Research Article

Grid-Based Whole Trajectory Clustering in Road Networks Environment

Fangshu Wang ¹, Shuai Wang ², Xinzheng Niu ³, Jiahui Zhu ³ and Ting Chen ³

¹*School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China*

²*School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China*

³*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China*

Correspondence should be addressed to Shuai Wang; wangshuai0601@uestc.edu.cn

Received 5 August 2021; Accepted 22 October 2021; Published 24 November 2021

Academic Editor: Jerry Chun-Wei Lin

Copyright © 2021 Fangshu 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.

In the data mining of road networks, trajectory clustering of moving objects plays an important role in many applications. Most existing algorithms for this problem are based on every position point in a trajectory and face a significant challenge in dealing with complex and length-varying trajectories. This paper proposes a grid-based whole trajectory clustering model (GBWTC) in road networks, which regards the trajectory as a whole. In this model, we first propose a trajectory mapping algorithm based on grid estimation, which transforms the trajectories in road network space into grid sequences in grid space and forms grid trajectories by recognizing and eliminating redundant, abnormal, and stranded information of grid sequences. We then design an algorithm to extract initial clustering centers based on density weight and improve a shape similarity measuring algorithm to measure the distance between two grid trajectories. Finally, we dynamically allocate every grid trajectory to the best clusters by the nearest neighbor principle and an outlier function. For the evaluation of clustering performance, we establish a clustering criterion based on the classical Silhouette Coefficient to maximize intercluster separation and intracluster homogeneity. The clustering accuracy and performance superiority of the proposed algorithm are illustrated on a real-world dataset in comparison with existing algorithms.

1. Introduction

With the advancement of Global Position System (GPS) technology and the growing economy, people's travel is becoming fast and convenient. The massive position and movement information of moving objects are generated continuously, forming large-scale trajectory data. It is of great academic significance and commercial value to mine the underlying distribution information and the evolution rules, such as urban function partition (Niu et al. [1]), traffic jam prediction (Yu et al. [2]), and privacy protection (Wang et al. [3]). In addition, some classical trajectory clustering algorithms such as DBSCAN (Ester et al. [4]) are widely used in anomaly trajectory detection and anomaly event preven-

tion (Belhadi et al. [5], Belhadi et al. [6], Djenouri et al. [7]). As a significant branch of trajectory data mining, trajectory clustering mainly divides trajectory with high similarity or small distance into one cluster. The main purpose of trajectory clustering is to find representative path or the common moving tendency of different moving objects and extract human behavioral pattern and distribution rules of hot events contained in massive information.

In recent years, a large number of trajectory clustering studies have been published, which can be divided into three categories: trajectory points clustering, subtrajectories clustering, and whole trajectories clustering. Trajectory point clustering partitions are with similar GPS points into the same cluster based on the similarity criteria. Subtrajectory

FIGURE 1: The comparison between subtrajectories clustering and whole trajectories clustering. (a) 9 trajectories in the road network. (b) Partitioning trajectories and subtrajectory clustering result. (c) Whole trajectory clustering result.

clustering firstly divides the whole trajectory into several trajectory segments according to the time stamp, the direction of trajectory points, and road condition and then clusters the trajectory segments based on the similarity between them, while whole trajectory clustering directly forms clusters by calculating the similarity between the whole trajectories. Trajectory point clustering is more applied to extract hotspots. Clustering based on subtrajectories tends to mine movement patterns within a specific period or road segment, but the unit of whole trajectories clustering is much wider in the range of time and space, so this approach can reflect the complete movement trend and rule of trajectories, and the continuous and whole trajectory in the road network shows the connection between the trajectory owner and the external world. That is to say, the whole trajectory clustering can provide more comprehensive support for analyzing different movement patterns in a day and predicting users' next travel information. Figure 1(a) shows 9 trajectories in the road network, which are marked with different colors. Figures 1(b) and 1(c) show the clustering results by the subtrajectory clustering algorithms and the whole trajectory clustering algorithm for the 9 trajectories in Figure 1(a). In both figures, the same color trajectories are in the same cluster. As can be seen from Figures 1(b) and 1(c), these 9 trajectories are clustered by the classic subtrajectory clustering algorithms TraClus [8] and the classic whole trajectory clustering algorithm GridCSD-TraceMob (Han et al. [9]), and they are partitioned into 4 clusters. In Figure 1(b), those marked black squares in some trajectories are the turning points where the trajectories are divided into a series of subtrajectories based on angular offset of adjacent trajectory points and road segments by the algorithm TraClus. In Figure 1(b), it can be observed that the subtrajectories clustering algorithm mines the local trajectory information located in subroad segment but ignores the complete movement rules of objects. However, the whole trajectory clustering treats the trajectory as a complete path, which can better reflect the overall information of the trajectory, as shown in Figure 1(c).

Most of the existing whole trajectories clustering algorithms usually take K -means [10], DBSCAN, and other basic clustering algorithms as the premise and introduce or improve different trajectories distance measurement standards to complete the partition of similar trajectories. Yanagisawa et al. [11] represented the trajectory data as the directed line segments in space and defined the similarity between trajectories as the Euclidean distance between the directed discrete lines, but the algorithm can only compare the trajectories with the same time interval or the same length. To solve this problem, several methods based on warping distance are defined in literatures [12, 13], while Lin and Su [14] propose a method to compare the space shape of trajectory. But these methods still rely on all the position points in trajectories and assume that these points are accurately collected in the road network, while Gariel et al. [15] reorganized the trajectory sequences by identifying the turning points in the trajectory or using the resampling of principal component analysis and the augmented trajectory method and then partitioned similar trajectories based on representative sequences. Although the whole trajectory clustering is well completed in this method, it is not suitable for the vehicle trajectories with very irregular moving process under road network constraints, and the clustering results are limited by the accuracy of the extraction of trajectory representative sequence.

In this paper, the concept of grid cell space is introduced, and a whole trajectory clustering model in road network environment is proposed. Firstly, the sequence of trajectory points in road network space is mapped to grid trajectory based on grid cell space and grid estimation algorithm. Secondly, the center density rule and shape similarity measure are introduced to extract the initial cluster centers, and finally, the outlier function and nearest neighbor principle are combined to dynamically adjust and update the clusters of grid trajectory. The key information of the original trajectory in the road network is retained, and the clustering results with high accuracy are obtained, while solving the problem of large amount of trajectory data and their complex structure.

In sum, our work makes the following technical contributions to the area of trajectory clustering:

- (i) A grid cell space is defined for the scattered and changing trajectory data, and an effective mapping algorithm based on grid estimation is designed to transform the complex trajectories in the road network space into the plane grid trajectories in the grid cell space with the original spatial structure preserved
- (ii) A clustering algorithm of whole grid trajectories is proposed based on center density rule, shape similarity measure, and anomaly function. The algorithm can accurately identify the abnormal trajectories in the dataset and quickly and efficiently divide the grid trajectories into clusters
- (iii) A mapping-clustering-verification framework provides a trajectory clustering analysis model with a Silhouette index-based criterion for clustering performance evaluation.

The rest of this paper is organized as follows. Section 2 conducts a survey of related work. The whole trajectory clustering problem is defined in Section 3. The design of a grid-based whole trajectory clustering model (GBWTC) in road networks is detailed in Section 4. We present experimental results in Section 5 and conclude our work in Section 6.

2. Related Work

Trajectory clustering is mainly divided into point-based clustering, subtrajectories clustering, and whole trajectories clustering. The point-based approaches take GPS points as the basic unit for clustering. The trajectory spatial aggregation pattern is mined and analyzed based on the sparse and dense spatial distribution of vehicle trajectory points, to extract hot area information or key road information. Qiu and Wang [16] improved the structure of the DBSCAN method by combining the position and orientation of the trajectory points, proposed the O-DBSCAN algorithm to divide the entire trajectory point set into representative clusters, and used Gestalt's law to infer the route map. Lu et al. [17] made a breakthrough from the DBSCAN algorithm, redefining the core neighborhood and core objects in the DBSCAN algorithm, introduced a kernel function to measure the similarity between trajectory points, and finally extracted road segments based on the optimized DBSCAN structure information. Yan-Wei et al. [18] further extended the related terminology of DBSCAN, introduced the concept of density through grid cells, converted DBSCAN's extended clustering based on the density of data points to an extended cluster based on the density of cell, and proposed a simple and efficient fast density clustering algorithm CBSCAN. The algorithm can quickly find clustering patterns and noises of arbitrary shapes in location big data. Different from Qiu and Lu, Yu et al. [19] proposed a grid density algorithm based on trajectory points to identify hot spots in different periods of time and used spatiotemporal trajectory cluster-

ing methods to mine frequent paths between hot spots. Although the clustering of trajectory points is convenient, concise, and easy to understand, its essence destroys the time continuity of the trajectory. At the same time, it increases the clustering cost due to the similarity calculation of the time complexity of the Cartesian product between the trajectory points in the most of point-based clustering algorithms.

The concept of subtrajectory clustering first appeared in the TraClus [8] algorithm proposed by Lee and Han. The algorithm divides the trajectory into several trajectory segments based on the principle of minimum description length and clusters these trajectory segments based on the DBSCAN algorithm and Euclidean distance. The algorithm has a good effect on hurricane data and animal migration data, but the results have not been very good on real road trajectory datasets, and there are problems such as many clustering parameters and parameter sensitivity. At present, there are a lot of researches to correct these shortcomings, such as the ATCGD algorithm (Mao et al. [20]), NEAT algorithm (Binh Han et al. [21]), and LBTC algorithm (Niu et al. [22]). The ATCGD algorithm maps the divided subtrajectory segments to the grid cell space, then calculates the number of trajectory segments in the grid cell and the distance of the trajectory segments based on this mapping space, adaptively determines the parameters based on the DBSCAN method, and finally completes the clustering. The NEAT algorithm comprehensively considers the speed, flow, density, and other factors of the trajectory. By revising the Hausdorff distance calculation formula, the calculation of the vertical distance, parallel distance, and angular distance between all the line segments of the two subtrajectory sets in the TraClus algorithm is transformed into the distance between the endpoints of two representative trajectories. The flow clusters are combined according to the revised flow distance calculation formula by optimizing the distance between the two flow clusters. However, the road segments are mainly clustered in this algorithm through the traffic flow, and user trajectories are not specifically clustered, so it cannot accurately mine a large amount of user trajectory clustering information. Kumar et al. [23] proposed the Fast-clusiVAT algorithm for this deficiency of NEAT. First, the trajectory is decomposed into a directed graph or an undirected graph. In the process of executing the DTW algorithm, a step is added of using the Dijkstra algorithm to calculate the shortest path between two trajectory segments within a specified range. Moreover, this algorithm can accurately find the trajectory clusters in the dense area of the real road network, but it cannot solve the multidimensional problem of the trajectory. Bermingham and Lee [24] proposed a highly versatile n -dimensional data clustering algorithm and an arbitrary-dimensional representative trajectory extraction algorithm within a cluster, which can cluster any number of trajectory datasets and express valuable, previously unknown higher dimensional trajectory patterns.

In addition, some subtrajectory clustering algorithms expand around subtrajectories, and they generally need to calculate the distance between each point on the subtrajectory and finally add the weights of several different distances. For example, Salarpour and Khotanlou [25] used spectral

clustering to segment the trajectory, proposed a trajectory description method based on the change of the subtrajectory direction, and measured the similarity of the described trajectory based on the time warping matching algorithm. Taking into account the uncertainty of trajectory data, Guo et al. [26] proposed a similarity measurement method based on an amended ellipse model, referred to as UTSM, to reduce the interpolation error and positioning error. This method has good robustness and tolerance to abnormal data and noise. In order to clearly describe the difference between the subtrajectories of a moving target, Liu and Zhang [27] proposed a distance measurement method between subtrajectories based on time, space, and direction, but this method ignores the key factor of moving speed. Yu et al. [28] put forward a multifeature subtrajectory similarity measurement method which comprehensively considered the subtrajectory's direction, speed, time, and space location. Trajectory segments are used as the basic unit of similarity evaluation in the subtrajectory clustering algorithm, which reduces the clustering cost to a certain extent and more comprehensively considers the characteristics of trajectory data to accurately identify local differences in trajectories. However, the segmented feature points are difficult to identify and easy to lose, so the subtrajectory clustering algorithm is not good at mining users' complete travel rules, and the clustering result is also easily affected by the segmentation method.

The whole trajectory clustering is to use the whole trajectory as a clustering unit from a more macroperspective, define different whole trajectory similarity evaluation methods according to the scene, and cluster trajectories to mine their information. Domingo-Ferrer and Trujillo-Rasua [29] proposed a trajectory similarity measurement algorithm spatially and temporally and clustered the trajectories through a microaggregation algorithm, while [30] comprehensively considered the spatial, temporal, and shaped characteristics of the trajectories to calculate the similarity between trajectories, and a greedy clustering algorithm is proposed based on this, but it needs to traverse all points in the trajectory to calculate the distance between the trajectories, which consumes more memory. In order to reduce computational cost and improve efficiency, Pan et al. [31] used specific sampling of the complete trajectory and evaluated the similarity between the trajectories based on the sampling points and their density. Experiments show that this method can significantly improve the whole trajectory clustering efficiency while ensuring the accuracy of clustering. The TAD algorithm (Yang et al. [32]) was effective for various complex or special trajectories with long-duration gaps by introducing a noise tolerance factor to evaluate and deal with the influence of noise. Wang et al. [33] proposed a novel vehicle trajectory clustering method based on dynamic network representation learning which can avoid biased results. Stefan et al. [34] proposed a time series distance measurement method MSM based on edit distance, which defines the three steps of move, split, and merge to calculate the cost of mutual conversion between two time series. However, this method is only suitable for simple time series, not for complex or long trajectory series. Yao et al. [35] used a sliding window to extract the movement features

of each attribute of the input trajectory and convert it into a feature sequence. The quality representation of each trajectory is obtained by a convolutional neural network, and finally, high-quality trajectory clusters are obtained. However, the automatic encoding of trajectories by deep learning belongs to supervised learning, and it is difficult to be widely used in trajectory data lacking label information. Han et al. [9] proposed the whole trajectory algorithm TRACEMOB, which uses the coincidence rate of the trajectory in the grid as the basis for the trajectory similarity and converts the distance in the grid space into a d -dimensional Euclidean space. Finally, the K -means-based algorithm is used to complete the clustering. But the algorithm does not screen abnormal trajectories or trajectory points and requires secondary mapping before clustering, which is inefficient and easy to cause errors.

The whole trajectory clustering algorithm regards the trajectory as a whole and ensures the integrity of the trajectory compared to the trajectory point clustering and subtrajectory clustering. This algorithm has achieved good results in trajectory clustering. However, the above methods lack effective trajectory preprocessing steps and concise and fast trajectory similarity measurement. In addition, most of them only focus on the accuracy of clustering and neglect their application in the actual road network. For these limitations, we propose a grid-based whole trajectory clustering model in the road network environment, which is aimed at solving the problem of inefficient clustering caused by redundant trajectory points in the road network and inaccurate positioning. Without destroying the internal structure of the trajectory, the complete trajectory is accurately divided into corresponding clusters.

3. Problem Statement

3.1. Trajectory. A trajectory Tr_i of any object in road networks is represented by a list of spatiotemporal points sampled at equal time intervals, denoted as $Tr_i = \langle p_1, p_2, \dots, p_j, \dots, p_n \rangle$, where $p_j = ((x_j, y_j), t_j)$ represents the geographic location coordinates (x_j, y_j) of the moving object and t_j is the time stamp recorded when the moving object passes through the location point.

According to the above definition, we formulate the problem of trajectories clustering as follows. Given a set of trajectories $TS = \{Tr_1, Tr_2, \dots, Tr_p\}$, it is divided into N_C different clusters $C = \{C_1, C_2, \dots, C_{N_C}\}$. The quality of clustering is usually evaluated by intercluster separation and intracluster homogeneity [36]. In general, a larger intercluster separation and a higher degree of intracluster homogeneity indicate a more accurate clustering. In this work, we adopt the Silhouette Coefficient (SI) which is widely used in clustering validation, to measure the clustering quality of road networks.

The method of Silhouette Coefficient combines the degree of separation and homogeneity to measure the similarity between any trajectory Tr_i and other trajectories of its cluster, and the similarity of other trajectories of different clusters. Specifically, the Silhouette Coefficient is defined as

FIGURE 2: Trajectories located in different road segments but close to each other.

$$SI = \frac{1}{N_C} \sum_{i=1}^{N_C} \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} \frac{b(j) - a(j)}{\max(a(j), b(j))}, \quad (1)$$

where $|C_i|$ is the number of trajectories in C_i , $a(j)$ is the average distance between trajectory Tr_j and other trajectories in the same cluster, and $b(j)$ is the minimum average distance between trajectory Tr_j and other clusters. Note that the distance here is calculated by the edit distance of grid trajectories algorithm (EDG). The average of the Silhouette Coefficients of all trajectories in TS is the total SI of the clustering result [37]. The value of SI ranges from -1 to 1 , and the closer to 1 , the better homogeneity and separation are.

We formally define the road trajectory clustering optimization problem as follows:

Given a set of trajectories $\{Tr_1, Tr_2, \dots, Tr_p\}$, our goal is to process trajectories and abnormal data, to divide the trajectories into groups $\{C_1, C_2, \dots, C_{N_C}\}$ under the defined similarity criteria, and to output each cluster of trajectories so that the value of SI is close to 1 .

In fact, the traditional trajectory clustering based on road networks divides the trajectories located in the same road segment into a group, while ignoring the trajectories located in different road segments but close to each other, as shown in Figure 2. In addition, there may be some subroad segments in a road segment, and the distance between trajectories in the same subroad segment is smaller than that in different subroad segments. However, these trajectories are divided into the same cluster since they all share the same road segment. Therefore, it is unreasonable to cluster vehicle trajectories only based on road segments, so we introduce a whole trajectory clustering model based on grid cell space to solve these problems.

4. A Grid-Based Whole Trajectory Clustering Model: GBWTC

This section will elaborate the proposed grid-based whole trajectory clustering model in road network environment, referred to as GBWTC, from the two stages of grid trajectory

FIGURE 3: Overview of our proposed model.

serialization and overall clustering algorithm based on grid trajectory. The specific flowchart is shown in Figure 3.

Phase 1 for grid trajectory serialization is as follows: based on the trajectory mapping algorithm, the trajectories in road network space are transformed into grid sequences in grid space, and the redundant, abnormal, and stranded information of grid sequences are eliminated to form the representative sequence of trajectories, i.e., grid trajectories. While retaining the key information of the original trajectory, they can express the moving trend of the trajectories concisely and accurately.

Phase 2 for overall clustering based on grid trajectory is as follows: in the grid trajectory serialization phase, the original trajectory clustering problem in road network is transformed into an overall clustering problem of plane grid trajectory. While K -means is taken as the core in this stage, and the center density rule, shape similarity measure, and outlier function are introduced to deal with the whole clustering of grid trajectories in plane space.

4.1. Grid Trajectory Serialization. Grid trajectory serialization (*gridTrS*) is the process of transforming trajectories from road network space into grid space. This section first gives the following definitions to better describe the process:

Grid space: given a set of trajectories $TS = \{Tr_1, Tr_2, \dots, Tr_p\}$, the grid space is the minimum boundary rectangle required to cover any trajectory in TS , defined as $R(G) = Rect(L, W)$, where $L = lat_{\max} - lat_{\min}$ and $W = lon_{\max} - lon_{\min}$, which is the actual geographic coordinate range of the trajectory set TS . $lat_{\max}$ and $lon_{\max}$ are the maximum x and y coordinates of all trajectory points in TS , and $lat_{\min}$ and $lon_{\min}$ are the minimum x and y coordinates of all trajectory points in TS .

Grid cell space: given grid space $R(G)$ and grid cell size *gridsize*, grid cell space is a square with *gridsize* side length in the grid space, defined as

$$S_G(R(G), \text{gridsize}) = \{g_{m,n} : 1 \leq m \leq \text{longrid}, 1 \leq n \leq \text{latgrid}, g_{m,n} = \text{Rect}(\text{gridsize}, \text{gridsize})\}, \quad (2)$$

where $longrid = \lceil L/gridsz \rceil$, $latgrid = \lceil w/gridsz \rceil$, $\lceil \cdot \rceil$ is rounding up the value, and $g_{m,n}$ denotes a grid cell with $gridsz$ of the length and width in row m and column n of $S_G(R(G), gridsz)$.

Given a set of trajectories $TS = \{Tr_1, Tr_2, \dots, Tr_p\}$, we first extract grid space $R(G)$ and divide $R(G)$ based on the given $gridsz$ to form grid cell space $S_G(R(G), gridsz)$. Secondly, every trajectory Tr_i in the TS is mapped to grid cell space $S_G(R(G), gridsz)$ -based mapping relationship, and each GPS point $p(x, y, t)$ in the trajectory Tr_i falls into the grid cell $g_{m,n}$ corresponding to its position, where the mapping relationship between $p(x, y, t)$ and $g_{m,n}$ is defined as

$$f(p(x, y, t), g_{m,n}) = \begin{cases} \left\lceil \frac{(x - lat_{\min})}{gridsz} \right\rceil, \\ \left\lceil \frac{(y - lon_{\min})}{gridsz} \right\rceil, \end{cases} \quad (3)$$

where x, y are the geographic coordinates of point p , $lat_{\min}$ and $lon_{\min}$ are the minimum values of the x and y coordinates of all trajectory points in TS , and $\lceil \cdot \rceil$ is rounding up the value. Any trajectory $Tr_i = \langle p_1, p_2, \dots, p_k, \dots, p_h \rangle$ can obtain the corresponding grid cell sequence representing the trajectory points, referred to as $gridTr_i$, i.e., the grid trajectory $gridTr_i = \langle c_{m_1, n_1}^i, c_{m_2, n_2}^i, \dots, c_{m_k, n_k}^i, \dots, c_{m_h, n_h}^i \rangle$, where c_{m_k, n_k}^i is the grid cell mapped by the trajectory point p_k on the $gridTr_i$, and m_k, n_k represent the row and column numbers of the grid in $S_G(R(G), gridsz)$, respectively, i is the unique identifier of $gridTr_i$. However, grid trajectory will be redundant, and the clustering cost will increase if several trajectory points are mapped to the same grid cell. Figure 4 shows grid trajectory $gridTr_1 = \langle c_{m_1, n_1}^1, c_{m_2, n_2}^1, \dots, c_{m_{16}, n_{16}}^1 \rangle$ mapped by trajectory Tr_1 in grid cell space S_G , i.e., $gridTr_1 = \langle g_{3,4}, g_{3,4}, g_{3,4}, g_{2,4}, g_{2,4}, g_{2,3}, g_{2,3}, g_{2,2}, g_{2,2}, g_{2,2}, g_{2,1}, g_{2,1}, g_{1,1}, g_{1,1}, g_{1,1} \rangle$. There are many duplicate grid cells in the grid trajectory because many points map the same grid cell; besides, the trajectory points are easy to drift due to the influence of the collected signals. Therefore, this paper proposes a trajectory mapping algorithm based on grid estimation (TMGE). This method combines the characteristics of grid cell space and further identifies and eliminates the abnormal, redundant, and stranded grid cells based on the grid trajectory and completes the data structure optimization of the grid trajectory.

Specifically, TMGE first forms grid trajectories set $GTS = \{gridTr_1, gridTr_2, \dots, gridTr_p\}$ corresponding to $TS = \{Tr_1, Tr_2, \dots, Tr_p\}$ based on grid cell space and the mapping relationship. Then, any grid cell c_{m_o, n_o}^i is selected from any grid trajectory $gridTr_i$ in GTS , and its previous grid cell $c_{m_{o-1}, n_{o-1}}^i$ in $gridTr_i$ is confirmed if it is the same as c_{m_o, n_o}^i . If they are the same, grid cell c_{m_o, n_o}^i is judged as the redundant cell and is removed from the grid trajectory $gridTr_i$. If they are not same, grid cell c_{m_o, n_o}^i is confirmed if it is adjacent to its previous grid cell $c_{m_{o-1}, n_{o-1}}^i$ and next grid cell $c_{m_{o+1}, n_{o+1}}^i$. If

FIGURE 4: Trajectory Tr_1 is mapped to grid cell space.

they are not adjacent, i.e., $c_{m_o, n_o}^i \notin Adja(c_{m_{o-1}, n_{o-1}}^i)$ and $c_{m_o, n_o}^i \notin Adja(c_{m_{o+1}, n_{o+1}}^i)$, grid cell c_{m_o, n_o}^i is judged as abnormal or outlier cell and is removed from $gridTr_i$, where the set of adjacent grid cells $Adja(c_{m_o, n_o}^i)$ of c_{m_o, n_o}^i are the grid cells with the same row(column) as c_{m_o, n_o}^i and whose column (row) spacing difference is 1 with c_{m_o, n_o}^i . $Adja(c_{m_o, n_o}^i)$ is denoted as

$$Adja(c_{m_o, n_o}^i) = \{c_{m_k, n_k}^i \mid (|m_k - m_o| = 1 \& n_k = n_o) \cdot |n_k - n_o| = 1 \& m_k = m_o\}. \quad (4)$$

As is shown in Figure 5, TMGE first maps trajectory $Tr_2 = \langle p_1, p_2, \dots, p_{13} \rangle$ to grid trajectory $gridTr_2 = \langle c_{m_1, n_1}^2, c_{m_2, n_2}^2, \dots, c_{m_{13}, n_{13}}^2 \rangle = \langle g_{2,1}, g_{2,1}, g_{2,2}, g_{2,2}, g_{3,2}, g_{2,3}, g_{2,4}, g_{2,3}, g_{2,4}, g_{2,3}, g_{2,4}, g_{3,4}, g_{3,4} \rangle$. Then, each redundant or abnormal grid cell of $gridTr_2$ is identified and processed. Figure 5(b) shows c_{m_2, n_2}^2 mapped by point p_2 is redundant grid cell and is removed from $gridTr_2$, so the point p_2 is drawn with dotted lines. Then, c_{m_3, n_3}^2 and c_{m_4, n_4}^2 are not identified to be either redundant or abnormal cells, so they are retained in $gridTr_2$. However, as shown in Figure 5(c), c_{m_5, n_5}^2 mapped by point p_5 is not adjacent to grid cells c_{m_4, n_4}^2 and c_{m_6, n_6}^2 . Figure 5(d) shows grid trajectory $gridTr_2 = \langle g_{2,1}, g_{2,2}, g_{2,3}, g_{2,4}, g_{2,3}, g_{2,4}, g_{2,3}, g_{2,4}, g_{3,4} \rangle$ by removing the abnormal grid cell c_{m_5, n_5}^2 and redundant grid cells c_{m_2, n_2}^2 , c_{m_6, n_6}^2 , and $c_{m_{13}, n_{13}}^2$ from original $gridTr_2$. In Figure 5(d), the points drawn with dotted lines are redundant, and the points painted with red are abnormal.

However, the subsequence $\langle c_{m_4, n_4}^2, c_{m_6, n_6}^2, c_{m_7, n_7}^2, c_{m_8, n_8}^2, c_{m_9, n_9}^2, c_{m_{10}, n_{10}}^2 \rangle = \langle g_{2,3}, g_{2,4}, g_{2,3}, g_{2,4}, g_{2,3}, g_{2,4} \rangle$ in $gridTr_2$ moves repeatedly two adjacent cells $g_{2,3}$ and $g_{2,4}$. Therefore, there exist a few stranded cells in this subsequence. As shown in Figure 6(a), only the first and last grid cells in the subsequence are retained, and the other grid cells of continuous repetition are deleted. In Figure 6(a), the points enclosed by the larger ellipse drawn with dotted lines are the removed stranded points, and all deleted points are drawn with dotted lines. Figure 6(b) shows the grid trajectory $gridTr_2 = \langle g_{2,1}, g_{2,2}, g_{2,3}, g_{2,4}, g_{3,4} \rangle$ after processing, the grid trajectory in Figure 6(b) is shown in Figure 6(c) with the background of the highlighted grid cells, and the

FIGURE 5: Identification and processing of redundant cells and abnormal cells. (a) The trajectory Tr_2 is mapped to the grid trajectory. (b) Identification of the redundant cell. (c) The adjacent grid cells. (d) Identification and processing of the abnormal cell.

sequence of the highlighted grid cells covering $gridTr_2$ is $\langle g_{2,1}, g_{2,2}, g_{2,3}, g_{2,4}, g_{3,4} \rangle$.

After the trajectory $Tr_i = \langle p_1, p_2, \dots, p_h \rangle$ in TS is mapped and abnormal, redundant, and stranded grid cells in Tr_i are removed, the final trajectory is denoted by $gridTr_i = \langle c_{m_1, n_1}^i, c_{m_2, n_2}^i, \dots, c_{m_l, n_l}^i \rangle$, where $l \leq h$.

The pseudocode of the grid trajectory serialization ($gridTrS$) algorithm is presented in Algorithm 1, in which lines 2-5 add grid cells mapped by all points of Tr_k in TS to sg_k , lines 6-10 remove the redundant and abnormal grid cells from sg_k , lines 11-19 remove stranded grid cells and form the eventual grid trajectory $gridTr_k$, and line 20 adds $gridTr_k$ to the set of grid trajectories GTS .

4.2. Overall Clustering Based on Grid Trajectory. As one of the most classic clustering algorithms, K -means is widely used in the trajectory field on account of its simplicity and rapidity, and this algorithm can be completed quickly. However, the algorithm usually uses the trajectory point as the basic unit, so it is not suitable for the whole trajectory clustering. On account of this, this paper proposes an overall clustering algorithm based on grid trajectories with K -means as the core. The algorithm is mainly divided into the formation of initial cluster centers based on density weights and the adjustment and update of clusters based on the grid trajectory.

4.2.1. Formation of Initial Cluster Centers Based on Density Weights. The formation of initial clustering centers based on density weights is to extract the initial cluster centers according to the generated grid trajectory set GTS . The clustering centers of original K -means algorithm are usually selected based on random algorithms. Although it is easy to understand and implement, the random selection of initial clustering mode may result in the clustering results not easy to converge and inconsistent. Therefore, this paper proposes an algorithm to select initial cluster centers ($SICC$). In this algorithm, the distance and density weight concept are introduced to evaluate the probability of grid trajectories becoming cluster centers. Specifically, given a set of grid trajectories $GTS = \{gridTr_1, gridTr_2, \dots, gridTr_p\}$ and the number of clusters k , $SICC$ first calculates the density $Den(gridTr_i)$ of each grid trajectory $gridTr_i$ in the grid cell space, and the trajectory with the maximum density is selected as the first initial clustering center. The density $Den(gridTr_i)$ is specifically defined as

$$Den(gridTr_i) = \frac{Den(c_{m_1, n_1}^i) + Den(c_{m_2, n_2}^i) + \dots + Den(c_{m_l, n_l}^i)}{l}, \quad (5)$$

where l is the length of grid trajectory $gridTr_i$ and $Den(c_{m_l, n_l}^i)$

FIGURE 6: Processing of stranded cells and trajectories mapping based on grid estimation. (a) Identification and processing of stranded cells. (b) Trajectories mapping based on grid estimation. (c) The highlighted grid trajectory.

```

Input:  $TS = \{Tr_1, Tr_2, \dots, Tr_p\}$ 
Output:  $GTS = \{gridTr_1, gridTr_2, \dots, gridTr_p\}$ 
1: for each trajectory  $Tr_k$  in  $TS$  ( $1 \leq k \leq p$ ) do
2:    $sg_k = \phi$ ;
3:    $n = \text{sizeof}(Tr_k)$ ;
4:   for each GPS point  $p_l$  in  $Tr_k$  ( $1 \leq l \leq n$ ) do
5:     Add  $c_{m_l, n_l}^k$  (grid cell mapped by  $p_l$ ) to  $sg_k$ ;
6:   for each grid cell  $c_{m_l, n_l}^k$  in  $sg_k$  ( $2 \leq l \leq n$ ) do
7:     if  $c_{m_l, n_l}^k = c_{m_{l-1}, n_{l-1}}^k$  then
8:       Remove  $c_{m_l, n_l}^k$  from  $sg_k$ ;
9:     if  $c_{m_l, n_l}^k \notin \text{Adja}(c_{m_{l-1}, n_{l-1}}^k) \ \& \ c_{m_l, n_l}^k \notin \text{Adja}(c_{m_{l+1}, n_{l+1}}^k)$  then
10:      Remove  $c_{m_l, n_l}^k$  from  $sg_k$ ;
11:    $gridTr_k = \phi$ ;
12:   Add  $c_{m_1, n_1}^k$  in  $sg_k$  to  $gridTr_k$ ;
13:   for each grid cell  $c_{m_l, n_l}^k$  in  $sg_k$  ( $2 \leq l \leq \text{sizeof}(sg_k)$ ) do
14:      $s = \text{sizeof}(gridTr_k)$ ;
15:     if  $s = 1$  then
16:       if  $(c_{m_l, n_l}^k \text{ in } sg_k) \neq (c_{m_s, n_s}^k \text{ in } gridTr_k)$  then
17:         Add  $(c_{m_l, n_l}^k \text{ in } sg_k)$  to  $gridTr_k$ ;
18:     if  $s \neq 1$  then
19:       if  $(c_{m_l, n_l}^k \text{ in } sg_k) \neq (c_{m_{s-1}, n_{s-1}}^k \text{ in } gridTr_k) \ \& \ (c_{m_l, n_l}^k \text{ in } sg_k) \neq (c_{m_s, n_s}^k \text{ in } gridTr_k)$  then
20:         Add  $(c_{m_l, n_l}^k \text{ in } sg_k)$  to  $gridTr_k$ ;
21:   Add  $gridTr_k$  to  $GTS$ ;
22: return  $GTS$ 

```

ALGORITHM 1: $gridTrS$.

Input: $GTS = \{gridTr_1, gridTr_2, \dots, gridTr_p\}$

Output: $Tg_centers_{ini}$

```

1:  $Tg\_num \leftarrow p$ ;  $Tg\_dens = \phi$ ;  $Tg\_centers = \phi$ ;
2: for  $i \leftarrow 1$  to  $Tg\_num$  do
3:   Calculate the density of each grid trajectory:  $CalculateTg\_dens(i)$ ;
4:   sort the vector  $Tg\_dens$  in descending order;
5:    $Tg\_centers_{ini}(1) \leftarrow Tg\_dens(1)$ ;
6:   for  $i \leftarrow 1$  to  $Tg\_num$  do
7:     For each grid trajectory in  $GTS$ , using  $EDG$  to calculate the distance between it and the selected cluster center:  $d_i \leftarrow EDG(Tg\_centers_{ini}(selected), gridTr_i)$ ;
8:     To sum up all  $d_i$ :  $s_{dis} \leftarrow sum(d)$ ;  $s_{den} \leftarrow sum(Tg\_dens)$ ;
9:      $random \leftarrow (s_{dis}/4) + (s_{den}/4)$ ;
10:    while  $random > 0$  do
11:       $random \leftarrow random - d_i - Tg\_dens(i)$ ;  $// i \in [1, p]$ 
12:       $Tg\_centers_{ini}(q) \leftarrow i$ ;  $// q \in [2, k]$ 
13:    return  $Tg\_centers_{ini}$ 

```

ALGORITHM 2: SICCC.

is the density of the grid cell c_{m_1, n_1}^i , that is, the number of trajectory points on all trajectories contained in the cell.

Secondly, to follow the principle that the distance between the initial cluster centers is as far as possible and the grid trajectory density of the cluster centers is as large as possible, the distance between the grid trajectories and the cluster centers should be calculated. Since the classical Euclidean distance cannot measure the distance between grid trajectories with different lengths, a new method based on edit distance ([38, 39]) is proposed to measure the shape similarity between grid trajectories, referred to as EDG .

The following concepts are introduced before defining this method.

The insertion cost of grid cells: given two trajectory grids $gridTr_i = \langle c_{m_1, n_1}^i, c_{m_2, n_2}^i, \dots, c_{m_k, n_k}^i, \dots, c_{m_q, n_q}^i \rangle$ and $gridTr_j = \langle c_{m_1, n_1}^j, c_{m_2, n_2}^j, \dots, c_{m_h, n_h}^j, \dots, c_{m_e, n_e}^j \rangle$, the insertion cost is to insert a grid cell c_{m_h, n_h}^j in $gridTr_j$ into the grid sequence of $gridTr_i$. The cost of the insertion operation is defined as the Euclidean distance between the grid cell c_{m_h, n_h}^j in $gridTr_j$ and the grid cell c_{m_k, n_k}^i being compared in $gridTr_i$. It is denoted as

$$insert(c_{m_k, n_k}^i, c_{m_h, n_h}^j) = \begin{cases} 0, & isMerging(c_{m_k, n_k}^i, c_{m_h, n_h}^j) = \text{true} \& k \geq 1, \\ 1, & isMerging(c_{m_k, n_k}^i, c_{m_h, n_h}^j) = \text{false} \& k=1 \& h=e, \\ \left\| c_{m_h, n_h}^j \cdot gid - c_{m_{h+1}, n_{h+1}}^j \cdot gid \right\|, & isMerging(c_{m_k, n_k}^i, c_{m_h, n_h}^j) = \text{false} \& k=1 \& h > 1, \\ \left\| c_{m_h, n_h}^j \cdot gid - c_{m_k, n_k}^i \cdot gid \right\|, & isMerging(c_{m_k, n_k}^i, c_{m_h, n_h}^j) = \text{false} \& k \geq 1, \end{cases} \quad (6)$$

where e is the length of grid trajectory $gridTr_j$, $c_{m_h, n_h}^j \cdot gid$ represent the row and column numbers of c_{m_h, n_h}^j in S_G , and $\|\cdot\|$ is the Euclidean distance between them. In this paper, it is transformed into the sum of the absolute value of the difference of each element to improve the efficiency. For example, $\|c_{m_h, n_h}^j \cdot gid - c_{m_k, n_k}^i \cdot gid\| = |(m_h - m_k)| + |(n_h - n_k)|$. In fact, the Euclidean distance between the grid cells of different grid

trajectories is calculated as the insertion cost, and the grid trajectories in different grid cells have certain distance by default, so the grid trajectories are not considered which are still in the adjacent interval and in the adjacent grid cells. If the case in Figure 7(a) occurs, the distance between the two grid trajectories is very close when transforming $gridTr_u$ to $gridTr_v$. However, there will be a large error in the calculation results due to the high operation costs in different grid cells.

FIGURE 7: Merging of grid cells. (a) The grid trajectories in the adjacent grid cells. (b) The condition of merging.

To solve this problem, $isMerging(c_{m_k,n_k}^i, c_{m_h,n_h}^j)$ is introduced to determine whether the grid cells c_{m_k,n_k}^i and c_{m_h,n_h}^j can be merged; the specific calculation formula is as follows:

$$isMerging(c_{m_k,n_k}^i, c_{m_h,n_h}^j) = \begin{cases} \text{true,} & c_{m_h,n_h}^j \in Adja(c_{m_k,n_k}^i) \& D(s_i, s_j) = \frac{\sum_{a=0}^r d_a^i + \sum_{b=0}^t d_b^j}{r + t/2} \leq \text{gridsize,} \\ \text{false,} & \text{other,} \end{cases} \quad (7)$$

where c_{m_k,n_k}^i ($1 \leq k \leq q$) and c_{m_h,n_h}^j ($1 \leq h \leq e$) are two grid cells located on $gridTr_i$ and $gridTr_j$ and q and e are the lengths of grid trajectory $gridTr_i$ and grid trajectory $gridTr_j$, respectively. While subsequence $s_i = \langle p_{i_1}, \dots, p_{i_r} \rangle$ in trajectory Tr_i is contained in the grid cell c_{m_k,n_k}^i , where r is the number of points in s_i , and subsequence $s_j = \langle p_{j_1}, \dots, p_{j_t} \rangle$ in trajectory Tr_j is contained in the grid cell c_{m_h,n_h}^j , similarly, t is the number of points in s_j . $\sum_{a=0}^r d_a^i$ and $\sum_{b=0}^t d_b^j$ are the sum of the vertical Euclidean distances from each location point on subsequences s_i and s_j to the coincident boundaries of grid cell c_{m_k,n_k}^i and c_{m_h,n_h}^j . If the value of distance $D(s_i, s_j)$ exceeds

gridsize, i.e., the size of the grid cell, the calculation will be terminated. Subsequences s_i and s_j can be regarded as located in one grid cell if the grid cells can be merged, as shown in Figure 7(b).

The replacement cost of grid cells: given two trajectory grids $gridTr_i = \langle c_{m_1,n_1}^i, c_{m_2,n_2}^i, \dots, c_{m_k,n_k}^i, \dots, c_{m_q,n_q}^i \rangle$ and $gridTr_j = \langle c_{m_1,n_1}^j, c_{m_2,n_2}^j, \dots, c_{m_h,n_h}^j, \dots, c_{m_e,n_e}^j \rangle$, the replacement cost is to transform a grid cell c_{m_k,n_k}^i specified in $gridTr_i$ to a grid cell c_{m_h,n_h}^j in $gridTr_j$. The cost of the replacement operation is defined as the Euclidean distance between c_{m_k,n_k}^i and c_{m_h,n_h}^j . It is denoted as

$$replace(c_{m_k,n_k}^i, c_{m_h,n_h}^j) = \begin{cases} 0, & isMerging(c_{m_k,n_k}^i, c_{m_h,n_h}^j) = \text{true,} \\ \|c_{m_h,n_h}^j \cdot gid - c_{m_k,n_k}^i \cdot gid\|, & isMerging(c_{m_k,n_k}^i, c_{m_h,n_h}^j) = \text{false,} \end{cases} \quad (8)$$

where the replacement cost between grid cells c_{m_k,n_k}^i and c_{m_h,n_h}^j is 0 if the grid cells merging condition in Equation (7) is satisfied.

The deletion cost of grid cells: given two trajectory grids $gridTr_i = \langle c_{m_1,n_1}^i, c_{m_2,n_2}^i, \dots, c_{m_k,n_k}^i, \dots, c_{m_q,n_q}^i \rangle$, the deletion cost

is to delete a grid cell specified in $gridTr_i$. The cost of the deletion operation is defined as the Euclidean distance between the current grid cell c_{m_k,n_k}^i to be deleted and the next uncompleted cell $c_{m_{k+1},n_{k+1}}^i$ in the grid sequence of $gridTr_i$. It is denoted as

FIGURE 8: Transformation between grid trajectories. (a) The insertion of grid cell. (b) The replacement of grid cells. (c) The deletion of grid cell.

$$\text{delete}(c_{m_k, n_k}^i) = \begin{cases} 1, & k = q, \\ \left\| c_{m_k, n_k}^i \cdot \text{gid} - c_{m_{k+1}, n_{k+1}}^i \cdot \text{gid} \right\|, & 1 < k \leq q. \end{cases} \quad (9)$$

Figure 8(a) shows the grid trajectory $\text{gridTr}_1 = \langle c_{m_1, n_1}^1, c_{m_2, n_2}^1, \dots, c_{m_6, n_6}^1 \rangle = \langle g_{5,2}, g_{4,2}, g_{3,2}, g_{2,2}, g_{1,2}, g_{1,1} \rangle$ and $\text{gridTr}_2 = \langle c_{m_1, n_1}^2, c_{m_2, n_2}^2, \dots, c_{m_6, n_6}^2 \rangle = \langle g_{5,4}, g_{4,4}, g_{3,4}, g_{3,5}, g_{2,5}, g_{1,5} \rangle$. Suppose c_{m_4, n_4}^1 in gridTr_1 and c_{m_3, n_3}^2 in gridTr_2 are compared. If the grid cell c_{m_3, n_3}^2 is inserted in front of c_{m_4, n_4}^1 in

gridTr_1 , the insertion cost is $|(3-2)| + |(4-2)| = 3$. Figure 8(b) shows that the grid cell c_{m_4, n_4}^1 in gridTr_1 is replaced by c_{m_3, n_3}^2 , and the cost of the replacement operation is $|(3-2)| + |(4-2)| = 3$. Figure 8(c) shows that c_{m_4, n_4}^1 is removed from the grid cell sequence of gridTr_1 , and the deletion cost is $|(2-1)| + |(2-2)| = 1$.

To sum up, the edit distance from grid trajectory gridTr_i to gridTr_j is the sum of the operation costs of insertion, replacement, and deletion; $\text{EDG}(\text{gridTr}_i, \text{gridTr}_j)$ is defined as

$$\text{EDG}(\text{gridTr}_i, \text{gridTr}_j) = \begin{cases} \sum_{h=1}^e \text{insert}(c_{m_i, n_i}^i, c_{m_h, n_h}^j), & q = 0, \\ \sum_{k=1}^q \text{delete}(c_{m_k, n_k}^i), & e = 0, \\ \min \left\{ \text{EDG}(\text{Rest}(\text{gridTr}_i), \text{Rest}(\text{gridTr}_j)) + \text{replace}(c_{m_i, n_i}^i, c_{m_h, n_h}^j), \text{EDG}(\text{gridTr}_i, \text{Rest}(\text{gridTr}_j)) + \text{insert}(c_{m_i, n_i}^i, c_{m_h, n_h}^j), \text{EDG}(\text{Rest}(\text{gridTr}_i), \text{gridTr}_j) + \text{delete}(c_{m_i, n_i}^i) \right\}, & \text{otherwise,} \end{cases} \quad (10)$$

where c_{m_k, n_k}^i ($1 \leq k \leq q$) is the k th grid cell in the trajectory sequence of gridTr_i and c_{m_h, n_h}^j is the h th grid cell in the trajectory sequence of gridTr_j . $\text{Rest}(\text{gridTr}_i)$ is defined as the grid cells other than the compared grid cells in gridTr_i , similarly, $\text{Rest}(\text{gridTr}_j)$ is defined as the grid cells other than the compared grid cells in gridTr_j . From formula (10), it can be concluded that the higher the value of edit distance

between two grid trajectories, the more dissimilar they are, otherwise, the higher the degree of similarity.

Finally, based on the first initial cluster center that has been determined, $k-1$ centers of grid trajectory cluster with larger distance and higher density are selected. A random value *random* is set to fuse the density and distance of grid trajectories, denoted as

$$\text{random} = \begin{cases} \frac{s_{\text{den}} + s_{\text{dis}}}{b}, & (a) \\ \text{random} - \text{EDG}(\text{Tg_centers}_{\text{ini}}(\text{selected}), \text{gridTr}_i) - D(\text{gridTr}_i) i \in [1, p] \& i! = m, & (b) \end{cases} \quad (11)$$

Input: $gridTr_1 = \langle c_{m_1, n_1}^1, c_{m_2, n_2}^1, \dots, c_{m_N, n_N}^1 \rangle$, $gridTr_2 = \langle c_{m_1, n_1}^2, c_{m_2, n_2}^2, \dots, c_{m_M, n_M}^2 \rangle$
Output: $EDG(gridTr_1, gridTr_2)$

- 1: Initialize the zeroth row and column of the $N + 1$ rows and $M + 1$ columns distance matrix D : $D[0, 0] \leftarrow 0$;
- 2: **for** each row $i \leftarrow 1$ to N **do**
- 3: $D[i, 0] \leftarrow D[i - 1, 0] + delete(c_{m_i, n_i}^1)$;
- 4: **for** each column $j \leftarrow 1$ to M **do**
- 5: $D[0, j] \leftarrow D[0, j - 1] + insert(c_{m_0, n_0}^1, c_{m_j, n_j}^2)$;
- 6: **for** each row $i \leftarrow 1$ to N **do**
- 7: **for** each column $j \leftarrow 1$ to M **do**
- 8: $D[i, j] = \min \{D[i - 1, j - 1] + replace(c_{m_i, n_i}^1, c_{m_j, n_j}^2), D[i, j - 1] + insert(c_{m_i, n_i}^1, c_{m_j, n_j}^2), D[i - 1, j] + delete(c_{m_i, n_i}^1)\}$;
- 9: **return** $D[N, M]$

ALGORITHM 3: EDG.

Input: $GTS = \{gridTr_1, gridTr_2, \dots, gridTr_p\}$, Predefined number of trajectory clusters k , The initial clustering centers $Tg_centers_{ini}$
Output: $TgClusters, TgAbnormal$

- 1: Initialize the parameter k ;
- 2: $iter \leftarrow 0$;
- 3: $Tg_Centers \leftarrow Tg_centers_{ini}$;
- 4: **while** $Tg_centerschanged \mid iter \leq 10$ **do**
- 5: $iter \leftarrow iter + 1$; $TgClusters = \phi$; $TgAbnormal = \phi$;
- 6: **for** $i \leftarrow 1$ to p **do**
- 7: **if** $i \notin Tg_centers$ **then**
- 8: $d_i \leftarrow \min (EDG(Tg_Centers, gridTr_i))$ and record the index q of $Tg_Centers$;
- 9: **if** $R(gridTr_i, TgClusters\{q\}) \geq 0$ **then**
- 10: $TgClusters\{q\} \leftarrow TgClusters\{q\} \cup \{i\}$;
- 11: **else**
- 12: $TgAbnormal \leftarrow TgAbnormal \cup \{i\}$;
- 13: **for** $h \leftarrow 1$ to k **do**
- 14: $numc \leftarrow length(TgClusters\{h\})$
- 15: **for** $i \leftarrow 1$ to $numc$ **do**
- 16: **for** $j \leftarrow 1$ to $numc$ **do**
- 17: $dsum(i) \leftarrow dsum(i) + dist(i, j)$;
- 18: **if** $dsum(i) < \min \{dsumin\}$ **then**
- 19: $dsumin \leftarrow dsum(i)$;
- 20: $ClustID \leftarrow i$;
- 21: $Tg_centers(h) \leftarrow ClustID$;
- 22: **return** $TgClusters, TgAbnormal$

ALGORITHM 4: GBWTC.

where s_{den} is the sum of the density of the grid trajectories other than clustering centers. $EDG(Tg_centers_{ini}(selected), gridTr_i)$ is the distance between the grid trajectory just selected as the clustering center $Tg_centers_{ini}(selected)$ and the grid trajectory of non clustering centers $gridTr_i$, and s_{dis} is the sum of these distances, that is, $s_{dis} = sum(EDG(Tg_centers_{ini}(selected), gridTr_i))$. m is the index of the grid trajectory just selected as the clustering center in GTS . In formula (11)(a), the value of $random$ is initialized, and after repeated experiments, the value of b is set as 4; after assigning the initial value to $random$, the formula (11)(b) is executed until the value of $random$ is less than 0, and the grid trajectory is the next selected cluster center at this time.

The above steps are repeated until all initial cluster centers are selected.

4.2.2. The Adjustment and Update of Clusters Based on the Grid Trajectory. The adjustment and update of clusters based on the grid trajectory are a process of dynamically allocating the optimal clusters of grid trajectories according to the nearest neighbor principle. After determining the cluster centers, the traditional K -means method divides each trajectory into the cluster which is closest to the trajectory, but there is no determination of the abnormal trajectories in the trajectory dataset in the iteration process. For this problem, an outlier function $R(gridTr_i, C_a)$ is introduced

to measure the influence of grid trajectory on other trajectories in cluster and determine whether the grid trajectory is abnormal. If it is not abnormal, it will be added to the cluster;

if it is, the grid trajectory is added to the abnormal trajectory set. The outlier function is specifically defined as

$$R(gridTr_i, C_a) = \begin{cases} d_{c_min} - EDG(gridTr_i, gridTr_c), & |C_a| = 1, \\ \left(\frac{1}{|C_a| - 1} \sum_{gridTr_j \in C_a} EDG(gridTr_j, gridTr_c) \right) - EDG(gridTr_i, gridTr_c), & |C_a| > 1, \end{cases} \quad (12)$$

where d_{c_min} is the minimum distance between cluster centers. C_a represents the cluster with the nearest cluster center to the grid trajectory $gridTr_i$, and $gridTr_c$ is the center of the cluster C_a . $|C_a|$ is the number of grid trajectories contained in cluster C_a .

Specifically, the outlier function first calculates the distance between the grid trajectory and the cluster center and finds the nearest cluster center. Then, based on the average distance between all the other grid trajectories in the cluster and the cluster center, the influence of the trajectory on the existing structure of the cluster is numerically calculated. As shown in Equation (12), generally, the smaller the value of $R(gridTr_i, C_a)$ is, the smaller the influence of judging the grid trajectory as abnormal. If the value of $R(gridTr_i, C_a)$ is less than 0, the grid trajectory will be marked as abnormal.

The pseudocode of the grid-based whole trajectory clustering GBWTC is presented in Algorithm 4. There are mainly two steps in the algorithm. Firstly, the related variables are initialized (lines 1-2), and the initial clustering centers determined by Algorithm 2 are assigned to the clustering centers (line 3). Then, the clusters are adjusted and updated based on the iterative process, and the clustering centers are updated after each iteration. If the conditions are met, the clustering process will stop, and the trajectory clusters and abnormal trajectory set will be output (lines 4-22). Specifically, according to the outlier function, it is determined whether each trajectory can be clustered into a cluster or temporarily as an exception (lines 4-12). The new trajectory centers are calculated according to the clustering results. The clustering process is terminated until the end condition of the iterative process is satisfied (lines 14-21). Finally, the algorithm GBWTC outputs a set of trajectory clusters TgClusters and a set of abnormal trajectories TgAbnormal (lines 22).

5. Experimental Evaluation

5.1. Experimental Data. We use two real-world datasets as experimental data to verify the efficiency and accuracy of the algorithm: (i) travel records of 536 taxis in San Francisco in more than 30 days that include the longitude and latitude of the vehicles, vehicle ID, time stamp, and whether they carry passengers or not. In this paper, we extract the data of a car in a day according to the time stamp, including

11943 trajectories and filter the longitude and latitude that do not belong to San Francisco city. Finally, we select the longitude and latitude, sampling time, and other attributes to participate in the experiment; (ii) a month's driving data of 320 taxis in Rome City that includes the longitude and latitude of the vehicle, vehicle ID, time stamp, and other information. The dataset is also preprocessed according to the above method, and a total of 7356 trajectories are obtained eventually. Table 1 and Figure 9 show the statistical information of the above two trajectory datasets and the road network composed of the trajectory.

Figure 10 plots the initial distribution of users by average travel time per day in San Francisco and Roman. It can be observed that 89.23% and 82.79% of users have more than 6 hours of travel time in a day in the two datasets, which provides abundant trajectory data for the experiment. There are differences in more subdivided periods, as shown in Figures 10(a) and 10(b), and 39.34% of users travel 6-9 hours a day in the San Francisco dataset, while 68.56% users travel 6-9 hours a day in the Roman dataset. The existence of difference increases the diversity and persuasiveness of experimental results. Further, we plot the mean and standard deviation of the number of users each day per week in the San Francisco and Roman datasets as showed in Figure 11. It shows the stability of the San Francisco data and the Rome data over time. The box plots of normalized longitude and latitude of all moving trajectories of users each week in a month in two datasets are shown in Figure 12. It can be observed that the movement of users is basically in the same region, and there are some differences in the range of motion. Therefore, the data are valuable and the results are representative.

5.2. Raw Algorithms in Comparison. GridCSD-TraceMob firstly executes the trajectory mapping algorithm to calculate the distance between trajectories iteratively. Each iteration is divided into three steps: (1) the two trajectories with the largest distance are selected as the initial pivots, (2) each trajectory is mapped to d -dimensional metric space, and (3) the Euclidean distances between trajectories in the space are calculated. After the iteration, the classical K -means algorithm is used for clustering.

TRACUS partitions a trajectory into a series of subtrajectories and performs DBSCAN to group similar subtrajectories together. The algorithm determines whether each

TABLE 1: Dataset statistics.

Regions	Memory of dataset	Acquisition time	Number of trajectories	Number of GPS points	Sampling interval
San Francisco Bay Area	411 MB	2008.5.17–6.10	11943	12787048	10 s
Roman	1.49 GB	2014.2.1–3.2	7356	21817852	7 s

FIGURE 9: The distribution of real datasets. (a) San Francisco Bay Area. (b) Roman.

FIGURE 10: Initial distribution of users by average travel time. (a) San Francisco datasets. (b) Roman datasets.

point of the trajectory meets the segmentation conditions based on the minimum description principle. In the DBSCAN phase, the iteration is executed for n_{sub} times, which is the number of subtrajectories. In each iteration, the subtrajectory in the cluster is performed two steps: (1) ϵ -neighborhood query and (2) cluster expansion performs a linear scanning for each selected subtrajectory's neighbors.

K-means is a classical clustering method, in which clusters are groups of elements characterized by a small distance to the cluster center. The general process of K-means is to assign each element to the nearest cluster center and update

the cluster centers. This process is repeated until a convergence condition is satisfied.

5.3. Parameter Settings. The parameter *gridsize* is as follows. To determine the sequence of trajectory grid cells, the size of grid cell *gridsize* needs to be provided to grid cell space. Users may provide their desirable parameters or use the suggested parameters. Generally, most of the dense road sections are concentrated in the city center in the whole dataset of road network, while the distribution of trajectory in some suburban or marginal areas is relatively sparse. Hence, the

FIGURE 11: The average number of users each day per week.

FIGURE 12: The box plots of the longitude and latitude distribution. (a) San Francisco datasets. (b) Roman datasets.

size of grid cell in grid cell space should not be set too large, to avoid the loss of vehicle driving conditions in dense road sections. But it should not also be set too small; otherwise, the efficiency of the subsequent trajectory clustering stage will be reduced. In our experiments, the grid cell size is set to be 0.1 by default.

In our experiments, to identify the optimal grid cell size, we run GBWTC and GridCSD-TraceMob algorithms with different grid cell sizes ranging from 0.025 to 0.2 at an interval of 0.025 on the Roman dataset and San Francisco dataset. Figure 13 shows the performance comparison of these two clustering algorithms in terms of SI under given trajectory number and cluster number on the datasets, respectively. As shown in Figures 13(a) and 13(b), the SI index of both decreases gradually with the increase of *gridsize*. The reason is that the trajectories located in different road segments or far away from each other will be divided into the same grid

when *gridsize* is set too large, which will cause the error of trajectory similarity measurement. Under the same *gridsize*, even when *gridsize* is larger, the SI index obtained by GBWTC is higher in two different road network datasets, which indicates that the trajectory clustering results obtained by GBWTC have higher similarity in the same cluster, and the separation degree of trajectories between different clusters is also higher, that is, compared with GridCSD-Tracemob, GBWTC has a better overall clustering quality.

However, the efficiency of the algorithm affected by *gridsize* cannot be taken into account only by the SI index, and the optimal parameters cannot be determined. We further compare GBWTC with GridCSD-TraceMob in terms of running time with different *gridsize* ranging from 0.025 to 0.2 at an interval of 0.025. We plot the mean and standard deviation of the algorithm running time across 8 grid cell spaces with different sizes in Figure 14. As illustrated in

FIGURE 13: Comparison of clustering quality using SI with different *gridsize*. (a) San Francisco datasets (2400 trajectories, $k = 15$). (b) Roman datasets (2000 trajectories, $k = 15$).

FIGURE 14: Comparison of algorithm running time with different *gridsize*. (a) San Francisco datasets (2400 trajectories, $k = 15$). (b) Roman datasets (2000 trajectories, $k = 15$).

Figures 14(a) and 14(b), the running time presents an overall downward trend with the increase of *gridsize*. The larger the *gridsize*, the shorter the running time. It is obvious that the running time of GBWTC is shorter and smoother than that of GridCSD-TraceMob algorithm, and the standard deviation of running time is smaller. Moreover, we observe that GBWTC is approximately four times faster than GridCSD-TraceMob on average.

It can be observed from Figures 13 and 14 that the SI index is higher and the running time is faster than other smaller grid cell sizes when *gridsize* is 0.1, which is a more suitable *gridsize* parameter. Therefore, the *gridsize* is set as 0.1 of GBWTC and GridCSD-TraceMob in the other experiments of this paper.

The parameter k is as follows. To better divide the trajectories, the number of clusters k needs to be provided. In our work, we run GBWTC, GridCSD-TraceMob, and K-means algorithms with different numbers of clusters ranging from 2 to 15 at an interval of 1. Since TraClus is a density-based clustering algorithm, it does not participate in the comparison. Figure 15 plots the mean and standard deviation of SI index of the three algorithms across 14 different numbers of clusters in the San Francisco and Roman datasets. For these two datasets, no matter how the value of k changes, GBWTC shows a stronger clustering effect than the other two algorithms, which shows the effectiveness of our improvement in the distance measurement and the steps in the

FIGURE 15: Comparison of clustering quality using SI with different numbers of clusters. (a) San Francisco datasets (2400 trajectories, $gridsize = 0.1$). (b) Roman datasets (2000 trajectories, $gridsize = 0.1$).

FIGURE 16: Comparison of clustering quality using SI with different numbers of trajectories. (a) San Francisco datasets ($k = 15$, $gridsize = 0.1$). (b) Roman datasets ($k = 15$, $gridsize = 0.1$).

clustering process to some extent. Since three algorithms have good clustering effect with 15 clusters in the two datasets, the number of clusters k of GBWTC, GridCSD-TraceMob, and K-means is set by 15 in all other experiments of this paper.

5.4. Performance Evaluation. We conduct a simulation-based evaluation of our proposed grid-based whole trajectory clustering model in terms of clustering quality and runtime performance in comparison with existing approaches. We implement all of these algorithms in IntelliJ IDEA 2018.3.5 (64-bit), and all experiments are conducted on a Windows PC workstation equipped with Intel(R) Core(TM) i5-5200U CPU@2.20 GHz and 4 GB of memory.

We evaluate the clustering quality of GBWTC in comparison with GridCSD-TraceMob, TraClus, and K-means in terms of Silhouette Coefficient, which are representatives of whole trajectory-based, subtrajectory-based, and point-based clustering. The value of Silhouette Coefficients is proportional to the clustering quality, and the closer to 1, the better the clustering quality is. We run these four algorithms, GBWTC, GridCSD-TraceMob, TraClus, and K-Means, with different numbers of trajectories ranging from 400 to 3200 at an interval of 400. For each number of trajectories, we generate 8 random trajectory datasets. Figure 16 plots the mean and standard deviation of the SI index across 8 different datasets for each number of trajectories in San Francisco and Roman datasets. It can be observed that the SI index

FIGURE 17: Comparison of algorithm running time with different numbers of trajectories. (a) San Francisco datasets ($k = 15$, $gridsize = 0.1$). (b) Roman datasets ($k = 15$, $gridsize = 0.1$).

TABLE 2: Comparison of running time.

Number of trajectories	20	40	60	80	100	120
Running time of Scheme 1 (s)	0.33	0.72	0.87	0.99	1.12	1.26
Running time of Scheme 2 (s)	37.27	68.62	87.71	122.76	157.29	184.95

of GBWTC algorithm is closer to 1. With the change of parameters, the SI index of GBWTC is higher than that of the other three clustering algorithms in most cases, that is to say, it shows good adaptability and effect in dealing with different number of trajectories.

Figure 17 compares the proposed clustering algorithm GBWTC with GridCSD-TraceMob, TraClus, and K-means in terms of running time. We run these four algorithms with different numbers of trajectories on the real-world San Francisco and Roman datasets, ranging from 400 to 3200 at an interval of 400 trajectories. Again, for each number of trajectories, we generate 8 random trajectory datasets. We plot the mean and standard deviation of the algorithm running time across 8 datasets for each number of trajectories in Figure 17. These results show that GBWTC runs significantly faster than the other three algorithms in comparison for both San Francisco and Roman datasets, and the change speed of GBWTC algorithm is more gentle. The superiority of GBWTC becomes more obvious as the number of trajectories increases. This is because the GBWTC algorithm eliminates some useless points in the dataset before clustering and optimizes the selection of the clustering center in the clustering process, so that the clustering process is easier to converge, and there is no need for secondary mapping in the clustering process.

In addition, there are two advantages in the implementation of the steps of deleting redundant, abnormal, and stranded cells in the trajectory grid serialization: one is to reduce the running time; the other is to reduce the interference of these cells on clustering results. Table 2 shows the

time comparison of whether to remove redundant, abnormal and stranded cells in San Francisco dataset using the GBWTC algorithm. The first scheme is to remove the redundant points, while the second scheme is not.

6. Conclusion

We proposed a novel grid-based whole trajectory clustering model, referred to as GBWTC, which leverages the mapping theory to form the simple and representative grid trajectory. The proposed approach has potential to determine not only a series of trajectory clusters but also some abnormal trajectories and GPS points. Extensive experiments demonstrated that GBWTC significantly improves the clustering quality over the existing methods. The proposed whole trajectory clustering approach has a wide range of applications in various traffic and location service systems, including vehicle path planning, urban planning, service recommendation, traffic navigation, logistics and distribution, and detection and prevention of abnormal events.

Data Availability

The San Francisco Bay Area Dataset analyzed during the current study is available in the Dataverse repository 10.15783/C7J010. The Roman Dataset during the current study is available in the Dataverse repository 10.15783/C7QC7M. These datasets were derived from the following public domain resources: <https://crawdad.org/epfl/mobility/20090224>, <https://crawdad.org/roma/taxi/20140717>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is sponsored by the Science and Technology Planning Project of Sichuan Province under Grant No. 2020YFG0054, and the Joint Funds of the Ministry of Education of China. We would also like to thank JiangAn Chen for his comments, which helped improve part of this research.

References

- [1] 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, 2020.
- [2] Q. Yu, Y. Luo, C. Chen, and X. Zheng, "Road congestion detection based on trajectory stay-place clustering," *International Journal of Geo-Information*, vol. 8, no. 6, p. 264, 2019.
- [3] H. Wang, Z. Xu, and S. Jia, "Cluster-indistinguishability: a practical differential privacy mechanism for trajectory clustering," *Intelligent Data Analysis*, vol. 21, no. 6, pp. 1305–1326, 2017.
- [4] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, pp. 226–231, 1996.
- [5] A. Belhadi, Y. Djenouri, D. Djenouri, T. Michalak, and J. C. W. Lin, "Deep learning versus traditional solutions for group trajectory outliers," *IEEE Transactions on Cybernetics*, pp. 1–12, 2020.
- [6] A. Belhadi, Y. Djenouri, G. Srivastava, A. Cano, and J. C. W. Lin, "Hybrid group anomaly detection for sequence data: application to trajectory data analytics," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.
- [7] Y. Djenouri, D. Djenouri, and J. C. W. Lin, "Trajectory outlier detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 15, no. 2, pp. 1–28, 2021.
- [8] J. Gil Lee and J. Han, "Trajectory clustering: a partition-and-group framework," in *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pp. 593–604, Beijing, China, 2007.
- [9] B. Han, L. Liu, and E. R. Omiecinski, "A systematic approach to clustering whole trajectories of mobile objects in road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 936–949, 2017.
- [10] J. Queen, "Some methods for the classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley: University of California Press, 1966.
- [11] Y. Yanagisawa, J. I. Akahani, and T. Satoh, *Shape-based similarity query for trajectory of mobile objects*, Springer, Berlin, Heidelberg, 2003.
- [12] D. Berndt and J. Clifford, *Using dynamic time warping to find patterns in time series*, pp. 359–370, Advances in Knowledge Discovery and Data Mining, 1994.
- [13] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th International Conference on Data Engineering*, San Jose, CA, USA, 2002.
- [14] B. Lin and J. Su, "Shapes based trajectory for moving objects," in *13th ACM International Workshop on Geographic Information Systems, ACM-GIS 2005*, Bremen, Germany, November 2005.
- [15] M. Gariel, A. N. Srivastava, and E. Feron, "Trajectory clustering and an application to airspace monitoring," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 1511–1524, 2011.
- [16] J. Qiu and R. Wang, "Inferring road maps from sparsely sampled gps traces," *Journal of Location Based Services*, vol. 10, pp. 111–124, 2016.
- [17] C. Lu, Q. Sun, X. Ji, L. Xu, B. Wen, and M. Cheng, "A method of vehicle trajectory points clustering based on kernel distance," 2020.
- [18] Y. U. Yan-Wei, J. Zhao-Fei, C. Lei, Z. Jin-Dong, L. Zhao-Wei, and L. Jing-Lei, "Fast density-based clustering algorithm for location big data," *Journal of Software*, vol. 29, no. 8, pp. 2470–2484, 2018.
- [19] L. Zheng, D. Xia, X. Zhao et al., "Spatial-temporal travel pattern mining using massive taxi trajectory data," *Physica A-statistical Mechanics Its Applications*, vol. 501, pp. 24–41, 2018.
- [20] Y. Mao, H. Zhong, H. Qi, P. Ping, and X. Li, "An adaptive trajectory clustering method based on grid and density in mobile pattern analysis," *Sensors*, vol. 17, no. 9, article 2013, 2017.
- [21] B. Han, L. Liu, and E. Omiecinski, "Road-network aware trajectory clustering: integrating locality, Flow, and density," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 416–429, 2015.
- [22] X. Niu, J. Zhu, C. Q. Wu, and S. Wang, "On a clustering-based mining approach for spatially and temporally integrated traffic sub-area division," *Engineering Applications of Artificial Intelligence*, vol. 96, article 103932, 2020.
- [23] D. Kumar, H. Wu, S. Rajasegarar, C. Leckie, S. Krishnaswamy, and M. Palaniswami, "Fast and scalable big data trajectory clustering for understanding urban mobility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 11, pp. 3709–3722, 2018.
- [24] L. Bermingham and I. Lee, "A general methodology for n -dimensional trajectory clustering," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7573–7581, 2015.
- [25] A. Salarpour and H. Khotanlou, "Direction-based similarity measure to trajectory clustering," *IET Signal Processing*, vol. 13, no. 1, pp. 70–76, 2019.
- [26] N. Guo, S. Shekhar, W. Xiong, L. Chen, and N. Jing, "Utsm: a trajectory similarity measure considering uncertainty based on an amended ellipse model," *International Journal of Geo-Information*, vol. 8, no. 11, p. 518, 2019.
- [27] F. Liu and Z. Zhang, "Adaptive density trajectory cluster based on time and space distance," *Physica A: Statistical Mechanics and its Applications*, vol. 484, pp. 41–56, 2017.
- [28] Q. Yu, Y. Luo, C. Chen, and S. Chen, "Trajectory similarity clustering based on multi-feature distance measurement," *Applied Intelligence*, vol. 49, no. 6, pp. 2315–2338, 2019.
- [29] J. Domingo-Ferrer and R. Trujillo-Rasua, "Microaggregation-and permutation-based anonymization of movement data," *Information Sciences*, vol. 208, pp. 55–80, 2012.
- [30] C. Wang, J. Yang, and J. P. Zhang, "Privacy preserving algorithm based on trajectory location and shape similarity," *Tongxin Xuebao/Journal on Communications*, vol. 36, 2015.

- [31] J. Pan, Q. Jiang, and Z. Shao, "Trajectory clustering by sampling and density," *Marine Technology Society Journal*, vol. 48, no. 6, pp. 74–85, 2014.
- [32] Y. Yang, J. Cai, H. Yang, J. Zhang, and X. Zhao, "Tad: a trajectory clustering algorithm based on spatial-temporal density analysis," *Expert Systems with Applications*, vol. 139, article 112846, 2020.
- [33] W. Wang, F. Xia, H. Nie et al., "Vehicle trajectory clustering based on dynamic representation learning of internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3567–3576, 2021.
- [34] A. Stefan, V. Athitsos, and G. Das, "The move-split-merge metric for time series," *IEEE Transactions on Knowledge Data Engineering*, vol. 25, no. 6, pp. 1425–1438, 2013.
- [35] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, USA, 2017.
- [36] N. R. Pal and J. Biswas, "Cluster validation using graph theoretic concepts," *Pattern Recognition*, vol. 30, no. 6, pp. 847–857, 1997.
- [37] R. C. D. Amorim and C. Hennig, "Recovering the number of clusters in data sets with noise features using feature rescaling factors," *Information Sciences*, vol. 324, pp. 126–145, 2015.
- [38] L. Kun and Y. Jie, "Trajectory distance metric based on edit distance," *Journal of Shanghai Jiaotong University*, vol. 43, pp. 1725–1729, 2009.
- [39] L. Chen, M. T. Ozsü, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, Baltimore, Maryland, USA, 2005.