Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5582341, 10 pages
https://doi.org/10.1155/2021/5582341

Research Article

An Approach to Spatiotemporal Trajectory Clustering Based on Community Detection

Xin Wang,[1] **Xinzheng Niu**,[1] **Jiahui Zhu**,[1] **and Zuoyan Liu**[2]

[1]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China*
[2]*Department of Rehabilitation Medical Center, West China Hospital, West China School of Nursing, Sichuan University, Chengdu, China*

Correspondence should be addressed to Zuoyan Liu; zuo.yan.2008@163.com

Received 20 February 2021; Revised 1 April 2021; Accepted 22 April 2021; Published 17 May 2021

Academic Editor: Zhipeng Cai

Copyright © 2021 Xin Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nowadays, large volumes of multimodal data have been collected for analysis. An important type of data is trajectory data, which contains both time and space information. Trajectory analysis and clustering are essential to learn the pattern of moving objects. Computing trajectory similarity is a key aspect of trajectory analysis, but it is very time consuming. To address this issue, this paper presents an improved branch and bound strategy based on time slice segmentation, which reduces the time to obtain the similarity matrix by decreasing the number of distance calculations required to compute similarity. Then, the similarity matrix is transformed into a trajectory graph and a community detection algorithm is applied on it for clustering. Extensive experiments were done to compare the proposed algorithms with existing similarity measures and clustering algorithms. Results show that the proposed method can effectively mine the trajectory cluster information from the spatiotemporal trajectories.

1. Introduction

Nowadays, a huge amount of data is collected and it is important to develop tools to analyze data to extract useful knowledge. The collected data is often multimodal, that is of different types (e.g., audio [1], video [2], text [3], and image [4]), and can be analyzed jointly or separately [5, 6]. An emerging type of data that is playing a key role in multimodal data analysis is trajectory data [7]. It consists of spatial and temporal information about moving objects. Common trajectory data can be divided into four categories, namely, human trajectories, vehicle trajectories, animal trajectories, and natural phenomenon trajectories. Analyzing and discovering patterns in trajectory data have applications in several fields such as intelligent transportation, human mobility analysis, urban planning, meteorology, and travel recommendations and can reveal insights that are not discovered from other data types.

The process of trajectory data analysis mainly consists of obtaining and preprocessing trajectory data, trajectory data management, and a variety of mining tasks, including trajectory pattern mining, privacy protection, outlier detection [8, 9], and clustering trajectories on complex road networks [10, 11]. Many studies have been published, and trajectory data analysis is a very active research field. A generative adversarial network (GAN) was used to predict pedestrian movement by analyzing multimodal trajectory data [12]. However, most techniques for trajectory data analysis require measuring trajectory similarity, which necessitates a large amount of calculations on trajectory data and results that the time complexity of these similarity measurement methods is relatively high. Based on the idea of branch and bound, a novel similarity measurement method, called FSTM [13], was proposed that sets a distance threshold to prune certain mismatched points. Still, FTSM only considers space constraints.

More recently, there is an increasing interest on time series clustering using graphs [14, 15]. Traditional analysis methods only focus on the local relationship between data samples, while ignoring the global information. Advanced trajectory data mining techniques take network dynamics of trajectories into account, such as to mine trajectory group patterns and to assess the importance of a moving object in trajectory networks [16–18]. A complex network is suitable for revealing important relationships in trajectory data visually and can provide global information as time series data. In addition, there is no restriction on the shape of clusters.

Based on the above advantages and limitations, we propose an approach to spatiotemporal trajectory clustering based on community detection (*STTC-CD*). The algorithm implements an improved branch and bound strategy based on time slice segmentation. While richer trajectory information is taken into consideration, redundant trajectory points are pruned. Then, the trajectory data is converted into a graph representation based on the similarity matrix. Finally, a suitable community detection algorithm is applied to perform clustering on the graph. The main contribution of this paper is as follows:

(i) An improved similarity calculation method is designed, which matches pairs of trajectory points and applies a pruning strategy based on time slicing to reduce the time complexity

(ii) A method is proposed to convert trajectories into a suitable data format to apply many types of techniques for trajectory data mining. Based on this, a community detection algorithm is applied to cluster trajectories, which captures global relationships among trajectories from a graph-based perspective

(iii) Experiments have been conducted to evaluate the proposed algorithm on several datasets to verify the influence of multiple factors. It was found that the proposed algorithm is more efficient than the compared methods

The rest of this paper is organized as follows: Section 2 surveys relevant related work. Section 3 formally defines the trajectory clustering problem. Section 4 presents the designed *STTC-CD* algorithm. Then, Section 5 describes the experimental evaluation and Section 6 draws a conclusion.

2. Related Work

The key problem in trajectory clustering is how to measure trajectory similarity. This section first reviews techniques for trajectory similarity measurement and then surveys relevant work on community detection.

2.1. Trajectory Similarity Measure. Most trajectory data analysis tasks require computing trajectory similarity measurements, such as trajectory clustering [19], transforming data for privacy-preservation [20], movement pattern mining [21], and abnormal trajectory detection [22]. Traditional trajectory measurement techniques such as EDR (edit distance on real sequence), LCSS (longest common subsequence), and DTW (Dynamic Time Warping) compute the overall trajectory similarity by analyzing each trajectory as a whole rather than considering subtrajectories or random trajectory points. Among these techniques, DTW [23] aligns trajectories of different lengths by warping a trajectory sequence and can match a point at a certain time from a trajectory to a number of continuous points from another trajectory. Hence, it has no restriction on the length of the compared trajectories. LCSS [24] calculates the longest common subsequence of two trajectories as their similarity. EDR calculates the minimum number of changes required to transform a trajectory into another as the similarity between the two trajectories. Clue-Aware Trajectory Similarity (CATS) [25] is aimed at overcoming the influence of track bias in time and space. Multidimensional Similarity Measure (MSM) [26] and Multiple-Aspect Trajectory Similarity Measure (MUITAS) [27] provide similarity measures for multidimensional sequences, adding information such as weather, user activity, and user interest into trajectory comparison.

However, DTW is a distance-based method, which directly accumulates the distances between trajectory point pairs. A problem of DTW is that the sum of the distances can greatly increase when there are noise points, which makes it sensitive to noise points. Quite the reverse, the ε-threshold-based measures use an ε-threshold value to determine if two points match, which can be more robust to noise. LCSS, EDR, CATS, and MSM fall all in the ε-threshold-based strategy, and the computation of similarity score is based on the point matching of two trajectories. They have a $O(n^2)$ time complexity and cause a performance bottleneck for trajectory clustering algorithms. Furtado et al. proposed a branch and bound method (FTSM) to achieve fast similarity measuring by utilizing a transitive range pruning strategy to reduce the number of matching point pairs.

2.2. Community Detection in Networks. A community is a subset of network nodes. Connections between nodes within a subset are relatively close, while connections between nodes from different subsets are relatively sparse, which is exactly in line with the needs and principles of clustering. Recently, community detection algorithms have been increasingly utilized for trajectory clustering.

Depending on whether a node can belong to multiple communities or only one, community detection methods can be categorized as finding nonoverlapping or overlapping communities. In a nonoverlapping community, each network node can belong to one community. Algorithms that detect communities of this type are Fastgreedy [28], Louvain [29], Label Propagation [30], and Infomap [31]. Modularity is used to measure the quality of community division. The Fastgreedy algorithm applies a bottom-up process. Initially, each node is regarded as a community. Then, at each iteration, the two communities providing the largest increase in modularity are merged until the entire network is merged into a single community. The final community structure is a division that maximizes the modularity. The Louvain algorithm improves upon the Fastgreedy algorithm by assigning each node to neighboring nodes for maximum modularity. When the ownership of a node no longer changes, the algorithm collapses each community into a node to form a new community for the next iteration. The basic idea of the Label Propagation algorithm (LPA) is to predict labels of unlabeled network nodes from labeled nodes. Each node label is propagated to neighboring nodes according to their similarity. At each step of node propagation, the node updates itself according to the label of the neighboring node until the label no longer changes. Similar to K-means, the results of LPA are affected by the initial label selection. The Infomap algorithm introduces a coding-based technique based on random walks. A good group division can lead to shorter coding length.

A trajectory clustering algorithm based on an improved Label Propagation algorithm was proposed where road network is modeled as a dual graph to capture and characterize the similarity between nodes [10]. Liu and Guo proposed a semantic trajectory clustering algorithm based on community detection [32], where different community detection algorithms were discussed.

3. Problem Statement

The following definitions are provided to facilitate the formulation of the problem under study:

Definition 1 (trajectory). A trajectory is a sequence of points in chronological order, denoted as $\mathrm{TR}_i = \{p_i^1, p_i^2, \cdots, p_i^j, \cdots, p_i^{n_i}\}$, where each point $p_i^j = (i, x, y, t)$ represents the spatial location (x, y) of an entity at given time t of trajectory TR_i, and n_i is the number of points in TR_i.

Definition 2 (silhouette coefficient SI). The silhouette coefficient is a metric to evaluate the quality of a clustering, which considers two aspects that are cohesion and resolution. The $s(i)$ of each trajectory point p_i is calculated as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad (1)$$

where $a(i)$ denotes the average distance from p_i to all trajectory points in the cluster to which p_i belongs, and $b(i)$ is the average distance between p_i and trajectory points in other clusters. Given a trajectory dataset $\mathrm{TS} = \{\mathrm{TR}_1, \mathrm{TR}_2, \cdots, \mathrm{TR}_N\}$, the silhouette coefficient of TS is the average of the silhouette coefficients of all trajectories, denoted as

$$\mathrm{SI} = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{n_j}\sum_{i=1}^{n_j} s(i), \quad (2)$$

where N is the number of trajectories, n_j is the number of trajectory points in TR_j, and $(1/n_j)\sum_{i=1}^{n_j} s(i)$ is the silhouette coefficient of trajectory TR_j.

The value of SI is between -1 and 1 such that a higher SI value indicates a better clustering result in general. According to the above definition, the road trajectory clustering optimization problem is defined as follows:

Definition 3 (trajectory clustering optimization problem). Given a set of trajectories $\mathrm{TS} = \{\mathrm{TR}_1, \mathrm{TR}_2, \cdots, \mathrm{TR}_N\}$ in Euclidean space for the time period $[0, T]$, the goal is to divide TS into groups $\{C_1, C_2, \cdots, C_{N_c}\}$ to maximize SI.

4. The Proposed STTC-CD Algorithm

This paper proposes an approach to spatiotemporal trajectory clustering based on community detection, named *STTC-CD*, which is applied in three steps: (1) trajectory partition, (2) graph generation, and (3) trajectory clustering, as illustrated in Figure 1.

Stage 1. Trajectory Partition. Given a collection of space-time trajectories $\{\mathrm{TR}_1, \mathrm{TR}_2, \cdots, \mathrm{TR}_N\}$, *STTC-CD* divides them into time slices and then utilizes transitive range pruning to calculate the number of pairs of matching points between trajectories in each time period to generate a matching matrix.

Stage 2. Graph Generation. STTC-CD aggregates the matching matrix of each time period to generate a global matching matrix. Then, the algorithm transforms the matching matrix into a similarity matrix according to similarity rules, and a trajectory-connected graph is generated.

Stage 3. Trajectory Clustering. Based on the trajectory graph obtained in the second stage, we utilize a community detection algorithm for clustering to capture global relationships between trajectories from the perspective of the network.

4.1. Trajectory Partition. An algorithm is proposed that takes the time characteristics of trajectories into account and utilizes a branch and bound strategy for fast trajectory similarity measurement. The algorithm is called *STTC-CD*. It not only improves the accuracy of similarity measurement but also only compares each trajectory segment with others from the same time slice instead of all trajectories, thereby improving computational efficiency through further pruning.

Given a trajectory dataset $\mathrm{TS} = \{\mathrm{TR}_1, \mathrm{TR}_2, \cdots, \mathrm{TR}_N\}$ and a partition threshold κ, TS is divided into κ subdatasets $\{\mathrm{TS}_1, \mathrm{TS}_2, \cdots, \mathrm{TS}_\kappa\}$ according to the time slice and then allocated to the corresponding subdataset of the time slice. Let $t_{\min}$ and $t_{\max}$ be the minimum and maximum timestamp in the dataset, respectively. The length of each time slice is defined as follows:

$$\Delta t = \frac{t_{\max} - t_{\min}}{\kappa}. \quad (3)$$

Each trajectory $\mathrm{TR}_i = \{p_i^1, p_i^2, \cdots, p_i^m, \cdots, p_i^{n_i}\} \in \mathrm{TS}$ is divided into subdatasets according to the time slice (as shown in Figure 2). The index of the subdataset to which a point p_i^m is assigned is $\lceil (t_i^m - t_{\min})/\Delta t \rceil$.

4.2. Graph Generation. The graph is generated based on the similarity matrix. The calculation of similarity in each time slice is done based on the following definitions:

Definition 4 (point matching (PM)). Let there be two points p_i and p_j, a matching threshold ε, and a distance function $\mathrm{dist}(p_i, p_j)$. If $\mathrm{dist}(p_i, p_j) \le \varepsilon$, then p_i and p_j match each other; otherwise, they do not match. The formula is defined as follows:

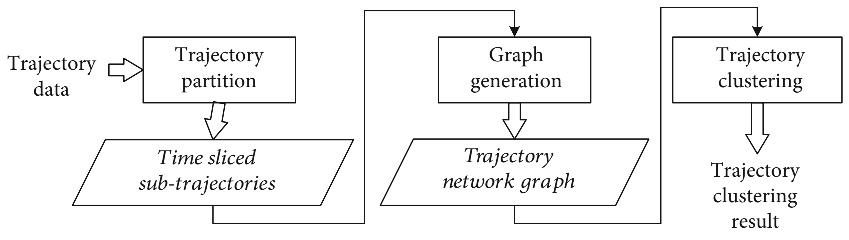


Figure 1: Algorithm flowchart.

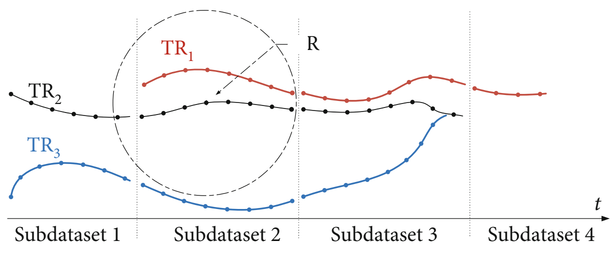


Figure 2: The schematic diagram of trajectory partition.

$$\mathrm{PM} = \begin{cases} 1, & \mathrm{dist}\left(p_i, p_j\right) \le \varepsilon, \\ 0, & \mathrm{dist}\left(p_i, p_j\right) > \varepsilon. \end{cases} \tag{4}$$

Definition 5 (trajectory segment matching (TM)). Given two trajectory segment $\mathrm{sTR}_i = \{p_1, p_2, \cdots, p_m\}$ and $\mathrm{sTR}_j = \{q_1, q_2, \cdots, q_n\}$, trajectory segment matching is defined as follows:

$$\mathrm{TM}\left(\mathrm{sTR}_i, \mathrm{sTR}_j\right) = \sum_{i=1}^{m}\sum_{j=1}^{n}\mathrm{PM}\left(p_i, q_j\right) + \sum_{j=1}^{n}\sum_{i=1}^{m}\mathrm{PM}\left(q_j, p_i\right), \tag{5}$$

where m and n are the numbers of the points of sTR_i and sTR_j.

Considering that trajectory elements are points in Euclidean space, the following definitions adopt the Euclidean distance as distance function to perform point matching. Hence, the matching threshold can be seen as the radius ε of a matching circle.

Definition 6 (pivot point). For a trajectory TR_i, the pivot point of TR_i is the point at half of the trajectory as follows:

$$p_i^k = p_i^{\lfloor (1+n_i)/2 \rfloor}, \tag{6}$$

where n_i is the number of trajectory points of TR_i.

Definition 7 (pruning radius (PR)). Given a pivot point $p_i^k \in \mathrm{TR}_i$ and a matching threshold ε, the pruning radius is a circle around p_i^k that covers all the points that are at maximum distance ε of any point in TR_i, that is,

$$\mathrm{PR} = \varepsilon + \max\left(\sum_{m=1}^{k-1}\mathrm{dist}\left(p_i^m, p_i^{m+1}\right), \sum_{n=k}^{n_i-1}\mathrm{dist}\left(p_i^n, p_i^{n+1}\right)\right). \tag{7}$$

Lemma 8 (transitive range pruning). *Let* $TR_i = \{p_i^1, p_i^2, \cdots, p_i^m, \cdots, p_i^{n_i}\}$ *and* $TR_j = \{p_j^1, p_j^2, \cdots, p_j^n, \cdots, p_j^{n_j}\}$ *be two trajectories,* ε *be a matching threshold,* $dist(p_i^m, p_j^n)$ *be the metric computing the distance between two points, and PR be a pruning radius around a pivot point* $p_i^k \in TR_i$*. Then, for any point* $p_i^m \in TR_i$ *and* $p_j^n \in TR_j$*,*

$$dist\left(p_j^n, p_i^m\right) \le \varepsilon \Rightarrow dist\left(p_j^n, p_i^k\right) \le PR. \tag{8}$$

This lemma [13] means that for any point in TR_j, if its distance to a certain point of TR_i is less than ε, then its distance to the pivot point of TR_i must be less than PR. Therefore, if the distance from a point to the pivot point of TR_i is greater than PR, the distance from it to all points of TR_i is greater than ε, and the pruning operation can be performed accordingly.

Based on the subdatasets generated in Stage 1, the number of matching points in each subdataset is calculated. Given two subtrajectories sTR_i and sTR_j, the calculation of point matching consists of three steps, as shown in Figure 3:

(a) Pruning step: the pivot point of sTR_i is denoted as p_i^k. For any point $p_j^m \in \mathrm{sTR}_j$, the distance is calculated from p_i^k to p_j^m and is compared with the threshold PR. If $\mathrm{dist}(p_i^k, p_j^m)\mathrm{PR}$, p_j^m is added to the matching queue

(b) Splitting step: sTR_i is separated from the pivot point p_i^k to form two subtrajectories. The center points of subtrajectories are taken as new pivot points, and the points in the matching queue form the new sT R$_j$. The pruning step is repeated until the matching queue is empty or the trajectory segment can no longer be divided

(c) Matching step: the points of sTR_j in the matching queue are matched with sTR_i to get the number of matching points

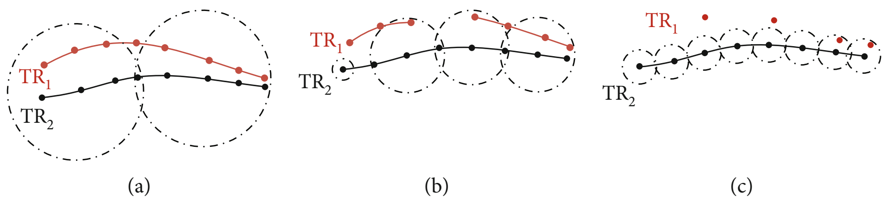


Figure 3: The schematic diagram of point matching.

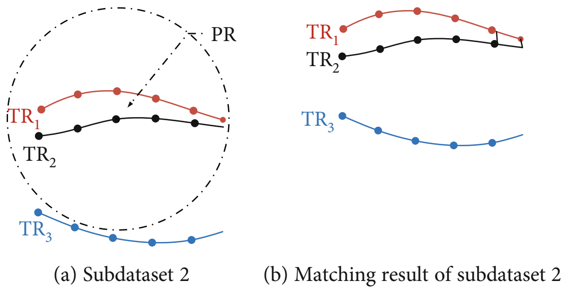


Figure 4: Matching step for a subdataset.

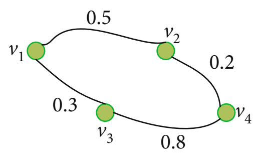


Figure 5: Trajectory graph.

Table 1: Datasets.

DS#	Dataset	Trajectory count	Avg. trajectory points	Time span
DS1	Trucks	1,100	85	39 days
DS2	T-drive	10,357	1448	7 days
DS3	UCI	163	111	493 days

For instance, Figure 4 shows one of the subdatasets after partition. Figure 4(a) is a subdataset consisting of three sub-trajectories, and Figure 4(b) shows the matching result of it, where the number of matching points between sTR_2 and other trajectories in subdataset 2 is calculated as 2 and 0.

The matching matrix is aggregated of each time slice. According to the matching point matrix, the similarity matrix can be obtained. The similarity is defined as follows:

Definition 9 (trajectory similarity measure). For two trajectories $\mathrm{TR}_1 = \{\mathrm{sTR}_1^{(1)}, \mathrm{sTR}_1^{(2)}, \cdots, \mathrm{sTR}_1^{(m)}\}$ and $\mathrm{TR}_2 = \{\mathrm{sTR}_2^{(1)}, \mathrm{sTR}_2^{(2)}, \cdots, \mathrm{sTR}_2^{(n)}\}$, the similarity of them is calculated as

$$\mathrm{Sim}(\mathrm{TR}_i, \mathrm{TR}_j) = \mathrm{Sim}(\mathrm{TR}_j, \mathrm{TR}_i) = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n}\mathrm{TM}\left(\mathrm{sTR}_i^{(p)}, \mathrm{sTR}_j^{(q)}\right) + \sum_{q=1}^{n}\sum_{p=1}^{m}\mathrm{TM}\left(\mathrm{sTR}_j^{(q)}, \mathrm{sTR}_i^{(p)}\right)}{m+n}, \tag{9}$$

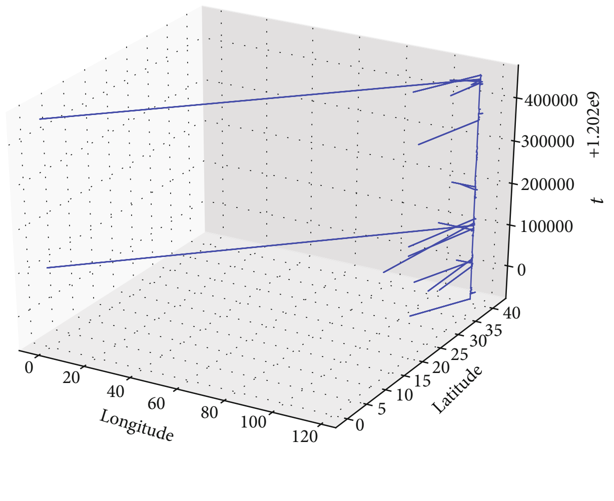
where m and n are the number of sub-trajectories in TR_i and TR_j, respectively. The similarity measure satisfies the property of nonnegativity, which means $\mathrm{Sim}(\mathrm{TR}_i, \mathrm{TR}_j) \geq 0$ in all cases, and a large score indicates a high similarity.

Then, the matching matrix is transformed by Equation (9) to obtain the similarity matrix S, where $\mathrm{Sim}(\mathrm{TR}_i, \mathrm{TR}_j)$ represents the similarity between TR_i and TR_j. A trajectory graph $G = (V, E)$ is constructed by exploiting the similarity matrix S. Firstly, N vertices are constructed for a dataset with N trajectories and each trajectory corresponds to a vertex. For each v_i corresponding to the trajectory TR_i and v_j corresponding to the trajectory TR_j, edge is added between them if $\mathrm{Sim}(\mathrm{TR}_i, \mathrm{TR}_j) > 0$. The weight of each edge is equal to the similarity between the two vertices. For instance, given a matrix $[[0,0.5,0.3,0],[0.5,0,0,0.2],[0.3,0,0,0.8],[0,0.2,0.8,0]]$, the trajectory graph is as shown in Figure 5.

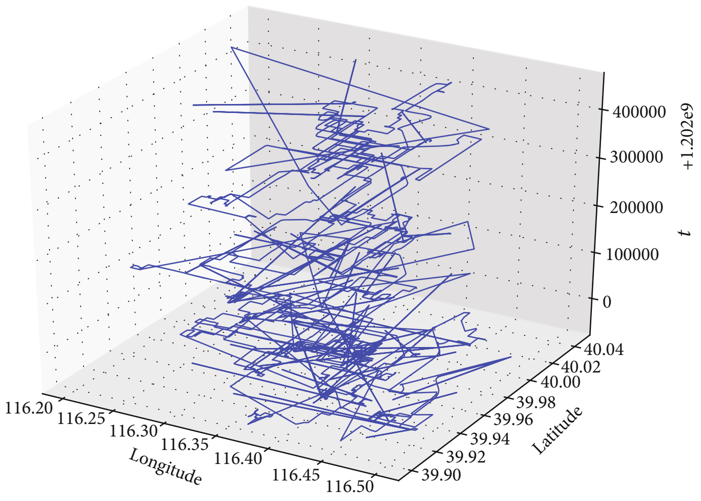
4.3. *Trajectory Clustering.*

A community is composed of a group of closely connected nodes that are sparsely connected with nodes outside the community. Community detection is to discover these closely connected community structures in a complex network, which coincides with the objective of clustering. Therefore, the Infomap algorithm [31] is employed for clustering, which combines community detection with information encoding.

The basic idea of the Infomap algorithm is to find the shortest codes to describe the path generated by a random walk on the network. This is done using a two-level coding of all network nodes to find the module partition with the shortest encoding length by minimizing entropy to find the optimal clustering. The two-level code assigns unique module names, and nodes in different modules are allowed to use repeated codewords. The module code is inserted before the nodes in the same module, and the termination mark is inserted at the end. The average code

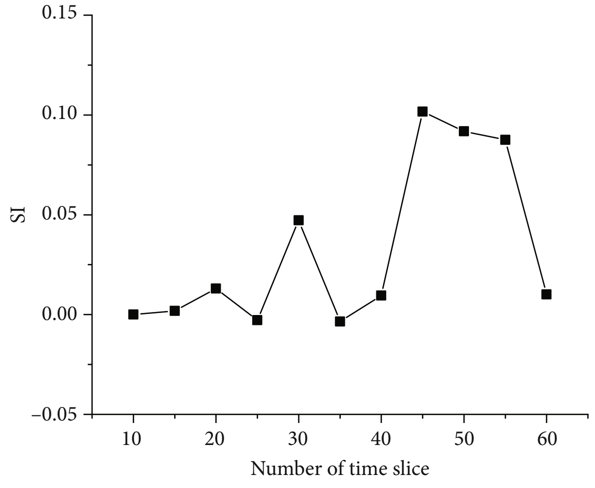


(a) Original trajectory 6275

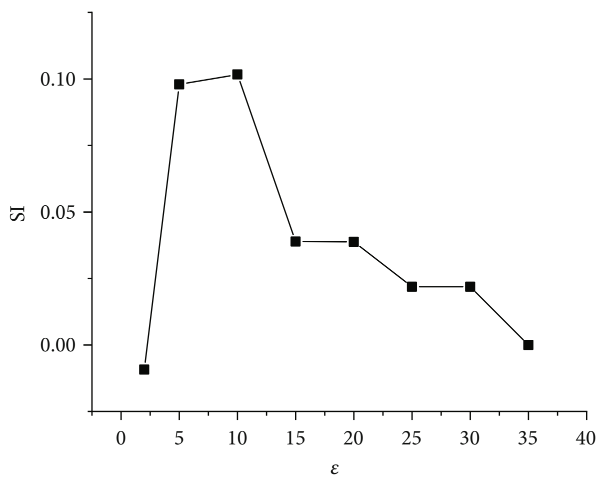


(b) Processed trajectory 6275

Figure 6: Comparison graph of trajectory processing.

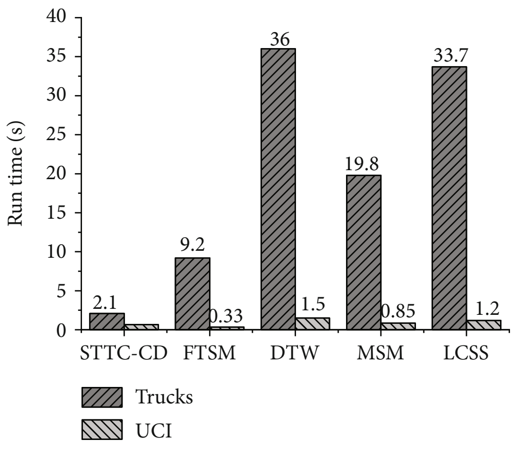


(a) Different time slice ($\varepsilon = 10$)

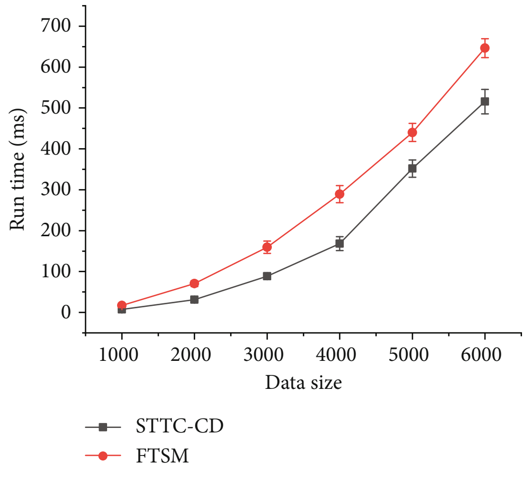


(b) Different ε (time slice = 45)

Figure 7: Clustering quality on trucks dataset using SI with different parameters.

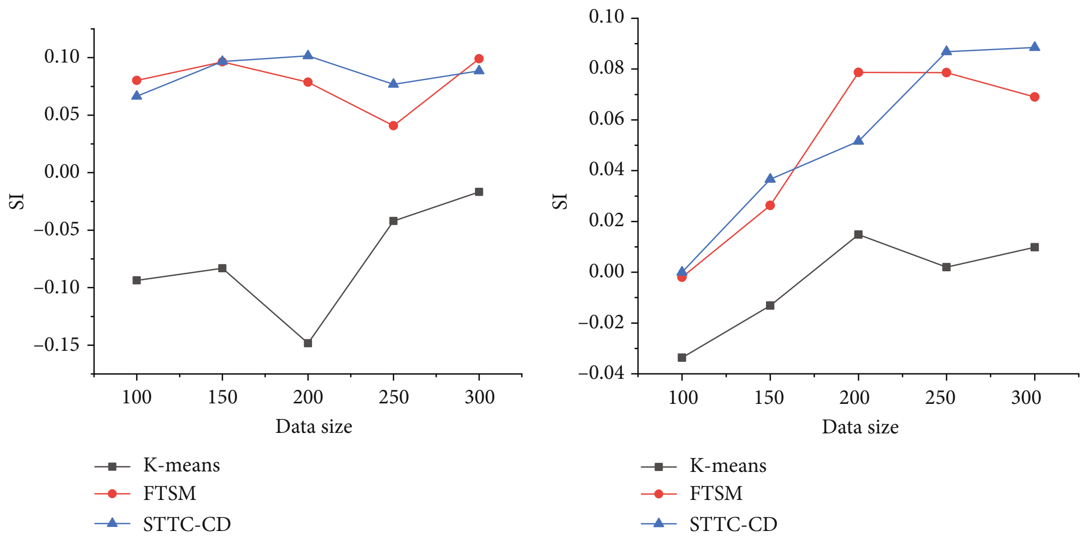


(a) Runtime on DS1 and DS3



(b) Runtime on DS2

Figure 8: Runtime.



(a) SI index on DS1

(b) SI index on DS2

Figure 9: SI.

length is calculated as follows:

$$L(M) = q \Subset H(Q) + \sum_i p^i_{O} H\left(P^i\right), \tag{10}$$

where $q \Subset$ represents the probability of switching from one module to another per step of the random walk, $H(Q)$ is the entropy of movements between modules, p^i_{O} denotes the proportion of all nodes in group i in the encoding, and $H(P^i)$ denotes the average code length required by all nodes in group i. The Infomap algorithm performs three steps:

Step 1. Initialization. Each graph node is treated as an independent group.

Step 2. Each node is traversed in a random order, and each point is assigned to the adjacent module that gives the largest decrease in Equation (10).

Step 3. Step 2 is repeated in a different random order until Equation (10) does not decrease.

5. Performance Evaluation

The performance of the proposed SSTC-CD algorithm was evaluated in terms of silhouette coefficient and runtime. All algorithms were implemented in Java 14, and all experiments were conducted on a Windows PC workstation equipped with an Intel(R) Core(TM) i5-10400 CPU@2.90 GHz and 16 GB of memory.

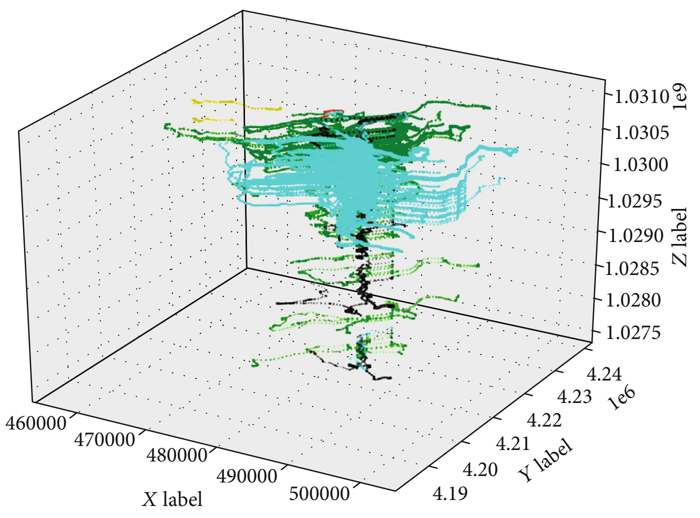
5.1. Datasets. The algorithm was evaluated on several widely used public datasets, described in Table 1. The trucks dataset (DS1) is a real-word dataset composed of 1,100 trajectories generated by 50 different trucks transporting concrete in Greece. T-drive dataset [33] (DS2), provided by Microsoft Research Asia, is a collection of trajectories generated by 10,357 taxis located in Beijing within a week. The UCI dataset (DS3) was collected by the GoTrack Android app in 2016. It has a high sampling rate for a single trajectory, but the interval between trajectories is long.

DS2 was collected in Beijing, which is located in longitude 115.7°E to 117.4°E and latitude 39.4°N to 41.6°N. Therefore, out-of-range points were deleted as abnormal points. The average trajectory length in DS2 is about 1,500 points. Yet, the longest trajectory has 150,000 points, and there are many repeated points and stay points, which we have removed from the dataset. Figure 6 presents the longest trajectory in DS2 with id 6275. Figure 6(a) is the original trajectory, and Figure 6(b) is the processed trajectory.

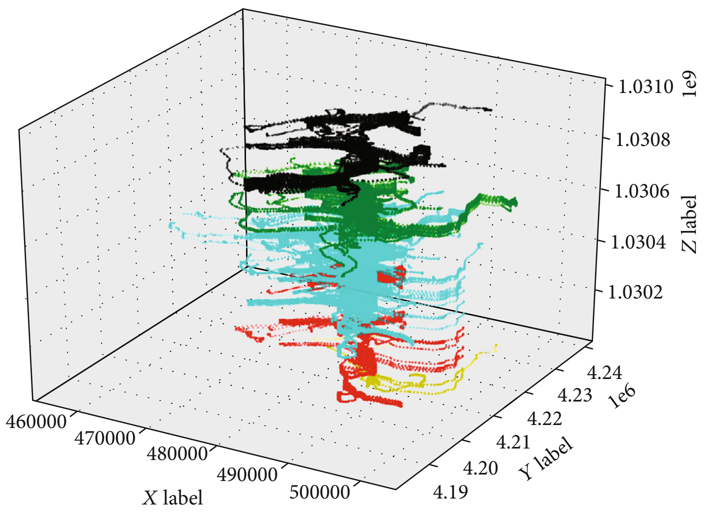
5.2. Evaluation. In our experiments, we run STTC-CD with different ε-threshold and different number of time slices to identify the optimal parameters. Figure 7 shows the influence of different parameters on the proposed algorithm. As shown in Figure 7(a), the SI index shows a trend of rising first and then falling as the number of time slices increases, and it reaches the maximum value when the number of time slices is 45. As shown in Figure 7(b), the value of ε was set from 2 to 35 and the SI index reaches its maximum value when ε is 10.

The performance of the proposed *STTC-CD* algorithm was compared with several similarity measurement algorithms, namely, FTSM [13], DTW [23], MSM [26], and LCSS [24], on DS1 and DS3. The parameter ε was set to 10, and the number of time slices was set to 45. Results are presented in Figure 8(a).

It can be observed that the running time of *STTC-CD* and FTSM on both datasets is shorter than that of other algorithms. For large datasets, the runtime gap is greater. The reason is that the other three algorithms are implemented using dynamic programming, which have quadratic time complexity. As the data size increases, the time required by these



(a) FTSM clustering results on DS1



(b) *STTC-CD* clustering results on DS1

Figure 10: Clustering result on DS1.

algorithms rises sharply. Since FTSM and *STTC-CD* pruned the sequence to be matched on the trajectory, the complexity is close to linear in the best case. When the data size is small, *STTC-CD* prunes more pair-wise trajectory points than FTSM by splitting in time slices. However, the operations of splitting and matching time slices take more time, which results in spending more time than FTSM.

To further evaluate FTSM and *STTC-CD*, DS2 was split into six subdatasets of different sizes and the two algorithms were applied. It can be seen in Figure 8(b) that when the dataset is small, the runtimes of the two algorithms are almost the same. As dataset size increases, the gap becomes more obvious. This result is also consistent with the results for the other two datasets.

The performance of algorithms was further compared in terms of the SI index. The time dimension of the dataset is considered in the algorithm; therefore, the three-dimensional Euclidean distance combined with the time dimension is utilized as the distance measure of SI. Compared with DTW, MSM, and LCSS implemented by dynamic programming, FTSM only pruned away some unnecessary comparisons, which improved the running speed of the algorithm without affecting the accuracy of the algorithm. Based on FTSM, the proposed algorithm further reduces the number of point matching in similarity calculation, but it also affects the accuracy of the algorithm. Therefore, the SI index was used to compare the accuracy of FTSM and STTC-CD, and K-means was used as the benchmark algorithm. As illustrated in Figure 9, the proposed algorithm was compared with FTSM and K-means with different numbers of trajectories on DS1 and DS2. It can be observed that the SI of FTSM and *STTC-CD* are greater than the SI of K-means on both datasets, and most of the time, *STTC-CD* results are better than FTSM, which indicates that the proposed *STTC-CD* takes better account of time correlation.

The clustering results of FTSM and *STTC-CD* on DS1 are displayed using lines of different colors, while trajectories from the same cluster are represented using the same color. Figure 10(a) shows the clustering result of FTSM, and Figure 10(b) shows the clustering result of *STTC-CD*. It is found that FTSM does not discriminate in the time dimension. In contrast, the proposed algorithm has better results in the division of time levels.

6. Conclusion

This article presented an approach to spatiotemporal trajectory clustering based on community detection (*STTC-CD*), which is based on time slicing to reduce the time for similarity calculation. *STTC-CD* relies on a new trajectory representation, which enables various algorithms such as for community detection to be applied for trajectory clustering. Experimental results have shown that the proposed algorithm can effectively reduce runtimes on large datasets and that clustering results are more meaningful in the time dimension.

The approach proposed in this paper is designed to analyze and cluster trajectory data. An interesting research possibility for future work is to see this work as a building block to build a system for analyzing multimodal data consisting not only of trajectory but also text, video, and audio data. In particular, a hybrid system could be developed combining the proposed approach with a neural network or other machine learning models.

Data Availability

The T-drive dataset used to support the findings of this study has been deposited in the Microsoft Research Asia (doi:10.1145/2020408.2020462). The trucks dataset used to support the findings of this study is included within the article "Clustering Trajectories of Moving Objects in an Uncertain World" (doi:10.1109/ICDM.2009.57). The UCI dataset used to support the findings of this study has been feed by Android app called GoTrack. It is available at Google Play Store.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is sponsored by the Key Research and Development Program under Grant No. 2020YFS0169, Science and Technology Department of Sichuan Province and the Science and Technology Planning Project of Sichuan Province under Grant No. 2020YFG0054, and the Joint Funds of the Ministry of Education of China.

References

[1] A. Crossan and S. Brewster, "Multimodal trajectory playback for teaching shape information and trajectories to visually impaired computer users," *ACM Transactions on Accessible Computing*, vol. 1, no. 2, pp. 1–34, 2008.

[2] C. Hori, T. Hori, T. Lee et al., "Attention-based multimodal fusion for video description," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4203–4212, Venice, Italy, 2017.

[3] B. Qiang, R. Chen, Y. Xie, M. Zhou, R. Pan, and T. Zhao, "Hybrid deep neural network-based cross-modal image and text retrieval method for large-scale data," *Journal of Circuits, Systems and Computers*, vol. 30, 2020.

[4] A. R. Groves, C. F. Beckmann, S. M. Smith, and M. W. Woolrich, "Linked independent component analysis for multimodal data fusion," *NeuroImage*, vol. 54, no. 3, pp. 2198–2217, 2011.

[5] D. Lahat, T. Adal, and C. Jutten, "Multimodal data fusion: an overview of methods, challenges and prospects," *Proceedings of the IEEE*, vol. 103, no. 9, pp. 1449–1477, 2015.

[6] J. Gao, P. Li, Z. Chen, and J. Zhang, "A survey on deep learning for multimodal data fusion," *Neural Computation*, vol. 32, no. 1, pp. 1–36, 2020.

[7] Z. Zhang, J. Rhim, M. TaherAhmadi, K. Yang, A. Lim, and M. Chen, "Sfu-store-nav: a multimodal dataset for indoor human navigation," *Data in Brief*, vol. 33, p. 106539, 2020.

[8] P. Peltola, J. Xiao, T. Moore, A. Jiménez, and F. Seco, "Gnss trajectory anomaly detection using similarity comparison methods for pedestrian navigation," *Sensors*, vol. 18, no. 9, article 3165, 2018.

[9] Z. Fu, W. Hu, and T. Tan, "Similarity based vehicle trajectory clustering and anomaly detection," in *IEEE International Conference on Image Processing*, vol. 2, pp. 602–605, Genova, Italy, 2005.

[10] X. Niu, T. Chen, C. Q. Wu, J. Niu, and Y. Li, "Label-based trajectory clustering in complex road networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 10, pp. 4098–4110, 2020.

[11] J. Zhu, X. Niu, and C. Q. Wu, "On a clustering-based approach for traffic sub-area division," in *Advances and Trends in Artificial Intelligence*, pp. 516–529, Springer, 2019.

[12] J. Amirian, J. B. Hayet, and J. Pettre, "Social ways: learning multi-modal distributions of pedestrian trajectories with gans," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2964–2972, Long Beach, USA, 2019.

[13] A. S. Furtado, L. L. Pilla, and V. Bogorny, "A branch and bound strategy for fast trajectory similarity measuring," *Data & Knowledge Engineering*, vol. 115, pp. 16–31, 2018.

[14] M. A. Ghamdi and Y. Gotoh, "Graph-based topic models for trajectory clustering in crowd videos," *Machine Vision and Applications*, vol. 31, no. 5, 2020.

[15] B. Sabarish, R. Karthi, and T. G. Kumar, "Graph similarity-based hierarchical clustering of trajectory data," *Procedia Computer Science*, vol. 171, pp. 32–41, 2020.

[16] T. Pechlivanoglou and M. Papagelis, "Fast and accurate mining of node importance in trajectory networks," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 781–790, Seattle, WA, USA, 2018.

[17] L. Zhao and G. Shi, "A trajectory clustering method based on Douglas-Peucker compression and density for marine traffic pattern recognition," *Ocean Engineering*, vol. 172, pp. 456–467, 2019.

[18] V. Mirge, K. Verma, and S. Gupta, "Dense traffic flow patterns mining in bi-directional road networks using density based trajectory clustering," *Advances in Data Analysis and Classification*, vol. 11, no. 3, pp. 547–561, 2017.

[19] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 2017.

[20] O. Abul, F. Bonchi, and M. Nanni, “Never walk alone: uncertainty for anonymity in moving objects databases,” in *IEEE International Conference on Data Engineering*, pp. 376–385, Cancun, Mexico, 2008.

[21] K. Zheng, Y. Zheng, N. J. Yuan, S. Shang, and X. Zhou, “Online discovery of gathering patterns over trajectories,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1974–1988, 2014.

[22] X. Zhao, Y. Rao, J. Cai, and W. Ma, “Abnormal trajectory detection based on a sparse subgraph,” *IEEE Access*, vol. 8, pp. 29987–30000, 2020.

[23] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series,” in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pp. 359–370, Seattle, Washington, 1994.

[24] M. Vlachos, D. Gunopoulos, and G. Kollios, “Discovering similar multidimensional trajectories,” in *Proceedings of the 18th International Conference on Data Engineering*, p. 673, San Jose, CA, USA, 2002.

[25] C. C. Hung, W. C. Peng, and W. C. Lee, “Clustering and aggregating clues of trajectories for mining trajectory patterns and routes,” *The VLDB Journal*, vol. 24, no. 2, pp. 169–192, 2015.

[26] A. S. Furtado, D. Kopanaki, L. O. Alvares, and V. Bogorny, “Multidimensional similarity measuring for semantic trajectories,” *Transactions in GIS*, vol. 20, no. 2, pp. 280–298, 2016.

[27] L. M. Petry, C. A. Ferrero, L. O. Alvares, C. Renso, and V. Bogorny, “Towards semantic-aware multiple-aspect trajectory similarity measuring,” *Transactions in GIS*, vol. 23, no. 5, pp. 960–975, 2019.

[28] M. E. Newman, “Fast algorithm for detecting community structure in networks,” *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 69, no. 6, article 066133, 2004.

[29] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, article P10008, no. 10, 2008.

[30] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” *Physical Review E*, vol. 76, no. 3, article 036106, 2007.

[31] M. Rosvall and C. Bergstrom, “Maps of random walks on complex networks reveal community structure,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 4, pp. 1118–1123, 2008.

[32] C. Liu and C. Guo, “Stccd: semantic trajectory clustering based on community detection in networks,” *Expert Systems with Applications*, vol. 162, p. 113689, 2020.

[33] Y. Zheng, “T-drive trajectory data sample,” 2011, t-Drive sample dataset, https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/.