Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Survey on Urban Traffic Anomalies Detection Algorithms

YOUCEF DJENOURI[1], ASMA BELHADI[2], JERRY CHUN-WEI LIN[3] DJAMEL DJENOURI [4] AND ALBERTO CANO[5,*]
[1]Dept. of Computer Science, NTNU, Trondheim, Norway (e-mail:youcef.djenouri@ntnu.no)
[2]RIMA, University of Science and Technology Houari Boumediene (USTHB), Algiers, Algeria (e-mail: abelhadi@usthb.dz)
[3]Department of Computing, Mathematics, and Physics, Western Norway University of Applied Sciences, Bergen, Norway, (email: jerrylin@ieee.org)
[4]CERIST Research Center, Algiers, Algeria (email: ddjenouri@acm.org)
[5]Dept. of Computer Science, Virginia Commonwealth University, Richmond, VA, USA, (*corresponding author, email: acano@vcu.edu)

Corresponding author: Alberto Cano (e-mail: acano@vcu.edu).

ABSTRACT This paper reviews the use of outlier detection approaches in urban traffic analysis. We divide existing solutions into two main categories: flow outlier detection and trajectory outlier detection. The first category groups solutions that detect flow outliers and includes statistical, similarity, and pattern mining approaches. The second category contains solutions where the trajectory outliers are derived, including offline processing for trajectory outliers and online processing for sub-trajectory outliers. Solutions in each of these categories are described, illustrated, and discussed, and open perspectives and research trends are drawn. Compared to state-of-the-art survey papers, the contribution of this paper lies in providing a deep analysis of all the kinds of representations in urban traffic data, including flow values, segment flow values, trajectories, and sub-trajectories. In this context, we can better understand the intuition, limitations, and benefits of the existing outlier urban traffic detection algorithms. As a result, practitioners can receive some guidance for selecting the most suitable methods for their particular case.

INDEX TERMS Urban traffic analysis, outlier detection, machine learning, data mining.

I. INTRODUCTION

RECENT advances in high-precision GPS technologies and infrastructure have made our cities smarter. Urban traffic analysis is one of the most attractive applications in a smart city [1], [2]. One of the main applications of urban traffic analysis lies in detecting anomalies from the traffic data. A useful way of detecting anomalies in urban traffic data is by utilizing outlier detection techniques. An outlier is defined as an observation (or a set of observations) which appears to be inconsistent with the remainder of that set of data [3]. Outlier detection has been intensively studied in recent decades [3]–[8], and an interesting recent survey which reviews existing outlier detection methods can be found in [9].

This paper presents a comprehensive overview of the existing urban traffic outlier detection algorithms. We split existing approaches into two main categories: flow outlier detection and trajectory outlier detection. The first one aims at detecting flow outliers, including statistical, similarity, and pattern mining approaches. The second category aims at detecting trajectory outliers and includes offline processing for trajectory outliers and online processing for sub-trajectory outliers. Solutions in each category are described, illustrated, and discussed, and open perspectives and research trends in this area are depicted. Compared to previous review papers, this paper provides a deep analysis of all kinds of urban traffic applications, including flow values, segment flow values, trajectories, and sub-trajectories. This allows us to clearly understand the merits and limitations of each urban traffic outlier detection algorithm. Consequently, mature solutions could be derived for intelligent transportation engineering.

A. PREVIOUS REVIEW PAPERS

This section summarizes survey papers from the literature that are relevant to this one, clarifies the differences, and makes a position for the contribution of this paper. This survey paper is composed of two main topics: outlier detection algorithms and urban traffic data mining. In the following section, we review some existing surveys of these topics. Schubert et al. [10] introduced the locality notion in identifying outliers. By defining the context and model functions, this notion represents locality. The context function outputs the set of reference objects that are relevant to judging the outliers, and the model function is the sequence of tasks applied to the reference objects aiming to determine whether the given object is outlier or inlier. A special case of locality has been shown on video temporal streams data. Zheng [11] reviewed detecting outliers, anomalous trajectories, sub-trajectories, finding noise points in the whole set of trajectories, and the identification of anomalous events in trajectories including accidents, controls, protests, sports, celebrations, disasters, and other events. Feng et al. [12] proposed a general framework of trajectory data mining, including preprocessing, data management, query processing, trajectory data mining tasks, and privacy protection. They also discussed some existing applications such as path discovery, location/destination prediction, movement behavior analysis, group behavior analysis, urban service, and making sense of trajectories. Gupta et al. [13] provided an interesting survey that discussed techniques for the detection of temporal outliers. This survey organized a discussion about different types of data, presented various outlier definitions, and discussed various applications for which temporal outlier techniques have been successfully employed (environmental sensor networks, trajectory, biological, astronomy, and web data). Zheng et al. [14] presented a general urban computing framework, which was composed of four steps: urban sensing, urban data management, data analytics, and service providing. i) Urban sensing aims to capture people's mobility using GPS sensors or their mobile phone signals. ii) Urban data management employs powerful indexing structures to store the spatio-temporal information obtained in the first step. iii) Data analytics are able to identify and extract useful patterns, such as clusters and outliers. This step benefits from the indexing structures done in the previous step. iv) The service providing goal is to interpret the obtained information and send it to the transportation authority for dispersing traffic and diagnosing anomalies. Chen et al. [15] reviewed existing flow outlier detection approaches, including the statistics-based approach, the distance-based approach, and the density based local outlier approach. Moreover, a comparative study was performed using Nanjing urban traffic data by considering two dimensions: travel time and traffic flow data. The results revealed that classical outlier detection algorithms were useful in detecting urban traffic flow outliers. Kiran et al. [16] presented existing works dealing with trajectory outliers in urban traffic data. They classified the existing trajectory outlier detection approaches according to the method used in the processing step. The approaches used for classification were distance-based, density-based, and motifs-based outliers. Djenouri et al. [17] sketched some existing urban traffic flow outlier detection algorithms by analyzing locality notion proposed in [10].

Compared to the existing surveys, ours is the first one that does so comprehensively. All the other works have been limited to only some categories of outlier detection

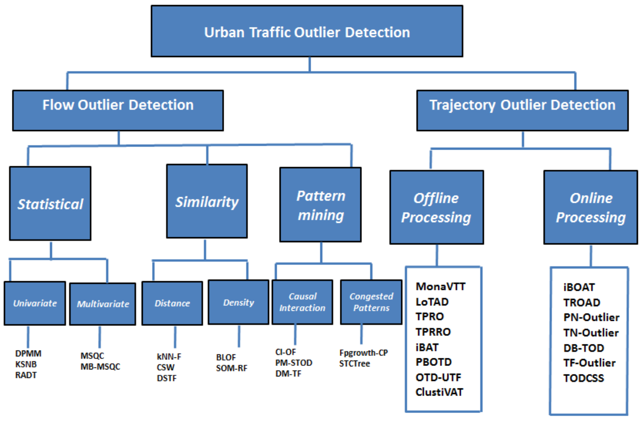


Figure 1. Taxonomy of urban traffic outlier detection algorithms.

(statistical, similarity, or pattern mining), or even to some category of urban traffic outliers (flows or trajectories). Compared to those dealing with flow outliers, ours differs by dealing with both flow values and segment flow values. Compared to those dealing with trajectory outliers, ours differs by dealing with both trajectories and sub-trajectories.

B. TAXONOMY AND PAPER ORGANIZATION

Figure 1 outlines the taxonomy of the urban traffic outlier detection algorithms presented in this paper. They are separated into two categories. Flow outlier detection aims to identify outliers from urban flow data, including 1) statistical, 2) similarity, and 3) pattern mining methods. The second category is trajectory outlier detection where clustering and similarity approaches are used to derive outliers. This includes trajectory outliers using offline processing and sub-trajectory outliers using online processing. Based on this taxonomy, the rest of the paper is organized as follows: Section 2 defines the background and concepts used in the paper. Section 3 presents approaches related to flow outlier detection algorithms. Section 3.1 presents the statistical approaches, Section 3.2 presents the similarity approaches, Section 3.3 presents the pattern mining approaches, and Section 3.4 gives an illustration of flow outlier detection algorithms. In Section 4, we show the trajectory outlier detection algorithms, and offline processing is presented in Section 4.1, online processing presented in Section 4.2, and Section 4.3 gives an illustration of trajectory outlier detection algorithms. Section 5 discusses the merits and limitations of the works presented in this paper. Section 6 concludes the paper and points out a few future directions for research.

II. PRELIMINARY

Before reviewing the existing approaches, we first introduce some basic definitions in urban traffic analysis.

Definition 1 (Flow Values): Consider the set of flow values $F = \{F_1, F_2, \dots, F_{|F|}\}$, each F_i contains the number of objects captured in the time stamps $[i, \dots, i+1]$.

Definition 2 (Road Network): A road network is modeled as a directed graph $G = (V, E)$, where V refers to the vertex set representing crossroads and E refers to the edge set representing road segments. Each edge $e_i^j \in E$ denotes a road segment from vertex v_i to v_j.

Definition 3 (Segment Flow Values): We consider a segment flow value $S^{o,d}$ to represent the flow values between the site o and the site d for a given time period. We also define the set of all segment flow values $S = \{S^{o,d} \mid \forall (o,d) \in V^2\}$, where V is the set of sites defined in Def. 2.

Definition 4 (Trajectory): A raw trajectory is a list of point locations $t = \{p_1 \rightarrow p_2 \ldots \rightarrow p_n\}$ with time stamps, obtained by localization techniques such as GPS. n is the number of sampled points in the trajectory. Each point p_i is composed by $< x_i, y_i, ts_i >$, where x_i is the longitude position, y_i is the latitude position, and ts_i is the time stamp that the trajectory t attends to point p_i for all $i \in [1, \ldots, n]$. We also note $T = \{t_1, t_2, \ldots, t_{|T|}\}$ as the set of all trajectories captured in the given time period.

Definition 5 (Sub-Trajectory): Consider a trajectory $ST = \{p_{i1} \rightarrow p_{i2} \ldots \rightarrow p_{im}\}$ that is defined as sub-trajectory of t in Def. 4, and denoted as $s \preceq t$, where $1 \leq i_1 < \ldots \leq n$. We also note $S = \{s_1, s_2, \ldots, s_{|S|}\}$ as the set of all trajectories captured in the given time period.

Definition 6 (Outlierness): Let $D = \{d_1, d_2, \ldots, d_{|D|}\}$ be the dataset of a given problem: (F for flow values, S for segment flow values, T for trajectories, and ST for sub-trajectories).

Consider a *score* function, defined as follows:

$$Score\colon D \rightarrow R \tag{1}$$

$$d_i \mapsto Score(d_i). \tag{2}$$

The outlier and the inlier sets are defined as follows:

$$\begin{cases} O = \{d_i \mid \forall d_j \in I,\ Score(d_i) \geq Score(d_j)\} \\ I = D/O. \end{cases}$$

Consider the flow values of Figure 2. This figure shows the flow values of the Anderupvej location at the city of Odense in Denmark on a Monday (17/04/2017). Each flow was determined every 15 minutes. The flows marked by small red circles are considered outliers.

III. FLOW OUTLIER DETECTION ALGORITHMS

Figure 3 shows the overall framework of the existing flow outlier detection algorithms. Flow outlier detection algorithms are generally composed of four main steps: i) Urban sensing and data acquisition: This first step aims to capture the data related to urban traffic flow by deploying sensors, GPS devices, and other IT infrastructures. ii) Data collection: The captured data is then collected and organized according to the application used (flow values or segment flow values). iii) Outlier detection: The data collected are then used to find outliers. Three main approaches have been used to find outliers: statistical, similarity, and pattern

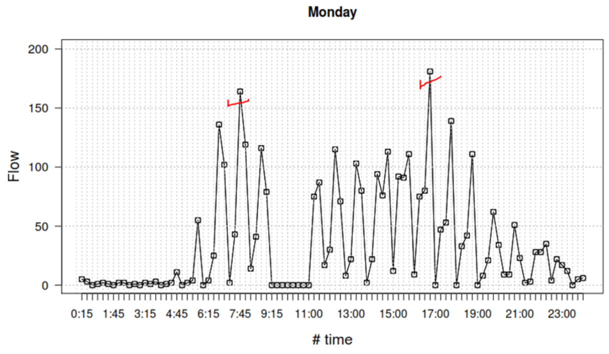


Figure 2. Outlierness illustration in Odense, Denmark.

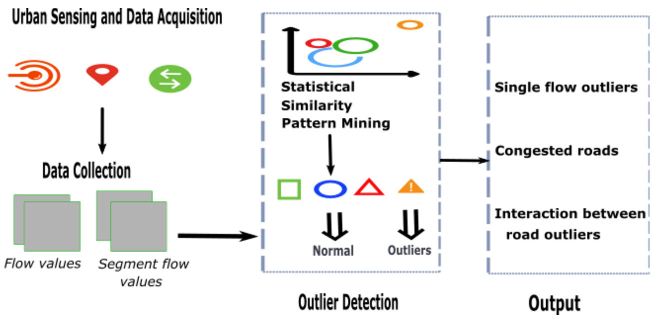


Figure 3. Flow outlier detection framework.

mining. iv) Output: Three kinds of outliers could be identified by the existing flow outlier detection algorithms: single flow outliers, congested roads, and interaction between road outliers. This section reviews existing flow outliers detection algorithms.

A. STATISTICAL APPROACHES

Statistical analysis models such as the Gaussian aggregation model [18], principle component analysis [19], stochastic gradient descent [20], and Dirichlet Process Mixture [21], are based on the fact that in general, inlier flows follow some statistical process represented by an alternative hypothesis and the outlier flows deviate from this statistical mechanism and respect to the null hypothesis. For detecting outliers in large-scale urban traffic data, Ngan et al. [22] proposed a Dirichlet Process Mixture Model (DPMM). The set of all flow values $F = \{f_1, f_2, \ldots, f_{|F|}\}$ is projected into a n-dimensional space, where the i^{th} dimension is defined by the flow values $\{f_i, \ldots, f_{i+w}\}$, where w is the window length projection such as $(1 \leq w \leq |F|$ and $n = |F| + w)$. The n dimensions are then entered in a Principal Component Analysis (PCA) kernel to reduce and transform the traffic data space into a two-dimensional (2D) (x, y) coordinate plane. In this step, the covariance matrix among the variables of the n dimensions is computed, and the Eigenvalues are then determined and sorted from the highest to the lowest. This provides the dimensions in the order of significance. The two highest significance dimensions are considered while the rest are ignored. The obtained flow vector represented by the two dimensions is injected into the Dirichlet process to detect flow outliers. Thus, the clusters are estimated using $G \sim DP(\mathcal{H}, \alpha)$, with α being the concentration parameters, and $\mathcal{H}$ is the hypothesis base distribution defined by $\mathcal{H} = \{\phi.\vec{\mu}\}$, ϕ, and $\vec{\mu}$ are the mixture density covariance and the mixture weights of the data, respectively. The clusters with a high number of flow values are considered as normal, and the other clusters are labeled as outliers. Lam et al. [23] proposed a Kernel Smoothing Naive Bayes (KSNB) approach to automatically determine any errors as well as abnormal traffic in data from Hong Kong. The authors assumed that inlier flow values followed a kernel smooth distribution. The KSNB model automatically determines regions formed by kernel distributions and then considers them as inliers. In contrast, any flow value outside of those regions is considered to be an outlier. The kernel estimator for the set of flow values F is defined as $\frac{1}{|F|}\sum_{i=1}^{|F|} K(F - F_i)$, with $K(F) = \frac{1}{2\phi}e^{-0.5F^2}$. Kingan et al. [24] presented a Regression model for Average Daily Traffic (RADT). A set of flows F represented by annual average daily traffic was constructed. The best-fit-least-squares line through the set of flows F, and during the time T, was given as $\hat{F} = mT + b$, where $m = \frac{\sum_{i=1}^{|F|}(T_i - \bar{T})(F_i - \bar{F})}{\sum_{i=1}^{|F|}(T_i - T)}$ and $b = \bar{F} - m\bar{T}$, where $\bar{F}$ and $\bar{T}$ are the average of the flows $(\frac{\sum_{i=1}^{|F|} F_i}{|F|})$ and the average of the time $(\frac{\sum_{i=1}^{|F|} T_i}{|F|})$, respectively. The standard deviation was then computed by $sd = \sqrt{\sum_{i=1}^{|F|}(\hat{F} - F)^2}$. The score of each flow value F_i was determined by the function $Score(F_i) = \sqrt{[\hat{F} - (\hat{F/F})]^t T^t T[\hat{F} - \hat{F_i}]}$. If the score of F_i is greater than 1, then it is considered an outlier, otherwise, it is considered a normal flow value.

Turochy et al. [25] proposed a Multivariate Statistical Quality Control (MSQC) approach for traffic congestion outlier detection. This approach took other traffic variables that contributed to the congested case, such as the average speed, and the occupancy rate, instead of using a single variable represented by the flow values. For more details about how to compute these variables, we refer the readers to [26]. The historical traffic flows T are fitted to the F-distribution $\mathcal{F}_{|T|,|T|-p}(\alpha)$, where p is the number of variables (in this case is set to 3), and α is the confidence significance level. When the new observation flows $x = \{x_{flow}, x_{speed}, x_{occupancy}\}$ are detected, T with the corresponding x are projected to the F-distribution with the α value. If the alternative hypothesis is accepted then x is considered as normal flow, otherwise, the score of x is computed as: $Score(x) = (x - \bar{x})^T S^{-1}(x - \bar{x})$ where S is the covariance matrix defined by $(x - \bar{x})(x - \bar{x})^t$. If the score is greater than the cutoff threshold, then x is considered to be an outlier, otherwise, it is a normal flow. Park et al. [27] proposed a Multiple Blocks on Multivariate Statistical Quality Control (MB-MSQC) approach to deal with the variability problem of flow during the hours of the day. For example, in almost all urban cities, there is an increase in traffic between 6:00 to 9:00 and 16:00 to 19:00. Thus, the set of flow values is grouped into five distinct blocks: (B_1: 00:00 to 6:00, B_2: 6:00 to 9:00, B_3: 9:00 to 16:00, B_4: 16:00 to 19:00, and B_5: 19:00 to 00:00). Afterwards, the MSQC proposed in [25] is independently applied on each block of flows. This algorithm has been tested on traffic data from San Antonio and Austin, USA. According to the authors, the results revealed the superiority of MB-MSQC compared to MSQC in terms of precision.

B. SIMILARITY APPROACHES

The approaches in this section use distance measures and neighborhood computation methods to find outliers [3], [4]. In general, the inlier flows produce dense regions whereas the outlier flows have less dense neighborhoods. Dang et al. [28] proposed a kNN-based approach for flow outlier detection named kNN-F. It adapts the kNN outlier algorithm presented in [3]. As input, it has the set of flows, the number of neighborhoods k, and the ϵ threshold. It also uses an internal data structure represented by a vector $dist$ to store the distance values. First, the distance between each two pairs of flow values is determined. The distance value between each flow f_i and its k^{th} nearest neighbor is then selected. If this value exceeds the ϵ threshold, then f_i will be considered to be an outlier, otherwise, it will be considered an inlier. The same authors proposed density-based bounded LOF (BLOF) [29]. This is an adapted version of the LOF algorithm [4]. It has as input the set of flow values and the number of neighbors k. It also uses an internal data structure represented by a vector kNN to store the k nearest neighbors of each flow value f_i. First, the local reachability density (lrd) of each f_i (see Eq. 3) is calculated. Second, the k nearest neighbor of f_i is given and stored in the kNN vector. Then, the sum of all *lrd* of all neighbors of f_i over the *lrd* of f_i is calculated. The LOF value is determined using Eq. 4. If this value exceeds 1, then f_i is considered an outlier, otherwise, it is considered an inlier.

$$lrd_k(p) = 1/(\frac{\sum_{o \in kNN(p)} reach_k(p, o)}{k}) \tag{3}$$

Note that $reach_k(p, o) = max\{d(p, o), d_k(o)\}$, $d(p, o)$ is the distance between the flow values p and o and $d_k(o)$ is the distance between the flow value o and its k nearest neighbor.

$$LOF_k(p) = \frac{1}{k} \times \sum_{o \in kNN(p)} \frac{lrd_k(o)}{lrd_k(p)} \tag{4}$$

Munoz-Organero et al. [30] proposed a distance-based algorithm called Center of Sliding Window (CSW) to detect abnormal driving locations caused by a particular traffic conditions such as traffic lights, street crossings, or roundabouts. The aim was to filter outlier driving points related to random traffic conditions such as traffic jams from infrastructural road elements. The sliding windows of n flow vectors with their speed and acceleration are created, and the center flow vector $\bar{f_i}$ in each sliding window sw_i is considered as reference flow. The Mahalanobis distance is used to compute the similarity between the j^{th} flow vector f_i^j and the center flow vector $\bar{f}_i$ for the sliding window sw_i as $d(f_i^j, \bar{f}_i) = \sqrt{(f_i^j - \bar{f}_i)^T Cov^{-1}(f_i^j, \bar{f}_i)}$ where $Cov(f_i^j, \bar{f}_i)$ is the covariance matrix of the vectors f_i^j and $\bar{f}_i$. Single flows are captured each second for 20 seconds. Then, dense flows with high similarity values are considered inliers and the others are detected as outliers. Shi et al. [31] proposed a dynamic spatio-temporal approach called Dynamic Spatial and Temporal Flows (DSTF) to detect local anomalies in spatio-temporal flow data. The flows are captured by considering the direction flows of the set of segment roads S. Each road segment s_i is denoted by (v_i^1, v_i^2, l_i), where v_i^1 is the starting point of s_i, v_i^2 is the ending point of s_i, and l_i is the length of s_i. The spatial neighbors of s_i, denoted as $SN(s_i)$ are deducted as $SN(s_i) = \{s_j | v_i^2 = v_j^1\}$. The dynamic neighborhood structure is then designed by computing the similarity between the spatio-temporal flows of each road segment s_i and each element s_j on its spatial neighbors SN_i by:

$$\begin{cases} |avg_i - avg_j| & |avg_i - avg_j| > \epsilon \\ \epsilon & Otherwise \end{cases}$$

where avg_i is the average unit journey flow of vehicles through the road segment s_i and ϵ is the similarity threshold. The kNN outlier is then used to detect segment road outliers, where k here represents the cardinality of each spatial neighborhood set $SN(s_i)$.

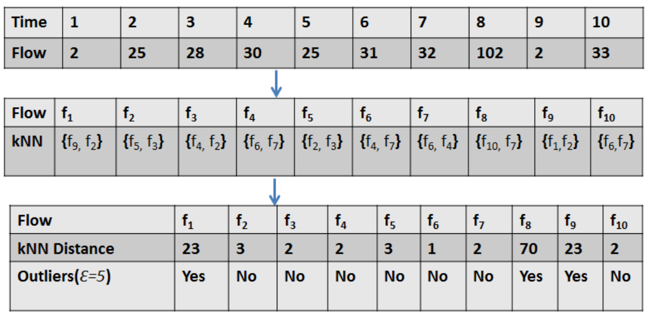
Cheng et al. [32] presented a Self-Organizing Map for Road Flows (SOM-RF). The flows of each road data from Bejing City were collected and transformed to a time series. The Self-Organizing Map (SOM) algorithm [33] is then used to cluster road flows into groups. The weights in the input layer of the map are initialized with regards to the input pattern X, where each x_i represents the time series of the i^{th} road flow. The distance between the input patterns X and the weight w_j is determined as $d_j = ||x - w_j|| = \sqrt{\sum_{i=1}^{n}(x_i - w_{ij})^2}$. At each step of the algorithm, the winner with the minimum distance is selected and considered an inlier. Afterwards, the weights connecting the input layer to the winning node and its neighborhoods are updated for the next iteration *t+1* according to the following learning rule, $w_{ij}(t+1) = w_{ij}(t) + \alpha(t)[x_i - w_{ij}(t)]$. $\alpha(t)$ is the neighborhood size decreased with the iteration algorithm t. This process is repeated until weights have stabilized. All winning nodes are considered inliers, and the others are outliers.

C. PATTERN MINING APPROACHES

The aim of pattern mining approaches such as Apriori [34] and FP-growth [35] are to analyze the flow values. Other useful data structures are extracted in the pre-processing step, such as the flow segment, the segment-route matrix, traffic volumes, jams, and incidents. The aim is to extract relevant patterns form these different urban traffic data variables. The process starts by transforming the urban traffic database into a transactional database $T = \{t_1, t_2, \dots, t_m\}$, and the possible urban traffic variables into the set of items $I = \{i_1, i_2, \dots, i_m\}$. A pattern P is a subset of I and it is said to be of size k if it contains k items. The relevance of a given pattern is calculated using different measures: support [34], and interestingness [36]. The support of a pattern P is the ratio between the number of transactions containing P and the number of all transactions $|T|$ without a loss of generality. The frequent pattern mining-based problem consists of extracting all frequent patterns from T, i.e, enumerating all patterns having a support that is no less than a user-defined $minsup$ threshold. The discovered patterns are then used to find anomalies such causal interaction, congested patterns, hot spot detection, and so on. In the following, we present the most relevant works of urban traffic outlier detection using the pattern mining process, including two real applications: causal interaction and congested pattern.

Liu et al. [37] introduced the problem of causal interaction in urban traffic data, i.e., the discovery of relationships among the detected outliers. The authors proposed a new algorithm called Causal Interaction in Outlier Flows (CI-OF). Flow segments are first created from the urban traffic database. Each segment that relates the site origin o and the site destination d at time window w_i is represented by $< S_i^{o,d}, s_i^1 = \frac{S_i^{o,d}}{S_i^{o,*}}, s_i^2 = \frac{S_i^{o,d}}{S_i^{*,d}} >$. $S_i^{o,d}$ represents the flow value between the site o and the site d at time window w_i. $S_i^{o,*}$ is the sum of all flow values at window w_i by considering the site o as the origin. $S_i^{*,d}$ is the sum of all flow values at window w_i by considering the site d as the destination. The distortion function is then computed between all segment flows in all time windows $w = \{w_1, w_2, \dots, w_n\}$ to find temporal outliers for two sites o, and d as $Distort(o,d,w_i,w_j) = \sqrt{(S_i^{o,d} - S_j^{o,d})^2 + (s_i^1 - s_j^1)^2 + (s_i^2 - s_j^2)^2}$. The score of the flow segment is determined as $Score(o,d,w) = \sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n}(\frac{Distort(o,d,w_i,w_j) - min}{max})$, where $min = min\{Distort(o,d,w_i,w_j) \mid i = [1,\dots,n], j = [1,\dots,n]\}$, and $max = max\{Distort(o,d,w_i,w_j) \mid i = [1,\dots,n], j = [1,\dots,n]\}$. Segment (o,d) is considered an outlier if the score is greater than a given threshold. Afterwards, the outlier trees are built, where each tree contains flow segments from top k outliers for different time windows. For a given tree, a node x is a child of node y if and only if the segment (x,y) belongs to the top k outliers. Association rule mining is then applied to find frequent sub-structures from the forest of the outlier trees. Each tree is considered to be one transaction, where the nodes of the trees are considered items. The association rules discovered represent the different causal interactions between the extreme sites. Pang et al. [38] developed the Pattern Mining for Spatio-Temporal Outlier Detection (PM-STOD) approach. In this method, the city is partitioned into grids and the number of taxis is recorded for each grid cell by using the GPS device. The Likelihood Ratio Test (LRT) [39] is used to identify outlier regions, where each region is composed of the adjacent grid cells. Afterwards, a pattern mining approach is proposed to study the interaction between the outlier regions. Regarding the patterns discovered from the outlier regions, two kinds of outliers can be identified: emerging outliers and persistent outliers. The flow value of each grid cell in the emerging region outliers is greater than the flow values of all neighborhood regions. However, the flow value of each grid cell in the persistent region outliers is higher than the flow values of the remaining regions. The flow values of the persistent outlier regions are increased over time and an upper bounding strategy for both outliers are derived. According to the authors, using Beijing taxi data, PM-STOD was able to detect regions with emerging and persistent outliers. Chawla et al. [40] suggested DM-TF, a data mining-based approach to detect anomalous behaviors from the traffic flow. This approach focuses on analyzing traffic between regions, rather than the entire flows. It builds two matrices. The first one is a segment-route matrix $A(m \times r)$ with m segment flow and r routes. Each entry $A_{i,j}$ is equal to 1 if the segment flow i is across the route j, otherwise, it is 0. It then builds matrix segment flow noted $L(m \times n)$, where each row L_i represents the segment flow values of each segment i during a time window $w = \{w_1, w_2, \ldots, w_n\}$. Patterns describing routes that have caused anomalies are extracted by applying the pattern mining approach presented in [41] on the two matrices A and L. This approach has been tested on a real world Beijing transportation dataset. The results revealed that this strategy reduces the computational cost of the causal interaction model. In addition, it can identify the most important routes that cause abnormal traffic. To understand the traffic congestion in urban cities, Inoue et al. [42] presented an FP-growth for Congested Patterns (FP-growth-CP) algorithm for discovering frequent patterns. The traffic data is considered to be a transactional database, where each row traffic data represents one transaction, and each variable flow, such as flow values, speed, and densities, represents one item. The state of the congestion (true or false) is also added to each row in the transactional database. Afterwards, the FP-growth algorithm is adopted by extracting only the closed frequent patterns that efficiently represent the transactional database. The aggregation function is also used to extract only the association patterns with only the congestion state as a consequence part. This algorithm has been tested on traffic data from Okiwara, Japan. The results indicated that higher dependencies exist between speed-density and speed-flow variables and the congested state value. The results also revealed that by using the aggregation function and the closed property, only relevant patterns could be discovered with reduced computational time. Nguyen et al. [43] proposed an STCTree algorithm inspired by frequent pattern mining to predict frequent spatio-temporal congested sites and causal relationships among them from traffic data streams. In this algorithm, an efficient tree structure is employed that is permitted to deal with a large traffic network with respect to the data stream processing constraint. The algorithm starts by building a forest of congested sites where each tree in the forest is presented by a list of connected congested sites. If the congestion time of s_1 succeeds the congestion time of s_2 then the site s_1 is a child of the site s_2. It should be noted that the given site is considered to be congested if the flow values in this site are greater than a user's threshold for a specified time. Afterwards, the sites on each tree are considered as an item sets, and the Apriori algorithm is then applied to determine frequently congested sites. The i^{th} transaction is represented by the i^{th} tree, and the items of this transaction are the congested sites that belong to the associated tree. The frequently congested sites that are discovered of all trees are used to determine the frequent trees of the congested site. Thus, two trees are combined if there are frequently congested sites belonging to both trees. This process is repeated until no possible combinations can be made. In the end, two possible patterns are identified: i) The set of frequently congested sites in the urban network traffic, and ii) the set of all the frequent trees of congested sites captured in the successive time period.

D. ILLUSTRATIVE EXAMPLE

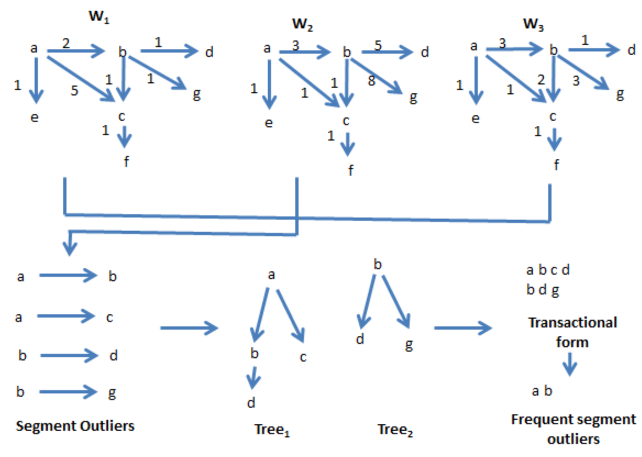


Time	1	2	3	4	5	6	7	8	9	10
Flow	2	25	28	30	25	31	32	102	2	33

Flow	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
kNN	$\{f_9, f_2\}$	$\{f_5, f_3\}$	$\{f_4, f_2\}$	$\{f_6, f_7\}$	$\{f_2, f_3\}$	$\{f_4, f_7\}$	$\{f_6, f_4\}$	$\{f_{10}, f_7\}$	$\{f_1, f_2\}$	$\{f_6, f_7\}$

Flow	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
kNN Distance	23	3	2	2	3	1	2	70	23	2
Outliers($\mathcal{E}$=5)	Yes	No	No	No	No	No	No	Yes	Yes	No

(a) kNN-F illustration



(b) CI-OF illustration

Figure 4. Flow outlier detection illustration.

In this section, we show how the flow outlier detection algorithms work. Two algorithms in particular will be illustrated. The first one is kNN-F [28], which focuses on single flow outliers. The second one is CI-OF [37], which focuses on the interaction between road outliers. Starting with kNN-F, shown by Figure 4(a), the k neighbors (k is set to 2) of each flow are returned by computing the Euclidean distance between the given flow and all the remaining flows. The score of each flow is its distance with the k^{th} neighbor. Finally, the flow outliers are returned by setting the ϵ threshold to 5. The flows outliers are $\{f_1 = 23, f_8 = 70, and f_9 = 23\}$. Concerning the second algorithm (CI-OF), as illustrated by Figure 4(b), the algorithm starts by constructing the segment flows during different time windows W (the window size is set to 3). The score of each segment flow is computed and only the segment flow outliers are returned: $a- > b$, $a- > c$, $b- > d$, and $b- > g$. Two trees are built from these outliers, and the first tree propagates the segment flow outliers initially started in node a. The second tree propagates the segment flow outliers initially started in the node b. From these two trees, frequent pattern mining is performed to extract the frequent segment outliers by considering the minimum support, set to 100%. The resulted frequent patterns are $\{ab\}$. That means segment $a- > b$ causes the other segment flow outliers. Thus, if we are to analyze the flows in the road network illustrated by Figure 4(b), we first have to analyze the flows between nodes a and b.

IV. TRAJECTORY OUTLIER DETECTION ALGORITHMS

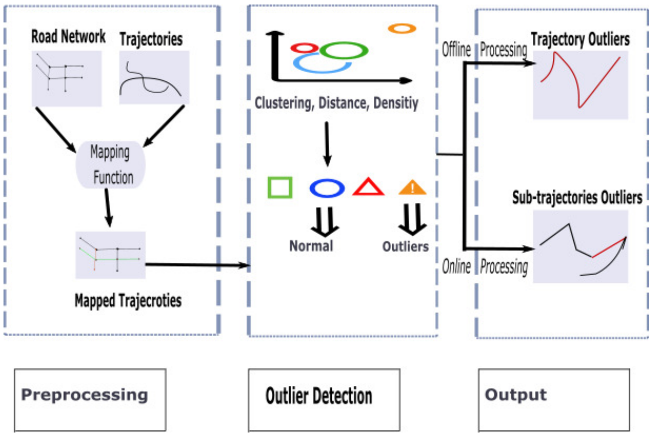


Figure 5. Trajectory outlier detection framework.

Figure 5 presents the overall framework of the existing trajectory outlier detection algorithms. Generally, the trajectory outlier detection algorithms are decomposed into three main steps: i) Preprocessing: Preprocessing is aimed at collecting the trajectories database and information related to the road network of the urban city. The mapping function allows generation of the mapped trajectory database. ii) Outlier detection: The mapped trajectory database is entered to the outlier detection algorithms, including clustering, density, and distance approaches, to find trajectory outliers. iii) Output visualization: After finding outliers, visualizing tools are needed to show the trajectory outliers to the user. In this context, two kinds of outliers are discovered. Only the trajectory outliers could be detected by applying offline processing. However, by applying online processing, the sub-trajectory that caused outlierness may be identified. In this section, both offline and online approaches will be reviewed.

A. OFFLINE PROCESSING

Zhang et al. [44] proposed a graph-based method, called MoNav-TT, for detecting two levels of taxi trip outliers in a large scale urban traffic network using NAVTEQ street map and the MoNav algorithm [45], that implements an efficient Contraction Hierarchy based on the shortest path computation algorithm and a spatial join algorithm to snap pickup and drop-off locations. Given a taxi trip database T, each row in the database represents one taxi trip that contains the following features: pickup location, pickup time, drop-off location, drop off time, and the recorded distance of the trip by the taxi driver. We also arrange a street network, called S, with N nodes and M edges. The method follows two stages: The first stage matches both the pickup and drop-off locations of each trip to their nearest street segments by computing the similarity between the trip features (the pickup and drop-off locations) and the street network S. The taxi is considered a Level I outlier if the distance value is greater than distance threshold D. The remaining trips are assigned to the node in the closer pickup or drop-off node in the street network S. The second stage computes the shortest paths using the MoNav algorithm for each unique pickup up and drop-off node pair. The computed shortest path distances are then compared with the recorded distances. If the computed distances are greater than W times longer than the recorded distances, then the trip is marked as Level II outlier. The algorithm has been tested on 166 million taxi trips in New York City (NYC). By setting D=200 feet and W=2 in MoNav-TT, among the 166 million taxi trip records, approximately 2.5 million (1.5%) pickup or drop-off locations could not be matched to a street segment of the NAVTEQ street map dataset and were identified as Level I outliers. While the majority of these outliers could be induced by GPS device errors, some of them may be associated with the incompleteness of street networks, e.g., picking up and dropping off at private land parcels. Similarly, 18,000 were identified as Level II outliers.

Kong et al. [46] proposed a long-term traffic anomaly detection (LoTAD) approach. This method consists of the following steps: i) TS-segments Creation: The aim of this step is to create the TS-segments database from both the bus trajectory and the bus station line databases. Each bus line bl_k is represented by a matrix M^k, where each element $a^k_{i,j}$ is a couple $(x_{i,j}, y_{i,j})$ that denotes the average velocity and average stop time, respectively, at road segment TS_i during the time slot t_j belonging to $[j\theta, (j+1)\theta]$, where θ is the duration of each time slot (θ is fixed to one hour by the authors). The average velocity $x_{i,j}$ is calculated by $x_{i,j} = \frac{\sum_{tr \in TS_i}(W_i \times tr.x)}{|TS_i|}$. tr is all the trajectories that belong to the same TS-segment. $tr.x$ is the velocity of the trajectory tr. W_i is the weight coefficient assigned to the road segment TS_i. This weight represents the importance of the road segment TS_i in the set of all road segments (for instance, it represents the number of buses that cross the road segment TS_i). $|TS_i|$ is the number of trajectories that belong to the road segment TS_i. The average stop time $y_{i,j}$ is calculated by $y_{i,j} = \frac{\sum_{tr \in TS_i}(\frac{\sum_{p \in tr} p.ts}{|tr|})}{|TS_i|}$. $|tr|$ is the number of points in the trajectory tr. $p.ts$ is the stop time in the point p. ii) Anomaly Index Computation: In this step, the Anomaly Index (AI) of the road segments is derived. The density of each road segment TS_i is first computed by $density_i = \sum_{l \neq i, k \neq i} e^{-(\frac{d_{l,k}}{d_i})^2}$. It should be noted that $d_{l,k}$ is the Manhattan distance [47] between the road segments TS_l and TS_k. d_i is the Manhattan distance between the road segment TS_i and all the remaining road segments. The LOF algorithm is then applied by computing the *lrd* and *LOF* values of each road segment based on the density values determined above. Instead of classical LOF that fixes 1 as the anomalies threshold, here, the threshold is fixed by the average of the sum of all densities of all road segments. In the end of this step, the set of anomaly road segments is obtained. iii) Traffic Anomaly Regions: In this step, the regions are first extracted by applying the k-means algorithm [48] on the bus station line database, where each cluster is considered a region. Based on the previous step, the anomaly score of each region is determined as $Score(r_i) = \sum_{TS \in r_i} LOF(TS)$. The scores obtained are sorted in descending order, and the top n regions having high scores are considered as outliers, where n is the user parameter.

Zhu et al. [49] proposed Time-dependent Popular Routes based trajectory Outlier detection (TPRO). It finds time-dependent outliers by using the popular routes for each time interval. The popular routes are first retrieved, where each popular route r_i has a weight w_{it} that represents its popularity during the given time interval $\delta t = [t, t+1]$. The trajectories dataset Σ is then divided into groups $G = \{G_1, G_2, \dots, G_k\}$, where each group contains trajectories having the same source and destination points in the given time interval δt. Formally, we obtain that: $\forall(\Sigma_i, \Sigma_j) \in \Sigma^2, (P_1^i.S_1 = P_1^j.S_1) \wedge (P_{l_i}^i.S_{l_i} = P_{l_j}^j.S_{l_j}) \wedge ((P_{l_i}^i.T_{l_i} - P_1^i.T_1) \in \delta t) \wedge ((P_{l_j}^j.T_{l_j} - P_1^j.T_1) \in \delta t) \Rightarrow G_{\Sigma_i} = G_{\Sigma_j}$ Σ_i is a time-ordered sequence of road network locations $(P_1^i, P_2^i, \dots, P_{l_i}^i)$, such that l_i represents the number of locations of the trajectory $Sigma_i$, and each location P is represented by (S, T). S is a spatial coordinate and T is the sampling time. G_{Σ_i} is the group ID of the trajectory Σ_i. The representative trajectory, noted $\bar{G_i}$ of the i^{th} group G_i, is then compared with the top l popular roads noted $R = \{r_1, r_2, \dots, r_l\}$ using the edit distance during the interval time δt as $Score(\bar{G_i}, R) = \sum_{j=1}^{l} w_{jt} \times edit(\bar{G_i}, r_j)$. $edit(\bar{G_i}, r_j, n) = min\{edit(\bar{G_i}, r_j, n-1\} + MC(\bar{G_i}, r_j, n, 0), MC(\bar{G_i}, r_j, n)$ is the mapping cost of the point P_n^i in the trajectory $\bar{G_i}$ to the point P_n^{rj} in the road r_j. If the score of the representative trajectory $\bar{G_i}$ is greater than θ threshold, than all trajectories of the cluster G_i are considered outliers. An improved version of TPRO, Time-dependent Popular Routes based Real-time trajectory Outlier detection (TPRRO) is proposed in [50]. The aim of the TPRRO is to detect outlier trajectories from new trajectories set Σ^{new}. It employs efficient data structures called Time-dependent Transfer Index (TTI) to record which trajectory has passed through which location at which time. It maps each new trajectory Σ_i^{new} on the grid-partitioned road network. It builds a B-tree like structure called transfer B-tree in each grid. Transfer B-tree records which trajectory has passed through this grid at which time period. TPRRO is able to map in real time the new trajectories on the grid of road network using TTI. Instead of computing the similarities between the top k popular routes as in TPRO algorithm, only the similarity between each Σ_i^{new} and the top l popular grids of the road network are calculated to determine the score of Σ_i^{new} using the same score function as TPRO.

Zhang et al. [51] proposed the Isolation-Based Anomalous Trajectory (iBAT) algorithm. It is composed of two main steps: pre-processing and anomaly detection. i) Pre-processing step: It first constructs the taxi trajectories from GPS traces, divides the city map into grid-cells of equal sizes, and groups all trajectories crossing the same source-destination cell-pair, where each trajectory is represented by the set of sequences of traversed cells; Then it builds frequently traversed cells using an inverted index mechanism [52]. ii) Anomaly detection step: Instead of using a distance or density measure, the "few and different" properties of anomalous trajectories are exploited. By exploring different locations or same locations with different orders, anomalous trajectories are *few* in number and *different* from the majority. The idea is to attempt to find a separate way for anomalous trajectories from the rest of "many and similar" trajectories by applying the adapted Isolation Forest (iForest) [53]. A random tree is generated by dividing the trajectories until almost all of them are isolated. This generation produces a shorter path for anomalous trajectories which are isolated faster than normal trajectories isolated in a longer path. The outlier score of each trajectory t is computed as $Score(t) = 2^{-\frac{N}{c(N)}}$, where N is the number of cells used for isolating t, and $c(N)$ represents the path length of unsuccessful searches in a binary search tree and it is computed as $C(N) = 2H(N-1) - 2(N-1)/N$. Note that $H(i)$ is the harmonic number that can be estimated as $ln(i) + 0.57727566$ (Euler's constant).

Zhongjian et al. [54] proposed a Prototype Based Outlier Detection (PBOTD) approach with the aim of understanding the historical trajectory database for identifying outlier taxi trajectories. The set of routes R is first grouped using the medoids algorithm [55]. Choosing medoids instead of k-means is due to the difficulty of computing the mean trajectories. The k initial centers are determined using the selection function S, such as $S(r_j) = \sum_{i=1}^{|R|} \frac{d(r_j, r_i)}{\sum_{l=1}^{|R|} d(r_i, r_l)}$, where $d(r_i, r_j)$ is the *edit* distance between two routes r_i

and r_j [56]. The top k routes that minimize the function S are considered as initial centers of the clusters. As in medoids, the routes are assigned to the nearest center and the sum of distances is calculated from all routes of the same cluster to update the center of each cluster by the route having a minimum value. This process is repeated until the sum of distances from all routes to their centers does not change. After the clustering step, the set of the centers of the clusters are considered as Representative Routes: $RR = \{r_1, r_2, \dots, r_k\}$. For each new trajectory t, its score is computed based on routes in RR as $Score(t) = min\{d(r_i, t) | r_i \in RR\}$. $d(r_i, t)$ is the *edit* distance between route $_i$ and the trajectory t. If the score of t is greater than a similarity threshold, then the trajectory t is an outlier, otherwise, it is considered a normal trajectory.

Zhou et al. [57] proposed Outlier Trajectory Detection approach for identifying Unmetered Taxi Frauds (OTD-UTF). A taximeter database is collected where each record is a tuple $< id, st, et >$. id is the TaxiID, st is the start time of the metered trip, and et is the end time of the metered trip. The trajectory database is matched to identify whether each point in the trajectories database is a metered or unmetered point. Thus, a point p_j^i of the trajectory t_i is metered if there is an entry in the taximeter database where $id = i$ and $time(p_j^i) \in [st, et]$. A trajectory t_i is a fraud trajectory if and only if for each point $p_j^i \in t_i$ is an unmetered point. The process starts by finding the trajectory outliers using a stochastic gradient model [58]. From the trajectory outliers, the fraud trajectories are identified by matching each point in the trajectory outlier with the taximeter database. OTD-UTF has been tested on real big databases including 154 million taxi records from a large city in China. The results revealed that the trajectory frauds are due to huge demands of taxis in congested areas, where taxis could be bringing more than one passenger for the same trip.

Kumar et al. [59] developed the Clustering for Improved Visual Assessment Tendency (ClustiVAT) approach to detect trajectory outliers. The clustering of trajectories is performed using the iVAT algorithm [60] by proposing a two-stage clustering procedure. The first step uses a non-directional similarity measure to group the trajectories according to which trajectories follow similar paths but have opposite starting and/or ending points, and assigned them to the same group. The second step uses directional similarity for each cluster generated to separate the trajectories going in opposite directions. From the clusters of trajectories, the set of trajectory outliers are obtained by identifying trajectories that are too far from other trajectories in the same cluster, or by identifying clusters that have too few a number of trajectories.

B. ONLINE PROCESSING

Chen et al. [61] proposed Isolation-Based Online Anomalous Trajectory (iBOAT) algorithm. This algorithm aims to find anomalous taxi sub-trajectories in real time. It is used to automatically detect fraud implications by rapacious taxi drivers who take unnecessary detours during trips. When the outlier sub-trajectory is determined, the notification of possible fraud can be suggested, even if the taxi is still in use. iBOAT is divided into two main steps: i) Preprocessing: The city area is broken down by a function noted $\lambda : \mathbb{R}^2 \rightarrow G$, that maps real locations (x, y) to matrix grid cells G (the grid cells size that gives the best accuracy is chosen in the experiment and set to "250 meter * 250 meter"). The set of the historical trajectories *T* are grouped according to the source-destination pairs and the time of the occurrences, and then mapped to T', where $t'_i = \lambda(t_i)$ for all $i \in [1, \dots, |T|]$. ii) Processing: When the new trajectory arrives, the points on the sub-trajectories that cause the outlierness are detected by using the*adaptive working window* strategy. For each new grid cell point g_i in the new trajectory t_{new}, the support of the sub-trajectory $< g_0, g_1, \dots, g_i >$ is computed by $\frac{|H(T_i, <g_0, g_1, \dots, g_i>)|}{|T_i|}$, such as $H(T, t) = \{t_k \in T | \forall i, j \in [1, \dots, |t|]^2, I_t(g_j) > I_t(g_i) \Rightarrow I_{t_k}(g_j) > I_{t_k}(g_i)\}$. $I_t(g_i)$ is the index of the grid cell point g_i in the trajectory t. If its value is less than the threshold θ, then grid cell point g_i is added to the set of anomalous grid cell points $\mathcal{O}$, otherwise, the set of historical trajectories used to process t_{new} is pruned for the next grid cell point g_{i+1} by $T_{i+1} = H(T_i, < g_0, g_1, \dots, g_i >)$. This process is repeated until all the points of t_{new} are processed. In the end, the outlier score of t_{new} is computed by $score(t_{new}, \mathcal{O}) = \sum_{i=1}^{|\mathcal{O}|} Support(T_i, < g_0, g_1, \dots, g_i >) + \sum_{i=1}^{|\mathcal{O}|-1} distance(g_i, g_{i+1})$, where $Support(T, t) = \frac{|H(T,t)|}{|T|}$.

Lee et al. [62] proposed TRAjectory Outlier Detection (TRAOD). It deals with the angular the sub-trajectory outlier detection problem, where the direction of anomalous sub-trajectories differ from those of neighboring sub-trajectories. The whole process of the algorithm exploits the *partition and detect* strategy. Each trajectory t_i, in the set of all trajectories T, is partitioned into different line segments noted t-partitions, where $P(t_i)$ is the set of all t-partitions of the trajectory t_i. In this step, a base unit approach is applied, where the partitions are defined as the smallest meaningful unit of a trajectory in a given application. After determining the partitions of each trajectory, the detection step is performed by computing the adjusting coefficient (notated as *adj*) of each t-partition L_k^i of the trajectory t_i as $adj(L_k^i) = \frac{(\sum_{t_r} density(L_k^r))/|T|}{density(L_k^i)}$ where $density(L_k^i) = |\bigcup_{t_r} \{L_k^r | distance(L_k^r, L_k^i) \leq radius\}|$, for a radius user's threshold. If this value is greater than 1, the t-partition is considered an outlier, otherwise, it is a normal t-partition. Note that if the density of the given t-partition is equal to 0, it is considered an outlier without computing the adjusting coefficient value. The particularity of this algorithm consists in the distance computation between two t-partitions. The projection and the angular dimensions are incorporated. The starting points and the ending points are s_1, s_2 e_1, and e_2 of the two t-partitions L_1 and L_2. Consider that x and y are the projection points of the starting point s_1 and e_1 onto L_2. The distance is given as $Distance(L_1, L_2) = \frac{a^2+b^2}{a+b} +$

$min\{a,b\} + c * Sin(\theta)$. a is $||x - s_1||$, b is $||y - e_1||$, c is $||y - e_1||$, and θ is the smallest positive intersecting angle between L_1 and L_2.

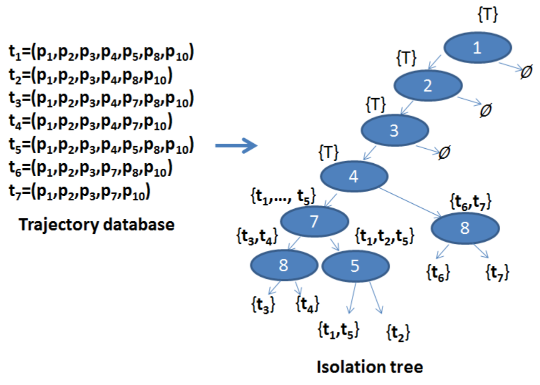
Yu et al. [63] developed Point-Neighbor based Trajectory Outlier (PN-Outlier). The aim is to find sub-trajectories outliers during a time window. It is based on point-neighbors principle, where the sub-trajectory neighbor set for each sub-trajectory is computed using the point neighbors set of each point in this sub-trajectory. First, the neighbor set N_j^i, of each point p_j^i, belongs to the trajectory t_i for a given threshold r, and is computed as $N_j^i = \{p_j^l | distance(p_j^i, p_j^l) \leq r\}$. Afterwards, the score of each sub-trajectory t_i is given by $Score(t_i) = |\{p_j^i || N_j^i| \geq k\}|$. k is the neighbor points count threshold. If this score exceeds the θ threshold, then t_i is an inlier, otherwise, it is an outlier.

Yu et al. [64] developed Trajectory-Neighbor based Trajectory Outlier (TN-Outlier). The aim is to find the sub-trajectories outliers during a time window W. It is based on trajectory-neighbors principle, where the sub-trajectory neighbors set for each sub-trajectory is determined. First, the neighbor function is defined between the j^{th} point of t_i and t_l for a given threshold r as $Neighbor(p_j^i, p_j^l) = 1$ if $distance(p_j^i, p_j^l) \leq r$, 0, otherwise. Afterwards, the score of each sub-trajectory t_i is given by $Score(t_i) = |\{t_j | \sum_{s=1}^{W} Neighbor(p_s^i, p_s^j) \geq k\}|$. k is the neighbor sub-trajectories count threshold. If this score exceeds θ threshold, then t_i is an inlier, otherwise, it is an outlier.

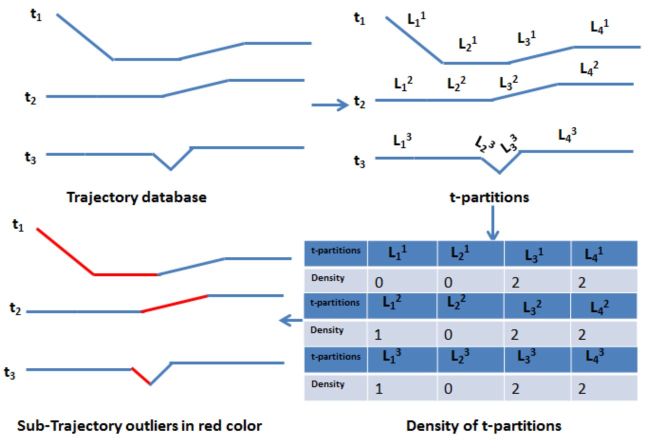
Wu et al. [65] presented the Driving Behavior-based Trajectory Outlier Detection (DB-TOD) approach. The set of historical trajectories is first matched to the road network of the city according to the source and destination points. The probabilistic learning model described by the maximum entropy inverse reinforcement [66] is then used to transform the mapped trajectories into historical action trajectories. Thus, each road segment is regarded as a state, the different road decisions such as turning left, turning right, on moving straight forward are regarded as actions, and the drivers are considered agents. Afterwards, the learning model is launched to estimate the cost of historical trajectories $cost(r_i) = \Theta^T f_{r_i}$, where $f_{r_i} = \sum_{a \in r_i} f_a$. The aim is to learn the Θ^T and f_a variables. For a new sub-trajectory t, the probability $P(t)$ is computed based on the set of action historical trajectories T as $P(t) = \frac{exp(-cost(t))}{\sum_{t' in T} exp(-cost(t'))}$. If the probability value is greater than a probability threshold then it is an outlier, otherwise, it is a normal sub-trajectory.

Mao et al. [67] proposed the Trajectory Fragment Outlier (TF-Outlier) method with the goal of determining the sub-trajectory outlier in a streaming way. In this approach, the set of trajectory fragments are derived. Each fragment tf_j^i of the trajectory $t_i = < p_1^i, p_2^i, \dots >$ is composed by a line segment of two consecutive points (p_j^i, p_{j+1}^i). The LOF algorithm is used to determine the fragment outliers, where the local difference density is used rather than the local reachability density as: $ldd(f_j^i) = \frac{|N(tf_j^i)|}{\sum_{tf_j^* \in N(tf_j^i)} distance(tf_j^i, tf_j^*)}$, where tf_j^* is the fragment neighbor of tf_j^i, and $distance(tf_j^i, tf_j^*)$ is the distance between the fragment tf_j^i and tf_j^*, computed as in [62]. The Local Anomaly Factor is then computed rather than the LOF value as: $LAF(tf_j^i) = \frac{\sum_{tf_j^* \in N(tf_j^i)} \frac{ldd(tf_j^i)}{ldd(tf_j^*)}}{|N(tf_j^i)|}$. If the LAF value is greater than the local outlier threshold, then the trajectory t_i is an outlier in the fragment tf_j^i. This process is repeated for all fragments in all trajectories.

Yu et al. [68] suggested Trajectory Outlier Detection based on Common Slices Sub-sequences (TODCSS) approach for discovering sub-trajectory outliers. For each trajectory t_i, a set of trajectory slices are generated where each j^{th} slice sl_j^i is obtained by connecting consecutive line segments having the same direction. A new definition of the sub-trajectory outlier is given based on the slice outlier. A trajectory slice is a slice outlier if the number of its neighbors is less than the given threshold. The neighbors refer to the other trajectory slices within a small distance from it. In addition, the distance between two slices sl_k^i of the trajectory t_i and sl_k^j of the trajectory t_j is determined by the number of common segments of both slices. Experiments performed on San Francisco urban traffic data revealed that TODCSS benefits from the slices definition in identifying the sub-trajectory outliers.



(a) iBAT illustration



t-partitions	L_1^1	L_2^1	L_3^1	L_4^1
Density	0	0	2	2
t-partitions	L_1^2	L_2^2	L_3^2	L_4^2
Density	1	0	2	2
t-partitions	L_1^3	L_2^3	L_3^3	L_4^3
Density	1	0	2	2

(b) TRAOD illustration

Figure 6. Trajectory outlier detection illustration.

Table 1. Discussion of Existing Urban Outlier Traffic Data Algorithms.

Task	Category	Merits	Limits
Flow outliers	Statistical	Low time consumption	Difficult to find the appropriate distribution of the given traffic flow data
	Similarity	Employ neighborhood computation methods to derive outliers	Ignore correlation between the single flows
	Pattern mining	Consider correlation between the flow outliers	Multiple scanning of the flow database
Trajectory outliers	Offline processing	Fast time consumption	Finds only whole trajectory outliers
	Online processing	Finds sub-trajectory outliers	High time consumption

C. ILLUSTRATIVE EXAMPLE

In this section, we demonstrate how the trajectory outlier detection algorithms work. In particular, two algorithms will be illustrated. The first one is iBAT [51], which performs the offline processing and detecting of the trajectory outliers. The second one is TROAD [62], which performs online processing and extracting sub-trajectory outliers. Starting by illustrating iBAT, consider the set of trajectories $T = \{t_1, t_2, \dots, t_7\}$ illustrated by Figure 6(a). Each trajectory is represented by the set traversed points from the set of all points $P = \{p_1, p_2, \dots, P_{10}\}$. The isolation tree is built, and the point p_4 is divided by the trajectory database into two subsets. The first subset $\{t_1, t_2, t_3, t_4, t_5\}$, added in the left child for the node labeled by 4, traverses point p_4 at the time window w_4, whereas the second subset $\{t_6, t_7\}$, does not traverse the point p_4 in the time window w_4, and will be added in the right child for the same node. The anomalous trajectories are ranked regarding the isolation level as $(t_6, t_7, t_3, t_4, t_1, t_5)$. Next, concerning the second algorithm, TROAD, three trajectories are assumed: $\{t_1, t_2, t_3\}$, as illustrated by Figure 6(b). TROAD first partitions these trajectories into t-partitions as: i) t_1: (L_1^1, L_2^1, L_3^1, L_4^1), ii) t_2: (L_1^2, L_2^2, L_3^2, L_4^2), and iii) t_3: L_1^3, L_2^3, L_3^3, L_4^3. The density of the t-partitions of all the trajectories is then computed. For instance, the density of the t-partition L_1^1 is equal to 0 because it is different from L_1^2 and L_1^3, while the density of the t-partition L_3^1 is equal to 2 because it is similar to L_3^2 and L_3^3. The t-partitions with a density of less than 1 are considered outliers. The anomalous sub-trajectories are then extracted as the following: i) For t_1: (L_1^1, L_2^1), ii) For t_2: (L_2^2), and iii) For t_3: (L_2^3).

V. DISCUSSIONS

Table 1 presents the merits and limitations of the existing urban outlier traffic data algorithms. As shown in the previous sections, we classify the urban outlier traffic algorithms into two groups according to the task employed.

Algorithms in the first group aim to find flow outliers, and these include three categories: i) Statistical approaches are fas, but very sensitive to the outliers, and it is not easy to find the corresponding distribution of the given traffic flow data. ii) Similarity approaches use neighborhood computation to find outliers. This helps to increase the accuracy of such approaches, however, they ignore the correlations between flow data and are only able to find single flow outliers. iii) Pattern mining approaches consider the correlation between the flow outliers. This helps to extract useful patterns and deals with two exciting applications: causal interaction and congested patterns. Nevertheless, these approaches are highly time consuming because they require multiple scans of the flow database.

The second task of the existing urban outlier traffic data algorithms is finding trajectory outliers. These approaches can be divided into two categories: i) Offline processing aims to find trajectory outliers after constructing the entire trajectory database. They are fast compared to those of the second category but they are only able to find trajectory outliers and not the part that is causing the outlierness. ii) Online processing deals with sub-trajectory outliers, i.e., they are able not only to find the trajectory outliers but also the sub-trajectories that cause the anomalies. These approaches extract outliers in online processing and require a large amount of time to update the output for each new sub-trajectory data.

From this literature review of existing urban outlier traffic algorithms, many directions for future research can be suggested:

1) Improving the run-time performance of existing pattern mining approaches by adapting the recent pattern mining approaches [69] in the mining process.
2) Some existing applications could be improved such hot spot and crime detection by considering the correlation between single flow outliers.
3) The existing algorithms find only single flow outliers in a specific time, and it is necessary to deal with other patterns such as constructing the distribution of flows in a given time measurement and finding the distribution of flows that deviate from the normal distribution of flows in a given period time. For example, finding all anomalous distribution of flows from 7:00 to 9:00 every Monday day in a given year.
4) Improving the runtime performance of the online algorithms to find sub-trajectory outliers by adapting computational intelligence approaches and high performance computing.

VI. CONCLUSION

Outlier detection algorithms have been largely used in urban traffic data for a long time. Solutions to urban outlier detection are divided into two main categories: flow outlier detection and trajectory outlier detection. Flow outlier detection groups solutions that detect flow outliers, and include the following approaches: 1) Statistical approaches which apply and combine the classical statistical model, and reach their limits because it is not straightforward to approximate the traffic flow values to the corresponding distribution. 2) Similarity approaches aim to explore neighborhood computation methods to determine dense regions in the traffic flow space. These approaches are efficient in terms of computational time, however, they ignore dependencies and relations that exist between traffic data. This issue is solved by 3) Pattern mining approaches investigate frequent pattern mining on traffic flow values data. Such approaches solve the correlation problem observed in the previous approaches, however, they are highly time consuming, especially for large traffic flow data. The second category groups solutions where the trajectory outliers are derived, and include the following processes: 1) Offline processing aims to extract trajectory outliers and is fast, but only considers whole trajectory databases. 2) Online processing aims to find the sub-trajectory outliers, and these approaches require a large amount of computational time for processing to update the output for each new sub-trajectory data. This paper ends in Section V with a summary of the most relevant lessons we concluded from this survey and the future direction of research trends in this arena. While the application of outlier detection reaches high efficiency in its traditional domains such as traffic networking, intrusion detection, and image processing, the use of outlier detection in urban traffic data is still in its infancy. We have seen only the tip of the iceberg, and further investigation is required in every direction, including computational intelligence, optimization, and high performance computing to attend to the sophisticated solutions that are suitable for smart city applications.

References

[1] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[2] Z. He, L. Zheng, L. Lu, and W. Guan, "Erasing lane changes from roads: A design of future road intersections," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 173–184, 2018.

[3] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM Sigmod Record*, vol. 29, no. 2, 2000, pp. 427–438.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *ACM Sigmod Record*, vol. 29, no. 2, 2000, pp. 93–104.

[5] M. Ernst and G. Haesbroeck, "Comparison of local outlier detection techniques in spatial multivariate data," *Data Mining and Knowledge Discovery*, vol. 31, no. 2, pp. 371–399, 2017.

[6] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," *Pattern Recognition*, vol. 74, pp. 406–421, 2018.

[7] N. Nesa, T. Ghosh, and I. Banerjee, "Outlier detection in sensed data using statistical learning models for iot," in *Wireless Communications and Networking Conference (WCNC), 2018 IEEE*. IEEE, 2018, pp. 1–6.

[8] H. Zhao, H. Liu, Z. Ding, and Y. Fu, "Consensus regularized multi-view outlier detection," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 236–248, 2018.

[9] H. Liu, X. Li, J. Li, and S. Zhang, "Efficient outlier detection for high-dimensional data," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 12, pp. 2451–2461, 2018.

[10] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 190–237, 2014.

[11] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, p. 29, 2015.

[12] Z. Feng and Y. Zhu, "A survey on trajectory data mining: techniques and applications," *IEEE Access*, vol. 4, pp. 2056–2067, 2016.

[13] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014.

[14] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, p. 38, 2014.

[15] S. Chen, W. Wang, and H. van Zuylen, "A comparison of outlier detection algorithms for its data," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1169–1178, 2010.

[16] K. Bhowmick and M. Narvekar, "Trajectory outlier detection for traffic events: A survey," in *Intelligent Computing and Information and Communication*, 2018, pp. 37–46.

[17] Y. Djenouri and A. Zimek, "Outlier detection in urban traffic data," in *International Conference on Web Intelligence, Mining and Semantics*, 2018, p. 3.

[18] F. Bunea, A. B. Tsybakov, M. H. Wegkamp *et al.*, "Aggregation for gaussian regression," *The Annals of Statistics*, vol. 35, no. 4, pp. 1674–1697, 2007.

[19] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM*, vol. 58, no. 3, p. 11, 2011.

[20] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[21] K. Kurihara, M. Welling, and Y. W. Teh, "Collapsed variational dirichlet process mixture models." in *International Joint Conference on Artificial Intelligence*, vol. 7, 2007, pp. 2796–2801.

[22] H. Y. Ngan, N. H. Yung, and A. G. Yeh, "Outlier detection in traffic data based on the dirichlet process mixture model," *IET Intelligent Transport Systems*, vol. 9, no. 7, pp. 773–781, 2015.

[23] P. Lam, L. Wang, H. Y. Ngan, N. H. Yung, and A. G. Yeh, "Outlier detection in large-scale traffic data by naïve bayes method and gaussian mixture model method," *Electronic Imaging*, vol. 2017, no. 9, pp. 73–78, 2017.

[24] R. Kingan and T. Westhuis, "Robust regression methods for traffic growth forecasting," *Journal of the Transportation Research Board*, no. 1957, pp. 51–55, 2006.

[25] R. E. Turochy and B. L. Smith, "Applying quality control to traffic condition monitoring," in *Intelligent Transportation Systems*, 2000, pp. 15–20.

[26] D. Gazis and L. Edie, "Traffic flow theory," *Proceedings of the IEEE*, vol. 56, no. 4, pp. 458–471, 1968.

[27] E. Park, S. Turner, and C. Spiegelman, "Empirical approaches to outlier detection in intelligent transportation systems data," *Journal of the Transportation Research Board*, no. 1840, pp. 21–30, 2003.

[28] T. T. Dang, H. Y. Ngan, and W. Liu, "Distance-based k-nearest neighbors outlier detection method in large-scale traffic data," in *IEEE International Conference on Digital Signal Processing*, 2015, pp. 507–510.

[29] J. Tang and H. Y. Ngan, "Traffic outlier detection by density-based bounded local outlier factors," *Information Technology in Industry*, vol. 4, no. 1, pp. 6–18, 2016.

[30] M. Munoz-Organero, R. Ruiz-Blaquez, and L. Sánchez-Fernández, "Automatic detection of traffic lights, street crossings and urban roundabouts combining outlier detection and deep learning classification techniques based on gps traces while driving," *Computers, Environment and Urban Systems*, vol. 68, pp. 1–8, 2018.

[31] Y. Shi, M. Deng, X. Yang, and J. Gong, "Detecting anomalies in spatio-temporal flow data by constructing dynamic neighbourhoods," *Computers, Environment and Urban Systems*, vol. 67, pp. 80–96, 2018.

[32] Y. Cheng, Y. Zhang, J. Hu, and L. Li, "Mining for similarities in urban traffic flow using wavelets," in *Intelligent Transportation Systems Conference*, 2007, pp. 119–124.

[33] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1, pp. 1–6, 1998.

[34] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm Sigmod Record*, vol. 22, no. 2, 1993, pp. 207–216.

[35] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM Sigmod Record*, vol. 29, no. 2, 2000, pp. 1–12.

[36] D. Martín, J. Alcalá-Fdez, A. Rosete, and F. Herrera, "NICGAR: A Niching Genetic Algorithm to mine a diverse set of interesting quantitative association rules," *Information Sciences*, vol. 355, pp. 208–228, 2016.

[37] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering spatio-temporal causal interactions in traffic data streams," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 1010–1018.

[38] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng, "On mining anomalous patterns in road traffic streams," in *International Conference on Advanced Data Mining and Applications*, 2011, pp. 237–251.

[39] M. Wu, X. Song, C. Jermaine, S. Ranka, and J. Gums, "A lrt framework for fast spatial anomaly detection," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 887–896.

[40] S. Chawla, Y. Zheng, and J. Hu, "Inferring the root cause in road traffic anomalies," in *IEEE International Conference on Data Mining*, 2012, pp. 141–150.

[41] D. Brauckhoff, K. Salamatian, and M. May, "Applying PCA for traffic anomaly detection: problems and solutions," in *IEEE International Conference on Computer Communications*, 2009, pp. 2866–2870.

[42] R. Inoue, A. Miyashita, and M. Sugita, "Mining spatio-temporal patterns of congested traffic in urban areas from traffic sensor data," in *IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 731–736.

[43] H. Nguyen, W. Liu, and F. Chen, "Discovering congestion propagation patterns in spatio-temporal traffic data," *IEEE Transactions on Big Data*, vol. 3, no. 2, pp. 169–180, 2017.

[44] J. Zhang, "Smarter outlier detection and deeper understanding of large-scale taxi trip records: a case study of NYC," in *ACM SIGKDD International Workshop on Urban Computing*, 2012, pp. 157–162.

[45] G. V. Batz, R. Geisberger, P. Sanders, and C. Vetter, "Minimum time-dependent travel times with contraction hierarchies," *Journal of Experimental Algorithmics*, vol. 18, pp. 1–4, 2013.

[46] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, and A. Tolba, "LoTAD: long-term traffic anomaly detection based on crowdsourced bus trajectory data," *World Wide Web*, vol. 21, no. 3, pp. 825–847, 2018.

[47] M. Mohibullah, M. Z. Hossain, and M. Hasan, "Comparison of euclidean distance function and manhattan distance function using k-mediods," *International Journal of Computer Science and Information Security*, vol. 13, no. 10, p. 61, 2015.

[48] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, 1967, pp. 281–297.

[49] Z. Jie, J. Wei, A. Liu, G. Liu, and Z. Lei, "Time-dependent popular routes based trajectory outlier detection," in *International Conference on Web Information Systems Engineering*, 2015, pp. 16–30.

[50] Z. Jie, J. Wei, L. An, L. Guanfeng, and Z. Lei, "Effective and efficient trajectory outlier detection based on time-dependent popular route," *World Wide Web*, vol. 20, no. 1, pp. 111–134, 2017.

[51] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "iBAT: detecting anomalous taxi trajectories from GPS traces," in *International Conference on Ubiquitous Computing*, 2011, pp. 99–108.

[52] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, p. 6, 2006.

[53] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[54] Z. Lv, J. Xu, P. Zhao, G. Liu, L. Zhao, and X. Zhou, "Outlier trajectory detection: A trajectory analytics based approach," in *International Conference on Database Systems for Advanced Applications*, 2017, pp. 231–246.

[55] H. Cardot, P. Cénac, and J.-M. Monnez, "A fast and recursive algorithm for clustering large datasets with k-medians," *Computational Statistics & Data Analysis*, vol. 56, no. 6, pp. 1434–1449, 2012.

[56] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, and H. Bunke, "Approximation of graph edit distance based on hausdorff matching," *Pattern Recognition*, vol. 48, no. 2, pp. 331–343, 2015.

[57] X. Zhou, Y. Ding, F. Peng, Q. Luo, and L. M. Ni, "Detecting unmetered taxi rides from trajectory data," in *IEEE International Conference on Big Data*, 2017, pp. 530–535.

[58] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[59] D. Kumar, J. C. Bezdek, S. Rajasegarar, C. Leckie, and M. Palaniswami, "A visual-numeric approach to clustering and anomaly detection for trajectory data," *The Visual Computer*, vol. 33, no. 3, pp. 265–281, 2017.

[60] T. C. Havens and J. C. Bezdek, "An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 813–822, 2012.

[61] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "iBOAT: Isolation-based online anomalous trajectory detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 806–818, 2013.

[62] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *IEEE International Conference on Data Engineering*, 2008, pp. 140–149.

[63] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Detecting moving object outliers in massive-scale trajectory streams," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 422–431.

[64] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Outlier detection over massive-scale trajectory streams," *ACM Transactions on Database Systems*, vol. 42, no. 2, p. 10, 2017.

[65] H. Wu, W. Sun, and B. Zheng, "A fast trajectory outlier detection approach via driving behavior modeling," in *ACM Conference on Information and Knowledge Management*, 2017, pp. 837–846.

[66] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Conference of the Association for the Advancement of Artificial Intelligence*, vol. 8, 2008, pp. 1433–1438.

[67] J. Mao, T. Wang, C. Jin, and A. Zhou, "Feature grouping-based outlier detection upon streaming trajectories," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2696–2709, 2017.

[68] Q. Yu, Y. Luo, C. Chen, and X. Wang, "Trajectory outlier detection approach based on common slices sub-sequence," *Applied Intelligence*, vol. 48, no. 9, pp. 2661–2680, 2018.

[69] K.-W. Chon, S.-H. Hwang, and M.-S. Kim, "GMiner: A fast GPU-based frequent itemset mining method for large-scale data," *Information Sciences*, vol. 439, pp. 19–38, 2018.