



Trajectory similarity clustering based on multi-feature distance measurement

Qingying Yu^{1,2,3} · Yonglong Luo^{1,2,3} · Chuanming Chen^{2,3} · Shigang Chen⁴

Published online: 12 January 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

With the development of GPS-enabled devices, wireless communication and storage technologies, trajectories representing the mobility of moving objects are accumulated at an unprecedented pace. They contain a large amount of temporal and spatial semantic information. A great deal of valuable information can be obtained by mining and analyzing the trajectory dataset. Trajectory clustering is one of the simplest and most powerful methods to obtain knowledge from trajectory data, which is based on the similarity measure between trajectories. The existing similarity measurement methods cannot fully utilize the specific features of trajectory itself when measuring the distance between trajectories. In this paper, an enhanced trajectory model is proposed and a new trajectory clustering algorithm is presented based on multi-feature trajectory similarity measure, which can maximize the similarity of trajectories in the same cluster, and can be used to better serve for applications including traffic monitoring and road congestion prediction. Both the intuitive visualization presentation and the experimental results on synthetic and real trajectory datasets show that, compared to existing methods, the proposed approach improves the accuracy and efficiency of trajectory clustering.

Keywords Trajectory · Similarity measure between trajectories · Trajectory features · Trajectory similarity clustering · Trajectory centers

1 Introduction

Nowadays, with the rapid development of satellites, wireless communication, storage and various positioning (such as

GPS, GSM and RFID) technologies, it is possible to collect and store a large number of trajectory data of moving objects, recording information about vehicle location, mobile user activity, animal migration, hurricane movement, etc. A trajectory is a sequence of time-ordered locations for a moving object. It contains a lot of spatial and temporal semantic information. Therefore, analysis on trajectory data has become practically useful [1–5]. For example, the moving trajectories of mobile users can reflect their interests and favorites. Analysis of these trajectories can help in planning road system, recommending tourist routes and sharing life experience [6]. One of the typical analysis tasks is trajectory clustering, which is also one of the simplest and most powerful methods for acquiring knowledge from trajectory data [7–9].

Trajectory clustering is a process of assigning a set of similar trajectories into groups (called clusters), so that the trajectories within each cluster are highly similar, and there is low similarity among different clusters [10, 11]. Its purpose is to extract the common motion characteristics of similar moving objects, in order to predict the behaviors of moving objects or to provide the decision-making guidance for management in many fields, such as location recommendation,

✉ Yonglong Luo
ylluo@ustc.edu.cn

Qingying Yu
ahnuyqu@mail.ahnu.edu.cn

Chuanming Chen
ccm_0@163.com

Shigang Chen
sgchen@cise.ufl.edu

¹ School of Geography and Tourism, Anhui Normal University, Wuhu 241003, China

² School of Computer and Information, Anhui Normal University, Wuhu 241003, China

³ Anhui Provincial Key Laboratory of Network and Information Security, Wuhu 241003, China

⁴ Department of Computer, Information Science, Engineering, University of Florida, Gainesville, FL 32611, USA

market research, vehicle destination prediction, urban planning, weather forecast [1, 2, 7, 12–14], etc. Much has been achieved in clustering research. The representative algorithms include k -means [15], DBSCAN [16], BIRCH [17] and OPTICS [18], which mainly focus on clustering of point data objects but cannot be directly used in trajectory clustering, because the distance measurement between trajectories is very complicated. In traditional clustering, each data point is a separate object. However, in trajectory clustering, each trajectory including several closely related sampling points is the basic object. Clustering of time series is another kind of clustering problem. Some application results have been achieved [19, 20]. For example, Literature [19] produced day-ahead forecasts of wind power by time series cluster analysis, where a Quality Threshold (QT) algorithm is adopted to cluster the time series. There are some similarities between time series and trajectories because both of them are sequences with time attributes, but the dissimilarity calculation in both problems cannot be generalized. The intrinsic characteristics and the movement trend of a trajectory cannot be utilized by these traditional clustering algorithms. New methods are needed to address the trajectory clustering problem. The key issue is how to measure the similarity degree between trajectories. An appropriate and valid model for defining the similarity between trajectories is critical to the quality of trajectory clustering. Clustering results with high accuracy play an important role in management decisions. For example, the municipal traffic department can plan urban roads and alleviate road congestion problems based on the cluster analysis of vehicle trajectories. The higher the accuracy of clustering results, the greater the guiding significance provided, otherwise misdirection will occur.

This paper presents a new similarity measurement method based on multiple trajectory features and a corresponding trajectory similarity clustering algorithm. The key contributions of this paper include:

- 1) An enhanced trajectory model is introduced for trajectory analysis.
- 2) A Multi-feature Trajectory Similarity Measure algorithm (MFTSM) is proposed to calculate the distance between any two trajectories based on multiple features of trajectories. In particular, the location distance based on area calculation is used to address the continuity problem of trajectory.
- 3) A trajectory clustering algorithm based on MFTSM is proposed, denoted as TC_MFTSM, in which the initial centers of the clusters are optimized based on contemporary trajectories.
- 4) Several comparative experiments are performed to verify the effectiveness and evaluate the efficiency of the

proposed algorithms. The experimental results demonstrate that our algorithms outperform the prior work on trajectory similarity clustering.

The rest of this paper is organized as follows. Section 2 introduces the related work. In Section 3, we introduce an enhanced trajectory model. In Section 4, we describe the rate and time interval of contemporary trajectories, as well as an algorithm RTofCT, which will be used later. Section 5 introduces the trajectory similarity measure algorithm, MFTSM. Section 6 introduces the trajectory clustering algorithm based on MFTSM. Experimental results and analysis are presented in Section 7. Section 8 concludes the paper and provides future research directions.

2 Related work

There are mainly two ways to address the problem of trajectory clustering: 1) Define a specific trajectory clustering algorithm based on the intrinsic properties of trajectories. For example, Wei et al. [21] proposed a trajectory clustering method based on the regression model. In their method, a trajectory is approximately represented by fitting polynomial. 2) Two-stage method. Firstly, design a similarity (distance) measurement method based on trajectory data. Secondly, improve some traditional clustering algorithm to realize trajectory clustering. The vast majority of studies are carried out in the second way. This paper also focuses on trajectory clustering in this way, where the emphasis is trajectory similarity measurement [2, 22–25]. At present, there are many related research results in this field.

Lee et al. [12] proposed a partition-and-group framework for clustering trajectories, i.e., to divide each trajectory into several t -partitions according to the idea of line segment Hausdorff distance in image processing, in order to represent the local features of complex trajectories. And then they used the traditional density-based clustering method DBSCAN to find the general sub-trajectories from the trajectory database. This method solves the problem of comparisons between complex trajectories. However, the distance calculation between two trajectories is based on spatial locations. Each t -partition is only the line between two endpoints of the trajectory segment, which is an approximate description of the trajectory, resulting in the loss of local features. Lin et al. [26] proposed a similarity measurement method named OWD (One Way Distance), which merely focuses on spatial shapes of moving trajectories without considering the time dimension. Morris et al. [27] compared various trajectory similarity clustering methods, and proposed that performance is actually dictated by the trajectory properties encountered

in a dataset. Ferreira et al. [24] proposed a trajectory similarity measurement method based on vector field, and then used k -means clustering method to achieve trajectory clustering. Domingo-Ferrer et al. [28] proposed a trajectory similarity calculation method based on the Euclidean distance between each point pair on the trajectory, which considers the time and space dimensions while ignoring other factors such as trajectory orientation, moving speed and continuity feature. Similarly, Wang et al. [29] proposed a similarity measurement method that takes trajectory shape into account on the basis of Literature [28]. It also does not consider the continuity of the trajectory. Gudmundsson et al. [30] proposed a subtrajectory clustering approach based on Fréchet distance, which merely considers a trajectory as a directed curve in 2D. Besse et al. [31] proposed a new distance SSPD (Symmetrized Segment-Path Distance). It is time insensitive and compares the shape and physical distance between two trajectories. Sanchez et al. [7] proposed hashing techniques based on DBH (Distance-Based Hashing) and LSH (Locality Sensitive Hashing) for fast trajectory similarity clustering. In their method, trajectory similarity measurement is based on DTW (Dynamic Time Warping) distance [32, 33] and trajectory clustering is realized using k -means algorithm. The above methods ignore the comprehensive impact of the internal and external characteristics of trajectory itself on the similarity between trajectories.

When calculating the distance between two trajectory segments, most existing methods use the center point and the length of trajectory segment as standards. For example, Zhang et al. [34] proposed a new trajectory clustering algorithm which considers semantic spatio-temporal information based on Traclus algorithm [12], where spatial distance consisting of perpendicular distance and horizontal distance is calculated based on those standards. Such methods ignore the continuity of trajectory. In this paper, the location distance is calculated using the area of quadrangle formed by two trajectory segments. For each pair of segments, our value of location distance is more accurate. In addition, we propose an enhanced trajectory k -means clustering method, in which the selection of initial trajectory centers is optimized based on the temporal features of trajectories. So we have made improvements in both stages of trajectory clustering.

In summary, there are a variety of trajectory similarity clustering methods, but almost none of them is widely approved. Aiming at the above problems, based on the characteristics of orientation, speed, shape, location and continuity of the trajectory, we propose a more accurate similarity measurement method based on trajectory multi-feature. And a trajectory similarity clustering algorithm is consequently proposed in this paper.

3 Preliminary concept and problem definition

3.1 Trajectory model

Definition 1 (Time-stamped location) [28]: Let t be a timestamp and (x, y) be a location in \mathbb{R}^2 . A time-stamped location is defined as a triple (t, x, y) , which means that an object is at location (x, y) at time t .

Hereinafter, triple and location will be used as synonyms for time-stamped location.

Definition 2 (Trajectory): A trajectory is an ordered set of time-stamped locations, denoted as T .

$$T = \{Tid, (t_1, x_1, y_1), (t_2, x_2, y_2), \dots, (t_n, x_n, y_n)\} \quad (1)$$

where $t_r < t_{r+1}$ for all $1 \leq r < n$, Tid is the identification of the trajectory, and n is the number of sampled points in the trajectory.

Consider a set of p trajectories $TS = \{T_1, T_2, \dots, T_p\}$, where $T_i = \{i, (t_1^i, x_1^i, y_1^i), \dots, (t_m^i, x_m^i, y_m^i)\}$ represents the i -th trajectory in TS , $1 \leq i \leq p$.

Figure 1 shows an example in which four trajectories are provided. The sampled points in each trajectory can be projected to the $\mathbb{R}^2(x-y)$ plane.

Definition 3 (Length of trajectory): Consider a trajectory T . The length of T is defined as the number of triples in T . It is denoted as $|T|$.

For example, $|T| = n$ in (1).

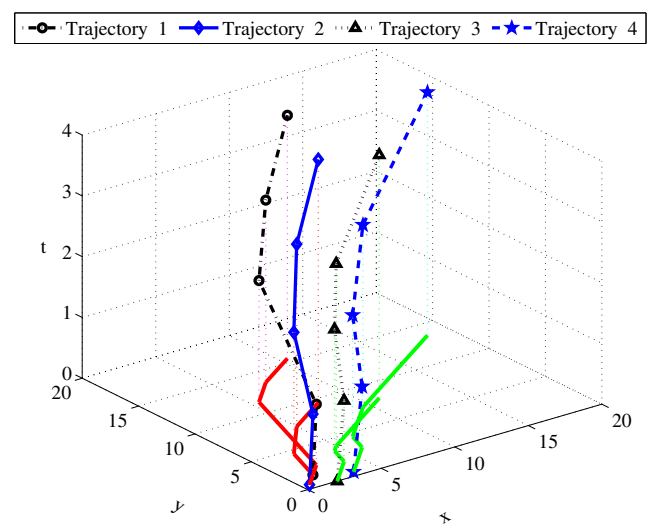


Fig. 1 The motion trajectories of different moving objects and their projections in the 2D-plane

Definition 4 (*Sub-trajectory*) [28]: Consider a trajectory S . $S = \{Sid, (t_{i_1}, x_{i_1}, y_{i_1}), (t_{i_2}, x_{i_2}, y_{i_2}), \dots, (t_{i_m}, x_{i_m}, y_{i_m})\}$ is defined as a sub-trajectory of T in (1), denoted as $S \preceq T$, where $1 \leq i_1 < \dots < i_m \leq n$.

Definition 5 (*Trajectory segment*): Consider a trajectory T . A trajectory segment is defined as the line segment between any adjacent pair of time-stamped locations $((t_r, x_r, y_r), (t_{r+1}, x_{r+1}, y_{r+1}))$ in the trajectory T , $1 \leq r < n$.

In analyzing trajectory data, it is necessary to know the interrelation between different trajectories. We use *pt%*-contemporary defined below to indicate whether two trajectories have common timestamps or not.

Definition 6 (*pt%-contemporary trajectories*) [28]: Consider two trajectories $T_i = \{i, (t_1^i, x_1^i, y_1^i), \dots, (t_m^i, x_m^i, y_m^i)\}$ and $T_j = \{j, (t_1^j, x_1^j, y_1^j), \dots, (t_n^j, x_n^j, y_n^j)\}$. Their *pt%*-contemporary value pt is defined as:

$$pt = 100 \cdot \min \left(\frac{\Delta t}{t_m^i - t_1^i}, \frac{\Delta t}{t_n^j - t_1^j} \right), \quad (2)$$

where Δt is calculated as

$$\Delta t = \max((\min(t_m^i, t_n^j) - \max(t_1^i, t_1^j)), 0). \quad (3)$$

If $pt = 0$, the two trajectories are not contemporary. If and only if they start at the same time and end at the same time, then $pt = 100$. Denote the overlap time of two trajectories T_i and T_j as $ol(T_i, T_j)$, which starts at $\max(t_1^i, t_1^j)$ and ends at $\min(t_m^i, t_n^j)$. Hence, $ol(T_i, T_j) = \{\max(t_1^i, t_1^j), \dots, \min(t_m^i, t_n^j)\}$.

Consider any two *pt%*-contemporary trajectories T_i and T_j for $pt > 0$, we assume that they have the same number of time-stamped locations within $ol(T_i, T_j)$ and those correspond to the same time-stamps. Therefore, in our following method, a process of locations interpolation is firstly conducted for the original trajectory dataset TS according to the algorithm proposed in Literature [28].

Definition 7 (*Trajectory outlier*): $\forall T_i \in TS$, it is called as a trajectory outlier (also known as an outlying trajectory) if it is not contemporary with any other trajectory T_j in TS . That is, $pt = 0$ for T_i and T_j , $j = 1 \dots p$, $j \neq i$.

3.2 Distance between two trajectories

As described in Sections 1 and 2, it is most important to design a suitable similarity measurement method (a certain type of distance between two trajectories) in trajectory clustering. Because trajectories have temporal and spatial

characteristics, many existing methods [28, 35] for measuring distance between two trajectories mainly consider the temporal and spatial factors while ignoring the shape and other trajectory characteristics, such as trajectory segment orientation, speed, continuity, etc. We propose a new trajectory similarity measure based on multiple features. Some related concepts and notions are given as follows.

Definition 8 (*The r -th segment vector*): The r -th segment vector of T_i , denoted as seg_r^i , refers to the directed path of a moving object in the r -th trajectory segment of the i -th trajectory. It is defined as:

$$seg_r^i = (t_{r+1}^i - t_r^i, x_{r+1}^i - x_r^i, y_{r+1}^i - y_r^i) \quad (4)$$

where (t_r^i, x_r^i, y_r^i) and $(t_{r+1}^i, x_{r+1}^i, y_{r+1}^i)$ respectively represent the r -th and $(r+1)$ -th locations of the i -th trajectory, $1 \leq r < m$.

Definition 9 (*The r -th segment speed*): The r -th segment speed of T_i , denoted as $seg_sp_r^i$, refers to the movement speed of a mobile object in the r -th trajectory segment of the i -th trajectory. It is defined as:

$$seg_sp_r^i = \frac{\sqrt{(y_{r+1}^i - y_r^i)^2 + (x_{r+1}^i - x_r^i)^2}}{t_{r+1}^i - t_r^i} \quad (5)$$

where (t_r^i, x_r^i, y_r^i) represents the r -th triple of the i -th trajectory, $1 \leq r < m$.

Definition 10 (*Trajectory orientation distance*): $\forall T_i, T_j \in TS$, the trajectory orientation distance, denoted as $dist_o(T_i, T_j)$, refers to the distance between the two trajectories calculated based on the angles between two segment vectors. If T_i and T_j are *pt%*-contemporary trajectories with $pt > 0$,

$$dist_o(T_i, T_j) = \frac{1}{pt\% \cdot (|ol(T_i, T_j)| - 1)} \sum_{r=st_{ij}}^{et_{ij}-1} \arccos \left(\frac{seg_r^i \bullet seg_r^j}{|seg_r^i| \cdot |seg_r^j|} \right) \quad (6)$$

where “ \bullet ” is the dot product operator, “arccos” function is used to calculate the angle between two vectors, pt is calculated by (2), st_{ij} and et_{ij} are respectively the start time and end time of $ol(T_i, T_j)$, which is defined in Definition 6, and finally seg_r^i and seg_r^j are calculated by (4).

Based on the different trajectory orientations, there are four different scenarios for the angle θ between two trajectory segment vectors, which are shown in Fig. 2.

Definition 11 (*Trajectory speed distance*): $\forall T_i, T_j \in TS$, the trajectory speed distance, denoted as $dist_s(T_i, T_j)$, refers

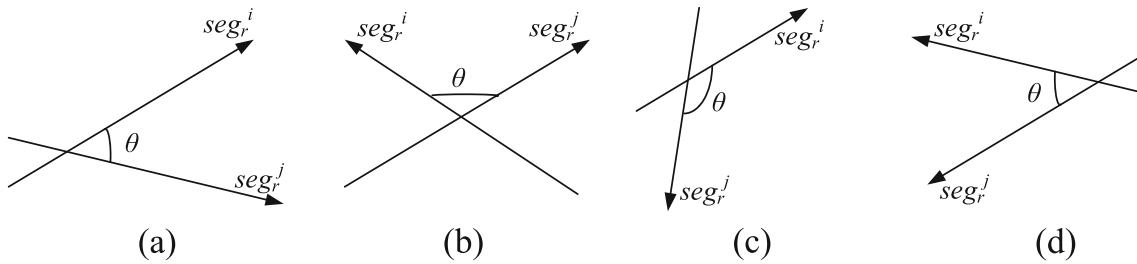


Fig. 2 Four different scenarios for the angle θ between two trajectory segment vectors

to the distance between the two trajectories calculated based on segment speed. If T_i and T_j are $pt\%$ -contemporary trajectories with $pt > 0$,

$$dist_s(T_i, T_j) = \frac{1}{pt\% \cdot (|ol(T_i, T_j)| - 1)} \times \sqrt{\sum_{r=st_{ij}}^{et_{ij}-1} (seg_sp_r^i - seg_sp_r^j)^2}, \quad (7)$$

where $seg_sp_r^i$ and $seg_sp_r^j$ are calculated by (5), and the other parameters are the same as defined earlier.

Definition 12 (Trajectory location distance): We call $triple_r^i = (t_r^i, x_r^i, y_r^i)$ the r -th time-stamped location of trajectory T_i . $\forall T_i, T_j \in TS$, the trajectory location distance, denoted as $dist_l(T_i, T_j)$, refers to the distance between the two trajectories calculated based on the time-stamped locations. If T_i and T_j are $pt\%$ -contemporary trajectories with $pt > 0$,

$$dist_l(T_i, T_j) = \frac{1}{pt\% \cdot (|ol(T_i, T_j)| - 1)} \sum_{r=st_{ij}}^{et_{ij}-1} \sqrt{\sigma_r}, \quad (8)$$

where σ_r represents the sum of areas of two triangles consisting of the four time-stamped locations $triple_r^i, triple_r^j, triple_{r+1}^i$ and $triple_{r+1}^j$, it is calculated based on the following equations, $1 \leq r, s < n$ and $1 \leq i, j \leq p$.

$$dt(triple_r^i, triple_s^j) = \sqrt{(x_r^i - x_s^j)^2 + (y_r^i - y_s^j)^2} \quad (9)$$

$$\alpha^i = dt(triple_r^i, triple_{r+1}^i) \quad (10)$$

$$\beta_r = dt(triple_r^i, triple_r^j) \quad (11)$$

$$\gamma = dt(triple_{r+1}^i, triple_{r+1}^j) \quad (12)$$

$$\mu_1 = \frac{\alpha^i + \beta_r + \gamma}{2} \quad (13)$$

$$\sigma_{r,1} = \sqrt{|\mu_1(\mu_1 - \alpha^i)(\mu_1 - \beta_r)(\mu_1 - \gamma)|} \quad (14)$$

$$\mu_2 = \frac{\alpha^j + \beta_{r+1} + \gamma}{2} \quad (15)$$

$$\sigma_{r,2} = \sqrt{|\mu_2(\mu_2 - \alpha^j)(\mu_2 - \beta_{r+1})(\mu_2 - \gamma)|} \quad (16)$$

$$\sigma_r = \sigma_{r,1} + \sigma_{r,2} \quad (17)$$

Definition 13 (Trajectory distance): $\forall T_i, T_j \in TS$, the trajectory distance, denoted as $dist(T_i, T_j)$, refers to the distance between the two trajectories. If T_i and T_j are $pt\%$ -contemporary trajectories with $pt > 0$, we define

$$dist(T_i, T_j) = \eta_o \cdot dist_o(T_i, T_j) + \eta_s \cdot dist_s(T_i, T_j) + (1 - \eta_o - \eta_s) \cdot dist_l(T_i, T_j), \quad (18)$$

where $\eta_o, \eta_s \in [0, 1]$, respectively represent the weights of trajectory orientation distance and speed distance.

If $pt = 0$, i.e. T_i and T_j are not contemporary, but there is at least one trajectory $T_{ijk} \in \tau \subseteq TS$, such that both (T_i, T_{ijk}) and (T_j, T_{ijk}) are $pt\%$ -contemporary with $pt > 0$, then

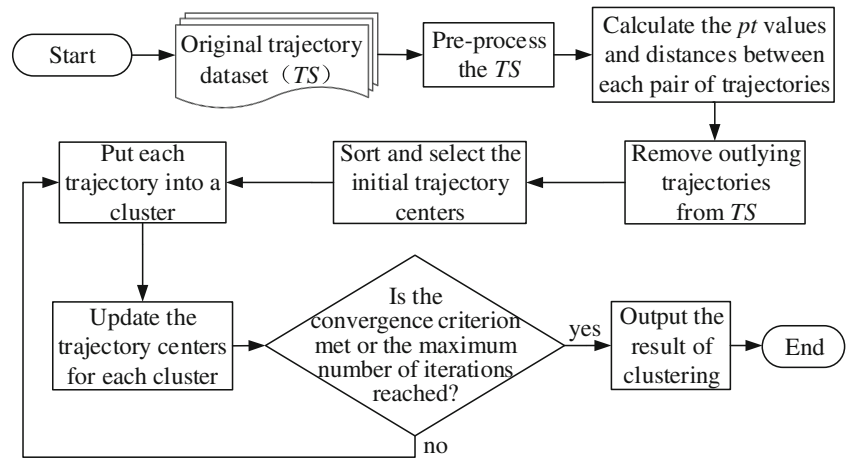
$$dist(T_i, T_j) = \min_{T_{ijk} \in \tau} (dist(T_i, T_{ijk}) + dist(T_j, T_{ijk})). \quad (19)$$

Otherwise, $dist(T_i, T_j)$ is not defined. Where τ is a set of trajectories being contemporary with both T_i and T_j .

4 Rate and time interval of contemporary trajectories

A schematic diagram for the proposed trajectory clustering algorithm is provided in Fig. 3. After pre-processing the original trajectory dataset, the pt values and distances between each pair of trajectories are calculated, and then the trajectory clustering process is completed. The various technical aspects of the proposed algorithm will be described in detail in this section and the next two sections.

Fig. 3 Schematic diagram for the proposed trajectory clustering algorithm



In our trajectory similarity measurement method, we take the temporal factor into consideration. The calculation of trajectory distance is based on the rate and time interval of contemporary trajectories, which are calculated by Algorithm 1, denoted with RTofCT.

The notations used in the RTofCT algorithm are listed in Table 1.

There are three steps in Algorithm 1. First, we pre-process the trajectory dataset TS in order to formalize each trajectory according to (1) (Line 1), and initialize the related variables used later (Lines 2-3). Second, the start time and the end time of each trajectory in TS are calculated and saved into the matrix **TIME** (Lines 4-9). Third, the pt values between each pair of trajectories in TS are calculated and saved into the matrix **PT** (Lines 10-17). The matrixes **PT** and **TIME** are calculated based on Definition 6.

The pseudo code of the RTofCT algorithm is given as follows:

Table 1 Notations used in the RTofCT algorithm

Notation	Description
TS	A dataset of p trajectories
$ TS $	The number of trajectories in TS
Δt	The parameter used to calculate the contemporary rate between two trajectories T_i and T_j
pt	The contemporary rate between two trajectories T_i and T_j
TIME	The matrix which consists of the start and end time of each trajectory
PT	The matrix which consists of the pt values for each pair of trajectories
TIME (i)	The i -th row of TIME , recording the start and the end time of the i -th trajectory T_i
PT (i, j)	The i -th row and j -th column of PT , recording the contemporary rate between T_i and T_j

Algorithm 1 RTofCT: Rate and time interval of contemporary trajectories

Input:

$$TS = \{T_1, T_2, \dots, T_p\}$$

Output:

TIME and **PT**

- 1: Pre-process the trajectory dataset TS , s.t. each $T_i \in TS$ is in the form: $T_i = \{i, (t_1^i, x_1^i, y_1^i), \dots, (t_m^i, x_m^i, y_m^i)\}$;
- 2: Initialize the two matrixes **TIME** and **PT**;
- 3: $p \leftarrow |TS|$;
- 4: **for** $i \leftarrow 1$ to p **do**
- 5: //Calculate the start time and the end time of the i -th trajectory T_i ;
- 6: $start_i \leftarrow$ the start time of T_i ;
- 7: $end_i \leftarrow$ the end time of T_i ;
- 8: **TIME**(i) $\leftarrow (i, start_i, end_i)$;
- 9: **end for**
- 10: **for** $i \leftarrow 1$ to $p-1$ **do**
- 11: **for** $j \leftarrow i+1$ to p **do**
- 12: Calculate Δt using (3);
- 13: Calculate pt using (2);
- 14: **PT**(i, j) $\leftarrow pt$;
- 15: **PT**(j, i) \leftarrow **PT**(i, j);
- 16: **end for**
- 17: **end for**
- 18: **return** **TIME**, **PT**;

The time complexity of Algorithm 1 depends on the following two parts: (a) the time for computing the **TIME** matrix, whose time complexity is $O(p)$, where p is the number of trajectories in dataset TS , i.e. $p = |TS|$; (b) the time to calculate the **PT** matrix. From Lines 10-17 in Algorithm 1, we know that the values of pt must be calculated between each pair of trajectories. Hence, the number of calculations for this part is $p \cdot (p - 1) / 2$. If the value of p is close to infinity, the time complexity of this part is $O(p^2)$.

So the total approximate time complexity of Algorithm 1 is $O(p^2)$. The space complexity of Algorithm 1 is $O(p^2)$, which is mainly due to storing the **PT** matrix.

Before all the algorithms are executed, *TS* has been interpolated based on the method mentioned in Section 3.1. In the following algorithms, *TS* as an input dataset, is assumed to have been pre-processed in the same way as the first step in Algorithm 1.

5 Trajectory similarity measure algorithm

According to the clustering results of trajectory dataset, a great deal of valuable information can be extracted. The similarity measure between trajectories is the basis of trajectory clustering.

Figure 4 shows two trajectories, each of which contains five time-stamped locations at times 0, 1, 2, 3 and 4, respectively. Each pair of the corresponding trajectory segments moves in the same direction. By calculating the Euclidean distance between each pair of the corresponding sampling points, it is easy to get the distance between these two trajectories. However, according to the moving directions and projections on the x - y plane of the two trajectories, we find that the similarity calculation method based only on the distance of the corresponding sampling points of the trajectories is not adequate to achieve accuracy in the similarity measure. As is shown in Fig. 4, the path in the period of time 2-4 of Trajectory 1 is exactly the same as the one in the period of time 0-2 of Trajectory 2. The similarity degree of these two trajectories will be very high if only the shape and locations in the x - y plane are considered. Obviously, if time factor is ignored in distance calculation, the distance between two trajectories will not reflect the real situation. An analysis based on this distance may cause erroneous conclusions, which may lead to wrong guidance in the real applications.

Therefore, in calculating the distance between two trajectories, time, location, shape, speed, continuity and other features of trajectory should be taken into account. The smaller the distance between two trajectories is, the greater the degree of similarity between them is.

The trajectory similarity measurement algorithm based on multiple features is described in Algorithm 2, which is the basis of our trajectory clustering algorithm. According to the value of pt , it is divided into three cases. Lines 8-26 give the distance calculation method. First, if $pt > 0$, the two trajectories are contemporary. The distance between them is calculated based on trajectory orientation distance, speed distance, and location distance. In the calculation of each of the three distances, the time aspect of trajectory is considered based on the value of pt . Equations (4)–(18) are

used in this case (Lines 8-15). Second, if pt is equal to 0, the two trajectories are not contemporary. The distance measurement between them is based on the intermediate ones that overlap partially each of them. For this case, the two trajectories have a certain connection in terms of time attribute, but the absolute time can not be obtained for distance calculation. So the third party trajectories are used to realize the distance measurement between them. Equation (19) is used here (Lines 16-23). Third, if pt can not be obtained, indicating that at least one of the trajectories has a single location, which is most likely to be a noisy point. The distance related to such trajectory is meaningless, therefore, it is assigned with Inf in this case.

Algorithm 2 MFTSM: Multi-feature Trajectory Similarity Measure

Input:

any two trajectories T_i and T_j , **TIME**, η_o (the weight of trajectory orientation distance), η_s (the weight of trajectory speed distance)

Output:

$dist(T_i, T_j)$

- 1: $(start_i, end_i) \leftarrow$ the start and the end time of T_i ;
 - 2: $(start_j, end_j) \leftarrow$ the start and the end time of T_j ;
 - 3: $st_{ij} \leftarrow \max(start_i, start_j)$;
 - 4: $et_{ij} \leftarrow \min(end_i, end_j)$;
 - 5: Calculate Δt using (3);
 - 6: Calculate pt using (2);
 - 7: $interval \leftarrow et_{ij} - st_{ij}$; // i.e. $|ol(T_i, T_j)| - 1$
 - 8: **if** ($pt > 0$) **then**
 - 9: $//r = st_{ij}, \dots, et_{ij} - 1$
 - 10: Respectively calculate seg_r^i and seg_r^j using (4);
 - 11: Calculate $dist_o(T_i, T_j)$ using (6);
 - 12: Respectively calculate $seg_{sp_r}^i$ and $seg_{sp_r}^j$ using (5);
 - 13: Calculate $dist_s(T_i, T_j)$ using (7);
 - 14: Calculate $dist_t(T_i, T_j)$ using (8)-(17);
 - 15: Calculate $dist(T_i, T_j)$ using (18); // η_o, η_s are needed here
 - 16: **else if** ($pt == 0$) **then**
 - 17: **for** $ijk \leftarrow 1$ to $|TS|$ **do**
 - 18: Calculate the $pt1$ between T_i and T_{ijk} ;
 - 19: Calculate the $pt2$ between T_j and T_{ijk} ;
 - 20: **if** ($pt1 > 0 \ \&\& \ pt2 > 0$) **then**
 - 21: Calculate $dist(T_i, T_j)$ using (19);
 - 22: **end if**
 - 23: **end for**
 - 24: **else**
 - 25: $dist(T_i, T_j) \leftarrow \text{Inf}$;
 - 26: **end if**
 - 27: **return** $dist(T_i, T_j)$;
-

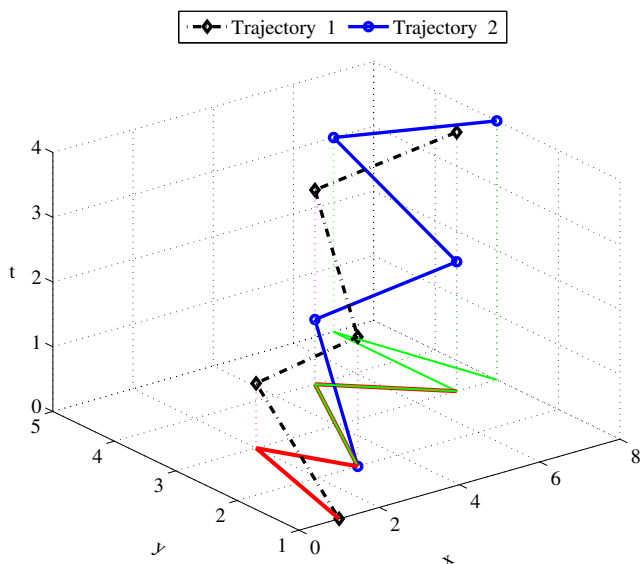


Fig. 4 Two trajectories having the same locations at different time stamp

To make the MFTSM algorithm better understood, a specific example is given as follows:

Example 1 Consider two trajectories T_{16} and T_{17} selected from the synthetic dataset introduced in Section 7.1. In order to facilitate demonstration, the two trajectories we choose are relatively short. $T_{16} = \{16, 2, 7827, 17747, 3, 7882.24, 17767.72, \dots, 49, 11437.04, 17720.53\}$, $T_{17} = \{17, 2, 4286, 15523, 3, 4265.75, 15467.59, \dots, 19, 3817.73, 14642.22\}$. According to Lines 1-2 of Algorithm 2, $(start_{16}, end_{16})=(2,49)$, $(start_{17}, end_{17})=(2,19)$. Then, as is shown in Lines 3-6, the Δt and pt values between T_{16} and T_{17} are calculated based on (2) and (3). That is, $\Delta t = \max((\min(end_{16}, end_{17}) - \max(start_{16}, start_{17})), 0) = \max((19 - 2), 0) = 17$, and $pt = 100 \cdot \min(\Delta t / (end_{16} - start_{16}), \Delta t / (end_{17} - start_{17})) = 100 \cdot \min(17 / (49 - 2), 17 / (19 - 2)) = 36.17$. Based on Line 7, the value of $(et_{ij} - st_{ij})$ is calculated, the result is 17. Because $pt > 0$, the value of $dist(T_{16}, T_{17})$ is calculated based on Lines 8-15. According to the corresponding steps, the result is as follows: $dist(T_{16}, T_{17}) = 19.0441$.

Note that MFTSM algorithm is able to compare trajectories that are time-wise overlapping only partially or not at all. A lemma proved in Literature [28] guarantees that any two trajectories at minimum distance for clustering must have some overlapping time span. Therefore, the distance between two non-overlapping trajectories can be calculated but is too large. So any pair of non-overlapping trajectories is hard to be clustered.

Algorithm 2 is used to calculate the distance between any two trajectories, whose time complexity mainly depends on the distance calculation. For one thing, the time complexity of Algorithm 2 is $O(interval)$ when $pt > 0$, where $interval$ is the common duration of any two trajectories. For another, from Lines 16-23 in Algorithm 2, we know that if the condition $pt == 0$ is met, the number of calculations for the MFTSM algorithm is $interval \cdot |TS|$. The other steps in the algorithm only need to be calculated once. Hence, the total approximate time complexity of Algorithm 2 is $O(interval \cdot |TS|)$. In general, $interval$ is much smaller than $|TS|$. For example, in our later experiments, the maximum value of $interval$ is 101 and $|TS|$ is 1000 for synthetic dataset. The space complexity of Algorithm 2 is $O(|TS|)$, which is mainly due to storing **TIME** (size: $|TS| \cdot 3$) and the intermediate results $dist(T_i, T_{ijk})$ and $dist(T_j, T_{ijk})$.

Based on the MFTSM algorithm, we calculate the distance matrix with a size of $|TS| \cdot |TS|$ to better serve the later research. The distances between each pair of trajectories need to be calculated and stored, so the time complexity of this calculation process is $O(interval \cdot |TS|^2)$. The space complexity is $O(|TS|^2)$.

6 Trajectory clustering algorithm based on MFTSM

There are similarities and differences between trajectory clustering and point data objects clustering. The same thing is the general idea of clustering, the main difference is the distance calculation method. The original trajectories are partitioned into several clusters based on some similarity measure metric. Compared with the original k -means algorithm, our method makes several improvements to address the different object — trajectory.

In our trajectory clustering algorithm, there are two main features: 1) The distance between any two trajectories is calculated based on Algorithm 2 (MFTSM). 2) The trajectory clustering method is an enhanced k -means clustering process. First, trajectory outliers are removed from the original trajectory dataset based on the concept of contemporary trajectories. Second, instead of randomly generating, our method selects initial trajectory centers based on the time duration of each trajectory. In particular, the top num_c trajectories with the longest time duration are used as the initial centres, where num_c is the number of trajectory clusters. Third, the number of trajectories within each trajectory cluster is approximately consistent, the value of “ k ” is consequently determined. This setting can also be applied to other applications such as trajectory privacy preservation.

The proposed trajectory clustering algorithm, denoted as TC_MFTSM, implements trajectory clustering based on MFTSM algorithm. The algorithm TC_MFTSM considers both temporal and spatial attributes of trajectories, it can be applied in areas such as congestion predicting and road-system planning.

The notations used in the TC_MFTSM algorithm are listed in Table 2.

There are mainly three steps in Algorithm 3. First, trajectory outliers are removed from TS according to Definition 7 in order to reduce their negative impact on clustering results (Line 1). Lines 2-4 initialize the related variables. Second, we calculate the time duration of each trajectory in TS (Lines 5-7), which are used for the selection of initial trajectory centers. As a result, the top num_c trajectories with the longest time duration are selected to initialize $TCenters$ (Lines 8-9). Third, the trajectory clustering is completed based on the iteration process (Lines 10-33). Specifically, Lines 13-23 determine if each trajectory can be classified into a cluster or be discarded. Lines 24-31 calculate the new trajectory centers based on the last clustering result. The parameter MR is updated in Line 32. The clustering process is terminated when the end condition of the iteration process is met (Line 33). The output of Algorithm 3 is a set of trajectory clusters $TClusters$ and a set of centers $TCenters$ (Line 34).

The pseudo code of the TC_MFTSM algorithm is given as follows:

Each iteration of Algorithm 3 contains three main steps. 1) The determination of trajectory centers. 2) The classification of each trajectory. For each trajectory T_i , we find the central trajectory closest to T_i and determine if the distance between them is less than the threshold MR . If so, T_i is added into the cluster where the central trajectory is located. 3) The updation of the parameter MR .

Table 2 Notations used in the TC_MFTSM algorithm

Notation	Description
MR	The threshold of distance radius within any trajectory cluster
MD	The threshold of the number of discarded trajectories
MI	The threshold of iteration times
$TClusters$	A set consisting of trajectory clusters
$TCenters$	A set consisting of centers of trajectory clusters
$TDiscards$	A set consisting of discarded trajectories
$tra_tp(i)$	The time period occupied by the i -th trajectory
$iter_ct$	The counter of iteration times
cid	The cluster id
num_c	The number of trajectory clusters

Algorithm 3 TC_MFTSM: Trajectory clustering based on MFTSM

Input:

$TS = \{T_1, T_2, \dots, T_p\}$ (a dataset of p trajectories), k (the presupposed size of each trajectory cluster), η_o (the weight of trajectory orientation distance), η_s (the weight of trajectory speed distance), MR , MD , MI

Output:

$TClusters, TCenters$

```

1: Remove the trajectory outliers from  $TS$  according to Definition 7;
2: Initialize the parameters  $MR$ ,  $MD$  and  $MI$ ;
3:  $tra\_number \leftarrow |TS|$ ;  $iter\_ct \leftarrow 0$ ; //  $tra\_number$  is equal to  $p$ 
4:  $num_c \leftarrow \lfloor \frac{tra\_number}{k} \rfloor$ ;
5: for  $i \leftarrow 1$  to  $tra\_number$  do
6:    $tra\_tp(i) \leftarrow \mathbf{TIME}(i,3) - \mathbf{TIME}(i,2)$ ; //  $\mathbf{TIME}$  is available after executing Algorithm 1
7: end for
8: Sort the vector  $tra\_tp$  in descending order;
9:  $TCenters \leftarrow$  the top  $num_c$  trajectory id; // The initial centers dataset
10: repeat
11:    $iter\_ct \leftarrow iter\_ct + 1$ ;
12:    $TClusters \leftarrow \phi$ ,  $TDiscards \leftarrow \phi$ ; // Initialization
13: for  $i \leftarrow 1$  to  $tra\_number$  do
14:   if  $i \notin TCenters$  then
15:      $cid \leftarrow \arg \min_{j \in [1, num_c]} dist(TCenters(j), T_i)$ ; //using Algorithm 2
16:      $dmin \leftarrow dist(TCenters(cid), T_i)$ ; //using Algorithm 2
17:     if  $dmin < MR$  then
18:        $TClusters\{cid\} \leftarrow TClusters\{cid\} \cup \{i\}$ ;
19:     else
20:        $Tdiscards \leftarrow Tdiscards \cup \{i\}$ ;
21:     end if
22:   end if
23: end for
24:  $oldTCenters \leftarrow TCenters$ ;
25:  $TCenters \leftarrow \phi$ ;
26: for  $z \leftarrow 1$  to  $num_c$  do
27:   Calculate the average trajectory  $Atr_z$  within the  $z$ -th cluster;
28:    $lenclz \leftarrow length(TClusters\{z\})$ ;
29:    $Tpid \leftarrow \arg \min_{j \in [1, lenclz]} dist(TClusters(z, j), Atr_z)$ ; //using Algorithm 2
30:    $TCenters \leftarrow TCenters \cup \{Tpid\}$ ;
31: end for
32: Reassign  $MR$ ;
33: until ( $oldTCenters == TCenters$ ) or ( $iter\_ct == MI$ ) or ( $|Tdiscards| \leq MD$ );
34: return  $TClusters, TCenters$ ;

```

Note that the improved k -means clustering method adopted here is based on relatively uniform division. The k value is adjusted in the experiments to verify the effect of TC_MFTSM algorithm. It is different from density-based clustering methods, such as the Quality Threshold (QT) algorithm mentioned in Section 1. QT clustering algorithm depends on the value of diameter threshold, which is mainly set according to the observations and empirical values. And the time complexity of QT clustering algorithm is relatively high, because only one cluster can be obtained after one iteration.

The reason that we use an improved k -means clustering method lies in two aspects. On the one hand, k -means is one of the most famous and most popular clustering methods. It is available in many applications. Moreover, k -means method belongs to the category of partitioning methods, which are the simplest and most basic version of clustering analysis. On the other hand, our proposed algorithm TC_MFTSM is mainly compared with the existing algorithms MDAV [28] and GC_DM [29], where the idea of k -means method is used in trajectory clustering. In order to obtain better comparison results on these three algorithms, we implement the trajectory clustering based on the improved k -means algorithm. In fact, we have improved trajectory k -means clustering method in terms of the selection of initial central trajectories, the selection of k value and the updation of trajectory centers.

In addition, the trajectory outliers as defined in Definition 7 are removed in the first step of Algorithm 3. The implementation of this step has two advantages: one is to reduce the running time, and the other is to reduce the interference of outlying trajectories on clustering results. Table 3 shows the execution time comparison of two schemes run on the synthetic dataset, which will be introduced in Section 7.1. Scheme 1 refers to the scheme implemented according to Algorithm 3, that is the TC_MFTSM algorithm. Scheme 2 does not perform the step of removing trajectory outliers in Algorithm 3, and the other steps in Scheme 2 are the same as those in Scheme 1.

As can be seen from Table 3, in the vast majority of cases, the execution time of Scheme 1 is less than that of Scheme 2 with the same k . It shows that the removal of trajectory outliers can effectively improve the efficiency of the algorithm. The accuracy verification can be seen from Section 7.

Table 3 Execution time comparison of two schemes run on the synthetic dataset [sec]

k	20	40	60	80	100	120	140
Scheme 1	31.84	64.51	18.31	10.82	28.59	7.92	0.79
Scheme 2	138.52	112.25	62.10	33.13	19.80	17.47	15.58

The time complexity of Algorithm 3 depends on the following four aspects:

(a) Line 1: the time for removing the trajectory outliers from TS , whose time complexity is $O(p)$, where $p = |TS|$. (b) Lines 5-7: the time for calculating the vector tra_tp based on **TIME**, the time complexity is $O(p)$. (c) Line 8: it is used to sort the vector tra_tp , whose time complexity is $O(p \log p)$. (d) Lines 10-33: First, from Lines 13-23 in Algorithm 3, we know that an iterative process of clustering must be executed $num_c \cdot p$ times. Hence, the time complexity of this sub-part is $O(num_c \cdot p)$. Second, from Lines 26-31, we know that the function of this sub-part is to calculate the new $Tcenters$, whose time complexity is $O(num_c \cdot lenclz)$. Therefore, the time complexity of the two sub-parts is $O(num_c \cdot p) + O(num_c \cdot lenclz)$. Because $lenclz \doteq p/num_c$, $p \geq lenclz$, the executive times do not exceed $2 \cdot num_c \cdot p$. The time complexity of Lines 13-31 is $O(num_c \cdot p)$. Further, Lines 11-32 are repeated $max(MI, MD)$ times according to the repeat conditions (as shown in Line 33). In general, $max(MI, MD) \ll p$, so the maximum time complexity of this part (Lines 10-33) is $O(num_c \cdot p^2)$.

The time complexity of the other lines is $O(1)$. So the total approximate time complexity of Algorithm 3 is $O(p) + O(p) + O(p \log p) + O(num_c \cdot p^2)$. In conclusion, its overall time complexity is $O(num_c \cdot p^2)$.

The space complexity of Algorithm 3 is $O(p^2)$, which is mainly due to storing the distance matrix. TC_MFTSM also needs space to store $TClusters$ and $Tcenters$, but the space complexity of TC_MFTSM does not exceed $O(p^2)$ because the space required by them are respectively $num_c \cdot lenclz$ and num_c . Both num_c (the number of trajectory clusters) and $lenclz$ (the number of trajectories within each cluster) are much smaller than p . So the space complexity of Algorithm 3 is $O(p^2)$.

7 Experiments

We perform experiments to evaluate the accuracy and efficiency of the proposed algorithm. The experiments are conducted with Matlab 8.3 (64-bit) on a PC with Intel (R) Core (TM) 2 Duo CPU 2.6 GHz and 8 GB of RAM. The operating system is Microsoft Windows 7.

Three datasets (Section 7.1) are used to verify the performance of TC_MFTSM algorithm. On the one hand, we compare TC_MFTSM with the prior work of GC_DM [29] and MDAV [28]. Trajectory clustering methods adopted in these two algorithms are used as benchmarks for evaluating the relative performance of TC_MFTSM in all of the following experiments. On the other hand, in Section 7.3.3, we also conduct several experiments for comparing TC_MFTSM with a new algorithm using the

distance calculation method proposed in Literature [34], where the trajectory clustering method is improved based on Traclus algorithm [12]. For ease of description, the new algorithm is denoted as IMDTraclus later.

The four algorithms have certain comparability. They use different trajectory similarity measurement algorithms, but similar trajectory clustering methods, which are all based on greedy clustering. The proposed algorithm adopts a new trajectory similarity measurement method and makes some improvements in the trajectory clustering process. The parameters allocation is presented in Section 7.3.

According to Algorithms 1, 2 and 3, for each dataset, our experimental process is specifically arranged as follows:

- 1) Pre-processing the experimental dataset based on our proposed trajectory model.
- 2) Calculating the matrixes **PT** and **TIME** by implementing Algorithm 1, and the distance matrix based on Algorithm 2.
- 3) Implementing the distance calculation function between any two trajectories according to Algorithm 2.
- 4) Completing the trajectory clustering using Algorithm 3.
- 5) Achieving the experimental results comparison based on various evaluation metrics.

7.1 Dataset

1) Synthetic dataset

The Brinkhoff generator [36] is used to generate 1000 synthetic trajectories, containing 46906 locations in the Oldenburg city of German. The parameters used are as follows: the number of timestamps is 100, the number of moving objects classes is 6, the number of external objects classes is 1, the number of moving objects generated per timestamp is 10, the number of external objects generated per timestamp is 1, the speed of the moving objects is 250, the value of “report probability (0-1000)” is 1000, which means that a moving object is reported at every timestamp during its moving. There are at most 100 locations in each trajectory, and 45.5 locations per trajectory on average.

2) Real-life dataset

We also use the dataset of taxi moving trajectories collected from San Francisco in the United States [37, 38] as the real-life dataset in our experiments. It contains GPS coordinates of approximately 500 taxis collected in the San Francisco Bay Area during May 2008. The locations in this dataset are very fine-grained because the average time interval between two consecutive locations is less than 10 seconds [38]. The format of each mobility trajectory file is as follows. Each line contains latitude, longitude, occupancy and time, where the occupancy is ignored in our experiments. Since the trajectory of a cab during an entire

month can hardly be considered a single trajectory, we use the method in the literature [28] to pre-process this dataset. In particular, the trajectory data of the day between May 25 at 12:04 and May 26 at 12:04 is extracted because during this period there was the highest concentration of locations in the dataset [28]. After a trajectory filtering and location interpolation process, we obtained 480 trajectories and 244 locations per trajectory on average.

3) Clustered reference dataset

In order to calculate the (Adjusted) Rand Index value (will be introduced in Section 7.2), a trajectory dataset with reference classes is required. Therefore, we generate a dataset of clustered synthetic trajectories using the publicly available trajectory generator program written by Piciarelli¹.

A set of 1000 normal trajectories from 10 different trajectory clusters and another set of 50 abnormal trajectories from 50 different clusters are created with the randomness parameter set to the default value 0.7, which is set referring to Literature [39]. After being added time dimension, the dataset contains 1050 random three-dimensional trajectories of length 10. This dataset is used as one reference dataset in our experiments. Moreover, another two datasets derived from it will also be used (Section 7.3.3).

7.2 Evaluation metrics

The evaluation metrics in our experiments are described in this section.

1) The intra-cluster and inter-cluster distances

Four types of distances are calculated to evaluate the clustering effect. Their calculation methods are provided in Table 4, where $indave_{c_i}$ and $indmax_{c_i}$ respectively refer to the average distance and the maximum distance between central trajectory and other trajectories inside the i -th cluster, $outdave_{c_i}$ and $outdmin_{c_i}$ respectively refer to the average distance and the minimum distance between the central trajectory of the i -th cluster and trajectories in other clusters.

The smaller the intra-cluster distance is, or the greater the inter-cluster distance is, the higher the similarity degree within each cluster is.

2) The coupling degree

The coupling degree, denoted as cp_{dg} , can be used to reflect the degree of closeness within a cluster. This metric is based on the distances presented in Table 4. The smaller

¹http://avires.dimi.uniud.it/papers/trclust/create_ts2.m

Table 4 Four types of distances

Type of distance	Notation	Equation
The average intra-cluster distance	<i>indave</i>	$indave = \frac{1}{num_c} \sum_{i=1}^{num_c} indave_{.c_i}$
The maximum intra-cluster distance	<i>indmax</i>	$indmax = \frac{1}{num_c} \sum_{i=1}^{num_c} indmax_{.c_i}$
The average inter-cluster distance	<i>outdave</i>	$outdave = \frac{1}{num_c} \sum_{i=1}^{num_c} outdave_{.c_i}$
The minimum inter-cluster distance	<i>outdmin</i>	$outdmin = \frac{1}{num_c} \sum_{i=1}^{num_c} outdmin_{.c_i}$

the coupling degree is, the better the clustering effect is. The coupling degree can be calculated as follows:

$$cp_dg = \frac{1}{num_c} \sum_{i=1}^{num_c} \frac{indave_{.c_i}}{outdave_{.c_i}} \tag{20}$$

3) The Total Sum of Squared Error

We use the Sum of Squared Error (SSE) [40] to measure the clustering quality. The total SSE, denoted as *TSSE*, can be calculated as follows:

$$TSSE = \sum_{i=1}^{num_c} \left(\frac{1}{2 \cdot |C_i|} \cdot \sum_{T_x \in C_i} \sum_{T_y \in C_i} dist(T_x, T_y)^2 \right) \tag{21}$$

where C_i represents the i -th trajectory cluster, and $dist(T_x, T_y)$ is the distance between trajectories T_x and T_y . The smaller the *TSSE* is, the better the clustering effect is.

4) The Silhouette Index

The silhouette value is here used as a measure of how similar a trajectory T_x is to its own cluster C_i (cohesion) compared to other clusters (separation), denoted as $S(T_x)$:

$$S(T_x) = \frac{b(T_x) - a(T_x)}{\max\{a(T_x), b(T_x)\}} \tag{22}$$

where $a(T_x)$ is the average dissimilarity of T_x to all T_y ($T_x, T_y \in C_i, T_y \neq T_x$), $b(T_x)$ is the minimum dissimilarity over all clusters C_j ($j \neq i$), of the average dissimilarities to $T_y \in C_j$ [41]. The calculation equations are:

$$a(T_x) = \frac{1}{|C_i| - 1} \sum_{T_x, T_y \in C_i, T_y \neq T_x} dist(T_x, T_y) \tag{23}$$

$$b(T_x) = \min_{C_i, C_j \in TClusters, j \neq i} \left\{ \frac{1}{|C_j|} \sum_{T_y \in C_j} dist(T_x, T_y) \right\} \tag{24}$$

We can then quantify the validity of the trajectory clustering by the Silhouette Index (*SI*), defined as follows:

$$SI = \frac{1}{num_c} \sum_{i=1}^{num_c} \left\{ \frac{1}{|C_i|} \sum_{T_x \in C_i} S(T_x) \right\} \tag{25}$$

The silhouette value $S(T_x)$ ranges from -1 to 1, where a high value indicates that the trajectory T_x is well matched to its own cluster and poorly matched to neighboring clusters.

If most trajectories have high values, then the trajectory clustering result is appropriate.

5) The Rand Index and Adjusted Rand Index

Given a set of n elements $S = \{o_1, \dots, o_n\}$ and two partitions of S to compare, $X = \{X_1, \dots, X_r\}$, a partition of S into r subsets, and $Y = \{Y_1, \dots, Y_s\}$, a partition of S into s subsets.

First, we evaluate the performance of the proposed trajectory clustering algorithm using Rand Index (*RI*) [42]. The *RI* value is calculated as follows [43]:

$$RI(X, Y) = \frac{a + b}{a + b + c + d} = \frac{a + b}{\binom{n}{2}} \tag{26}$$

where each element is a trajectory object, a represents the number of pairs of elements in S that are in the same subset in X and in the same subset in Y , b represents the number of pairs of elements in S that are in the different subsets in X and in the different subsets in Y , c represents the number of pairs of elements in S that are in the same subset in X and in the different subsets in Y , d represents the number of pairs of elements in S that are in the different subsets in X and in the same subset in Y .

The Rand Index has a value between 0 and 1, with 0 indicating that the two clusterings do not agree on any pair of objects and 1 indicating that the two clusterings are identical. The larger the *RI* value the better is the clustering.

Second, the Adjusted Rand Index (*ARI*) [42, 43] is also used in measuring the quality of trajectory clustering. *ARI* is the corrected-for-chance version of the Rand Index. The overlap between X and Y can be summarized in a contingency table $[n_{ij}]$ where each entry n_{ij} ($i = 1, \dots, r; j = 1, \dots, s$) refers to the number of objects in common between X_i and Y_j , that is, $n_{ij} = |X_i \cap Y_j|$. No more expatiation about the contingency table here as it can be obtained from Literature [43].

The *ARI* value is calculated as follows:

$$ARI(X, Y) = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}{\frac{1}{2} [\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}} \tag{27}$$

where $a_i = \sum_{j=1}^s n_{ij}$, $b_j = \sum_{i=1}^r n_{ij}$. In essence, (27) calculates the value of $\frac{Index-ExpectedIndex}{MaxIndex-ExpectedIndex}$. The *ARI* value ranges from -1 to 1 and takes on the value 0 when the

Index is equal to its expected value (under the generalized hypergeometric distribution assumption for randomness). The greater the *ARI* value is, the higher the clustering quality is.

7.3 Experimental results

Experimental results and the related analysis are given in this section, in order to verify the accuracy and efficiency of the proposed trajectory clustering method based on multi-feature distance measurement. Specifically, we have conducted a set of experiments in which we executed TC_MFTSM, GC_DM, MDAV and IMDTraclus algorithms on the synthetic and real datasets.

The parameters used in our algorithms include $\eta_o, \eta_s, k, MR, MD, MI$.

MD and *MI* respectively represent the threshold of the number of discarded trajectories and the threshold of iteration times. *MR* represents the threshold of distance radius within any trajectory cluster. The value of *MD* depends on the size of a dataset. As a rule of thumb, we choose *MD* so that the number of discarded trajectories is between 2% and 5% of the total number of trajectories.

The three datasets used in the experiments are respectively with size of 1000, 480 and 1050, so it is appropriate to set *MD* as 20. It is not necessary to assign a large value to the parameter *MI* for multiple iterations, because the iteration terminates when one of the three conditions (Line 33 of Algorithm 3) is met. It is enough to set the number of iterations to 100. According to the experimental methods in Refs. [28, 29], the initial value of *MR* is set based on the map size and is increased by 5% for each iteration.

Therefore, the parameters are set as follows: *MD* = 20, *MI* = 100, η_o and η_s are equal to 0.3 to give the same importance to the orientation and speed distance. The selection of *k* values is presented in Sections 7.3.1 and 7.3.2.

7.3.1 Synthetic dataset

1) Visual display of the clustering results

To intuitively demonstrate the effect of clustering, without loss of generality, we set the parameter $k = 100$, and accordingly, the number of clusters is 9 in each algorithm. Figures 5, 6 and 7 respectively illustrate the clustering results of TC_MFTSM, GC_DM and MDAV on the synthetic dataset.

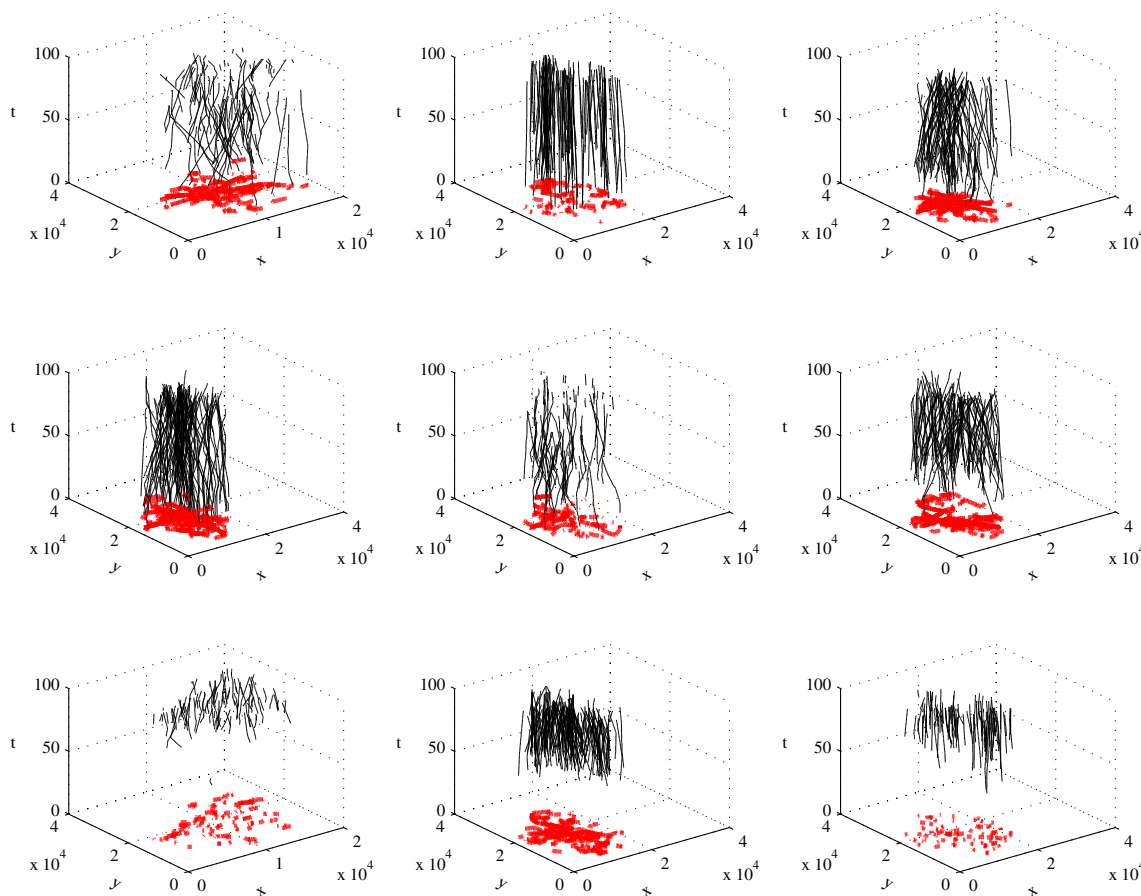


Fig. 5 Clustering results of TC_MFTSM algorithm on the synthetic dataset

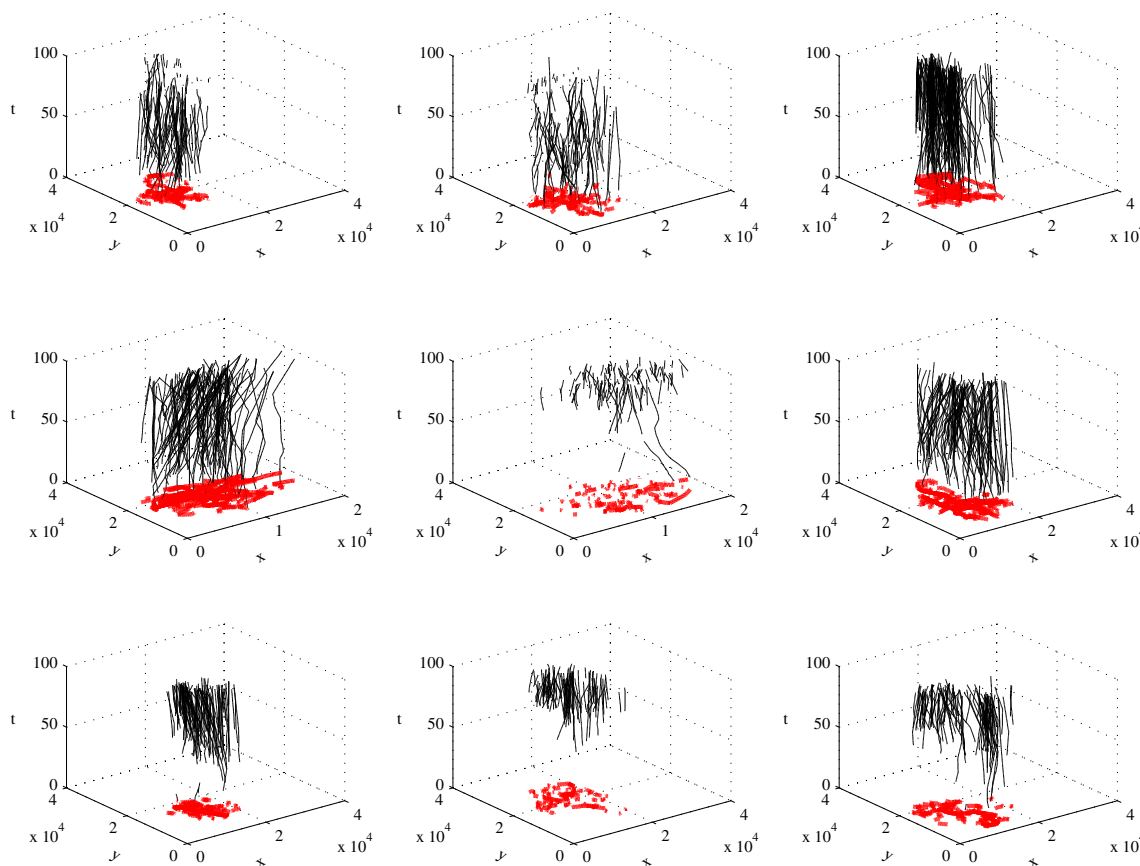


Fig. 6 Clustering results of GC_DM algorithm on the synthetic dataset

The trajectories in 3-dimensional space are depicted with black lines, and their corresponding projections in the 2D-plane are marked in red.

In Figs. 5–7, each sub-figure represents a cluster. In each figure, there are 9 clusters that are respectively shown in 9 sub-figures. In Fig. 5, all the clusters are highly gathered except the first one. The aggregation effect in Fig. 5 is superior to the other two figures in terms of time dimension, spatial dimension and trajectory direction. For example, in the second sub-figure (Row 1, Col 2) of Fig. 5, all the trajectories in this cluster have highly similar time intervals, centralized geographical locations, and consistent directions. Other sub-figures in Fig. 5 are the same.

From the results of the latter two algorithms, there are obvious trajectory outliers in some clusters. For example, in the fifth sub-figure (Row 2, Col 2) of Fig. 6 and in the seventh sub-figure (Row 3, Col 1) of Fig. 7, the trajectories in the corresponding cluster are very different with each other regardless of the time, location or direction. The same goes for the other clusters in Figs. 6–7. The reason is that some characteristics are ignored in measuring trajectory similarity. As is seen from Figs. 5–7, the clustering effect of the TC_MFTSM algorithm outperforms that of the other two algorithms GC_DM and MDAV.

In the following experiments, we vary the value k between 20 and 140.

2) Execution time

Execution time is the time it takes an algorithm to complete trajectory clustering. The Execution time comparison of 3 algorithms executed on the synthetic dataset is shown in Fig. 8.

As can be seen from Fig. 8, in the vast majority of cases, the execution time of TC_MFTSM algorithm is less than that of the other two algorithms GC_DM and MDAV with the same k . The reason is that in our algorithm the trajectory outliers have been removed from the dataset before clustering. The general trend is, the larger the value of k , the fewer the number of clusters, and the faster the clustering speed.

3) The intra-cluster and inter-cluster distances

According to the calculation methods described in Table 4, the four types of distances in the experimental results on the synthetic dataset are provided in Table 5 and Fig. 9.

Table 5 and Fig. 9 show that the average intra-cluster distance (*indave*) of three algorithms is far less than the average inter-cluster distance (*outdave*), which indicates

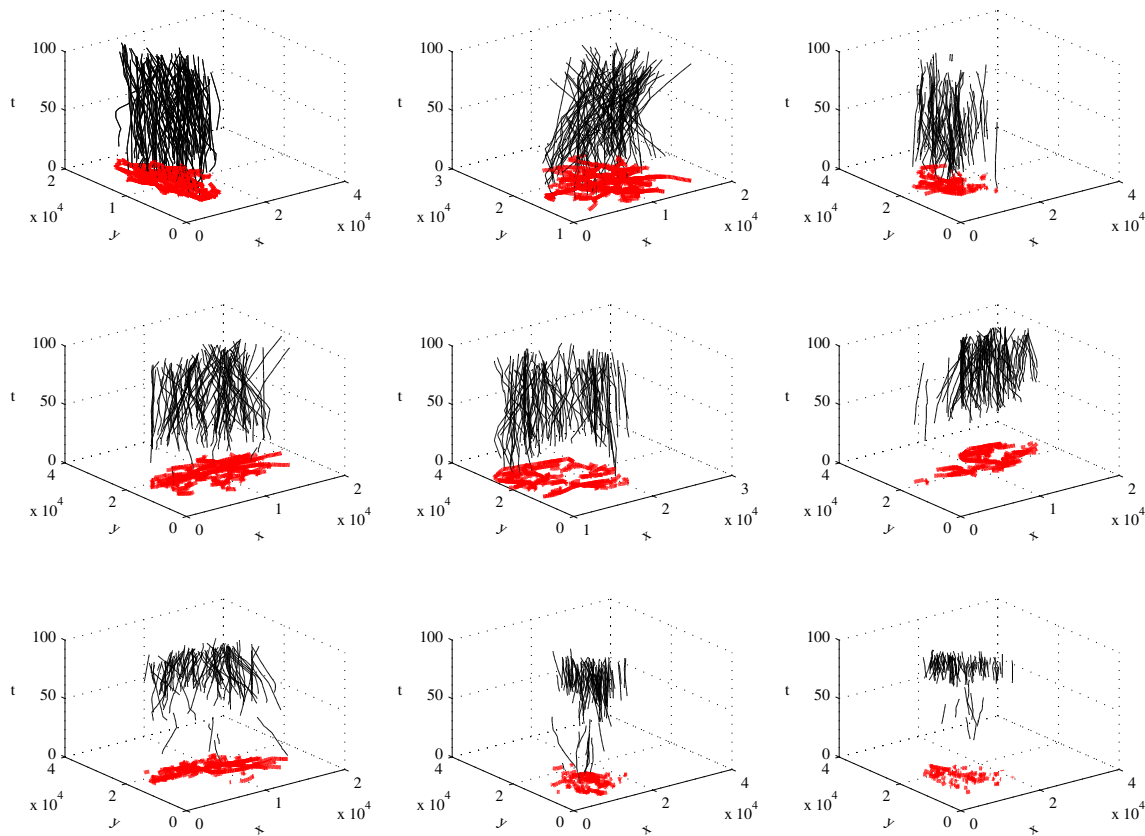


Fig. 7 Clustering results of MDAV algorithm on the synthetic dataset

the good clustering effect. And all the four distances of TC_MFTSM algorithm are less than that of the other two algorithms. This depends on the different distance measure metric. In addition, for the three algorithms at the same k , it is normal that the minimum inter-cluster distance ($outdmin$) is smaller than the maximum intra-cluster distance

($indmax$) because the distance between some trajectories and trajectory centers in other clusters is less than the maximum distance radius in those clusters.

4) The coupling degree

Based on (20), the coupling degree comparison of 3 algorithms run on the synthetic dataset is shown in Fig. 10. The coupling degree (cp_dg) of TC_MFTSM is substantially less than that of the algorithms GC_DM and MDAV at the same k in the interval [60,140]. For the dataset containing more than 1000 trajectories, it is proper to set the presupposed size of each trajectory cluster (k) within this interval. As described in Section 7.2, the coupling degree represents the degree of closeness within a cluster. Therefore, Fig. 10 shows that our algorithm outperforms the existing algorithms.

5) The Total Sum of Squared Error

Based on (21), we calculate $TSSE$ values of the three algorithms on the synthetic dataset. Table 6 shows the comparison results. Obviously, the $TSSE$ value of our algorithm is smaller than that of the other two algorithms at the same k .

6) The Silhouette Index

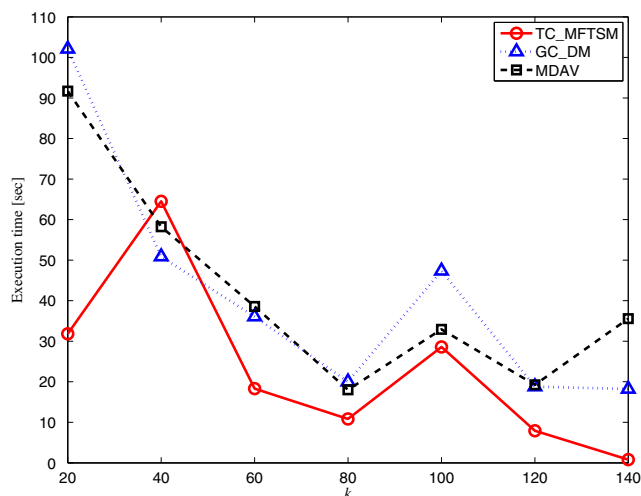


Fig. 8 Execution time comparison of 3 algorithms run on the synthetic dataset

Table 5 The distances comparison of clustering results of 3 algorithms on the synthetic dataset

Type of value	Algorithm	k						
		20	40	60	80	100	120	140
The average intra-cluster distance	TC_MFTSM	3.85	4.34	5.77	5.33	6.14	6.34	7.48
	GC_DM	15.7	20.7	21.5	19.7	19.4	20.0	22.6
	MDAV	17.4	18.1	17.6	20.4	17.2	21.7	25.8
The maximum intra-cluster distance	TC_MFTSM	6.67	8.39	11.9	12.6	14.8	18.0	18.5
	GC_DM	20.4	28.0	33.7	36.6	35.9	44.4	33.3
	MDAV	24.1	27.3	29.2	49.8	42.3	41.6	49.8
The average inter-cluster distance	TC_MFTSM	90.2	136	183	206	251	197	205
	GC_DM	203	315	295	325	267	305	359
	MDAV	404	511	478	759	607	679	990
The minimum inter-cluster distance	TC_MFTSM	2.98	2.88	3.41	3.28	3.19	2.78	3.80
	GC_DM	9.66	11.9	12.3	9.47	10.0	11.2	12.5
	MDAV	10.8	8.42	8.11	10.1	7.11	8.14	9.71

Based on (22)–(25), we calculate SI values of the three algorithms on the synthetic dataset. Table 7 shows the comparison results. As is described in Section 7.2, higher SI value indicates higher quality of trajectory clustering. Obviously, the SI value of our algorithm is higher than that of the other two algorithms at the same k .

7.3.2 Real-life dataset

Because the number of trajectories in the real-life dataset is 480, which is less than the number of trajectories in the

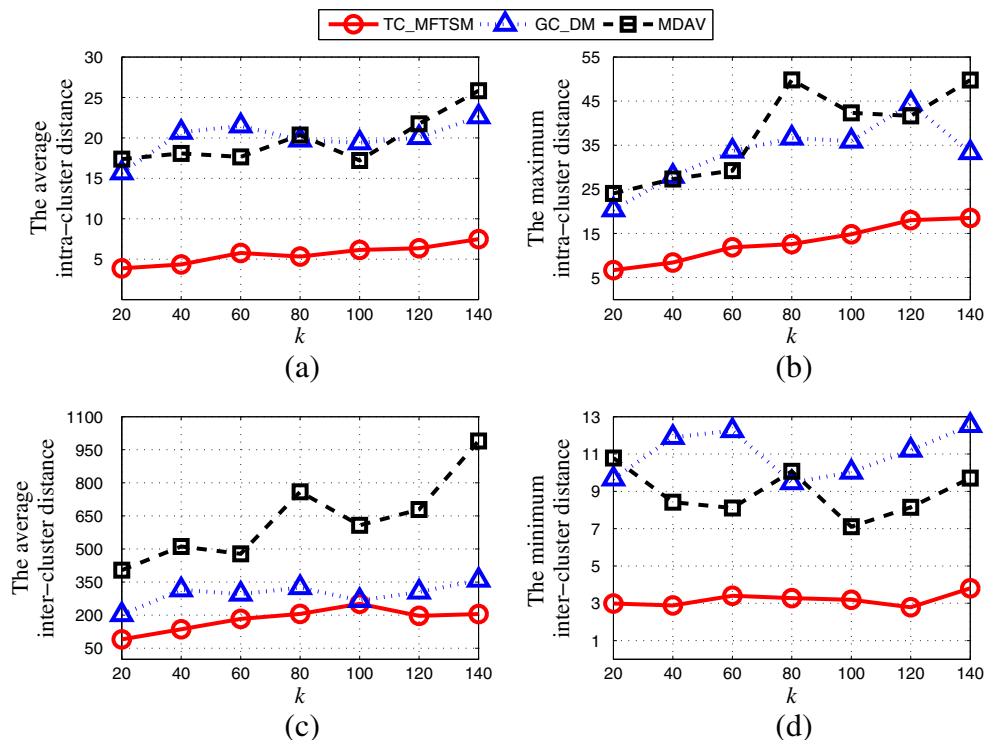
above synthetic dataset, in the following experiments, we change the value k , varying it between 10 and 100.

1) Execution time

The Execution time comparison of 3 algorithms run on the real-life dataset is shown in Fig. 11.

As can be seen from Fig. 11, in the vast majority of cases, the execution time of TC_MFTSM algorithm is less than that of the other two algorithms GC_DM and MDAV with the same k . And the time curve of TC_MFTSM algorithm is much smoother with different k . Because in our algorithm

Fig. 9 The four types of distances comparison of clustering results on the synthetic dataset



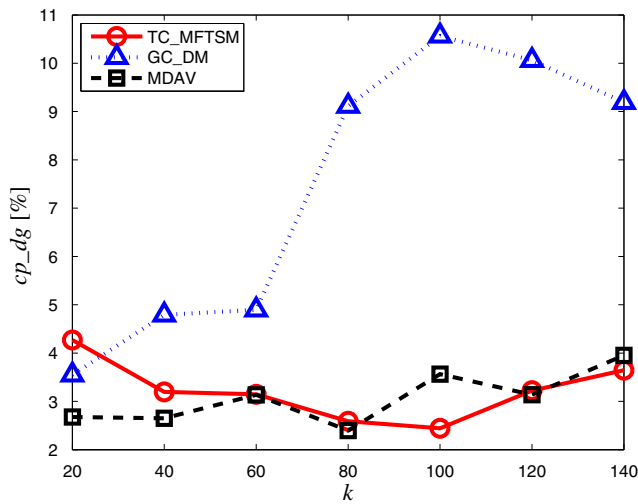


Fig. 10 Coupling degree comparison of 3 algorithms run on the synthetic dataset

the selection of initial trajectory centers is also optimized except for the outliers elimination.

2) The intra-cluster and inter-cluster distances

The four types of distances in the experimental results on the real-life dataset are provided in Table 8 and Fig. 12.

Table 8 and Fig. 12 show that, for this real-life cab dataset, all of the four distances are very small, and thus the difference among the 3 algorithms is not very significant. The results of our algorithm do not outperform the algorithm MDAV, but the difference is controlled in a small range. More features are considered in our algorithm, so the absolute values are bigger than the ones of MDAV.

3) The coupling degree

Based on (20), the coupling degree comparison of 3 algorithms run on the real-life dataset is shown in Fig. 13. The coupling degree of TC_MFTSM is substantially less than that of GC_DM algorithm and a little greater than that of MDAV algorithm at the same k in the interval [10,100]. For the dataset containing 480 trajectories, it is proper to set the presupposed size of each trajectory cluster (k) within this interval. As described in Section 7.2, the coupling

Table 6 The TSSE comparison of 3 algorithms on the synthetic dataset ($\times 10^7$)

Algorithm	k						
	20	40	60	80	100	120	140
TC_MFTSM	0.0211	0.086	0.039	0.152	0.055	0.231	0.125
GC_DM	0.589	1.009	1.387	0.993	2.064	1.331	1.411
MDAV	0.724	1.065	1.739	1.586	2.44	2.964	3.106

Table 7 The SI comparison of 3 algorithms on the synthetic dataset

Algorithm	k						
	20	40	60	80	100	120	140
TC_MFTSM	0.849	0.818	0.773	0.754	0.698	0.750	0.739
GC_DM	0.801	0.777	0.737	0.738	0.703	0.698	0.697
MDAV	0.755	0.656	0.672	0.668	0.636	0.631	0.652

degree represents the degree of closeness within a cluster. Therefore, Fig. 13 shows that our algorithm is effective in terms of this metric, it outperforms GC_DM algorithm and is similar to the result of MDAV algorithm. Moreover, the curve of our algorithm with different parameter is smoother than those of the others.

4) The Total Sum of Squared Error

Based on (21), we calculate TSSE values of the three algorithms on the real-life cab dataset. Table 9 shows the comparison results. Obviously, the TSSE value of our algorithm is far smaller than that of the other two algorithms at the same k . It represents the priority of our algorithm in effect and accuracy of clustering.

5) The Silhouette Index

Based on (22)–(25), we calculate SI values of the three algorithms on the real-life dataset. Table 10 shows the comparison results. The SI value of TC_MFTSM algorithm is obviously higher than that of GC_DM algorithm at the same k . Moreover, TC_MFTSM outperforms MDAV algorithm in the vast majority of cases.

In summary, based on most evaluation metrics, experimental results on the synthetic and real-life datasets show that the proposed TC_MFTSM algorithm outperforms the

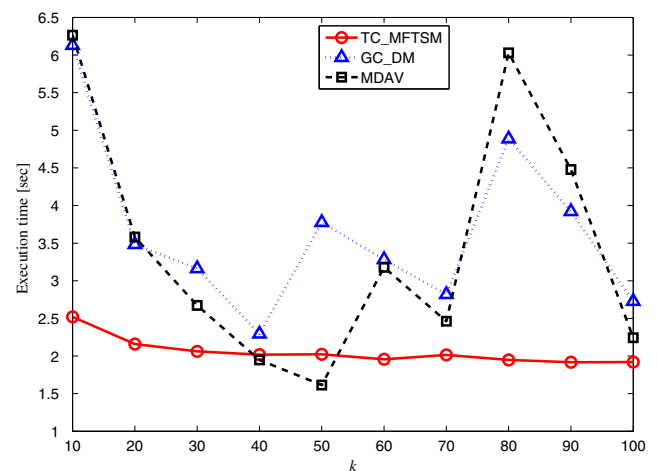


Fig. 11 Execution time comparison of 3 algorithms run on the real-life dataset

Table 8 The distances comparison of clustering results of 3 algorithms on the real-life dataset ($\times 10^{-2}$)

Type of value	Algorithm	k									
		10	20	30	40	50	60	70	80	90	100
The average intra-cluster distance	TC_MFTSM	0.264	0.303	0.303	0.286	0.264	0.280	0.284	0.291	0.293	0.300
	GC_DM	0.576	0.754	0.835	0.943	0.943	0.943	0.697	0.697	0.697	1.171
	MDAV	0.017	0.031	0.040	0.052	0.049	0.046	0.051	0.043	0.044	0.034
The maximum intra-cluster distance	TC_MFTSM	0.285	0.338	0.363	0.346	0.445	0.493	0.519	0.527	0.530	0.599
	GC_DM	1.449	1.290	1.571	2.056	2.056	2.056	2.516	2.516	2.516	2.553
	MDAV	0.018	0.034	0.045	0.064	0.058	0.082	0.069	0.082	0.082	0.100
The average inter-cluster distance	TC_MFTSM	0.425	0.534	0.444	0.417	0.340	0.371	0.383	0.395	0.401	0.429
	GC_DM	0.905	0.922	0.948	0.965	0.960	0.962	1.018	1.018	1.037	1.064
	MDAV	0.084	0.163	0.237	0.314	0.333	0.376	0.439	0.474	0.480	0.117
The minimum inter-cluster distance	TC_MFTSM	0.202	0.228	0.188	0.187	0.193	0.201	0.192	0.201	0.202	0.220
	GC_DM	0.396	0.401	0.406	0.402	0.375	0.365	0.371	0.371	0.353	0.362
	MDAV	0.014	0.022	0.030	0.035	0.041	0.039	0.045	0.040	0.040	0.030

existing algorithms GC_DM and MDAV in terms of execution efficiency and clustering utility, and can better reflect the multiple internal and external characteristics of the trajectory itself. Therefore, the TC_MFTSM algorithm is feasible and effective.

7.3.3 Clustered reference dataset

In this section, we evaluate the clustering performance of TC_MFTSM vs. GC_DM, MDAV and IMDTraclus based on the calculation of Rand Index (*RI*) and Adjusted Rand

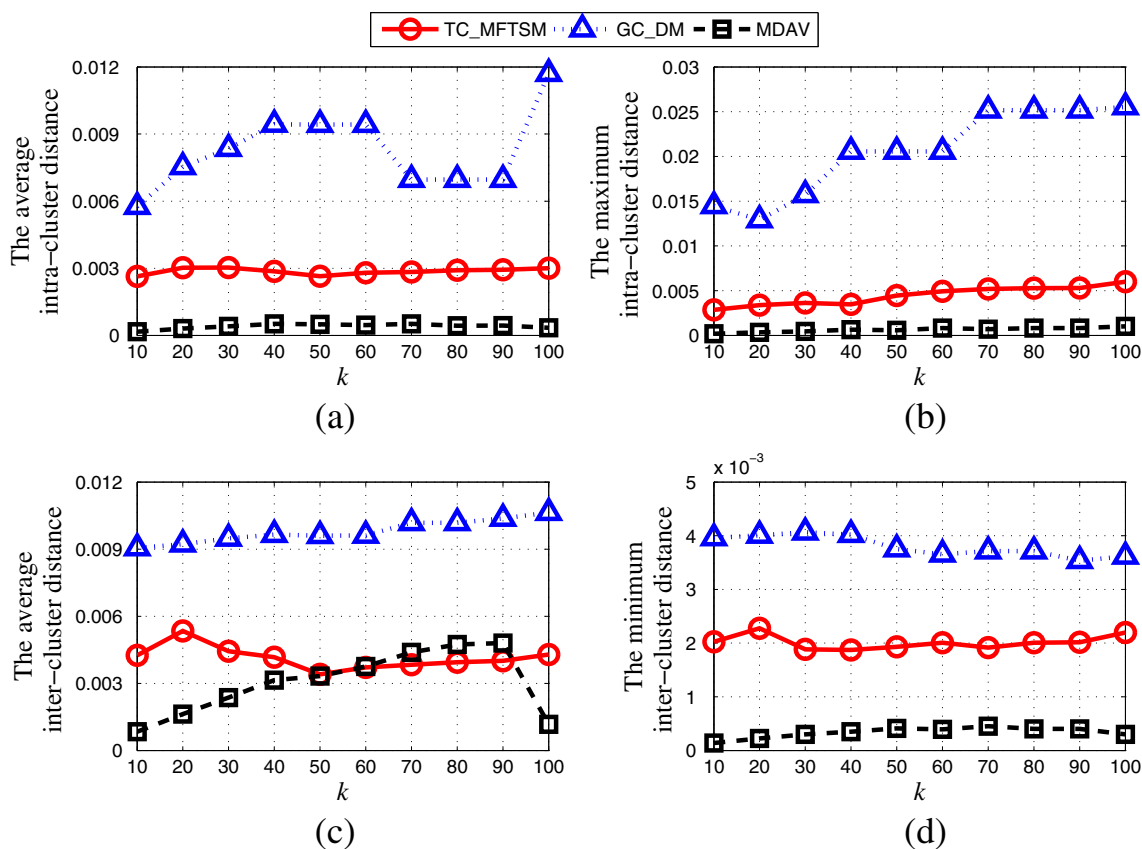


Fig. 12 The four types of distances comparison of clustering results on the real-life dataset

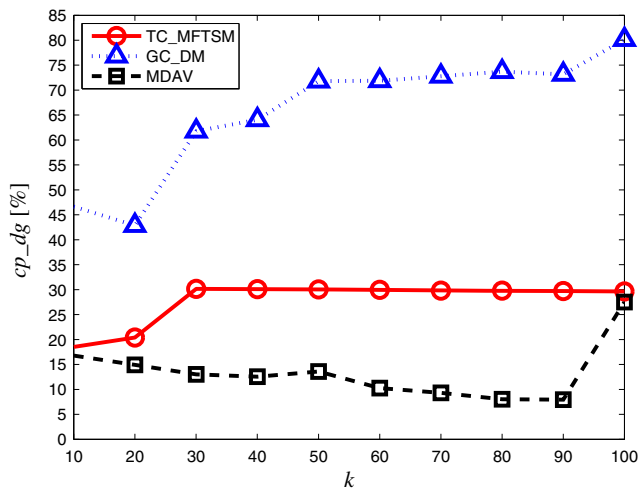


Fig. 13 Coupling degree comparison of 3 algorithms run on the real-life dataset

Index (*ARI*) values. These two metrics are very popular for testing the performance of clustering algorithms. The *RI* (*ARI*) measurement result is representative in measuring the effect of clustering, so we choose to add a set of experiments comparing the proposed algorithm TC_MFTSM with the new algorithm IMDTraclus to this section. To calculate *RI* and *ARI*, a clustered dataset is needed as a reference standard. Therefore, a clustered reference dataset (denoted as *ClusDS*) is generated by a trajectory generator, as described in Section 7.1.

We conduct a set of experiments based on the following 3 reference datasets, which are all derived from *ClusDS*:

- (a) *ClusDS* itself. That is, a generated dataset of all the normal and abnormal trajectories without any processing. There are 1000 normal trajectories and 50 abnormal trajectories, where the normal trajectories are grouped into 10 clusters and there are 100 trajectories in each cluster, all the abnormal trajectories are treated as 1 cluster. In order to unify the numbering of 3 datasets and to facilitate the later description, *ClusDS* is also denoted as *ClusDS1*. It is shown in Fig. 14. The 3-dimensional effects of these synthetic trajectories are shown in Fig. 14a, while their projections in the 2D-plane are shown in Fig. 14b. Note that

for clarity, different trajectory clusters are represented with different colors in Fig. 14. The trajectories marked with solid lines are labelled normal and the trajectories marked with blue dotted lines are labelled abnormal.

- (b) A dataset of all the normal trajectories without any locations deletion, denoted as *ClusDS2*, it is a subset of *ClusDS1*. There are 10 clusters and 100 trajectories in each cluster. As is shown in Fig. 14, the trajectories within *ClusDS2* are those marked with solid lines. That is, after the trajectories marked with dotted lines are excluded from Fig. 14, the dataset of remaining trajectories is *ClusDS2*.
- (c) A dataset of all the normal trajectories with random deletion of some locations, denoted as *ClusDS3*. It is shown in Fig. 15. The number of trajectories in *ClusDS3* is the same as the number of trajectories in *ClusDS2*, but the number of locations in *ClusDS3* is less than the number of locations in *ClusDS2*. For each trajectory, several locations have been randomly deleted. This is a dataset that is not synchronized in time. It can be used to better verify the performance of our algorithm because both temporal and spatial factors should be considered in trajectory distance measurement.

As can be seen from Figs. 14 and 15, the clustering results in trajectory datasets *ClusDS1-ClusDS3* are perfect for its high similarity within each cluster. It is appropriate that we use one of these datasets as a reference/standard to calculate the (Adjusted) Rand Index values. In order to make comparisons on the *RI* and *ARI* values with different parameters, in the following experiments, we change the value num_c (the number of trajectory clusters), varying it between 5 and 40.

The experiments in this section are conducted based on three labelled trajectory datasets with reference significance. In order to make the proposed algorithm more convincing, we add a new algorithm IMDTraclus to the following comparison experiments in addition to the above two algorithms GC_DM and MDAV. The reason for selecting IMDTraclus here is as follows. This algorithm is also implemented in two steps. First, trajectory similarity measurement method proposed in Literature [34] is adopted

Table 9 The *TSSE* comparison of 3 algorithms on the real-life dataset

Algorithm	k									
	10	20	30	40	50	60	70	80	90	100
TC_MFTSM	0.216	0.212	0.207	0.209	0.208	0.208	0.210	0.210	0.209	0.620
GC_DM	0.912	0.960	0.991	0.988	0.985	0.991	0.993	0.995	0.993	1.00
MDAV	0.909	0.956	0.973	0.979	0.983	0.990	0.993	0.991	0.992	0.999

Table 10 The SI comparison of 3 algorithms on the real-life dataset

Algorithm	k									
	10	20	30	40	50	60	70	80	90	100
TC_MFTSM	0.677	0.583	0.509	0.517	0.498	0.510	0.493	0.437	0.421	0.367
GC_DM	0.259	0.206	0.172	0.152	0.148	0.146	0.137	0.120	0.115	0.119
MDAV	0.586	0.513	0.547	0.452	0.494	0.505	0.499	0.403	0.486	0.362

to calculate the distance between two trajectories. Second, being similar with the other two algorithms MDAV and GC_DM, a greedy clustering method is used to complete the trajectory clustering process. All of the four algorithms are two-stage methods mentioned in Section 2. So it is reasonable to conduct the comparison among these algorithms.

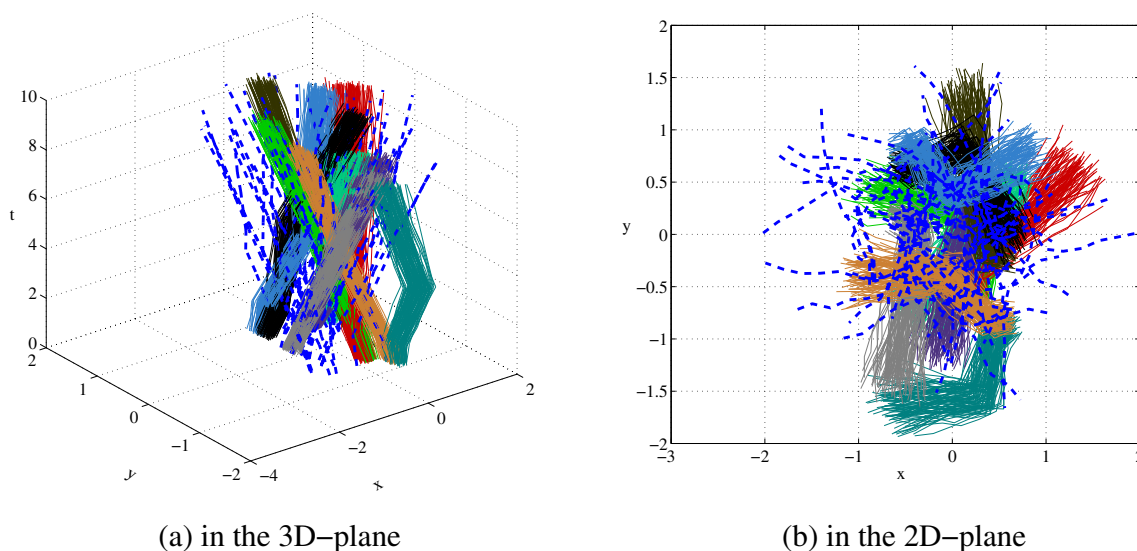
1) The Rand Index

Based on (26), we respectively calculate RI values of the four algorithms on datasets *ClusDS1-ClusDS3*. In calculation of $RI(X, Y)$, the clustering of a reference dataset is acted as X and the clustering result of an algorithm is acted as Y . For example, the clustering results of TC_MFTSM, GC_DM, MDAV and IMDTraclus are respectively acted as Y to calculate $RI(X, Y)$.

Figure 16 shows the comparison of the four algorithms on the RI value, where Fig. 16a and c respectively represent the running results on the trajectory datasets *ClusDS1-ClusDS3*.

As can be seen from Fig. 16, the results of TC_MFTSM algorithm are slightly better than that of the other three algorithms with different values of parameter num_c . With different values of num_c , the average RI values of TC_MFTSM, GC_DM, MDAV and IMDTraclus are respectively 0.935, 0.940, 0.939, 0.894 for *ClusDS1*. They are respectively

0.940, 0.938, 0.939, 0.888 for *ClusDS2*, and 0.900, 0.898, 0.891, 0.852 for *ClusDS3*. The RI values of TC_MFTSM algorithm are higher than that of IMDTraclus algorithm in all cases. Taken together, the TC_MFTSM algorithm has higher accuracy. One of the advantages of our algorithm is that multiple features are considered in trajectory distance measurement. For a dataset consisting of trajectories with standard length, the superiority of our algorithm is not very notable. But if a dataset consisting of trajectories with different lengths and shapes is used, it is clear that TC_MFTSM algorithm is better than the others. As described above, *ClusDS1* is a complete dataset including all the normal and abnormal trajectories. *ClusDS2* is a subset of *ClusDS1*, it includes all the normal trajectories. For each trajectory in *ClusDS1* or *ClusDS2*, there are 10 locations without any deletion. On the other hand, *ClusDS3* is a dataset consisting of trajectories with different lengths. As is shown in Fig. 16c, the TC_MFTSM algorithm outperforms the other algorithms in the vast majority of cases. The reason is that the initial trajectory centers are optimized in TC_MFTSM algorithm, which is applicable to irregular datasets. In real life, the trajectories collected from various devices are heterogeneous, so the TC_MFTSM algorithm is more applicable. In general, we can see the priority of our algorithm in accuracy of clustering from Fig. 16.

**Fig. 14** The reference dataset without locations deletion (*ClusDS1*)

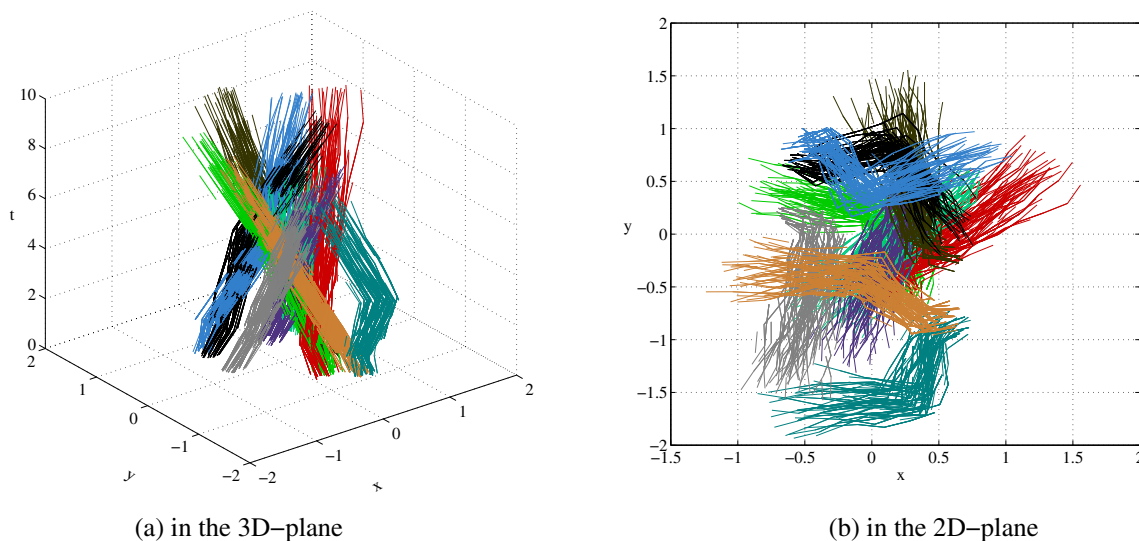


Fig. 15 The reference dataset with random deletion of some locations (*ClusDS3*)

2) The Adjusted Rand Index

Based on (27), we respectively calculate *ARI* values of the four algorithms on datasets *ClusDS1-ClusDS3*. In calculation of $ARI(X, Y)$, the clustering of a reference dataset is acted as *X* and the clustering result of an algorithm is acted as *Y*. For example, the clustering results of TC_MFTSM, GC_DM, MDAV and IMDTraclus are respectively acted as *Y* to calculate $ARI(X, Y)$.

Figure 17 shows the comparison of the four algorithms on the *ARI* value, where Fig. 17a and c respectively represent the running results on the trajectory datasets *ClusDS1-ClusDS3*.

As can be seen from Fig. 17, with different values of parameter num_c , the results of TC_MFTSM algorithm are better than that of GC_DM and MDAV algorithms in the vast majority of cases, and are better than that of IMDTraclus algorithm in all cases. Just as reviewed in Section 2, the

existing similarity measurement methods cannot fully utilize the specific features of trajectory itself when measuring the distance between trajectories. Another factor is that the selection method of initial trajectory centers is optimized in our algorithm. The comparison results indicate the effectiveness and superiority of the proposed algorithm.

Additionally, one can observe that the *ARI* values in Fig. 17c are relatively smaller than the *ARI* values in Fig. 17a and b. The reason is that, as described above, *ClusDS3* is a dataset of all the normal trajectories with random deletion of some locations. In the case of clustering results being randomly generated, the *ARI* value is close to zero.

Based on various evaluation metrics, the proposed algorithm has been verified on 3 different datasets mentioned in Section 7.1. According to the analysis of algorithm complexity and experimental results, TC_MFTSM algorithm is stable, reasonable and effective.

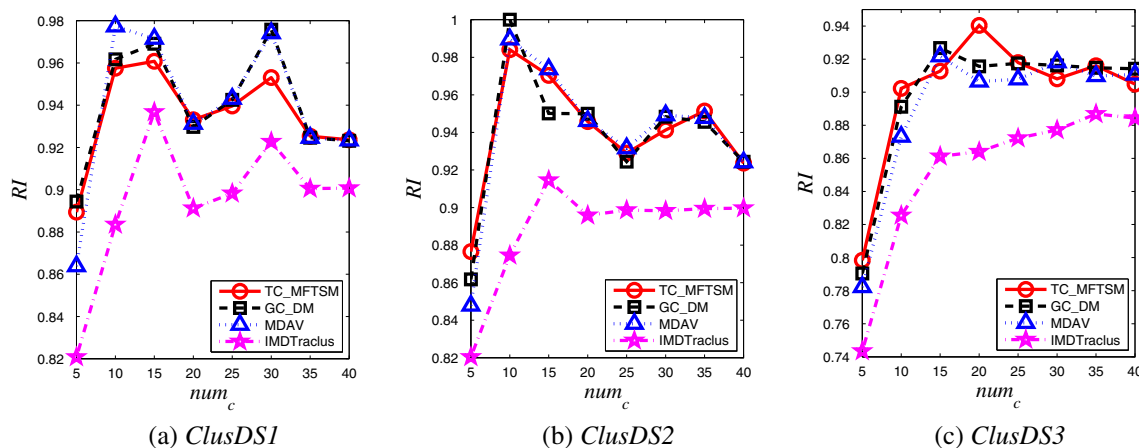


Fig. 16 The *RI* comparison of 4 algorithms on datasets *ClusDS1-ClusDS3*

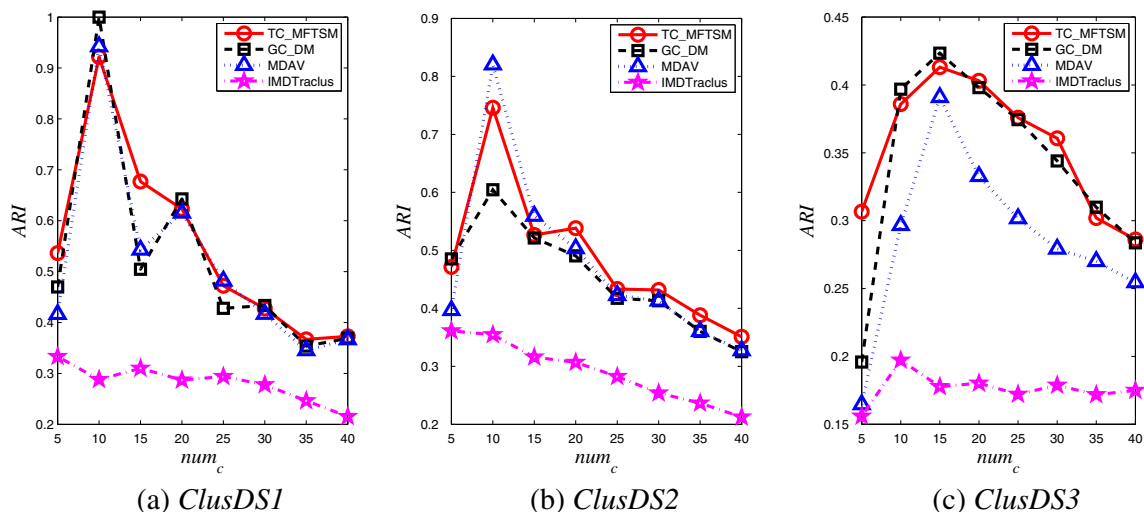


Fig. 17 The ARI comparison of 4 algorithms on datasets *ClusDS1*–*ClusDS3*

8 Conclusion

In this paper, we introduce the problem of trajectory similarity clustering and its applications. We propose a trajectory clustering algorithm based on multi-feature distance measurement. The proposed trajectory clustering algorithm TC_MFTSM uses the characteristics of orientation, speed, shape, location and continuity of each trajectory to achieve more accurate clustering results. In addition, the initial centers of trajectory clusters are optimized based on the time interval of each trajectory. Both the intuitive visualization presentation and the experimental results show that the proposed algorithm achieves high accuracy and efficiency. Compared to the GC_DM, MDAV and IMDTraclus algorithms, the TC_MFTSM algorithm reduces the negative impact of outlying trajectories when selecting the initial centers. Furthermore, it improves accuracy and stability in clustering results. The proposed algorithm can be used in many geographical research fields. In future research, we first plan to develop a trajectory privacy preserving method based on the proposed trajectory clustering algorithm to achieve trajectory k -anonymity within each cluster while maintaining higher utility of published trajectory data. Second, we will explore further applications for the TC_MFTSM method, such as tourist routes recommendation and congestion prediction. Third, in order to facilitate further research and application, we will develop a trajectory data analysis and visualization application system that integrates the functions of trajectory input, query, processing, analysis, output and display. The program that implements the proposed method will be used as a background part of the system.

Acknowledgements The authors would like to thank the reviewers for their useful comments and suggestions for this paper. This work

was supported by the National Natural Science Foundation of China (61702010, 61672039), the Key Program for University Top Talents of Anhui Province (gxbjZD2016011), the University Natural Science Research Program of Anhui Province (KJ2017A327), and the Science and Technology Project of Wuhu City (2016cxy04).

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

References

- Zheng Y, Xie X, Ma W-Y (2010) GeoLife: a collaborative social networking service among user, location and trajectory[J]. *IEEE Data Eng Bull* 33(2):32–39
- Hu W, Li X, Tian G et al (2013) An incremental DPMM-based method for trajectory clustering, modeling, and retrieval[J]. *IEEE Trans Pattern Anal Mach Intell* 35(5):1051–1065
- Luo W, Tan H, Chen L et al (2013) Finding time period-based most frequent path in big trajectory data[C]. In: *Proceedings of the ACM SIGMOD international conference on management of data*, pp 713–724
- Zheng Y. (2015) Trajectory data mining: An overview[J]. *ACM Trans Intell Syst Technol (TIST)* 6(3):29
- Lv M, Chen L, Xu Z et al (2016) The discovery of personally semantic places based on trajectory data mining[J]. *Neurocomputing* 173:1142–1153
- Zheng K, Zheng Y, Yuan NJ et al (2014) Online discovery of gathering patterns over trajectories[J]. *IEEE Trans Knowl Data Eng* 26(8):1974–1988
- Sanchez I, Aye ZMM, Rubinstein BIP et al (2016) Fast trajectory clustering using hashing methods[C]. In: *Proceedings of the IEEE international joint conference on neural networks (IJCNN)*, pp 3689–3696
- Han B, Liu L, Omiecinski E (2015) Road-network aware trajectory clustering: integrating locality, flow, and density[J]. *IEEE Trans Mob Comput* 14(2):416–429
- Zhao P, Qin K, Ye X et al (2016) A trajectory clustering approach based on decision graph and data field for detecting hotspots[J]. *Int J Geogr Inf Sci* 31(7):1–27
- Nikolaou TG, Kolokotsa DS, Stavrakakis GS et al (2012) On the application of clustering techniques for office buildings' energy

- and thermal comfort classification[J]. *IEEE Trans Smart Grid* 3(4):2196–2210
11. Yuan G, Sun P, Zhao J et al (2017) A review of moving object trajectory clustering algorithms[J]. *Artif Intell Rev* 47:123–144
 12. Lee J-G, Han J, Whang K-Y (2007) Trajectory clustering: A partition-and-group framework[C]. In: *Proceedings of the ACM SIGMOD international conference on management of data*, pp 593–604
 13. Pelekis N, Kopanakis I, Kotsifakos EE et al (2011) Clustering uncertain trajectories[J]. *Knowl Inf Syst* 28(1):117–147
 14. Besse PC, Guillouet B, Loubes JM et al (2017) Destination prediction by trajectory distribution based model[J]. *IEEE Transactions on Intelligent Transportation Systems*:1–12
 15. Lloyd SP (1982) Least squares quantization in PCM[J]. *IEEE Trans Inf Theory* 28(2):129–137
 16. Ester M, Kriegel HP, Sander J et al (1996) A density-Based algorithm for discovering clusters in large spatial databases with noise[C]. In: *Proceedings of the 2nd international conference on knowledge discovery and data mining*, pp 226–231
 17. Zhang T, Ramakrishnan R, Livny M. (1996) BIRCH An efficient data clustering databases method for very large databases[C]. *Proc ACM SIGMOD Int Conf Manag Data* 1:103–114
 18. Ankerst M, Breunig MM, Kriegel H-P et al (1999) Optics: ordering points to identify the clustering structure[C]. *ACM Sigmod Record* 28(2):49–60
 19. Pravirovic S, Appice A, Lanza A et al (2014) Wind power forecasting using time series cluster analysis[G]. In: *Discovery science*. Springer International Publishing, pp 276–287
 20. Laurinec P, LóDerer M, Vrablecová P et al (2017) Adaptive time series forecasting of energy consumption using optimized cluster analysis[C]. In: *Proceedings of the IEEE international conference on data mining workshops*, pp 398–405
 21. Wei J, Yu H, Chen JH et al (2011) Parallel clustering for visualizing large scientific line data[C]. In: *Proceedings of the 1st IEEE symposium on large-scale data analysis and visualization (LDAV)*, pp 47–55
 22. Atev S, Miller G, Papanikolopoulos NP (2010) Clustering of vehicle trajectories[J]. *IEEE Trans Intell Transp Syst* 11(3):647–657
 23. Choong M Y, Chin RK Y, Yeo KB et al (2014) Trajectory clustering for behavioral pattern learning in transportation surveillance[C]. In: *Proceedings of the 4th IEEE international conference on artificial intelligence with applications in engineering and technology (ICAJET)*, pp 199–203
 24. Ferreira N, Klosowski JT, Scheidegger CE et al (2013) Vector field k-means: clustering trajectories by fitting multiple vector fields[C]. *Eurographics Conf Vis (EuroVis)* 32(3pt2):201–210
 25. Su H, Zheng K, Huang J et al (2015) Calibrating trajectory data for spatio-temporal similarity analysis[J]. *VLDB J* 24(1):93–116
 26. Lin B, Su J (2008) One way distance: for shape based similarity search of moving object trajectories[J]. *Geoinformatica* 12(2):117–142
 27. Morris B, Trivedi M (2009) Learning trajectory patterns by clustering: experimental studies and comparative evaluation[C]. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp 312–319
 28. Domingo-Ferrer J, Trujillo-Rasua R (2012) Microaggregation-and permutation-based anonymization of movement data[J]. *Inf Sci* 208:55–80
 29. Wang C, Yang J, Zhang J (2015) Privacy preserving algorithm based on trajectory location and shape similarity[J]. *J Commun* 36(2):144–157
 30. Gudmundsson J, Valladares N (2015) A GPU approach to subtrajectory clustering using the fréchet distance[J]. *IEEE Trans Parallel Distrib Syst* 26(4):924–937
 31. Basse P, Guillouet B, Loubes J-M et al (2015) Review and perspective for distance based trajectory clustering[J]. *Comput Sci* 47(2):169–179
 32. Keogh E, Pazzani M (2001) Derivative dynamic time warping[C]. In: *Proceedings of the SIAM international conference on data mining*, pp 1–11
 33. Keogh E, Ratanamahatana CA (2005) Exact indexing of dynamic time warping[J]. *Knowl Inf Syst* 7(3):358–386
 34. Zhang D, Lee K, Lee I (2018) Hierarchical trajectory clustering for spatio-temporal periodic pattern mining[J]. *Expert Syst Appl* 92:1–11
 35. Dominigo-Ferrer J, Sramka M, Trujillo-Rasua R (2010) Privacy-preserving publication of trajectories using microaggregation[C]. In: *Proceedings of the 3rd ACM SIGSPATIAL international workshop on security and privacy in GIS and LBS - SPRINGL'10*, pp 26–33
 36. Brinkhoff T (2002) A framework for generating network-based moving objects[J]. *Geoinformatica* 6(2):153–180
 37. Piorkowski M, Sarafijanovic-Djukic N, Grossglauer M (2009) CRAWDAD dataset epl/mobility (v. 2009-02-24)[EB/OL]. <https://doi.org/10.15783/C7J010>
 38. Piorkowski M, Sarafijanovic-Djukic N, Grossglauer M (2009) A parsimonious model of mobile partitioned networks with clustering[C]. In: *Proceedings of the 1st international conference on communication systems and NETWORKS*, pp 1–10
 39. Laxhammar R, Falkman G (2014) Online learning and sequential anomaly detection in trajectories[J]. *IEEE Trans Pattern Anal Mach Intell* 36(6):1158–1173
 40. Han J, Kamber M, Pei J (2013) *Data mining: concepts and techniques*[M]. Morgan Kaufmann, San Francisco
 41. Amorim RCD, Hennig C (2015) Recovering the number of clusters in data sets with noise features using feature rescaling factors[J]. *Inf Sci* 324:126–145
 42. Xie J, Gao H, Xie W et al (2016) Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors[J]. *Inf Sci* 354(C):19–40
 43. Vinh NX, Epps J, Bailey J (2009) Information theoretic measures for clusterings comparison: is a correction for chance necessary?[C]. In *Proceedings of the 26th ACM international conference on machine learning*, pp 1073–1080



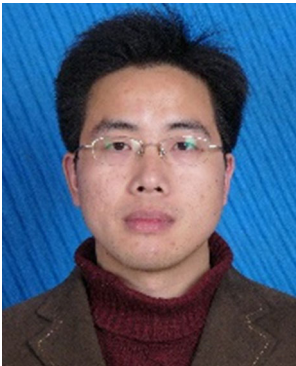
Qingying Yu received her B.S. degree and M.S. degree from Department of Computer Science and Technology, Anhui University, Hefei, China, respectively in 2002 and 2005. Currently, she is a Ph.D. Candidate at Anhui Normal University, Wuhu, China. Her main research interests are spatial data processing and information security.



Yonglong Luo received his Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China in 2005. Since 2007, he has been a professor in School of Mathematics and Computer Science, Anhui Normal University. Currently, he is a Ph.D. supervisor of Anhui Normal University. He is the Director of Anhui Provincial Key Laboratory of Network and Information Security. His main research interests are information security and spatial data processing.



Shigang Chen received his M.S. and Ph.D. degrees in Computer Science from University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. Since 2013, he has been a professor in Department of Computer and Information Science and Engineering, University of Florida. Currently, he is a Ph.D. supervisor of University of Florida. He is an IEEE Fellow, an ACM Distinguished Member, and an IEEE Com-Soc Distinguished Lecturer. His main research interests are computer networks, big data, internet security, etc.



Chuanming Chen received his B.S. degree and M.S. degree from Department of Computer Science and Technology, Anhui University, Hefei, China, respectively in 2002 and 2005. Currently, he is an associate professor at Anhui Normal University. His main research interests are data mining and intelligent computing.