IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 7, 2020, accepted February 15, 2020, date of publication February 28, 2020, date of current version March 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977114

KNN-Based Approximate Outlier Detection Algorithm Over IoT Streaming Data

RUI ZHU[1], XIAOLING JI[1], DANYANG YU[2], ZHIYUAN TAN[3], LIANG ZHAO[1], JIAJIA LI[1], AND XIUFENG XIA[1]

[1]College of Computer Science, Shenyang Aerospace University, Shenyang 110036, China
[2]Artificial Intelligence and Big Data College, He University, Shenyang 110000, China
[3]School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, U.K.

Corresponding author: Liang Zhao (lzhao@sau.edu.cn)

This work was supported in part by the National Natural Science Foundation for Young Scientists of China under Grant 61702344 and Grant 61701322, in part by the Young and Middle-Aged Science and Technology Innovation Talent Support Plan of Shenyang under Grant RC190026, in part by the Natural Science Foundation of Liaoning Province under Grant 2019-MS-253, and in part by the Liaoning Provincial Department of Education Science Foundation under Grant JYT19052.

ABSTRACT KNN-Based outlier detection over IoT streaming data is a fundamental problem, which has many applications. However, due to its computational complexity, existing efforts cannot efficiently work in the IoT streaming data. In this paper, we propose a novel framework named GAAOD(Grid-based Approximate Average Outlier Detection) to support KNN-Based outlier detection over IoT streaming data. Firstly, GAAOD introduces a grid-based index to manage summary information of streaming data. It can self-adaptively adjust the resolution of cells, and achieve the goal of efficiently filtering objects that almost cannot become outliers. Secondly, GAAOD uses a *min-heap*-based algorithm to compute the distance upper-/lower-bound between objects and their k-th nearest neighbors respectively. Thirdly, GAAOD utilizes a k-skyband based algorithm to maintain outliers and candidate outliers. Theoretical analysis and experimental results verify the efficiency and accuracy of GAAOD.

INDEX TERMS IoT streaming data, KNN-based outliers, indexes, error guarantee.

I. INTRODUCTION

With the development of information science and technology [1]–[8], outlier detection [9] becomes more and more important. In this paper, we study the problem of KNN-Based outlier detection over IoT streaming data. It is a fundamental problem in the domain of data mining, which has been paid more and more attention as many IoT applications need to detect anomalies as soon as they occur. The applications range from forest fire protection system, mobile traffic monitoring to health-care monitoring and geological disaster prediction [10]. All these applications would benefit from identifying those critical events in real-time [11]–[17].

For example, as an important application in the domain of IoT [32]–[37], a forest fire protection system uses multitudes of sensors to monitor the temperature, humidity and air pressure in a given region. These sensors timely send collected data to the system. When a sensor returns an object which is different from others, it means the region that the sensor located may have the risk of fire. Therefore, outlier detection is a key component of forest fire protection system. In addition, collected data are continuously send to the system, we regard this type of data as IoT streaming data. Therefore, in this paper, we study the problem of outlier detection over IoT streaming data(short for streaming data).

Multitudes of methods could be used for evaluating whether an object in a data set $\mathcal{D}$ is an outlier. Among all of them, distance-based outlier detection is the most popular one [18]. The key behind it is if an object is far from most objects in the data set, this object is regarded as an outlier. Distance-based outlier detection could be further divided into two types: *threshold-based* and *neighbour-based*. For the first one, given two parameters k and r, an object o is an outlier if the number of points within a distance r from o is smaller than k. For the second one, given another two parameters k and n, outliers are the top-n objects that have the largest distances to their k-th nearest neighbors in the dataset.

Due to the importance of outlier detection over streaming data, many scholars have studied this problem, but

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

most of them focus on the threshold-based outlier detection. Yamanishi and Takeuchi [19] propose the framework Smart Sifter. It uses the *online discount learning algorithm* to incrementally learn a probability model. In addition, it uses this model for outlier detection. Zhang *et al.* [20] propose a framework named SPOT(short for Stream Projected Outlier DeTector) for outlier detection in high-dimensional data streams. Cao *et al.* [21] propose the Thresh-LEAP algorithm. Kontaki *et al.* [22] propose a micro-clustering based algorithm named MCOD.

Their key idea is maintaining the last k arrived neighbours for each object, using maintenance results to evaluate which objects have chance to become outliers or cannot become outliers before they expire from the window. For objects that cannot become outliers, these algorithms need not to maintain any neighbour information for them. In other words, these algorithms only need to maintain the last k arrived neighbours for a small number of objects in the window.

However, these efforts only can support threshold-based outlier detection. A serious issue of threshold-based outlier detection is that it is difficult to select a proper distance threshold, which requires users to have sufficient background knowledge. By contrast, *neighbour-based* outlier detection uses distance relationships among objects for evaluating which objects should be regarded as outliers. This kind of definition does not require users to have sufficient background knowledge [23]. However, its computational complexity is high, and can not efficiently work under IoT streaming data environments.

To solve this problem, this paper studies the problem of approximate neighbour-based outlier detection over sliding window. Without loss of generality, the query window can be either time- or count-based. In both cases, the query window has a fixed window size or a fixed slide (either a time interval or an object count). Formally, in a count-based window, it returns the n outliers in the query window containing N objects whenever the window slides; in a time-based window, it returns the n outliers in the last N time units whenever the window slides [11]. Given an approximate outlier detection over sliding window $\langle N, s, n, k, \rho\rangle$, it monitors the window, and returns approximate outliers to the system whenever the window slides. Let $\{o_1, o_2, \ldots, o_n\}$ be the exact outliers, $\{a_1, a_2, \ldots, a_n\}$ be the approximate outliers. We should guarantee that $\frac{D(o_i)}{D(a_i)} \leq \rho$. Here, $D(o_i)$ refers to the distance between o_i and its k-th nearest neighbour. We will discuss the problem definition in details in Section II.

In order to efficiently support approximate neighbour-based outlier detection, this paper proposes a novel framework named GAAOD(short for Grid-based Approximate Average Outlier Detection). GAAOD first learns the distance distribution among objects and their k-th nearest neighbours. Based on the learning result, GAAOD constructs a grid-based index $\mathcal{I}$ to maintain streaming data in the window. We want to highlight that a benefit of GAAOD is that it could self-adaptively adjust the resolution of grid file according to the distance relationship among objects, leading that it can efficiently support approximate neighbour-based outlier detection. Secondly, we propose a novel outlier candidate algorithm. It uses the key of k-skybands algorithm to evaluate which objects may become outliers in a relatively high probability, and dynamic adjust candidate set accordingly. The main contributions of this paper are summarized as follows:

- *A Self-Adaptive Index:* We propose a grid-based index to maintain streaming data. We find that the cost of maintaining each object in the index is $\mathcal{O}(1)$ per object. In addition, we propose a novel algorithm to self-adaptively adjust the cell size of the grid file according to the distribution of outliers. In this way, we could filter non-outliers as fast as possible when they arrive in the window.
- *The Approximate K Nearest Neighbour Algorithm MAKN:* Based on our proposed index, we propose a min-heap based algorithm to compute the approximate distance upper-bound/lower-bound between objects and their k-th nearest neighbours. Compared with the exact algorithms, the cost overall could be reduced a lot.
- *The Candidate Maintenance Algorithm:* We propose a k-skyband based method to maintain objects which may become outliers. The benefit is, when some objects expire from the window, we could efficiently know which objects may become outliers. In addition, it could efficiently remove candidates which almost cannot become outliers.

Section II reviews the related work. Section III introduces the definition of KNN-based approximate outlier detection over sliding window. Section IV describes the framework GAAOD. Section V gives the experimental results and analysis. Section VI concludes this paper.

TABLE 1. The summary of notations.

Notation	definition
W	the sliding window
o_{in}	newly arrived object
o_{exp}	expired object
$SKYK(o)$	the k-skyband neighbour of o
$NC(o)$	the neighbour cells of the cell c
$o_i^s.s$	the dominate number of o_i
N	the size of the window
n	the number of outliers that should be reported
s	the number of neighbours that an object should monitor

II. RELATED WORK

This section reviews the algorithms about the problem of continuous outlier detection over streaming data. With the development of data management [24]–[28], this problem has been extensively studied.

Yamanishi et al.propose the algorithm named Smart-Sifter. It uses an on-line discount learning algorithm to incrementally learn the probability hybrid model [19]. Considering the factor of drift, outliers are calculated according to the probability fitting value of learning hybrid model. In addition, online discounts can also be used to mix attribute data streams to maintain the frequency of a set of attribute values [29].

Angiulli and Fassetti [30] propose an algorithm named STORM. It is used to detect outliers accurately. The algorithm maintains objects in the window via an index structure. In addition, when an object o arrives in the window, STORM firstly inserts o into the index. Secondly, it searches the k last arrived neighbours of o. Thirdly, it deletes expired objects. Last of all, STORM checks which objects are outliers via monitoring objects' neighbours.

Yang *et al.* [31] suggest that maintaining neighborhood relationships among objects may spend relatively high cost. Therefore, they use an important feature of sliding windows, namely the "predictability" of the expiration objects to evaluate which objects have chance(or have no chance) to become outliers. In particularly, given objects in the current window, they predict the pattern structure in subsequent windows by only considering objects in these windows (the objects in the current window). These predicted pattern structures can be abstracted as "predictive view" of each future window, and the outliers can be calculated by using predictive view.

Zhang *et al.* [20] propose the framework SPOT(short for Stream Outlier DeTector) for outlier detection in high-dimensional data streams. The key behind it is capturing the statistical information such as relative density and reversely relative standard deviation of objects to support outlier detection in the high-dimensional streaming data environment.

Kontaki *et al.* [22] propose a micro-clustering based algorithm named MCOD. The key behind it is that if there exist k objects contained in a circle with radius $\frac{r}{2}$, the corresponding circle could be regarded as a micro-clustering. One important property is if the number of objects in a micro-clustering is more than k, these objects all cannot become outliers. Compared with other algorithms, MCOD could effectively reduce the cost of range query.

Cao *et al.* [21] propose the algorithm LEAP in 2014. It contains two optimization principles, that are, minimum detection and life-first. For the first one, it only searches for objects' neighbors when necessary. In this way, it could effectively minimizes the number of neighbors that should be accessed. In addition, for each object o, once the number of neighbors that should be monitored reduces to 0, the algorithm need not to monitor o any longer. For the second one, L. Cao et al. find that objects arrive later are more important than the ones arrive earlier. Through the above two principles, the LEAP algorithm can effectively reduce both CPU cost and space cost.

Discussion: We find that most of these algorithms can only be used for supporting threshold-based outlier detection. However, they cannot be used for supporting neighbour-based outlier detection. The reason behind it is that, under neighbour-based outlier detection, objects' score are timely changed when objects arrive in the window/expire from the window. All objects have chance to become outliers before they expire from the window.

III. PROBLEM DEFINITION

Definition 1 (Distance): Let o_1 and o_2 be two objects in d-dimensional space. They are expressed as the tuple $\langle o_1[1], o_1[2], \ldots, o_1[d]\rangle$ and $\langle o_2[1], o_2[2], \ldots, o_2[d]\rangle$ respectively. The distance between o_1 and o_2, denoted by $\mathsf{D}(o_1, o_2)$, equals to $\sqrt{\sum(o_1[i] - o_2[i])^2}$.

According to different evaluation methods, the following two definitions can be used to evaluate whether a given object is an outlier, which are threshold-based and neighbour-based. For simplicity, let o be an object in the data set $\mathcal{D}$, o^k be the k-th nearest neighbour of o. We use $\mathsf{D}(o, o^k)$ as the score of o, i.e., denoted by $F(o)$.

Definition 2 (Threshold-Based Outliers): Let r and k be two parameters, $\mathcal{D}$ be a set of $d-$dimensional objects. We call an object $o \in \mathcal{D}$ as an outlier if the number of points within a distance r from o is smaller than k.

Definition 3 (Neighbour-Based Outliers): Let n and k be two parameters, $\mathcal{D}$ be a set of $d-$dimensional objects. Outliers are the top-n objects that have the highest scores.

Definition 4 (Continuous Neighbour-Based Outlier Detection): Let W be the query window, $q\langle n, k\rangle$ be the neighbour-based outlier detection, q monitors the window, when the window slides, q returns n objects with the highest scores.

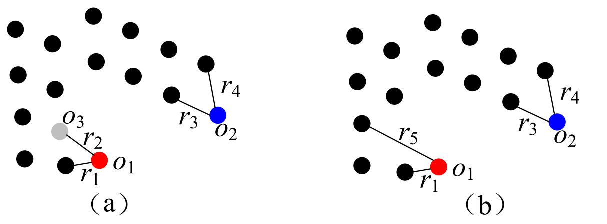

FIGURE 1. **Neighbour-based outliers detection over sliding window($k = 2, n = 1$). (a) Under Window W_i. (b) Under Window W_{i+1}.**

Take an example in Figure 1. Under the window W_i, the distance between o_1 and its 1-nearest neighbour (and 2-nearest neighbour) is r_1(and r_2). The distance between o_2 and its 1-nearest neighbour (and 2-nearest neighbour) is r_3(and r_4). Since $k = 2$, $n = 1$ and $r_4 \geq r_2$, o_2 is regarded as a neighbour-based outlier under W_i. By contrast, when the window slides from W_i to W_{i+1}, o_3 expires from the window. At that moment, the distance between o_1 and its 2-nearest neighbour turns to r_5. At that moment, o_1 is regarded as the neighbour-based outlier under W_{i+1}.

Compared with neighbour-based outlier detection, distance-based outliers detection uses the distance threshold to evaluate whether an object is an outlier, but it is difficult to find a suitable threshold r. Neighbour-based outliers detection uses distance relationships among objects to evaluate whether an object is an outlier. However, its computational complexity is high, and can not efficiently work under streaming data environments. Therefore, we attempt to propose an approximate algorithm to solve the neighbour-based outlier detection. It is rather surprising that such a small sacrifice of accuracy brings tremendous gain in running time.

For simplicity, $\{r_1, r_2, \ldots, r_n\}$ refers to the n exact neighbour-based outliers, and $\{a_1, a_2, \ldots, a_n\}$ refers to the n approximate neighbour-based outliers. $\mathsf{D}(o, o^k)$ refers to the distance between o and its $k-$th nearest neighbour.

Definition 5 (Continuous Approximate Neighbour-Based Outlier Detection): Let W be the query window, $q\langle n, k, \rho\rangle$ be the approximate outlier detection, q monitors the window, when the window slides, q returns n approximate outliers. For each r_i and a_i, they should satisfy that $\frac{\mathsf{D}(r_i, r_i^k)}{\mathsf{D}(a_i, a_i^k)} \le \rho$.

For example, let $O\{o_1, o_2, \ldots, o_n\}$ be a set of objects in the window, $n = 2$, and $k = 2$. Among all of them, the exact results are $\{o_i, o_j\}$, their corresponding scores are $\{20, 18\}$. If we set ρ to 2, we could use $\{o_i', o_j'\}$ as the approximate query results. Here, their corresponding scores are $\{15, 12\}$.

IV. THE FRAMEWORK GAAOD

In this section, the framework GAAOD is proposed to support approximate neighbour-based outlier detection over sliding window. Section IV-A is the framework overview. In Section IV-B, we propose a grid-based index to maintain streaming data. Section IV-C discusses the approximate k nearest neighbour searching algorithm. Last of this section, we discuss the candidate outliers maintenance.

A. THE FRAMEWORK OVERVIEW

When the window slides, a newly arrived object o_{in} flows into the window, another object o_{out} expires from the window. At that moment, we first insert o_{in} into the streaming data set $\mathcal{S}$, and remove the expired object o_{out} from $\mathcal{S}$. Next, we insert o_{in} into the grid-based index named GBOI(short for grid-based outliers index).

Here, GBOI maintains the position information of objects. Specifically, we compute the number of objects in each cell. In addition, we use an inverted-list to maintain objects in each cell based on their arrived order. In order to find a suitable partition resolution for GBOI, the scores of historical outliers are considered. The benefit is once we know the distribution of outliers' scores, we can associate cells' size with a proper size. In this way, we can efficiently evaluate which objects may become outliers via GBOI. More important, GBOI can self-adaptively adjust grid file resolution according to the outliers' scores distribution. The index maintenance algorithm will be discussed in Section IV-B.

When processing a newly arrived object o_{in}, we first find the cell c that contains o_{in}. Next, we find all the neighbour cells of c. Here, we call a cell c' as the neighbour cell of c if the minimal distance between these two cells is 0. Thirdly, we access these neighbour cells for finding approximate k-th nearest neighbours of o_{in}. In particularly, if the searching results satisfy the following two conditions, we do not regard it as a candidate outlier. Otherwise, we regard it as a candidate outlier, and insert it into the candidate set C. Here, l refers to the cell size. The approximate algorithm details will be discussed in Section IV-C.

Algorithm 1 The Framework Overview

```
Input: Index GBOI $\mathcal{G}$, newly arrived object $o_{in}$,expired
       object $o_{out}$
Output: Candidate Set $C$
1  Streaming data set $\mathcal{S} \leftarrow \mathcal{S} \cup o_{in}$,$\mathcal{S} \leftarrow \mathcal{S} - o_{out}$;
2  insert($\mathcal{G}, o_{in}$), delete($\mathcal{G}, o_{out}$);
3  K($o_{in}$) ← searchRange($o_{in}$);
4  if $|K(o_{in})| \ge k$ then
5      Candidate $C \leftarrow C \cup o_{in}$;
6      SKYK($o_{in}$)← searchSKY($o_{in}$);
7  if $K_t(o_{in}) \ge T_n - \frac{N}{2}$ then
8      Candidate $C \leftarrow C \cup o_{in}$;
9      SKYK($o_{in}$) ← searchSKY($o_{in}$);
10 O← findImpact($o_{in}$);
11 for $i$ from 1 to $|O|$ do
12     $o_i.SKY$ ← update();
13     if $o_i.SKY = \emptyset$ then
14         $C \leftarrow C - o_i$;
15 if $|C| = \theta_{max}$ then
16     $C$ ←remove($\theta_{max} - \theta_{min}$);
17     $\tau$ ← reset();
18 return;
```

- there are at least k objects whose distance to o_{in} are smaller than $l\sqrt{d}$;
- let $o.t$ be the arrival order of o. For each object o in the query result set, $o_{in}.t - o.t \le \frac{N}{2}$ is satisfied;

Candidates in C are maintained by an index named M-Tree. In addition, we associate each object in C with an inverted-list, denoted by SKYK(o). It maintains the k-skyband neighbours of o. Given two object o_1 and o_2, if $\mathsf{D}(o, o_1) \le \mathsf{D}(o, o_2)$, and o_1 arrives later than o_2, we say o_1 dominates o_2 under o. If o_1 cannot be dominated by k objects, we regard o_1 as a $k-$skyband neighbour of o. The candidate maintenance algorithm will be discussed in Section IV-D.

In particularly, after insertion, if the number of objects contained in the candidate set is larger than a parameter m_{max}, we re-construct the index GBOI and the candidate set respectively. When an object o_{exp} expires from the window, we should check which objects may become outliers via accessing the candidate set C.

B. THE GRID-BASED INDEX GBOI

This section discusses how to manage streaming data in the window. Compared with the traditional grid file, the key of GBOI is to find a suitable cell size, which helps us efficiently evaluate whether an object may become an outlier. Intuitively, if the cell size is small, we have to access many cells so as to find enough neighbours. By contrast, if the cell size is large, the error may turn to large, leading that the accuracy of our proposed algorithm will be reduced. Therefore, it is very important to select a proper cell size.

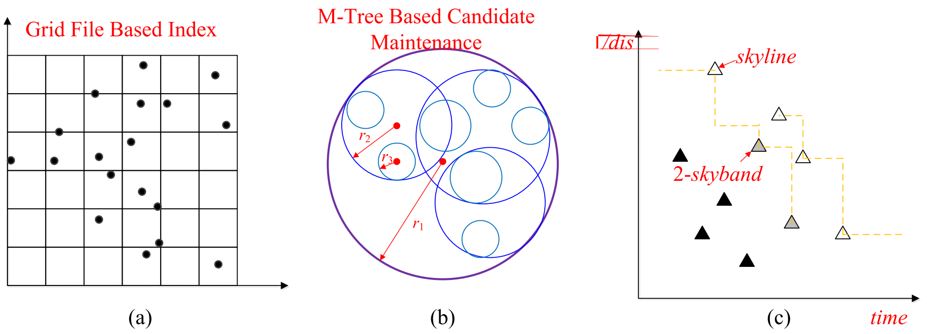

FIGURE 2. The framework overview. (a) The Grid-based index. (b) The M-Tree structure. (c) The 2-skybands.

Specifically, the algorithm uses the scores of historical outliers to find a suitable cell size. Let $\mathcal{H}$ be the historical outlier set, $\mathsf{U}(\mathcal{H})$ refers to the $\frac{|\mathcal{H}|}{20}$-th highest score among all outliers in $\mathcal{H}$. According to Theorem 6, for each cell $c \in I$, if we set the size of c to $\mathsf{U}(\mathcal{H})$, we can efficiently evaluate whether an object could become an outlier in most cases. For simplicity, let o be an object, $F(o)$ refers to its score, $F^n(S)$ refers to the n-th highest score among all objects in S.

Theorem 6: Let $\mathcal{H}$ be the historical outlier set, o be an object contained in the cell c, and $\mathsf{N}(c)$ be the neighbour cells of c. If we set the cell size to $\mathsf{U}(\mathcal{H})$, and there exists k objects in $\mathsf{N}(c) \cup c$ whose distance to o is smaller than $\mathsf{U}(\mathcal{H})$, it cannot become an outlier with probability at least 0.95.

Proof: According to 3-sigma-rule, if data scale is large enough, data distribution obeys normal distribution. In other words, we could use normal distribution $N(mp, \sqrt{mp(1-p)})$ to approximate $\Pr(F(o) \geq F^n(D))$ according to *demovire-laplace theorem*. Here, m equals to $|\mathcal{H}|$, p equals to 0.05. Since $\Pr(F(o) \geq F^n(D)) \approx \Phi(\frac{n-mp}{\sqrt{mp(1-p)}})$. If $\Phi(\frac{n-mp}{\sqrt{mp(1-p)}}) \geq 0.95$, we could obtain a suitable threshold. Thus, if there exists k objects in $\mathsf{N}(c) \cup c$ whose distance to o is smaller than l, it cannot become an outlier with probability at least 0.95.

Note, the distribution of streaming data may be timely changed, we should adjust the cell size, and re-construct GBOI in some cases. Specially, after n objects flow into the window and another n objects expire from the window, we compute how many outliers having scores higher than $\mathsf{U}(\mathcal{H})$. If the corresponding count is larger than $0.1 \times |\mathcal{H}|$, it means the distribution of streaming data is changed. At that moment, we should update GBOI accordingly.

C. THE APPROXIMATE K-NEAREST NEIGHBOR SEARCHING ALGORITHM

This section proposes the algorithm MAKS(short for Max-heap based Approximate k-nearest Neighbor Searching) to compute the score of an object, i.e., find the approximately k-th nearest neighbour for each object.

Let o_{in} be a newly arrived objects. We first find the cell c that contains o_{in}. Next, we find the neighbour cells of c, i.e., $\mathsf{N}(c)$. Thirdly, we use objects in c and $\mathsf{N}(c)$ for evaluating whether o_{in} may become an outlier. For simplicity, let $\{c_1, c_2, \ldots, c_m\}$ be a set of cells contained in the max-heap H. Q_i be a queue that maintains objects in the cell c_i. $Q_i[0]$ refers to the element located at the head of Q_i. In this paper, we use $\{Q_1[0], Q_2[0], \ldots, Q_m[0]\}$ as keys for constructing the max-heap H.

In order to achieve this goal, we construct a max-heap H. Elements in H are objects contained in c and the neighbour cells of c. After constructing, we are going to find the approximate k-nearest neighbor for o_{in}, i.e., compute the score of o_{in}. Specially, let c_i be the cell located at the top of H. We first access $Q_i[0]$, and then insert it into a set S. Next, we update the key of c_i to $Q_i[1]$, and update H accordingly. In particularly, if the size of S achieves to k, the searching is terminated. In addition, if all elements in Q_i are all inserted into S, we remove Q_i from H. Finally, we compute the score of o_{in} according to objects in S. Here, the score of o_{in} equals to the distance between o_{in} and its $k-$ nearest neighbour among objects in the window. Lastly, we evaluate whether o_{in} should be regarded as an outlier or a candidate outlier via the rule discussed in Section IV-A.

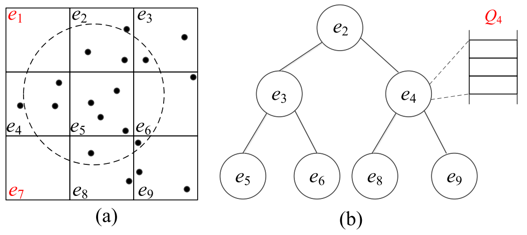

FIGURE 3. Approximate $K-$ nearest neighbour search. (a) The searching region. (b) The heap-based searching.

Take an example in Figure 3. Let $\mathcal{D}$ be a set of 2-dimensional data, o be a newly arrived object. After finding the cell that contains o, we construct a max-heap with size 9. Among all objects contained in these cells, since o_5 arrives latest of all, it is located at the top of H. In addition, since there is no object contained in the cell e_2 and e_6, we only maintain 7 queues in the max-heap H. After constructing H, we first insert o_5 into the set S. Next, we update H. Since n equals to 5, when $|S|$ achieves to 5, the searching algorithm is terminated.

Algorithm 2 The Searching Algorithm

```
   Input: Index GBOI $\mathcal{G}$, newly arrived object $o_{in}$
   Output: Candidate Set $C$
1  Cell $c \leftarrow$ search($o_{in}$, $\mathcal{G}$);
2  Cell Set $CS \leftarrow$ neighbourCell($c$, $\mathcal{G}$) ;
3  Max-heap $H \leftarrow$ construction($CS$);
4  while $H \neq \emptyset \vee |S| \leq k$ do
5      $S \leftarrow S \cup$ Top($H$);
6      if Top($H$)$= \emptyset$ then
7          $H \leftarrow H -$Top($H$);
8      update($H$);
9  if $|S| \leq k \vee$ minT($S$)$\leq T_N - \frac{N}{2}$ then
10     $C \leftarrow C \cup o_{in}$;
11 return;
```

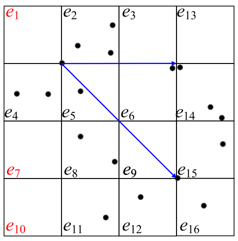

FIGURE 4. The error analysis.

The Error Analysis: We now discuss the error analysis. Let o be an object, c be the cell that contains c. As is shown in Figure 4, we only consider the objects in c and the neighbour cells of c. Therefore, the maximal distance between o and objects in $c \cup \mathsf{N}(c)$ is bounded by $2\sqrt{d}r$. For each object out of $c \cup \mathsf{N}(c)$, the minimal distance between o and o' is bounded by r. Therefore, the error ratio is smaller than $\frac{2\sqrt{d}r}{r}$, that is $2\sqrt{d}$.

D. THE CANDIDATE MAINTENANCE ALGORITHM

In this section, we discuss how to maintain candidate outliers in the window. Before discussing the algorithm, we first propose the concept of dominance.

Definition 7 (Dominance): Let W be the sliding window W, o, o_1 and o_2 be three objects contained in W. If o_1 arrives later than that of o_2, and $\mathsf{D}(o, o_1) \leq \mathsf{D}(o, o_2)$, we say o_1 dominates o_2 under o.

In particularly, if o_2 is dominated by another k objects under o, o_2 have no chance to become a k nearest neighbour of o. We call it as a non-k-skyband neighbour of o. In this paper, we call SKYK(o) as the k-skyband neighbour set of o. As shown in Figure 5, assuming that they satisfy that $o_1.t < o_2.t < o_3.t < 0_4.t < o_5.t$, since o_2 arrives later than that of o_1 and $\mathsf{D}(o, o_2) \leq \mathsf{D}(o, o_1)$, we say that o_2 dominates o_1 under o. Similarly, o_1 is also dominated by o_4, it can not become a k nearest neighbour of o. By contrast, o_5 arrives the latest of all. It also has chance to become an outlier of o. Therefore, SKYK(o) is $\{o_2, o_3, o_4, o_5\}$.

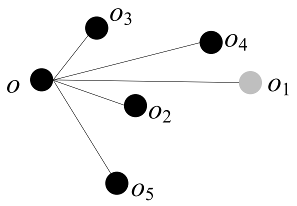

FIGURE 5. Continuous $K-$ nearest neighbour query over sliding window.

Algorithm 3 The SKYK Maintenance Algorithm

```
   Input: Index SKYM $\mathcal{M}$, newly arrived object $o_{in}$
   Output: Candidate Set $C$
1  $Q \leftarrow$ search($\mathcal{M}$,$o_{in}$,$\theta$);
2  for $i$ from 1 to $|Q|$ do
3      while D($Q_i$, $Q_i.o_j$)$<$ D($Q[i].o_j$, $o_{in}$) do
4          $Q_i.o_j.s \leftarrow Q_i.o_j.s + 1$;
5          if $Q_i.o_j.s = k$ then
6              $Q_i \leftarrow Q_i - Q_i.o_j$;
7          if $F(Q_i) \geq \theta$ then
8              $C \leftarrow C - Q_i$;
9  if $F(o_{in}, k) \geq \theta$ then
10     construct($o_{in}$)
11 return;
```

After discussing the concept of dominance and SKYK(o), in the following, we will formally discuss the candidate maintenance algorithm. Let o be a candidate outlier. We first insert o into the candidate set C. Here, objects in C are maintained by an index named M-Tree. Next, we construct SKYK(o) for o.

As is depicted in Algorithm 3, we first submit a range query to the system for finding objects contained in the query region. Let Q be the searching result set. We sort them according to their arrived order. Then, we scan Q to find $k-$skyband neighbours of o. To be more specifically, let Q_s be the scanned objects set. We use a *max-heap* to maintain the k nearest neighbours of o in Q_s. For each object $o' \in Q - Q_s$, if $F(o') \geq \max(H)$, we delete it directly. Otherwise, we insert it into SKYK(o) and H respectively. Next, we adjust the *max-heap* H. In addition, we compute the dominate number of o'.

Thirdly, we update the $k-$skyband based structure named SKYK for other candidate outliers. As is shown in algorithm 4, for each object $o' \in C$, if it is contained in the query region, we first compute the distance between o and o', i.e., $\mathsf{D}(o, o')$. Next, we access SKYK(o'). For each object $o_i^s \in$ SKYK(o'), if $\mathsf{D}(o, o_i^s) \leq \mathsf{D}(o', o_i^s)$, we set $o_i^s.s$ to $o_i^s.s + 1$. Here, $o_i^s.s$ refers to the dominate number of o_i^s under o'. After accessing SKYK(o'), if it satisfies the following two conditions discussed as above, we remove it from the candidate set.

Discussion: If we cannot find n objects with scores lower than $2\sqrt{d}l$, it means other objects may become query results. In these cases, we should re-construct the GBOI and the candidate set C. For the limitation of space, we skip the

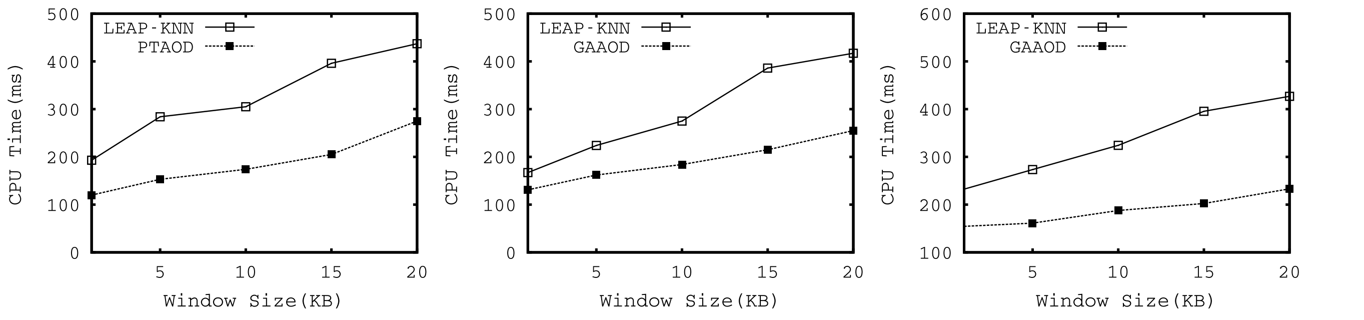

FIGURE 6. **Running time comparison under different window size.**

Algorithm 4 The SKY Construction Algorithm

```
Input: Object $o$, newly arrived object $o_{in}$
Output: SKYK SKYK($o_{in}$)
1  $Q \leftarrow$ search($\mathcal{I}$,$o_{in}$,$\theta$);
2  sort($Q$);
3  for $i$ from 1 to $|Q|$ do
4      if $|H| < k \vee F(H) < D(Q[i], o_{in})$ then
5          $H \leftarrow H \cup Q[i]$;
6          if $|H| = k + 1$ then
7              $H \leftarrow H \cup max(H)$;
8          while $H[i] < Q[i], o_{in}$ do
9              $Q[i] \leftarrow Q[i] + 1$;
10 return;
```

details. In addition, when we re-construct the index GBOI, since we have to re-map all objects in the window into the new index, the computational cost is high. We want to highlight that, in most applications, the distribution of streaming data is not timely changed, we need not to re-construct GBOI and candidate set in most cases.

V. EXPERIMENTAL EVALUATION

In this section, we conduct extensive experiments to demonstrate the efficiency of GAAOD. In the following, we first explain the settings of our experiments, and then report our findings.

A. EXPERIMENTAL SETTING

1) DATA SET

In this paper, three real data sets are used for evaluating our proposed framework. We call them as TAO, Stock and HPC. Here, Stock contains 104,8575 records. It is a 1-dimensional data set. TAO and HPC contain 57,5648 and 248,692 records respectively. They are a 3-dimensional data set and a 7-dimensional data set. For HPC, we select 3 dimensions for evaluating.

TABLE 2. **Parameter settings.**

Parameter	value
The Window Size N	1KB, 5KB, **10KB**, 15KB, 20KB
The Number of neighbours k	5, 10, **20**, 50, 80, 100
The parameter s	$\frac{N}{5}$, $\frac{N}{10}$, $\frac{N}{20}$, $\frac{N}{50}$, $\frac{N}{100}$
The Number of outliers n	5, 10, **20**, 50, 80, 100

2) PARAMETERS SETTING

In our experiments, we evaluate algorithms differences via the following metrics, which are *response time* and *space cost*. Here, *response time* is the total running time we spend after processing all objects in a data set. *Space cost* refers to the average memory we consume. Besides, we evaluate algorithms differences via the following four parameters. They are the window size N, the number s of new objects that slide into the window whenever the window slides. In addition, we also compare algorithms differences under different k and n. Table 2 with the default values are bolded. Specially, the parameter N ranges from $1KB$ to $20KB$ with default value $10KB$. The parameter s ranges from $10^{-2}N$ to $\frac{N}{5}$ with default value $\frac{N}{10}$. The parameter k ranges from 5 to 100 with default value 20. Similarly, the parameter n ranges from 5 to 100 with default value 20.

3) EXPERIMENT METHOD

When implementation, objects in the each data set are all inserted into the memory. Next, we use two pointers for simulating the sliding window. Since the method is simple, we skip the details for saving space. After processing all objects in the dataset, we compute the average CPU time and memory we consume. All the algorithms are implemented with C++, and all the experiments are conducted on a CPU i7 with 16GB memory, running Microsoft Windows 7.

B. ALGORITHM PERFORMANCE

In this section, we compare our proposed framework GAAOD with that of KNN-LEAP. First of all, we evaluate their performance under different window sizes. The other parameters are default values.

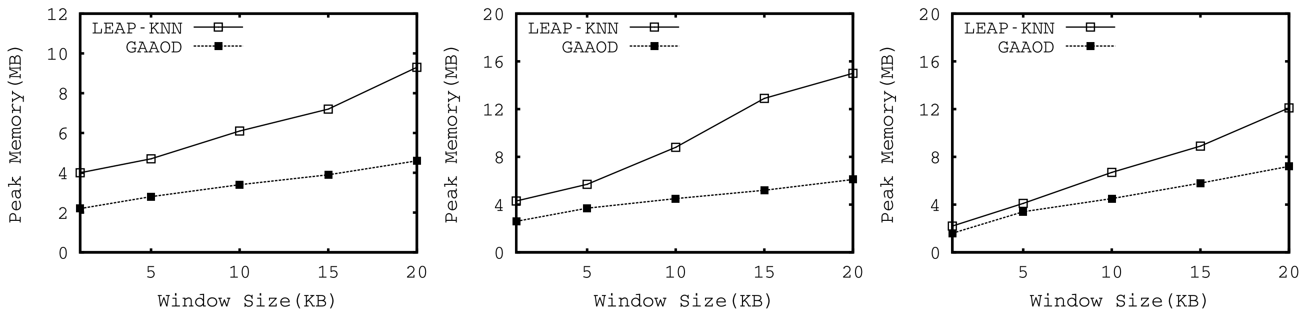

FIGURE 7. Space cost comparison under different window size.

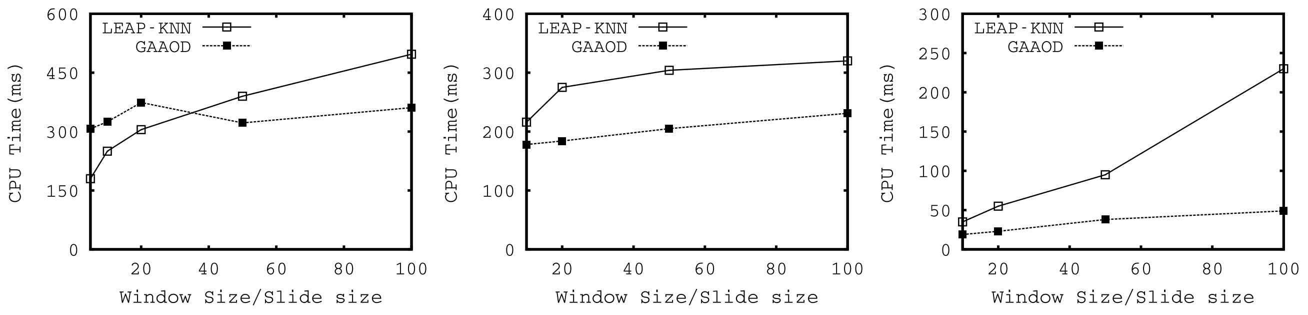

FIGURE 8. Running time comparison under different *s*.

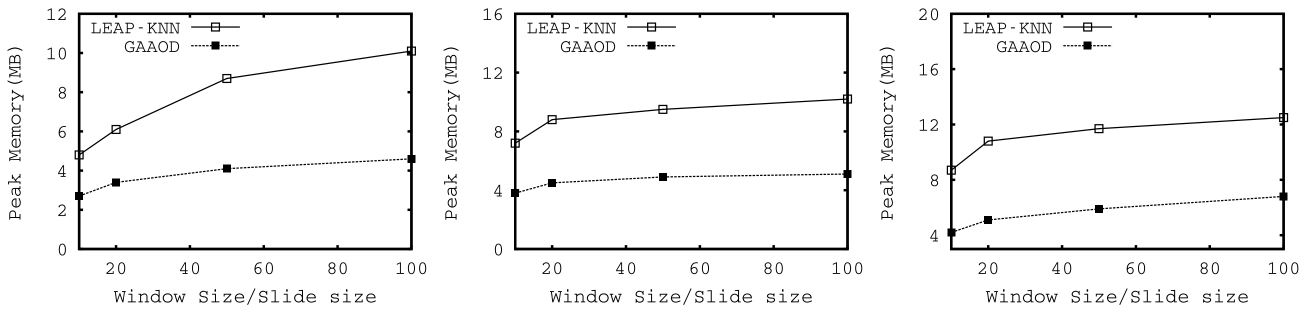

FIGURE 9. Space cost comparison under different *s*.

As is depicted in Figure 6, GAAOD performs the best of all. The reason behind it is, for one thing, we use a grid-based index to manage streaming data in the window. Therefore, we can use $\mathcal{O}(1)$ cost to insert an object into the window, or delete an object from the window. In addition, in most cases, we can use $\mathcal{O}(k\log k)$ cost for finding an object's k-th nearest neighbour. In other words, we can use low cost for evaluating whether an object has relatively high chance to become an outlier. By contrast, KNN-LEAP should spend relatively high cost to compute the distance between objects and their corresponding k-th nearest neighbour.

For the space cost, as is depicted in Figure 7, GAAOD also performs best of all. The reason behind it is we only need to maintain the SKYK structure for a small number of objects. For the others, we only maintain them in the index. Therefore, the space cost of GAAOD is the smallest of all.

Next, we compare GAAOD with other algorithms under different parameter s. The experiment results are shown in Figure 8. We find that GAAOD performs best of all. In addition, our proposed framework is insensitive to the parameter s. To be more specifically, with the increasing of s, the performance of both GAAOD and KNN-LEAP all turn to better. The reason behind is the larger the s, the more objects having the same arrived order. It means we need to construct the SKYK structure for fewer objects. Therefore, the CPU cost they spend all turn to small.

For the space cost, as is depicted in Figure 9, GAAOD also performs better than that of KNN-LEAP. Similarly, with the increasing of the parameter s, the space cost of both KNN-LEAP and GAAOD all turn to small. The main reason behind it is when s turns to large, we only need to maintain SKY structure for a small number of objects.

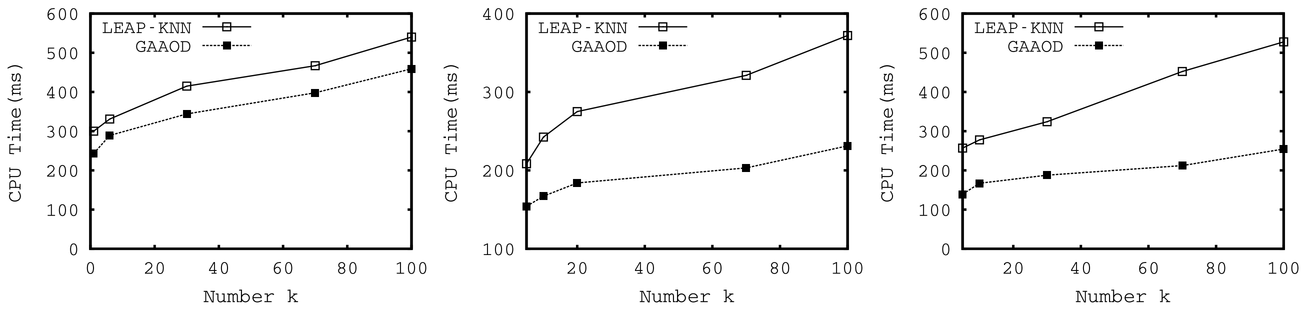

FIGURE 10. Running time comparison under different *k*.

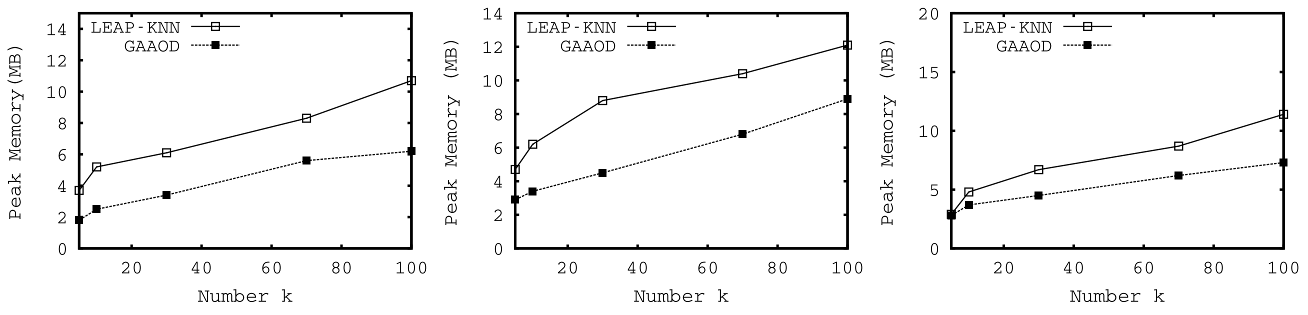

FIGURE 11. Space cost comparison under different *k*.

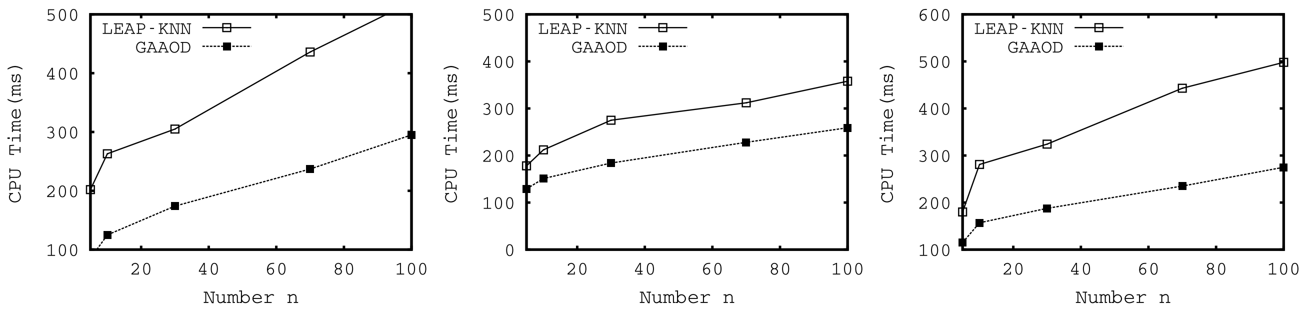

FIGURE 12. Running time comparison under different *n*.

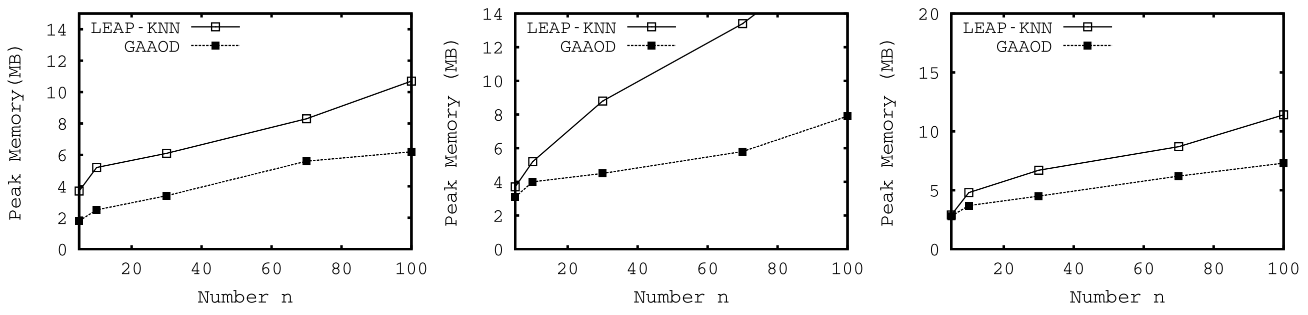

FIGURE 13. Space cost comparison under different *n*.

Thirdly, we compare GAAOD with KNN-LEAP under different k. The experiment results are shown in Figure 10 and Figure 11. We find that the running time of these algorithms all increases with the parameter k. However, the running time of KNN-LEAP goes up more rapidly than that of GAAOD. The reason is GAAOD uses an approximate algorithm to search k nearest neighbours for each object. In this way, we can use lower computational cost for evaluating whether an object has no chance to become an outlier.

For the space cost, GAAOD also consumes smaller cost than that of KNN-LEAP. Besides the reasons discussed as above, another important reason is we use an important property of sliding window to enhance algorithm performance. That is, we can use the arrival order relationship among objects to predict which objects may(or not) become an outlier. In this way, we only need to maintain a small number of candidates.

Next, we compare GAAOD with KNN-LEAP under different n. The experiment results are shown in Figure 12 and Figure 13. We find that GAAOD also performs better than that of KNN-LEAP. The reason is GAAOD only needs to maintain a smaller number of candidates. Another important reason is we use a more efficient method to maintain SKYK structure. In this way, the CPU cost of GAAOD could be reduced a lot.

For the space cost, as is depicted in Figure 13, GAAOD also performs best of all. Similar with the reason discussed as above, we only need to maintain the SKYK structure for a small number of candidates.

VI. CONCLUSION

In this paper, we study the problem of KNN-based approximate outlier detection over IoT Streaming Data. In order to solve this problem, we first propose a grid file based index to manage streaming data in the window. Next, we propose a novel algorithm to answer approximate KNN search. Thirdly, we propose a $k-$skyband based method to maintain candidate outliers in the window. Theoretical analysis and experiment verify the efficiency and accuracy of our proposed algorithms

REFERENCES

[1] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, "Time-aware distributed service recommendation with privacy-preservation," *Inf. Sci.*, vol. 480, pp. 354–364, Apr. 2019.

[2] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web J.*, May 2019.

[3] J. Yu, Z. Kuang, B. Zhang, W. Zhang, D. Lin, and J. Fan, "Leveraging content sensitiveness and user trustworthiness to recommend fine-grained privacy settings for social image sharing," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1317–1332, May 2018.

[4] J. Yu, C. Zhu, J. Zhang, Q. Huang, and D. Tao, "Spatial pyramid-enhanced NetVLAD with weighted triplet loss for place recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 661–674, Feb. 2020.

[5] J. Yu, M. Tan, H. Zhang, D. Tao, and Y. Rui, "Hierarchical deep click feature prediction for fine-grained image recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[6] J. Yu, J. Li, Z. Yu, and Q. Huang, "Multimodal transformer with multi-view visual representation for image captioning," *CoRR*, 2019.

[7] H. Gao, W. Huang, and X. Yang, "Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data," *Intell. Autom. Soft Comput.*, vol. 25, no. 3, pp. 547–559, 2019.

[8] H. Gao, W. Huang, Y. Duan, X. Yang, and Q. Zou, "Research on cost-driven services composition in an uncertain environment," *J. Internet Technol.*, vol. 20, no. 3, pp. 755–769, 2019.

[9] J. Pu, Y. Wang, X. Liu, and X. Zhang, "STLP-OD: Spatial and temporal label propagation for traffic outlier detection," *IEEE Access*, vol. 7, pp. 63036–63044, 2019.

[10] X. Qin, L. Cao, A. E. Rundensteiner, and S. Madden, "Scalable kernel density estimation-based local outlier detection over large data streams," in *Proc. 22nd Int. Conf. Extending Database Technol. (EDBT)*, Lisbon, Portugal, Mar. 2019, pp. 421–432.

[11] R. Zhu, B. Wang, X. Yang, B. Zheng, and G. Wang, "SAP: Improving continuous top-K queries over streaming data," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 6, pp. 1310–1328, Jun. 2017.

[12] C. Zong, X. Xia, B. Wang, X. Yang, J. Li, X. Liu, and A. Zhu, "Answering why-not questions on structural graph clustering," in *Proc. 23rd Int. Conf. Database Syst. Adv. Appl. (DASFAA)*, Gold Coast, QLD, Australia, May 2018, pp. 255–271.

[13] R. Zhu, B. Wang, S. Luo, X. Yang, and G. Wang, "S-MRST: A novel framework for indexing uncertain data," *World Wide Web*, vol. 20, no. 4, pp. 697–727, Sep. 2016.

[14] B. Wang, R. Zhu, X. Yang, and G. Wang, "Top-K representative documents query over geo-textual data stream," *World Wide Web*, vol. 21, no. 2, pp. 537–555, Jul. 2017.

[15] R. Zhu, B. Wang, S.-Y. Luo, X.-C. Yang, and G.-R. Wang, "Approximate continuous top-k query over sliding window," *J. Comput. Sci. Technol.*, vol. 32, no. 1, pp. 93–109, Jan. 2017.

[16] B. Wang, R. Zhu, S. Luo, X. Yang, and G. Wang, "H-MRST: A novel framework for supporting probability degree range query using extreme learning machine," *Cognit. Comput.*, vol. 9, no. 1, pp. 68–80, 2017.

[17] B. Wang, R. Zhu, S. Zhang, Z. Zhao, X. Yang, and G. Wang, "PPVF: A novel framework for supporting path planning over carpooling," *IEEE Access*, vol. 7, pp. 10627–10643, 2019.

[18] S. Yoon, J.-G. Lee, and B. S. Lee, "NETS: Extremely fast outlier detection from a data stream via set-based processing," *Proc. VLDB Endowment*, vol. 12, no. 11, pp. 1303–1315, Jul. 2019.

[19] K. Yamanishi and J.-I. Takeuchi, "A unifying framework for detecting outliers and change points from non-stationary time series data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Edmonton, AB, Canada, Jul. 2002, pp. 676–681.

[20] J. Zhang, Q. Gao, and H. Wang, "SPOT: A system for detecting projected outliers from high-dimensional data streams," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Cancún, Mexico, Apr. 2008, pp. 1628–1631.

[21] L. Cao, D. Yang, Q. Wang, Y. Yu, J. Wang, and E. A. Rundensteiner, "Scalable distance-based outlier detection over high-volume data streams," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Chicago, IL, USA, Mar. 2014, pp. 76–87.

[22] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Continuous monitoring of distance-based outliers over data streams," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Hannover, Germany, Apr. 2011, pp. 135–146.

[23] X.-T. Wang, D.-R. Shen, M. Bai, T.-Z. Nie, Y. Kou, and G. Yu, "An efficient algorithm for distributed outlier detection in large multi-dimensional datasets," *J. Comput. Sci. Technol.*, vol. 30, no. 6, pp. 1233–1248, Nov. 2015.

[24] Y. Cheng, Y. Yuan, L. Chen, C. Giraud-Carrier, and G. Wang, "Complex event-participant planning and its incremental variant," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, San Diego, CA, USA, Apr. 2017, pp. 859–870.

[25] Y. Cheng, Y. Yuan, L. Chen, C. Giraud-Carrier, G. Wang, and B. Li, "Event-participant and incremental planning over event-based social networks," *IEEE Trans. Knowl. Data Eng.*, to be published.

[26] Y. Cheng, Y. Yuan, L. Chen, G. Wang, C. Giraud-Carrier, and Y. Sun, "DistR: A distributed method for the reachability query over large uncertain graphs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 11, pp. 3172–3185, Nov. 2016.

[27] Y. Cheng, L. Chen, Y. Yuan, and G. Wang, "Rule-based graph repairing: Semantic and efficient repairing methods," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Paris, France, Apr. 2018, pp. 773–784.

[28] B. Li, Y. Cheng, Y. Yuan, G. Wang, and L. Chen, "Three-dimensional stable matching problem for spatial crowdsourcing platforms," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Anchorage, AK, USA, Aug. 2019, pp. 1643–1653.

[29] A. Ghoting, M. E. Otey, and S. Parthasarathy, "LOADED: Link-based outlier and anomaly detection in evolving data sets," in *Proc. 4th IEEE Int. Conf. Data Mining (ICDM)*, Brighton, U.K., Nov. 2004, pp. 387–390.

[30] F. Angiulli and F. Fassetti, "Distance-based outlier queries in data streams: The novel task and algorithms," *Data Mining Knowl. Discovery*, vol. 20, no. 2, pp. 290–324, Jan. 2010.

[31] D. Yang, E. A. Rundensteiner, and M. O. Ward, "Neighbor-based pattern detection for windows over streaming data," in *Proc. 12th Int. Conf. Extending Database Technol. Adv. Database Technol. (EDBT)*, Saint Petersburg, Russia, Mar. 2009, pp. 529–540.

[32] L. Zhao, X. Li, B. Gu, Z. Zhou, S. Mumtaz, V. Frascolla, H. Gacanin, M. I. Ashraf, J. Rodriguez, M. Yang, and S. Al-Rubaye, "Vehicular communications: Standardization and open issues," *IEEE Commun. Standards Mag.*, vol. 2, no. 4, pp. 74–80, Dec. 2018.
[33] L. Zhao, A. Al-Dubai, A. Y. Zomaya, G. Min, A. Hawbani, and J. Li, "Routing schemes in software-defined vehicular networks: Design, open issues and challenges," *IEEE Intell. Transp. Syst. Mag.*, to be published.
[34] J. Wu, L. Zou, L. Zhao, A. Y. Al-Dubai, L. Mackenzie, and G. Min, "A multi-UAV clustering strategy for reducing insecure communication range," *Comput. Netw.*, vol. 158, pp. 132–142, Jul. 2019.
[35] R. Chaudhary and N. Kumar, "Detection scheme for software-defined networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12329–12344, Dec. 2019.
[36] W. Miao, G. Min, Y. Wu, H. Huang, Z. Zhao, H. Wang, and C. Luo, "Stochastic performance analysis of network function virtualization in future Internet," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 613–626, Mar. 2019, doi: 10.1109/JSAC.2019.2894304.
[37] A. Hawbani, E. Torbosh, X. Wang, P. Sincak, L. Zhao, and A. Al-Dubai, "Fuzzy based distributed protocol for vehicle to vehicle communication," *IEEE Trans. Fuzzy Syst.*, to be published, doi: 10.1109/TFUZZ.2019.2957254.

RUI ZHU received the M.Sc. degree in computer science from the Department of Computer Science, Northeastern University, China, in 2008, and the Ph.D. degree in computer science from Northeastern University, in 2017. He is currently an Associate Professor with Shenyang Aerospace University. His research interests include design and analysis of algorithms, databases, data quality, and distributed systems.

XIAOLING JI received the master's degree from Shenyang Aerospace University, in January 2019. Her research interests include design and analysis of algorithms, query processing over streaming data, and distribution of spatio-temporal data.

DANYANG YU received the Ph.D. degree in biomedical engineering from Beihang University, in 2015. She is currently a Teacher with the Artificial Intelligence and Big Data College, He University. Her research interests include human physiological parameter detection, intelligent medical equipment, and the IoT applications.

ZHIYUAN TAN received the B.Eng. degree in computer science and technology from Northeastern University, China, the M.Eng. degree in software engineering from the Beijing University of Technology, China, and the Ph.D. degree in computer systems from the University of Technology Sydney, Ultimo, NSW, Australia. He is currently an Associate Editor of IEEE Access and an Organizer of Special Issues for the *Ad Hoc and Sensor Wireless Networks* journal.

LIANG ZHAO received the Ph.D. degree in computing from the School of Computing, Edinburgh Napier University, in 2011. He is currently an Associate Professor with Shenyang Aerospace University, China. Before joining Shenyang Aerospace University, he worked as an Associate Senior Researcher with Hitachi (China) Research and Development Corporation, from 2012 to 2014. His research interests include VANETs, SDN, and WMNs.

JIAJIA LI received the M.S. and Ph.D. degrees in computer science from Northeastern University, in 2010 and 2014, respectively. She is currently an Associate Professor with Shenyang Aerospace University. She undertakes one National Natural Science Foundation of China and one Natural Science Foundation of Liaoning Province of China. Her research interests include spatial-temporal database, uncertain data management, and machine learning.

XIUFENG XIA received the M.Sc. degree in computer science from the Department of Computer Science, Northeastern University, China, in 1991, and the Ph.D. degree in computer science from Northeastern University, in 2005. He is currently a Professor with Shenyang Aerospace University. His research interests include design and analysis of algorithms, databases, data quality, and distributed systems.