IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 17, 2020, accepted August 28, 2020, date of publication September 4, 2020, date of current version September 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3021684

Online Clustering of Evolving Data Streams Using a Density Grid-Based Method

MUSTAFA TAREQ[1], ELANKOVAN A. SUNDARARAJAN[1], (Member, IEEE), MASNIZAH MOHD[2], (Member, IEEE), AND NOR SAMSIAH SANI[2]

[1]Center for Software Technology and Management, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM), Bangi 43600, Malaysia
[2]Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM), Bangi 43600, Malaysia

Corresponding author: Elankovan A. Sundararajan (elan@ukm.edu.my)

This work was supported in part by the Ministry of Higher Education Malaysia under Grant FRGS/1/2018/ICT04/UKM/02/1, and in part by the Research University under Grant DIP-2018-041.

ABSTRACT In recent years, a significant boost in data availability for persistent data streams has been observed. These data streams are continually evolving, with the clusters frequently forming arbitrary shapes instead of regular shapes in the data space. This characteristic leads to an exponential increase in the processing time of traditional clustering algorithms for data streams. In this study, we propose a new online method, which is a density grid-based method for data stream clustering. The primary objectives of the density grid-based method are to reduce the number of distant function calls and to improve the cluster quality. The method is conducted entirely online and consists of two main phases. The first phase generates the Core Micro-Clusters (CMCs), and the second phase combines the CMCs into macro clusters. The grid-based method was utilized as an outlier buffer in order to handle multi-density data and noises. The method was tested on real and synthetic data streams employing different quality metrics and was compared with the popular method of clustering evolving data streams into arbitrary shapes. The proposed method was demonstrated to be an effective solution for reducing the number of calls to the distance function and improving the cluster quality.

INDEX TERMS Clustering, data stream, evolving, grid-based method, core-micro-cluster, online.

I. INTRODUCTION

A prime application of big data is the Internet of Things (IoT) and its emergence is primarily due to the increase in the number of devices connected to the Internet. All these devices are typically outfitted with various sensors that can accumulate large amounts of data in real-time or several times per minute [1]–[6]. The IoT creates enormous possibilities in several industries, such as health, resource consumption and transportation [2], [7]–[11]. In the realm of IoT, data streams are common in many applications, such as for comprehensive web searching, the real-time detection of anomalies within network traffic, social networks, environmental monitoring, cyber-physical systems and sensor networks. In these applications, data evolve significantly over time and continuously arrive [12]–[16].

In fact, large data are produced continually as data streams from many applications [17]–[19]. The considerable amount of data generated by healthcare, social media, devices, sensors and software applications have three forms, namely, structured, unstructured and semi-structured data [20], [21]. The diagnostics necessary for these forms of applications are frequently real-time; therefore, the procedures employed must be equipped with the capacity to impart real-time results [22], [23]. Moreover, data-mining procedures are exceptionally functional for this type of diagnostic [24]–[26].

Data stream mining is a comparatively innovative approach in the domain of data mining [27]–[32]. The monitoring of environmental sensors, investigations of social network issues and the real-time identification of irregularities in computer network transmissions and web searches are among the many areas where this procedure is used [12], [33], [34].

Evolving clustering is an essential data analysis topic for a wide range of applications, such as the following: evolving clusters can be generated in forecasting weather conditions [4], [35]–[37], in earthquake forecasting software based on the analysis of different sources of data from the Earth [38], [39], in intelligent transportation systems for

The associate editor coordinating the review of this manuscript and approving it for publication was Amir Masoud Rahmani.

traffic congestion prediction in smart cities [40], in chemistry for forecasting the results of molecular interactions [41], in network intrusion detections (NIDS) [15], [42], [43], in various software performs, and in predicting stock increases or decreases based on their relations with different time series factors [44].

Clustering plays a significant role in the data-stream mining process [42], [45]–[51]. In recent years, many researchers have proposed density-based data stream clustering algorithms. However, several issues related to these clustering algorithms must be considered [13], [52]–[59], such as most are not entirely online methods, are unable to handle evolving data streams, are unable to manage the noisy characteristics of data streams, or suffer from high memory requirements, low processing rates, or the ''curse of dimensionality'' [45], [60]–[63]. Moreover, the existing density-based clustering algorithms have high computational times and low cluster quality for clustering data streams. As such, these algorithms require a large number of distance function calls in order to calculate the distance between data points and micro-clusters.

In this study, our proposed scheme consists of two fundamental steps. In the initial step, the density grid-based procedure is adopted to generate CMCs, upon the occurrence of new data points, in data spaces that are unclustered. The present radius r_0 of the CMCs must be suitably sized to support operational objectives. In this method, a simple linear aging process is used to minimize the life of the CMCs and allows unused CMCs to be removed altogether. Each time new data are received, the CMCs life is usually renewed. In the absence of incoming data, the CMCs will lose a quantity of energy and eventually vanish. If no data are received for a period of time, the CMCs energy will reach zero and be discarded.

The subsequent step entails the integration of any overlapping CMCs into global clusters. CMCs consist of a kernel region and a shell region. The edge CMCs can be discerned by linking the CMCs whose shell regions overlap the kernel regions of other CMCs. CMCs that do not have at least the local density specified by the user (the minimum number of samples within a radius) remain as separate outlier micro-clusters. Each macro cluster consists of the graph of intersecting CMCs where the adjacency relations for each CMC are stored as a property of that CMC. For convenience, we call the CMCs in adjacency relations (i.e., intersecting CMCs) edges. Using this graph structure reduces the calculations needed to separate clusters if a cluster dies and breaks a chain graph, resulting in two groups of CMCs being no longer connected.

Motivated by this observation, in this paper, we propose the 'Clustering of Evolving Data streams via a density Grid-based Method' (CEDGM). To the best of our knowledge, this is the first article that has presented a clustering approach for the evolving nature of clusters incorporating grid granularity as a data reduction stage to simplify the calculation and eliminate the effect of fine data occurring that does not play any role in the clustering result. Different from [47], our approach performs the clustering operations on the grid-based mapping of the data with adjustable granularity, which allows more efficient operations and avoids outlier effects. Additionally, dissimilar from [48], our approach uses CMC generation in an online model, which means that there is no need to store any data before generating the structure of the CMCs, also in the same iteration of updating CMCs the cluster creation is called. We rely on various sample speeds and times to analyze the efficiency of the CEDGM. The results have demonstrated that the proposed algorithm significantly improves the clustering results compared to Clustering of Evolving Date-streams into Arbitrary Shape (CEDAS) [47] and Cauchy [64]. Our algorithm also has better clustering quality, scalability, and efficiency than existing methods.

The remainder of this paper is organized as follows. Section 2 presents a review of the previous studies. Section 3 defines the principles and the methodology associated with the CEDGM. Section 4 describes the use of datasets for evaluating the efficiency of the recommended algorithm, and Section 5 provides the conclusions drawn from this study.

II. LITERATURE REVIEW

Online or data stream clustering has attracted the attention of numerous researchers and analysts. In clustering data streams, an important issue is how to process this infinite data that are evolving over time or how to maintain the vast amount of data for later processing [24], [47], [48], [65], [66]. The literature has provided numerous methods that include data stream clustering.

In the field of density-based data stream clustering, DBSCAN [67] is considered a primitive algorithm that generates arbitrarily shaped clusters incrementally and is unsuitable for high-dimensional datasets due to it suffering from the curse ofdimensionality [68]. Two density-based clustering algorithms, namely, DenStream [69] and CluStream [6], are other density-based clustering algorithms that summarize the data stream information by storing the temporal locality of data in what is called a micro-cluster. Both algorithms are actively applied to evolving data streams. However, DenStream suffers from increased time consumption due to pruning the outlier micro-cluster, whereas CluStream is limited to generating spherical clusters only.

C_DenStream [70], rDenStream [71], SDStream [72], HDDStream [73], and VDStream [74] are all DenStream improvements and are capable of generating arbitrarily shaped clusters. C_DenStream is a semisupervised algorithm in which an expert in the application defines the constraints, although it is unable to handle limited memory issues and cannot handle high-dimensional data streams. rDenStream is appropriate for applications where many outlier micro-clusters are produced and it improves the clustering accuracy; however, it has high memory requirements and a high processing time due to processing and saving the historical outlier buffer through a relearning step. A sliding window-based SDStream algorithm can handle evolving data

streams and noisy data appropriately. However, the maximum number of micro-clusters is predefined, and SDStream is incapable of handling a high-dimensional data stream set due to it suffering from the curse of dimensionality in its offline stage, which is comparable to DBSCAN.

Another clustering algorithm, HDDStream, can generate high-quality clusters and handle high-dimensional data streams. HDDStream's efficiency is further enhanced by adapting the fading function [69] in PreDeConStream [73] to detect the evolving data streams. However, PreDeConStream suffers from consuming too much time, even though VDStream's clustering accuracy is high. As such, searching for density-reachable micro-clusters requires high processing time. In contrast, SOStream [75] is another density-based clustering algorithm that has gained popularity. This algorithm works by accepting the threshold for density-based clustering to sense the structure associated with the evolution of data streams. Moreover, its online phase allows it to dynamically create, remove and merge clusters. SOStream adopts self-organizing maps, which are a competitive learning technology, and it achieves high-quality clustering while occupying less memory. However, this algorithm suffers from a major drawback, i.e., increased time consumption, thereby making this method unfit for data stream clustering.

D-Stream, which is a density-based clustering framework, has been applied to cluster data streams in real-time and was first proposed by [76]. Offline and online phases are involved in D-Stream, in which a new data point is read in the online phase, which is then mapped into a grid. Subsequently, the grid's characteristic vector is updated. Moreover, the clusters are adjusted in the offline phase for each of the time interval gaps. D-Stream also has the ability to cluster data streams in real-time based on the density and the grid. Likewise, handling the outliers in D-Stream is the key motivation for grid density, which regards the outliers as sporadic grids. A sparse grid is a sporadic grid that possesses few data and cannot be transformed into a dense grid. However, high-dimensional data cannot be handled by D-Stream since it considers most of the grids to be empty when a high-dimensional situation occurs. Other current clustering methods, such as the active grid density stream (AGD-Stream) [77], and CLIQUE [78] use density grid decaying technology that identifies active grids, thereby creating clusters through active grids. Here, the data space of the AGD-Stream algorithm is segmented into small cube grids, after which the data object is mapped to this structure. AGD-Stream is time efficient and improves as the stream length increases while achieves high-quality clustering. However, high computational times and large amounts of memory are required for AGD-Stream.

For IoT streams, a density-based clustering algorithm that can be used is hybrid density-based clustering for data streams (HDC-Stream) [17]. This algorithm achieves a quicker processing time compared with its predecessors, thus making it appropriate for a real-time application related to IoT devices. The benefits of micro-clustering and density grid-based methods are incorporated into this method, which can handle outliers and detect arbitrary-shaped clusters. Studies have shown that high-quality clustering results, along with a low processing time, are achieved when working with a data stream. However, the clustering of multi-density data is not feasible and is associated with low memory efficiency [17]. Recently, HDC-Stream was improved by a multi-density data stream (MuDi-Stream) method, which was introduced by [48] in order to resolve the issue of the dramatic decrease in clustering quality when dense data exist. This method is regarded as an online-offline algorithm that incorporates four main components. In the online phase, an information summary of the evolving multi-density data stream is stored in the form of core mini-clusters. The final clusters are generated via the offline phase by applying an adapted density-based clustering algorithm. A hybrid method that integrates the micro-clustering and grid is then applied to store information on the data points. A grid-based method is used to map outliers and form new mini-clusters with various radii.

Four key components are included in MuDi-Stream, which are the components for forming the core mini-clusters, pruning the grids and core mini-clusters, merging or mapping, and forming the final clusters. In the online phase, the first three components are applied, and the last component is associated with the offline phase. M-DBSCAN is another density-based clustering algorithm, which was also suggested for the offline phase, and it forms final clusters with different densities for synopsis data. MuDi-Stream achieves high-quality clustering and occupies less memory. However, the abovementioned streams are inappropriate for high-dimensional data because the number of empty grids will increase and thus will result in a slow processing time.

The algorithms, as mentioned earlier, are either hybrid online/offline or incremental clustering processes. Baruah and Angelov developed two online evolving clustering algorithms for data streams called ELM [79] and DEC [4]. These algorithms require low processing time and provide high cluster purity, but they cannot generate arbitrarily shaped clusters. However, the problem associated with the ELM clustering algorithm lies in its inability to identify a sample's neighborhood given the previously discarded samples. Even though DEC can form hyper ellipsoidal cluster shapes, this technique cannot be adapted to data streams but it can be applied to selecting the optimal radius.

A novel framework for clustering the evolving data stream is known as Cauchy [64]. It is an online learning algorithm that can adapt to the classifier online. In addition, the authors present the idea of designing methodologies for the creation of a cyber-attack detection system. To solve the expensive and time-consuming problems of traditional/offline methods, the method was tested on a 1999 KDD intrusion detection database. The results showed a reduction in the cost of labeling the learning data. Since this approach is online, it is much easier and simpler to include definitions for new attacks than with batch learning algorithms. However, similar to other density-based clustering algorithms, the Cauchy method suffers from a major drawback, i.e., it does not implement the

cluster merging mechanism that could decrease the number of created clusters and it is inappropriate for high-dimensional data.

The previous approaches have their strengths and weaknesses; this is true for almost any clustering approach. Hence, the concept of ensemble clustering emerged such as with RCESCC [80], WOCCE [81], RCEIFBC [82]. The ensemble clustering approach calls for using more than one clustering approach at the same time and it fuses or aggregates their results in order to achieve more robust performance [80]–[82]. Such a category of approaches can combine our evolving algorithm with other clustering algorithms as the aggregated approach.

Another online clustering algorithm called clustering online data streams into arbitrary shapes (CODAS) [18] was proposed to allow the formation of arbitrary-shaped clusters via the online clustering of data streams. CODAS is a data-driven algorithm that generates micro-clusters in order to summarize data points and creates a high quality cluster that can be scaled to multidimensional data streams. However, in CODAS, the generated clusters do not evolve. Recently, CODAS was improved by CEDAS [47] by introducing a simple linear aging process to handle the characteristics of evolving data streams. This algorithm is the first fully online clustering algorithm for evolving data streams. CEDAS includes two mean stages. In the first stage, micro-clusters are produced, or data are added to the current micro-clusters, which is followed by information adjustment. The second stage includes intersecting micro-clusters, in which the micro-clusters are grouped into kernel and shell regions. In this technique, each macro-cluster includes a graph that demonstrates intersecting micro-clusters for each micro-cluster. The storing of the adjacency relations is performed as a characteristic of that micro-cluster. CEDAS immediately provides high-quality clustering results and can also handle the properties of an evolving data stream and noise. However, similar to other density-based clustering algorithms, CEDAS consumes considerable computational time. In the next section, we present our developed methodology.

III. METHODOLOGY

In this study, the CEDGM algorithm is proposed to provide high-quality clusters, detect noise, and determine the characteristics of the data point in an evolving data stream. The proposed algorithm uses the data point information to formulate the CMC. Moreover, this clustering algorithm is entirely online and it uses a density grid-based method to reduce the means of calling the distance function.

The present study conducts clustering based on the density grid. This mechanism forms grids by splitting the data space into small segments. Illustrative neighbor research is then performed on the grids to group them into cluster grids. The density grids and data distribution are displayed in Fig. 1. After a comparison with other clustering algorithms, 'clustering based on a grid' provides a fast processing time since it

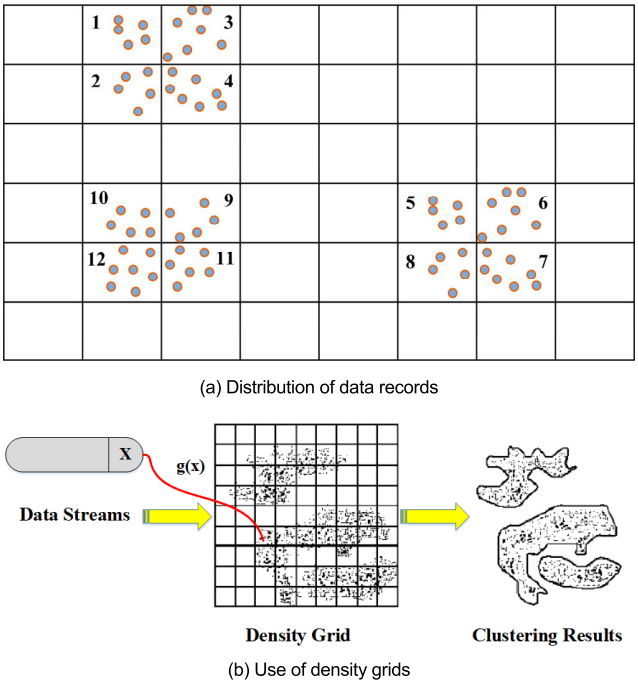

FIGURE 1. Illustration of data record distribution and density grids.

does not rely on the number of data objects, but rather it relies on the number of cells in each dimension. This method is very successful for high density datasets and is robust against noise with almost linear time complexity and distinct arbitrarily shaped clusters.

In the CEDGM, each CMC among radii $r_0/2$ contains a shell region r_0 and a kernel region $r <= r_0/2$. Macro-clusters are formed by intersecting the shell region of CMCs and the kernel regions of other CMCs. The CMCs with a density that exceeds the minimum threshold but with no intersections are also considered macro-clusters. From the data stream, a new data point will fall into three regions. First, if the data point falls in the empty space of a grid granularity, it will then create a new outlier. Second, if the data point falls in the shell region of a CMC, then it can be assigned to the cluster, and the CMC center and cluster count will be recursively updated. Third, the data point allocated to the CMC and the cluster count is updated when the data point falls in a kernel region. The created or modified CMC is examined to determine if the cluster density is greater than the minimum threshold. This CMC is then examined for new intersections with other CMCs. When new intersections are created, these CMCs are linked and assigned to the same macro-cluster. All connected CMCs must have the same macro-cluster and create an arbitrarily shaped cluster in an online manner.

A. PROBLEM FORMULATION

We assume that we have time-series data (xt, yt), where $t = 1, 2, \ldots, t$, $x_t = (x_1^t, x_2^t, \ldots, x_m^t) \in R^m$, $y_t \in \{1, 2, \ldots, Nc_t\}$ and Nc_t denotes the number of clusters at moment t. The clusters C_i are defined as $C_i = \{C_i^t\}$, which means that the

cluster is defined based on the points that belong to it at each time unit, or in other words, the shape of the cluster changes concerning time. The problem seeks to partition the data into its corresponding clusters to obtain the least difference between the predicted clusters and the actual ones. That is, it seeks to find the predicted y_t^p for each x_t such that $y_t^p = y_t$ for most samples. The difference between the classical clustering problem and the evolving one is dynamic, which is considered in the result of the evolving problem. The cluster in classical clustering is not dependent on time C_i, while in the evolving problem, it is dependent on time $C_i = \{C_i^t\}$.

B. PRELIMINARIE

In this section, we introduce the CEDGM. The following terminologies are used in the CEDGM algorithm.

1) GRAPH OF CLUSTERS

This structure illustrates the building of the macro-clusters by intersecting CMCs. An edge will collectively record the intersection of each CMC with the suitable CMC assignment in 'Macro'. Two CMCs are considered edged if the kernel region of a CMC intersects with the shell region of another CMC. Mathematically, two CMCs with radii R_1 and R_2 are considered intersected if the (d) distance between the centers is less than or equal to the intersecting distance $(R_1 + R_2/2)$. An example of the relationship between the graph structure and CMCs is illustrated in Fig. 2(a).

2) CORE MICRO-CLUSTERS (CMCS)

They are defined as the primary entities that construct the cluster. It is a data structure that includes two properties: the center and density. All CMCs have the same radius according to the minimum allowable density. At time t, a CMC is described as a set of close data points $X_1, X_2, X_3, \ldots, X_{Nt}$ in a high-density area wherein the local density N_t is equal to or exceeds the threshold $(N_t \geq Th_{\text{density}})$. An example of the data points and CMCs is depicted in Fig. 2(b).

3) OUTLIERS

They are defined as the group of one or more data points X_1, $X_2, X_3, \ldots, X_{Nt}$ in a low-density area at time t, where the local density N_t is less than a predefined threshold $(N_t < Th_{\text{density}})$.

4) SAMPLE

It is a streaming data point within d dimension.

5) GRID DIMENSION

It is how many sub-segments are considered in a certain dimension to define the coordinates of this dimension. Assume that in dimension d, the minimum value is V_{min}, and the maximum value is V_{max}. Then, the resolution is defined as follows:

$$Resolution = \frac{[Vmax - Vmin]}{GridGranularity} \tag{1}$$

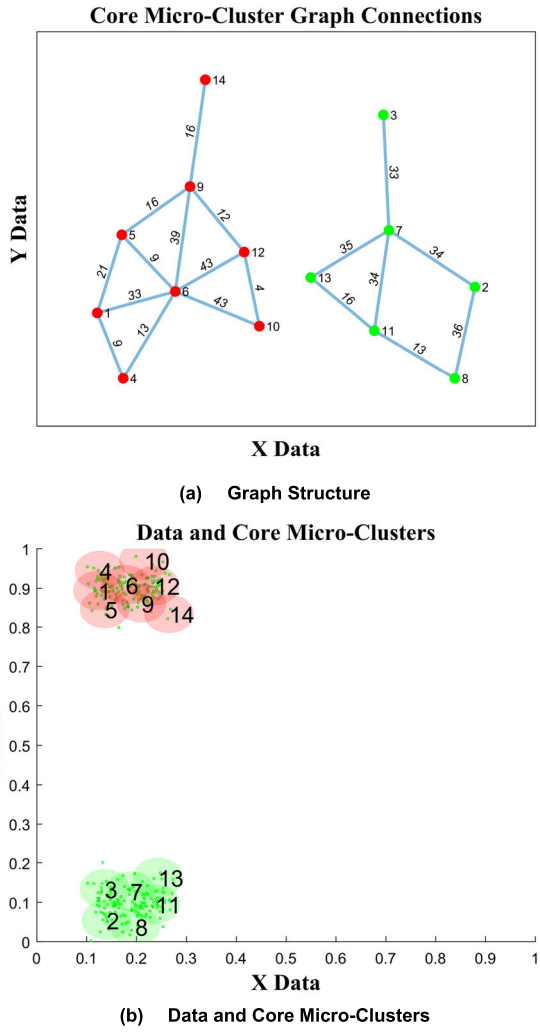

FIGURE 2. Graph structure of the CEDGM algorithm and the core micro-clusters. The graph structure with subgraph nodes is demonstrated in Fig. 1(a). The data with core micro-clusters are exhibited in Fig. 1(b).

The resolution is already calculated by the equation provided in Eq. 1. To elaborate more, we have to first set the value of the grid granularity, which is selected based on a tuning process. A smaller grid granularity implies more calculations and sensitivity to low-frequency noise while a high grid granularity implies less sensitivity in the data changes before clustering. Hence, we set a suitable value depending on the data. In the experiment, we selected a grid granularity of 30.

C. DESCRIPTION OF THE PROPOSED CEDGM ALGORITHM

Before the implementation of the proposed CEDGM algorithm, a few application-dependent parameters are described based on the expertise of the application comparable to other density-based clustering algorithms, such as CluStream, DenStream, Cauchy, CODAS, DEC, MuDi Stream and CEDAS. The CEDGM algorithm requires several parameters to be performed.

The parametric values depend on the applications as follows:

1. **Decay:** This parameter denotes how many last samples we consider for processing at the current time t. If the decay is set to N and we are at moment t, then we consider samples $t, t-1, t-2, \ldots, t-N+1$.
2. **Fade:** It indicates the time in which a CMC has to be removed if no new point or sample was added to it within this time. It is calculated as the inverse of the decay, as shown in Eq. (2):
$$FD = \frac{1}{DC} \tag{2}$$
where FD denotes the fade, and DC denotes the decay.
3. **Radius**: This parameter is the minimum allowable distance for a sample to be from the center of a CMC to still belong to it. Otherwise, it belongs to either an outlier, or it creates a new outlier.
4. **Minimum Threshold**: This parameter is the minimum number of data points that are required to form a CMC or convert an existing outlier to a CMC.
5. **Grid Granularity**: It is an important parameter for the computational time of the CEDGM algorithm, and it represents the difference in the counting structure nodes used for grids.

The CEDGM is a new algorithm for discovering the clusters of evolving data streams in multi-density environments. This algorithm maintains a summary of information on evolving data streams in the form of CMCs. A grid-based method is used as an outlier buffer to handle noises and multi-density data and reduce the number of distance function calls. After setting the application parameters, the proposed CEDGM algorithm is executed on the data stream $\{X_1, X_2, X_3, \ldots, X_m\}$ using the following steps:

- Assign the core micro-clusters,
- Kill the weak core micro-clusters, and
- Update the cluster graph.

The characterization of each step is provided next. For each data sample, the algorithm executes the three steps sequentially.

Step 1: Assign the Core Micro-Clusters

Input: *x, radius, counter, grid Granularity, grid Dimensions, min Threshold*
Output: *CMCs, outliers, clusters Relations, number of Clusters*
Start:
Determine the grid coordinates (*grid Granularity, x, grid Dimensions*)
Compute the distance between data point x and the near CMCs and outliers' centers
If (*the distance* > *radius*) **then**
  Create a new outlier
End
If (*the data point x is nearer to a CMC*) **then**
  Update the CMC and determine the grid coordinates
  Determine the life of CMC % go to step 2
Else If (*the data point x is nearer to an outlier*) **then**
    Update the outlier and determine the grid coordinates
  If (*density(outlier)* > *min Threshold*) **then**
    Update the Outlier to be a new CMC
    CMC = CMC + 1
    Determine the life of CMC % go to step 2
  End
End
If (*new CMC was arrived*) **then**
  Update cluster graph % go to step 3
  Edges = determine Edges (CMC)
End

1) ASSIGN THE CORE MICRO-CLUSTERS

This part of the algorithm uses a density grid-based method in which the data space is partitioned into small segments called grids. The segments and intersection points in the standard grid are called cells and nodes, respectively. Each data point X_i in the datastreams are mapped into a grid, and the grids are clustered based on their density. When a new data point arrives, the algorithm determines the CMCs and outliers in the grid and its neighbors. Step 1 shows a process for assign core-micro clusters.

Next, the algorithm checks the incoming data points that belong to any existing CMCs or outliers. If the distance (d) between the point and the nearest outlier or CMC is less than the radius, it is expressed as given in Eq. 3. Then, the CMC or outlier is updated and the grid coordinates are determined; otherwise, a new outlier is created. Further verification is conducted to determine if the update unit is an outlier and if the count is larger than the minimum threshold. Then, the algorithm will promote the outlier to a CMC. The new CMC must be assigned to the cluster of the nearest CMC in the shell by determining the edges. Fig. 3 shows the flowchart for assign the core micro-clusters.

$$d\,(X_i, C) < R \tag{3}$$

2) KILL THE WEAK CORE MICRO-CLUSTERS

This part of the algorithm minimizes the life of CMCs and removes them when their life is below zero. The life of CMCs is reduced using the fade. When a CMC is removed, all edges that refer to it will also be removed, and the total number of CMCs will be decreased. Step 2 shows a process for kill core-micro clusters. Fig. 4 shows the flowchart for kill the core micro-clusters.

3) UPDATE THE CLUSTER GRAPH

Like other density-based algorithms, such as CEDAS and CODAS, a clustering graph is maintained in order to create an online macro-cluster. The clustering graph makes a change if either of the following occurs.

Step 2: Kill the Core Micro-Clusters

Input: *CMCs, Fade*
Decrease the life of *CMCs* using the Fade
If (*CMC. life <= 0*) **then**
Remove the dead CMCs and all references
Remove all edges containing the CMCs
Reduce the number of CMCs
End

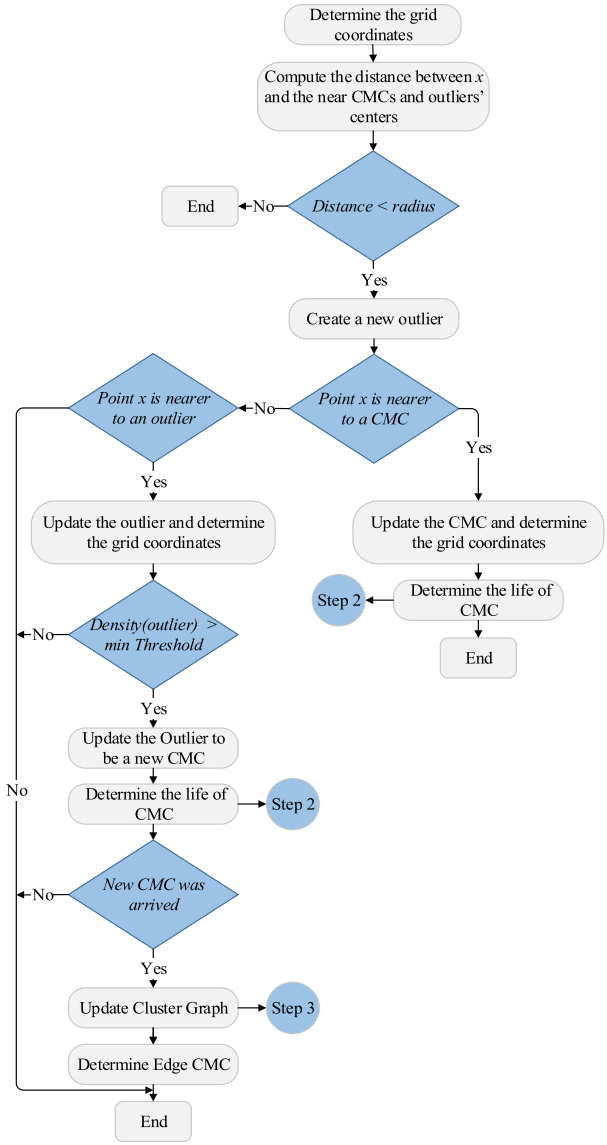

FIGURE 3. Flowchart for assign the core micro-clusters.

- A new link has been created when a new CMC arrives or when old CMCs move. For each new link connecting two CMCs from different clusters, the link ID is added to the cluster table.
- An old link has been removed (when a CMC is removed or moved).

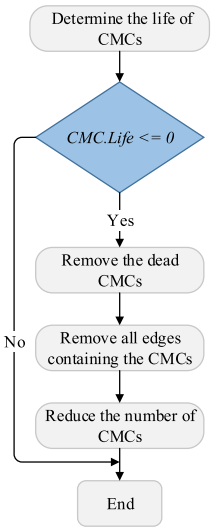

FIGURE 4. Flowchart for kill the core micro-clusters.

Step 3: Update the Cluster Graph

Input: *CMCs*
Output: *Clusters*
Start:
If (*new CMC was created*) **then**
Calculate the distances between new CMC centers and all others
If (*distances <1.5∗radius*) **then**
List the CMCs that are edges
End
End
If (*CMC edge list has changed*) **then**
Set a new macro-cluster number throughout the graph
End

Step 3 shows a process for update the cluster graph. The changes are made to any CMC that has been modified when either its center location is moved or by being in a CMC that has newly reached the threshold. In this case, the graph edges may have changed. If the edge list has changed, then the new graph has its macro-cluster number set to a new value. Fig. 5 shows the flowchart for update the cluster graph.

D. COMPUTATIONAL COMPLEXITY

Computational time mainly involve two sub-processes i.e., cluster assignment and CMC update. Cluster assignment is performed based on the Euclidean distance between the arriving data point and the current CMC centre. Time complexity is $O(ND)$, where N is the total number of data points clustered and D is the number of dimensions.

This algorithm checks the arriving data point that belongs to any existing CMC or outlier. Otherwise, the data point will be mapped to the grid and a new outlier will b created. In the CEDGM algorithm, the grid is implemented as a tree that allows fast lookup, update, and deletion. The key feature

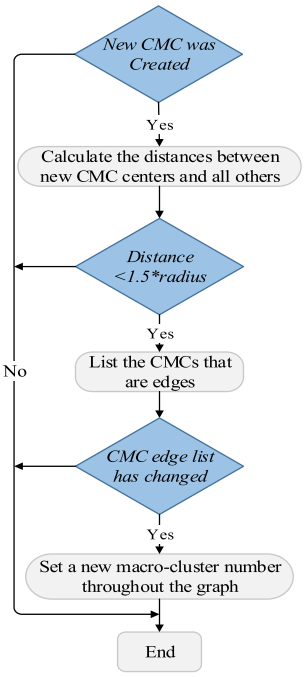

FIGURE 5. Flowchart for update the cluster graph.

of the tree is determining the grid coordinates. Meanwhile, the associated data for each grid entry is the grid's synopsis. Therefore, we show that the size of the grid in our algorithm is $O(log\, D)$, where D is the number of dimensions. The time complexity for update and search in the tree is $O(log\, log\, D)$; which is very small.

In the CMC update, the intersection of the modified CMC with other CMCs is performed with the time complexity of $O(N)$. The three complexities, $(O(ND) + O(log\, log\, D) + O(N))$, are combined. The resulting time complexity of the proposed CEDGM algorithm is therefore $O(ND)$.

IV. RESULTS AND DISCUSSION

This section analyses and compares the performance of the CEDGM algorithm with those of the CEDAS [47], Cauchy [64], MR-Stream [60], WOCCE [81] RCESCC [80], RCEIFBC [82], DBSCAN [67], and CLIQUE [78]. These algorithms are implemented in MATLAB R17a, and their performances are evaluated on a PC with an Intel Core i5 processor @ 2.66 GHz and 16.0 GB of RAM. The parameters of the proposed and other algorithms are *Decay* $=$ 1000, *Minimum Threshold* $=$ 4 *data points, Radius* $=$ 0.05, *and Grid Granularity* $=$ 30. This experiment aims to verify the average number of distance function calls, average purity and average accuracy across sample speeds and times. For logical samples of numbers, the decay is set to ensure appropriately sized macro-clusters for demonstrating the efficiency of the technique. The minimum threshold of CMCs is set to 4. The radius is constantly set small to ensure multiple CMCs. The grid granularity is set to 30, higher grid granularity causes a higher number of children for each node in the tree, which leads to better results. We introduce datasets that are used to determine the efficiency of the suggested algorithm in handling evolving data streams. Real/synthetic datasets are used to assess the proposed algorithm.

A network intrusion detection dataset (KDDCUP'99) is a real dataset for testing the performance of evolving clustering algorithms and contains TCP connection logs from 2 weeks of LAN traffic. The dataset comes from the 1998 DARPA Intrusion Detection dataset. It includes training data consisting of 7 weeks of network-based intrusions inserted in the normal data and 2 weeks of network-based intrusions and normal data for 4,999,000 connection records described by 42 characteristics. Each record corresponds to either a normal connection or an attack. All 34 continuous attributes of the KDD CUP '99 are used, as in [46]–[48], [69], [75], [83]. A conversion of the dataset into data streams is conducted by taking the data input order as a streaming order. In addition, three datasets were used from the UCI machine learning repository [84] i.e., Half-Ring has 373 data points, Iris has 150 data points, and Galaxy has 323 data points.

Fig. 6 plots the synthetic datasets used called DS1 and Spiral. DS1 has 9,199 data points [85]–[87], and Spiral has 6,012 data points.

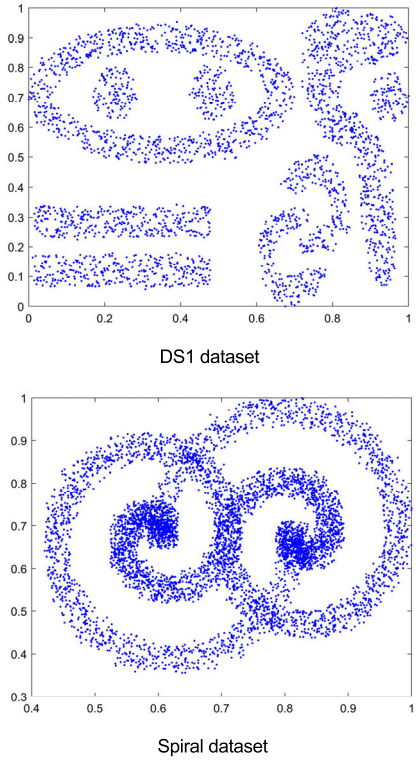

FIGURE 6. Plots of the datasets used for testing the CEDGM.

The class labels are known for all the datasets used in our experiments. Therefore, the quality of the clustering obtained is assessed by considering outstanding external criteria, namely, the average number of distance function calls, the average purity and the average accuracy.

1. **Purity**: For each cluster, purity is the class most frequently divided by the number of data points in that cluster [47], [88], [89]. Purity may be used for the clustering analysis of data streams in various studies and is defined as:

$$Purity = \frac{\sum_{i=1}^{N} n_i^d}{n_i} \times 100\% \tag{4}$$

where n_i^d is the dominant class sample, n_i is the number of samples that a cluster contains, and N is the number of clusters.

2. **Accuracy**: To evaluate the data stream-clustering, the accuracy is used and can be defined as the number of samples in a cluster that belong to that cluster and do not belong to any other cluster [47], [88], [89].

$$Accuracy = \frac{\sum_{i=1}^{N} n_i^d}{\sum_{i=1}^{N} n_i} \times 100\% \tag{5}$$

where n_i^d is the dominant class sample, n_i is the number of samples that a cluster contains, and N is the number of clusters.

3. **Normalized Mutual Information (NMI)**: NMI is derived from entropy in information theory. For a discrete random variable P^*, which measures the mutual dependence between P^* and L^t [80]–[82], the NMI is defined as follows:

$$NMI\left(P^*, L^t\right) = \frac{\sum_{i=1}^{c}\sum_{j=1}^{c} n_{ij} \log_2\left(\frac{nn_{ij}}{n_i^* n_j^t}\right)}{\sqrt{\sum_{i=1}^{c} n_i^* \log_2\left(\frac{n_i^*}{n}\right) \times \sum_{i=1}^{c} n_j^t \log_2\left(\frac{n_j^t}{n}\right)}} * 100\% \tag{6}$$

where P^* is the consensus partition, L^t is the ground-truth of the dataset, n is the total number of data points in the given dataset X, n_{ij} is the number of data points in the intersection of the i^{th} cluster of P^* and the j^{th} cluster of L^t, n_i^* and n_j^t are the number of data points in the i^{th} cluster in P^* and the number of data points in the j^{th} cluster in L^t, respectively.

A. NETWORK INTRUSION DETECTION

The comparisons between the CEDGM, CEDAS, Cauchy, and MR-Stream algorithms on the data stream set for network interruption detection are displayed in Fig. 7. The outcomes are calculated at different sample speeds and times, where the parameters of the proposed and other algorithms are $Decay = 1,000\ samples, Radius = 0.05, Minimum\ Threshold = 4$ and $Grid\ Granularity = 30$. The result of the average number of distance function calls is compared at different sample speeds and times on the high dimensional KDDCUP'99 dataset, which is precisely the same test dataset used in [47]. When the sample speed varies from 5 pits per second (PPS) to 25 PPS, the average number of distance function calls increases, as illustrated in Fig. 7(a). The CEDGM algorithm consistently has a lower average number of distance function calls compared to the CEDAS algorithm. When the sample speeds are 15 and 25 PPS, the CEDGM exhibits average numbers of distance function calls of 1,030 and 1,472 compared with the CEDAS algorithm, which has values of 1,201 and 1,665, respectively.

However, when the time varies from 100 s to 500 s, the average number of distance function calls increases, as illustrated in Fig. 7(d). The average number of distance function calls of the CEDGM exceeds that of the CEDAS. For example, with times of 100 s and 500 s, the average numbers of distance function calls for the CEDGM are 6,396 and 13,448 compared with the averages of 8,444 and 17,108 for the CEDAS algorithm, respectively. The CEDAS algorithm can detect outliers and arbitrarily shaped clusters; however, when analyzing the membership of arriving data points regarding any current CMCs, CEDAS calculates the distance for the new data point with all other visibility agents. Many Euclidean distance calculations are therefore required. By contrast, the CEDGM algorithm has a lower average number of distance function calls because when using grid-based density algorithms, not only can outliers and arbitrarily shaped clusters be located but also fast processing times can be achieved. In other words, they do not depend on the number of data objects but on the number of cells in the quantized space in each dimension.

The comparison between the CEDGM, CEDAS and Cauchy algorithms in terms of the average clustering purity is depicted in Fig. 7(b). The average purity, as defined in Eq. (4), is determined by the sample speed. Our proposed algorithm has a higher purity than those of the CEDAS and Cauchy algorithms. For example, one cluster appears in one window at speeds of 5 and 25 PPS, and the CEDGM algorithm achieves average purity values of 99.88% and 99.83%, respectively. This result indicates that nearly all samples are adequately empowered to be the predominant clusters. The average purity values of the CEDAS algorithm at sample speeds of 5 and 25 PPS are 99.63% and 99.47%, respectively, and those of Cauchy are 98.97% and 99.05%, respectively, considering the few misallocated samples in clusters with few numbers. Fig. 7(e) demonstrates the results of the mean purity analysis for the time of 25 s. We use this estimation that is favored by Hyde and observe that the average purity of the CEDGM surpasses those of the CEDAS, MR-Stream and Cauchy. The two periods of 50 s and 150 s are used. We determined that the average purity values of the CEDGM are 100% and 99.29% compared with the values of the CEDAS of 98.29% and 96%, respectively; the values of MR-Stream of 90% and 82.05%, respectively; and the values of Cauchy of 98.66% and 98.75%, respectively. We then examine the results at periods of 475 s and 500 s and find that the average purity of the CEDGM is 100%.

The following reasons contribute to the improved performance of the CEDGM algorithm.

1. A new CMC that can capture the characteristics of a mixed data object and is accurately distributed is introduced. This

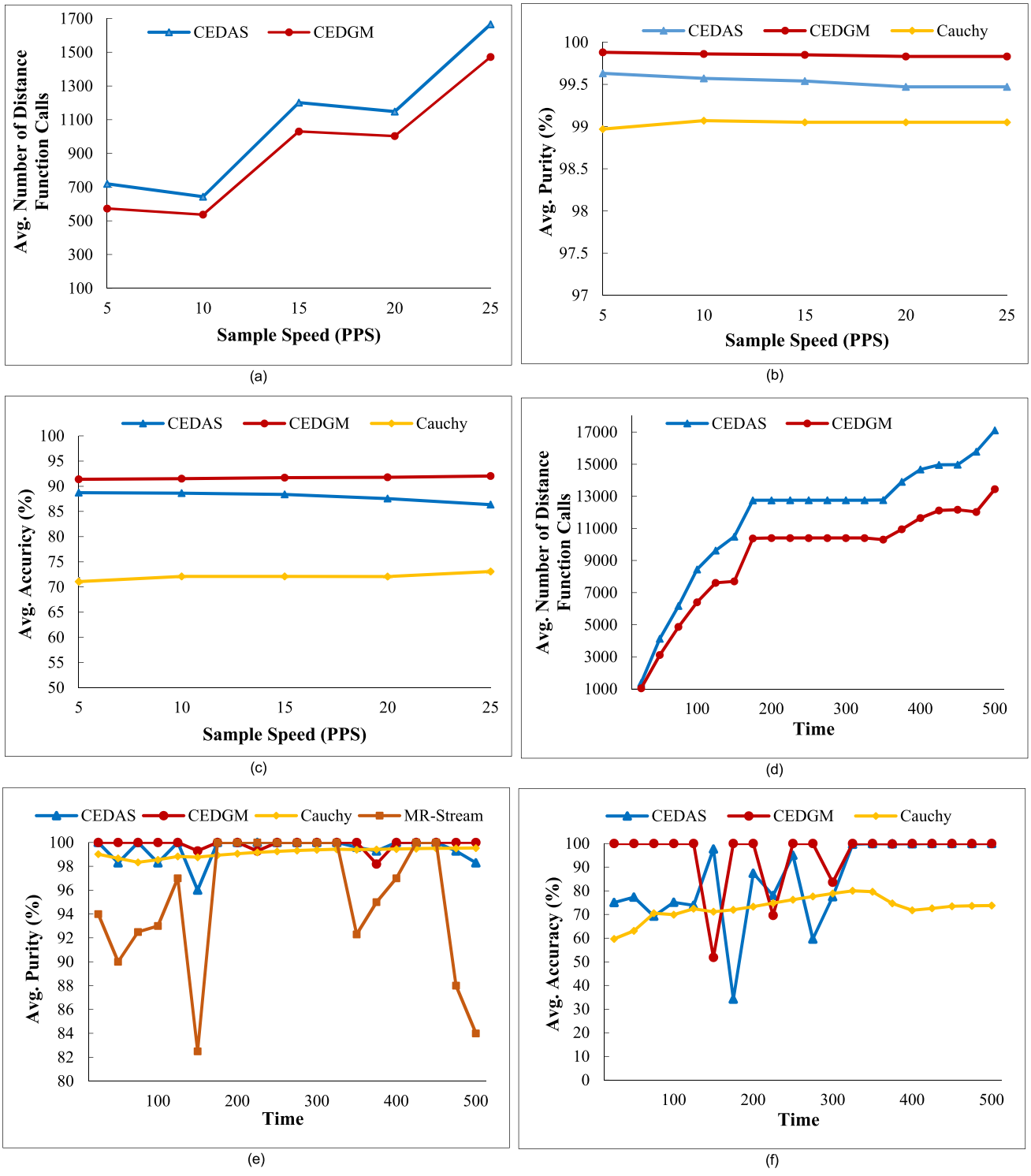

FIGURE 7. CEDGM, CEDAS, Cauchy, and MR-stream performance comparison with sample speed and time using high dimensional KDDCUP'99.

feature makes the cluster purity of the CEDGM algorithm increasingly more accurate.

2. The CMC maintenance mechanism of the online CEDGM algorithm can remove the outliers in time and cluster the potential CMCs, thereby enhancing the clustering purity.

The experimental results of the average clustering accuracy of the CEDGM and CEDAS algorithms are exhibited in Fig. 7(c). The average clustering accuracy of the proposed algorithm outperforms that of the CEDAS and Cauchy. When the sample speed varies from 5 to 25 PPS, the average accuracy of the CEDGM is constantly higher than 91%, whereas those of the CEDAS and Cauchy algorithms are less than 89%. When the sample speed is 25 PPS, the average accuracy of the CEDGM algorithm is 92.02%, whereas that

TABLE 1. Performances of different state-of-the-art methods compared with the performance of the proposed method, i.e., CEDGM, in terms of the sample speed using high dimensional KDDCU'99 as validated by the paired t-test.

Sample Speed	Average Number of Distance Function Calls		Average Purity			Average Accuracy		
	CEDGM	CEDAS	CEDGM	CEDAS	Cauchy	CEDGM	CEDAS	Cauchy
5	**572.99**	719.84	**99.88**	99.63	98.97	**91.39**	88.72	71.07
10	**536.74**	643.07	**99.86**	99.57	99.07	**91.51**	88.63	72.1
15	**1030.31**	1201.47	**99.85**	99.54	99.05	**91.7**	88.38	72.09
20	**1003.08**	1148.35	**99.83**	99.47	99.05	**91.79**	87.54	72.07
25	**1472.07**	1665.69	**99.83**	99.47	99.05	**92.02**	86.34	73.08
t.test	-	**0.00047**	-	**0.00011**	**5.17E-06**	-	**0.00241**	**1.02E-07**

TABLE 2. Performances of different state-of-the-art methods compared with the performance of the proposed method, i.e., CEDGM, in terms of the time using high dimensional KDDCU'99 as validated by the paired t-test.

Time	Average Number of Distance Function Calls		Average Purity				Average Accuracy		
	CEDGM	CEDAS	CEDGM	CEDAS	Cauchy	MR-Stream	CEDGM	CEDAS	Cauchy
25	**1045.35**	1380.45	**100**	100	99.02	94	**100**	75.03	59.66
50	**3114.08**	4127.50	**100**	98.29	98.66	90	**100**	77.33	63.06
75	**4865.26**	6169.13	**100**	100	98.33	92.5	**100**	69.29	70.50
100	**6396.61**	8444.07	**100**	98.29	98.54	93	**100**	75.14	69.90
125	**7610.31**	9620.04	**100**	100	98.83	97	**100**	73.87	72.42
150	**7699.61**	10486.94	**99.29**	96	98.75	82.5	**51.86**	97.66	71.14
175	**10377.47**	12749.19	**100**	100	98.93	100	**100**	34.21	71.98
200	**10400**	12750.02	**100**	100	99.06	100	**100**	87.40	73.32
225	**10400**	12750.03	**99.28**	100	99.16	100	**69.64**	78.05	74.79
250	**10400**	12750.04	**100**	100	99.25	100	**100**	95.06	76.24
275	**10400**	12750.06	**100**	100	99.31	100	**100**	59.62	77.60
300	**10400**	12750.07	**100**	100	99.37	100	**83.64**	77.52	78.86
325	**10400**	12750.08	**100**	100	99.42	100	**100**	99.69	80.01
350	**10301.21**	12763.36	**100**	99.6	99.39	92.3	**100**	99.94	79.60
375	**10946.62**	13902.82	**98.19**	99.28	99.41	95	**99.87**	99.90	74.66
400	**11642.93**	14670.60	**100**	100	99.44	97	**100**	99.89	71.78
425	**12116.17**	14956.15	**100**	100	99.48	100	**100**	100	72.60
450	**12164.00**	14968.91	**100**	100	99.50	100	**100**	100	73.46
475	**12024.83**	15778.36	**100**	99.28	99.51	88	**100**	100	73.64
500	**13448.98**	17108.79	**100**	98.29	99.53	84	**100**	100	73.76
t.test	-	**5.14E-11**	-	**0.05003**	**4.77E-05**	**0.00166**	-	**0.02768**	**7.81E-07**

of the CEDAS algorithm is 86.34% and Cauchy is 73.08%. The results of the 25 s period, as displayed in Fig. 7(f), are also used as a reference. For the period of 25 s to 125 s, the average accuracy of the CEDGM algorithm is 100% compared with those of CEDAS and Cauchy that are less than 78%. The CEDGM algorithm is based on the density grid of the enhanced clustering. When calculating the density of grid cells, the impact of the boundary data points on the grid is calculated rationally by calculating the impact coefficient of the added data points on the adjacent grid cells. Thus, the data points cannot be treated as noise points, and this characteristic improves the clustering accuracy. Table 1 includes the performances of different state-of-the-art methods compared with the proposed method in terms of the sample speed using the ''KDD CUP'99'' dataset as validated by the paired t-test. Meanwhile, Table 2 includes the performances of different

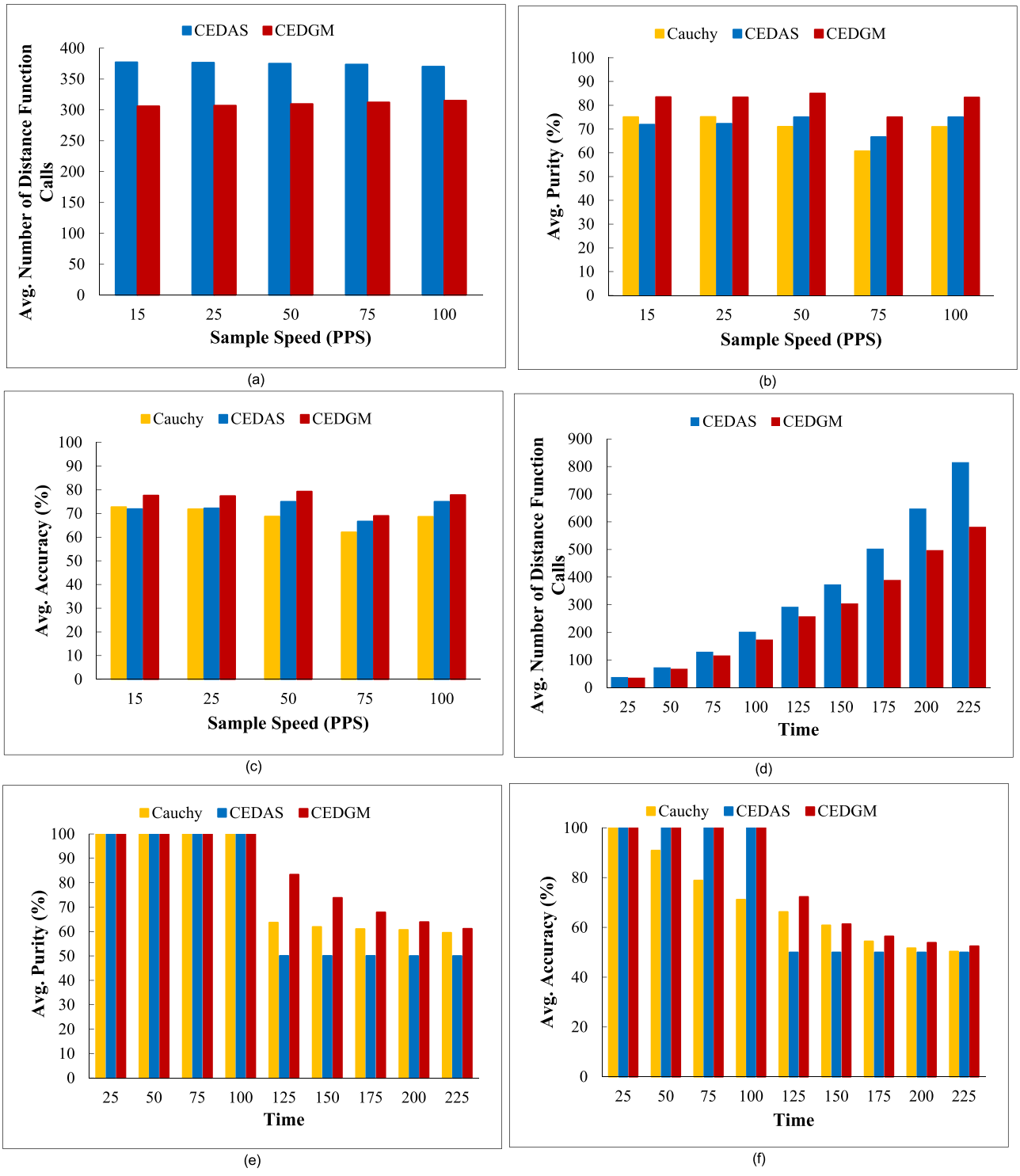

FIGURE 8. CEDGM, CEDAS and Cauchy performance comparisons with the sample speed and time using the spiral dataset.

state-of-the-art methods compared with the proposed method in terms of time using the "KDD CUP'99" dataset as validated by the paired t-test.

B. SPIRAL DAT

The comparisons between the CEDGM, CEDAS and Cauchy algorithms in terms of the spiral data stream set results are plotted in Fig. 8. The outcomes are calculated at different sample speeds and times, where the parameters of the proposed and other algorithms are *Decay* = 500 *samples*, *Radius* = 0.05, *Minimum Threshold* = 4 and *Grid Granularity* = 30. The average numbers of distance function calls are compared at different sample speeds and times on the 'spiral data' evolving data stream. When the sample speed

TABLE 3. Performances of different state-of-the-art methods compared with performance of the proposed method, i.e., CEDGM, in terms of the sample speed using the spiral dataset as validated by the paired t-test.

Sample Speed	Average Number of Distance Function Calls		Average Purity			Average Accuracy		
	CEDGM	CEDAS	CEDGM	CEDAS	Cauchy	CEDGM	CEDAS	Cauchy
15	**306.06**	377.39	**83.47**	71.89	75.05	**77.58**	71.87	72.73
25	**307.11**	376.77	**83.36**	72.25	75.11	**77.38**	72.22	71.83
50	**309.73**	375.21	**85.02**	75.03	70.99	**79.26**	74.99	68.75
75	**312.33**	373.67	**75.03**	66.66	60.69	**68.99**	66.66	62.06
100	**315.13**	370.22	**83.35**	75.02	70.93	**77.78**	75	68.65
t.test	-	**2.53E-05**	-	**0.00012**	**0.00098**	-	**0.00349**	**0.00228**

varies from 15 to 100 PPS, as presented in Fig. 8(a), and the 25 s period is used, as illustrated in Fig. 8(d), the CEDGM algorithm outperforms the CEDAS algorithm regarding the mean number of distance function calls. At sample speeds of 15 PPS and 100 PPS, the average numbers of distance function calls of the proposed algorithm are 306.06 and 315.13, respectively, compared with those of CEDAS of 377.39 and 370.22, respectively. At a period of 175 s, the CEDGM reaches 389.72, whereas the CEDAS reaches 503.18. We conclude that the CEDGM algorithm can yield a lower average number of distance function calls than the CEDAS algorithm due to the same reason of improving the average of distance function calls (Section 4.1).

The experimental results of the CEDGM, CEDAS and Cauchy algorithms in terms of the average clustering purity are depicted in Figs. 8(b) and (e). We test them on the same 'spiral' evolving data stream. The average purity, as defined in Eq. (1), is determined by sample speed and time. Our proposed algorithm has a higher purity than the CEDAS and Cauchy algorithms. At a speed of 15 PPS, the CEDGM algorithm achieves an average purity of 83.47%, whereas CEDAS reaches 71.89% and Cauchy reaches 75.05%. Regarding the average purity at periods from 25 s to 225 s, both algorithms achieve 100% from 25 s to 100 s; meanwhile, at a period of 225 s, the CEDGM reaches 61.15%, whereas the CEDAS reaches 50.02% and Cauchy reaches 59.52%. Thus, nearly every sample is appropriately allocated to the predominant clusters. We conclude that the CEDGM algorithm can obtain better average clustering purity than the CEDAS and Cauchy algorithms due to the same reason of improving the average clustering purity (Section 4.1).

Figs. 8(c) and (f) depict the average accuracy results of the CEDGM, CEDAS and Cauchy algorithms on the 'spiral' evolving data stream. The average accuracy, as defined in Eq. (2), is determined by the sample speed and time. When the sample speed varies from 15 to 100 PPS, as demonstrated in Fig. 8(c), our proposed algorithm exceeds the CEDAS and Cauchy algorithms in terms of the average accuracy. At a sample speed of 15 PPS, the average accuracy of the CEDGM algorithm is 77.58%, whereas that of CEDAS is 71.87% and that of Cauchy 72.73%. At a sample speed of 50 PPS, the clustering accuracy of the CEDGM is 79.26%, whereas that of the CEDAS is 74.99% and that of Cauchy 68.75%. The results of the 25 s period, as exhibited in Fig. 8(f), are also adopted. At a period ranging from 25 s to 225 s, both algorithms achieve 100% from 25 s to 100 s; meanwhile, for the period of 225 s, the CEDGM reaches 52.48%, whereas the CEDAS reaches 49.99% and Cauchy reaches 50.33%. This result confirms that the CEDGM algorithm has better average clustering accuracy than the CEDAS and Cauchy algorithms due to the same reason of improving the average clustering accuracy (Section 4.1). Table 3 includes the performances of different state-of-the-art methods compared with the proposed method in terms of sample speed using the "Spiral" dataset as validated by paired t-test. In addition, Table 4 includes the performances of different state-of-the-art methods compared with the proposed method in terms of time using the "Spiral" dataset as validated by the paired t-test.

C. DENSITY DATASET (DS)

A comparison between the CEDGM, CEDAS and Cauchy algorithms on the DS is displayed in Fig. 9. The outcomes are calculated at different sample speeds and times, where the parameters of the proposed and other algorithms are $Decay = 500\ samples$, $Radius = 0.05$, $Minimum\ Threshold = 4$ and $Grid\ Granularity = 30$. The CEDGM has a lower average number of distance function calls on this dataset. For example, when the sample speed varies from 5 to 25 PPS, as presented in Fig. 9(a), the average number of distance function calls of the CEDGM is less than 63, whereas that of the CEDAS exceeds 63. When the time varies from 25 s to 125 s, as illustrated in Fig. 9(d), the average number of distance function calls is constantly less for the CEDGM algorithm than for the CEDAS algorithm. Therefore, the CEDGM algorithm performs better in terms of the average number of distance function calls because when using the density grid-based method, the average number of distance function calls is lower compared with the CEDAS algorithm due to the same reason of improving the average number of distance function calls (Section 4.1).

Figs. 9(b) and (e) depict the average purity results of the CEDGM, CEDAS and Cauchy algorithms on the same DS.

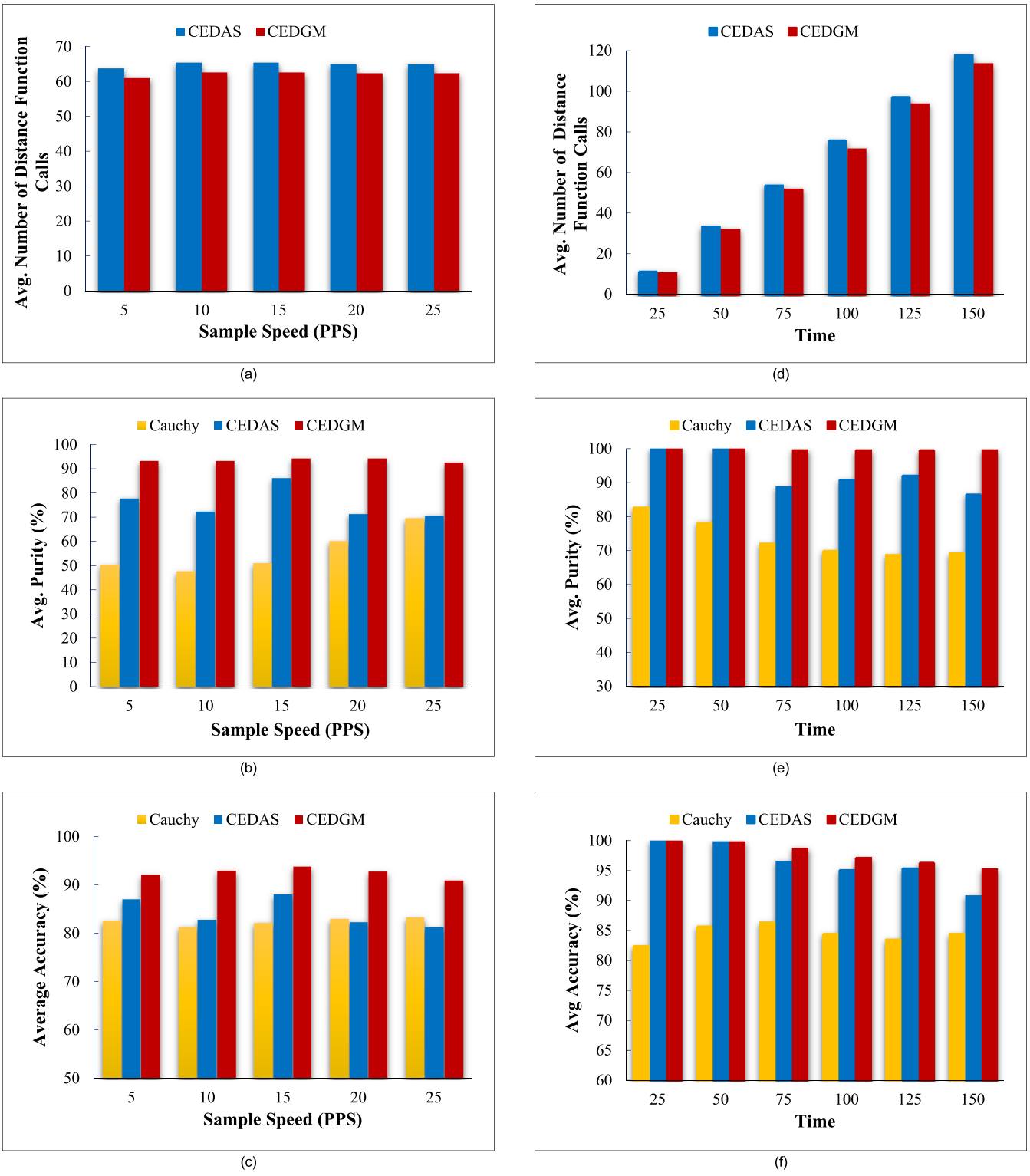

FIGURE 9. CEDGM, CEDAS and Cauchy performance comparisons with the sample speed and time using the DS.

The average purity, as defined in Eq. (1), is determined the sample speed and time. When the sample speed varies from 5 to 25 PPS, the average purity of the CEDGM is higher than 98%. When the time varies from 25 s to 150 s, the average purity of the CEDGM is higher than 99%. The CEDAS achieves 100% for the time from 25 s to 50 s but it is less than 93% when the time ranges from 75 s to 150 s and that of Cauchy is below 83%. We conclude that the CEDGM

TABLE 4. Performances of different state-of-the-art methods compared with performance of the proposed method, i.e., CEDGM, in terms of time using the spiral dataset as validated by the paired t-test.

Time	Average Number of Distance Function Calls		Average Purity			Average Accuracy		
	CEDGM	CEDAS	CEDGM	CEDAS	Cauchy	CEDGM	CEDAS	Cauchy
25	**35.96**	38.24	**100**	100	100	**100**	100	100
50	**68.29**	73.45	**100**	100	100	**100**	100	90.88
75	**116.73**	129.89	**100**	100	100	**100**	100	78.86
100	**174.02**	202.41	**100**	100	100	**100**	100	71.20
125	**258.05**	292.77	**83.36**	50.08	63.73	**72.25**	49.98	66.26
150	**304.65**	373.35	**73.85**	50.08	61.93	**61.37**	49.99	60.85
175	**389.72**	503.18	**67.90**	50.06	61.07	**56.41**	49.99	54.44
200	**497.66**	648.23	**63.93**	50.04	60.68	**53.88**	49.99	51.67
225	**581.93**	815.64	**61.15**	50.03	59.53	**52.49**	49.99	50.34
t.test	**-**	**0.02554**	**-**	**0.02621**	**0.03481**	**-**	**0.03618**	**0.04698**

TABLE 5. Performances of different state-of-the-art methods compared with performance of the proposed method, i.e., CEDGM, in terms of the sample speed using DS as validated by the paired t-test.

Sample Speed	Average Number of Distance Function Calls		Average Purity			Average Accuracy		
	CEDGM	CEDAS	CEDGM	CEDAS	Cauchy	CEDGM	CEDAS	Cauchy
5	**61.11**	63.79	**93.31**	77.92	50.46	**92.16**	87.03	82.71
10	**62.77**	65.53	**93.42**	72.44	47.85	**92.91**	82.82	81.4
15	**62.67**	65.43	**94.42**	86.22	51.06	**93.82**	88.08	82.21
20	**62.39**	65.11	**94.29**	71.53	60.25	**92.82**	82.42	83
25	**62.39**	65.11	**92.64**	70.88	69.78	**90.92**	81.33	83.28
t.test	**-**	**5.43E-09**	**-**	**0.00282**	**0.00085**	**-**	**0.00196**	**0.00016**

algorithm outperforms the CEDAS and Cauchy algorithms regarding average purity due to the same reason of improving the average purity (Section 4.1).

The experimental results for the average clustering accuracy of the CEDGM, CEDAS and Cauchy algorithms are demonstrated in Figs. 9(c) and (f). Here, we test them on the same 'DS.' The average accuracy, as defined in Eq. (2), is determined by the sample speed and time. The average clustering accuracy of the CEDGM is better than those of the CEDAS and Cauchy when the sample speed varies from 5 to 25 PPS. When the sample speeds are 5 and 10 PPS, the average accuracy of the CEDGM algorithm is 92.16% and 92.91, respectively, whereas those of the CEDAS algorithm are 87.03% and 82.82%, respectively, and those of Cauchy are 82.71% and 81.40%, respectively. When the sample speed is 25 PPS, the CEDGM achieves 90.92%, whereas the CEDAS yields 81.33% and Cauchy yields 83.28%. When the time varies from 25 s to 150 s, with the mean over 25 s, the average accuracy of both algorithms is 100% at 25 s. At a time of 150 s, the CEDGM achieves 95.32%, whereas the CEDAS achieves 90.86% and Cauchy achieves 84.55%. The CEDGM algorithm is based on the density grid of the enhanced clustering algorithm. When calculating the density of grid cells, the impact of the boundary data points on the grid is calculated rationally by calculating the impact coefficient of the added data points on the adjacent grid cells. Thus, the data points cannot be treated as noise points, and this condition will improve the clustering accuracy. Table 5 includes the performances of different state-of-the-art methods compared with the proposed method in terms of the sample speed using the "DS" dataset validated by paired t-test. Meanwhile, Table 6 includes the performances of different state-of-the-art methods compared with the proposed method in terms of time using the "DS" dataset as validated by the paired t-test.

D. HALF RING, IRIS, AND GALAXY

Figs. 10 and 11 illustrate the performances of different state-of-the-art methods compared to the performance of the proposed method, i.e., CEDGM, in terms of the average accuracy and average Normalized Mutual Information (NMI), respectively. The parameters of the proposed and other algorithms are *Decay* = 100, *Minimum Threshold* = 2 *data points,*

TABLE 6. Performances of different state-of-the-art methods compared with the performance of the proposed method, i.e., CEDGM, in terms of time using DS as validated by the paired t-test.

Time	Average number of Distance Function Calls		Average Purity			Average Accuracy		
	CEDGM	CEDAS	CEDGM	CEDAS	Cauchy	CEDGM	CEDAS	Cauchy
25	**11.14**	11.60	**100**	100	82.8	**100**	100	82.48
50	**32.42**	33.85	**99.8**	99.8	78.27	**99.85**	99.85	85.69
75	**52.08**	54.10	**99.7**	88.8	72.17	**98.77**	96.60	86.41
100	**71.8**	75.90	**99.62**	90.82	69.92	**97.22**	95.11	84.5
125	**93.72**	97.26	**99.51**	92.17	68.88	**96.33**	95.42	83.59
150	**113.57**	118.02	**99.57**	86.49	69.34	**95.33**	90.87	84.55
t.test	-	**0.00954**	-	**0.03149**	**8.40E-05**	-	**0.03387**	**3.05E-05**

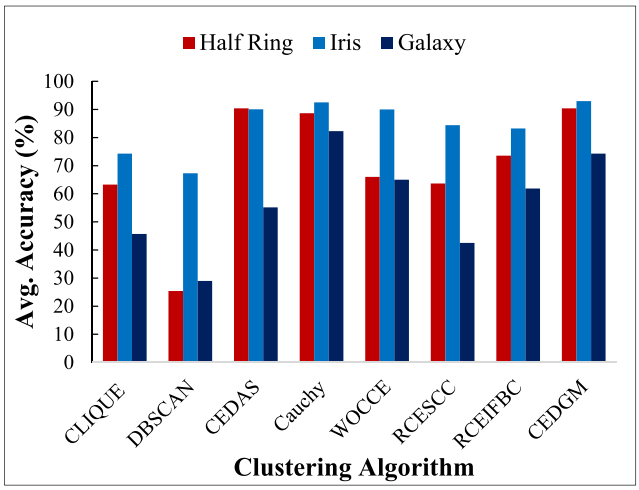

FIGURE 10. The performances of different methods in terms of clustering accuracy.

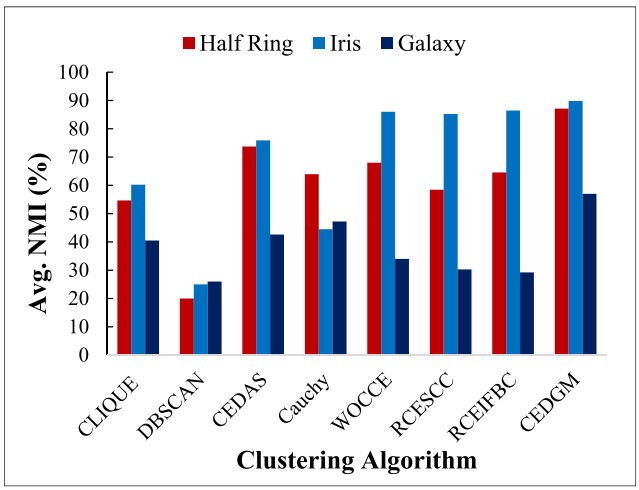

FIGURE 11. The performances of different methods in terms of the NMI.

Radius = 1, *Grid Granularity* = 30, *minPts* = 2 *data points, eps* = 1, *and number of interval* = 5. In the three datasets such as 'Half-Ring, Iris, and Galaxy', the proposed algorithm has better average clustering accuracy and average NMI than the other existing algorithms due to the same reason of improving the average clustering accuracy (Section 4.1). Table 7 summarizes the results of the average clustering accuracy, and Table 8 summarizes the results of the average NMI.

TABLE 7. Performances of different state-of-the-art methods compared to the performance of the proposed method, i.e., CEDGM, in terms of the clustering accuracy.

Clustering Algorithms	Datasets		
	Half Ring	Iris	Galaxy
CLIQUE	63.24	74.3	45.7
DBSCAN	25.4	67.3	29
CEDAS	90.41	90.06	55.16
Cauchy	88.65	92.53	72.28
WOCCE	66	90	65
RCESCC	63.64	84.41	42.5
RCEIFBC	73.54	83.26	61.89
CEDGM	**90.41**	**92.95**	**74.3**

TABLE 8. Performances of different state-of-the-art methods compared to the performance of the proposed method, i.e., CEDGM, in terms of the NMI.

Clustering Algorithms	Datasets		
	Half Ring	Iris	Galaxy
CLIQUE	54.65	60.21	40.5
DBSCAN	20	25	26
CEDAS	73.73	75.86	42.57
Cauchy	63.94	44.46	47.2
WOCCE	68	86	34
RCESCC	58.43	85.2	30.25
RCEIFBC	64.54	86.45	29.21
CEDGM	**87.13**	**89.85**	**56.98**

V. CONCLUSION

An enhanced method for clustering evolving data streams is introduced in this paper using a density grid-based method

called the CEDGM. The main idea of the CEDGM algorithm is to use a density grid-based method to improve the clustering quality. This technique is used to reduce the number of distance function calls and handling outliers. The CEDGM can handle evolving data streams online. This algorithm is compared with a familiar technique in terms of the average number of distance function calls, the average purity and the average accuracy. The proposed algorithm is particularly efficient if data streams are constantly evolving.

The CEDGM is tested using different data streams and is confirmed to be capable of accurately discovering anomalies within specified periods. It further demonstrates its capability to generate high-quality clusters in practical network attacks in the KDDCUP'99 data stream. Extensive evaluations of different synthetic and real datasets using different quality metrics shows that the clusters generated in the CEDGM are pure and more accurate compared to similar existing clustering algorithms due to summarizing the data points into a grid and generating CMC. Nevertheless, it has a low average number of distance function calls due to the grid-based method and does not depend on the number of data objects, but rather it depends on the number of cells in the quantized space in each dimension. In summary, the CEDGM is an accurate technique with a lower average number of distance function calls across different data stream speeds and times. However, one of the limitations of the CEDGM algorithm in high dimensional data is the non-effectiveness of the grid in reducing memory consumption, especially with high dimensional data with sparse nature. Our future work will focus on the improvement of the memory consumption of the CEDGM algorithm in high dimensional datasets with sparse nature.

REFERENCES

[1] D. V. Dimitrov, "Medical Internet of Things and big data in healthcare," *Healthcare Informat. Res.*, vol. 22, no. 3, pp. 156–163, 2016.
[2] M. Z. Ge, H. Bangui, and B. Buhnova, "Big data for Internet of Things: A survey," *Future Gener. Comput. Syst.*, vol. 87, pp. 601–614, Oct. 2018.
[3] Y. Yang, X. Zheng, W. Guo, X. Liu, and V. Chang, "Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system," *Inf. Sci.*, vol. 479, pp. 567–592, Apr. 2019.
[4] A. Oussous, F. Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 30, no. 4, pp. 431–448, Oct. 2018.
[5] M. A. Azzawi, R. Hassan, and K. A. A. Bakar, "A review on Internet of Things (IoT) in healthcare," *Int. J. Appl. Eng. Res.*, vol. 11, no. 20, pp. 10216–10221, 2016.
[6] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Int. Conf. Very Large Data Bases*, vol. 29, 2003, pp. 81–92.
[7] Y. Qin, Q. Z. Sheng, N. J. G. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, "When things matter: A survey on data-centric Internet of Things," *J. Netw. Comput. Appl.*, vol. 64, pp. 137–153, Apr. 2016.
[8] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.
[9] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. Maasberg, and K.-K. R. Choo, "Multimedia big data computing and Internet of Things applications: A taxonomy and process model," *J. Netw. Comput. Appl.*, vol. 124, pp. 169–195, Dec. 2018.
[10] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A. V. Vasilakos, "The role of big data analytics in Internet of Things," *Comput. Netw.*, vol. 129, pp. 459–471, Dec. 2017.
[11] M. B. Mohamad Noor and W. H. Hassan, "Current research on Internet of Things (IoT) security: A survey," *Comput. Netw.*, vol. 148, pp. 283–294, Jan. 2019.
[12] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, The Netherlands: Elsevier, 2011.
[13] J. Shao, Y. Tan, L. Gao, Q. Yang, C. Plant, and I. Assent, "Synchronization-based clustering on evolving data stream," *Inf. Sci.*, vol. 501, pp. 573–587, Oct. 2019.
[14] P. P. Ray, "A survey on Internet of Things architectures," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 30, no. 3, pp. 291–319, 2018.
[15] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. De Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, p. 13, 2013.
[16] X. Yan, M. Razeghi-Jahromi, A. Homaifar, B. A. Erol, A. Girma, and E. Tunstel, "A novel streaming data clustering algorithm based on fitness proportionate sharing," *IEEE Access*, vol. 7, pp. 184985–185000, 2019.
[17] A. Amini, H. Saboohi, T. Ying Wah, and T. Herawan, "A fast density-based clustering algorithm for real-time Internet of Things stream," *Sci. World J.*, vol. 2014, pp. 1–11, Jan. 2014.
[18] M. Černý, "Narrow big data in a stream: Computational limitations and regression," *Inf. Sci.*, vol. 486, pp. 379–392, Jun. 2019.
[19] M. Tareq, E. A. Sundararajan, and M. Mohd, "Online clustering of evolving data stream based on adaptive Chebychev distance," in *Proc. 281st Int. Conf. IIER*, Mar. 2020, pp. 41–46.
[20] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big IoT data analytics: Architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.
[21] A. Al-Shammari, R. Zhou, M. Naseriparsaa, and C. Liu, "An effective density-based clustering and dynamic maintenance framework for evolving medical data streams," *Int. J. Med. Informat.*, vol. 126, pp. 176–186, Jun. 2019.
[22] C. C. Aggarwal, N. Ashish, and A. Sheth, "The Internet of Things: A survey from the data-centric perspective," in *Managing and Mining Sensor Data*. Boston, MA, USA: Springer, 2013, pp. 383–428.
[23] R. Lacuesta, G. Palacios-Navarro, C. Cetina, L. Peñalver, and J. Lloret, "Internet of Things: Where to be is to trust," *EURASIP J. Wireless Commun. Netw.*, vol. 2012, no. 1, p. 203, Dec. 2012.
[24] Y. N. Malek, A. Kharbouch, H. E. Khoukhi, M. Bakhouya, V. D. Florio, D. E. Ouadghiri, S. Latre, and C. Blondia, "On the use of IoT and big data technologies for real-time monitoring and data processing," *Procedia Comput. Sci.*, vol. 113, pp. 429–434, 2017.
[25] M. Mousavi and A. A. Bakar, "Improved density based algorithm for data stream clustering," *Jurnal Teknologi*, vol. 77, no. 18, pp. 64–92, 2015.
[26] M. Mousavi, A. A. Bakar, and M. Vakilian, "Data stream clustering algorithms: A review," *Int. J. Adv. Soft Comput. Appl.*, vol. 7, no. 3, p. 13, 2015.
[27] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: A clustering algorithm for data streams," *J. Exp. Algorithmics*, vol. 17, May 2012, Art. no. 2.4.
[28] C. C. Aggarwal and S. Y. Philip, "A survey of synopsis construction in data streams," in *Data Streams*. Boston, MA, USA: Springer, 2007, pp. 169–207.
[29] C. C. Aggarwal, *Data Streams: Models and Algorithms*. Boston, MA, USA: Springer, 2007.
[30] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 685–694.
[31] C. Fahy and S. Yang, "Dynamic feature selection for clustering high dimensional data streams," *IEEE Access*, vol. 7, pp. 127128–127140, 2019.
[32] N. Ohadi, A. Kamandi, M. Shabankhah, S. M. Fatemi, S. M. Hosseini, and A. Mahmoudi, "SW-DBSCAN: A grid-based DBSCAN algorithm for large datasets," in *Proc. 6th Int. Conf. Web Res. (ICWR)*, Apr. 2020, pp. 139–145.
[33] M. Aljibawi, M. Z. A. Nazri, and Z. Othman, "A survey on clustering density based data stream algorithms," *Int. J. Eng. Technol.*, vol. 7, no. 36, pp. 147–153, 2018.
[34] P. Huang, X. Li, and B. Yuan, "A parallel GPU-based approach to clustering very fast data streams," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 23–32.
[35] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann, "Time-hierarchical clustering and visualization of weather forecast ensembles," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 831–840, Jan. 2017.

[36] H. Parmar and S. Saket, "Overview of clustering algorithm for weather data," *Int. J. Adv. Res. Innov. Ideas Educ.*, vol. 3, no. 6, pp. 1023–1029, 2017.

[37] J. Sui, Z. Liu, A. Jung, L. Liu, and X. Li, "Dynamic clustering scheme for evolving data streams based on improved STRAP," *IEEE Access*, vol. 6, pp. 46157–46166, 2018.

[38] P. Novianti, D. Setyorini, and U. Rafflesia, "K-means cluster analysis in earthquake epicenter clustering," *Int. J. Adv. Intell. Inform.*, vol. 3, no. 2, pp. 81–89, 2017.

[39] R. Kamat and R. Kamath, "Earthquake cluster analysis: K-means approach," *J. Chem. Pharmaceutical Sci.*, vol. 10, no. 1, pp. 250–253, 2017.

[40] E. Al Nuaimi, H. Al Neyadi, N. Mohamed, and J. Al-Jaroodi, "Applications of big data to smart cities," *J. Internet Services Appl.*, vol. 6, no. 1, p. 25, 2015.

[41] K. Drab and M. Daszykowski, "Clustering in analytical chemistry," *J. AOAC Int.*, vol. 97, no. 1, pp. 29–38, Jan. 2014.

[42] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowl. Inf. Syst.*, vol. 45, no. 3, pp. 535–569, Dec. 2015.

[43] W. Liu and J. OuYang, "Clustering algorithm for high dimensional data stream over sliding windows," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Nov. 2011, pp. 1537–1542.

[44] L. Chen, L.-J. Zou, and L. Tu, "A clustering algorithm for multiple data streams based on spectral component similarity," *Inf. Sci.*, vol. 183, no. 1, pp. 35–47, Jan. 2012.

[45] T. Li and C. Ding, *Data Clustering: Algorithms and Applications*. Boca Raton, FL, USA: CRC Press, 2013.

[46] A. Forestiero, C. Pizzuti, and G. Spezzano, "A single pass algorithm for clustering evolving data streams based on swarm intelligence," *Data Mining Knowl. Discovery*, vol. 26, no. 1, pp. 1–26, Jan. 2013.

[47] R. Hyde, P. Angelov, and A. R. MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," *Inf. Sci.*, vols. 382–383, pp. 96–114, Mar. 2017.

[48] A. Amini, H. Saboohi, T. Herawan, and T. Y. Wah, "MuDi-stream: A multi density clustering algorithm for evolving data stream," *J. Netw. Comput. Appl.*, vol. 59, pp. 370–385, Jan. 2016.

[49] B. Aubaidan, M. Mohd, and M. Albared, "Comparative study of K-means and K-means++ clustering algorithms on crime domain," *J. Comput. Sci.*, vol. 10, no. 7, pp. 1197–1206, Jul. 2014.

[50] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*, vol. 2. London, U.K.: Chapman & Hall, 2013.

[51] J. Youn, J. Shim, and S.-G. Lee, "Efficient data stream clustering with sliding windows based on locality-sensitive hashing," *IEEE Access*, vol. 6, pp. 63757–63776, 2018.

[52] N. Arbin, N. S. Suhaimi, N. Z. Mokhtar, and Z. Othman, "Comparative analysis between K-Means and K-Medoids for statistical clustering," in *Proc. 3rd Int. Conf. Artif. Intell., Modeling Simulation (AIMS)*, Dec. 2015, pp. 117–121.

[53] A. I. Hammouri and S. Abdullah, "Comparison between compactness and connectedness criteria in data clustering," *Int. J. Data Anal. Techn. Strategies*, vol. 8, no. 4, pp. 281–295, 2016.

[54] D. B. Setyohadi, A. A. Bakar, and Z. A. Othman, "Optimization overlap clustering based on the hybrid rough discernibility concept and rough K-means," *Intell. Data Anal.*, vol. 19, no. 4, pp. 795–823, Jul. 2015.

[55] M. Mohd *et al.*, "Optimal initial centroid in K-means for crime topic," *J. Theor. Appl. Inf. Technol.*, vol. 45, no. 1, pp. 19–26, 2012.

[56] M. Ahmed, "Buffer-based online clustering for evolving data stream," *Inf. Sci.*, vol. 489, pp. 113–135, Jul. 2019.

[57] M. D. Parmar, W. Pang, D. Hao, J. Jiang, W. Liupu, L. Wang, and Y. Zhou, "FREDPC: A feasible residual error-based density peak clustering algorithm with the fragment merging strategy," *IEEE Access*, vol. 7, pp. 89789–89804, 2019.

[58] M. Parmar, D. Wang, X. Zhang, A.-H. Tan, C. Miao, J. Jiang, and Y. Zhou, "REDPC: A residual error-based density peak clustering algorithm," *Neurocomputing*, vol. 348, pp. 82–96, Jul. 2019.

[59] D. Brown, A. Japa, and Y. Shi, "A fast density-grid based clustering method," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2019, pp. 0048–0054.

[60] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Density-based clustering of data streams at multiple resolutions," *ACM Trans. Knowl. Discovery From Data*, vol. 3, no. 3, p. 14, 2009.

[61] J. J. Jung, "Semantic preprocessing for mining sensor streams from heterogeneous environments," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 6107–6111, May 2011.

[62] J.-Y. Chen and H.-H. He, "A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data," *Inf. Sci.*, vol. 345, pp. 271–293, Jun. 2016.

[63] D. Brown and Y. Shi, "A distributed density-grid clustering algorithm for multi-dimensional data," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2020, pp. 0001–0008.

[64] I. Škrjanc, S. Ozawa, T. Ban, and D. Dovžan, "Large-scale cyber attacks monitoring using evolving cauchy possibilistic clustering," *Appl. Soft Comput.*, vol. 62, pp. 592–601, Jan. 2018.

[65] B. Wu and B. M. Wilamowski, "A fast density and grid based clustering method for data with arbitrary shapes and noise," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1620–1628, Aug. 2017.

[66] L. Wang and H. Li, "Clustering algorithm based on grid and density for data stream," *AIP Conf.*, vol. 1839, no. 1, 2017, Art. no. 020202.

[67] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, vol. 96, no. 34, pp. 226–231.

[68] T. Ali, S. Asghar, and N. A. Sajid, "Critical analysis of DBSCAN variations," in *Proc. Int. Conf. Inf. Emerg. Technol.*, Jun. 2010, pp. 1–6.

[69] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2006, pp. 328–339.

[70] C. Ruiz, E. Menasalvas, and M. Spiliopoulou, "C-denstream: Using domain knowledge on a data stream," in *Proc. Int. Conf. Discovery Sci.* Berlin, Germany: Springer, 2009, pp. 287–301.

[71] L.-X. Liu, Y.-F. Guo, J. Kang, and H. Huang, "A three-step clustering algorithm over an evolving data stream," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, vol. 1, Nov. 2009, pp. 160–164.

[72] J. Ren and R. Ma, "Density-based data streams clustering over sliding windows," in *Proc. 6th Int. Conf. Fuzzy Syst. Knowl. Discovery*, vol. 5, 2009, pp. 248–252.

[73] M. Hassani, P. Spaus, M. M. Gaber, and T. Seidl, "Density-based projected clustering of data streams," in *Proc. Int. Conf. Scalable Uncertainty Manage.* Berlin, Germany: Springer, 2012, pp. 311–324.

[74] N. Su, J. Liu, C. Yan, T. Liu, and X. An, "An arbitrary shape clustering algorithm over variable density data streams," *J. Algorithms Comput. Technol.*, vol. 11, no. 1, pp. 93–99, Mar. 2017.

[75] C. Isaksson, M. H. Dunham, and M. Hahsler, "SOStream: Self organizing density-based clustering over data stream," in *Proc. Int. Workshop Mach. Learn. Data Mining Pattern Recognit.* Berlin, Germany: Springer, 2012, pp. 264–278.

[76] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 133–142.

[77] J. Yang, W. Zhu, J. Zhang, and Y. Yang, "Data stream clustering algorithm based on active grid density," in *Proc. 5th Int. Conf. Internet Comput. Sci. Eng.*, Nov. 2010, pp. 97–101.

[78] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1998, pp. 94–105.

[79] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 685–694.

[80] M. Mojarad, S. Nejatian, H. Parvin, and M. Mohammadpoor, "A fuzzy clustering ensemble based on cluster clustering and iterative fusion of base clusters," *Int. J. Speech Technol.*, vol. 49, no. 7, pp. 2567–2581, Jul. 2019.

[81] H. Alizadeh, M. Yousefnezhad, and B. M. Bidgoli, "Wisdom of crowds cluster ensemble," *Intell. Data Anal.*, vol. 19, no. 3, pp. 485–503, Jun. 2015.

[82] M. Mojarad, H. Parvin, S. Nejatian, and V. Rezaie, "Consensus function based on clusters clustering and iterative fusion of base clusters," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 27, no. 01, pp. 97–120, Feb. 2019.

[83] L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," *ACM Trans. Knowl. Discovery Data*, vol. 3, no. 3, p. 12, 2009.

[84] C. Newman, S. Hettich, and C. Merz, "UCI repository of Mach Learn databases," Dept. Inf. Comput. Sci., Univ. California Irvine, Irvine, CA, USA, Tech. Rep., 1998.

[85] M. Hahsler and M. Bolaños, "Clustering data streams based on shared density between micro-clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1449–1461, Jun. 2016.

[86] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *Proc. 16th IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2004, pp. 576–584.
[87] C. F. Eick, B. Vaezian, D. Jiang, and J. Wang, "Discovery of interesting regions in spatial data sets using supervised clustering," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Berlin, Germany: Springer, 2006, pp. 127–138.
[88] R. Hyde and P. Angelov, "A new online clustering approach for data in arbitrary shaped clusters," in *Proc. IEEE 2nd Int. Conf. Cybern. (CYBCONF)*, Jun. 2015, pp. 228–233.
[89] C. Fahy, S. Yang, and M. Gongora, "Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2215–2228, Jun. 2019.

MUSTAFA TAREQ received the B.Sc. degree in computer science from Al Turath University College, Iraq, in 2012, and the M.Sc. degree in network technology from Universiti Kebangsaan Malaysia (UKM), Malaysia, in 2016, where he is currently pursuing the Ph.D. degree in computer science with the Center for Software Technology and Management, Faculty of Information Science and Technology. His current research interests include the Internet of Things, data mining, clustering algorithms, and evolving data streams.

ELANKOVAN A. SUNDARARAJAN (Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in computer science from Universiti Kebangsaan Malaysia (UKM), in 1995 and 1997, respectively, and the Ph.D. degree in computer science from The University of Melbourne, Australia, in 2008. He is an Associate Professor with the UKM. His current research interests include high performance computing, big data, cloud computing, and computer networks.

MASNIZAH MOHD (Member, IEEE) received the bachelor's and master's degrees in information science from Universiti Kebangsaan Malaysia (UKM), in 1999 and 2002, respectively, and the Ph.D. degree in computer and information sciences from the University of Strathclyde, in 2010. She is an Associate Professor with the UKM. Her current research interests include information retrieval and natural language processing.

NOR SAMSIAH SANI received the bachelor's degree in information technology from Universiti Tenaga Nasional (UNITEN) and the Ph.D. degree from The University of Sheffield, in 2016. She is currently a Senior Lecturer with the Center of Artificial Intelligence Technology (CAIT), Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia. Her works have been published in reputable journals such as *MATCH-Communications in Mathematical and in Computer Chemistry* and the *Journal of Chemical Information and Modeling*. Her research interests include the visualization of complex chemical data sets, machine learning, evolutionary algorithms, and data mining chemoinformatics.