Journal of Intelligent Information Systems (2021) 56:73–95
https://doi.org/10.1007/s10844-020-00607-8

# Trajectory clustering method based on spatial-temporal properties for mobile social networks

Ji Tang[1,2] · Linfeng Liu[1,2] · Jiagao Wu[1,2] · Jian Zhou[1] · Yang Xiang[1,2]

Received: 341 October 2019 / Revised: 25 April 2020 / Accepted: 27 April 2020 /
Published online: 14 January 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

**Abstract**
As an important issue in the trajectory mining task, the trajectory clustering technique has attracted lots of the attention in the field of data mining. Trajectory clustering technique identifies the similar trajectories (or trajectory segments) and classifies them into the several clusters which can reveal the potential movement behaviors of nodes. At present, most of the existing trajectory clustering methods focus on some spatial properties of trajectories (such as geographic locations, movement directions), while the spatial-temporal properties (especially the combination of spatial distances and semantic distances) are ignored, and thus some vital information regarding the movement behaviors of nodes is probably lost in the trajectory clustering results. In this paper, we propose a Joint Spatial-Temporal Trajectory Clustering Method (JSTTCM), where some spatial-temporal properties of the trajectories are exploited to cluster the trajectory segments. Finally, the number of clusters and the silhouette coefficient are observed through simulations, and the results show that JSTTCM can cluster the trajectory segments appropriately.

**Keywords** Trajectory clustering · Spatial-temporal properties · Spatial distances · Semantic distances

## 1 Introduction

Currently, with the rapid development of location-aware technology, most of mobile communication devices have been equipped with GPS positioning modules. In mobile social networks (MSNs), nodes (the nodes denotes the human beings carrying communication devices in the mobile social networks) always carry the communication devices when they

✉ Linfeng Liu
liulf@njupt.edu.cn

1 School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, China

2 Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, China

travel or walk, and the geographic locations of nodes are recorded at intervals. Then, these sequential location records can be transformed into some trajectories. Note that a great deal of information regarding the movement behaviors of nodes is hidden in these trajectories, and the information can be extracted by analyzing these trajectories and can be widely used in many applications, such as the location recommendations (Bao et al. 2015), destination predictions (Besse et al. 2018) and personal navigations (Li et al. 2015). Some valuable results can be mined and exploited through trajectory data mining techniques, e.g., the movement behaviors of nodes are obtained and are integrated into some location-based services where some personalized recommendations should be provided for nodes.

The mass of trajectory data contains the movement patterns and activity rules of group objects, such as the movement and activity characteristics of the crowds, the driving routes and the parking positions of vehicles, etc. These mobile patterns and activities reflect the mobile behaviors and living habits of the group objects. In addition, we can obtain some valuable information through a trajectory clustering process, and apply the valuable information to various applications such as traffic systems and location service applications. For example, (1) Path recommendation. The map service company analyzes the users' mobile routes by collecting the trajectory data, we can mine the ubiquitous movement patterns from the large-scale trajectory data, and then according to the real-time background environment and users' behaviors, reasonable paths are recommended for the users. (2) Personalized service recommendation. The trajectory data can record the user's locations in mobile social networks, and the user's movement trajectories are exploited for the understanding of trajectory behaviors, the characterization of users' characteristics, the mining of users' behavior patterns and so on. According to the implication of users' activities, the search engines will recommend satisfactory business services or personal services.

Trajectory data mining typically consists of the parts of trajectory classification, trajectory clustering, outlier detection and trajectory pattern mining. The trajectory classification belongs to the field of supervised learning, which relies on the labeled trajectory data. Actually, most of the trajectory data is unlabeled, and the cost of labeling the trajectory data manually is extremely large. In contrast, the trajectory clustering is a kind of unsupervised learning technique, and it does not require to label the training data, which seems more available in MSN applications than the trajectory classification.

By observing the movement behaviors of nodes in some real MSNs, there are two findings: (*a*) the nodes may exhibit a frequent movement trend when traveling on some road segments, as shown in Fig. 1a, the similar movement trends exist in the four trajectories of the same node; (*b*) there are common movement trends among different nodes, as illustrated in Fig. 1b, where the trajectories of node A, B, C and D have a common movement trend. The movement similarities between nodes are usually related to the number of similar road segments, i.e., more similar road segments give rise to a larger movement similarity (a closer relationship between the movements of two nodes). In order to exploit the common movement trends from a large amount of trajectories, the trajectory clustering technique is expected to allocate the similar trajectory segments into the same clusters.

The existing trajectory clustering techniques are divided into two categories: global trajectory clustering and sub-trajectory clustering. The global trajectory clustering methods classify the complete trajectories according to the trajectory similarities, whereas the complete trajectories are always of little similarities. The similarities between sub-trajectories are much easier to be explored than those between different complete trajectories, the original trajectories are first partitioned into several sub-trajectories (or trajectory segments), and then the similarities between these sub-trajectories (or trajectory segments) are calculated for further clustering process. In addition, most existing clustering methods consider

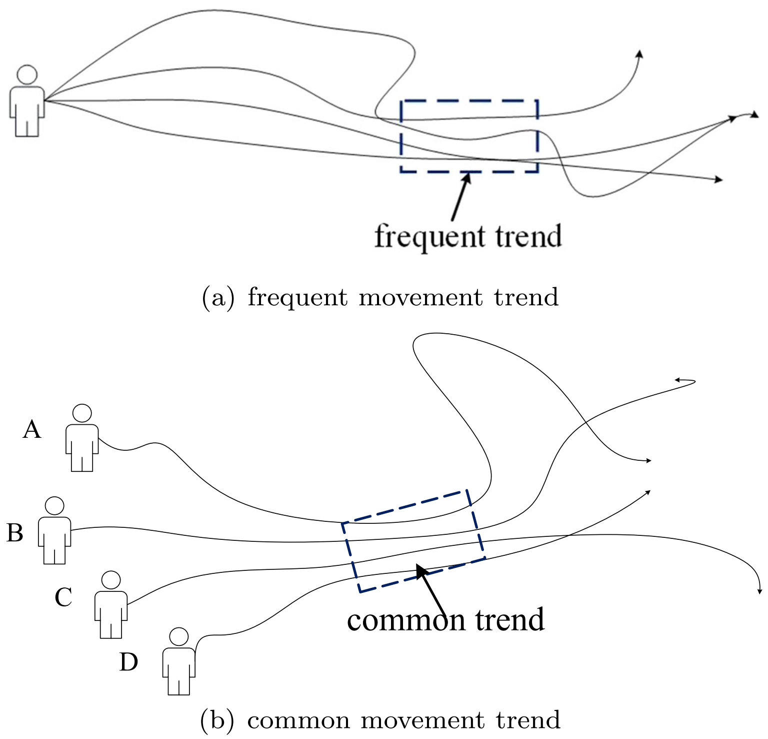


**Fig. 1** Two types of movement trends

the spatial-positions or temporal-information attached to the trajectories respectively in the similarity measurements of sub-trajectories, and the spatial properties and temporal properties of trajectories are not taken into account jointly, which makes some trajectory segments allocated into wrong clusters inevitably.

In this paper, we focus on the human trajectories, and the human beings carrying communication devices play the role of nodes in mobile social networks, so that the simulation dataset is also human related. A spatial-temporal distance function combining the spatial distances and semantic distances is designed to measure the similarities between trajectory segments, and then a density-based trajectory clustering method is proposed. To obtain the preferable clusters of trajectory segments, this work is consisted of two stages, as depicted in Fig. 2:

- trajectory partition, which partitions the original trajectories into trajectory segments;
- trajectory clustering, which clusters the trajectory segments through investigating the similarities between trajectory segments.

The rest of the paper is organized as follows: Section 2 gives some related works. Section 3 introduces the joint spatial-temporal trajectory clustering method. Section 4 reports the simulation results of trajectory clustering. Finally, Section 5 concludes the paper.

This work is a significant extension of our early work (Tang et al. 2019b). Specifically, many related works are investigated, and our research motivation is introduced. The proposed JSTTCM is improved and explained with more details, e.g., a kernel function is give

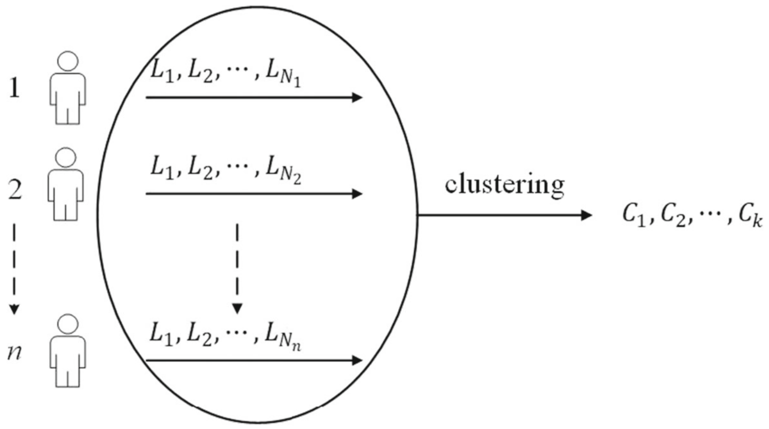

1
2
$n$
$L_1, L_2, \cdots, L_{N_1}$
$L_1, L_2, \cdots, L_{N_2}$
$L_1, L_2, \cdots, L_{N_n}$
clustering
$C_1, C_2, \cdots, C_k$

**Fig. 2** Trajectory partition and trajectory clustering

to measure the trajectory clustering results, and some example results regarding the combination and the normalization are provided to illustrate these mechanisms. Besides, much more simulation results are provided to further clarify the merits of JSTTCM.

## 2 Related work

The issue of trajectory clustering has attracted lots of attention. For example, Gaffney et al. propose a hybrid model-based trajectory clustering algorithm (Gaffney and Smyth 1999), which models each trajectory as a measurement sequence given by a time function, and then the objects generated by the core trajectory and Gaussian noise are aggregated by the EM algorithm. Alon propose a clustering method on the basis of a model in which each cluster is expressed by a Markov model (Alon et al. 2003), and the parameter estimation is also completed by the EM algorithm. However, all these methods treat each trajectory as a clustering object, and thus it is difficult to find the local similarities of the trajectories.

The trajectory clustering algorithms typically denote a trajectory by an eigenvector. Each dimension in the eigenvector is related to an attribute of the trajectory, and hence the number of the dimensions is equal to that of attributes. Besides, the distance between eigenvectors is applied to measure the similarities between trajectories, and a shorter distance is concerned with a larger similarity. Notice that the dimensions of different eigenvectors may be different, and this fact makes some existing trajectory clustering algorithms become unavailable, this is because the trajectories can embody many attributes, such as the lengths, shapes and sampling rates of trajectories. Therefore, the trajectory segments should be comprehensively compared according to their differences. Fortunately, there are some distance indices to measure the similarities between trajectories: the Longest Common Sub-Sequence (LCSS) distance aims to find the longest common sub-trajectory (Michail et al. 2006); Hausdorff distance is used to measure how far two trajectories are from each other (Liu et al. 2014); Dynamic Time Warping (DTW) targets to seek out an optimal alignment between two trajectory sequences (Chen et al. 2005); Chen and Ng treat each trajectory as a time series and then design an edit distance with actual penalty to measure the trajectory similarities (Chen and Ng 2004). The aforementioned distance indices are more suitable to calculate the distances of different trajectories rather than those of different trajectory segments.

Considering the difficulty of dealing with the complete trajectories, Lee et al. put forward a density-based trajectory clustering method TraClus (Lee et al. 2007), which first divides the trajectories into continuous trajectory segments by Minimum Description Length (MDL), and then the similar segments are clustered using this density-based clustering algorithm. A hierarchical trajectory clustering algorithm utilizing HDBSCAN (Zhang et al. 2018) based on TraClus is presented in Campello et al. (2013), where some semantic information (such as direction, speed and time) is considered. In addition, a trajectory clustering algorithm CTHD (clustering of trajectory based on hausdorff distance) is proposed (Chen et al. 2011), where a sequence of flow vectors are described and partitioned into a set of sub-trajectory. Then, the similarity between trajectories is measured by their respective Hausdorff distances (including the positions and directions). Finally, the trajectories are clustered by the DBSCAN clustering algorithm.

Besides, some existing literatures analyze the semantic trajectory data to find interesting places, such as CB-SMoT (Palma et al. 2008) and DB-SMoT (Rocha et al. 2010). In CB-SMoT, a trajectory clustering method based on velocity variation is provided to estimate the trajectory sample points and then form the clusters. In DB-SMoT, the changes in movement directions are taken as the key index for the clusters, and the proposed method is verified by the trajectory data of marine fishing vessels. An adaptation of a density-based clustering algorithm to trajectory data based on a simple notion of distance between trajectories is proposed (Nanni and Pedreschi 2006), and a new approach called temporal focusing to the trajectory clustering problem is sketched, which having the aim of exploiting the intrinsic semantics of the temporal dimension to improve the quality of trajectory clustering. A visually-driven analysis of movement data by progressive clustering is proposed (Rinzivillo et al. 2008), which focus in more detail on the use of cluster analysis with different similarity measures. In particular, the authors assume that each distance function aggregates objects according to its own semantics, the user can choose a sequence of functions to be used progressively. In Andrienko et al. (2016) the spatial-temporal aggregation is applied to traffic data consisting of vehicle trajectories, and a spatially abstracted transportation network consisting of areas in geographic space and the possible paths is generated, which is used to traffic analysis, forecasting and simulation leveraging spatial abstraction. The literature (Trasarti et al. 2011) introduces the concept of the mobility profile of a user as the set of his routine trips, and define a general method based on trajectory clustering to extract such profiles; then an instantiation of the method on the GPS data of vehicles with a route similarity function is shown; finally, the downgrading of the spatial-temporal richness of the data is studied.

In 2015, an unsupervised trajectory clustering method based on adaptive multi-kernel shrinkage is presented (Xu et al. 2015), where an adaptive multi-kernel-based estimation process is performed to estimate the “reduction” positions and the velocities of the track points. Further, a speed regularization optimization process that utilizes the estimated velocity is introduced to adjust the optimal shrunk points to ensure the smoothness and discriminant mode of the final shrunk trajectory. However, the proposed method does not take into account the time information attached to trajectories. In 2017, a trajectory clustering method via deep representation learning is proposed (Yao et al. 2017), where the trajectory clustering problems are re-examined by learning the low-dimensional trajectory representations. Especially, a sliding window is used to extract a set of movement behavior features that capture the spatial-temporal invariant features in the trajectories, and then each trajectory is converted into a sequence of features that can describe the object movements. Finally, the $K$-means clustering algorithm is utilized to cluster the learned trajectory representations.

In fact, many existing works have focused and addressed the trajectory clustering problem, and most of them ususally employ some spatial properties or temporal properties to cluster the trajectories. Yes, there are also some work taking into account of spatial properties or temporal properties simultaneously. We add some explanations about the motivation of this paper:

- Although some works have considered the spatial-temporal properties, there are some differences in terms of the problem objectives and application scenarios. The objective of our work is to explore the human trajectories, which indicates that there are some spatial-temporal properties hidden in human trajectories, and thus it is very necessary to analyze the mobile behaviors of human beings. However, the existing works mainly focus the vehicle trajectories, and the movement behaviors of vehicles are much different from those of human beings.
- We take the human beings carrying communication devices as nodes in the mobile social networks, i.e, the nodes have some social attributes. We observe that there are some similarities between these nodes due to the social attributes. Therefore, a spatial-temporal trajectory clustering method is appropriate to exploit the relations between nodes.
- This paper consists of two parts: trajectory partition and trajectory cluster. We consider some movement features of nodes in the process of trajectory partition, the results of which are served for the trajectory cluster. Therefore, we think the proposed method is quite different from other works.

In this paper, the spatial-temporal properties of the trajectories are investigated to cluster the trajectory segments, especially, a spatial-temporal distance function combining the spatial distances and semantic distances is defined to measure the similarities between trajectory segments, and then a Joint Spatial-Temporal Trajectory Clustering Method (JSTTCM) is proposed.

## 3 Joint spatial-temporal trajectory clustering method

JSTTCM is consisted of the stages of trajectory partition and trajectory clustering. In this section, we first introduce the trajectory partition method to partition the original trajectories into trajectory segments. Then, a new density-based trajectory clustering method is given to cluster the trajectory segments. To clarify the description, several definitions are provided as following:

**Definition 1** A GPS point: a GPS point $p_i$ is expressed as a triple $(lat_i, lng_i, t_i)$, which is composed of the latitude, longitude and timestamp of the $i$-th point, where $i = 1, 2, \cdots, n$, and $n$ is the number of GPS points in a trajectory.

**Definition 2** A trajectory: a trajectory is a sequence of GPS points arranged in a chronological order and is denoted by $TR = p_1, p_2, \cdots, p_n$, as shown in Fig. 3, the label of Axis Y indicates time stamp.

**Definition 3** A trajectory segment: a trajectory segment $L_i = \{p_s, p_e\}$ is denoted by the start point and the end point, where $p_s$ denotes the start point, and $p_e$ denotes the end point.

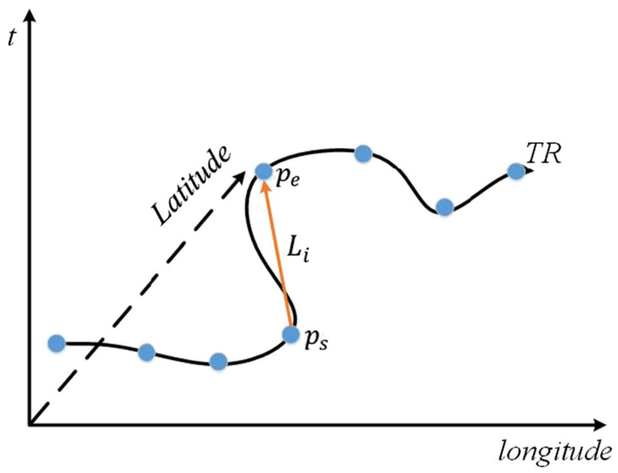


**Fig. 3** An example trajectory

## 3.1 Trajectory partition

In MSNs, we note three significant characteristics regarding the node movements: (*i*) the moving speed of nodes changes greatly due to the switches of different travel modes or some special events (e.g., sudden braking or abrupt deceleration in driving) (Zheng et al. 2010a); (*ii*) the nodes are prone to visit some locations intentionally or unintentionally, and the locations visited frequently or stayed for a long period can be marked as stop points; (*iii*) the moving directions of nodes are varied frequently when some travel modes (such as the walking mode) are adopted. To this end, we provide a Trajectory Partition mechanism based on Movement Features(TPMF) (Tang et al. 2019a).

In TPMF, the stop points and change points are extracted according to the moving speeds and the speed changes, respectively, and then the extracted stop points and change points are treated as feature points. Finally, a Douglas-Peucker algorithm based on Direction changes and Perpendicular Euclidean Distance (DPDPED) is applied to simplify the sub-trajectories between the adjacent feature points.

## 3.2 Trajectory clustering

In this section, we define a spatial-temporal distance function and raise a density-based clustering method to cluster the similar trajectory segments. Besides, a clustering validation is given to evaluate the clustering results.

### 3.2.1 Spatial-temporal distance function

As mentioned above, it is hard to identify the similar movement behaviors in different trajectories, and thus we exploit the similarities between trajectory segments. Especially, some spatial-temporal properties are taken into account to calculate the similarities between trajectory segments. As shown in Fig. 4, the spatial distances include the perpendicular Euclidean distance and the parallel distance, and the spatial distances can measure the spatial relationships between trajectory segments. The semantic distances include the direction

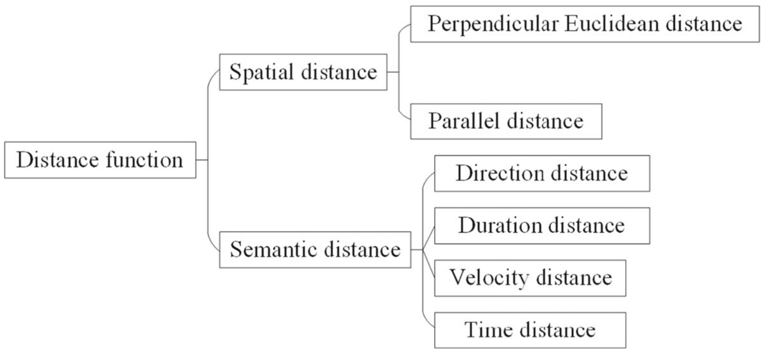


**Fig. 4** Spatial distances and Semantic distances

distance, duration distance, velocity distance and time distance, i.e., the semantic distances integrate the temporal dimension to explore the temporal relationships between trajectory segments.

The spatial-temporal distance function considers the spatial distances and semantic distances jointly:

(***a***) Note that the perpendicular Euclidean distance remains the same unless the two trajectory segments are parallel to each other. Therefore, we first compare the lengths of two trajectory segments, and then calculate the perpendicular Euclidean distance from the shorter trajectory segment (such as $L_i$ in Fig. 5) to the longer trajectory segment (such as $L_j$ in Fig. 5). The perpendicular Euclidean distance is given by:

$$d_{\perp} = \frac{l_{\perp 1}^2 + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}}, \tag{1}$$

where

$$\begin{cases} l_{\perp 1} = \sqrt{\|\overrightarrow{p_c p_a}\|^2 - (\frac{\overrightarrow{p_c p_a} \cdot \overrightarrow{p_c p_d}}{\|\overrightarrow{p_d p_c}\|})^2}, \\ l_{\perp 1} = \sqrt{\|\overrightarrow{p_d p_b}\|^2 - (\frac{\overrightarrow{p_d p_b} \cdot \overrightarrow{p_d p_c}}{\|\overrightarrow{p_d p_c}\|})^2}. \end{cases} \tag{2}$$

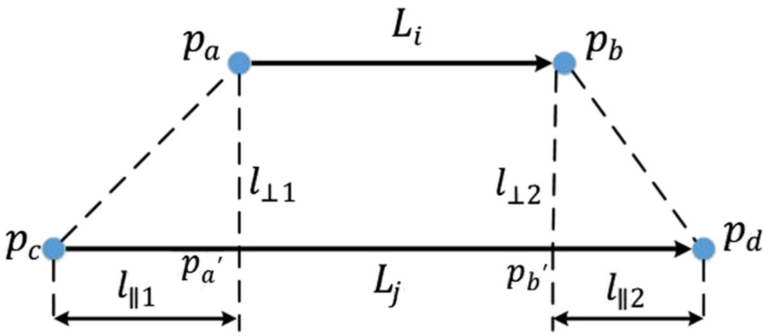


**Fig. 5** Perpendicular Euclidean distance and parallel distance

(***b***) In Fig. 5, the parallel distance consists of $l_{\|1}$ and $l_{\|2}$. We take the maximum value of $l_{\|1}$ and $l_{\|2}$ as the parallel distance, and the parallel distance is expressed as:

$$d_{\|} = max(l_{\|1}, l_{\|2}), \tag{3}$$

where $l_{\|1}$ and $l_{\|2}$ are calculated as:

$$\begin{cases} l_{\|1} = \frac{\overrightarrow{p_c p_a} \cdot \overrightarrow{p_c p_d}}{\|\overrightarrow{p_c p_d}\|}, \\ l_{\|2} = \frac{\overrightarrow{p_d p_b} \cdot \overrightarrow{p_d p_c}}{\|\overrightarrow{p_d p_c}\|}. \end{cases} \tag{4}$$

The semantic distances focus on the trajectory differences in temporal dimensions. As depicted in Fig. 6, the angle between two trajectory segments denotes the directional deviation.

(***c***) The direction distance can be expressed by an angle cosine and is written as:

$$d_\theta = \begin{cases} 1 - \cos\theta, & 0 \le \theta < \frac{\pi}{2} \\ 1, & \frac{\pi}{2} \le \theta \le \pi \end{cases} \tag{5}$$

where $\theta$ denotes the angle between two trajectory segments.

(***d***) The duration distance reflects the difference of time intervals between trajectory segments. The duration distance is computed as:

$$d_{\Delta t} = \begin{cases} 1 - \frac{\Delta t^{(L_i)}}{\Delta t^{(L_j)}}, & \Delta t^{(L_i)} \le \Delta t^{(L_j)}, \\ 1 - \frac{\Delta t^{(L_j)}}{\Delta t^{(L_i)}}, & otherwise, \end{cases} \tag{6}$$

where $\Delta t^{(L_i)} = t_e^{(L_i)} - t_s^{(L_i)}$, and $\Delta t^{(L_j)} = t_e^{(L_j)} - t_s^{(L_j)}$.

(***e***) The velocity distance measures the difference of traffic modes between nodes. The velocity distance is given by:

$$d_v = \begin{cases} 1 - \frac{v_i}{v_j}, & v_i \le v_j, \\ 1 - \frac{v_j}{v_i}, & otherwise, \end{cases} \tag{7}$$

where $v_i = \frac{\|L_i\|}{\Delta t^{(L_i)}}$, and $v_j = \frac{\|L_j\|}{\Delta t^{(L_j)}}$.

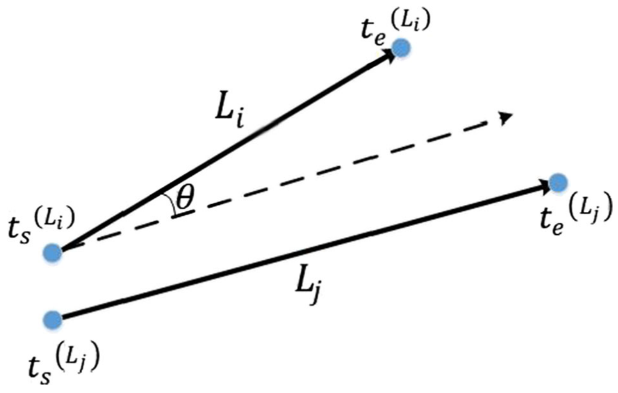


**Fig. 6** Illustration of direction distance

(*f*) The time distance is to measure how close are the time stamps of the trajectory segments, and it is defined as:

$$d_t = \left| t_s^{(L_i)} - t_s^{(L_j)} \right| + \left| t_e^{(L_i)} - t_e^{(L_j)} \right|. \tag{8}$$

In order to combine the different kinds of distances technically, a nonlinear function is introduced for normalization, and it is expressed as:

$$f(x) = \frac{1}{1 + e^{-ax+b}}, \tag{9}$$

where $a$ and $b$ are preset parameters. Some example results for the combination and normalization are given in Fig. 7.

### 3.2.2 Comparisons of different distance functions

Three types of distance functions (weighted summation function, kernel function and time focusing function) are applied to construct the spatial-temporal distance function, respectively. The weighted summation distance function is written as:

$$dist_w(L_i, L_j) = w_1 \cdot d_{\perp'} + w_2 \cdot d_{\|'} + w_3 \cdot d_\theta + w_4 \cdot d_v + w_5 \cdot d_{\Delta t} + w_6 \cdot d_{t'}, \tag{10}$$

where $d_{\perp'}$, $d_{\|'}$ and $d_{t'}$ denote the normalizations of perpendicular Euclidean distance, parallel distance and time distance, respectively.

The kernel distance function is similar to the Gaussian kernel, and it is written as:

$$dist_k(L_i, L_j) = 1 - e^{-\frac{\|\boldsymbol{F}\|^2}{2\sigma^2}}, \tag{11}$$

where $\boldsymbol{F} = (d_{\perp'}, d_{\|'}, d_\theta, d_v, d_{\Delta t}, d_{t'})$ denotes an eigenvector, and $\sigma$ denotes a reduction factor.

The time focusing distance function is written as:

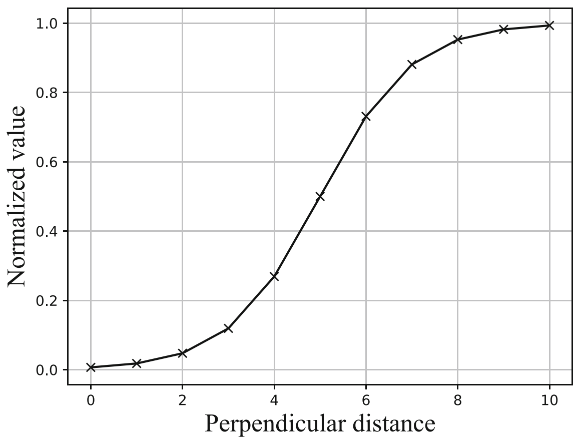
$$dist_T(L_i, L_j) = \frac{\sum_{t_0}^{t_0+T} d\left(L_1(t), L_2(t)\right)}{|T|}, \tag{12}$$

where $t \in (t_0, t_0 + |T|)$, $t_0$ indicates a start moment, $T$ is the temporal interval over which trajectories $L_1$ and $L_2$ exist. Such a definition requires a temporal domain common to all objects, in general, it is a hard requirement. Actually, it is hard to find a common temporal domain between two different trajectory segments in real-life dataset. However, it is unnecessary to require common temporal interval in our work. Although such a definition considers the temporal property, the speed and direction was not taken into account in time focusing distance function.

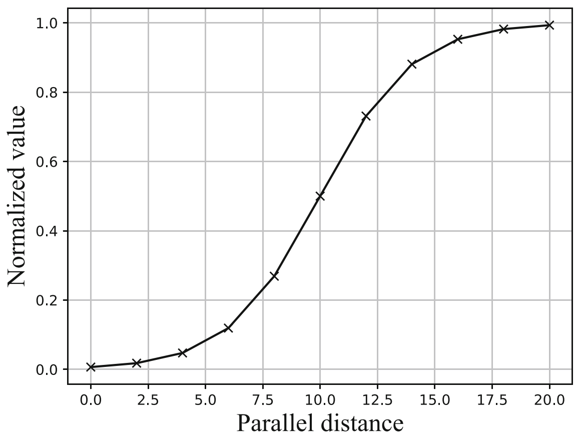
Firstly, we compare the different characteristics between the weighted summation distance function and kernel distance function. Suppose there are two trajectory segments $L_1 = \{(4.25, 2.53, 12:00:15), (5.65, 10.24, 12:30:45)\}$ and $L_2 = \{(4.26, 2.535, 11:50:25), (5.64, 10.25, 12:45:10)\}$. A trajectory segment consists of a start point and an end point, and each point consists of latitude, longitude and time stamp. For example, $L_1$ contains two points, and note that the latitude and longitude should be transformed to adapt the coordinate system. The start point of $L_1$ is (4.25, 2.53, 12:00:15), where 4.25 and 2.53 represent the transformed latitude and transformed longitude, respectively, and 12:00:15 denotes the timestamp. We have defined six different distance measures in pages 8 and 9, and provided a nonlinear function to normalize these distances.

Therefore, we get $\boldsymbol{F} = (0.995, 0.277, 0.000, 0.442, 0.442, 0.277)$, $d_{\perp'} = 0.995$ is the normalized perpendicular Euclidean distance between $L_1$ and $L_2$ with $a = 1$ and $b = 5$,

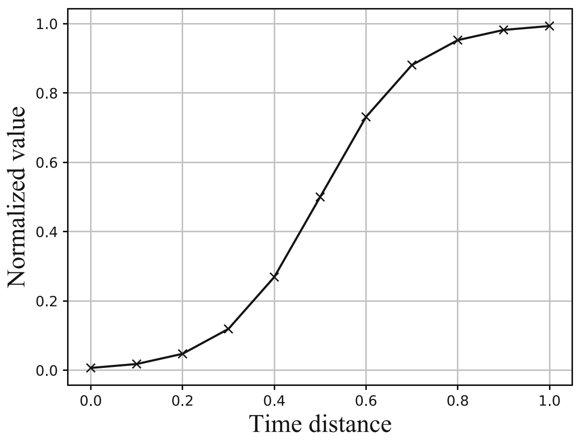
**Fig. 7** Normalization curves under different values of *a* and *b*



(a) $a = 1,\ b = 5$



(b) $a = 0.5,\ b = 5$



(c) $a = 10,\ b = 5$

which indicates the perpendicular Euclidean distance is very large. $d_{\|'} = 0.277$ is the normalized parallel distance between $L_1$ and $L_2$ with $a = 0.5$ and $b = 5$, which indicates the parallel distance is relatively small. Note that $d_\theta = 0.000$ represents the directions of $L_1$ and $L_2$ remain almost parallel. $d_v = 0.442$ represents the speed distance by (7), and $d_{\Delta t} = 0.442$ represents the duration distance by (6). $d_{t'} = 0.277$ is the normalized time distance by (9) with $a = 10$ and $b = 5$. Then, the distance between $L_1$ and $L_2$ calculated by the weighted summation function is equal to 0.406, and the distance calculated by the kernel function is equal to 0.536. From the results of $\boldsymbol{F}$, the perpendicular Euclidean distance between $L_1$ and $L_2$ is relatively large, which indicates that $L_1$ and $L_2$ are not similar. However, we cannot determine that whether $L_1$ and $L_2$ are similar or not according to the distances calculated by two distance functions. Note that there is an obvious difference between the results calculated by the two distance functions. The reason is that the kernel distance function is sensitive to the variation of each distance. In fact, the difference will be reduced when all the distances are not large. For example, we set the spatial positions of $L_1$ and $L_2$ be equal, which gives rise to 0.195 (the result obtained by the weighted summation distance function) and 0.209 (the result obtained by the kernel distance function), respectively, and it is obvious that the difference is reduced significantly.

We also compare the different characteristics between the weighted summation distance function and the time focusing distance function. Suppose there are two different trajectory segments $L_1 = \{(4.25, 2.53, 8:00:00), (5.65, 10.24, 8:10:00)\}$ and $L_2 = \{(4.26, 2.535, 8:00:00), (5.64, 10.25, 8:10:00)\}$, which have a common temporal domain. Therefore, we get $|T|$=600s, it's easy to get the average distance $dist_T(L_i, L_j) = \frac{11.180+14.142}{600} = 0.042$ by the time focusing distance function. In our work, both the time distance and the duration distance are equal 0. In addition, the distance $dist_w(L_i, L_j) = 0.212$ between $L_1$ and $L_2$ by the weighted summation function. From the results of two different distance functions, it indicates that the value of $dist_w(L_1, L_2)$ is five times more than the value of $dist_T(L_i, L_j)$, which is attributed to the fact that our distances have considered spatial-temporal properties. In fact, if the temporal distances are very small, yet the spatial distance are much large, which makes the similarity of trajectories extremely low.

### 3.2.3 Density-based clustering

We propose a density-based clustering algorithm, which is an improved version of DBSCAN (Ester et al. 1996). The outstanding advantage of DBSCAN is that it can filter out the noise and automatically identify the number of clusters. DBSCAN characterizes the closeness of the distribution of trajectory segments based on the neighborhood parameters $(\epsilon, MinPts)$. The basic idea is that each cluster continues to grow, until the trajectory segments in its neighborhood (the range of $\epsilon$) cannot become the core trajectory segments. Some definitions are first given to illustrate our proposed density-based clustering algorithm:

- $\epsilon$-**neighborhood**: there is $L_i \in \boldsymbol{D}$ (trajectory segment dataset), if its $\epsilon$-neighborhood contains the trajectory segments whose distances to $L_i$ are not larger than $\epsilon$, i.e, $N_\epsilon(L_i) = \left\{L_j \in \boldsymbol{D} | dist_w(L_i, L_j) \le \epsilon\right\}$.
- **Core trajectory segment**: if $L_i$'s $\epsilon$-neighborhood contains at least $MinPts$ other trajectory segments, i.e., $|\epsilon(L_i)| \ge MinPts$, and then $L_i$ is taken as a core trajectory segment.

- **Directly density-reachability**: if $L_j$ falls into the $L_i$'s $\epsilon$-neighborhood, and $L_i$ is a core trajectory segment, then $L_j$ is directly density-reachable to $L_i$.
- **Density-reachability**: if there is a set of trajectory segments $p_1, p_2, \cdots, p_n$, where $p_1 = L_i$, $p_n = L_j$ and $p_{i+1}$ is directly density-reachable to $p_i$ , then $L_j$ is considered to be density-reachable to $L_i$.
- **Density-connectivity**: if $L_j$ and $L_i$ are density-reachable to the same trajectory segment $L_k$, and then $L_j$ and $L_i$ are density-connected to each other.

An example is given to illustrate the definitions in Fig. 8, where $L_1$ is a core trajectory segment, and $L_2$ is directly density-reachable to $L_1$. $L_3$ is density-reachable to $L_1$, and $L_3$ is density-connected to $L_4$.

All core trajectory segments can be identified when the parameters $\epsilon$ and $MinPts$ are set. Then, the core trajectory segments can "carry" all trajectory segments of their $\epsilon$-neighborhoods into density regions according to the directly density-reachable relationships. Furthermore, the density-reachable relationships are utilized to connect the small density regions with the centers of core trajectory segments. Finally, each core trajectory segment and the density-connected neighbors will constitute a density region (cluster). The time complexity of the above steps reaches $\boldsymbol{O}(n^2)$ due to the fact that all of the neighborhoods of trajectory segments must be searched. In order to reduce the time complexity, we reduce the neighborhoods through exploiting some temporal attributes:

1. The trajectory segment dataset is sorted in a chronological order.
2. $t_s^{(L_i)}$ and $t_e^{(L_i)}$ are found from each trajectory segment $L_i$, and $t_s^{(L_i)} - \Delta t$, $t_e^{(L_i)} + \Delta t$ are calculated, respectively, where $\Delta t$ is a preset offset.
3. The binary search is used to find the index $l$ of $t_s^{(L_i)} - \Delta t$ and the index $r$ of $t_e^{(L_i)} + \Delta t$ in the sorted dataset.
4. The initial neighborhood range is obtained according to $l$ and $r$, and the distances between trajectory segments in the neighborhoods are recalculated.

The pseudo-code of our proposed density-based clustering algorithm is shown in Algorithm 1.

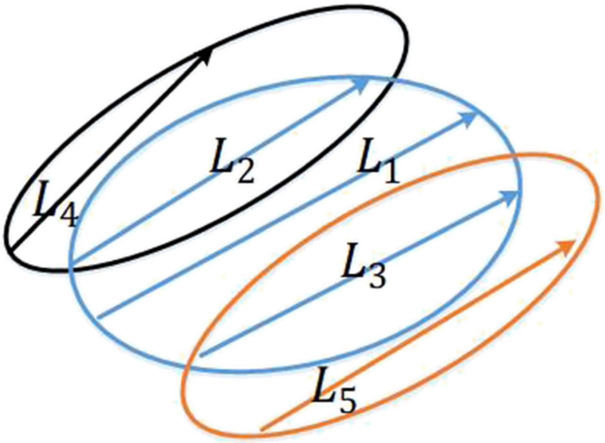


**Fig. 8** Relations of trajectory segments

**Algorithm 1** Density-based clustering pseudo-code.

```
Input: $\boldsymbol{D} = \{L_1, L_2, L_3, \cdots, L_n\}, \epsilon, MinPts, \Delta t$.
Output: $\boldsymbol{C} = \{C_1, C_2, C_3, \cdots, C_k\}$.
 1: Initialize the core trajectory segment set: $\boldsymbol{Core} = \phi$.
 2: Sort trajectory segment set in a chronological order $\rightarrow$ $\boldsymbol{SortedD}$.
 3: for $i = 1, 2, 3, \cdots, n$ do
 4:     $l, r = BinarySearch(\boldsymbol{SortedD}, t_s^{(L_i)}, t_e^{(L_i)})$.
 5:     Ensure $L_i$'s $\epsilon$-neighborhood $N_\epsilon \in (L_i)$.
 6:     if $|N_\epsilon(L_i)| \geq MinPts$ then
 7:         $\boldsymbol{Core} \leftarrow \boldsymbol{Core} \cup L_i$.
 8:     end if
 9: end for
10: Initialize the number of clusters $k = 0$.
11: Initialize the untraversed set $\boldsymbol{F} \leftarrow \boldsymbol{D}$.
12: while $\boldsymbol{Core} \neq \phi$ do
13:     Record the current untraversed set $\boldsymbol{F_{old}} \leftarrow \boldsymbol{F}$.
14:     Randomly select a core trajectory segment $L \in D$ and initialize the queue $\boldsymbol{Q} \leftarrow \langle L \rangle$.
15:     $\boldsymbol{F} \leftarrow \boldsymbol{F} \setminus \boldsymbol{D}$.
16:     while $\boldsymbol{Q} \neq \phi$ do
17:         Retrieve the first trajectory segment $q$ in queue $\boldsymbol{Q}$.
18:         $l, r = BinarySearch(\boldsymbol{SortedD}, t_s^{(q)}, t_e^{(q)})$.
19:         if $|N_\epsilon(q)| \geq MinPts$ then
20:             $\Delta \leftarrow N_\epsilon(q) \cap \boldsymbol{F}$.
21:             $\boldsymbol{Q} \leftarrow \boldsymbol{Q} \bigcup \Delta$.
22:             $\boldsymbol{F} \leftarrow \boldsymbol{F} \setminus \Delta$.
23:         end if
24:     end while
25:     $k = k + 1, C_k = \boldsymbol{F_{old}} \setminus \boldsymbol{F}$.
26:     $\boldsymbol{Core} \leftarrow \boldsymbol{Core} \setminus C_k$.
27: end while
```

### 3.3 Clustering validation

To evaluate the clustering results, such as the number of appropriate clusters and the cluster structures, some unsupervised metrics have been applied to validate the clustering results, and these unsupervised metrics can be typically classified into two categories: the metric *cohesion* measures how closely related are the trajectory segments falling into the same cluster; the metric *separation* measures how distinctly separated are the trajectory segments in a cluster from those in other clusters.

In this paper, we use the silhouette coefficient (Rousseeuw 1987) to evaluate the clustering results, because the silhouette coefficient is a combination metric of cohesion and separation. Given the obtained results $\boldsymbol{C} = \{C_1, C_2, \cdots, C_k\}$, $\mathcal{A}(C_i)$ represents the average distance of the trajectory segments in $C_i$, and $\mathcal{B}(C_i)$ represents the minimum average distance from $C_i$ to other clusters, where $i = 1, 2, \cdots, k$. $\mathcal{A}(C_i)$ and $\mathcal{B}(C_i)$ are given by:

$$\mathcal{A}(C_i) = \frac{1}{|C_i|} \cdot \sum_{L \in C_i} \frac{\sum_{L' \in C_i} dist(L, L')}{|C_i| - 1}, \tag{13}$$

$$\mathcal{B}(C_i) = \min_{1\leq j\leq k, j\neq i} \frac{1}{|C_i|} \cdot \sum_{L\in C_i} \frac{\sum_{L'\in C_j} dist(L, L')}{|C_j|}. \tag{14}$$

Besides, the silhouette coefficient of $C_i$ is defined as:

$$\mathcal{S}(C_i) = \frac{\mathcal{B}(C_i) - \mathcal{A}(C_i)}{max\,\{\mathcal{A}(C_i), \mathcal{B}(C_i)\}} \tag{15}$$

A smaller $\mathcal{A}(C_i)$ gives rise to a larger cohesion, and a larger $\mathcal{B}(C_i)$ gives rise to a larger separation. The average of the silhouette coefficient of all clusters can measure the quality of clustering results effectively. Notice that the value of the silhouette coefficient falls into the interval $[-1, 1]$, and the best clustering results are achieved when $\mathcal{S}(C_i)$ is equal to 1.0.

## 4 Simulations

The simulations are run on a PC equipped with Windows 7, 3.20GHz CPU and 4 GB memory. The trajectory clustering methods are realized by Java language.

### 4.1 Simulation setup

GPS trajectory dataset is obtained from (Microsoft Research Asia) *Geolife* project (Zheng et al. 2010b), which collects the trajectories of 182 users (nodes) during five years (from April 2007 to August 2012). The GPS trajectory of the dataset is represented by a sequence of time stamps, each of which contains latitude, longitude and altitude information. The dataset contains 17621 trajectories with a total distance of 1251654 km and a total duration of 48203 hours. These trajectories record different GPS loggers and GPS telephones, and have various sampling rates. The trajectory of 91% is recorded in dense representation, for example, every 1 to 5 seconds or 5 to 10 meters per point. The trajectory data set can be used in many research fields, such as mobile pattern mining, user activity recognition, location-based social networks, location privacy and location recommendation. Table 1 provides some distance parameter settings.

We set an upper bound for each distance parameter, which indicates that each kind of distance between two trajectory segments should be confined to an upper bound. As the road length is typically far larger than the road width, we set the upper bound of the perpendicular Euclidean distance to 5 meters, and then its normalized value is equal to 0.500 by (9). Likewise, we set the upper bound of the parallel distance to 10 meters, and its normalized value is equal to 0.500 as well. Besides, we assume that the upper bound of the angle

**Table 1** Distance parameter settings

| Distance parameters | Upper bounds | Normalized values |
|---|---|---|
| $d_\perp$ | 5 | 0.500 |
| $d_\parallel$ | 10 | 0.500 |
| $d_\theta$ | $\pi/6$ | 0.134 |
| $d_{\Delta t}$ | 2/3 | 0.333 |
| $d_v$ | 2/3 | 0.333 |
| $d_t$ | 1/2 | 0.500 |

between two trajectory segments is within $\pi/6$, so that we can obtain the normalized value of direction distance equal to 0.134 by (5). With regard to speed distance and duration distance, the upper bounds of speed ratio and duration ratio between two trajectory segments are set to 2/3, and both the two distances are 0.333 by (6) and (7), respectively. The time stamps of two trajectory segments are close to each other, and the upper bound of time distance is set to 30 minutes by (8), which indicates that its normalized value is equal to 0.500.

The upper bound of the parameter $\epsilon$ is calculated by the weighted summation function: $dist_w(L_i, L_j) \approx 0.394$, which is close to the result calculated by the kernel function $dist_k(L_i, L_j) \approx 0.389$. Thus, the weighted summation function is chosen as the distance function in our simulations, and the value of the $\epsilon$ falls into interval [0, 0.394].

## 4.2 Trajectory clustering

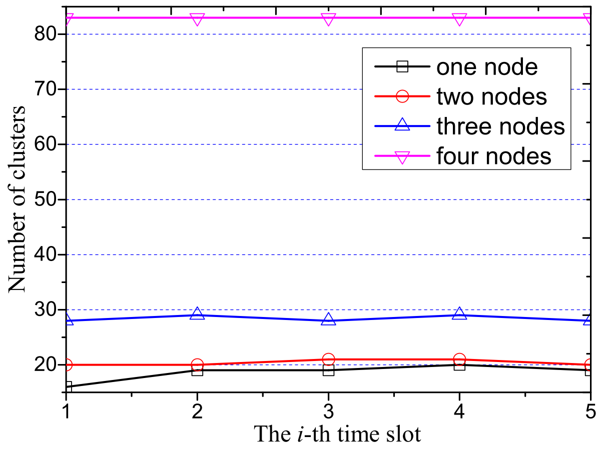
In this section, we observe the impacts of the time sequences and the number of nodes on the trajectory clustering results, in terms of the number of clusters and the silhouette coefficient, as illustrated in Fig. 9.

From Fig. 9a, the number of clusters becomes more stable as the number of nodes increases from 1 to 4, which indicates that the number of clusters is irrelevant with the input order of trajectory segments when enough trajectory data has been provided. Besides, it is obvious that the curve with trajectories of more nodes is much higher than those with trajectories of less nodes. In Fig. 9b, the silhouette coefficient fluctuates slightly with the increase of time sequences while the number of clusters almost keeps as a constant, which is due to the density differences in the trajectory clustering results.

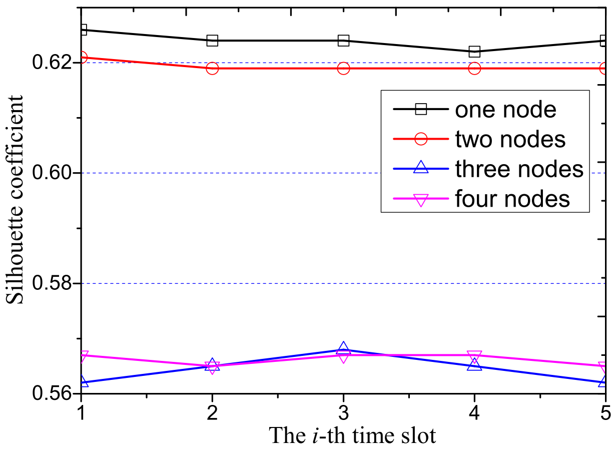
Figure 10 illustrates that the variations of the number of clusters and silhouette coefficient under different $MinPts$. In Fig. 10a, the number of clusters decreases with the increase of $MinPts$ when $Eps$ is fixed. Notice that the number of clusters decreases significantly when the value of $MinPts$ increases from 10 to 25, while the number of clusters reduces slowly when the value of $MinPts$ is larger than 25. In Fig. 10b, the silhouette coefficient increases with the rise of $MinPts$, and we find that the silhouette coefficient is slightly varied when the value of $MinPts$ increases from 25 to 30. As discussed above, $MinPts$ is set to 25 in the following simulations.

Likewise, Fig. 11 shows the number of clusters and silhouette coefficient under different settings of $Eps$. In Fig. 11a, two observations are obtained: ($a$) the curves rise quickly when $Eps$ is smaller than 0.36, and the reason is that the neighborhood of each core trajectory segment rapidly increases with the increase of $Eps$, and thus more clusters are produced; ($b$) the curves descend slowly when $Eps$ is larger than 0.36, which is attributed to the fact that the density of neighborhood of each core trajectory segment has exceeded the size of cluster when $Eps$ continues to increase, while the number of clusters is reduced. Hence, $Eps$ is set to 0.36 in the following simulations. In Fig. 11b, the curves first drop quickly because of the increasing number of cluster, and then the curves rise slowly because the number of clusters number is reduced as aforementioned phenomenon in Fig. 11a.

Therefore, the main parameter settings of JSTTCM are provided in Table 2. The distance radius ($\epsilon$) and the minimum number of neighbors ($MinPts$) are vital to trajectory clustering methods, they are determined by simulations in Figs. 10 and 11. In addition, the time interval threshold ($\Delta t$) is preset to 0.5 hour, which indicates the maximum error of time interval between two trajectory segments does not exceed 30 minutes, it can be considered that it's similar between two trajectory segments in time dimension when setting time interval to 0.5 hour. As for the weight of different distance measures, in fact, it is indeed very hard to



(a) Number of clusters



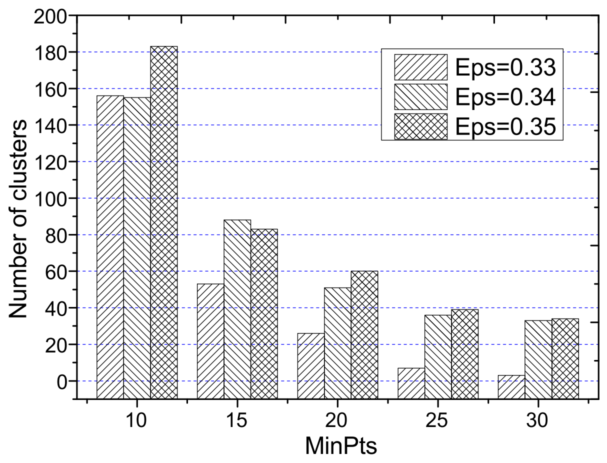
(b) Silhouette coefficient

**Fig. 9** Number of clusters and silhouette coefficient ($Eps(\epsilon) = 0.35$, $MinPts = 15$)

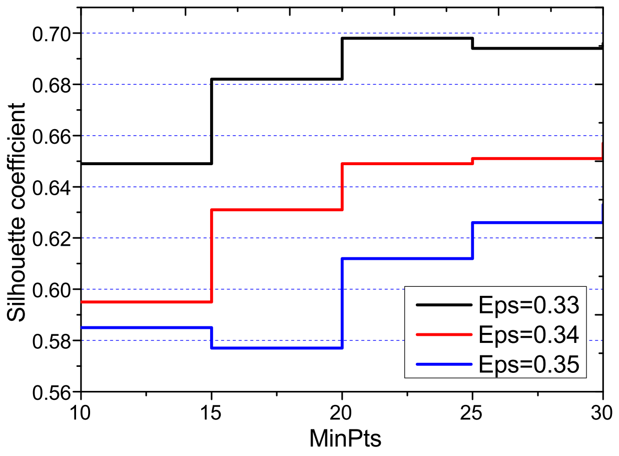
estimate the weight settings by a theoretical analysis, and the optimal settings are always related to the parameters of environments or applications. At the beginning each distance measure is assigned the same weight while the sum of these weights is equal to 1, and then the weight settings can be gradually adjusted through observing the clustering results.

## 4.3 Comparisons of different trajectory clustering methods

Firstly, we compare JSTTCM with trajectory clustering method using Optics, the mechanism of Optics algorithm is very similar to that of JSTTCM, because both of them belong to the density-based clustering algorithms. However, there are differences between JSTTCM and Optics in the clustering process. The Optics algorithm does not generate the clusters explicitly, i.e., it sorts the objects in the data object set and outputs an ordered list of objects.



(a) Number of clusters

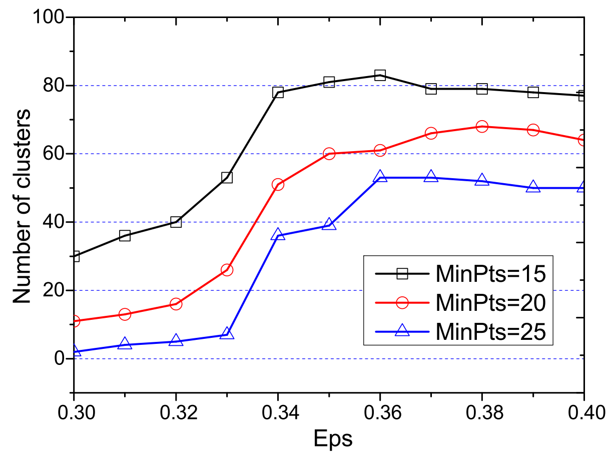


(b) Silhouette coefficient

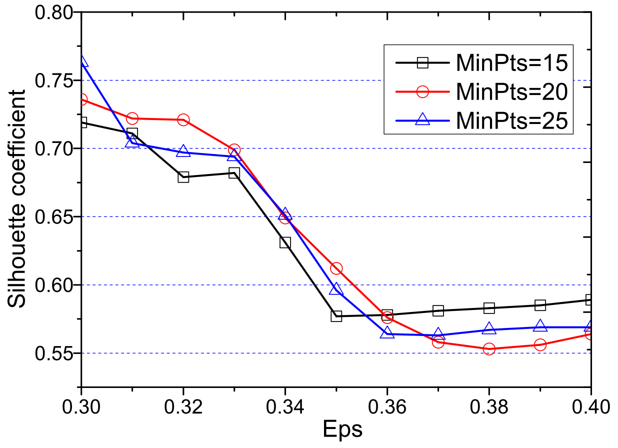
**Fig. 10** Numbers of cluster and silhouette coefficient vs. $MinPts$

The ordered list contains enough information to extract the clusters. The most outstanding advantage of Optics is that it is typically insensitive to the input parameters. The simulation results are given in Fig. 12.

In Fig. 12a, we can observe the variations of the number of clusters and the silhouette coefficient when the number of nodes is varied from 1 to 10. Obviously, the number of clusters increases gradually with the increase of the number of nodes. Specially, Optics always generates more clusters than JSTTCM when a large number of trajectories need to be processed, which is attributed to the fact that Optics has a stronger ability to resist the outlier noise, so that a larger cluster can be divided into several smaller clusters. Besides, note that the number of clusters generated by JSTTCM ascends much more slowly than that of Optics. In Fig. 12b, the silhouette coefficient of JSTTCM outperforms that of Optics especially when there are a large number of trajectories need to be processed, and this



(a) Number of clusters



(b) Silhouette coefficient

**Fig. 11** Numbers of cluster and silhouette coefficient vs. $Eps$

is because more clusters generated by Optics probably lead to a higher coupling of these clusters.

We compare JSTTCM with other trajectory clustering methods including TraClus, CTHD and Extended TraClus. In TraClus, there is an additional filter operation performed

**Table 2** Parameters setting

| Parameters | Values | Description |
|---|---|---|
| $\epsilon$ | 0.36 | distance radius |
| $MinPts$ | 0.25 | minimum number of neighbors |
| $\Delta t$ | 0.5h | time interval threshold |
| $w_1 \sim w_6$ | 1/6 | weight of different distances |

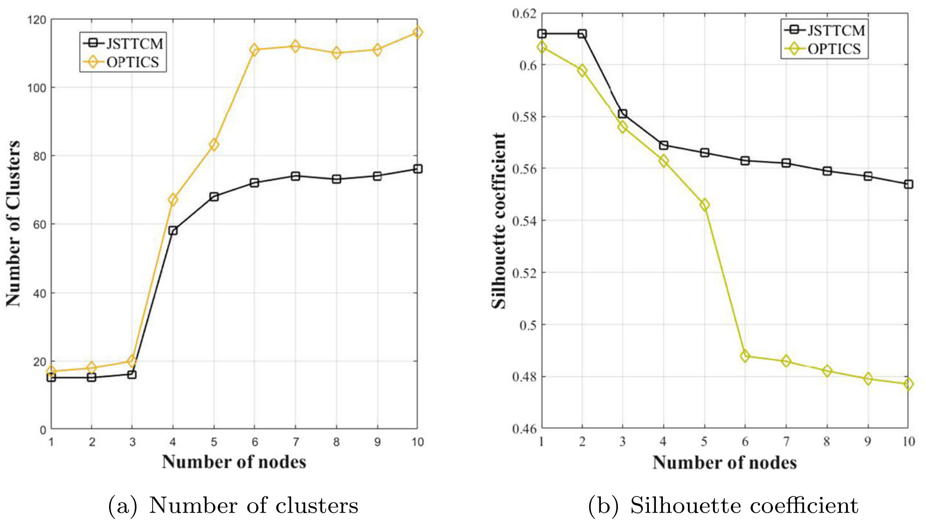
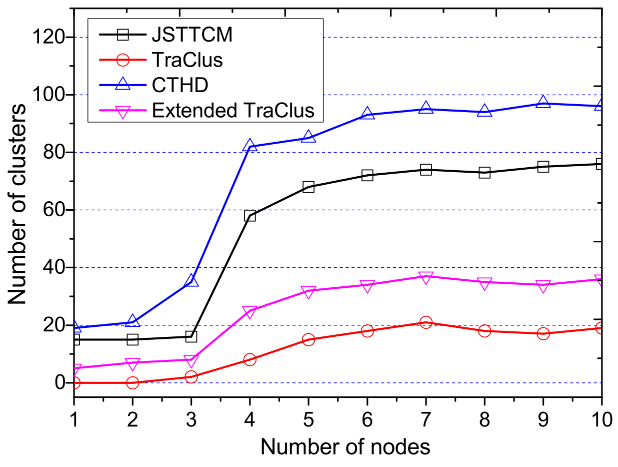


**Fig. 12** Comparisons of different trajectory clustering methods

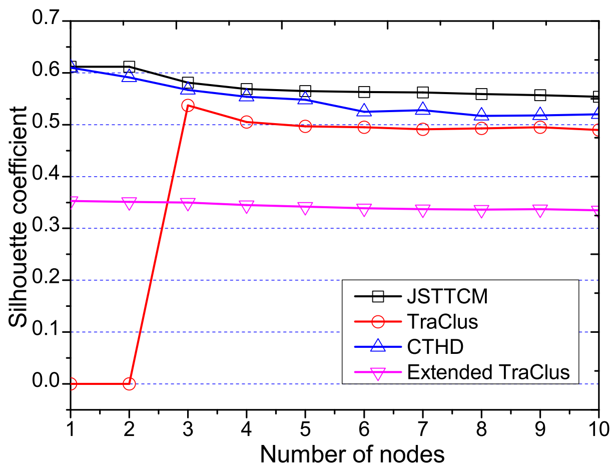
to remove the clusters where the number of trajectory segments are less than a preset threshold. CTHD uses Hausdorff distance that the flow vectors include the positions and directions. Extended TraClus extends HDBSCAN to handle line segments and plug it into TraClus to detect hierarchical clusters. We compare these methods from the aspects of the number of clusters and the silhouette coefficient, and the simulation results are shown in Fig. 13.

We have observed the variations of the number of clusters and the silhouette coefficient with the increase of the number of nodes (from 1 to 10), as shown in Fig. 13. The number of clusters increases gradually when the number of nodes is varied from 1 to 10, and this is because the number of clusters is proportional to the number of trajectories. Specially, the least clusters are produced by TraClus due to its filter mechanism, which filters some clusters where the number of trajectory segments are less than a preset threshold. However, the most clusters are generated by CTHD, which is attributed to the fact that CTHD only considers the positions and directions, so that more clusters are generated due to some simple properties. Extended-TraClus generates more clusters than TraClus due to the discovery of layered clusters. Note that the number of clusters generated by JSTTCM is always between CTHD and Extended TraClus, and this is attributed to the fact that JSTTCM considers spatial-temporal trajectory properties, which brings less clusters than CTHD. From Fig. 13b, the silhouette coefficients of all methods decrease with the increase of the number of nodes slowly, however, the silhouette coefficient of JSTTCM is always larger than those of other methods, which indicates that better clustering results are obtained by JSTTCM. Especially, such gaps are much larger when the trajectories of more nodes have been provided. This is because JSTTCM clusters the trajectory segments through considering these spatial-temporal properties of trajectories, and hence JSTTCM can obtain the cluster structures more appropriately (Fig. 13).

With regard to the time complexity, the trajectory clustering methods in Fig. 13 have the same time complexity equal to $O(n^2)$. Essentially, these trajectory clustering methods are based on DBSCAN algorithm, our proposed method has improved the algorithm efficiency by introducing the timestamp as described in Section 3.2.3.



(a) Number of clusters



(b) Silhouette coefficient

**Fig. 13** Comparisons of different trajectory clustering methods (The dashed segment in Fig. 13b indicates that the silhouette coefficient cannot be obtained while the number of clusters is equal to 0)

## 5 Conclusions

In this paper, we propose a Joint Spatial-Temporal Trajectory Clustering Method (JSTTCM), where some spatial-temporal properties of the trajectories are exploited to cluster the trajectory segments. The simulation results demonstrate that JSTTCM can be used to cluster the trajectory segments appropriately. Future research will focus on an enhancement of JSTTCM, which improves the running efficiency of JSTTCM by building an index structure, and a comprehensive evaluation over various datasets of MSNs will be performed.

**Acknowledgements** This research is supported by National Natural Science Foundation of China under Grant Nos. 61872191, 61872193, 61972210; Six Talents Peak Project of Jiangsu Province under Grant No. 2019-XYDXX-247.

## References

Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V. (2003). Discovering Clusters in Motion Time-series Data. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 375–381).

Andrienko, N., Andrienko, G., Rinzivillo, S. (2016). Leveraging spatial abstraction in traffic analysis and forecasting with visual analytics. *Information System*, *57*, 172–194.

Bao, J., Zheng, Y., Wilkie, D., Mokbel, M. (2015). Recommendations in Location-based Social Networks A Survey. *GeoInformatica*, *19*(3), 525–565.

Besse, P.C., Guillouet, B., Loubes, J.M., Royer, F. (2018). Destination prediction by trajectory Distribution-Based model. *IEEE Transactions on Intelligent Transportation Systems*, *19*(8), 2470–2481.

Campello, R.J.G.B., Moulavi, D., Sander, J. (2013). Density-based Clustering based on Hierarchical Density Estimates. *Lecture Notes in Computer Science*, *7819*, 160–172.

Chen, L., & Ng, R. (2004). On the Marriage of Lp-norms and Edit Distance. *Thirtieth International Conference on Very Large Data Bases*, *30*, 792–803.

Chen, L., Ozsu, M., Oria, V. (2005). Robust and Fast Similarity Search for Moving Object Trajectories. *ACM SIGMOD International Conference on Management of Data*, 491C–502.

Chen, J., Wang, R., Liu, L., Song, J. (2011). Clustering of Trajectories based on Hausdorff Distance. *2011 International Conference on Electronics, Communications and Control (ICECC)* (pp. 1940–1944).

Ester, M., Kriegel, H.P., Sander, J., Xu, X. (1996). Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *2nd International Conference on Knowledge Discovery and Data Mining* (pp. 226–231).

Gaffney, S., & Smyth, P. (1999). Trajectory clustering with mixtures of regression models. *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 63–72).

Lee, J.G., Han, J., Whang, K.Y. (2007). Trajectory Clustering: A Partition-and-Group Framework. *ACM SIGMOD Conference on Management of Data*, 593–604.

Li, Y., He, Z., Nielsen, J., Lachapelle, G. (2015). Using Wi-Fi/Magnetometers for Indoor Location and Personal Navigation. *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*: Banff.

Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R. (2014). Entropy-rate Clustering: Cluster Analysis via Maximizing a Submodular Function Subject to a Matroid Constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(1), 99C112.

Michail, V., Marios, H., Dimitrios, G. (2006). Indexing Multidimensional Time-series. *The VLDB Journal*, *15*(1), 1C-20.

Nanni, M., & Pedreschi, D. (2006). Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, *27*(3), 267–289.

Palma, A.T., Bogorny, V., Kuijpers, B., Alvares, L.O. (2008). A Clustering based Approach for Discovering Interesting Places in Trajectories. *ACM Symposium on Applied Computing*, 863C–868.

Rinzivillo, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., Andrienko, G. (2008). Visually driven analysis of movement data by progressive clustering. *Information Visualization*, *7*(3-4), 225–239.

Rocha, J.A.M.R., Times, V.C., Oliveira, G., Alvares, L.O., Bogorny, V. (2010). DB-SMoT: A Direction-Based Spatio-Temporal Clustering Method. *5th IEEE International Conference Intelligent Systems* (pp. 114–119): London.

Rousseeuw, P.J. (1987). Silhouette: a graphical aid to the interpolation and validation of cluster analysis. *Computational and Applied Mathematics*, *20*, 53–65.

Tang, J., Liu, L., Wu, J. (2019a). A trajectory partition method based on combined movement features. *Wireless Communications and Mobile Computing*, *2019*, 1–13.

Tang, J., Liu, L., Wu, J., Zhou, J., Xiang, Y. (2019b). Joint Spatial-Temporal Trajectory Clustering Method for Mobile Social Networks. *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*: Tianjin China.

Trasarti, R., Pinelli, F., Nanni, M., Giannotti, F. (2011). Mining mobility user profiles for car pooling. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1190–1198).

Xu, H., Zhou, Y., Lin, W., Zha, H. (2015). Unsupervised Trajectory Clustering via Adaptive Multi-Kernel-based Shrinkage. *IEEE International Conference on Computer Vision*, 4328–4336.

Yao, D., Zhang, C., Zhu, Z., Huang, J., Bi, J. (2017). Trajectory clustering via deep representation learning. *International Joint Conference on Neural Networks*, 3880–3887.

Zheng, Y., Chen, Y., Li, Q., Xie, X., Ma, W. (2010a). Understanding transportation modes based on GPS data for web applications. *ACM Transactions on the Web*, *4*(1), 1–36.

Zheng, Y., Xie, X., GeoLife, W.M.A. (2010b). A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, *33*(2), 32–40.

Zhang, D., Lee, K., Lee, I. (2018). Hierarchical Trajectory Clustering for Spatio-temporal Periodic Pattern Mining. *Expert Systems with Applications*, *92*, 1–11.

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.