Trajectory Outlier Detection: New Problems and Solutions for Smart Cities

YOUCEF DJENOURI, SINTEF Digital
DJAMEL DJENOURI, University of the West of England
JERRY CHUN-WEI LIN, HVL

This article introduces two new problems related to trajectory outlier detection: (1) *group trajectory outlier (GTO) detection* and (2) *deviation point detection* for both individual and group of trajectory outliers. Five algorithms are proposed for the first problem by adapting *DBSCAN*, *k nearest neighbors (kNN)*, and *feature selection (FS)*. *DBSCAN-GTO* first applies *DBSCAN* to derive the *micro clusters*, which are considered as potential candidates. A pruning strategy based on density computation measure is then suggested to find the group of trajectory outliers. *kNN-GTO* recursively derives the trajectory candidates from the individual trajectory outliers and prunes them based on their density. The overall process is repeated for all individual trajectory outliers. FS-GTO considers the set of individual trajectory outliers as the set of all features, while the FS process is used to retrieve the group of trajectory outliers. The proposed algorithms are improved by incorporating ensemble learning and high-performance computing during the detection process. Moreover, we propose a general two-phase-based algorithm for detecting the deviation points, as well as a version for graphic processing units implementation using sliding windows. Experiments on a real trajectory dataset have been carried out to demonstrate the performance of the proposed approaches. The results show that they can efficiently identify useful patterns represented by group of trajectory outliers, deviation points, and that they outperform the baseline group detection algorithms.

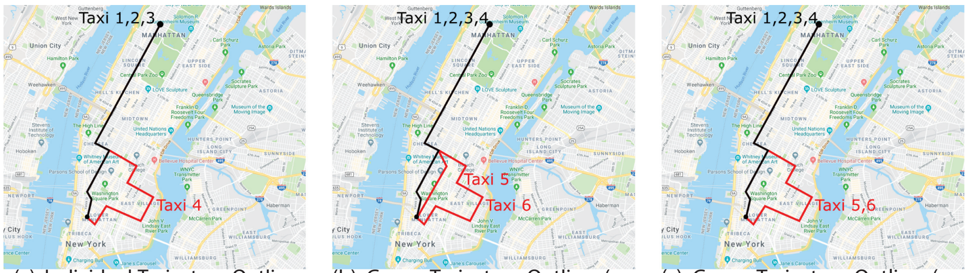
CCS Concepts: • **Information systems** → *Data mining*; • **Computing methodologies** → *Anomaly detection*;

Additional Key Words and Phrases: Trajectory analysis, outlier detection, data mining, road traffic management, smart city application

1 INTRODUCTION

Today's road traffic in smart cities is being monitored by ubiquitous sensing technologies such as cameras, embedded Global Positioning System (GPS) receivers, road sensors, and in-vehicle sen-

Authors' addresses: Y. Djenouri, Department of Mathematics and Cybernetics, SINTEF Digital, Gaustadalléen 23 C, 0373 Oslo, Norway; email: youcef.djenouri@sintef.no; D. Djenouri, CSRC, Department of Computer Science and Creative Technologies, University of the West of England, UWE Bristol - Frenchay Campus, Coldharbour Ln, Bristol BS16 1QY, United Kingdom; email: djamel.djenouri@uwe.ac.uk; J. C.-W. Lin, Department of Computing, Mathematics, and Physics, HVL, Inndalsveien 28, 5063 Bergen, Norway; email: jerrylin@ieee.org.



(a) Individual Trajectory Outlier (b) Group Trajectory Outliers (example1) (c) Group Trajectory Outliers (example2)

Fig. 1. Motivated example: trajectory and outliers.

sors [19]. These technologies enable to generate countless number of sequence points represented by trajectory databases, which are stored and analyzed using high-performance computing (HPC). These trajectory databases simulate various behaviors of different objects in several real-world applications such as intelligent transportation [2, 8, 34, 35, 47, 87], mobile traffic network [28, 36, 63], smart buildings and large indoor spaces [1, 18, 39], and climate change analysis [5, 16]. In the intelligent transportation domain, data analysts face myriad of trajectories derived by the mobility of people, cars, buses, and taxis. One of the problems dealt with by data analysis of trajectory databases is trajectory outlier detection, which represents the topic of this article. From the data mining perspective, outlier detection consists in separating unusual observations, objects, and/or points from the normal observations [13, 31, 62]. In the context of trajectory analysis, this translates into discovering trajectories or sub-trajectories that do not conform with the rest of trajectories in a database [52, 54, 88]. Current solutions only consider single view of outliers in a whole trajectory or a sub-trajectory. However, it is not only individual outliers that could be identified in real-world scenarios. Different types of patterns and useful features can be derived such as group of trajectory outliers, deviation point for both individual and group of trajectory outliers. We define the group of trajectory outliers by the set of individual outliers that are closer to each other, or proving some common features, and the deviation points by the set of points that causes anomalies. In this work, we explore new solutions to retrieve these patterns.

1.1 Motivating Example

Consider the three examples of taxi trajectories as illustrated in Figure 1. Each taxi trajectory starts from the source point and ends up at the destination point. Traditional trajectory outlier detection algorithms [26, 41, 50, 84] may detect the outlier illustrated in Figure 1(a) (represented with red color). Taxi 4 follows the normal trip from the source to the destination up to a given point, where it highly deviates from the trajectory followed by taxis (1, 2, and 3). Traditional trajectory outlier detection algorithms are not able to identify the *groups* of trajectory outliers presented in Figure 1(b) and 1(c). This is because they calculate the score of each individual trajectory and then rely on the fixed threshold (set by the user) to decide whether the trajectory is an outlier or not. They do not examine the different correlations between the individual trajectory outliers and do not consider the pattern represented by the group of trajectories as input during the detection process. In Figure 1(b), taxis (5 and 6) deviate from the normal trip at the same deviation point but follow different trajectories. However, in Figure 1(c), the two taxis deviate from the normal trip at the same deviation point and follow the same trajectory. Detecting these different types of outliers could help the city planners and local authorities to extract patterns and to discover relevant knowledge. For instance, detecting individual outliers and deviation points (the case of Figure 1(a), which allows to determine individual taxi fraud), detecting frequently individual taxi fraud at the same individual deviation point (IDP) with different time periods. This will help supporting the city planners on taking appropriate measures and decisions such as putting surveillance cameras at this point or reinforcing security there. Detecting group of trajectory outliers (Figure 1(b) and 1(c)) allows the city planners to make fair decisions regarding the taxi outliers. Taxis which deviate from the same deviation point but follow different trajectories have strong probabilities that their aim is avoiding unpleasant circumstances on the main trajectory such as traffic jam, rather than taxi fraud. However, group of taxis outliers at the same or different deviation point with the same trajectory taking place in different time periods might be partners in the taxi fraud. Notice both temporal and spatial information are required to retrieve the taxi frauds. The major problem in the existing taxi fraud detection algorithms is the the detection of group of trajectory outliers, and the deviation points identification on the spatio-temporal trajectory data.

1.2 Contribution

This article introduces two new problems related to trajectory outlier detection. The first problem is Group Trajectory Outlier (GTO) *detection*, while the second is *Deviation Point Detection* (DPD) for both individual and group of trajectory outliers. The main contributions of this work can be summarized as follows.

(1) **GTO:** We introduce and formulate the *GTO* problem, for which we develop two algorithms. The first is based on *DBSCAN* [24] (say *DBSCAN-GTO*) and the second is based on *k*NN [61] (*k*NN-GTO). DBSCAN-GTO first applies *DBSCAN* to derive the *micro clusters*, which are considered as potential candidates. A pruning strategy based on density computation measure is then suggested to find the group of trajectory outliers. *k*NN-GTO starts recursively by deriving the trajectory candidates from the individual trajectory outliers, and then it prunes these candidates using the density computation. The overall process is repeated for all individual trajectory outliers. A new algorithm called FS-GTO is also proposed, in which the set of individual trajectory outliers are considered as the set of all features, and the feature selection (FS) process is adopted to identify the group of trajectory outliers. Finally, two improvements of the above-mentioned solutions are proposed by incorporating ensemble learning and HPC.

(2) **DPD:** We introduce and formulate this problem and then develop a general two-phase-based algorithm. The first phase aims at identifying individual trajectory outliers based on the distance of each point in each trajectory, while the second phase explores the individual trajectory outliers to derive the group of trajectory outliers by using the FS process. Moreover, a Graphic Processing Units (GPU)-based version of the two-phase based algorithm is incorporated with a sliding windows to boost the performance of the outlier detection on large-scale trajectory point-spaces.

We investigate the performance of the proposed algorithms on different real trajectory databases. The results confirm the scalability of the new approaches, and that they outperform the baseline algorithms for group detection. They also show that the two-phase-based algorithm is able to derive deviation points for both individual and group of trajectory outliers for 92% of cases. Regarding the big trajectory databases, the results show that the proposed GPU-based solutions outperform the baseline GPU solutions for detecting trajectory outliers.

1.3 Outline

The remainder of the article is organized as follows. Section 2 reviews the main existing trajectory and group outlier detection algorithms. Section 3 formulates the two problems (*GTO* and *DPD*).

Section 4 describes *DBSCAN-GTO*, *k*NN-GTO, FS-GTO, and the two improvements using ensemble learning and HPC. Section 6 presents the performance evaluation. Section 7 discusses the lessons learned and draws future directions of this work. Finally, Section 8 concludes the article.

2 RELATED WORK

2.1 Trajectory Outlier Detection

We split out the trajectory outlier detection algorithms into two categories: (1) *offline methods* that can only detect the trajectory outliers; and (2) *online methods* in which the sub-trajectory that causes the outlierness can be identified.

2.1.1 Offline Methods. Zhang et al. [81] proposed a graph-based method for detecting multi levels of taxi trip outliers in a large scale urban traffic network. The method implements a contraction hierarchy based on the shortest path computation algorithm and a spatial join algorithm to snap, pickup, and drop-off locations. Kong et al. [38] proposed an adopted local outlier factor by considering the anomaly index score as a local reachability density. Zhu et al. [89, 90] proposed an approach that determines time-dependent outliers by using the common routes for each time interval. The trajectories' database is divided into groups with the same source and destination points in the given time interval. The score of the representative trajectory of each group is determined by comparing its similarity with the most common roads using the edit distance. If a group's score is below a predefined threshold, then all the trajectories of the group are considered as outliers. Zhang et al. [80] proposed the isolation-based anomalous trajectory algorithm, in which the "few and different" properties of anomalous trajectories are explored instead of using a distance or density measure. By exploring different locations or the same locations with different orders, anomalous trajectories are *few* in number and *different* from the majority of trajectories. The algorithm attempts to find a separate way for anomalous trajectories from the rest of "many and similar" trajectories by applying the adapted isolation Forest (iForest) [49].

Zhongjian et al. [51] proposed an approach in which the set of routes is grouped using the medoids algorithm [9]. The choice of medoids instead of $k - means$ is justified by the difficulty of computing the mean trajectories. The set of the centers of the clusters are considered as representative routes, while the scores of the new trajectories' scores are computed based on the representative routes using *edit* distance between routes and trajectories. Trajectories with scores exceeding a similarity threshold are considered as outliers. In the work by Zhou et al. [86], the trajectory database was matched to identify whether each point in the trajectories database is a metered or unmetered point. The process starts by finding the trajectory outliers using a stochastic gradient model [25]. From the trajectory outliers, the fraud trajectories are identified by matching each point in the trajectory outlier with the taximeter database.

2.1.2 Online Methods. Chen et al. [12] automatically detected fraud implications of rapacious taxi drivers who take unnecessary detours during trips. This is by using an approach based on *adaptive working window*. Lee et al. [41] dealt with the angular sub-trajectory outlier detection problem, i.e., when the direction of anomalous sub-trajectories differs from those of neighboring sub-trajectories. Each trajectory in the set of all trajectories was partitioned into different line segments noted t-partitions. The particularity of this algorithm consists in the computation of distances between two t-partitions, where the projection and the angular dimensions are incorporated. Yu et al. [76, 79] found sub-trajectories outliers during a time window using two strategies. The first one is based on the point-neighbors principle in which the sub-trajectory neighbor set for each sub-trajectory is calculated using the point neighbors set of every point in this sub-trajectory. The second is based on trajectory-neighbors principle that determines the sub-trajectory neighbors set for each sub-trajectory. In Wu et al. [73], the set of historical trajectories was 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 [91] was used to transform the mapped trajectories into historical action trajectories. The probability of every sub-trajectory is computed based on the set of action historical trajectories. If the probability value exceeds a defined threshold then the sub-trajectory is considered as outlier. Mao et al. [53] used the local outlier factor algorithm to determine the fragment outliers, as well as the local difference density (instead of the local reachability density). In this method, the set of trajectory fragments is derived in fragments, and each fragment of a trajectory is composed by a line segment of two consecutive points. This process repeats for all fragments in all trajectories. Yu et al. [77] provided a new definition of the sub-trajectory outlier based on the concept of "slice outlier." Slices are obtained by connecting consecutive line segments having the same direction, and a trajectory slice is considered slice outlier if the number of its neighbors is less than a given threshold. The neighbors refer to the other trajectory slices within a small distance from it.

2.2 Group Outlier Detection

The generic problem of group outlier detection is not new, but only a few solutions have reported. Some solutions employed statistical models to derive the group of outliers [46, 68, 75, 78]. Chalapathy et al. [11] considered the use of deep generative model to find out the group outliers. The approach was applied on various image processing applications. The outlierness for each group in the input data was estimated by group reference function using the standard backpropagation algorithm. Liang et al. [75] considered the use of flexible genre model to find specific group outliers. Gibbs sampling [27] was used for inference and the Monte Carlo approach for learning [10]. Das et al. [15] considered the different correlation between the data outliers to detect pattern anomalous by investigating Bayesian network anomaly detection [56, 14]. The correlation score between the individual outliers was determined by the probability of possible outlier values in the training data. Tang et al. [67] defined contextual outlier detection as small group of points that share similarity, on some attributes, with a significantly larger reference group of data but deviates dramatically on some other attributes. The authors proposed to maintain only the closure context outliers (to avoid enumerating all contextual outliers). Furthermore, their approach retrieves only contextual outliers with a statistical significance test greater than a given threshold. Li et al. [45] proposed assigning feature weights on each group outlier and computing chain rule entropy to determine correlation between different feature groups. Parallel computing was used in [82] to deal with contextual outlier detection in high and sparse dimensional spaces. Xiong et al. [74] proposed a solution that detects two kinds of group anomalies: (1) a group of individual anomalous points; and (2) a set of normal points but with abnormal distribution as a group. An application of this algorithm in social media analysis has been investigated in [78]. Other approaches gather group individual outliers into similar clusters [15, 65, 67]. Each cluster is then considered as a group of outliers. Soleimani et al. [65] proposed supervised learning approach that groups anomalous patterns when memberships are previously unknown. This approach was applied on topic documents modeling, and it is able to discover irregular topic mixtures from a collection of documents. Sun et al. [66] proposed abnormal group-based joint medical fraud approach. The abnormal group problem was converted into the maximal clique enumeration problem [55] by considering the set of patients as the set of vertices, while each edge indicates that the two connected patients are similar. Since the maximal clique enumeration is NP-hard problem, different partitioning methods [23] have been investigated to reduce the graph size, and every maximal clique was considered as abnormal group of patients.

2.3 Discussion

We conclude from this short literature review that there is no work that explores group outlier detection for trajectory data. Existing solutions for trajectory only find individual outliers of whole trajectory using offline processing, and the sub-trajectory outliers using online processing. Also notice that the current group outlier detection algorithms find the group outliers from the set of candidate groups, and not from the individual outliers. Moreover, most of the existing group outlier detection algorithms are based on some well-known distributions. However, it is hard to fit the data to such distributions in real scenarios. This article deals with the challenging problem of GTO detection. It proposes several strategies based on clustering, neighborhood computation, and FS to detect GTOs from individual trajectory outliers. It also propose algorithms to detect the deviation point of the trajectories, both for individual and group outliers.

3 PROBLEM STATEMENT

Throughout the article, a trajectory is considered as a sequence of location points in space and time. Some preliminary definitions are given in the following before introducing the GTO and DPD problems.

Definition 1 (Trajectory Database). A trajectory database is defined as a set of raw trajectories $T = \{T_1, T_2, \ldots, T_m\}$. Each raw trajectory, T_i, is a sequence of time-ordered points $(p_1, \ldots, p_n)$ that might be obtained by localization techniques (e.g., GPS).

Definition 2 (Time Interval). The time period is split into several time intervals, and $p(interval)$ denotes the time interval to which the point p belongs.

Similar location points are aggregated into regions [20]. Let us denote by R a location *region* in space.

Definition 3 (Mapped Trajectory Database). A mapped trajectory database is a set $\Lambda = \{\Lambda_1, \ldots, \Lambda_m\}$, in which each mapped trajectory Λ_i is a sequence of spatio-temporal regions, $(R_1, \ldots, R_n)$, obtained by mapping every point in T_i to the closest region R_i. The time interval of R_i represents the time interval of the point p_i, i.e., $R_i(interval) = p_i(interval)$.

Definition 4 (Trajectory Dissimilarity). The dissimilarity between any two trajectories, $d(\Lambda_i, \Lambda_j)$, is defined as the distance between them, which is the difference between the number of all regions to the number of shared regions belonging to the same time interval between the two trajectories. That is,

$$d(\Lambda_i, \Lambda_j) = n - |\{(R_l, R'_l) : R_l = R'_l \vee \forall l \in [1 \ldots n]\}|. \tag{1}$$

The trajectory candidates are represented by the set of potential trajectories that belong to a group of trajectory outliers. These trajectory candidates are retrieved from the individual trajectory outliers and are formally defined as.

Definition 5 (Trajectory Candidates). The set of the top l individual trajectory outliers found by a given trajectory outlier detection algorithm, denoted $\mathcal{G}^+ = \{\Lambda_1^+, \Lambda_2^+ \ldots \Lambda_l^+\}$, is,

$$\mathcal{G}^+ = \{\Lambda_i^+ : \forall j \in \Lambda \setminus \mathcal{G}^+, Score_{\mathcal{A}}(\Lambda_i) \geq Score_{\mathcal{A}}(\Lambda_j)\}. \tag{2}$$

Note that $Score_{\mathcal{A}}(\Lambda_i)$ is the outlier score of the trajectory Λ_i using the outlier detection algorithm $\mathcal{A}$. The score depends on the algorithm used for identifying the individual trajectory outliers and finding out the trajectory candidates. In the experimentation part, we use the local outlier factor [6] to retrieve the trajectory candidates.

The area of the point and the density of a group of trajectories are important concepts in our analysis. Intuitively, the area of a point is defined by the sum of distances between this point and the points of all trajectories with the same time timestamp. The density of the group is defined as the number of shared regions between all the trajectories of the group.

Definition 6 (Point Area). We define the area of the point, p_i, as:

$$\mathcal{PA}(p_i) = \frac{\sum_{l=1}^{m} distance(p_i, p_l)}{m}. \tag{3}$$

Note that the distance between two points is calculated using the Euclidean distance by considering the spatio-temporal information (latitude, longitude, and interval) in the Euclidean space.

Definition 7 (Group Density). The density of the candidate GTOs, $\mathcal{G}$, is,

$$Density(\mathcal{G}) = |\{R : \forall \Lambda_i \in \mathcal{G}, R \in \Lambda_i\}|. \tag{4}$$

Definition 8 (GTO). A set of trajectories, $\mathcal{G}$, is called a GTO iff,

$$\begin{cases} \mathcal{G} \subseteq \mathcal{G}^+ \\ Density(\mathcal{G}) \geq \gamma \end{cases} \tag{5}$$

Note that γ is the density threshold from the interval $[1 \ldots n]$.

Definition 9 (GTO Problem). The GTO problem aims to discover from the set of all individual trajectory outliers the set of all groups of trajectory outliers, denoted by $\mathcal{G}^*$.

Definition 10 (Individual Deviation Point). Consider Λ_i an individual trajectory outlier. Λ_i could be divided into two sub-trajectories, Λ_i^{1x} and Λ_i^{xn}. x is called IDP iff Λ_i^{xn} is sub-trajectory outlier, and every point, $y \in [1..x]$, is a normal sub-trajectory.

Definition 11 (Group Deviation Point (GDP)). Consider a group of trajectory outliers: $\mathcal{G} = \{\Lambda_1^{x_1, y_1}(\mathcal{G}), \Lambda_2^{x_2, y_2}(\mathcal{G}), \ldots, \Lambda_|\mathcal{G}|^{x_{|\mathcal{G}|}, y_{|\mathcal{G}|}}(\mathcal{G})\}$. x is called GDP iff the point x is a starting point for at least one trajectory in $\mathcal{G}$, and x is highly ordering all remaining starting points in $\mathcal{G}$.

Definition 12 (DPD Problem). It aims to discover from the set of all individual and group trajectory outliers, all the IDP and GDP points.

The trivial solution for GTO and DPD problems is to consider all possible combinations between the individual trajectory outliers, and then evaluates every subset separately using Definition 7. The group of trajectory outliers are first derived, and all points for all individual and group of trajectory outliers are scanned and checked if they are IDP and GDP points. This method evaluates the candidate sets and save the potential groups of trajectory outliers. It then processes all points for individual and group of trajectory outliers. This requires high computational and memory resources. The theoretical complexity of such an approach would be $O(2^{|\mathcal{G}^+|})$ for identifying GTOs, further to $O(n \times |\mathcal{G}^+|)$ for determining IDP and GDP points. To address these issues, we propose in the next sections different solutions to improve the detection of GTOs and deviation points.

4 PROPOSED FRAMEWORK AND ALGORITHMS FOR GTO PROBLEM

This section first presents the overall framework proposed for the GTO problem. The framework detects group of trajectory outliers with different methods based on machine learning, computational intelligence, and HPC. Several machine learning techniques are used including clustering, FS, neighborhood computation, and ensemble learning. Computational intelligence and HPC boost the runtime performance and enable dealing with big trajectory databases in a reasonable time. As illustrated in Figure 2, the framework includes the following steps:

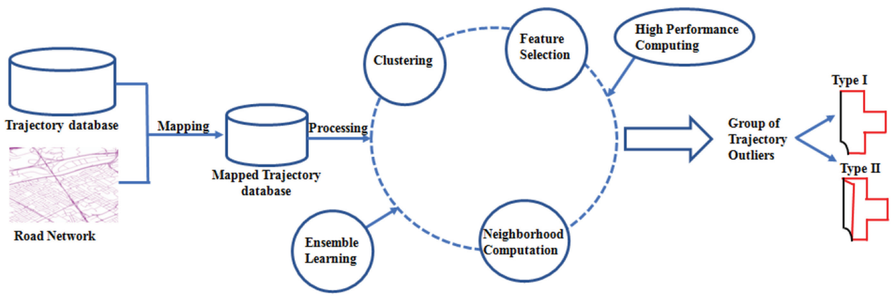


Fig. 2. General framework.

(1) Mapping: Trajectories usually provide noisy GPS data points, with possible errors at the order of several meters. This will negatively influence the final output of the trajectory outlier detection systems. In practice, a mapping step is used to project GPS data points of each trajectory onto the road network. Several approaches have been developed in this area [7, 29, 48]. Since we are interested in dealing with sparse trajectory databases, we are inspired by a probabilistic model represented by a Hidden Markov Model [29, 60].

The mapped trajectory database is consequently created in which every observed trajectory is assigned to the associated road segment.

(2) Processing: After constructing the mapped trajectory database, a processing step is performed to find out the group of trajectory outliers. We propose two approaches in this context: (1) first, determine the individual trajectory outliers and then find out the group of trajectory outliers; and (2) derive directly the group of trajectory outliers from the mapped trajectory database. Furthermore, we suggest more sophisticated approaches by incorporating ensemble learning, computational intelligence, and HPC. In the remaining of this section, the use all these concepts in the proposed framework is explained.

4.1 DBSCAN-GTO

This section presents the adaptation of the DBSCAN algorithm [24] for the GTO problem. The outputs of the DBSCAN algorithm are: (i) the set of clusters with different density (high density and low density); and (ii) the set of individual trajectory noises. The latter cannot form the group of trajectory outliers since they are considered as noises and are far from one other. Solutions to trajectory clustering [42, 43] are able to derive clusters with different densities. However, they do not explore the *micro clusters* property for anomaly detection. This section presents a new approach for identifying group of trajectory outliers, called *DBSCAN-GTO*. It uses *DBSCAN* to search for clusters by checking the neighborhood of each trajectory. The neighborhoods of a trajectory Λ_i is defined as a subset of trajectories close to $Lambda_i$. Two trajectories are close if their distance is less than a given threshold ϵ, while a trajectory Λ_i is called core trajectory if the size of its neighborhoods exceeds a minimum number of trajectories *MinPts*. The core trajectories are determined, then the density-reachable trajectories are collected directly from the core trajectories. This may involve merging a few density-reachable clusters. The process terminates when no new trajectories can be added to any cluster. Algorithm 1 presents the pseudo-code of *DBSCAN-GTO*. Initially, the set of trajectories are grouped using *DBSCAN*. This generates several clusters with different sizes. Each micro cluster is considered as group of candidates. A micro cluster is the cluster of trajectories that contains no more than μ trajectories, with μ is a user-defined threshold. The density of each group is determined using Def. 7. If the density exceeds a threshold γ, then the group is considered as outlier.

ALGORITHM 1: DBSCAN-GTO Algorithm

```
1: Input: $\Lambda = \{\Lambda_1, \Lambda_2 \dots \Lambda_n\}$: The set of all trajectories.
   $\epsilon$, $MinPts$: DBSCAN parameters.
   $\mu$: User threshold for micro clusters.
   $\gamma$: density threshold.
2: Output: $\mathcal{G}^*$: sets of all group trajectory outliers.
3: C $\leftarrow$ DBSCAN($\Lambda$, $\epsilon$, $MinPts$)
4: $\mathcal{G}^* \leftarrow \emptyset$
5: for each $C_i \in C$ do
6:     if $|C_i| \leq \mu \vee$ Density$(C_i) \geq \gamma$ then
7:         $\mathcal{G}^* \leftarrow \mathcal{G}^* \cup C_i$
8:     end if
9: end for
10: return $\mathcal{G}^*$
```

4.2 *k*NN-GTO

This section presents the adaptation of the *k*NN algorithm [61] for identifying the group of trajectory outliers. The *k*NN of a trajectory Λ_i is defined as the *k* closest trajectories to λ_i. The following proposition holds:

Proposition 1. *Let us consider two trajectories* Λ', Λ'', *and* $\mathcal{G}^*(s)$ *is the group of trajectory outliers at the iteration* s *such that* $\Lambda' \in \bigcup_{\Lambda_i^* \in \mathcal{G}^*(s)} kNN(\Lambda_i^*)$ & $\Lambda'' \notin \bigcup_{\Lambda_i^* \in \mathcal{G}^*(s)} kNN(\Lambda_i^*)$.

We have this implication: $\Lambda' \notin \mathcal{G}^*(s+1) \Rightarrow \Lambda'' \notin \mathcal{G}^*(s+1)$.

Proof. $\Lambda' \in \bigcup_{\Lambda_i^* \in \mathcal{G}^*(s)} k\text{NN}(\Lambda_i^*) \quad \vee \quad \Lambda'' \notin \bigcup_{\Lambda_i^* \in \mathcal{G}^*(s)} k\text{NN}(\Lambda_i^*) \quad \Rightarrow \quad Density(\mathcal{G}^*(s) \cup \{\lambda''\}) \leq Density(\mathcal{G}^*(s) \cup \{\lambda'\})$... (1)

$\Lambda' \notin \mathcal{G}^*(s+1) \Rightarrow Density(\mathcal{G}^*(s) \cup \{\lambda'\}) \leq \gamma$... (2)

From (1) and (2) it yields $Density(\mathcal{G}^*(s) \cup \{\lambda''\}) \leq \gamma \Rightarrow \Lambda'' \notin \mathcal{G}^*(s+1)$. □

From the above proposition, one may argue that if a trajectory belongs to the *k nearest neighbors* (kNN) of at least one trajectory in the current group of trajectory outliers, and it if is not in the group of trajectory outliers of the next iteration, then any trajectory that belongs to the *kNNs* of at least one trajectory in the current group of trajectory outliers could not be in the group of trajectory outliers of the next iteration. Consequently, it seems to be judicious to prune the search into *kNNs* of the individual trajectory outliers.

In the following, we present an adapted *k*NN algorithm for the GTO ptoblem. The algorithm considers as input the set of the top *p* individual trajectory outliers, $\mathcal{G}^+ = \{\Lambda_1^+, \Lambda_2^+ \dots \Lambda_p^+\}$, that are ranked according to the *k*NN value, i.e., $\forall i \geq j$, $k\text{NN}(\Lambda_i^+) \geq k\text{NN}(\Lambda_j^+)$. The process aims to enumerate the sets of GTOs, $\mathcal{G}^*$, by exploring a search tree of $\mathcal{G}^+$. It first adds the most outlier trajectory, Λ_1^+, to the group denoted $\mathcal{G}_1^*$. It then generates all potential candidates from Λ_1^+. A trajectory t is a potential candidate from Λ_1^+ iff $t \in \mathcal{G}^+ \vee t \in k\text{NN}(\Lambda_1^+)$. The density of $\mathcal{G}_1^*$ is updated by adding the potential candidates (one by one) to $\mathcal{G}_1^*$. Only the potential candidates respecting the density threshold are saved, and all the remaining candidates are removed. Once the potential candidate is added to $\mathcal{G}_1^*$, it is removed from $\mathcal{G}^+$. If $\mathcal{G}_1^*$ contains less than two elements, it is removed from $\mathcal{G}^*$. The same process is recursively applied for all potential candidates added to $\mathcal{G}_1^*$, and the overall process is repeated for all trajectory outliers in $\mathcal{G}^+$. Algorithm 2 presents the pseudo-code of *k*NN-GTO.

ALGORITHM 2: *k*NN-GTO Algorithm

```
1: Input: $\Lambda = \{\Lambda_1, \Lambda_2 \ldots \Lambda_n\}$: The set of all trajectories.
   $\mathcal{A}$: trajectory outlier detection algorithm.
   $\gamma$: density threshold.
2: Output: $\mathcal{G}^*$: sets of all group trajectory outliers.
3: $\mathcal{G}^+_{\mathcal{A}} \leftarrow \mathcal{A}(\Lambda)$
4: for each trajectory $\Lambda^+_i \in \mathcal{G}^+_{\mathcal{A}}$ do
5:     $node \leftarrow \Lambda^+_i$
6:     for each trajectory $t \in (\text{kNN}(node) \cap \mathcal{G}^+_{\mathcal{A}})$ do
7:         if Density$(\mathcal{G}^*_i \cup \{t\}) \geq \gamma$ then
8:             $\mathcal{G}^*_i \leftarrow \mathcal{G}^*_i \cup \{t\}$
9:             $\mathcal{G}^+_{\mathcal{A}} \leftarrow \mathcal{G}^+_{\mathcal{A}} \setminus \{t\}$
10:            {repeat lines from 5 to 8 for a trajectory $t$}
11:        end if
12:    end for
13:    if $|\mathcal{G}^*_i| = 1$ then
14:        $\mathcal{G}^* \leftarrow \mathcal{G}^* \setminus \{\mathcal{G}^*_i\}$
15:    end if
16: end for
17: return $\mathcal{G}^*$
```

4.3 FS-GTO

This section presents the use of FS approaches to identify the group of trajectory outliers, starting by transforming the GTO problem to the FS problem. Consider a GTO problem represented by $< \mathcal{G}^+_{\mathcal{A}}, \mathcal{G}^* >$. This could be fitted to the FS problem represented by the set of all features, F, and the subset of selected features, F^*, such as $F = \mathcal{G}^+_{\mathcal{A}}$, and $F^* = \mathcal{G}^*$. In this approach, a FS technique is considered. Every individual trajectory outlier is considered as one feature, and the aim is to select the most relevant features from the whole individual trajectory outliers. The relevant set of features is then considered as GTOs. The evaluation of the selected features (trajectories) F^* is determined, whose aim is to maximize $Quality(F^*) - \frac{|F^*|}{|F|}$. Note that $Quality(F^*)$ is calculated using Def. 7. According to the recent work by Li et al. [44], FS algorithms have been categorized into three categories: (1) similarity based, (2) information theoretical based, and (3) statistical-based methods. Methods of the first category explore different heuristics such as information gain [37], minimum redundancy maximum relevance [59], and joint mutual information [30]. These algorithms can only work on supervised learning, whereas the ground truth is not always available in our case. Methods of category (2) calculate different statistical measures such as low variance [64], T-score [17], and Chi-Square Score [85]. These algorithms can only work on discrete data, while a preprocessing step is required for numerical and continuous data. Methods of the category (3) investigate similarity computation to associate weight importance on each feature candidate, such as Laplacian score [32], SPEC [83], Fisher score [58], and trace ratio criterion [33]. These algorithms proved excellent performance in both supervised and unsupervised scenarios, and they are easy to implement. We therefore opt for this category and particularly choose the SPEC algorithm, for which we propose an adaptation that enables to find a group of trajectory outliers. The process starts by applying the SPEC algorithm on the set of individual trajectory outliers. The output of this step is the descendant ranking of individual outliers in terms of the score feature relevance. Using the SPEC ranking vector of individual outliers, a search enumeration tree is generated by applying the Breadth First Search algorithm. If the quality of the current group candidate does

ALGORITHM 3: FS-GTO Algorithm

```
1: Input: $\Lambda = \{\Lambda_1, \Lambda_2 \dots \Lambda_n\}$: The set of all trajectories.
   $\mathcal{A}$: trajectory outlier detection algorithm.
   $\gamma$: density threshold.
2: Output: $\mathcal{G}^*$: sets of all group trajectory outliers.
3: $\mathcal{G}^+_{\mathcal{A}} \leftarrow \mathcal{A}(\Lambda)$
4: $Ranking \leftarrow SPEC(\mathcal{G}^+_{\mathcal{A}})$
5: $\mathcal{G}^* \leftarrow BFS(\mathcal{G}^+_{\mathcal{A}}, \gamma, Ranking)$
6: return $\mathcal{G}^*$
```

not reach the criteria in Definition 7, a backtracking procedure is launched by taking the next trajectory in the SPEC ranking vector. Algorithm 3 presents the pseudo-code of *FS*-GTO.

4.4 Improvement

In this part, extension of the proposed solutions by exploring ensemble learning and HPC is considered. This is for the purpose of increasing the accuracy and reducing the runtime.

4.4.1 Ensemble Learning. Each solution proposed in this work (including clustering, neighborhood computation, and FS) returns potential groups of trajectory outliers with varied quality, i.e., some groups are useful while others are not. In order to improve the accuracy of this output, we propose EL-GTO that is based on the use of ensemble learning [71, 92]. Three learners are first launched, DBSCAN-GTO, kNN-GTO, and FS-GTO. The three sets G^{DBSCAN}, G^{kNN}, and G^{FS} are then combined and merged to derive the final set of groups of trajectory outliers. The challenge in this approach is the combination of the results of each learner, which should ensure accurate final result, as well as the capability of capturing new relevant patterns. Some solutions select the best learner and discard the results from the remaining learners. These solutions are not able to detect outliers inside the output of the learners. Other solutions combine the trajectory outliers retrieved from all the learners, which have the capability to capture more trajectory outliers but only inside the outliers of the learners. In this work, we propose an efficient approach not only to detect the outliers of the learners, but also to capture other relevant patterns that have not been identified by the learners. The process of this approach is presented in the following:

(1) First, we suggest to determine the number of occurrences for each group of trajectory outliers on the three learners and ranking them accordingly. The results will be the groups that are highly frequent on the three learners. For instance, if there is two groups: the first group $\{\Lambda_1, \Lambda_2, \Lambda_4\}$ appears twice, one on G^{DBSCAN}, and another on G^{kNN}, and the second group $\{\Lambda_1, \Lambda_2, \Lambda_3\}$ appears only once on G^{FS}, then the first group is better ranked than the second one.
(2) Second, in case there are no redundancy among the outputs of the learners, the similarity between the group of trajectory outliers of different learners are calculated. If the similarity is less than γ, then the two group outliers of these learners are merged.

To summarize, in addition to accurately keeping the relevance of the GTOs of the three learners G^{DBSCAN}, G^{kNN}, and G^{FS}, the proposed ensemble learning method is able to capture more relevant group of trajectory outliers that are not in the initial outputs of the learners.

4.4.2 HPC. In this section, we first propose a generic approach to implement the proposed solutions on parallel architectures. An instantiation on GPU architecture of this generic approach is then presented.

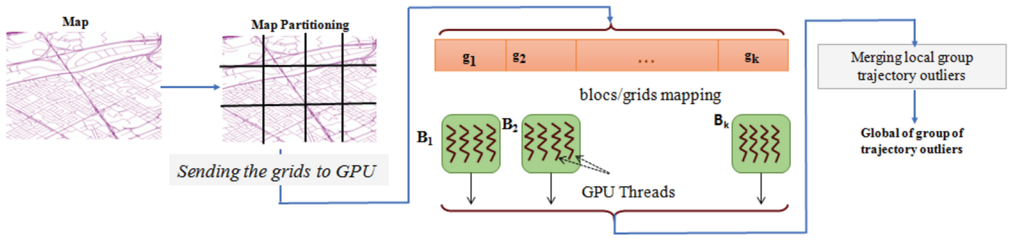


Fig. 3. GPU-GTO framework.

(a) Generic Approach. The following sequential steps are needed for running GTO solutions on any parallel architecture. (i) Map partitioning: in which the map is divided into several grids, whereby each grid contains a set of similar trajectories. This step is performed in Central Processing Unit (CPU). (ii) Computing and storing the local results: in this step, each parallel node applies one of the GTO solutions on every cluster and generates all GTOs from the grid that is assigned to it. The set of all group outliers is built following the same logic used in the serial implementation of the GTO solutions. Once the local GTOs are calculated, they will be sent to the CPU for further processing. (iii) Merging the local results: the local group of trajectory outliers are merged into a global one on the CPU. This can be done using a simple concatenation of all local results.
(b) GPU-GTO. The instantiation of the three steps defined above must be carefully designed to fit the targeted hardware. The GPUs' architecture has been gaining field during the last decade as a powerful computing resource [21, 22, 69, 70, 72]. This hardware is composed of two hosts: (i) the CPU and (ii) the GPU. The former contains the processors and the main memory. The latter is a multi-threading system that consists of multiple computing *cores*, where each core executes a block of threads. Threads of a block in the same core communicate with one another using a shared memory, whereas the communication between blocks relies on the global memory. The CPU/GPU communication is made possible by hardware buses. In the following, the adaptation of GTO for deployment on GPU architectures is denoted GPU-GTO. In GPU-GTO (see Figure 3), the map is first partitioned into k grids $\{g_1, \dots, g_k\}$ in the map partitioning step. The set of designed grids are then sent to the GPU. Each block of threads is mapped onto one grid, while the GTO solutions are applied on each block in parallel. If we consider the size of the shared memory of each block to be r, the fi rst r trajectories of the grid, g_i, are allocated to the shared memory of the block, and the remaining trajectories of g_i are allocated to the global memory of the GPU host. GPU-GTO defines a *local table*, $table_i$, for storing the group of trajectory outliers of the grid g_i. The local table of each grid is sent to CPU for further processing. In this context, CPU host merges the results to find the global group of trajectory outliers.

From a theoretical standpoint, GPU-GTO improves the GTO solutions by exploiting the massively threaded computing of GPUs while mining the grids of trajectories. GPU-GTO also minimizes the CPU/GPU communication by defining o nly t wo p oints o f C PU/GPU communication. The first o ne t akes p lace w hen t he g rids a re l oaded i nto t he G PU h ost, a nd t he second one when the local tables are returned to the CPU. GPU-GTO also provides an efficient memory management by using different l evels o f m emories i ncluding g lobal a nd s hared memories. However, GPU-GTO may suffer f rom a s ynchronization p roblem b etween t he G PU blocks. This takes place when the GPU blocks process grids with different n umber o f t rajectories. This issue degrades the performance of the GPU-based implementation of the GTO solutions. In real scenarios, different n umber o f t rajectories p er g rid m ay b e o btained. T his d epends o n t he way the trajectories are placed into the map. The more the sizes of the grids are different, the higher the synchronization cost of the GPU-based implementation will be. A solution to minimize the number of thread divergence (TD) is proposed in the following. The number of TD should first be determined. In the proposed GPU-based solution, every grid contains different number of trajectories. To identify the group of trajectory outliers on GPU, each thread compares trajectories to the grid it is mapped with. Consequently, TD may be caused by two reasons: First, each thread handles different number of trajectories. In this case, there are threads that finish before others. Second, the comparison process of a given thread is stopped when it does not find the trajectory outliers in the grid it is mapped with. These two parameters affect the number of TDs, which can be computed according to the number of comparisons by the different threads using Equation (6).

$$TD = max\{max\{|t_{(r*w)+i}| - |t_{(r*w)+j}|\}/(i,j,r) \in [1 \ldots w]^3\}, \tag{6}$$

where $|t_{(r*k)+i}|$ is the size of the $(r*k)+i^{th}$ trajectory that is assigned to the i^{th} thread and allocated to the r^{th} grid. Note that k is the number of grids.

Furthermore, TD can be computed according to the distribution of trajectories in the grids. The following two cases can be distinguished:

Irregular distribution of trajectories: This takes place when the grids are highly different in size. TD can be approximated in this case to,

$$\lim_{k \to +\infty} TD(m) = m - 1, \tag{7}$$

where is m the maximal number of trajectories.

Regular distribution of trajectories: This takes place when there is a slight difference between the size of grids in terms of trajectories. Let us consider r_1 the variation between grids. This yields,

$$\lim_{m \to +\infty} TD(m) = r_1. \tag{8}$$

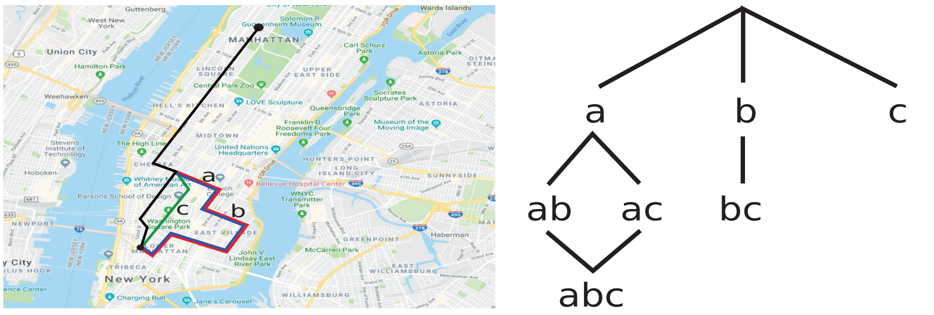
A solution that minimizes TD is proposed in the following, which also improves the assignment of the grids on different blocks. The assignment of the grids is performed according to the number of trajectories in each grid, and the grids of i trajectories are assigned to the i^{th} block. The number of blocs then is equal to the number of trajectories. The divergence between threads of the same grid is minimized this way, as the threads of each block have the same number of grids. However, the load balancing between blocks is not taken into account if many grids have the same number of trajectories. In fact, some blocks handle much more grids than the others. This degrades the performance of the GTO detection process on GPU. To deal with this problem, we propose to capture the grids that reduce the load balancing and to sort theme according to the number of trajectories. Every grid is then assigned to one thread, while the i^{th} grid is handled by the i^{th} thread. This way, all blocks have the same number of grids, which ensures load balancing between blocks.

5 DPD

5.1 Two-phase Approach

Motivated by the fact that a GTO is composed of individual sub-trajectory outliers, we propose in the following a two-phase approach that includes, (1) determining individual deviation and (2) determining GTOs and GDP.

(1) **Determining IDP:** The process starts by computing the area of every point in the whole set of trajectories Λ using the point area measure (see Definition 6). This allows deriving the IDP, if it exists, for each trajectory. From the IDP, the process continues determining the individual trajectory outlier. The area of every point that is highly ordered to the IDP is checked. If it is below a threshold, say μ, then this point is added to the individual trajectory outlier. This process is repeated until a normal point is found (a point with area



(a) First Phase (b) Second Phase

Fig. 4. Two-phase approach Illustration.

value is greater than μ). The whole process is repeated until all points of all trajectories are scanned.

(2) **Determining GTOs and GDP:** After determining the IDP, the next step is to derive the GTOs. A FS technique is used, in which every individual trajectory outlier is considered as one feature. The aim here is to select the most relevant features from the whole individual trajectory outliers. The relevant set of features is then considered as GTOs. The evaluation of the selected features is computed using the group density measure (see Definition 7), which should be maximized. A depth first strategy is used for this purpose, which starts with the empty node $\emptyset$ that may contain any trajectory, to the full node that contains all individual trajectory outliers. The GDP is finally obtained, which is the last IDP of all trajectories in each GTO.

Algorithm 4 presents the pseudo-code of the two-phase-based approach for solving the multi-view trajectory outlier detection problem. The input consists of the trajectory database Λ, the point area, and the group density thresholds. The output comprises the sets IDP, ITO, GDP, and GTO. The algorithm starts by computing the area of every point in the trajectory database (lines 4 throughout 8). It then constructs the sets, IDP and ITO, using the μ threshold (lines 13 throughout 30). The depth-first search is then applied to each ITO_j to determine the best features (GTOs) with the GDP (line 33 throughout 47). The algorithm uses the following pre-defined methods:

(1) *AddElementToList(L, e)*: Adds an element e to the end of the list L.
(2) *AddElementstoList(L, E)*: Adds all elements in E to the end of the list L.
(3) *CreateList(L, e)*: Creates a new list (L) and assign an element e as the head of this list.
(4) x $\leftarrow$ *RemoveFirstElement(L)*: Assigns to, x, the first element of the list, L, before removing it.
(5) *SaveBest(L))*: Returns the current best element in the list, L, with respect to the $\mathcal{DG}$ formula.
(6) *LastPoint(L)*: Returns the last point of the list L.

Figure 4 illustrates the Two-Phase algorithm on the set of trajectories shown in Figure 4(a). The first phase aims to identify the individual trajectory outliers, as well as the IDPs. We assume the trajectories {a, b, and c} are individual trajectory outliers and deviate from the whole trajectories marked by black color. The second phase aims to identify the group of trajectory outliers, and the GDPs. Therefore, an enumeration tree is generated in which the root is an empty set, where the nodes of the i^{th} level of the tree contains potential groups with i trajectories. The score of every

ALGORITHM 4: Two-phase-based algorithm

```
1: Input:
   $\Lambda = \{\Lambda_1, \Lambda_2 \ldots, \Lambda_m\}$: The trajectory database.
   $\mu$: The point area threshold.
   $\gamma$: The group density threshold.
2: Output:
   $IDP$, $GDP$, $GTO$.
3: {First Phase: Determine individual deviation point}
4: for i=1 to m do
5:     for j=1 to n do
6:         $d_{ij} \leftarrow \mathcal{PA}(p_{ij})$ {See Def. 6}
7:     end for
8: end for
9: $IDP \leftarrow \emptyset$
10: $ITO \leftarrow \emptyset$
11: $flag \leftarrow false$
12: $c \leftarrow 1$
13: for i=1 to m do
14:     for j=1 to n do
15:         if $d_{ij} \leq \mu$ then
16:             if flag=false then
17:                 $AddElementToList(IDP[i], p_{ij})$
18:                 $IDP \leftarrow IDP \cup p_{ij}$
19:                 $CreateList(ITO_c[i], p_{ij})$
20:                 $flag \leftarrow true$
21:             else
22:                 $AddElementToList(ITO_c[i], p_{ij})$
23:             end if
24:         else
25:             $flag \leftarrow false$
26:             $ITO \leftarrow ITO \cup ITO_c[i]$
27:             $c \leftarrow c + 1$
28:         end if
29:     end for
30: end for
31: {Second Phase: Determine group trajectory outliers, and group of deviation point}
32: $Open \leftarrow \emptyset$
33: for i=1 to m do
34:     if $IDP[i] \neq \emptyset$ then
35:         for j=1 to c do
36:             $AddElementToList(Open, ITO_j[i])$
37:         end for
38:     end if
39: end for
40: while $Open \neq \emptyset$ do
41:     $node \leftarrow RemoveFirstElement(Open)$
42:     $\mathcal{DG}(node)$ {See Def. 7}
43:     $AddElementsToList(Open, GPN(node))$
        {With respect to Def. 8}
44:     $Best \leftarrow SaveBest(Open)$
45: end while
46: $GTO \leftarrow Best$
47: $GDP \leftarrow LastPoint(IDP)$
48: return $(IDP, GDP, GTO)$
```

node is evaluated, and the best group is derived. At each level, the potential GDPs are saved as described by Algorithm 4.

The complexity of Algorithm 4 depends on the number of trajectories, m, the number of points, n, and the number of the individual trajectory outliers generated in the first stage, c. The cost is the sum of costs for the first and the second phases. In the first phase, the area computation requires $m^2 \times n$ operations, while the construction of the individual trajectory outliers is performed in m operations. The total cost of this stage is then $(m^2 \times n) + m$. The FS in the second phase is performed on the c individual trajectory outliers, which generates $(2^c - 1)$ nodes, and c operations is needed to evaluate every node. The total cost of this phase is $c \times 2^c - c$. The total cost of

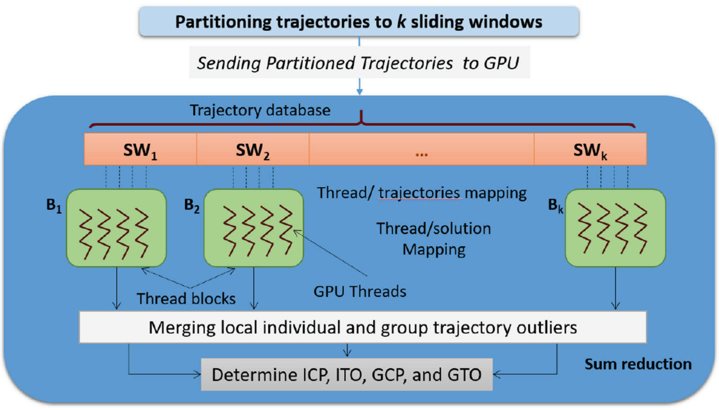

Partitioning trajectories to k sliding windows
Sending Partitioned Trajectories to GPU
Trajectory database
SW1
SW2
...
SWk
B1
B2
Bk
Thread/ trajectories mapping
Thread/solution Mapping
GPU Threads
Thread blocks
Merging local individual and group trajectory outliers
Determine ICP, ITO, GCP, and GTO
Sum reduction

Fig. 5. GSW-TP illustration.

the two-phase algorithm is of complexity $O(m^2 \times n + c \times 2^c)$. From this analysis, we conclude that the two-phase algorithm is quadratic on the number of trajectories, polynomial on the number of points, and exponential on the number of individual trajectory outliers. The overall performance of this algorithm is then hugely affected by the number of trajectories. In the next section, we propose a GPU-based sliding windows algorithm for boosting the performance of the two-phase algorithm.

5.2 GSW-TP: GPU-based Sliding Windows for Two-Phase-Based Algorithm

The aim of this approach is to improve the overall performance of the two-phase-based algorithm proposed in the previous section using a sliding window approach, as well as the GPU architecture. Figure 5 illustrates how the proposed approach benefits from the GPU threaded to improve the runtime performance of the two-phase algorithm. The trajectory database is first divided into k sliding windows, where k is the number of the GPU-blocks used in the mining process. This step is performed sequentially on CPU host. The k sliding windows are then sent to the GPU global memory using to the CPU/GPU communication channels. Each sliding window, SW_i, is transmitted to the shared memory of the bloc, b_i, where the threads of the block, b_i, are mapped to the trajectories of, SW_i. Therefore, the j^{th} thread in b_i, say th_{ij}, determines the local individual trajectory outliers on the trajectory Λ_j (lines 4 throughout 30 of Algorithm 4). If the trajectory Λ_j's ending point area on the sliding window SW_{i-1}, and Λ_j's starting point area on the sliding window SW_i are less than μ, then the part of the individual trajectory outlier Λ_j on SW_{i-1} is concatenated with the part of Λ_j on SW_i. In this case, the average point will be the average of both parts of Λ_j. Afterwards, the threads of each block compute the local average points for every individual trajectory outliers. Each block finds the local GTO at every sliding window and sends the local results to the GPU host global memory. A global GTO will be a local trajectory outlier that maximizes a function described in Definition 9. Once the individual trajectory outliers, the group of trajectory outliers, and the deviation points are determined, they will be sent to the global memory of the CPU host. The GSW-TP improves the sequential version of the two-phase based algorithm by exploiting the massively threaded computing of GPUs while determining both individual and global trajectory outliers. GSW-TP also minimizes the CPU/GPU communication and reduces it to two points. The first one takes at the beginning place when the trajectory database is loaded into the GPU, while the second when the individual and the global trajectory outliers are returned to the host memory. Moreover, GSW-TP minimizes TD, which is a typical problem in GPU-based computing. TD only takes place when the threads of different blocks process different number of individual trajectory outliers, which needs several GPU synchronization points. A strategy that minimizes the number of TD is proposed in the following. The number of TD is first determined similarly to GPU-GTO (using Equations (7) and (8)). The previous strategy developed in GPU-GTO divides the grids on all blocks and minimizes the TD between threads while respecting the load balancing between blocks. However, if the size of blocks is too large, then any two threads of the same block cannot be allocated to the same block. The first thread handles the grid of size $lenght$, and the second handles the one of size $lenght + x$, This creates additional TD due the difference in the size of the grids. To deal with this problem, we propose to fix the size of blocks according to the statistical analysis of the grids. Similarly the previous strategy, the i^{th} element of the grid's vector determines the number of the i^{th} grid's trajectories. The median of this vector is proportional to the size of each block. The same process used in GPU-GTO is then repeated. To determine the median of a given vector of l elements, this vector is first sorted and then $\frac{l}{2}$ is considered as the median value.

6 PERFORMANCE EVALUATION

Intensive experiments have been carried out to evaluate the proposed algorithms on different trajectory databases in four steps. (1) Serial implementation of the GTO solutions are compared with the state-of-the art group outlier detection algorithms using standard trajectory databases. (2) The ability of the GPD solution (two-phase-based algorithm) to detect deviation points is investigated in several scenarios. (3) The scalability of the GPU-based solutions is investigated on big trajectory databases and compared with the existing GPU-based outlier detection solutions. (4) A case study of the proposed framework on intelligent transportation is illustrated.

In all experiments, the time period of each interval was set to 5 minutes. A $64 - bit$ computer was used for the serial implementation, which features a core $i7$ processor running Windows 10 and $16GB$ of RAM. GPU-based implementations were carried out on a CPU host coupled with a GPU device. The CPU host was a $64 - bit$ quad-core Intel Xeon $E5520$, with $2.27GHz$ clock. The GPU device was a $1.15GHz$ Nvidia Tesla $C2075$ with 448 CUDA cores (14 multiprocessors with 32 cores each), $2.8GB$ of global memory, and $49.15KB$ of shared memory. Both the CPU and GPU were used in single precision. The evaluation procedure of the returned outliers' quality represents a common problem of outlier detection techniques, in particular for new applications such as GTOs and DPD in which a ground truth is not defined or does not exist. Due to the lack of ground truth in trajectory datasets, we injected synthetic GTOs as follow:

Injecting individual trajectory outliers: Individual trajectory outliers were generated by adding noise *several times* with a certain probability $p \sim \mathcal{U}(0.8, 1.0)$ and a given threshold μ.

Injecting GTOs: From the individual trajectory outliers, noise was added *few times* with a certain probability $p \sim \mathcal{U}(0.0, 1.0)$ and a given μ.

In both cases, each point p_{il} in the trajectory Λ_i was changed as,

$$p_{il} = \begin{cases} p_{il} + n \sim \mathcal{N}(0,1) & \text{if } p \geq \mu, \\ p_{il} & \text{otherwise.} \end{cases} \tag{9}$$

For both individual and group trajectory outliers, the starting noise points are considered as individual and group deviation points. In the following, the evaluation of the GTOs is performed using Fmeasure, and ROCAUC, which are common measures for the evaluation of the outlier detection methods.

Table 1. Best Parameters of the Proposed Solutions

Data	DBSCAN-GTO (User Threshold)	kNN-GTO (k)	FS-GTO (Tree Depth)	Two-Phase (μ)
Intelligent Transportation	4	6	6	0.8
Climate Change	6	5	6	0.6
Environment	6	5	7	0.5

6.1 Data Description

The following datasets have been used.

Intelligent Transportation: database from the *ECML PKDD 2015* competition[1] has been used. It contains 7, 733 real trajectories retrieved from 01/07/2013 to 30/06/2014 of 442 taxis in the city of Porto, Portugal. Further information about this trajectory database can be found in [57].

Climate Change: Atlantic hurricanes track dataset has been used [40], which contains latitude, longitude, maximum sustained surface wind, and minimum sea-level pressure of hurricane trajectories in USA at 6 hourly intervals for the period from 1851 to 2018. This includes 52, 775 hurricane trajectories.

Environment: Starkey Project's dataset [2] has been used. It includes animal movement data illustrated by the radio-telemetry locations of elk, deer, and cattle, collected from 1989 to 1999. The locations has been recorded at 30-minute intervals. With 100 trajectories, and more than 40,000 different points. This dataset is considered sparse.

Big Databases: Two big trajectory databases have also been used: (i) taxi $13-1$ containing 1.89 million trajectories, and (ii) taxi $13-2$ containing 3.69 million trajectories. Both databases were generated from the taxis of Shanghai during the period from 01/10/2013 to 31/12/2013 [52].

6.2 Serial GTO Performance

The aim of the first part of this experiment is to tune parameters related to the different solutions proposed in this article. Several tests have been carried out by varying the user threshold from 1 to 10 for DBSCAN, the number of neighborhood from 1 to 10 for kNN, the tree depth from 1 to 10 for FS, the population size from 10 to 100 for computational intelligence improvement, and the Point Area threshold from 0.1 to 1.0 for the two-phase-based algorithm. The results show that for trajectory databases (intelligent transportation, climate change, or environment), the accuracy of DBSCAN and kNN increases up to reaching the optimum point and then starts decreasing. The accuracy of the other solutions (FS, computational intelligence, and the two-phase-based algorithm) also goes up with the increase of the overspending parameters until reaching the optimum values then stabilizes. Table 1 summarizes the best parameters' values of the proposed solutions for different trajectory databases, which are used in the remaining of the article.

The second part of the experiment aims to compare the proposed solutions with the state-of-the art algorithms in terms of accuracy and processing time. To the best of our knowledge, this is the first work that explores GTO detection, i.e., there is no relevant candidate dealing with GTO to compare with. Therefore, comparing the proposed solutions with some generic group outlier detection algorithms is the only option we have. For this purpose, four baseline algorithms have been adapted to trajectory data including DGM [11], WATCH [45], ATD [65], and AGJFD

[1]http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html.
[2]https://www.fs.fed.us/pnw/starkey/introduction.shtml.

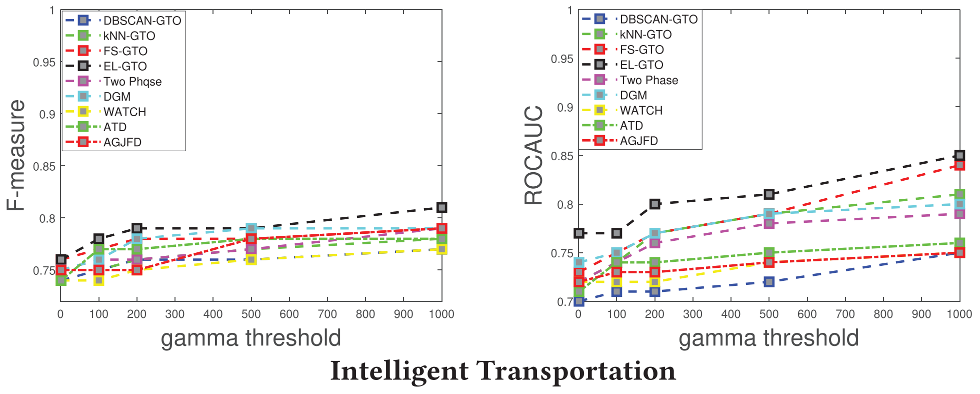


Fig. 6. Accuracy on intelligent transportation dataset.

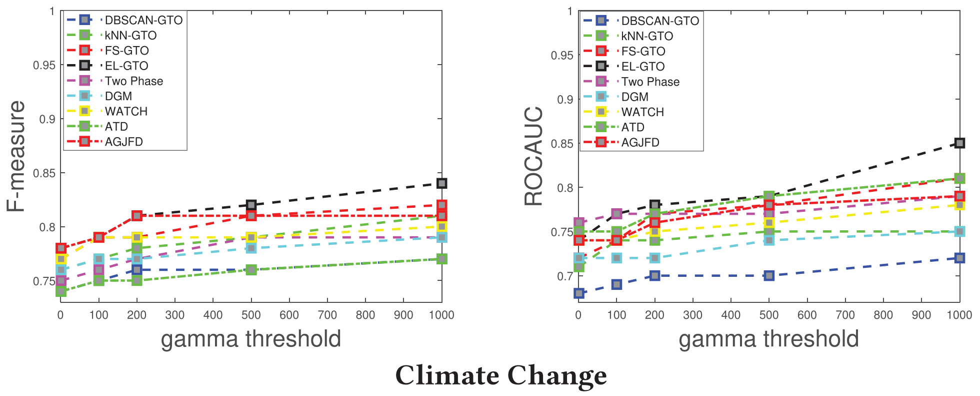


Fig. 7. Accuracy on climate change dataset.

[66]. Figures 6–9 show the accuracy and the runtime of the proposed solutions (DBSCAN-GTO, kNN-GTO, FS-GTO, EL-GTO, and the two-phase-based algorithm) in comparison with the above-mentioned baseline algorithms. The figures show that solutions based on FS and ensemble learning methods outperform the baseline algorithms in terms of accuracy (this holds for trajectory databases). However, solutions based on neighborhood computation, DBSCAN, and the two-phase algorithms show lower performance. The results also reveal that solutions demonstrating high accuracy needs higher run time, which represents the main obstacle when dealing with big data trajectory. The parallel solutions that proposed to deal with this problem are evaluated in the following.

The last part of this experiment is to show the performance of the two-phase algorithm for identifying both individual and group deviation points. To the best of our knowledge, there is no work that explores DPD. We use Equation (9) to calculate the ability of the two-phase algorithm for identifying starting noise points of trajectories. Figure 10 presents the error rate (%) of the two-phase algorithm using different number of individual and group deviation points, with different trajectory databases. The results show that the error rate slightly increases with the rise in the number of deviation points for all trajectory databases, and that the rate for the intelligent transportation is the highest. They also reveal that the two-phase algorithm has the ability to identify the individual and group deviation points for more than 67%, even for complex data.

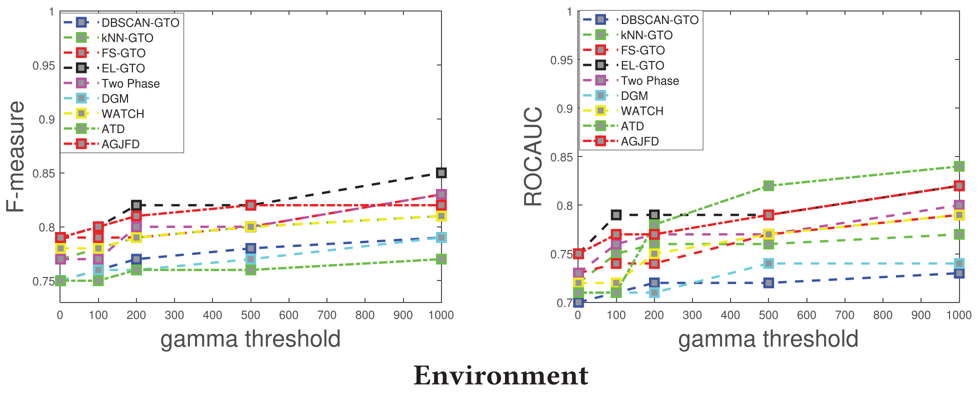


Fig. 8. Accuracy on environment dataset.

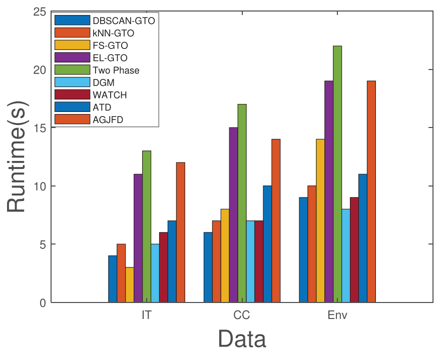


Fig. 9. Runtime: the proposed solutions vs. state-of-the art group detection.

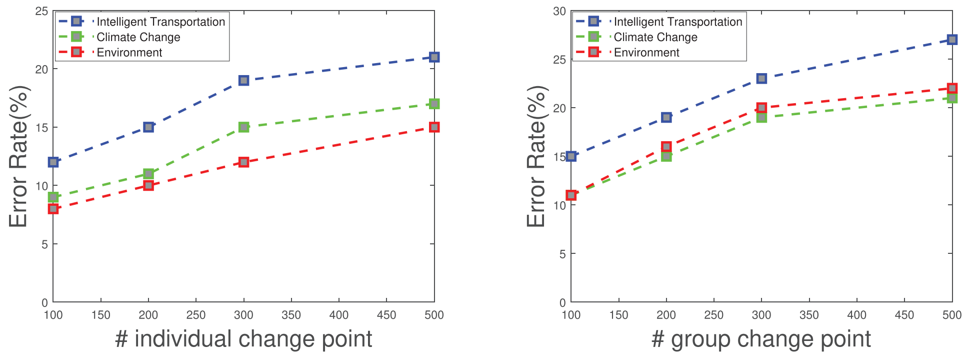


Fig. 10. Error rate (%) of the two-phase approach for detecting individual and group deviation points.

6.3 Performance of Parallel Solutions

Several experiments have been carried out on GPU architecture using big trajectory databases. The results reported in Figures 11 and 12 show that the speed rises with the increase of number of blocks and the number of threads per bloc, more considerably for ensemble learning and computation intelligence based solutions. This performance is justified by the partitioning strategy for mapping the trajectories on GPU blocks that takes advantage of the massively GPU threaded.

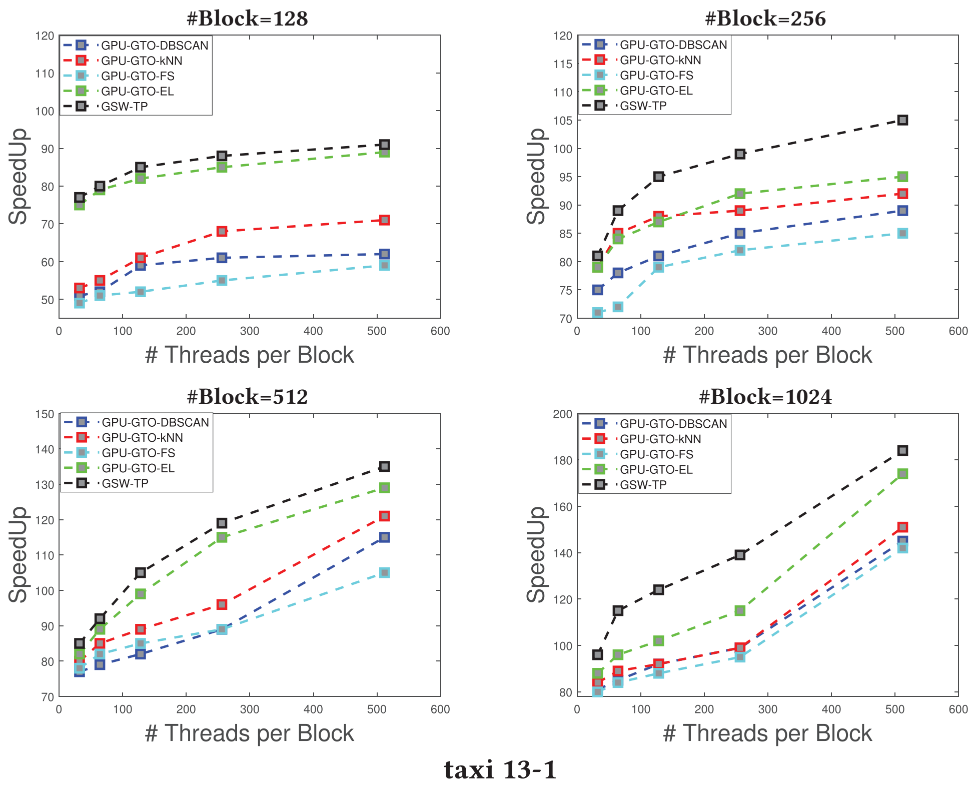


Fig. 11. Speed of the GPU-based GTO solutions taxi 13-1 dataset.

Figure 13 compares the proposed GPU solution that uses the ensemble learning (GPU-GTO), the two-phase-based (GSW-TP), with the baseline GPU-based outlier detection algorithms (SolvingSet [3], SolvingSet+ [4], and MoNavGPU [81]) on big trajectory databases. The figure shows that the proposed solutions outperform the baseline approaches. The runtime of the proposed solutions does not exceed 300*seconds* to deal with the whole trajectory taxi $13-2$ database, while the other solutions need 500 seconds for processing such trajectory database. These results can be explained by: (i) the use of intelligent mapping between trajectories and the GPU blocks, (2) the parallelism of ensemble learning approach, and (3) the efficient exploration of the trajectory space by the genetic operators.

6.4 Case Study on Intelligent Transportation

The last experiment aims at demonstrating the usefulness of the proposed framework in a real-world scenario. The taxi trajectory service of Porto[3] is used. Figure 14(a) presents the number of trajectory outliers with different number of trajectory sizes and different area point threshold values of the GSW-TP. The figure shows that the number of individual trajectory outliers increases as the number of trajectories goes up. The number of trajectory outliers increases from 35 for 700 trajectories to reaches 72 for 7,000 trajectories. However, when the area point threshold (density)

[3]http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html.

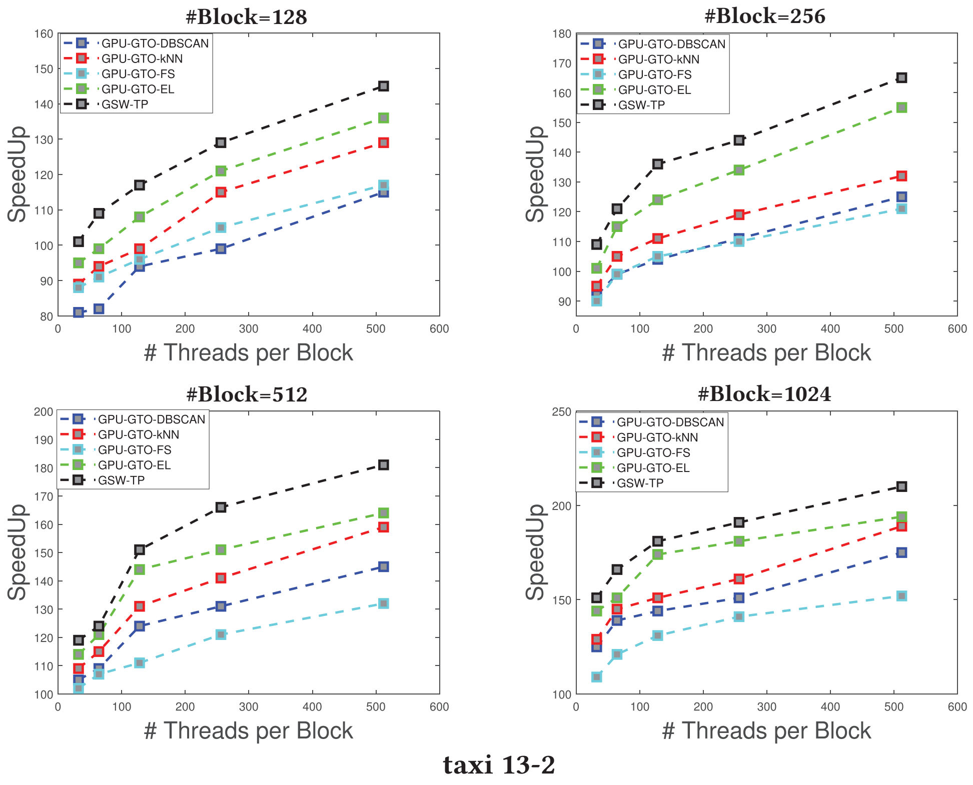


Fig. 12. Speed of the GPU-based GTO solutions for taxi 13-2 dataset.

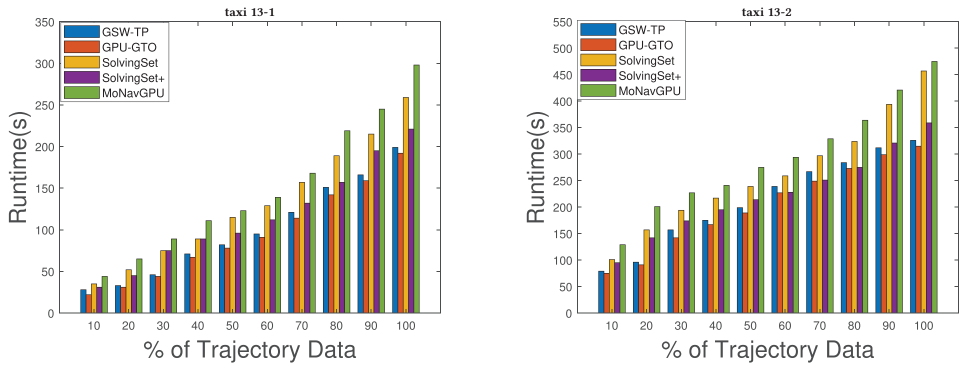
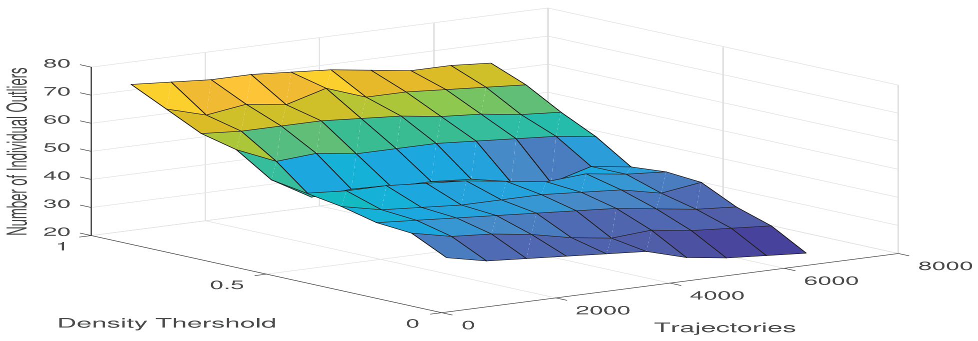
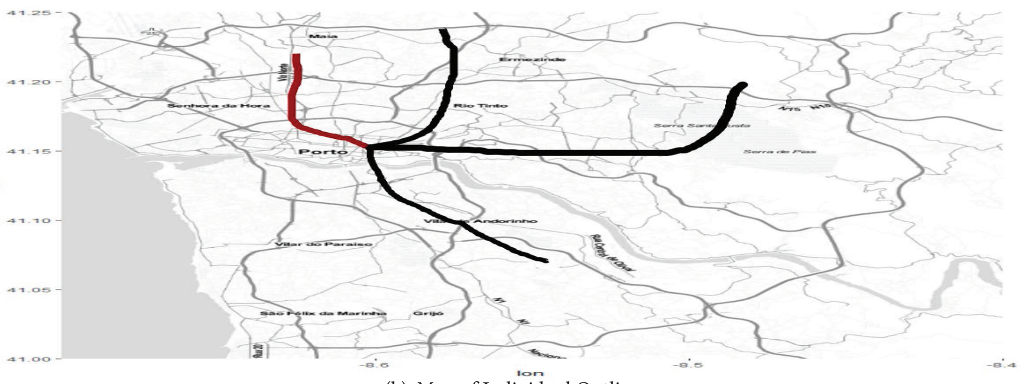


Fig. 13. The proposed GPU-based solutions vs. state-of-the art GPU-based outlier detection.

increases, the number of individual trajectory outliers decreases. Figure 14(b) shows the results of both individual and group trajectory outliers by applying the two-phase algorithm on the same trajectory database. From this figure, we remark that the two-phase algorithm is able to detect GTOs (marked by red color) against normal trajectories (marked by black color). These GTOs represent different taxi trips. One of the reason that these taxis deviate from the normal trajectory is the high traffic jam of Porto city in peak hours.



(a) Number Of Individual Outliers



(b) Map of Individual Outliers

Fig. 14. Case study of real taxi trajectory database.

7 DISCUSSION AND FUTURE DIRECTIONS

The first finding in this work is that the proposed framework identifies new patterns represented by group of trajectory outliers. This is different from previous trajectory outlier detection approaches, which are only able to derive individual trajectory outliers. The second finding is that the combination of several concepts (from different fields) improves the overall performance in detecting group of trajectory outliers (for both quality and runtime). This includes exploring the micro clusters, nearest neighbors, feature of the individual trajectory outliers, ensemble learning, genetic operators, and HPC. The proposed GTO solutions are examples of the application of outlier detection algorithms to the context of trajectory outliers. The literature calls for this type of research, particularly for urban analysis and smart city applications where a large number of trajectories are present in the daily life. However, porting a data mining and machine learning approach to any specific application domain always requires methodological refinement and adaptation [47, 65].

While this work is the first milestone in the context of *GTO detection*, much investigation is still required before reaching advanced solutions that could be exploited by city planners. A deep progress in all directions is recommended including, (i) techniques for GTO: more sophisticated techniques should be developed for the *GTO* problem. For instance, other traditional outlier detection techniques may be adopted such as Local Outlier Factor [6]. This is by developing and introducing new concepts of density, local reachability density for *GTO* problem. (ii) Visualization: new visualization techniques should be developed for *GTO* providing city planners with tools that enable visualizing and interpreting GTOs. (iii) GTO applications: investigating and targeting new applications of *GTO*, such as climate change analysis. In this case, a typical example is finding a group of hurricane trajectories that deviates from the normal hurricane trajectories. This allows to identify and explore other cities that could be affected by the *Hurricanes*. The last hurricanes observed in the United States during the period 2018–2019 could be a real sketch of this study.

8 CONCLUSION

The problem of GTO detection has been introduced in this article. Three algorithms (DBSCAN-GTO, *k*NN-GTO, and FS-GTO) have been proposed and investigated from a set of individual trajectory outliers. In DBSCAN-GTO, the candidate trajectories are first determined using the DBSCAN algorithm. The density computation is then calculated for each candidate trajectory to find the group of trajectory outliers. The process in *k*NN-GTO starts recursively by determining the potential candidates from the individual trajectory outliers and pruning them using density computation. This repeats for all individual trajectory outliers, and then a set of GTOs is determined in which outliers in every group share some common points. FS-GTO considers the GTO problem as a FS problem, while the set of individual trajectory outliers are viewed as the set of all features, and the FS process is applied to identify the group of trajectory outliers. Furthermore, three improvements for these algorithms have been suggested by using ensemble learning, computational intelligence, and HPC. All the proposed approaches have been tested on real trajectory databases. The results reveal the usefulness of exploring computational intelligence and HPC in identifying GTOs, and the superiority of the proposed approaches over the baseline group detection. The results also show that the GPU-parallel approach outperforms the existing HPC approaches when dealing with big trajectory databases.

REFERENCES

[1] A. Sharifi. 2020. A typology of smart city assessment tools and indicator sets. *Sustainable Cities and Society* 53 (2020), 101936

[2] Georgi Ajaeiya, Imad H. Elhajj, Ali Chehab, Ayman Kayssi, and Marc Kneppers. 2018. Mobile apps identification based on network flows. *Knowledge and Information Systems* 55, 3 (2018), 1–26.

[3] Fabrizio Angiulli, Stefano Basta, Stefano Lodi, and Claudio Sartori. 2013. Fast outlier detection using a GPU. In *Proceedings of the 2013 International Conference on High Performance Computing & Simulation (HPCS'13)*. IEEE, 143–150.

[4] Fabrizio Angiulli, Stefano Basta, Stefano Lodi, and Claudio Sartori. 2016. GPU strategies for distance-based outlier detection. *IEEE Transactions on Parallel and Distributed Systems* 27, 11 (2016), 3256–3268.

[5] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. 2018. Spatio-temporal data mining: A survey of problems and methods. *ACM Computing Surveys* 51, 4 (2018), 83.

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying density-based local outliers. *ACM SIGMOD Record* 29, 2 (2000), 93–104.

[7] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. 2015. Map-based probabilistic visual self-localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 4 (2015), 652–665.

[8] Nan Cao, Chaoguang Lin, Qiuhan Zhu, Yu-Ru Lin, Xian Teng, and Xidao Wen. 2018. Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 23–33.

[9] Hervé Cardot, Peggy Cénac, and Jean-Marie Monnez. 2012. A fast and recursive algorithm for clustering large datasets with k-medians. *Computational Statistics & Data Analysis* 56, 6 (2012), 1434–1449.

[10] Gilles Celeux, Didier Chauveau, and Jean Diebolt. 1996. Stochastic versions of the EM algorithm: An experimental study in the mixture case. *Journal of Statistical Computation and Simulation* 55, 4 (1996), 287–314.

[11] Raghavendra Chalapathy, Edward Toth, and Sanjay Chawla. 2018. Group anomaly detection using deep generative models. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 173–189.

[12] Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, Shijian Li, and Zonghui Wang. 2013. iBOAT: Isolation-based online anomalous trajectory detection. *IEEE Transactions on Intelligent Transportation Systems* 14, 2 (2013), 806–818.

[13] Raffaele Conforti, Marcello La Rosa, and Arthur H. M. ter Hofstede. 2017. Filtering out infrequent behavior from business process event logs. *IEEE Transactions on Knowledge and Data Engineering* 29, 2 (2017), 300–314.

[14] Kaustav Das and Jeff Schneider. 2007. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 220–229.

[15] Kaustav Das, Jeff Schneider, and Daniel B. Neill. 2008. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 169–176.

[16] Mahashweta Das and Srinivasan Parthasarathy. 2009. Anomaly detection and spatio-temporal analysis of global climate system. In *Proceedings of the 3rd International Workshop on Knowledge Discovery from Sensor Data*. 142–150.

[17] David Dernoncourt, Blaise Hanczar, and Jean-Daniel Zucker. 2014. Analysis of feature selection stability on high dimension and small sample data. *Computational Statistics & Data Analysis* 71, C (2014), 681–693.

[18] Djamel Djenouri, Roufaida Laidi, Youcef Djenouri, and Ilangko Balasingham. 2019. Machine learning for smart building applications: Review and taxonomy. *ACM Computing Surveys* 52, 2 (2019), 1–36.

[19] Youcef Djenouri, Asma Belhadi, Jerry Chun-Wei Lin, and Alberto Cano. 2019. Adapted K-nearest neighbors for detecting anomalies on spatio–temporal traffic flow. *IEEE Access* 7 (2019), 10015–10027.

[20] Youcef Djenouri, Asma Belhadi, Jerry Chun-Wei Lin, Djamel Djenouri, and Alberto Cano. 2019. A survey on urban traffic anomalies detection algorithms. *IEEE Access* 7 (2019), 12192–12205.

[21] Youcef Djenouri, Ahcene Bendjoudi, Zineb Habbas, Malika Mehdi, and Djamel Djenouri. 2017. Reducing thread divergence in GPU-based bees swarm optimization applied to association rule mining. *Concurrency and Computation: Practice and Experience* 29, 9 (2017), e3836.

[22] Youcef Djenouri, Djamel Djenouri, Asma Belhadi, and Alberto Cano. 2019. Exploiting GPU and cluster parallelism in single scan frequent itemset mining. *Information Sciences* 496 (2019), 363–377.

[23] John D. Eblen, Charles A. Phillips, Gary L. Rogers, and Michael A. Langston. 2012. The maximum clique enumeration problem: Algorithms, applications, and implementations. In *Proceedings of the 7th International Symposium on Bioinformatics Research and Applications*. Vol. 13.

[24] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. 226–231.

[25] Jerome H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.

[26] Yong Ge, Hui Xiong, Zhi-hua Zhou, Hasan Ozdemir, Jannite Yu, and Kuo Chu Lee. 2010. Top-eye: Top-k evolving trajectory outlier detection. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, 1733–1736.

[27] Stuart Geman and Donald Geman. 1987. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in Computer Vision*. Elsevier, 564–584.

[28] Mohammadhossein Ghahramani, MengChu Zhou, and Chi Tin Hon. 2018. Mobile phone data analysis: A spatial exploration toward hotspot detection. *IEEE Transactions on Automation Science and Engineering* 16, 1 (2018), 351–362.

[29] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. 2012. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 776–781.

[30] Baofeng Guo and Mark S. Nixon. 2008. Gait feature subset selection by mutual information. *IEEE Transactions on Systems, MAN, and Cybernetics-part a: Systems and Humans* 39, 1 (2008), 36–46.

[31] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2014. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (2014), 2250–2267.

[32] Xiaofei He, Deng Cai, and Partha Niyogi. 2006. Laplacian score for feature selection. In *Proceedings of the Advances in Neural Information Processing Systems*. 507–514.

[33] Yi Huang, Dong Xu, and Feiping Nie. 2012. Semi-supervised dimension reduction using trace ratio criterion. *IEEE Transactions on Neural Networks and Learning Systems* 23, 3 (2012), 519–526.

[34] Ilias Kalamaras, Alexandros Zamichos, Athanasios Salamanis, Anastasios Drosou, Dionysios D. Kehagias, Georgios Margaritis, Stavros Papadopoulos, and Dimitrios Tzovaras. 2018. An interactive visual analytics platform for smart intelligent transportation systems management. *IEEE Transactions on Intelligent Transportation Systems* 19, 2 (2018), 487–496.

[35] Elmouatezbillah Karbab, Djamel Djenouri, Sahar Boulkaboul, and Antoine B. Bagula. 2015. Car park management with networked wireless sensors and active RFID. In *Proceedings of the IEEE International Conference on Electro/Information Technology*. IEEE, 373–378.

[36] Tung Kieu, Bin Yang, and Christian S Jensen. 2018. Outlier detection for multidimensional time series using deep neural networks. In *Proceedings of the 2018 19th IEEE International Conference on Mobile Data Management (MDM'18)*. IEEE, 125–134.

[37] Kenji Kira and Larry A. Rendell. 1992. A practical approach to feature selection. In *Machine Learning Proceedings 1992*. Elsevier, 249–256.

[38] Xiangjie Kong, Ximeng Song, Feng Xia, Haochen Guo, Jinzhong Wang, and Amr Tolba. 2017. LoTAD: Long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web* 21, 3 (2017), 1–23.

[39] Roufaida Laidi and Djamel Djenouri. 2018. UDEPLOY: User-driven learning for occupancy sensors DEPLOYment in smart buildings. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops*. IEEE Computer Society, 209–214.

[40] Christopher W. Landsea and James L. Franklin. 2013. Atlantic hurricane database uncertainty and presentation of a new database format. *Monthly Weather Review* 141, 10 (2013), 3576–3592.

[41] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. 2008. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. 140–149.

[42] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. 593–604.

[43] Huanhuan Li, Jingxian Liu, Kefeng Wu, Zaili Yang, Ryan Wen Liu, and Naixue Xiong. 2018. Spatio-temporal vessel trajectory clustering based on data mapping and density. *IEEE Access* 6 (2018), 58939–58954.

[44] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2018. Feature selection: A data perspective. *ACM Computing Surveys* 50, 6 (2018), 94.

[45] Junli Li, Jifu Zhang, Ning Pang, and Xiao Qin. 2018. Weighted outlier detection of high-dimensional categorical data using feature grouping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 99 (2018), 1–14.

[46] Sheng Li, Ming Shao, and Yun Fu. 2018. Multi-view low-rank analysis with applications to outlier detection. *ACM Transactions on Knowledge Discovery from Data* 12, 3 (2018), 32.

[47] Wenjia Li, Houbing Song, and Feng Zeng. 2018. Policy-based secure and trustworthy sensing for internet of things in smart cities. *IEEE Internet of Things Journal* 5, 2 (2018), 716–723.

[48] Yang Li, Qixing Huang, Michael Kerber, Lin Zhang, and Leonidas Guibas. 2013. Large-scale joint map matching of GPS traces. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 214–223.

[49] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. IEEE, 413–422.

[50] Zhipeng Liu, Dechang Pi, and Jinfeng Jiang. 2013. Density-based trajectory outlier detection algorithm. *Journal of Systems Engineering and Electronics* 24, 2 (2013), 335–340.

[51] Zhongjian Lv, Jiajie Xu, Pengpeng Zhao, Guanfeng Liu, Lei Zhao, and Xiaofang Zhou. 2017. Outlier trajectory detection: A trajectory analytics based approach. In *Proceedings of the International Conference on Database Systems for Advanced Applications*. Springer, 231–246.

[52] Jiali Mao, Pengda Sun, Cheqing Jin, and Aoying Zhou. 2018. Outlier detection over distributed trajectory streams. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 64–72.

[53] Jiali Mao, Tao Wang, Cheqing Jin, and Aoying Zhou. 2017. Feature grouping-based outlier detection upon streaming trajectories. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2696–2709.

[54] Amin Hosseinpoor Milaghardan, Rahim Ali Abbaspour, and Christophe Claramunt. 2018. A Dempster-Shafer based approach to the detection of trajectory stop points. *Computers, Environment and Urban Systems* 70 (2018), 189–196.

[55] Natwar Modani and Kuntal Dey. 2008. Large maximal cliques enumeration in sparse graphs. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, 1377–1378.

[56] Andrew Moore and Weng-Keen Wong. 2003. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In *Proceedings of the 20th International Conference on International Conference on Machine Learning*. Vol. 3. 552–559.

[57] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.

[58] Feiping Nie, Shiming Xiang, Yangqing Jia, Changshui Zhang, and Shuicheng Yan. 2008. Trace ratio criterion for feature selection. In *Proceedings of the 23rd National Conference on Artificial intelligence*. Vol. 2. 671–676.

[59] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 27, 8 (2005), 1226–1238.

[60] Anatolii Prokhorchuk, Justin Dauwels, and Patrick Jaillet. 2019. Estimating travel time distributions by Bayesian network inference. *IEEE Transactions on Intelligent Transportation Systems* 21, 5 (2019), 1867–1876.

[61] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record* 29, 2 (2000), 427–438.

[62] Mahsa Salehi, Christopher Leckie, James C. Bezdek, Tharshan Vaithianathan, and Xuyun Zhang. 2016. Fast memory efficient local outlier detection in data streams. *IEEE Transactions on Knowledge and Data Engineering* 28, 12 (2016), 3246–3260.

[63] Hansi Senaratne, Manuel Mueller, Michael Behrisch, Felipe Lalanne, Javier Bustos-Jiménez, Jörn Schneidewind, Daniel Keim, and Tobias Schreck. 2018. Urban mobility analysis with mobile network data: A visual analytics approach. *IEEE Transactions on Intelligent Transportation Systems* 19, 5 (2018), 1537–1546.

[64] Kai-Quan Shen, Chong-Jin Ong, Xiao-Ping Li, Zheng Hui, and Einar PV Wilder-Smith. 2007. A feature selection method for multilevel mental fatigue EEG classification. *IEEE Transactions on Biomedical Engineering* 54, 7 (2007), 1231–1237.

[65] Hossein Soleimani and David J. Miller. 2016. ATD: Anomalous topic discovery in high dimensional discrete data. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2267–2280.

[66] Chenfei Sun, Zhongmin Yan, Qingzhong Li, Yongqing Zheng, Xudong Lu, and Lizhen Cui. 2019. Abnormal group-based joint medical fraud detection. *IEEE Access* 7 (2019), 13589–13596.

[67] Guanting Tang, Jian Pei, James Bailey, and Guozhu Dong. 2015. Mining multidimensional contextual outliers from categorical relational data. *Intelligent Data Analysis* 19, 5 (2015), 1171–1192.

[68] Edward Toth and Sanjay Chawla. 2018. Group deviation detection methods: A survey. *ACM Computing Surveys* 51, 4 (2018), 77.

[69] Md Zia Uddin. 2019. A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system. *Journal of Parallel and Distributed Computing* 123 (2019), 46–53.

[70] Thé Van Luong, Nouredine Melab, and El-Ghazali Talbi. 2013. GPU computing for parallel local search metaheuristic algorithms. *IEEE Transactions on Computers* 62, 1 (2013), 173–185.

[71] Jan N. van Rijn, Geoffrey Holmes, Bernhard Pfahringer, and Joaquin Vanschoren. 2018. The online performance estimation framework: Heterogeneous ensemble learning for data streams. *Machine Learning* 107, 1 (2018), 149–176.

[72] José R. Vázquez-Canteli, Stepan Ulyanin, Jérôme Kämpf, and Zoltán Nagy. 2019. Fusing TensorFlow with building energy simulation for intelligent energy management in smart cities. *Sustainable Cities and Society* 45 (2019), 243–257.

[73] Hao Wu, Weiwei Sun, and Baihua Zheng. 2017. A fast trajectory outlier detection approach via driving behavior modeling. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 837–846.

[74] Liang Xiong, Barnabás Póczos, Jeff Schneider, Andrew Connolly, and Jake VanderPlas. 2011. Hierarchical probabilistic models for group anomaly detection. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 789–797.

[75] Liang Xiong, Barnabás Póczos, and Jeff G. Schneider. 2011. Group anomaly detection using flexible genre models. In *Proceedings of the Advances in Neural Information Processing Systems*. 1071–1079.

[76] Cao Lei Yu, Yanwei, Elke A. Rundensteiner, and Qin Wang. 2017. Outlier detection over massive-scale trajectory streams. *ACM Transactions on Database Systems* 42, 2 (2017), 10.

[77] Qingying Yu, Yonglong Luo, Chuanming Chen, and Xiaohan Wang. 2017. Trajectory outlier detection approach based on common slices sub-sequence. *Applied Intelligence* 48, 9 (2017), 1–20.

[78] Rose Yu, Xinran He, and Yan Liu. 2015. Glad: Group anomaly detection in social media analysis. *ACM Transactions on Knowledge Discovery from Data* 10, 2 (2015), 18.

[79] Yanwei Yu, Lei Cao, Elke A. Rundensteiner, and Qin Wang. 2014. Detecting moving object outliers in massive-scale trajectory streams. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 422–431.

[80] Daqing Zhang, Nan Li, Zhi-Hua Zhou, Chao Chen, Lin Sun, and Shijian Li. 2011. iBAT: Detecting anomalous taxi trajectories from GPS traces. In *Proceedings of the 13th International Conference on Ubiquitous Computing*. 99–108.

[81] Jianting Zhang. 2012. Smarter outlier detection and deeper understanding of large-scale taxi trip records: A case study of NYC. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*. ACM, 157–162.

[82] Xujun Zhao, Jifu Zhang, Xiao Qin, Jianghui Cai, and Yang Ma. 2019. Parallel mining of contextual outlier using sparse subspace. *Expert Systems with Applications* 126 (2019), 158–170.

[83] Zheng Zhao and Huan Liu. 2007. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 1151–1157.

[84] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology* 6, 3 (2015), 29.

[85] Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. 2004. Feature selection for text categorization on imbalanced data. *ACM SIGKDD Explorations Newsletter* 6, 1 (2004), 80–89.

[86] Xibo Zhou, Ye Ding, Fengchao PEng, Qiong Luo, and Lionel M. Ni. 2017. Detecting unmetered taxi rides from trajectory data. In *Proceedings of the IEEE International Conference on Big Data*. 530–535.

[87] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. 2018. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems* 20, 1 (2018), 383–398.

[88] Zhihua Zhu, Di Yao, Jianhui Huang, Hanqiang Li, and Jingping Bi. 2018. Sub-trajectory-and trajectory-neighbor-based outlier detection over trajectory streams. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 551–563.

[89] An Liu Guanfeng Liu Zhao Lei Zhu Jie, Jiang Wei. 2015. Time-dependent popular routes based trajectory outlier detection. In *Proceedings of the International Conference on Web Information Systems Engineering*. Springer, 16–30.

[90] Liu An Liu Guanfeng Zhao Lei Zhu Jie, Jiang Wei. 2017. Effective and efficient trajectory outlier detection based on time-dependent popular route. *World Wide Web* 20, 1 (2017), 111–134.

[91] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. Vol. 8. 1433–1438.

[92] Arthur Zimek, Matthew Gaudet, Ricardo J. G. B. Campello, and Jörg Sander. 2013. Subsampling for efficient and effective unsupervised outlier detection ensembles. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 428–436.