IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 4, 2020, accepted February 12, 2020, date of publication February 17, 2020, date of current version February 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974521

Trajectory Outlier Detection on Trajectory Data Streams

KEYAN CAO[1,2], YEFAN LIU[1], GONGJIE MENG[1], HAOLI LIU[1], ANCHEN MIAO[1], AND JINGKE XU[1,2]

[1]College of Information and Control Engineering, Shenyang Jianzhu University, Shenyang 110168, China
[2]Liaoning Province Big data Management and Analysis Laboratory of Urban Construction, Shenyang Jianzhu University, Shenyang 110168, China

Corresponding author: Keyan Cao (caokeyan@sjzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602323, in part by the National Postdoctoral Foundation of China under Grant 2016M591455, in part by the Youth Seedling Foundation of Liaoning Province under Grant lnqn201913, and in part by the Natural Science Funds of Liaoning Province under Grant 2019MS264 and Grant 20180550019.

ABSTRACT The detection of abnormal moving on trajectory data streams is an important task in spatio-temporal data mining. An outlier trajectory is a trajectory grossly different from others, meaning there are few or even no trajectories following a similar route. In this paper, we propose a lightweight method to measure the outlier in trajectory data streams. Furthermore, we propose a basic algorithm (Trajectory Outlier Detection on trajectory data Streams-TODS), which can quickly determine the nature of the trajectory. Finally, we propose an Approximate algorithm (ATODS) to reduce the detection cost. It is space approximate algorithm which can effectively reduce the amount of calculation. The cost of ATODS algorithm can satisfy the demand of trajectory data streams. Our method are verified using both real data and synthetic data. The results show that they are able to reduce the running time without reducing the accuracy.

INDEX TERMS Outlier, trajectory stream, moving object.

I. INTRODUCTION

With the development of sensor network,GPS (Global Positioning System), wireless communication and other technologies, mobile intelligent devices with positioning function are widely used, generating a large number of trajectory data streams of moving objects, such as taxi track data streams, animal migration data streams, personal mobile data streams in large public places and so on. The trajectory data stream is continuously generated every day. For example, the vehicle trajectory is based on the collection of vehicle GPS, with an average daily data volume which varies from 10 million to 100 million, generating TB level data trajectory stream information every day. The mobile phone trajectory is based on the sampling of cellular base station, with an average daily data volume which varies from 1 billion to 10 billion, with the total data volume of TB or PB [1]. It can be seen that we have entered the era of trajectory data streams.

A lot of valuable time-effect information is hidden in the trajectory data streams, and researchers are urgently required to analyze it in time to find out the problems and solve them in time [2], [3]. Moving objects in public places tend to have similar moving trajectories, such as the flow of people in scenic spots or in stations, and behaviors that deviate from this trend are considered abnormal movements [4]. In terms of traffic, vehicles deviating from their normal trajectory may be caused by drivers' drunken driving or sudden physical discomfort. In terms of safety supervision in public places, some moving trajectories deviate from the flow of people, and there may be potential security risks, such as theft or acts that endanger public safety. As shown in Figure 1 in order to ensure the safety of tourists in tourist attractions and maintain the normal sightseeing order, real-time monitoring of the movement of passengers in a certain scenic spot is performed. $1 \sim 3$ indicate the sequence of collection of tourist location information and the location of the tourist at that moment. It can be seen from the figure that tourists o_1, o_2, o_4, and o_5 all follow the flow of people from attractions 1 to attractions 2, but tourist o_3 is different from others. So,tourist o_3 is regarded as an abnormal moving object. There are several possibilities for this abnormal situation, maybe the tourist o_3 is lost, or he is not an ordinary tourist, but a criminal with a special attempt. However, no matter what the reason is, it needs to attract the attention of scenic area managers, and timely provide help or stop the occurrence of some situations. It can be seen that real-time detection of abnormal moving objects based

The associate editor coordinating the review of this manuscript and approving it for publication was Yongqiang Zhao.

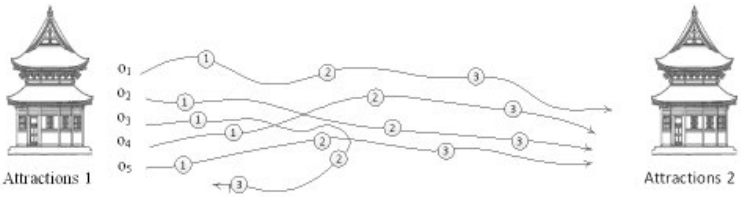


FIGURE 1. Application example of trajectory data streams.

on trajectory data streams has important practical application value.

In this paper, we present efficient algorithms for the outliers detection on trajectory data streams over sliding windows. In summary, the major contributions of this work are as follows:

A new algorithm Trajectory Outlier Detection on trajectory data Streams(TODS) is designed for outlier detection on trajectory data streams. The algorithm is able to quickly determine the nature of an outlier by lightweight definition, to further improve the efficiency.

We propose an approximate approach for Trajectory Outlier Detection (ATODS) based on TODS by distance tree structure. In this way, the such amount of calculation is effectively reduced, thus meeting with the real-time criteria of processing trajectory data streams.

In the remainder of this paper, we introduce related work in Section II. After that, we formally define outlier problem in Section III. Section IV introduces our methods for outlier detection on trajectory data streams. An extensive experimental evaluation of the performance of the algorithms is presented in Section V. Finally, Section VI concludes the work and briefly discusses future work.

II. RELATED WORK

A. TRAJECTORY ANOMALY MEASUREMENT

Trajectory data is a kind of spatiotemporal data, which has become a hot research topic in recent years [5], [6]. Trajectories are usually represented by several interrelated static points, which makes the object distance measurement method [7], [8] in traditional outlier detection unable to be directly used to measure the distance between trajectories.

In 2000, reference [9] proposed to represent the trajectory as several independent global attributes. The disadvantage of this method is that too many parameters are considered when measuring the abnormality of the trajectory, and the determine method is too complicated, which is only suitable for the trajectory with short and simple path.

In 2008, reference [10] proposed to measure the trajectory outlier based on the Hausdorf distance function,which was originally used for image similarity measurement. Although the trajectory is also composed of points, it is different from the image because the trajectory has more time dimension than the image, which has an impact on the measurement of outliers. The Hausdorf distance function only considers the geometry of the trajectory, not the time dimension.

In 2009, reference [11] proposed to use k continuous trajectory points as the basic comparison unit to represent the local characteristics of the trajectory, and proposed a distance function based on the measurement of the degree of matching of the basic comparison unit.

The existing measurement of trajectory outliers usually divides the trajectory into trajectory segments, and then calculates the distance between trajectory segments. The main method of measurement is very complex and cannot meet the real-time requirements of trajectory data stream algorithm.

B. MOVING OBJECT INDEX

This application is directed to the environment of trajectory data streams, and provides outlier moving object detection services in the context of free space and limited space applications [12]–[14]. The following describes the existing index structure from two aspects.

At present, there are many ways to optimize based on update strategies. LUGrid (Lazy Update Grid) [15] and R-tree (LUR-tree) [16] proposed the use of delayed insertion and deletion methods to deal with the problem of frequent index updates caused by moving object location updates. RUM-tree (RUM-tree) [17] proposed to use update memo to solve the problem of frequent index update caused by the change of the position of the moving object.

In some practical applications, the information of the moving object in the future is usually queried [18], which requires the prediction of the position of the moving object and the index of the current and future position of the moving object. The main methods include: transform domain method, original space temporal domain method and temporal parameterization method.

Temporal parameterization method. The main idea is to construct a minimum bounding rectangle (MBR) function based on time, so that moving objects in the same time period can execute any query at any time in the same rectangle. The time parameter R-tree (TPR-tree) [19] is based on R-tree and adopts the idea of parameter bounding boxes. The MBR of the TPR-tree is used to cover a set of moving objects, which is called a conservative bounding rectangle.

The indexes for historical trajectories are mainly divided into: overlap method, time-dimension method and index trajectory method.

Method of adding time dimension. The main idea is to add time dimension on the basis of the R tree to build an

index, which is used to process spatial queries with temporal predicates. The MTSB tree [20] divides the space into disjoint cells, and then uses the TSB tree [21] to index the objects. In this way, the TSB tree results of all cells are combined to support all spatial-temporal range trajectory queries. STR tree [22] ensure the tightness of the object in time and space, the identification of the trajectory and the information of its starting position in MBR are added to the node.

The existing trajectory indexes are based on static trajectory data, which cannot be incrementally maintained and the indexes are costly. They are not suitable for frequently updated trajectory data stream.

C. ABNORMAL MOVING OBJECT DETECTION QUERY

In this section, the research status at home and abroad is introduced from two aspects: outlier detection on trajectory data and outlier detection on trajectory data streams.

1) OUTLIER DETECTION ON TRAJECTORY DATA

In 2008, Lee *et al.* [10] proposed a partition-based abnormal trajectory detection framework, which divided the trajectory into a set of line segments, and used the line segments to represent the local characteristics of the trajectory. The algorithm is mainly composed of a two-layer trajectory segmentation strategy and a distance-based density hybrid method. The disadvantage of this method is that when the number of trajectories is huge, the calculation of the distance between trajectory segments is very time-consuming, which directly affects the operation efficiency of the algorithm.

In 2010, Ge *et al.* [23] proposed a $Top-k$ evolving trajectory outlier detection method (Top-EVE) to detect $Top-k$ outlier trajectories. First, a grid is established for the space area, and a direction matrix is defined, and a direction-based summary vector is generated based on the historical data of the trajectory. Then the distance between the specified trajectory and the summary vector is calculated to determine the abnormal situation of the trajectory.

In 2009, Liu *et al.* [11] proposed an outlier trajectory detection algorithm based on R-Tree. This method determines whether the two trajectories match entirely by detecting the local abnormality of the trajectory, and then detects the abnormal trajectory. It is proposed to use the distance feature matrix between the R-Tree and the trajectory to find all possible pairs of basic comparison units, and then determine whether they are locally matched by distance calculation. The algorithm only considers the geometric characteristics of the trajectory, it does not consider the velocity dimension of the trajectory.

2) OUTLIER DETECTION ON TRAJECTORY DATA STREAMS

In 2009, Bu *et al.* [23] studied the problem of real-time detection of outliers in single-trajectory data streams, and the detection target was anomalous sub-trajectory segments in single-trajectory streams. Due to the huge amount of data in the trajectory streams, outlier detection cannot be performed after being stored. Therefore, the detection in the trajectory stream environment requires high real-time performance, and the outliers must be detected in real-time processing. Three sliding windows are defined in the article, and outlier detection is performed through incremental maintenance between windows. The algorithm is only applicable to single-trajectory data streams.

In 2014, reference [24] proposed to use position information between the points in the trajectory for traditional outlier detection. Further more, in order to consider the speed dimension, the update of the neighbors of the moving object was also used as one of the judgment conditions. In brief, if the moving object does not have a sufficient number of trajectory neighbors, the trajectory is considered as an outlier. This method used many parameters to determine the outlier of the trajectory, and the accuracy of the detection results will be restricted.

In 2017, Mao *et al.* [25] proposeed a feature grouping-based mechanism that divided all the features into two groups, where the first group (Similarity Feature) is used to find close neighbors and the second group (Difference Feature) is used to find outliers within the similar neighborhood.

Most of algorithms mentioned above all work on static trajectory data. This means that the algorithm must be executed from scratch if there are changes in the underlying data objects, leading to performance degradation when operated on trajectory data streams. A few outliers detection algorithms of trajectory data streams, which divided the trajectory into sub-trajectory segments and the index structure cannot be incrementally updated, so the algorithms cannot meet the real-time requirements of trajectory data streams.

III. PROBLEM DEFINITION

In this section, we first introduce the definition of outliers on trajectory data streams. Then, we denote the set of n moving objects as $O = o_1, o_2 \cdots o_n$, where o_i is a moving object with $id = i$ and $1 \leq i \leq n$. Trajectory T_i produced by the moving object o_i. The point p_i^μ is a point at timebin t_μ that is in the trajectory T_i, so trajectory T_i consist of points $p_i^1, p_i^2, \cdots\cdots$. Outlier definition on trajectory data streams is based on the definition of outlier for deterministic data.

Definition 1 (Point Neighbor): The two trajectory points p_i^μ and p_j^μ are respectively points of trajectory T_i and T_j at time μ, where $i \neq j$. Let $d(p_i^\mu, p_j^\mu)$ be the distance between point p_i^μ and p_j^μ. If $d(p_i^\mu, p_j^\mu) \leq R$, we say that point p_j^μ is the point neighbor of p_i^μ at timet_μ, with R a given distance threshold.

For point p_i^μ, the set of point neighbors is denoted as $N(p_i^\mu)$.

Definition 2 (Neighbor Update Probability): Let p_i^μ be the point of trajectory T_i at time t_μ, where $T_i \in T$. Let $p_i^{\mu-1}$ be the previous time point of p_i^μ. Let $|N(p_i^{\mu-1})|$ be the number of points in set $N(p_i^{\mu-1})$, and $N(p_i^\mu) - N(p_i^{\mu-1})$ the number of point neighbors in $N(p_i^{\mu-1})$ and not in $N(p_i^\mu)$. P_i^μ is the neighbor update probability, then

$$P_i^\mu = \frac{|N(p_i^{\mu-1}) - N(p_i^\mu)|}{|N(p_i^\mu)|}. \tag{1}$$

TABLE 1. Frequently used symbols.

Symbol	Interpretation
o_i	The *i-th* moving object
O_i	Set of points of the o_i moving object
T_i	Trajectory of o_i moving object
T	Set of trajectories T_i for $i = 1$ to n
p_i^μ	Point of trajectory T_i at time t_μ
$d(p_i^\mu, p_j^\mu)$	Distance between point p_i^μ and p_j^μ
R	Given distance threshold
$N(p_i^\mu)$	Set of point neighbors of p_i^μ
$\|N(p_i^\mu)\|$	Number of points in set $N(p_i^\mu)$
P_i^μ	Update neighbor probability of point p_i^μ
$\|N(p_i^\mu) - N(p_i^{\mu-1})\|$	Number of update neighbors of p_i^μ
$F(T_i)$	Outlier factor of Trajectory T_i
$\|O(T_i)\|$	Number of point outlier in Trajectory T_i
$\|W\|$	Number of points in trajectory T_i in Sliding Window
$dis(p_i^\mu, p_j^\mu)$	Distance between p_i^μ and p_j^μ, $i \neq j$

Definition 3 (Point Outlier): p_i^μ is the point of trajectory T_i at time t_i. $N(p_i^\mu)$ is the set of point neighbors of point p_i^μ,and $|N(p_i^\mu)|$ is the number of points in the p_i^μ. The k and P be user specified thresholds, where $0 < k$ and $0 < P < 1$. If the $|N(p_i^\mu)| < k$ or $P_i^\mu > P$ then point $p_i^{\mu-1}$ is a point outlier.

In this paper, we use sliding window. Let $W_{current}$ mean current window, the sliding window is based number points in window. W is the number of points of each T_i in the current window.

Definition 4 (Trajectory Outlier Factor): Trajectory T_i consist of p_i^μ, where $0 < i < n$ and $\mu > 0$, W is the window size, $|O(T_i)|$ is the number of point outlier in Trajectory T_i, then

$$F(T_i) = \frac{|O(T_i)|}{W}. \tag{2}$$

Definition 5 (Top-k Trajectory Outlier): Let $F(T_i)$ be trajectory outlier factor of Trajecoty T_i, where $T_i \in T$. According to a $F(T_i)$ of each T_i value, arrange in descending order. The first $top - k$ outliers are trajectory outlier.

Problem: Trajectory Outlier. Given the parameters, R, k and P and window size W, the trajectory outlier at each position of the window are outputted.

An incremental algorithm should be designed, which only changes the previous dataset with the previous result to produce the new result. In this paper, we only consider the case that the slide size of window is 1. We used the count-based window which always maintains the W most recent points. Our technique can be easily extended to time-based sliding window.

IV. OUTLIER DETECTION ON TRAJECTORY DATA STREAMS

First, we describe the basic algorithm-TODS, it is based on a lightweight method. Further more, we introduce the approximate algorithm-ATODS, that according to the Spatiotemporal correlation of trajectory data streams.

A. BASIC ALGORITHM

In this subsection, a basic algorithm for outlier detection on trajectory data streams is presented. It is able to quickly determine the nature of each trajectory. The research based on trajectory data streams, so the main processes are mainly divided into two parts, initialization and incremental update.

In the initialization phase, the window contains w points of each trajectory. Firstly, according to the Definition 1, the point neighbors should be find for every trajectory point p_i^μ, and set up the set of point neighbors $N(p_i^\mu)$ for every p_i^μ. According to Definition 2, comparing the point neighbor sets $N(p_i^\mu)$ of neighbor time data point p_i^μ, get the neighbor update probability $p(p_i^\mu)$. According to the Definition 3, we can get the point outlier at every timebin. The next step is to calculate the outlier factor of every trajectory T_i. Finally, according to the outlier factors of each trajectory T_i, trajectories are arranged in descending order, and the $Top-k$ tracks are on the outlier trajectory in initialization phase. The costs of TODS algorithm are $O(n^2)$, where n is the number of trajectories. The initialization framework of the outlier detection on trajectory data streams is illustrated in Figure 2.

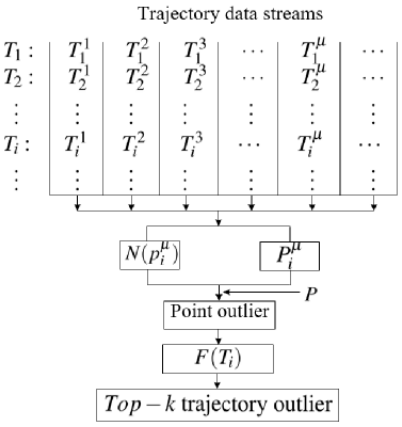


FIGURE 2. Algorithm Framework for Trajectory Outlier Detection.

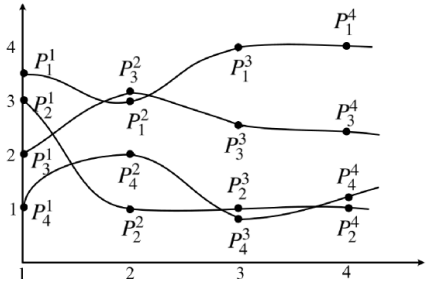


FIGURE 3. Example of trajectory outlier detection.

As example, the Figure 3 shows that the four trajectories, $T_1 \ldots T_4$, the window size is $W = 3$. Table 2 shows all points of trajectory in initialization window. We assume that $R = 1.5$, $m = 2$, $P = 0.75$ and $k = 1$. At time t_1, the point neighbors of trajectory point p_1^1 are p_2^1 and p_3^1, so the set of

TABLE 2. Example of trajectory data.

Trajectory	t_1	t_2	t_3	t_4
T_1	(1.0, 3.5)	(2.0, 3.0)	(1.9, 2.7)	(3.0, 4.0)
T_2	(1.0, 3.0)	(2.0, 1.0)	(3.0, 1.0)	(4.0, 1.0)
T_3	(1.0, 2.0)	(2.0, 3.2)	(3.0, 2.5)	(4.0, 2.4)
T_4	(1.0, 1.0)	(2.0, 2.0)	(3.0, 0.8)	(4.0, 1.2)

TABLE 3. Details of initialization window.

T_i	t_1	t_2	P_i^2	t_3	P_i^3	t_4	P_i^4
T_1	$\{p_2^1, p_3^1\}$	$\{p_3^2, p_4^2\}$	0.5	$\{p_3^3\}$	0	$\emptyset$	1
T_2	$\{p_1^1, p_3^1, p_4^1\}$	$\{p_4^2\}$	1	$\{p_3^3, p_4^3\}$	0	$\{p_3^4, p_4^4\}$	0
T_3	$\{p_1^1, p_2^1, p_4^1\}$	$\{p_1^2, p_4^2\}$	0.33	$\{p_1^3, p_2^3\}$	0.5	$\{p_2^4, p_4^4\}$	0.5
T_4	$\{p_3^1\}$	$\{p_1^2, p_4^2\}$	0	$\{p_3^3\}$	1	$\{p_2, p_3\}$	0
PointOutlier	$\{p_4^1\}$	$\{p_2^2\}$		$\{p_1^3, p_4^3\}$		$\{p_1^4\}$	

point neighbor p_1^1 is $N(p_1^1) = \{p_2^1, p_3^1\}$. At time t_1, the neighbor set of point p_4^1 is $\{p_3^1\}$. The number of neighbors of point p_4^1 is $1 < m$, so point p_4^1 is outlier in time t_1. At timebin t_2, the point neighbors of trajectory point p_1^2 are p_3^2 and p_4^2, the set of point neighbors p_1^2 is $N(p_1^2) = \{p_3^2, p_4^2\}$, and neighbor update probability P_1^2 of point p_1^2 is 0.5. The neighbor updates probability p_4^2 of point p_4^4 is equal 1. At timebin t_2, point p_2^2 and p_4^2 are outliers. At timebin t_3, point p_1^3 and p_4^3 are outliers. According Definition 2, the outlier factor of every trajectory: $F(T_1) = 0.3$; $F(T_2) = 0.3$; $F(T_3) = 0$; $F(T_4) = 1$. In the initialization window, the trajectory T_4 is trajectory outlier. The Table 3 shows that all details of the example.

In order to meet the requirements of data streams, the ability of incremental update is needed for the detection of trajectory outliers. When windows slide, the trajectory points at time $t(\mu)$ expired, and the points at time t_{i+w} will arrival, w is windows size. Outlier factor incremental update:

$$F_{w+1}((T_i)) = \frac{F_w(T_i) * W - \alpha + \beta}{W} \tag{3}$$

where $F_w(T_i)$ is the outlier factor of trajectory T_i in current window, $F_{w+1}((T_i))$ is the new factor of trajectory T_i in next window that is the next window after current window. If trajectory point p_i^μ is outlier, then $\alpha = 1$, otherwise $\alpha = 0$. If trajectory point $p_i^{\mu+w}$ is outlier, then $\beta = 1$, otherwise $\beta = 0$.

When the window slides, time t_1 expired, time t_4 arrives. Based on the above, we can get the outlier factor in the new window.

B. APPROXIMATE ALGORITHM

In the basic algorithm, we need to find the nearest neighbor of all the trace points at every time point. If the number of trajectories is n, then the complexity of computing the neighborhood is n^2. When there is a lot of trajectories, it will affect the detection efficiency. In this section, we introduce the approximate algorithm to improve the efficiency of finding nearest neighbors.

Although the trajectory points may exist in any position at a time t_μ, from the point of view of the track, they have time correlation, that is to say, the two neighbor time

Algorithm 1 TODS Algorithm

Input: Trajectory data streams T
Output: Trajectory outlier

```
Initialize Trajectory outlier set T_O = ∅;
for i = 1 to |W| do
    Calculate the distance between p_i^μ to p_j^μ, j ≠ i
    Set up set of point neighbor N(p_i^μ)
end for
for i = 1 to |W| do
    Calculate the neighbor update probability p_i^j, j ≠ i
    According to probability threshold determined point
    outlier;
    Calculate the outlier factor F(T_i)
end for
Descending order of outlier factor F(T_i)
Get the top − k trajectories
```

TABLE 4. Example of ATODS.

Trajectory point	Neighbor set
p_a^μ	$\{p_b^\mu, p_c^\mu\}$
p_b^μ	$\{p_a^\mu, p_c^\mu, p_d^\mu\}$
p_c^μ	$\{p_a^\mu, p_b^\mu, p_f^\mu, p_i^\mu\}$
p_d^μ	$\{p_b^\mu, p_g^\mu, p_h^\mu\}$
p_e^μ	$\{p_f^\mu, p_b^\mu\}$
p_f^μ	$\{p_c^\mu, p_i^\mu\}$
p_g^μ	$\{p_d^\mu\}$
p_h^μ	$\{p_d^\mu\}$
p_i^μ	$\{p_c^\mu, p_j^\mu, p_k^\mu\}$
p_j^μ	$\{p_i^\mu\}$
p_k^μ	$\{p_i^\mu\}$

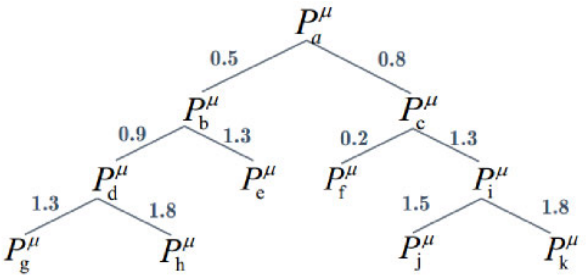


FIGURE 4. Tree structure of ATODS algorithm.

points, the change of the trajectory has some potential rulers. We need to build a tree structure to store this relationship, to make use of these rules to conduct neighbor query.

After initialization, we can get the neighborhood set of every trajectory point. We choose high density point as root node of tree, and its neighbors are its children's nodes. The weight of an edge is the distance between two nodes. The nearest neighbor with the smallest distance is the left child node, and the nearest neighbor with the longest distance is the right child node. Continue using this method to find grandchildren. When a node has more than one path to the root node, it is placed at the nearest position to the root node.

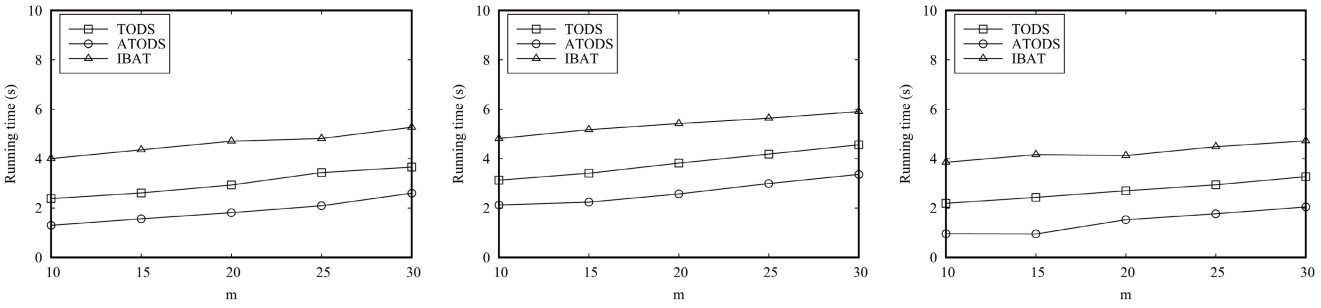


FIGURE 5. Running time vs *m* value. (a) BJ. (b) TAXI. (c) SYN.

Use this method to fill in the leaf node until the leaf node cannot be found, It means that the tree has been established. In the same way, continue building another tree.

For example, in Table 4. At time t_μ, for trajectory point p_a^μ, the neighbor set $N(p_a^\mu)$ of is $\{p_b^\mu, p_c^\mu\}$ and neighbor set $N(p_b^\mu)$ is $\{p_a^\mu, p_c^\mu, p_d^\mu\}$. We choose high density point, such as p_a^μ as the tree node. Its neighbors p_b^μ and p_c^μ are its children's nodes. Because of the distance between p_a^μ and p_b^μ is 0.5, and the distance between p_a^μ and p_c^μ is 0.8, so p_a^μ is the left child of root node p_a^μ. The neighbor set of p_b^μ is $\{p_a^\mu, p_c^\mu, p_d^\mu\}$, and $dis(p_b^\mu, p_a^\mu) = 0.5$, $dis(p_b^\mu, p_d^\mu) = 0.6$, $dis(p_b^\mu, p_b^\mu) = 0.5$, then $dis(p_b^\mu, p_a^\mu) > dis(p_b^\mu, p_d^\mu) > dis(p_b^\mu, p_b^\mu)$. Firstly, we should insert the p_a^μ into the tree structure, if it has already existed, then the p_a^μ is no longer inserted into the tree again. According to the information in the Table 4, the tree can be established as shown in Figure 4.

When the window slides, time $t_{\mu+1}$ arrives, we can use this tree to quickly determine the attributes of trajectory points, instead of calculating the distance between any two points. Generally, the number of trajectory is large, and the threshold m is very small, so it is not necessary to find all its neighbors, but to find m neighbors to meet the condition of non-outliers. At time t_μ, p_b^μ is the nearest neighbor of p_a^μ. For trajectory point p_a^μ, $p_b^{\mu+1}$ is given priority to determine whether it's the neighbor to $p_a^{\mu+1}$. Accurately calculate the distance between $p_a^{\mu+1}$ and $p_b^{\mu+1}$, and update distance information on the tree. The second priority is $p_c^{\mu+1}$, because p_c^μ is the second nearest neighbor of p_a^μ. Continue searching until m neighbors are found. For other non-root data points, such as $p_c^{\mu+1}$, $p_c^{\mu+1}$ is preferred. When looking for the point neighbor, we should also consider the update rate of the neighbors. When you can't find the neighbors continuously with this tree structure, it need to consider whether this trajectory point is an point outlier, or the tree structure needs to be updated. Inserting the data point into the candidate set, only if the trajectory point's neighbors cannot be found in the current tree. When the number of trajectory points in the candidate set exceeds the threshold value, it means that the current tree structure is no longer applicable, so it needs to re-establish the tree structure according to the trajectory position of the current time point.

V. EXPERIMENTAL EVALUATION

We conduct extensive experiments on two real datasets and a synthetic dataset to evaluate the running time and accuracy of the three algorithms developed in this paper: the basic algorithm (TODS), and the approximate algorithm (ATODS). For comparison, we implement the outlier detection algorithm (IBAT) [26]. We introduced the experimental setup and evaluation criteria in the first subsection. The experimental results and analysis are then introduced in other subsections.

A. EXPERIMENT SETTINGS

We conduct extensive experiments on two real datasets and one synthetic dataset to evaluate the running time and accuracy of the algorithms developed in this paper: the basic algorithm (TODS), the approximate algorithm (ATODS). For comparison, we implement the outlier detection algorithm IBAT. All the algorithms are implemented using Microsoft Visual C++. The experiments are run on PC with a Core i3-3.3 GHz CPU and 4G of main memory. We use two real-world datasets and one synthetic dataset. The real datasets are (1) Beijing trajectories dataset (BJ), which contains 5,660,692 trajectories. (2) Taxi GPS dataset (TAXI) [27], [28], which generated by 10,357 taxis. The total numbers of points is about 15 million. And we generate a synthetic trajectory stream dataset (SYN).

B. RUNNING TIME EVALUATION

In order to prove the efficiency of our proposed algorithms, we first discuss the cost of algorithms. We vary the values of parameters to compare our algorithms and investigate the effect of each parameter.

Figure 5 shows the running time of the three algorithms on three datasets. The running time for all three algorithms proposed in this paper increase as m increases, because more data points need to be processed in the *R-neighbor* set. TODS and ATODS are faster than IBAT. As expected, the perform of TODS and ATODS is better than IBAT since R, P, w are constants. In general, ATODS runs faster than TODS because it reduces unnecessary calculation.

Next, we study the performance of the proposed methods with varying distance threshold R. The results are given

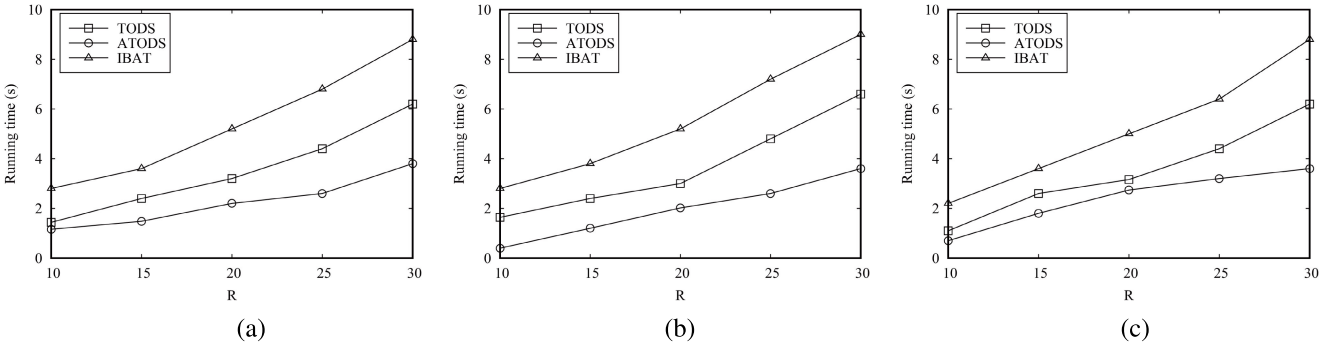


FIGURE 6. Running time vs *R* value. (a) BJ. (b) TAXI. (c) SYN.

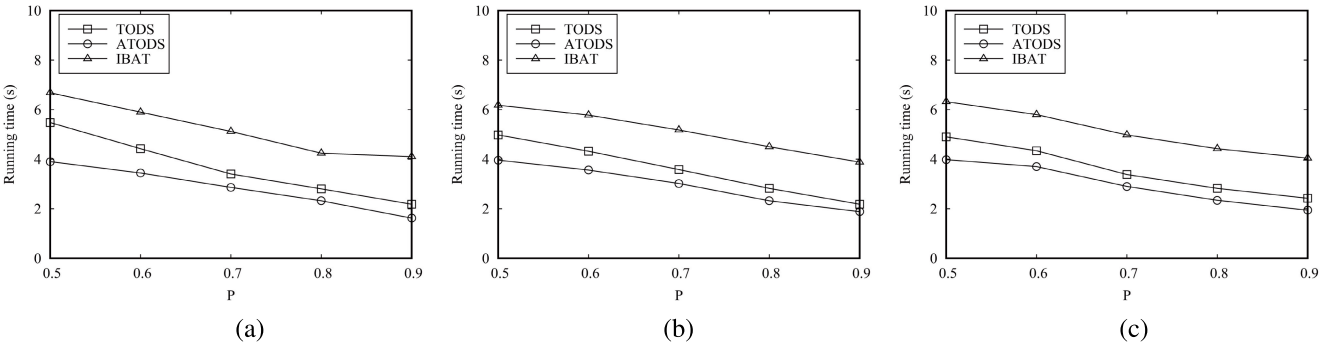


FIGURE 7. Running time vs *P* value. (a) BJ. (b) TAXI. (c) SYN.

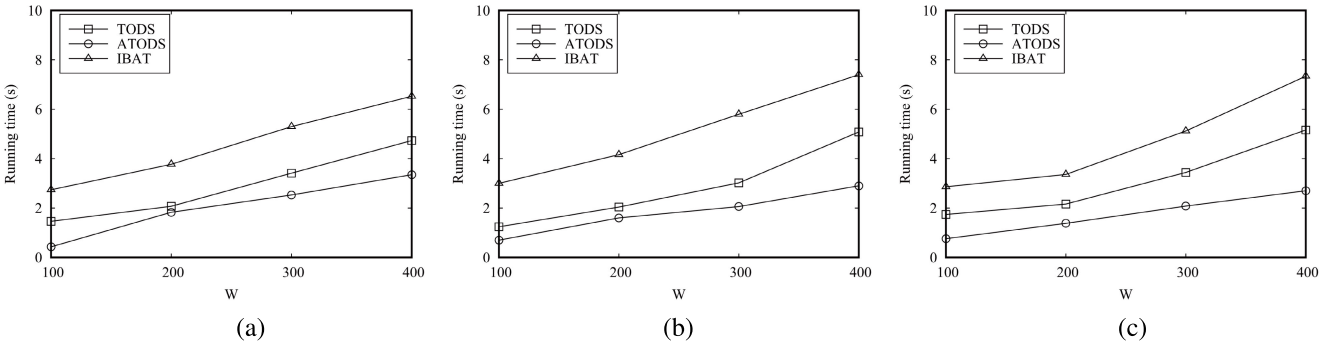


FIGURE 8. Running time vs *W* value. (a) BJ. (b) TAXI. (c) SYN.

in Figure 6. R varies from 10 to 30, and the performance of our algorithms may degrade as R increases. Because more data points need to be processed in the $R - neighbor$ set. TODS and ATODS are faster than IBAT. In general, ACUOD runs faster than CUOD, because it reduces the number of data points.

In the third experiment, we investigate the running time of varying neighbor update probability threshold P. The results are given in Figure 7. The probability threshold varies from 0.5 to 0.9. The running time decrease as parameter P increases. Notice that TODS and ATODS are faster than IBAT.

The next experiment studies the running time of the algorithms for window size W. The running time increase for all algorithms as parameter W increases,because more data points need to be processed in the current window. The results are shown in Figure 8.

C. ACCURACY EVALUATION

In this subsection, we give the experimental results about accuracy on synthetic dataset and real datasets. We study the effect on accuracy of different parameters. In Figure 9, the m from 10 to 30. The results show that m value has relatively small impact on accuracy.

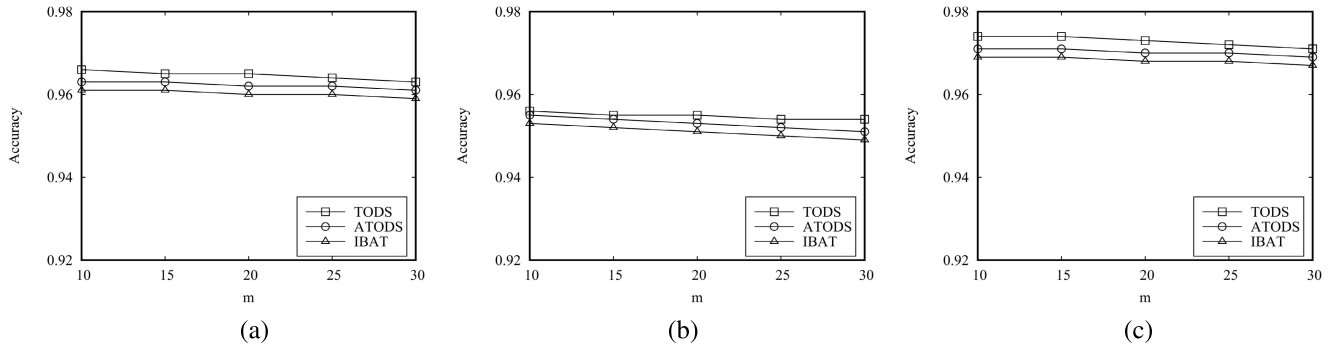


FIGURE 9. Accuracy vs *m* value. (a) BJ. (b) TAXI. (c) SYN.

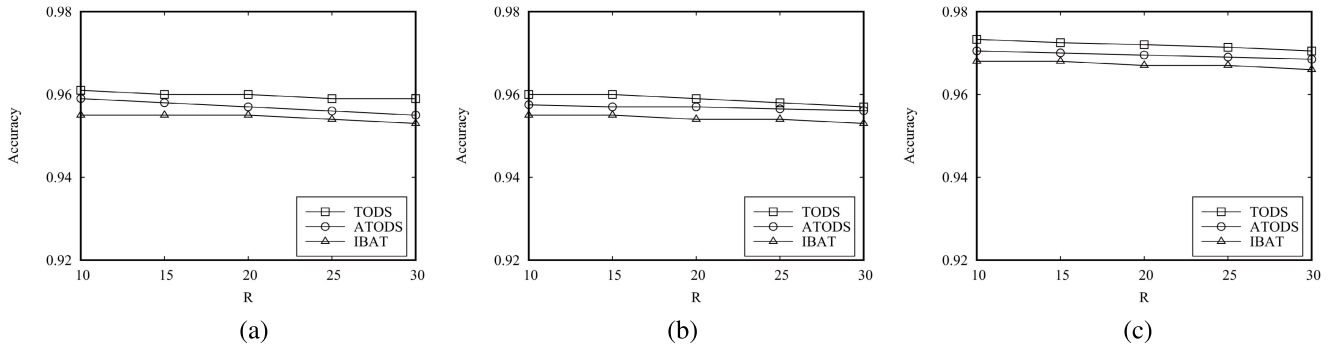


FIGURE 10. Accuracy vs *R* value. (a) BJ. (b) TAXI. (c) SYN.

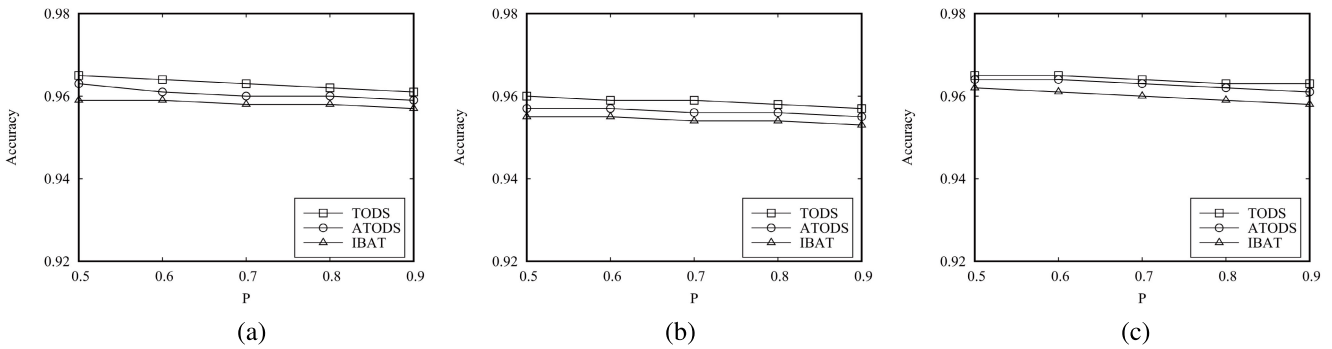


FIGURE 11. Accuracy vs *P* value. (a) BJ. (b) TAXI. (c) SYN.

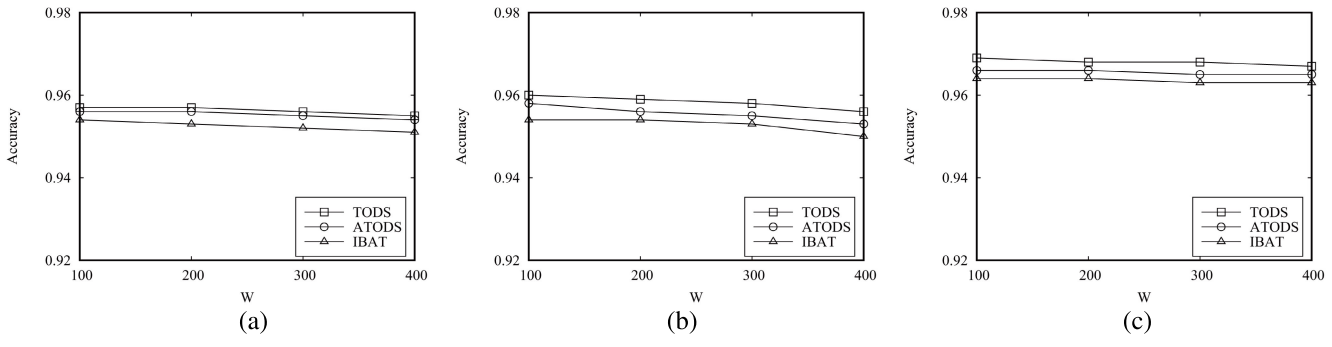


FIGURE 12. Accuracy vs *W* value. (a) BJ. (b) TAXI. (c) SYN.

We study the accuracy of the three methods with varying R. The results are shown in Figure 10, where R varies from 10 to 30. As expected, our algorithms have good efficiency without loss of accuracy. Because the approximate algorithm does not have any impact on the generation of outlier, it prunes strictly according to the nature of outliers.

We investigate the accuracy of varying P as Figure 11. The probability P varies from 0.5 to 0.9. The experimental results show that the accuracy of the algorithm is almost unaffected by the probability value.

Finally, we study the accuracy of the four methods with varying windows size. Figure 12 show the accuracy of the three algorithms on the three data sets. As we expected, the window size has relatively small impact on accuracy, because even for the biggest window size, we can still get a very high accuracy.

The parameters has relatively small impact on accuracy. As expected, our algorithms have good efficiency without loss of accuracy. Because the approximate method does not have more impact on the generation of outlier.

VI. CONCLUSION

In this work, we focus on the detection of abnormal moving objects over trajectory data streams. After analyzing the requirements of stream trajectory applications, we propose a lightweight method to measure the outlier. Our empirical study result shows that it can effectively detection moving object outliers. Furthermore we design an ATODS algorithm on data trajectory streams to reduce the amount of calculations. In the future work, we will continue studing the trajectory outlier detection in various common data environments.

REFERENCES

[1] J. J. Xu, K. Zheng, M. W. Chi, Y. Y. Zhu, X. H. Yu, and X. F. Zhou, "Trajectory big data: Data, application and technology status," *J. Commun.*, vol. 12, pp. 97–105, Dec. 2015.

[2] J. Zhu, W. Jiang, A. Liu, G. Liu, and L. Zhao, "Effective and efficient trajectory outlier detection based on time-dependent popular route," *World Wide Web*, vol. 20, no. 1, pp. 111–134, Jan. 2017.

[3] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "IBOAT: Isolation-based online anomalous trajectory detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 806–818, Jun. 2013.

[4] H. Wu, W. Sun, and B. Zheng, "A fast trajectory outlier detection approach via driving behavior modeling," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2017, pp. 837–846.

[5] S. Shang, L. Chen, C. S. Jensen, J.-R. Wen, and P. Kalnis, "Searching trajectories by regions of interest," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 7, pp. 1549–1562, May 2017.

[6] S. Shang, L. Chen, K. Zheng, C. S. Jensen, Z. Wei, and P. Kalnis, "Parallel trajectory-to-Location join," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 6, pp. 1194–1207, Jun. 2019.

[7] S. Shang, L. Chen, Z. Wei, S. Christian, C. S. Jensen, K. Zheng, and P. Kalnis, "Parallel trajectory similarity joins in spatial networks," *VLDB J.*, vol. 27, no. 3, pp. 395–420, 2018.

[8] L. Cao, M. Wei, D. Yang, and E. A. Rundensteiner, "Online outlier exploration over large datasets," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2015, pp. 89–98.

[9] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications," *VLDB J. Int. J. Very Large Data Bases*, vol. 8, nos. 3–4, pp. 237–253, Feb. 2000.

[10] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Apr. 2008, pp. 140–149.

[11] L. X. Liu, S. J. Qiao, and B. L. An, "Efficient anomaly trajectory detection algorithm based on r-tree," *J. Softw.*, vol. 20, no. 9, pp. 2426–2435, 2009.

[12] S. Shang, L. Chen, Z. Wei, C. S. Jensen, J.-R. Wen, and P. Kalnis, "Collective travel planning in spatial networks," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1132–1146, May 2016.

[13] S. Shang, R. Ding, K. Zheng, C. S. Jensen, P. Kalnis, and X. Zhou, "Personalized trajectory matching in spatial networks," *VLDB J.*, vol. 23, no. 3, pp. 449–468, Jun. 2014.

[14] L. Chen, S. Shang, C. Yang, and J. Li, "Spatial keyword search: A survey," *GeoInformatica*, vol. 24, no. 1, pp. 85–106, Jan. 2020.

[15] X. Xiong, M. F. Mokbel, and W. G. Aref, "LUgrid: Update-tolerant grid-based indexing for moving objects," in *Proc. 7th Int. Conf. Mobile Data Manage. (MDM)*, 2006, pp. 13–20.

[16] D. Kwon, S. Lee, and S. Lee, "Indexing the current positions of moving objects using the lazy update R-tree," in *Proc. 3rd Int. Conf. Mobile Data Manage. (MDM)*, Oct. 2003, pp. 113–120.

[17] Y. N. Silva, X. Xiong, and W. G. Aref, "The RUM-tree: Supporting frequent updates in R-trees using memos," *VLDB J.*, vol. 18, no. 3, pp. 719–738, Jun. 2009.

[18] X. Yue, M. Xi, B. Chen, M. Gao, Y. He, and J. Xu, "A revocable group signatures scheme to provide privacy-preserving authentications," in *Mobile Networks and Applications*. 2019, pp. 1–18.

[19] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," in *Proc. SIGMOD*, 2000, pp. 331–342.

[20] P. Zhou, D. Zhang, B. Salzberg, G. Cooperman, and G. Kollios, "Close pair queries in moving object databases," in *Proc. Int. Workshop Geographic Inf. Syst. (GIS)*, 2005, pp. 2–11.

[21] D. Lomet and B. Salzberg, "Access methods for multiversion data," *ACM SIGMOD Rec.*, vol. 18, no. 2, pp. 315–324, Jun. 1989.

[22] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel approaches in Query processing for moving object trajectories," in *Proc. VLDB*, 2000, pp. 395–406.

[23] Y. Bu, L. Chen, A. W.-C. Fu, and D. Liu, "Efficient anomaly monitoring over moving object trajectory streams," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 159–168.

[24] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Detecting moving object outliers in massive-scale trajectory streams," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2014, pp. 422–431.

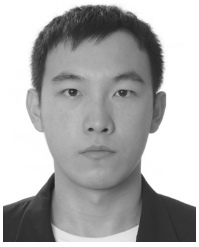
[25] J. Mao, T. Wang, C. Jin, and A. Zhou, "Feature grouping-based outlier detection upon streaming trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2696–2709, Dec. 2017.

[26] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "IBAT: Detecting anomalous taxi trajectories from GPS traces," in *Proc. 13th Int. Conf. Ubiquitous Comput. (UbiComp)*, 2011, pp. 99–108.

[27] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 316–324.

[28] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: Driving directions based on taxi trajectories," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS)*, 2010, pp. 99–108.

KEYAN CAO received the M.S. and Ph.D. degrees in computer science and technology from Northeastern University, China, in 2009 and 2014, respectively. She is currently an Associate Professor with the College of Information and Control Engineering, Shenyang Jianzhu University. Her current research interests include data management, cloud computing, and query process and optimization.



YEFAN LIU received the B.S. degree from the Shenyang University of Technology, China. He is currently pursuing the master's degree with the School of Computer Science and Technology, Shenyang Jianzhu University. His current research interest is big data processing.

GONGJIE MENG received the bachelor's degree in software engineering from the Jincheng College, Nanjing University of Aeronautics and Astronautics, in 2019. He is currently pursuing the master's degree in software engineering with Shenyang Jianzhu University. He is also conducting research on pulmonary nodule detection and urban big data management and application.

HAOLI LIU was born in Bozhou, China, in 1996. She received the B.S. degree from Shenyang Jianzhu University, China, in 2019, where she is currently pursuing the master's degree with the School of Computer Technology. Her research interest is big data analysis.

ANCHEN MIAO received the B.S. degree from the Liaoning University of Technology, in 2015. He is currently pursuing the M.S. degree with the Information and Control Engineering Faculty, Shenyang Jianzhu University. His research interests include big data, trajectory data mining, and outlier detection.

JINGKE XU was born in Anshan, China, in 1976. He received the M.E. degree in computer software and theory from Northeastern University, Shenyang, China, where he is currently pursuing the Ph.D. degree. He is also an Associate Professor of computer science with Shenyang Jianzhu University. He has published more than 20 articles in important journals and conferences at home and abroad, such as the *Journal of Computer Research and Development*, the *Chinese Journal of Computers* and NDBC. His main research interests include spatiotemporal data management and analysis, data mining, and intelligent optimization. He is a CCF member and has hosted and participated in a number of national and provincial scientific research projects.