches remain almost constant when SI increases. Because the mapping schemes of DTW and wDF require locating the nearest mapping points, silent durations significantly affects them. Now, we investigate the impact of P_{clue} on the randomness of silent durations. Figure 13b shows that, when P_{clue} is 70%, the average error rates of DTW and wDF increase slightly and those of CATS, LCSS and EDR decrease slightly in comparison with the setting of $P_{\text{clue}} = 100\%$. However, when P_{clue} decreases to 50%, the average error rates of all similarity measurements significantly decrease. The reason could be that there are many nearby locations in all types of trajectories, especially those places around the starting points, making all similarity measures predict inaccurately. When P_{clue} decreases, the number of these nearby locations may be reduced due to silent durations. This phenomenon benefits all similarity measures in distinguishing different types of trajectories.

7.3 Performance comparison of clustering algorithms

Here we compare our proposed clustering algorithm CATC with the traditional clustering algorithms (i.e., DBSCAN and Hierarchical clustering) and the existing trajectory clustering algorithms (i.e., PISA [25] and TraClus [19]).

To evaluate the quality of clustering, two common metrics, *entropy* and *purity*, are used. Since our real dataset is labeled, entropy is a function to estimate the distribution of class labels in clusters, and purity is a function of the relative size of the largest class in the resulting clusters. Ideally, a desirable cluster should contain trajectories with the same class label. However, in practice, a derived cluster may contain trajectories with several class labels. Entropy and purity are used to reflect how far the derived cluster to the desirable cluster is. If a derived cluster contains trajectories with fewer class labels, the entropy value of this cluster will be small. On the other hand, to label the class of a cluster, the major class of trajectories is used as the class of this cluster. Thus, a desirable case is that most trajectories in the same cluster have the same class. Purity is used to measure the proportion of the largest class trajectories in a cluster. Formally, given a particular cluster, S_r, of size n_r, the entropy of this cluster is defined as $E(S_r) = -\frac{1}{\log(q)} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r}$, where q is the number of classes in the dataset, and n_r^i is the number of trajectories of the ith class that are assigned to the rth cluster. The entropy of the overall clustering result is then defined as $\sum_{r=1}^{k} \frac{n_r}{n} E(S_r)$. In general, the smaller the entropy value, the more favorable the clustering result is. Similarly, the purity of a cluster is defined as $P(S_r) = \frac{1}{n_r} \max_i(n_r^i)$. Thus, the overall purity of the clustering result is formulated as $\sum_{r=1}^{k} \frac{n_r}{n} P(S_r)$. The larger the value of purity, the more favorable the clustering result is.

There are two kinds of approaches that can achieve the goal of clustering trajectories: the first one is to apply CATS with traditional clustering algorithms (i.e., DBSCAN and hierarchical clustering)[6]; the second one is to use existing trajectory clustering algorithms, such as PISA [25] and

[6] Note that the distance function used in DBSCAN and hierarchical clustering is required to be symmetric. We adopt the average similarity of measured trajectories to address the issue of asymmetry in CATS.

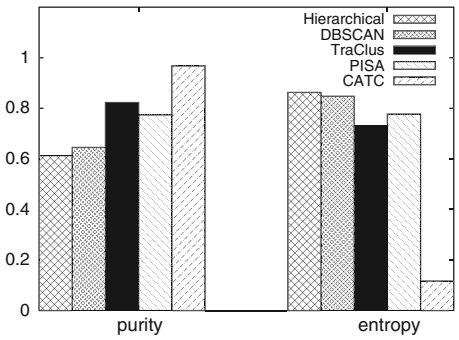

Fig. 14 Performance comparison of clustering algorithms via the real dataset

TraClus [19]. PISA uses its proposed distance function and then applies OPTICS (a self-tuning DBSCAN) to cluster trajectories. TraClus aims to discover clusters of "sub-trajectories", which does not align to our goal. Thus, we slightly modify the TraClus algorithm as follows. Suppose that the set of all sub-trajectory clusters is $\{sc_1, sc_2, \ldots, sc_n\}$, which can be treated as features of trajectories. Thus, each trajectory T_i could be represented as a vector $\overrightarrow{v_i} =< x_1, \ldots, x_n >$, where $x_j = 1$ if T_i passes sc_j. Accordingly, *cosine similarity* can be employed to measure the similarity of two vectors. Finally, DBSCAN is used to cluster trajectories. The following experimental results are the best results by fine tuning the parameters of the corresponding algorithms.

Figure 14 shows the entropy and purity values of all clustering algorithms. It can be seen that the approaches that apply CATS with traditional clustering algorithms have the highest entropy and the lowest purity values. We observe that DBSCAN and hierarchical clustering tend to form large clusters of both Type 1 and Type 2 trajectories (which have many close neighboring regions) and thus not do perform well.

7.4 Impact of silent durations on trajectory pattern mining

In this experiment, we study the impact of silent durations on trajectory pattern mining. Here we compare our CACT framework with two existing trajectory pattern mining techniques: Spatio-temporal Frequent Pattern mining (SFP) [2] and Trajectory Pattern Mining (TPM) [11].[7] Note that SFP, similar to CACT, uses representative line segments as hot regions to form trajectory patterns, and TPM is a state-of-the-art algorithm for trajectory pattern mining.

We argue that the inferior performance of SFP and TPM compared to our CACT framework is due to (i) imprecise hot regions generated by SFP and TPM, and (ii) impact of silent durations. Obviously, CACT addresses (i) pretty well

[7] The clue-aware trajectory aggregation (CATA) algorithm in our framework is based on the clustering generated by our clue-aware trajectory clustering (CATC) algorithm. Thus, it is not evaluated separately.

Fig. 15 Visualized results of SFP (**a**), TPM (**b**) and CACT (**c**)

because our framework, based on our observation and analysis, has chosen to cluster trajectories before mining trajectory patterns. To further study the impact of silent durations on trajectory pattern mining, we eliminate the factor (i) from the dataset by using trajectories of the same types in the experiments. Now, we present the visualized results of the competitors. In this experiment, all trajectories in the dataset are used. Figure 15 shows the results of SFP, TPM, and CACT. Figure 15a shows that SFP cannot derive the hot regions that fully describe the frequent routes. Since our dataset is generated by a user who drives on the road, and the derived hot regions are not aligned on the road, that is not a desirable result. Similarly, Fig. 15b shows that the sizes of the derived hot regions are too large such that these regions cross many regions without any roads. On the other hand, as shown in Fig. 15c, our mining result could infer the route of a user very well.

Because our trajectory datasets contain silent durations, linear interpolation and cubic spline interpolation are applied to estimate the missing points for SFP and TPM in order to better compare the mining results generated by SFP, TPM, and CACT. Linear and cubic splines pass through these points with piecewise linear and cubic polynomials, respectively. Low-order polynomials are usually used for interpolation because they can reduce not only the computational costs but also the numerical instabilities arising with high-degree curves. As reported in [32], cubic polyno-

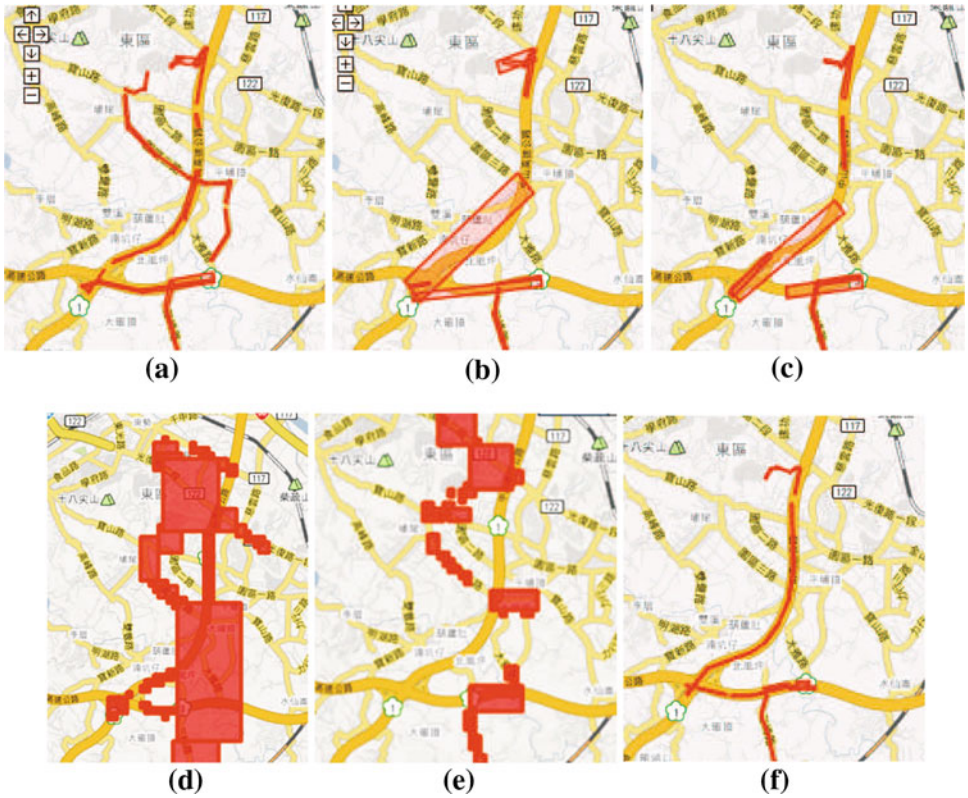

Fig. 16 Comparison of TPM, SPF, and CACT. **a** Ground truth. **b** SFP-L. **c** SFP-C. **d** TPM-L. **e** TPM-C. **f** CACT

mials are commonly used because no low-degree polynomial allows a curve to pass through two specified endpoints, guaranteeing continuous first and second derivatives across all polynomial segments. Thus, these piecewise cubic polynomials can be connected smoothly. For ease of presentation, we use SFP-L (SFP-C, respectively) to represents that we apply the algorithm SFP to linear-interpolated (cubic-interpolated, respectively) trajectories. Similarly, we denote TPM-L (TPM-C, respectively) as the algorithm TPM with linear-interpolated (cubic-interpolated, respectively) trajectories. Moreover, since SFP and TPM derive trajectory patterns without clustering the trajectories, the Type 1 trajectories are selected for these two approaches to further investigate whether the derived trajectory patterns can fit the ground truth more precisely.

Figure 16b–f show the trajectory patterns derived by all approaches. Figure 16b shows trajectory patterns derived by SFP-L. Because silent durations fragment trajectories, the trajectory patterns discovered by SFP-L have large hot regions. Even though the cubic spline interpolation attempts to form trajectories more smoothly, trajectory patterns mined by SFP-C are of higher quality than those of SFP-L. As shown in Fig. 16b, c, some regions are not found, as compared to the ground truth in Fig. 16a. Figure 16d, e show trajectory patterns discovered by TPM-L and TPM-C. Algorithm TPM uses a set of neighboring grids as hot regions. To "smoothen" the hot regions, linear regression is also used to further connect two nearby hot regions. Consequently, Fig. 16d shows that TPM-L generates many large hot regions when trajectories are linearly interpolated. Some of the hot regions even cover areas that a user will never pass by. On the other hand, when trajectories are smoothed by cubic spline interpolation, Fig. 16e shows that there are more hot regions located on the roads. Compared with the ground truth, this trajectory pattern does not describe the movement behavior (i.e., type 1) effectively. Figure 16f shows the trajectory patterns mined by CACT. Compared with the ground truth in Fig. 16a, CACT derives trajectory patterns that are almost identical to the original trajectory path. Obviously, some less frequent hot regions cannot be discovered when trajectories have certain silent durations. Some experiments are also conducted based on each type of trajectories. Similar to this experiment, the hot regions derived by the competitors are not well-aligned on the roads even though some interpolations are used for completing silent durations of trajectories. Due to the space limitation, we do not show all the similar results of trajectories with different types.

To evaluate the mining results of trajectory patterns, two performance metrics, *precision* and *recall*, are used. Since both SFP and CACT are able to derive a sequence of hot regions, smaller regions are more precise in describing movement behaviors. To quantify the precision of the mining results, a number of road segments covered by hot regions are used. Note that the trajectories are obtained while a

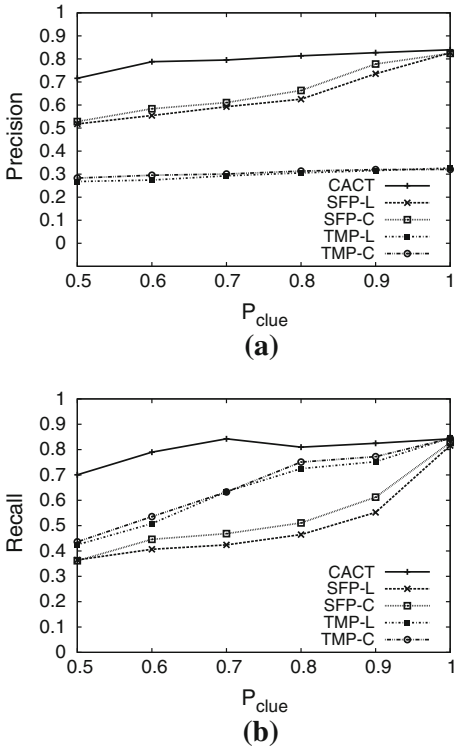

Fig. 17 Precision and recall of SFP-L, SFP-C, TMP-L, TMP-C and CACT with P_{clue} varied

user moves along particular road segments. Thus, trajectories can be represented as sequences of road segments. Using frequent itemset mining, a set of frequent road segments, denoted as F, can be obtained. Consequently, the *precision* is defined as follows. Let C be the road segments covered by the derived regions and F be the frequent road segments derived by frequent itemset mining. The *precision* is formulated as $L(C \cap F)/(L(C \cap F) + L(C \cap \bar{F}))$, where $L(\cdot)$ represents the total length of the roads. A higher precision value means that the derived region tends to cover more frequent road segments. On the other hand, the *recall* measures the number of frequent road segments within the derived hot regions. As such, the *recall* is formulated as $L(C \cap F)/(L(C \cap F) + L(\bar{C} \cap F))$. A higher recall value indicates that more frequent road segments are covered by the derived regions.

Figure 17a shows that CACT leads to the highest precision among all approaches. The precision of CACT is higher than 70% in all cases. Because the hot regions of SFP are rectangular, SFP achieves a higher precision than TMP. Figure 17b shows the recall values of all approaches. CACT outperforms all the other approaches as well. Since the hot regions of TMP are much larger than those of SFP, they cover a lot of road segments. Thus, the recall of TMP is higher than that of SFP. In summary, CACT handles silent durations very well when $P_{\text{clue}} > 50\%$. For the other approaches, P_{clue} is the most critical factor for the precision, whereas using interpolation on trajectories does not increase the precision considerably.

Table 3 Execution time (in seconds)

	CACT	SFP-L	SFP-C	TPM-L	TPM-C
CarWeb	7.24	2.12	2.23	1.42	1.57
Synthesis: short	27.4	5.31	5.51	4.76	4.82
Synthesis: long	32.3	5.84	6.23	5.22	5.23

Finally, we compare the efficiency of CACT with SFP and TPM. Table 3 shows the execution time of all approaches. Three datasets are used in this experiment, including CarWeb dataset, short synthetic dataset (the average length of trajectories is 1,000) with 100 trajectories, and long synthetic dataset (the average length of trajectories is 4,000) with 100 trajectories. The execution time of CACT takes longer than the other approaches because CACT first clusters trajectories and then generates trajectory patterns, whereas the other approaches do not cluster trajectories first. However, this tradeoff makes a significant improvement in the precision and the recall of the trajectory patterns and thus is worthwhile.

7.5 Performance study of CACT framework

We evaluate the execution time by varying the number of trajectories. The unit of execution time is 1 s in the following experiments. Two synthetic trajectory datasets are used. One consists of short trajectories of 2,000 points on average, while the other one consists of long trajectories of 6,000 points on average. The number of trajectories in these two synthetic datasets is ranged from 100 to 5,000 to evaluate the execution of CATC. The execution time includes three parts: (1) computing similarities between trajectories (CATS), (2) clustering trajectories (CATC), and (3) generating trajectory patterns (CATA). Figure 18 plots the execution time of various components in CACT in log-scale. Overall, the execution time increases as the number of trajectories increase. However, the proportion of the clustering phase and the aggregation phase are not the same under different kinds of trajectories. In the short trajectories, as shown in Fig. 18a, the clustering phase takes more execution time than the aggregation phase does. On the contrary, Fig. 18b shows that CACT spends most of its execution time deriving the hot regions when long trajectories are considered. Longer trajectories needs more time to aggregate the information and execute the Douglas-Peucker line simplifier.

8 Conclusions

In this paper, we propose the CACT framework to discover trajectory patterns and routes. In addition to spatial and temporal bias, we observe that trajectories usually contain

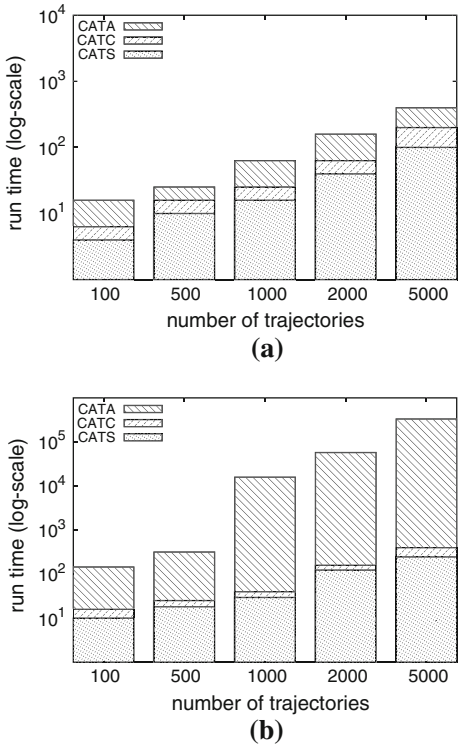

Fig. 18 Execution time under **a** short and **b** long trajectories

silent durations, during which detailed movement information is missing. Furthermore, since users may have multiple movement behaviors, trajectories should be clustered before hot regions are identified for trajectory pattern mining. Existing trajectory clustering techniques do not deal with trajectories with silent durations. Notice that clues about the same movement behavior are usually left in the trajectories sampled from this movement behavior. We argue that clues of a movement behavior are usually reflected by spatially and temporally co-located data points in trajectories. Thus, we formulate a clue-aware trajectory similarity and a clue-aware clustering algorithm to cluster similar trajectories into groups. For each group, CACT aggregates trajectories in the group to identify hot regions and to discover trajectory patterns. We have evaluated CACT using both real and synthetic datasets in experiments. Experimental results show that CACT is able to effectively discover trajectory patterns even if trajectories only capture fragments of movement behaviors.

For future works, the computation of CACT could be further improved. Specifically, we intend to derive the upper bound of CATS by approximating trajectories into a sequence of bounding boxes and propose some pruning rules to avoid the expensive pair-wise computation in CATS. As for the aggregation of trajectories, points in a trajectory are added into the base trajectory one by one and then clustered to form the final trajectory route. The above computations could be reduced by indexing the trajectories. Even though prior works have proposed index structures for moving objects [13,27], we aim to devise an index structure for trajectories with silent durations in order to facilitate efficient mining of trajectory patterns and routes.

References

1. EveryTrail—GPS Travel Community. http://www.everytrail.com/
2. Cao, H., Mamoulis, N., Cheung, D.W.: Mining frequent spatio-temporal sequential patterns. In: Proceedings of ICDM (2005)
3. Chen, L., Ng, R.T.: On the marriage of Lp-norms and edit distance. In: Proceedings of VLDB (2004)
4. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: Proceedings of SIGMOD (2005)
5. Chen, Y., Nascimento, M.A., Ooi, B.C., Tung, A.K.H.: SpADe: on shape-based pattern detection in streaming time series. In: Proceedings of ICDE, pp. 786–795 (2007)
6. Ding, H., Trajcevski, G., Scheuermann, P.: Efficient similarity join of large sets of moving object trajectories. In: Proceedings of TIME, pp. 79–87 (2008)
7. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. VLDB **1**(2), 1542–1552 (2008)
8. Dodge, S., Weibel, R., Forootan, E.: Revealing the physics of movement: comparing the similarity of movement characteristics of different types of moving objects. Comput. Environ. Urban Syst. **33**(6), 419–434 (2009)
9. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: Proceedings of KDD, pp. 63–72 (1999)
10. Giannotti, F., Nanni, M., Pedreschi, D.: Efficient mining of temporally annotated sequences. In: Proceedings of SDM (2006)
11. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: Proceedings of KDD (2007)
12. Gramm, J., Guo, J., Huffner, F., Niedermeier, R.: Data reduction, exact, and heuristic algorithms for clique cover. In: Proceedings of SIAM Workshop on Algorithm Engineering and Experiments (2006)
13. Hadjieleftheriou, M., Kollios, G., Tsotras, V.J., Gunopulos, D.: Efficient indexing of spatiotemporal objects. In: Proceedings of EDBT, pp. 251–268 (2002)
14. Jeung, H., Liu, Q., Shen, H.T., Zhou, X.: A hybrid prediction model for moving objects. In: Proceedings of ICDE (2008)
15. Jeung, H., Shen, H.T., Zhou, X.: Mining trajectory patterns using Hidden Markov models. In: Proceedings of DaWaK, pp. 470–480 (2007)
16. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S., Shen, H.T.: Discovery of convoys in trajectory databases. Proc. VLDB **1**(1), 1068–1080 (2008)
17. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Proceedings of SSTD, pp. 364–381 (2005)
18. Keogh, E.J.: Exact indexing of dynamic time warping. In: Proceedings of VLDB (2002)
19. Lee, J.-G., Han, J., Whang, K.-Y.: Trajectory clustering: a partition-and-group framework. In: Proceedings of SIGMOD (2007)
20. Li, Z., Ding, B., Han, J., Kays, R.: Swarm: mining relaxed temporal moving object clusters. Proc. VLDB **3**(1), 723–734 (2010)
21. Li, Z., Lee, J.-G., Li, X., Han, J.: Incremental clustering for trajectories. In: Proceedings of DASFAA, pp. 32–46 (2010)
22. Lo, C.-H., Peng, W.-C., Chen, C.-W., Lin, T.-Y., Lin, C.-S.: CarWeb: a traffic data collection platform. In: Proceedings of MDM (2008)

23. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, indexing, and querying historical spatio-temporal data. In: Proceedings of KDD (2004)
24. Meratnia, N., de By, R.A.: Spatiotemporal compression techniques for moving point objects. In: Proceedings of EDBT, pp. 765–782 (2004)
25. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. J. Intell. Inf. Syst. **27**(3), 267–289 (2006)
26. Pei, J., Hua, M., Tao, Y., Lin, X.: Query answering techniques on uncertain and probabilistic data. ACM SIGMOD Tutorial (2008). doi:10.1145/1376616.1376774
27. Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the positions of continuously moving objects. In: Proceedings of SIGMOD, pp. 331–342 (2000)
28. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA (2005)
29. Trajcevski, G., Ding, H., Scheuermann, P., Tamassia, R., Vaccaro, D.: Dynamics-aware similarity of moving objects trajectories. In Proceedings of GIS (2007)
30. Verhein, F., Chawla, S.: Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In: Proceedings of DASFAA (2006)
31. Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., Keogh, E.J.: Indexing multidimensional time-series. VLDB J. **15**(1), 1–20 (2006)
32. Wolberg, G., Alfy, I.: Monotonic cubic spline interpolation. In: Proceedings of the International Conference on Computer Graphics, pp. 188–195, Washington, DC, USA (1999)
33. Yi, B.-K., Jagadish, H.V., Faloutsos, C.: Efficient retrieval of similar time sequences under time warping. In: Proceedings of ICDE (1998)