IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 28, 2019, accepted October 8, 2019, date of publication October 11, 2019, date of current version November 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946966

An Effective Minimal Probing Approach With Micro-Cluster for Distance-Based Outlier Detection in Data Streams

MOHAMED JAWARD BAH[1], HONGZHI WANG[1], (Member, IEEE), MOHAMED HAMMAD[2], FURKH ZESHAN[3], AND HANAN ALJUAID[4]

[1]Massive Data Computing Laboratory, School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[2]Faculty of Computers and Information, Menoufia University, Menoufia 32511, Egypt
[3]Department of Computer Science, COMSATS University Islamabad at Lahore, Lahore 54000, Pakistan
[4]Computer Sciences Department, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University (PNU), Riyadh 84428, Saudi Arabia

Corresponding author: Mohamed Jaward Bah (easybah@yahoo.com)

This work was supported in part by the NSFC under Grant U1509216, Grant U1866602, and Grant 61602129, in part by the Microsoft Research Asia, and in part by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University through the Fast-Track Research Funding Program.

ABSTRACT Outlier detection in data streams is considered a significant task in data mining that targets the discovery of elements in an unprecedented data arrival rate. The fast arrival of data demands fast computation within the shortest period, and with minimal memory usage. Detecting distance-based outliers in such a scenario are more complicated. Existing techniques such as the two best-known methods - Micro-Cluster Outlier Detection (MCOD) and Thresh_LEAP have presented some solutions to these challenges. However, the combination of the strength of both techniques can be a lot more improvement to the individual methods proposed. Therefore, in this paper, we propose a method called Micro-Cluster with Minimal Probing (MCMP), which is a hybrid approach of the combination of the strength of MCOD and Thresh_LEAP. We offer a new distance-based outlier detection technique to minimize the computational cost in detecting distance-based outliers effectively. The proposed MCMP technique is comprised of two approaches. Firstly, we adopt micro-clusters to mitigate the range query search. Then, to deal with the objects outside the micro-clusters, we propose the concept of differentiating between strong and trivial inliers. The proposed method improves the computational speed and memory consumption, while simultaneously maintaining the outlier detection accuracy. Our experiments are conducted on both real-world and synthetic data sets. We varied the window size (w), neighbor count threshold (k) and distance threshold (R), and observed that our method outperforms the state-of-the-art methods in both CPU time and memory consumption in the majority of the datasets.

INDEX TERMS Outlier detection, data streams, distance-based, micro-cluster.

I. INTRODUCTION

The process of detecting data points that do not conform to expected normal behaviors(outliers), is a progressively significant domain in many fields [1]–[3]. It has attracted more attention, especially in the data mining community. The traditional approach in previous years for detecting outliers was mainly focused on batch processing, where data was readily available [4], [5]. However, it is apparent with the revolution of the digital age, the world has become wholly digitalized with data emanating from different devices and application areas at very high speeds and large volumes [6], [7]. Data gathering techniques have advanced, and the static collection of data is no longer the trend. Most data are now viewed as dynamic, and as fast incoming data streams [8]. The processing of these data streams has become a significant area of study, especially in the area of detecting abnormal behavior or unusual data points.

Several studies [9]–[15] have focused on solving various issues in data streams, such as the issue of high and unbounded data volumes [7], [16], [18], single discovery of distance-based outliers in data streams [10], [19], multiple

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

outlier discoveries [20] and in the detection of outliers in continuous uncontrollable arrival rate of data [21], [22]. However, there is still a need for better techniques to address these issues. Among these issues, detecting distance-based outliers in continuous unbounded data streams is a more challenging and significant study. This is because the data streams are in the form of continuous fast-paced generated instances of data, which makes it difficult to compute, store, and analyze. A couple of application domains generate massive and fast-changing data streams that require continuous monitoring to facilitate the discovery of useful knowledge, like fraud, rare, unusual, or anomalous events [23]. These are critical issues that can potentially cost companies and individuals millions of dollars if not addressed. Failing to detect and compute useful outliers efficiently not only causes slow processing of these data but also adds to cost through dirty data, eventually affecting the accuracy or capacity to discover vital knowledge. This ultimately will result in the poor computation of the data streams and cause poor performance and failures in systems, loss of crucial data, and in poor data analytics [23]. In recent times, many companies and individuals have embraced the fast computation of data streams, while ensuring that the data is clean. The need to design effective algorithms based on mined knowledge for handling incoming data is very crucial for making significant decisions in near real-time. Our proposed method tries to address some of these challenges with improved computational time and memory, which are the most important underlying metrics for outlier detection methods. Moreover, our approach tries to address an additional issue of determining the significance of a particular data point within a specific period. This poses challenges, because a prompt decision on whether to discard, archive or keep data temporarily in memory is needed.

Among the existing techniques, Cao *et al.* [10] and Kontaki *et al.* [21] focused on detecting distance-based outliers in continuous data streams. Our proposed method also with similar objective tries to address the same problem with a different approach, by adopting and enhancing the strengths of these methods rather than addressing their individual shortcomings. Tran *et al.* [11] recommend that the design of a hybrid approach that benefits from Thresh_LEAP minimal probing [10] and MCOD micro-clusters [21] could be further explored. Using the strength of both algorithms could improve performance. Taking this into consideration, in this paper, we experiment with this premise. MCOD, which uses the concept of micro-clusters to reduce range query search, shows improved performance when compared to other algorithms in most cases [10], [11]. We propose a method that effectively computes the inliers outside the micro-clusters to reduce computational cost. Thresh_LEAP, on the other hand, makes use of minimal probing using life span aware prioritization but does not use micro-clusters, which effectively reduces the query search. Therefore, we apply the strength of both algorithms to complement each other. We propose a new solution to solve the problem of distance-based outlier detection by implementing a new technique called Micro-Cluster with Minimal Probing (MCMP), which is inherited from the strength of MCOD [21] and Thresh_LEAP [10]. We address two fundamental problems.

Firstly, it is computationally demanding to accurately compute distance-based outliers within the shortest possible time when data arrives in continuous unbounded and uncontrollable state. Such computation requires first the creation of a window to compute the bound and then the approximation of the distance-based outliers. This is because it is time-consuming to compute or query every data point within the window. To address this issue, we propose a method that applies effective minimal probing on data points outside the micro-clusters (PD) [24] by introducing the concept of strong and trivial outliers. The minimal probing outside the micro-clusters ensures that even though the search technique is maintained, the cost of the search technique is minimized. This is unlike that of MCOD, which is characterized by the absence of minimal probing and only makes use of micro-clusters.

Secondly, in terms of memory usage, consuming large memory results in slow processing, which is an undesirable characteristic of data streams. To avoid this problem, we must tackle the huge memory cost. We do not know beforehand the amount of data coming in, and saving every data point will consequently lead to a queue in memory. When the memory is full, it will eventually lead to vital points expiring and which can no longer be computed. It is apparent that to store all data points is not feasible. Therefore, to address this issue, we stored only samples of crucial inliers. This is also because it is essential, we take into consideration memory constraints. We propose a method similar to MCOD Micro-Cluster technique, but one which also reduces memory cost. Thus, the key difference is that our method uses the strong and trivial inliers for which in terms of memory management, we do not store all the information collected in memory.

Finally, to evaluate the performance of the proposed MCMP, we employ both real and synthetic data sets used in previous experiments and compare them with two baseline algorithms. In summary, the following are the major contributions in this paper:

1) We propose a novel solution for the problem of detecting distance-based outliers in data streams by simultaneously applying the concept of minimal probing and adopting micro-clusters.
2) We propose a different approach that applies the concept of strong and trivial inliers outside the micro-clusters. This approach ensures we approximate the most suitable subset of the data points to minimize the computational demand.
3) Our proposed method optimizes and solves the problems and challenges of time and memory constraints in detecting distance-based outliers in data streams.
4) Our proposed method through an extensive experiment on both real-world and synthetic datasets verifies the effectiveness and accuracy of our approach against some baseline algorithms.

The remaining section of the paper is organized as follows. In Section II, we introduce some related studies, in Section III we present the preliminaries and problem formalization. Section IV describes the methodology, while Section V focuses on the experimental studies, including the results obtained and discussions. We finally conclude our work in Section VI.

II. RELATED WORK

Several studies have been proposed in the area of outlier detection, but few studies relate to the discovery of distance-based outliers in data streams. Knorr and Ng [4] and Knorr *et al.* [5] introduced one of the first tailored methods to tackle distance-based outliers. They define an object as a distance-based outlier if less than k objects in the set of objects are within the distance R of the object [5], where k is the nearest neighbor and R the radius from the center of the object. A major drawback of their method concerns the efficient scaling of large datasets. Bay and Schwabacher [25] later try to address that drawback. They proposed a naïve block nested-loop algorithm for distance-based outliers by using the principle of simple pruning and randomization. However, their method lacks proper investigation of some practical issues that need to be considered in distance-based outlier mining, such as the parameter k, the score function, and distance measures. In subsequent years, a study was proposed by Angiulli and Pizzuti [26] which deals with the prediction of hidden objects, and they proposed a method to find the top n distance-based outliers in an unlabeled dataset. Ghoting *et al.* [27] took the *speed* into consideration and proposed a fast mining algorithm for distance-based outliers primarily targeted for high dimensional datasets. Angiulli and Fassetti [9], introduced a method called DOLPHIN for better computation speed that uses only two sequential scans for detecting distance-based outliers.

In all the above-conducted research, all the techniques were not based on data streams until Angiulli and Fassetti [9] proposed the first method related to outlier detection in data streams. The method adopts the sliding window model [12] to detect distance-based outliers in data streams. They presented two approaches, Exact-Storm and Approximate-Storm, to answer to queries. Yang *et al.* [28] in later years, proposed the following algorithms, Exact-N, Abstract-C, Abstract-M, and Extra-N, which were all solutions for incremental detection of neighbor-based patterns for sliding window scenarios. Among these algorithms, Extra-N shows better overall performance and achieves both linear memory consumption and the least number of range query searches required. Kontaki *et al.* [21], similar to this paper, studied the problem of continuous outlier detection in data streams. They proposed three approaches for continuous outlier monitoring in data streams based on the sliding window. They are- the Continuous Outlier Detection COD, the Advanced COD (ACOD) and the Micro-Cluster based COD (MCOD) algorithms. We adopt a method similar to MCOD, which shows better performance than the other two techniques. Another extension to the study of distance-based outliers is in the case of the large volume of data streams. Cao *et al.* [10] proposed a method to discover distance-based outliers from huge volumes of streaming data. They used the minimal probing technique to enhance the neighbor search to identify outliers. In addition, they applied the lifespan-aware prioritization principle to leverage the temporal relationships among streaming data points. Finally, Tran *et al.* [11] presented an evaluation scheme for the current distance-based approaches for detecting outliers in data streams. Their experimental results depict that MCOD [21] shows the best performance, second to Thresh_LEAP [10]. They suggested a hybrid approach using the two techniques for further research work. Therefore, in our work, we try to strike a balance between the two better methods and design a hybrid method that combines the strength of both algorithms.

III. PRELIMINARIES AND PROBLEM FORMALIZATION

A. PRELIMINARIES AND DEFINITION OF TERMS

In this section, we present the definition of the terms used in our study and then give the limitations of previous studies to formulate our problem definition. We first present a summary of the symbols used with their interpretations in Table 1.

TABLE 1. Symbols with their interpretations.

Symbols	Interpretation
d_i	*i-th* data points, *i=1,..,n*
R	the distance threshold
K	the number of neighbors
W	the size of the window
S	the window slide size
$I(R, k)$	inliers; not greater than R and k
S_i	the strong inliers
T_i	the trivial inliers.
∞	data streams
t_i	the specific time
d_{ci}	data points in the current window.
d_{ei}	expired data points
$d_i.a$	the arrival time of object i
$d_i.e$	the expiration time of object i
d_{si}	the succeeding neighbors of d_i
d_{pi}	The preceding neighbors of d_i
$O_d(R, k)$	The detected outlier/s
PD	Data points outside micro-clusters

Definition 1 (Outliers and Inliers): Given a dataset D with n number of points, $D = \{d_1, d_2, d_3, \ldots, d_n\}$. If the data point d_i or set of data points $d_1, d_2, \ldots, d_n$ totally deviate from other sets of n data points, these points are considered outliers. While inliers I are n data points that belong or does not deviate from other n data points in a given dataset D. They have at least k neighbors in their current window.

If we consider Fig.1, for $k = 4$. Here, d_1 and d_3 are outliers and $d_2$is an inlier. This is because d_1, d_3 has less than $k = 4$, with the values of $K = \{d_1 = 2$ and $d_3 = 1\}$ respectively, while d_2 has four neighbors.

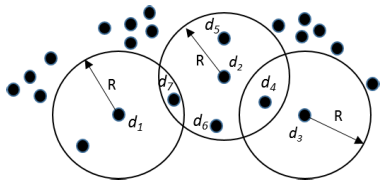


FIGURE 1. Static dataset [21].

Definition 2 (Neighbor): The data point d_i, is a neighbor of another data point d_n, when the distance between d_i to d_n is not greater than the given distance threshold $R>0$. i.e., if d_i does not lie at a distance greater than R from d_n, then d_i is a neighbor of d_n. The data point d cannot be a neighbor of itself. From Fig. 1, d_4, d_5, d_6 and d_7 are neighbors of d_2.

An important concept we take into consideration in terms of the neighbors is the ordering of data points according to their arrival time, $d_i.a$. It is significant to differentiate between the notion of succeeding neighbor, d_{si}, and preceding neighbor d_{pi}. When a data point d_i whose neighbor is d_n expires d_{ei}, before d_n, then d_i is a preceding neighbor of d_n. d_{pi} are neighbors of a particular data point that precede another data point in the data stream of the current window d_{ci}. While when a data point d_i has d_n as its neighbor, which expires in the same slide with or after d_n, then the data point d_i is a succeeding neighbor of data point d_n.

Definition 3 (Strong and Trivial Inliers): Strong inliers S_i are data points within a subset of a micro-cluster with n number of neighbors and have n succeeding neighbors. They are made up of at least k succeeding neighbors. It can be seen from Fig. 2 that $d'_{13}s$ succeeding neighbor is d_{11}, and its preceding neighbors are d_3, d_4, and d_6. On the contrary, other inliers are labeled as trivial inliers T_i, due to the nature of the preceding neighbors in the future. These preceding neighbors will eventually become outliers when these neighbors expire. The T_i is characterized by less than k succeeding neighbors. For instance, if $k = 5$, then d_{11} is a trivial inlier with less than k succeeding neighbor.

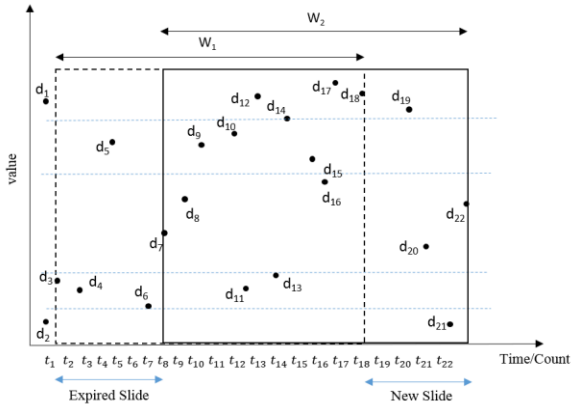


FIGURE 2. Data streams.

Definition 4 (Distance-Based Outliers): In a given dataset D, with count threshold of $k(k >= 0)$ and a distance threshold $R(R > 0)$, a data point d_i is defined as a distance-based outlier if it does not lie more than the distance threshold R and has fewer than k neighbors in the dataset [4].

Definition 5 (Data Stream): A data stream is an unlimited number of data points $\infty_i = \{d_t|0 \leq t\}$, *w*here d_t are data points with a specific timestamp. They are infinite series of data points $\ldots, d_{t-2}, d_{t-1}, d_t$, observed at a particular time t.

Definition 6 (Sliding Window): There are two types of window models for data streams; the time-based window and the count-based window. Given the identifiers of the two data points, t_p and t_q, with $t_p \leq t_q$, the data stream window $[t_p, t_q]$, is the set of $d_q - d_p + 1$ data points $d_{tp}.d_{tp+a}, \ldots, d_{tq}$, where $t_p = t_2$ and $t_q = t_{27}$ (size w_1) in Fig. *2* and $d_q - d_p + 1$ is the data stream window size.

In the count-based window, for a fixed window of size $W_n$and data point d_i, the count-based window W_n is the set of W data points: $D_i{-}W{+}1, D_i - W + 2, \ldots, D_n$ within the specified window. While in the time-based window, given the time period t with a data point D_i, the time-based window W *(i, T)* is the set of W_n data points: D'_i, $D'_i{+}1$, . .., on with $W_n = n - n' + 1$ and $Dn.t{-}Dn'.t = T$.It is the period allocated for a set of data point computation.

Definition 7 (Current Window and Expired Window): In Fig. 2, the current window size is *DS [t_iW+1, t]*, where *t* is the arrival time of the final data point, $d_i.a$. In this case, W_2 from t_7 to t_{22} is considered the current window, while the expired window is from t_1 to t_7.

Definition 8 (Micro-Clusters):

Data points that have a radius of *R/2* from the center of a data point, as shown in C_1 form a micro-cluster. The distance between two data points, d_1, and d_2 within a micro-cluster should not be greater than R.

B. PROBLEM FORMULATION

Our focus in this paper is to propose better solutions to respond to the issues highlighted below.

Issue 1: The problem of effectively detecting distance-based outliers in data streams for every sliding window given the distance threshold R, the count threshold k, the window size W, and slide size S.

The main challenge in solving the above problem is to ensure that the algorithm adapts to changes in a set of current data points d_{ci} within the sliding window. In the streaming environment, in a particular window model, a set of active data points change due to the arrival of new data points $d_i.a$, and some data point's neighbors will expire d_{ei}. This, therefore, creates confusion in the computation of distance-based outliers. For example, Fig. 2 shows how the arrival and expiration of data points within the sliding window influence the data point's outliers. The object, d_{15} as an example, experiences both states, in being an inlier and an outlier. In W_1 between the time interval of t_1 to t_{17}, when $k{=}4$, d_{15} is an inlier with the following neighbors d_5, d_9, d_{10}, and d_{14}. However, in W_2 on the expiration of a single data point d_5,

it changes to an outlier state, since it has less than k neighbors. In such scenarios, it is shown to be computationally tasking to store and recompute the neighbors when the window slides.

Issue 2: The issue of detecting multiple distance-based outliers in data streams for every sliding window given different distance threshold R, multiple count threshold k, and the window size W.

In the first scenario, to detect distance-based outliers over a single data stream, it compares the data point with respect to the history data points. However, in the case of detecting multiple distance-based outliers in data streams, it demands extra computation. Also, adjusting and varying the other parameters is a complicated task which might result in either poor or improved performance of the outlier detection algorithm.

IV. METHODOLOGY

In this section, we introduce our proposed scheme for detecting and querying single and multiple distance-based outliers. We start by describing the theoretical foundation for the distance-based outlier detection method in comparison to previous related techniques. We propose a hybrid technique based on two of the best performing state-of-the-art distance-based methods, according to Tran *et al.* [11] DODDS evaluation study. In the subsequent subsections, we present the process of our proposed approach in handling inliers and distance-based outliers in different scenarios.

A. MICRO-CLUSTER WITH MINIMAL PROBING

The newly proposed method called *Micro Cluster with Minimal Probing* (*MCMP*) is a hybrid approach that combines the strengths of the Micro-Cluster Outlier Detection (MCOD) and Thresh_LEAP methods. We applied micro-clusters to minimize the range queries and minimize distance-based computations. The micro-clusters eliminate the need for excessive range queries by storing the neighbor's data points in the micro-clusters. This, therefore, improves the underlying evaluation metrics – memory and time consumption. Also, we adopted Thresh_LEAP's concept of minimal probing for objects that are outside the micro-clusters. The minimal probing principle helps to mitigate the expensive range queries outside the perimeter of the clusters. In our method, using the minimal probing technique, we extend its function further by computing and differentiating the strength of the inliers. We introduced the concept of effective probing and handling of strong and trivial inliers.

The key difference between our approach *MCMP*and the other two algorithms are: (1) Concerning Thresh_LEAP, MCMP inherits the benefits of Thresh_LEAP's minimal probing principle and further explores the handling process of the different kinds of inliers. Besides, it differs from Thresh_LEAP since MCMP adopts micro-clusters that do not exist in Thresh_LEAP. (2) Whilst when compared to MCOD, although our method also embraces the concept of micro-clusters similar to MCOD, our approach ensures that it differentiates between the strength of the inliers to minimize those inliers that are falsely regarded as potential outliers. Also, MCOD does not profoundly explore data points outside micro-clusters, unlike our technique which further explores these data points. In MCOD, this shortcoming results in performance degradation. Therefore, we did not store in memory every data point in the micro-clusters. We consider only the significant ones by exploring the weight of the inliers. This is done to minimize memory consumption and also ensure that we use less time to store these data points, which overall improves its computational time.

Our approximate minimal probing approach applied to compute data points outside the micro-clusters renders our method vulnerable to the risk of losing its accuracy. The fundamental goal of many distance-based outlier detection algorithms [7], [21], [27] is to ensure that the algorithm runs fast with limited use of memory. However, we still do not want to do so at the expense of accuracy. Therefore, we take into consideration the most significant representation of these data points in order not to miss out on crucial outliers, and ensuring that the algorithm does not lose its accuracy. In addition, the algorithm selectively chooses to store only significant inliers.

In the subsequent sections, we will explain how we compute and deal with data points in the current window d_{ci}, in the new and expired slides, and further explain how outliers are reported.

B. DATA POINTS WITHIN THE CURRENT WINDOW

Figure 3 shows the current window space (W_{start} to W_{end}) of evolving data stream objects$\{d_1, \ldots, d_n\}$ with the slide, $S = \{\phi\}$, and a fixed radius R, and the neighbor count threshold, k. In the Figure, the abscissa represents the object's arrival time, while the ordinate shows the number of objects within R. Let the number of objects d_n, from time interval $t_1, \ldots, t_n$ represent the entire window size, $W_n = d_n$ subset or element in $\{t_1 - t_n\}$. The number of micro-cluster for $k = 3$ in the whole window is 2. We first explore the data points $d_1, \ldots, d_n$ in the window. In the current window, we initially define the time/count window size, i.e., t_1 to t_n or d_1 to d_n, which in this case is from t_1 to t_8. d_n is the number of data points in the window. As can be seen, the window contains objects $\{d_1, d_2, \ldots, d_n\}$. We initially apply the concept of micro-clusters to cluster $K+1$ data points, where $k = 3$. From the center of the micro-cluster of a data point, the radius is $R/2$.

We apply the micro-cluster technique to cluster data points that meet the following requirement:

(1) The distance R between two data points within the micro-cluster should meet the Euclidean space.

(2) The micro-cluster should have a radius of $R/2$ from the center of a data point, as shown in C_1.

(3) If $k + 1$ data point is found that is not greater than the distance R from each other, these points are formed into a single micro-cluster.

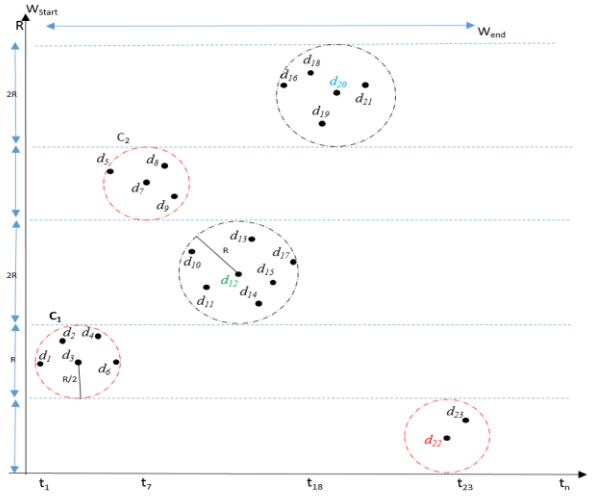


FIGURE 3. Four micro-clusters with K = 3.

From Fig. 3, we have two micro-clusters, C_1 and C_2. i.e. $C_1 = \{d_1, d_2, d_3, d_4, d_6\}$ and $C_2 = \{d_5, d_7, d_8, d_9\}$. Using the concept of triangular inequality in metric space [14], [29], the data points within the micro-clusters are no greater than the distance R, which makes every data point within the micro-cluster an inlier. It is apparent that not every data point belongs to a micro-cluster and the status of these data points are important to be determined; whether they can be labeled as distance-based outliers or inliers. However, all data points outside the micro-clusters are potential outliers, unless otherwise. The key difference between our technique when compared to the others [10], [21] in dealing with data points within the current window is that we implement a different way of dealing with the objects outside the micro-clusters. In the previous method [21], the objects outside the micro-clusters are stored in *PD*, and all the neighbors of these objects are computed to form either a new cluster or to know their outlier status. In our case, we apply the minimal probing in *PD*to avoid computing the distance of every object within *PD* and ensure that not every data point is stored in memory. We label the data points outside the micro-clusters as either strong or trivial inliers depending on their status with respect to the current object.

A data point might not belong to any micro-cluster, but it cannot be considered automatically as a vivid outlier. Even though it can be assumed that all objects outside the micro-clusters are outliers. However, they are not, and it is important to differentiate these objects in the current window. Some objects are inliers, and they can either be strong or trivial inliers depending on different scenarios. To determine the status of these data points, we consider the neighbors before and after the current data point. Since in the current window, we do not take into consideration slide size S; the concept can be better explained when we observe the window slides. If the window slides, within the shortest time, these uncertain data points have the propensity to change status. In the current window, we pay more attention to the proceeding and succeeding neighbors count. Let us consider the whole window W, taking into consideration K and R to label the data objects. As per the definitions above, the requirement of what constitutes a strong and a trivial inlier or a real outlier in the current window is as described in subsection (1) and (2).

1) THE STRONG INLIERS

When the number of succeeding neighbors within the current window exceeds or is equal to the neighbor count threshold, i.e., for $D_s >= K$, such an inlier will be referred to as a strong inlier. The succeeding neighbors have a longer lifespan before they can expire. For instance, d_{12} w.r.t to other objects within R has $\{d_{10}, d_{11}, d_{13}, d_{14}, d_{15}, d_{17}\}$ as its neighbors with $D_{p12}(2) = \{d_{10}, d_{11}\}$ and $D_{s12}(4) = \{d_{13}, d_{14}, d_{15}, d_{17}\}$. Its $D_s + D_p >= K$ with $D_{s12} >$ k (3). Even if the window slides, it still maintains its status. Therefore, it can be referred to as a strong inlier.

2) THE TRIVIAL INLIERS

The data point d_{20} can be considered as a trivial inlier in the current window even though it has greater than k neighbors. But one important thing to note is the number of succeeding neighbors is less. When the number of neighbors is less, the probability of the object changing to an outlier when the window slides is greater. When the window slides, the preceding neighbors will expire, causing it to lose its status. They have a shorter lifespan, but they last longer the closer they are to the current object. If we consider again, in the current window, the data point d_{20}. We can see that $d_{20} = \{d_{16}, d_{18}, d_{19}, d_{21}\}$. It has $D_s + D_p >= K\,(3)$; thus, it is an inlier. Its preceding neighbor is $D_{p20} = \{d_{16}, d_{18}, d_{19}\}$ and its succeeding neighbor is $D_{s20} = \{d_{21}\}$. When these preceding neighbors expire, it will become an outlier since it has very few succeeding neighbors. i.e. $D_s(1) + D_p(0) < K\,(3)$. This is because there are no succeeding data points at this stage that will ensure it will have more neighbors.

3) THE OUTLIERS

As stated earlier, a data point that does not fall within a micro-cluster or does not fit within the state of both strong and trivial inliers can be declared automatically as an outlier. i.e., if its preceding and succeeding neighbor count threshold is less than the neighbor count threshold, $Np + Ns < K$.It becomes a vivid potential outlier if it does not satisfy any of step II (1), II (2), and II (3). For instance, the data point d_{22} has only one neighbor $d_{22} = \{d_{23}\}$. The sum of its preceding and succeeding neighbors is $D_{p23} + D_s(0) < k\,(3)$. Therefore, d_{22} is an outlier in the current window.

Since our technique uses minimal probing, when the search technique is activated for the current object, we do not randomly search for all neighbors. Instead, we use priority ranking by considering first the trivial inliers via all preceding neighbors of the objects and according to their timestamp. This is done by selecting a subset of inliers. The status of the object can be determined once k neighborhood is attained.

From the initial stage, since we have already clearly distinguished between the two different kinds of inliers, then computing the distance for a new object and the current object is not needed. This is because we want to ensure we identify outliers using the lowest memory consumption possible. However, it is also very important for our algorithm to be effective and not to miss the most significant data points. We do not store all the inliers within the micro-clusters, but we are still able to detect outliers accurately. We deal with data points in the event queue by first taking into consideration whether the data point is in the window or micro-clusters, then we look for the expired data points and remove them from the queue. We pay attention also to the number of neighbors of d_i. When d_i is an inlier, we add it to the event queue; with preference given to strong inliers. The trivial inliers are added with respect to their life span aware prioritization; a concept inherited from Thresh_LEAP.

C. PROCESSING THE NEW DATA POINTS AND NEW SLIDE

How we process the sliding window, new slides, and new data points is explained in Fig. 4. When these new changes are applied, they will ultimately influence and create new challenges in the outlier detection process. In most cases, additional computation is required. In the Figure shown, we use different colored rectangles to depict the outline of the sliding windows(w_1-w_4) from the time interval of t_1 - t_n and dashed lines to show the new slides, and boldly marked lines to show new data points.

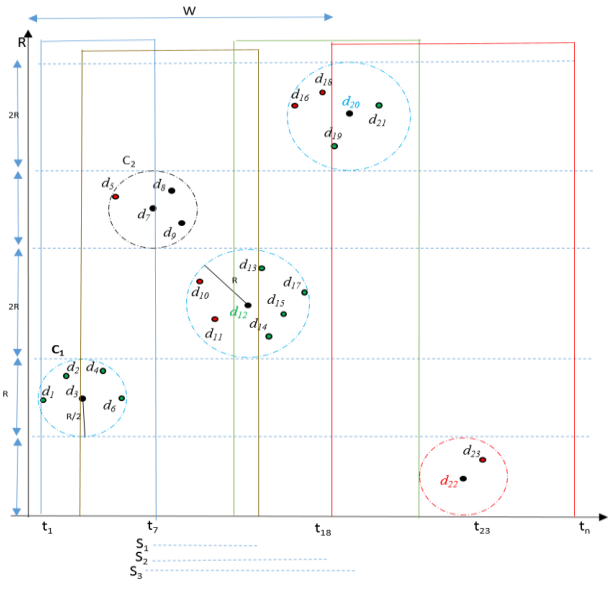


FIGURE 4. Dealing with a new slide for K = 3.

1) THE WINDOW SLIDES

In the first window, w_1(*blue*),t_1-t_7, C_1 still maintains its cluster state while C_2 dissolves because it does not fit in the window size. We can see that d_7 is an outlier having a single neighbor d_5. When the window slides in w_2(orange), we see that C_1 slowly slides out and dissolves because d_1 and d_2 have expired and C_2 regains its status. The object d_{12} does not have enough neighbors $\{d_{10}$ and $d_{11}\}$ to form a cluster; it is a trivial inlier at this stage. In w_3 (green), even though the preceding neighbors are initially observed before the succeeding neighbors, however, in this case, the preceding neighbors D_{p10} and D_{p11} have expired. The object d_{12} remains an inlier since the lifetime of its later neighbor is longer, and it is a strong inlier with four succeeding neighbors. It will continue to stay in the window until it expires. The object $d_{20}$is an inlier, but at this stage, it is a trivial inlier and we can only prove it when the window slides. In the last window slides, w_4 (red), objects d_{16} and d_{18} expire, thereby exposing d_{20} to the outlier state. It only has one succeeding and one preceding neighbor.

2) THE NEW SLIDES

In terms of the new slide, considering slide S_1, and $S_2$with slide size, S_z=*3*. When the slide size is large, the algorithms do not scale well to process the newly arrived data points. Furthermore, when the new slide slides, it may influence the position and status of the active objects. From the Fig. 4, we assume a case where the slide size is 3 for both S_1 and S_2. In S_1, d_{12} is an inlier but a trivial inlier while in S_2, d_{12} is an inlier but a strong inlier. d_{18} and $d_{16}$are not inliers. In the last scenario in S_3, d_{20} an added object in the slide is considered a trivial inlier with three preceding neighbors.

3) THE NEW DATA POINTS

For the new data points, let us assume that d_9 is a new data point in w_2. Initially, although the window slides but still d_7 does not have enough neighbors to form a micro-cluster, but upon the arrival of d_9, d_7 now has enough neighbors to form a micro-cluster. Usually, upon the arrival of new data points, there are additional computations that need to be done. At the onset, with respect to the cluster centers, we execute the *3/2R* range query search. We search for the closest micro-cluster center and if found we check to see if it can accommodate the data point. If it can accommodate the new data point, we then add it to the micro-cluster and update the neighbors of the objects outside the micro-cluster. On the contrary, we execute the *3/2 R* range query on data points outside the micro-clusters. As can be seen from Fig.4, some new data points can easily fit and bond with existing data points to create a new micro-cluster, such as d_7. Meanwhile, some other points within the new slide cannot fit into any new micro-cluster, rendering them prone to becoming an outlier, such as d_{22} and d_{23}. During this process, we differentiate between strong and trivial inliers in *PD* and prioritize processing strong inliers in *PD*. Objects that are not close enough to any cluster can form their clusters after the range query search, provided the requirement of having a *k+1* data point is met. If a new micro-cluster is formed, all the nodes within *R/2* are added. This, in turn, requires an updating of the list of micro-cluster identifiers outside the micro-cluster. We store the data points using a similar technique, except that some data points that were previously in memory must fade out, making it redundant to keep them. In addition, some data points bond with other

data points to form new micro-clusters. Even though new micro-clusters are formed, it shows that other micro-clusters have faded out resulting in reduced computational demand in the window. For the new data points, additional memory is allocated, or it fills in existing memory.

D. PROCESSING THE EXPIRED WINDOW AND SLIDE

In Fig. 5, we illustrate how we deal with the data points when the window slides and expires. We have three expired sliding windows, Ws_1, Ws_2, and Ws_3 from the time interval of t_1 to t_n. The expired data points and expired micro-clusters are marked in red. The current window size w_1 is from t_1-t_7, and during this window interval, we have cluster C_1 and C_2, with C_1 maintaining its status, while cluster C_2 dissolve since, at this time, d_7 has only one neighbor d_5. Therefore, the micro-cluster will dissolve, and then d_7 becomes an outlier in that window. However, C_1 still maintains its current status as a micro-cluster. In slide Ws_2, we notice that d_{12} remains an outlier even though two of its preceding neighbors expired. This concept is analogous to Thresh_LEAP's principle of dealing with the slides. We adopt this approach because we want to correctly identify outliers and improve the accuracy of the method instead of just focusing on satisfying DODDS underlying metrics. In the first instance, when we differentiate between strong and trivial inliers, we define d_{12} as a strong inlier. From Fig. 5, we can see that even though two of its neighbors expired $\{d_{10}, d_{11}\}$ and became obsolete after the slide, it still maintains its status. The object d_{12} still has enough succeeding neighbors to maintain its inlier status, $D_{s12}(4) = \{d_{13}, d_{14}, d_{15}, d_{17}\}$. Therefore, $d_{12} >= k(3)$ is a strong inlier. For d_{20}= $\{d_{16}, d_{18}, d_{19}, d_{21}\}$, after Ws_3 slides, $\{d_{16}, d_{18}\}$ already expired and $\{d_{19}, d_{21}\}$ is not sufficient to allow d_{20} to be an inlier. We regarded it as trivial inlier because when the slide changed, its preceding neighbors expired and created a change of state.

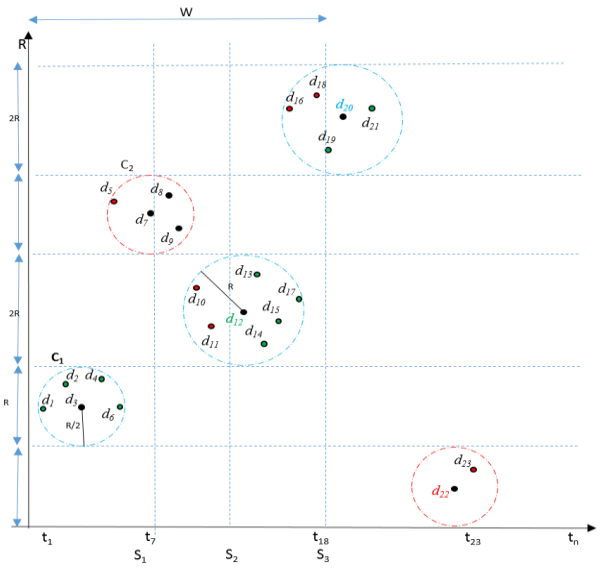


FIGURE 5. Dealing with expired window and slides.

E. PROCESSING AND REPORTING OUTLIERS

The outlier detection process involves the development from the previous Sections- B, C and D. From *Definition 4*, a data point d_i is a distance-based outlier if it does not lie more than the distance threshold R and has fewer than k neighbors in the dataset. In Fig. 3, for the current window model, an outlier by definition is one that does not satisfy any of the requirements stated in *B*. When a distance-based outlier is detected, and after it has been processed, it is reported. Before the outlier detection and reporting process, the algorithm first verifies if the data points belong to a micro-cluster or if they are trivial or strong inliers. For objects outside the micro-clusters, they are all potential outliers and are stored in the latent memory with all trivial inliers stored within the array. As can be seen from Fig. 3, d_{23} and d_{22} are outliers in the current window, since they do not meet the requirement of being an inlier.

Among the outliers, we have vivid outliers that can be automatically detected in the current window. Also, there are those that are susceptible to change of state when the window slides or on the arrival of a new data point, such as d_7, d_{12}, and d_{20}. When we observe carefully, d_{12} cannot be considered a vivid outlier, although initially, it does not meet the requirement of the number of neighbors in the current slide. Therefore, in such a case, instead of being reported as an outlier, it is reported as a trivial inlier. The next incoming window thus verifies and qualifies it to be an inlier. The steps involved in computing and reporting outliers are as follows.

1) Implementing the steps and procedures of previous Sections (B, C, and D). The method verifies if the data point has less than k neighbors, including the strong inliers of the succeeding and preceding neighbors that have not yet expired. All data points that do not meet the requirements are reported as outliers.
2) In Fig. 4, when the new and expired slide processing is completed, the data points with less than k neighbors in the outlier list are reported.
3) All detected distance-based outliers are stored in the outlier list and reported after processing.

V. EXPERIMENTS AND RESULTS

Our work is implemented using Java in Eclipse Java EE IDE, and all experiments are done on a PC with 3.20GHz CPU and 8GB of RAM that runs Windows 10 Operating System. The baseline algorithms were prepared by Tran *et al.* [30].

For the real datasets, we use two accessible datasets, the Forest Cover (FC) [31] and the tropical atmospheric ocean project (TAO) datasets from the Pacific Marine Environmental Laboratory [33]. There are 581, 012 records with 55 attributes and 575, 648 records with three attributes in FC and TAO respectively.

For the Synthetic dataset, we use the Gauss dataset, similar to [7], [17], [20]. The dataset is obtained from a data generator that produces streams with a precise number of outliers and data distribution types. It contains 1 million records with one attribute with uniform noise and a mixture of 3 Gaussian distribution.

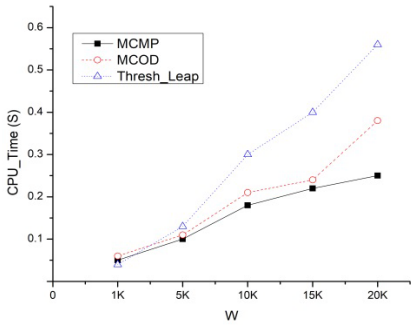
We set the default parameters considering the four key parameters that influence the performance of distance-based outlier methods, which are: R - distance threshold, k- neighbor count threshold, S- the slide size, and W- the window size. We set the parameters to $W = 10,000$ for FC and TAO, and 100,000 for Gauss. For S and K, we set $S = 300$ for FC and TAO and 3000 for Gauss. Then, we set $k = 50$ for all datasets and R is set according to the dataset type. We use $R = 500$ for FC, 1.5 for TAO, and 0.025 for Gauss.

In our experimental work, we evaluate our method in terms of the two most significant metrics used for distance-based outlier detection in the data stream [4], [8], [32]. The CPU time - this record the time consumed in processing the new and expired slides and in detecting the outliers for each window. Meanwhile, the peak memory records the maximum memory consumed by the algorithm during processing, more specifically the data storage for each window.

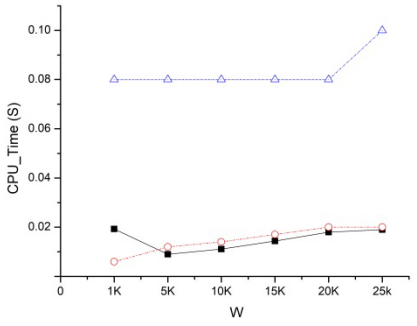
Our proposed method is evaluated by varying the parameters such as the distance threshold R, the window size W, and the K neighbors. We then experiment on the effect of varying these parameters on the CPU time and memory consumption on the different datasets. In our experiment, we focus on two major cost factors, that is, the CPU and the memory usage after varying the above parameters. We evaluated our method with a series of experiments against two key baseline algorithms MCOD and Thresh-LEAP.

A. VARYING WINDOW SIZE, W

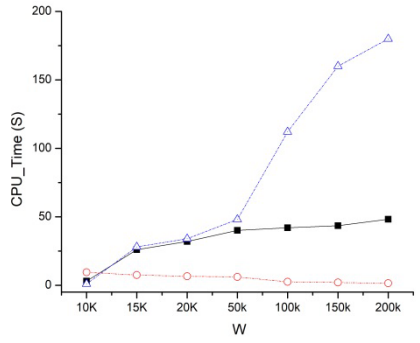
In varying the window size, we expect some changes in time and memory consumption. With an increase in W, we expect the CPU time to increase proportionally; however, as can be seen from the three figures, this is not always the case. Figure 7a, in FC, confirms the hypothesis that increasing W will proportionally increase the consumption time, while for TAO and Gauss dataset, in Fig. 7b and 7c respectively, this only holds for MCOD in TAO and MCMP in Gauss. An exception here is the Thresh_LEAP, which remains stable for the TAO in Fig. 7b and Gauss for MCOD in Fig. 7b, which shows a trend contrary to the inclining trend of other methods for the different datasets. The stable state of the Thresh_LEAP is as a result of each slide maintaining its trigger list. The trigger list inlier status needs to be recomputed before the slide expires since, in each of the slides, there is a trigger list. While for the Gauss dataset, the declining trend is as a result of the fewer number of neighbors for the sequentially generated dataset. When W is small, it consumes more time as compared to when W is larger. MCMP in Fig. 7a and 7b overall show the least time consumed since it uses minimal probing outside the micro-clusters, which ensures its fast computation. However, in Fig. 7c in Gauss, it doesn't show the best performance because the Gauss dataset has fewer neighbors. Therefore, it takes more time for the algorithm to query and differentiate between the points outside the micro-clusters. This also results in the incurring of lower CPU cost among the other algorithms.



(a) FC



(b) TAO



(c) Gauss

FIGURE 6. CPU Time-varying the window size W.

In terms of the peak memory consumption, the figures show that varying the window size W affects the memory. It is also expected that with an increase in W, the memory size will also increase since the algorithms need to store the data points and neighbor information. Both MCMP and MCOD have lower memory consumption as a result of the memory-efficient micro-clusters. However, MCMP is superior in most of the datasets because not all inliers are stored in memory and only crucial inliers are stored through the help the concept of strong and trivial outliers, that reduces the memory cost. The MCMP happens to be inferior to MCOD again in the gauss dataset as a result of the more time it consumes, and the more memory used.

B. VARYING THE NEAREST NEIGHBOR COUNT, K

Increasing the neighbor count threshold k means more points are to be computed and a lower outlier rate. Therefore,

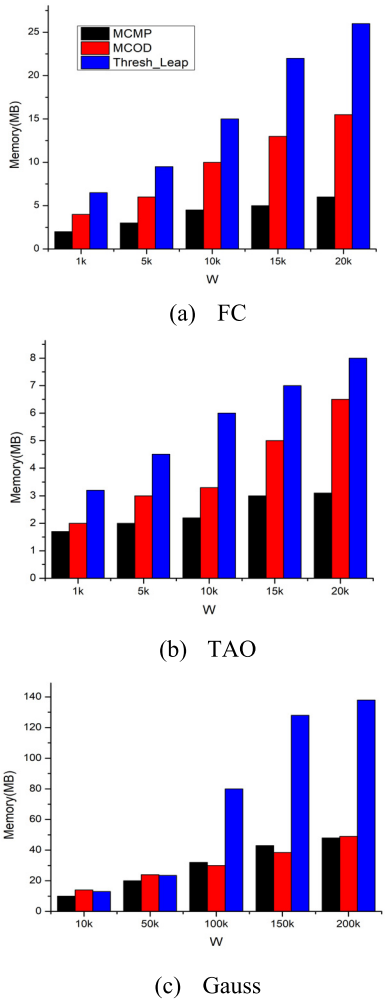


FIGURE 7. Memory - varying the window size W.

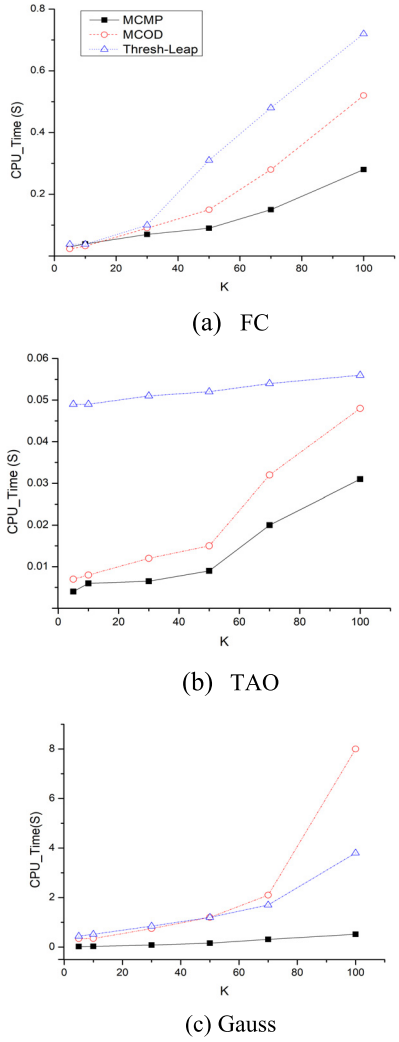


FIGURE 8. CPU Time - varying neighbor count threshold k.

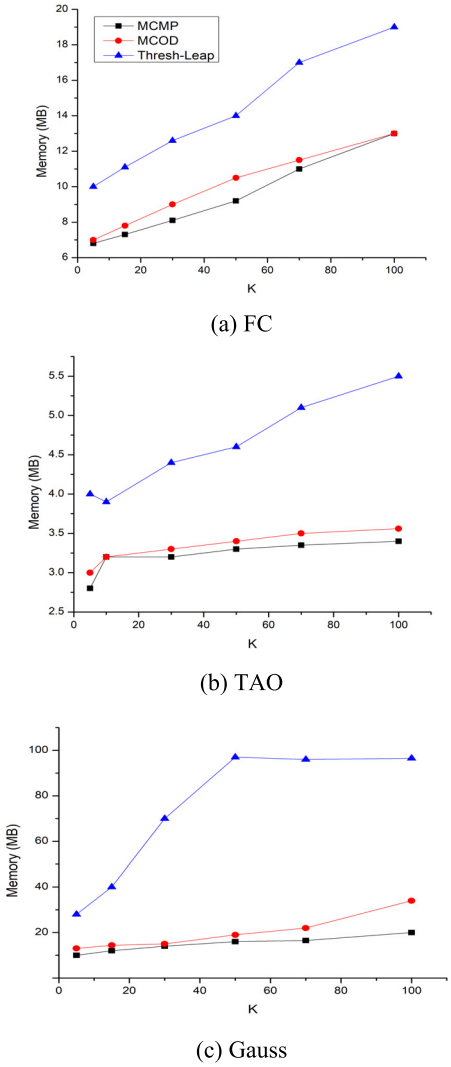
we expect an increase in CPU time and memory consumption, except for algorithms that are not dependent on the neighbor count threshold. From Fig. *8a*, *8b*, and *8c*, we can see that MCMP in all cases shows better performance, and *MCOD* shows superior performance over Thresh_LEAP in all cases. This is because *MCOD* is made up of micro-clusters, therefore with an increase in *k*, few data points fall in these micro-clusters. With *MCMP* also having micro-clusters, we expect the probing process to cost more time to compute *k* neighbors. However, it turns out that, since there is not much to calculate, the resulting time cost remains negligible or continues to remain stable. This trend also holds for Thresh leap except for 7c where it shows a very sharp increase as *k* increases because of the probing requirement needed to find each data point, *k* neighbors. In all the algorithms, as can be seen in Fig. 8a, the *TAO* dataset consumes more time as compared to the others because not many neighbors can be found locally in the dataset.

In terms of memory consumption, we also expect that an increase in the value of *k*will result in more memory consumption since storing the neighbors is dependent on the count threshold. However, this does not hold in all cases, for instance, even though Thresh_LEAP needs more time to process or compute its neighbor list, it also depends on the nature of the data set as in the case of TAO. For MCMP as a result of the probing within micro-clusters and the absence of several data points within the micro-clusters, it turns out it is similar to the trend shown by the CPU time.

C. VARYING THE DISTANCE THRESHOLD, R

In the process of varying R through increasing R, we expect that the CPU time and memory requirements will not increase proportionally since all three algorithms do not make use of range queries. The task of computing the range queries usually is costly in terms of both time and memory requirements. From Fig 9a, 9b, and 9c, it can be seen that at the onset, all the algorithms increase with MCOD consuming more time as a result of the preprocessing involved in the *PD*. In MCMP, the process of adding, removing, and sorting neighbors is computationally costly, more especially for cases where R is less than $1/10^{th}$ of its original value.

In MCMP, the micro-clusters contain fewer or no data points, while MCOD is almost useless with few data points in them. As *R* continues to increase, as can be observed from

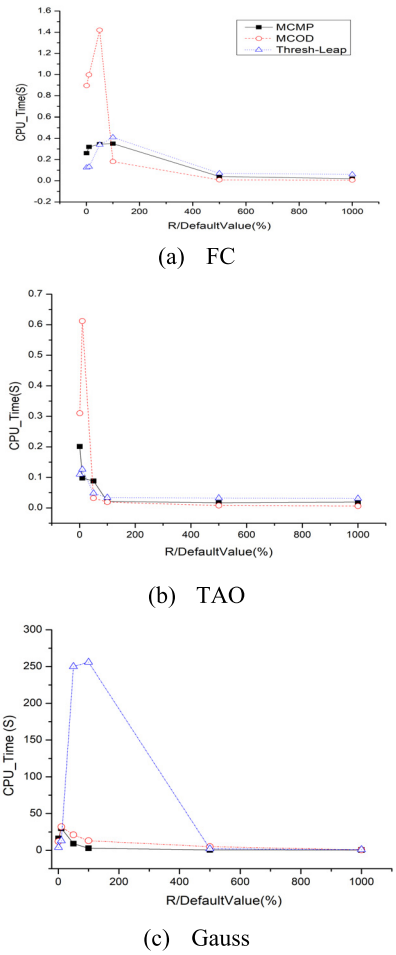


(a) FC

(b) TAO

(c) Gauss

FIGURE 9. Memory - varying neighbor count threshold k.



(a) FC

(b) TAO

(c) Gauss

FIGURE 10. CPU Time- varying distance threshold R.

Fig. 10, we find out more micro-clusters will be engaged with more data points, thus reducing its CPU time. MCMP shows a trend that is slightly below MCOD at the onset and follows a similar trend as the other two algorithms as R increases. It requires the least memory requirement since it does not need to compute large preceding neighbors as in the case of Thresh_LEAP, which shows a similar pattern as MCOD because of the above reason.

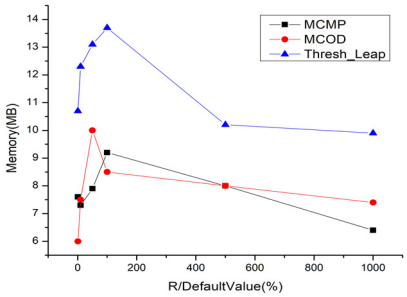
In terms of the memory requirement, to our surprise, the memory requirement for MCMP and MCOD is almost the same with the former showing the least memory requirement. All the algorithms follow a similar trend as in CPU time; this is because the more time it takes means; the more memory is consumed. MCOD and Thresh_LEAP have similar reasons as in CPU time consumption, that is, an increase in R results in a decrease in memory because of the utilization of the micro-clusters, while for Thresh_LEAP, its preceding neighbor count is shorter, resulting in a reduction in memory with an increase in R.

An increase in R also means more neighbors for each data point, which results in a smaller number of outliers. As can be seen in Table 2, R is directly proportional to the outlier rate. For all the datasets, as R increases the outlier rate also decreases.

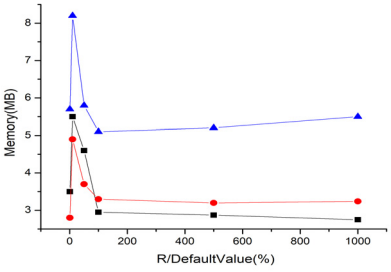
The Advantages and Disadvantages of the Proposed Method:

The Main Advantages of Our Proposed Method Are:

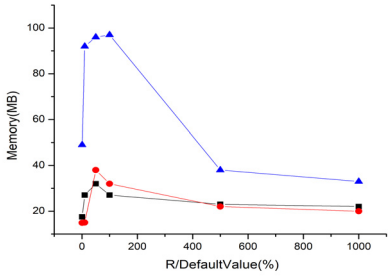
- The use of both micro-clusters and Thresh_LEAP's minimal probing techniques as a hybrid approach shows better performance than earlier methods in terms of both reducing the CPU time and memory consumption.
- The proposed approach of using effective pruning and differentiating of data points outside the micro-clusters is more advantageous in identifying crucial outliers missed by the other baseline algorithms. This, in turn, results in better accuracy. The consideration of crucial inliers is also advantageous as it lessens the method process time and memory consumption.
- The proposed MCMP method does not totally rely on micro-clusters and on Thresh_LEAP's minimal probing, as both have some limitations. It makes the most of the strength of both techniques with additional strategies,



(a) FC



(b) TAO



(c) Gauss

FIGURE 11. Memory - varying distance threshold R.

which makes it better in performance than the previous techniques.

- The proposed MCMP does not need to store every object in memory, which greatly helps in minimizing the storage overhead and cost.
- The micro-clusters adopted in the proposed methods give it the advantage of eliminating the need for range queries and in curbing the distance computations. In addition to only storing crucial inliers in memory, the micro-clusters also improve the memory constraints, since a single micro-cluster has the ability to obtain the neighborhood information of each object in the same cluster.
- The proposed method is shown to be effective for large continuous data streams. It is evaluated using both real-world and synthetic datasets. The results shown have an advantage over the baseline methods as they are more consistent.

The major disadvantages of our proposed method are:

- One major drawback of our technique is the complexity in differentiating between strong and trivial inliers.

TABLE 2. The outlier rate – varying r.

R/Default_R(%)	FC(%)	TAO(%)	Gauss(%)
1	100.0	99.3	98.9
5	100.0	79.7	61.2
10	99.80	49.5	32.3
50	9.9	3.1	3.0
70	7.8	1.1	1.6
100	1.00	0.8	0.7
300	0.06	0.4	0.5
500	0.0	0.1	0.2
700	0.0	0.1	0.2
1000	0.0	0.1	0.2

- MCMP sometimes misses the identification of crucial outliers, since an approximate technique is used for the pruning of objects outside the micro-clusters.

VI. CONCLUSION AND FUTURE WORK

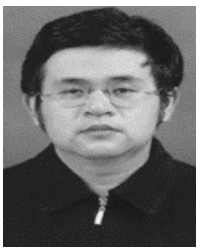
Detecting outliers in data streams continue to be invaluable in the research domain, more especially in the area of data stream analysis and data mining. In this paper, we presented an effective minimal probing approach with micro-clusters to solve the problem of detecting distance-based outliers in data streams. We applied the concept of minimal probing for inliers outside the micro-clusters while simultaneously maintaining the micro-clusters, which minimizes the distance-based computations and range queries. In addition, for inliers outside the microclusters, we applied a concept to differentiate between the strong and trivial inliers. We used real and synthetic datasets to demonstrate the effectiveness of our method and compared it to two well-known baseline methods of DODDS. Our technique showed improved performance in most of the datasets in comparison to the other methods. The method showed superior performance in both underlining metrics used for data streams, i.e., in terms of CPU and memory usage. We plan to extend our work for the detection of multiple distance-based outliers of different data types. In addition, we plan to attempt computing these distance-based outliers for huge datasets within the shortest possible time by using deep learning techniques.

REFERENCES

[1] C. C. Aggarwal, "Outlier analysis," in *Data Mining*. New York, NY, USA: Springer, 2015, pp. 237–263.
[2] Y. Djenouri, A. Belhadi, J. C.-W. Lin, D. Djenouri, and A. Cano, "A survey on urban traffic anomalies detection algorithms," *IEEE Access*, vol. 7, pp. 12192–12205, 2019.
[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.
[4] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proc. Int. Conf. Very Large Databases (VLDB)*, 1998, pp. 392–403.
[5] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications," *VLDB J.*, vol. 8, pp. 237–253, Feb. 2000.
[6] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Outlier detection over massive-scale trajectory streams," *ACM Trans. Database Syst.*, vol. 42, no. 2, p. 10, 2017.

[7] Z. Chen, X. Yu, Y. Ling, B. Song, W. Quan, X. Hu, and E. Yan, "Correlated anomaly detection from large streaming data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 982–992.
[8] C. C. Aggarwal, "Data streams models and algorithms," in *Advances in Database Systems*, vol. 31. New York, NY, USA: Springer, 2007.
[9] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *Proc. 16th ACM Conf. Conf. Inf. Knowl. Manage. (CIKM)*, New York, NY, USA, 2007, pp. 811–820.
[10] 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.*, Mar./Apr. 2014, pp. 76–87.
[11] L. Tran, L. Fan, and C. Shahabi, "Distance-based outlier detection in data streams," in *Proc. VLDB Endowment (PVLDB)*, vol. 9, no. 12, pp. 1089–1100, Aug. 2016.
[12] J. Clark, Z. Liu, and N. Japkowicz, "Adaptive threshold for outlier detection on data streams," in *Proc. IEEE 5th Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2018, pp. 41–49.
[13] S. Sadik, L. Gruenwald, and E. Leal, "Wadjet: Finding outliers in multiple multi-dimensional heterogeneous data streams," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 1232–1235.
[14] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang, "Efficient clustering-based outlier detection algorithm for dynamic data stream," in *Proc. 5th Int. Conf. Fuzzy Syst. Knowl. Discovery*, 2008, pp. 298–304.
[15] C. H. Park, "Outlier and anomaly pattern detection on data streams," *J. Supercomput.*, vol. 75, pp. 6118–6128, Sep. 2019.
[16] X. Qin, L. Cao, E. A. Rundensteiner, and S. Madden, "Scalable kernel density estimation-based local outlier detection over large data streams," in *Proc. EDBT*, 2019, pp. 421–432.
[17] R. Sun, S. Zhang, C. Yin, J. Wang, and S. Min, "Strategies for data stream mining method applied in anomaly detection," *Cluster Comput.*, vol. 22, no. 2, pp. 399–408, 2019.
[18] G. Zhao, Y. Yu, P. Song, G. Zhao, and Z. Ji, "A parameter space framework for online outlier detection over high-volume data streams," *IEEE Access*, vol. 6, pp. 38124–38136, 2018.
[19] 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, 2010.
[20] L. Cao, J. Wang, and E. A. Rundensteiner, "Sharing-aware outlier analytics over high-volume data streams," in *Proc. Int. Conf. Manage. Data*, 2016, pp. 527–540.
[21] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Continuous monitoring of distance-based outliers over data streams," in *Proc. IEEE Int. Conf. Data Eng.*, Apr. 2011, pp. 135–146.
[22] T. Toliopoulos, A. Gounaris, K. Tsichlas, A. N. Papadopoulos, and S. Sampaio, "Parallel continuous outlier mining in streaming data," in *Proc. IEEE 5th Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2018, pp. 227–236.
[23] U. Porwal and S. Mukund, "Credit card fraud detection in e-commerce: An outlier detection approach," *CoRR*, vol. abs/1811.02196, pp. 1–10, Nov. 2018.
[24] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. SIGMOD Conf.*, 1996, pp. 103–114.
[25] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *Proc. KDD*, 2003, pp. 29–38.
[26] F. Angiulli and C. Pizzuti, "Outlier mining in large high-dimensional data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 2, pp. 203–215, Feb. 2005.
[27] A. Ghoting, S. Parthasarathy, and M. E. Otey, "Fast mining of distance-based outliers in high-dimensional datasets," in *Proc. SIAM Int. Conf. Data Mining (SDM)*, Bethesda, MD, USA, 2006, pp. 1–5.
[28] 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)*, New York, NY, USA, 2009, pp. 529–540.
[29] M. A. Khamsi and A. W. Kirk, "The triangle inequality in IR," in *An Introduction to Metric Spaces and Fixed Point Theory*. Hoboken, NJ, USA: Wiley, 2001.
[30] (2018). Information Laboratory University of Southern California. *Distance-Based Outlier Detection in Data Streams Repository*. Accessed: Mar. 27, 2019. [Online]. Available: http://infolab.usc.edu/Luan/Outlier/
[31] S. Hettich and S. D. Bay. (1999). The UCI KDD Archive Irvine. Department of Information and Computer Science. University of California. Oakland, CA, USA. Accessed: Mar. 13, 2019. [Online]. Available: http://kdd.ics.uci.edu
[32] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.
[33] Pacific Marine Environmental Laboratory. (2019). *National Oceanic and Atmospheric Administration*. [Online]. Available: http://www.pmel.noaa.gov

MOHAMED JAWARD BAH received the B.Eng. degree in electrical and electronics engineering from the University of Sierra Leone, in 2013, and the M.S. degree in computer science from the Nanjing University of Information Science and Technology, China, in 2016. He is currently pursuing the Ph.D. degree with the Harbin Institute of Technology, China. His research interests include data mining, outlier detection in data streams, and big data.



HONGZHI WANG is currently a Professor and a Doctoral Supervisor with the Harbin Institute of Technology. He has published more than 150 articles in refereed journals and conferences. His research interests include big data management, data mining, data quality, and graph data management.

MOHAMED HAMMAD received the M.Sc. degree from the Information Technology Department, Faculty of Computers and Information, Menoufia University, Egypt, in 2015. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. He has been a Demonstrator and an Assistant Lecturer with the Faculty of Computers and Information, Menoufia University, since April 2012. His research interests include computer vision, machine learning, pattern recognition, and biometrics.

FURKH ZESHAN received the Ph.D. degree from the Universiti Teknologi Malaysia, in 2014. Since 2014, he has been an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad (CUI) at Lahore, Pakistan. His research interests include service-oriented computing, embedded and real-time computing, agent-oriented software engineering, project management, big data, and semantic web.

HANAN ALJUAID received the B.S. degree from KAU University and the M.S. and Ph.D. degrees in computer science from the UTM University, in 2014. She is with the Computer Science Department, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University. Much of her work has been on improving the understanding, design, and performance of pattern recognition, mainly through the application of data mining and machine learning. She has given numerous invited talks and tutorials. She published numerous articles in pattern recognition, the IoT, and data science. Her research interests include computer vision and NLP.