Received May 14, 2019, accepted May 30, 2019, date of publication June 10, 2019, date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922162

A Novel Streaming Data Clustering Algorithm Based on Fitness Proportionate Sharing

XUYANG YAN[1], MOHAMMAD RAZEGHI-JAHROMI[2], ABDOLLAH HOMAIFAR[1], (Member, IEEE), BERAT A. EROL[1], ABENEZER GIRMA[1], AND EDWARD TUNSTEL[3], (Fellow, IEEE)

[1]North Carolina Agricultural and Technical State University, Greensboro, NC 27401, USA
[2]Energy and Automation Department, ABB Corporate Research United States (USCRC), Raleigh, NC 27606, USA
[3]The Systems Department, United Technologies Research Center, East Hartford, CT 06118, USA

Corresponding author: Abdollah Homaifar (homaifar@ncat.edu)

This work was supported in part by the Air Force Research Laboratory and the Office of the Secretary of Defense (OSD) under Agreement FA8750-15-2-0116, and in part by the National Institute of Aerospace's Langley Distinguished Professor Program under Grant C16-2B00-NCAT.

ABSTRACT As an unsupervised learning technique, clustering can effectively capture the patterns in a data stream based on similarities among the data. Traditional data stream clustering algorithms either heavily depend on some prior knowledge or predefined parameters while the characteristics of real-time data are considered unknown. Besides, the user-specified threshold is used to overcome the effect of outliers and noises, which significantly affects the clustering performance. The overlap among clusters is another major challenge for the existing stream clustering methods. These constraints strongly limit their real-time applications. In this paper, a two-phase stream clustering algorithm based on fitness proportionate sharing is proposed. It handles streaming data when prior knowledge is not available and maps the clustering problem into a multimodal optimization problem. It introduces a density-based objective function and adopts the fitness proportionate sharing strategy to perform a more effective search for the cluster centers. To capture the dynamic characteristics of streaming data, a recursive formula for the lower bound of the density function is derived, and a summary of historical data is established for the proposed algorithm. The proposed algorithm is applied to different data sets, and a comprehensive comparison between the proposed algorithm and five well-known data stream clustering algorithms in the literature is provided. Results show comparable or better performance relative to five popular data stream clustering algorithms. A scalability analysis of the proposed streaming clustering method is presented in this paper as well.

INDEX TERMS Data streams, clustering, unsupervised learning, data mining.

I. INTRODUCTION

In recent years, data streams have attracted the interests of many researchers in different fields, such as cyber security [1], sensor network management [2], data mining [3], and other data science related studies. Unlike the traditional static data set, a data stream is defined as a large volume of data in a continuous flow that provides input to a system; and thus, the number of data samples from this flow is infinite. The characteristics of data are not fixed and different patterns will appear as time passes. Based on the characteristic types, there are a wide diversity of applications developed for streaming data, the most applicable ones are online shopping recommendation systems [4], traffic network monitoring [5], network intrusion detection [6], stock market prediction [7], and many real-time autonomous systems.

Streaming data is highly advocated and utilized in many real-time applications due to its capability of tracking the information of real-time processes continuously. However, it

The associate editor coordinating the review of this manuscript and approving it for publication was Pascual Martinez-Gomez.

still presents several major challenges [8]–[10]. At first, the data is only accessed once and it is irretrievable in the future, which requires an accurate interpretation of the data. Secondly, due to the high transferring speed of streaming data, an efficient processing technique is preferred to reduce the processing time. At the same time, a large memory space needs to be considered because of the infinite amount of data in the stream, which may not be feasible for real-time applications. The high dimensionality and randomness of streaming data also increase the difficulty of extracting information from streaming data. Hence, systematic approaches should be developed to conduct a comprehensive analysis of streaming data.

Clustering is one of the most widely used techniques to partition data spaces based on data similarity and to discover patterns from data streams. It is an effective tool to capture the nature of the data by separating the data based on similarities, which can greatly help users to understand the concept behind the information. However, traditional clustering algorithms are usually run offline, and thus, lack the capability to capture the characteristics of the data stream over time. For example, the number of clusters varies as new data samples arrive, and the centers of clusters evolve by time. In addition to this, the structure of the data stream is unknown while the majority of existing clustering algorithms depend on prior information. For instance, the k-means based streaming clustering algorithm in [11] needs a predefined value of k, and a proper initialization of those k centers significantly affect the clustering performance. The presence of outliers poses another challenge on the cluster analysis of the streaming data [9].

In this paper, a dynamic clustering algorithm based on fitness proportionate sharing is discussed. Instead of relying on the availability of any prior information about the data, such as cluster number or cluster radius, the proposed dynamic fitness proportionate sharing clustering algorithm utilizes the density value and a novel sharing strategy to drive the search for an optimal number of the clusters. The density values of samples measure the attractiveness of each sample in its neighborhood and provide the priority order of samples as the potential cluster centers. Samples with the higher density values are more likely to be selected as the pseudo centers -rather than randomly searching for cluster centers, which avoids unnecessary explorations for cluster centers. The fitness proportionate sharing (FPS) [12] strategy encourages the exploitation of the potential clusters until a set of good pseudo clusters are discovered. The cluster expansion in [13], [14] is used to eliminate repeated clusters and find an optimal number of clusters. Also, by using our approach, there is no need for any user-specified parameter that needs to be finely tuned to achieve a high quality of clustering results. The Gaussian kernel function has been widely used in many offline density-based clustering methods and it can effectively address the issues of outliers or noises. Considering the dynamics of streaming data, we have derived and presented a novel recursive lower bound of the Gaussian kernel function that allows the proposed algorithm to track the patterns of data continuously.

The remainder of this paper is organized as follows: Section 2 presents the existing streaming data clustering methods and technical challenges. Also, the motivations and contributions of this work are summarized in Section 2. Section 3 briefly reviews the offline FPS-clustering method and discusses the proposed dynamic FPS-clustering algorithm for data streams. A recursive lower bound of the Gaussian kernel function, and a related Lemma to handle outliers in the streaming data cluster analysis are proposed in Section 3. A detailed mathematical proof associated with the dynamic FPS-clustering algorithm (DFPS-clustering) is presented in the Appendix. Simulation results and performance evaluation of the proposed algorithm are presented in Section 4. Finally, Section 5 concludes the paper and future work is discussed.

II. BACKGROUND

In this section, a literature review on the state-of-the-art streaming clustering methods is presented and a summary of the technical challenges is discussed. Also, the motivations and contributions of this work are summarized in this section.

A. STATE-OF-THE-ART METHODS REVIEW

Recently, there have been various approaches developed for streaming data clustering as presented in [9], [10], [15]–[22]. As one of the earliest streaming clustering methods, [11] proposed an offline k-means clustering algorithm by dividing the data stream into sequential partitions and identifying clusters with the k-means algorithm within each partition. Then, the clusters in each partition are saved until it exceeds the memory threshold; so, a reclustering procedure can be applied to those clusters to obtain a small set of clusters. With its reclustering procedure, the memory limitation is easily resolved. However, due to its unresponsiveness over time, this clustering scheme has an obvious limitation of recognizing the evolving characteristics of the given data.

An alternate solution is proposed in [23] by applying the concept of *"Moving Window"* and introducing time-index parameters to decay the effect of historical data as time changes. Based on this concept, a *"two-phase scheme"* is introduced in the streaming data clustering procedure and the CluStream algorithm is developed [16]. It consists of the continuous partitioning of raw data into data chunks, and the offline clustering for each data chunk from the summary of historical data. Based on this work, many extensions are proposed to improve the existing offline clustering algorithms for the online processing. In [24], Guha and Sudipto applied the weighted k-means in CluStream. Zhou *et al.* proposed the *Sliding-window clustering* (SWClustering) procedure based on the *Exponential Histogram of Cluster Feature* (EHCF) in [25]. Unlike CluStream, a number of EHCFs are developed from the evolving data stream and the outdated record can be effectively eliminated from the EHCF. A bounded memory space is guaranteed by the predefined maximum number of EHCFs. A k-means clustering procedure is performed whenever the clustering request arrives. Nevertheless, k-means based streaming clustering algorithms have a strong dependence on the prior knowledge of data (e.g., cluster number or radius) and are incapable of recognizing clusters with arbitrary shapes or sizes, which limits their use in practical applications.

With the "*two-phase scheme*" discussed previously, many density-based stream clustering algorithms are proposed to overcome the challenges of discovering clusters with arbitrary sizes and identifying outliers. As an extension of the *"Density-based spatial clustering of applications with noise"* (DBSCAN) algorithm, DenStream [26] forms clusters based on the reachability of the micro-clusters by checking their distance between each other with a specified threshold. Like CluStream in [16], there is an online maintenance of the micro-clusters list and offline reclustering of micro-clusters. It can effectively handle the noise and outliers in the stream by expanding the concepts of micro-clusters as core micro-clusters, potential micro-clusters, and outlier micro-clusters. A variation of the DenStream clustering algorithm, named *SDStream*, is proposed by Ren and Ma in [27]. It applies the "*sliding window*" to the DenStream method and stores micro-clusters in the form of EHCFs. Similar to the SWClustering algorithm, outdated samples are effectively eliminated and only the most recent records are used in the cluster analysis. Since it does not use the entire data stream, it suffers the risks of losing important historical information. Two recent one-pass density-based streaming clustering algorithms, namely HCMstream [28] and BEstream [29], are proposed to apply the discard-after-cluster concept to record the entire history of the streaming data without causing the memory overflow. As the first density-based projected clustering method, HDDstream [30] was proposed to address the high-dimensionality of data streams. The *dimension preference vector* is introduced to overcome the effect of noises and the concept of *micro-clusters* is used to handle the outliers. Hahsler and Bolanos proposed another density-based clustering algorithm named DBStream in [20] to capture the density between micro-clusters. It improves the clustering performance by considering the density between micro-clusters especially when micro-clusters are overlapped with each other. An Enhanced Density-based Data Stream (EDDS) clustering technique is proposed in [31] to improve the performance of the DBSCAN-based stream clustering methods with comparable time complexity. To handle mixed data streams, Chen and He developed a fast density-based data stream clustering algorithm to cluster data streams with mixed data types [32]. They proposed a novel distance measure for the mixed data stream and introduced a micro-cluster characteristic vector to track the mixed data stream continuously. In [33], Gong *et al.* extended the *"Density Peak Clustering"* algorithm from [34], and proposed a novel streaming clustering algorithm named "*EDMStream*" by exploring the evolution of the density mountains in the streaming data. To enhance the computational efficiency, it adopts the *Dependency Tree* (DP-Tree) structure to abstract the density mountains and some filtering schemes are employed to further reduce the computational costs. Nonetheless, the prior knowledge requirement is still a key factor to influence the performance of a density-based clustering algorithm for data streams. Besides, the high computational complexity is another major concern for most of these techniques.

A number of the *Grid-based* streaming clustering algorithms such as D-Stream [19], DENGRIS [35], DD-Stream [36], EXCC [37], and MuDi-Stream [38], have been proposed recently as well. The D-Stream algorithm partitions the data space into a number of grid cells and uses them to compute clusters by merging adjacent dense cells. The density of the grid cell depends on the number of points within the cell and a time-dependent decaying factor is introduced in the density computation to capture the dynamic changes in streaming data. As an extension of the D-Stream, DENGRIS [35] primarily focuses on the most recent data by applying the *sliding window*. DD-Stream adopts the same clustering procedure from the D-Stream while considering the sporadic grids in the cluster analysis, which can significantly enhance the clustering quality. In [37], an exclusive and complete clustering (ExCC) algorithm based on the grid synopsis is proposed to cluster data streams with both the numeric and categorical features. The hold-queue strategy is used to address the presence of noises in the data stream with a compromise to more time and memory complexity [39]. MuDi-Stream [38] is designed to enhance the cluster analysis of data streams with multi-density clusters. However, the time complexity of MuDi-Stream increases dramatically when the dimensionality of the data stream grows, which makes it less appropriate to cluster high-dimensional data streams. In regard to the high computational complexity of a grid-based clustering algorithm, extensions like MR-Stream [40] and LeaDen-Stream [41] are developed to provide a finer grid spacing by applying some effective techniques. On the other hand, the resolution of the grid space is still an open problem when very little information about the data is available.

B. CHALLENGES, MOTIVATIONS AND CONTRIBUTIONS

As stated previously, k-means streaming clustering algorithms not only have a strong dependency on the prior knowledge about data, but also are sensitive to outliers and noise. Also, they are not capable of detecting clusters with arbitrary shapes and sizes, which strongly limits their applications. As an alternative of the k-mean based streaming clustering approaches, the density-based clustering methods utilize the cluster radius and local sample density distribution to overcome the dependency on the cluster number. With the density distribution of the samples, clusters with arbitrary shapes and sizes can be simply discovered. The density threshold is used to overcome the effect of outliers and some time-decaying parameters need to be given in advance to capture the change of the data stream over time. However, the initialization of those user-specified parameters needs to be carefully selected, and some tuning procedures are required to ensure the clustering performance. The high time complexity of the density-based clustering algorithms makes them less applicable in real-time data stream analysis. Unlike density-based stream clustering approaches, the grid-based stream clustering methods are faster because no distance calculations are required in the clustering stage. However, the partition of the data space requires a good prior understanding of the data to achieve a high clustering accuracy and some user-specified parameters are necessary. Moreover, the time complexity of the grid-based stream clustering methods become intractable in the cluster analysis of high-dimensional data. The summary of the existing challenges are outlined as follows:

1) Majority of the existing streaming clustering methods either need a predefined cluster number or a cluster radius while very little, or no prior information about the real-world data streams is available. *How to deal with the unknown streaming data without user-specified parameters* is still a challenging and ongoing research topic.
2) A poor clustering performance can be caused by the existence of the highly overlapped clusters and outliers. To identify outliers, some user-specified parameters need to be given in advance. *The separation of highly overlapped clusters* and *the outlier detection without user-specified thresholds* are another two challenging tasks in clustering streaming data.
3) A huge amount of memory space is required to store useful information about the data stream. For example, CluStream encoded the data information into a set of micro-clusters with different pyramidal time frames, which takes a relatively large memory space. *How to save the memory space with a more concise summary of data* is the major concern in the real-time streaming data cluster analysis.

In order to solve these challenges, Angelov proposed a *Recursive Density Estimation* (RDE) [42] formula to model the density distribution of the data stream over time without any user-specified parameters. Based on RDE, several one-pass stream clustering approaches are proposed in [42]–[44] to automatically extract the cluster structure from streaming data, and it can effectively overcome the above challenges. However, these methods are developed to handle data streams where the individual sample is arriving continuously while most of the real-world data streams arrive as a sequence of data chunks, which may not be appropriate in the analysis of real-world data streams. Also, they adopted the simple density function to approximate the density distribution of the data, which may not be able to capture the uncertainty in the data. Since the Gaussian Kernel function is widely used in many offline clustering algorithms such as DENCLUE [45], Mean-Shift based clustering [46] and DOE-AND-SCA [47] algorithms, the efficacy of the Gaussian kernel function has been proved in handling the large datasets with noise and outliers when the number of clusters is unknown in advance. However, there is no recursive formula for the Gaussian kernel function in the literature and majority of the existing density-based streaming clustering algorithms utilize the simple density function. Hence, in this paper, we proposed a recursive lower bound of the Gaussian kernel function to approximate the density distribution of the data stream and developed a novel dynamic clustering algorithm to automatically extract the cluster structure from scratch. The main contributions of this paper are summarized as follows:

1) Due to the success of the Gaussian kernel function in many offline clustering algorithms, we proposed a recursive lower bound formula for the *Gaussian kernel function* to estimate the density value of samples in the data stream over time. It is the first work that derives the recursive lower bound of the Gaussian kernel and a related mathematical proof is presented in the Appendix. It is capable of capturing the density evolution of the samples in the stream without using all historical data samples, which reduces the computational costs significantly. Also, there is no need to store all historical samples in the density evaluation of newly arrived data.
2) Based on the proposed recursive lower bound of the data density, we extended the offline FPS-clustering algorithm and proposed a two-phase dynamic clustering procedure to automatically extract data structure from the very beginning. It *does not require* the user-specified radius and cluster numbers in advance. On the contrary, it evolves the cluster structure from the data itself. Besides, the fitness proportionate sharing strategy is adopted to guide the search of the potential cluster centers by treating it as a multi-modal optimization problem. It can effectively enhance the cluster analysis of the streaming data, especially when there are many overlapped clusters.
3) With the recursive lower bound of the Gaussian kernel function, we applied the three-sigma principle in the outlier detection procedure and developed a related Lemma to discriminate the outliers from novel clusters. Unlike most of the state-of-the-art methods, no user-specified threshold is required to distinguish the outliers.

III. PROPOSED METHODOLOGY

As mentioned earlier, the existing streaming clustering algorithms are highly restricted by the unbounded memory complexity, dynamic characteristics and prior knowledge on streaming data. In order to overcome these constraints, a dynamic clustering algorithm referred to as DFPS-clustering is proposed in this section. First of all, due to the success of the Gaussian Kernel function in many offline density-based clustering techniques, a novel recursive lower bound of the Gaussian kernel function is derived to capture the density evolution of the data stream over time. Based on the recursive lower bound, the offline FPS-clustering is applied within each arriving data chunk by considering the historical samples. In this way, a temporary cluster summary of each data chunk can be obtained and an online update of the global cluster summary can be performed to track the change of clusters in the data stream. At the same time, the three-sigma principle is used to distinguish the outliers from the novel clusters. The details are discussed in the following subsections.

A. OFFLINE FPS-CLUSTERING ALGORITHM

Unlike the most often used offline clustering algorithms, the FPS-clustering algorithm is developed to handle a data set without any prior knowledge, i.e., cluster number or radius. It is a density-based clustering algorithm and it maps the clustering algorithm into a multi-modal optimization problem. Considering the density function as an objective function (fitness), the cluster centers are defined as those peaks that reach the local maximum of the density distribution. With the Gaussian kernel type of density function used in [48]–[50], the fitness of each data sample is evaluated and it is defined as follows:

$$f\left(x^i\right) = \sum_{m=1}^{N} \left(e^{-\frac{\|x^i - x^m\|}{\beta}}\right)^{\gamma}, \quad i \in [1, N]. \tag{1}$$

The normalization parameter β denotes the variance of data, and γ represents the scaling parameter that approximates the shape of clusters. N is the total number of samples and the norm here refers to the Euclidean distance between sample x^i and sample x^m. A candidate set is established from the whole data set, and FPS-clustering will evaluate the potential for every point to be a cluster center based on their density value. Then, a search for candidate cluster centers is implemented by ranking individuals in the candidate set and the rank of individuals follows the descending order. The first individual in the rank is considered as a potential cluster center. Fitness proportionate sharing [51] is performed on individuals within the niche radius of the current potential cluster center so that their fitness values are scaled down, which provides a good opportunity for other local peaks to be searched in the next competition. The niche here refers to the neighboring region of each local peak and the idea of fitness proportionate sharing is motivated from the procedure of population migration. For example, the city with the best living conditions can attract many people to come; then, the resources of the city are shared by a large population, which decreases the average resource usage of each individual and makes the city become unpleasant to stay in. Due to a depletion of resources, people will then search for the second most pleasant city and migrate to it. In an analogy to this procedure, all of the potential cluster centers can be explored with fitness proportionate sharing as detailed in [52]. After the search of potential cluster centers, a dynamic niche expansion [13], [53] is employed to remove redundant clusters, and obtain an optimal set of clusters.

B. RECURSIVE LOWER BOUND OF THE DENSITY ESTIMATION

As previously discussed, the proposed streaming clustering framework is developed based on the density-based objective function and it is necessary to obtain a good estimation of the density distribution of the data stream dynamically. Hence, a recursive lower bound of the Gaussian kernel function [54] is derived here to track the sample density distribution in the data stream. Data chunk (C_t) [55] is used to partition the data stream (D_t) into several partitions evenly, and each partition is processed based on their arriving time for the continuous flow and high rate of the data stream. By partitioning, each data chunk C_t becomes a subset of the data stream and D_t is the collection of data chunks until the arrival of C_t, which can be expressed as:

$$D_t = \bigcup_{t_i=1}^{t} \{C_{t_i}\}, \quad C_t = \bigcup_{i=1}^{M} \{x^i(t)\}. \tag{2}$$

The parameter t denotes the time index of the arriving data chunk and M is the size of the data chunk. For each data chunk, the FPS-clustering algorithm is implemented to locate all clusters offline while the summary of those clusters is kept online, which is updated with time. To use the DFPS-clustering to analyze streaming data, both a recursive formula for the lower bound of density estimation and a summary of clustering history are required. Specifically, the summary of the clustering history reflects the change in patterns of the data stream and it consists of: global mean, global variance, stabilization parameter, cluster centers, the radius, and density values of the centers. It is represented as $S_t = \{\mu_t, \beta_t, \gamma_t, \left[Z_t^j, f\left(Z_t^j\right), R_t^j\right], j = 1, 2, \ldots, |Z_t|\}$. The cardinality notation $|*|$ is adopted to represent the number of elements in the set. The summary of the clusters is continuously updated by performing the merge between new clusters and historical clusters, which is described in Algorithm 1. Also, the global statistical information of the data stream, including mean, variance, and stabilization parameter, is recursively updated with Equations (4), (5), and Algorithm 2. Thus, there is no need to save the entire data stream, which effectively overcomes the memory limitations. A recursive formula is proposed based on Equation (1) and the cluster summary, which is expressed as:

$$\hat{f}\left(x^i(t)\right) = \sum_{j=1}^{|Z_{t-1}|} \left[\left(e^{-\frac{d_{ij}^2}{\beta_t}}\right)^{\gamma_t} \times f\left(Z_{t-1}^j\right)\right] + \sum_{m_2=1}^{|C_t|} \left(e^{-\frac{-d_{im_2}^2}{\beta_t}}\right)^{\gamma_t}, \tag{3}$$

where d_{ij} is the Euclidean distance from the newly arriving sample x^i to the center of the previous cluster Z_{t-1}^j and $f\left(Z_{t-1}^j\right)$ is the density value of the previous cluster center j. Assume $|P_{t-1}^j|$ refers to the number of samples that belong to the j_{th} clusters at $t-1$, the value of $f\left(Z_{t-1}^j\right)$ can be obtained from the following equations:

$$f\left(Z_{t-1}^{j}\right)=\sum_{k=1}^{|P_{t-1}^{j}|}\left(e^{-\frac{d_{kj}^{2}}{\beta_{t-1}}}\right)^{\gamma_{t-1}}, \quad \beta_t=\frac{t-1}{t}\times\beta_{t-1}+\frac{\beta_{C_t}}{t}. \tag{4}$$

where

$$\beta_{C_t}=\frac{\sum_{i=1}^{M}\left(x^i-\mu_t\right)^2}{M-1}, \quad \mu_t=\frac{(t-1)\times\mu_{t-1}}{t}+\frac{\mu_{C_t}}{t}, \tag{5}$$

The normalization parameter β_t denotes the variance of the data stream at t, and β_{C_t} is the variance of the current data chunk C_t, where μ_t denotes the mean of the data stream at t and μ_{C_t} is the mean of samples in the current data chunk C_t. γ_t is a stabilization parameter that determines the shape of the cluster and t is the time index of the data chunk. Considering the property of Gaussian distribution, the three-sigma principle is adopted in Lemma 1 below to remove outliers. The sample with a density value that falls outside of the three-sigma distance away from the average density values of the existing clusters has a probability of 0.00135 to be considered as the cluster center, which can be used to remove the effect of outliers in discovering new potential clusters. Based on Equation (3) and the three-sigma principle, a lower bound condition to check the existence of new clusters by the mean $\mu\left(f(Z_{t-1})\right)$ and variance $\sigma\left(f(Z_{t-1})\right)$ of the density values of the previous cluster centers is derived as:

Lemma 1: If $\hat{f}(x^i(t)) \geq |\mu\left(f(Z_{t-1})\right) - 3\sigma\left(f(Z_{t-1})\right)|$. A new cluster will appear and the cluster expansion procedure will be triggered to check for possible cluster expansions.

A detailed mathematical proof for Equation (3) and Lemma 1 is presented in the Appendix. Considering the uncertainty of streaming data, most of the density-based clustering algorithms utilize a Gaussian kernel to approximate the true density distribution of the data stream, and identify clusters in the data streams. For clusters that are identified with a Gaussian kernel, we exploit the symmetric property of the distribution that can simplify the density calculation of streaming data. Therefore, for the mathematical proof, a Gaussian density distribution with symmetric property is considered in the Appendix.

C. DYNAMIC FITNESS PROPORTIONATE SHARING (DFPS)-CLUSTERING ALGORITHM

As described above, the recursive lower bound of the Gaussian kernel function is derived to approximate the density of samples over time. Also, a related Lemma is defined to discriminate the outliers from the novel clusters. With Equation (3), the fitness of samples in the data stream is approximated recursively. Then, the FPS-clustering algorithm is implemented to locate all clusters offline while the summary of those clusters is kept online, which is updated with time. According to Lemma 1, a possible expansion between the previous clusters and the new clusters is performed to remove all redundant clusters. Then, the summary of clusters is updated to capture the characteristics of the data stream continuously, as detailed in Algorithm 1.

Algorithm 1 Dynamic Fitness Proportionate Sharing Clustering (DFPS-Clustering)

```
1: Data chunk: $C_t, t = 1, 2 \ldots, T_{in}$
2: The summary of clusters at $t$
3: $S_t = \{[Z_t^j, f\left(Z_t^j\right), R_t^j], j = 1, 2, \ldots, |Z_t|\}$
4: The size of data chunk: $Buffersize$
5: for $t = 1 : T_{in}$ do
6:     Input the data chunk $C_t$
7:     Form a sampling data set $S$ by randomly selecting $N_s$ samples from the data chunk $C_t$
8:     if $t = 1$ then
9:         Compute the statistical information of the data stream: $\{\mu_t = \mu_{C_1}, \beta_1 = \beta_{C_t}, \gamma_1 = \gamma_{C_1}\}$
10:        Conduct the fitness proportionate sharing clustering in the first data chunk $C_1$ with Equation (1)
11:        Create the summary of the identified clusters $S_1$
12:    else
13:        Update the statistical information of the data stream ($\beta_t$, $\mu_t$, and $\gamma_t$)
14:        Conduct the fitness proportionate sharing clustering based on the fitness values that are obtained from Equation (3) and $S_{t-1}$
15:        Check the estimated density values of new identified potential clusters with the boundary condition in Lemma 1 to compute new clusters
16:        Perform possible cluster expansions among the previous clusters and new clusters
17:        Update the cluster summary $S_t$ with the latest statistical information and clusters that are obtained from cluster expansions
18:    end if
19: end for
```

Algorithm 1 suggests clusters can be evolved with the change of the data stream continuously and the summary of the previous clusters can be updated simultaneously. This is implemented by applying *offline fitness proportionate sharing clustering* for each data chunk with an online estimation of density. Similar to the FPS-clustering algorithm, the cluster expansions between the previous and new clusters are performed to update the cluster summary, which can effectively capture the association between the past and new patterns of streaming data. The density estimation that uses Equation (3) can greatly reduce the computational complexity without losing any of the characteristics of the previous data, which can speed up the clustering procedure.

D. PARAMETER TUNING AND SELECTION

As shown in the density function, Equation (1), the parameter γ [46], [48] can be considered as a scaling factor to model the stable density distribution, which has previously

Algorithm 2 Dynamic Correlation Comparison Algorithm (DCCA)

```
1: Data chunk: $C_t, t = 1, 2 \ldots, T_{in}$
2: Stabilization parameter of the current data stream: $\gamma_t$
3: Stabilization parameter of the previous data stream:$\gamma_{t-1}$
4: Stabilization parameter of the first data chunk $C_1$:$\gamma_0$
5: Variance of the current data stream: $\beta_t$
6: Variance of the previous data stream: $\beta_{t-1}$
7: Initial variance of the data stream: $\beta_0$
8: if $t = 1$ then
9:     Obtain $\gamma_0$ from the first data chunk $C_1$ using CCA [48]
10: else
11:     Compute the new stabilization parameter: $\gamma_t = \gamma_0 \times \frac{\beta_t}{\beta_0}$
12: end if
```

been defined as a stabilization parameter. In [46], the "*Correlation Comparison Algorithm*" (CCA) [48] is used to obtain a good estimate of γ in the mean-shift based clustering procedure. It is implemented by continuously increasing the value of γ step by step until there are no further significant improvements to the approximation of density distribution. It is similar to the procedure of hypothesis testing and a threshold value of R_t in [48] is used here. According to the confidence level of 95%, a threshold of 0.97 is widely used in the CCA to provide an acceptable estimation of the density distribution. However, due to the dynamic characteristics of streaming data, the value of γ also needs to be updated with the arrival of new data to approximate the density distribution of data as time varies. In order to use CCA in the proposed Algorithm 1, modifications are made on the CCA, and we introduced a dynamic version of CCA that is summarized in Algorithm 2.

According to Algorithm 2, the value of the stabilization parameter γ is also updated as the variance of the data stream varies. Instead of keeping γ fixed, the relative value between β and γ is kept constant and is referred to as bandwidth BW, which is defined to reduce the overlap between clusters.

IV. SIMULATION RESULTS AND DISCUSSIONS

To evaluate the performance of the proposed algorithm, seven datasets, three synthetic created by the authors and four others from [56] and [57] are used. For simplicity, they are labeled as $D1$, $D2$, $D3$, $D4$, $D5$, $D6$, and $D7$, respectively. Also, two performance metrics, Rand Index [58] and Kappa Index [59], are used to provide a good interpretation of its performance. Since the proposed approach is developed based on the two-phase clustering framework, five representatives from the existing well-known two-phase algorithms, including STREAM [17], [24], CluStream [16], [19], D-stream [19], HDDStream [30], and DBStream [20], are applied and a comparison between them and DFPS-clustering is discussed to show the effectiveness of the proposed algorithm in this section. Due to the popularity of these five clustering algorithms, there are several public software packages written in R programming language for statistical computing, which makes it easier for analysis and testing. Besides, the complexity analysis of the proposed clustering approach and the other five baseline methods is discussed in this section. The scalability analysis of the proposed DFPS-clustering method is presented in this section as well. By default, all of the datasets are partitioned into several data chunks to generate streaming data. The size of the data chunk used for different data is selected based on the criterion that the number of samples in the data set is divisible by the size of the data chunk. According to this criterion, different sizes of data chunks are used in this paper and the details are discussed below.

A. SYNTHETIC DATASETS

Three synthetic continuous datasets are generated to validate the efficacy of the proposed streaming clustering method, and their characteristics are summarized in Table 1. The detailed descriptions of these two synthetic datasets are discussed below:

- $D1$ has 18, 000 samples that are randomly generated around nine clusters and it is a 2D synthetic dataset. Clusters in $D1$ are separate from one another and each cluster has an equal number of samples. It is divided into six data chunks with an equal size of 3, 000. By partitioning, $D1$ is processed in the streaming fashion and the corresponding clusters at different sampling time indices are displayed respectively in FIGURE 1.
- Similar to $D1$, $D2$ is generated by distributing samples within each cluster with a Gaussian distribution. It has 20, 000 instances that lie around ten clusters with unequal sizes and each instance has two features. However, clusters in dataset $D2$ both have high overlapped coverage and uneven sizes. It is divided into five data chunks with an equal size of 4, 000 and the performance of DFPS-clustering is presented in FIGURE 2.
- $D3$ has 50, 000 samples that are distributed around ten non-overlapped clusters and each instance consists of ten features. There are 1, 000 outliers that are randomly distributed in the entire data space and the remaining 49, 000 data samples lie around those ten clusters. With a default chunk size of 1000, it is divided into 50 data chunks and then processed sequentially as a stream of data.

B. REAL DATASETS

Considering the complexity and uncertainty of real data, the proposed algorithm is applied to several real datasets. The summary of all real datasets is presented in Table 1. Also, the last two columns of Table 1 indicate the presence of the overlapped clusters and outliers for each dataset, respectively. The symbol T asserts true and symbol F denotes the false. The details of those four real datasets are summarized as follows:

TABLE 1. Datasets descriptions in terms of number of instances, features, actual clusters, predicted number of clusters, the presence of overlapped clusters and outliers.

Data sets	Instances	Features	Clusters	Predicted Clusters	Overlapped Clusters	Presence of Outliers
D1 (Synthetic Data)	18, 000	2	9	9	F	F
D2 (Synthetic Data)	20, 000	2	10	10	T	F
D3 (Synthetic Data)	50, 000	10	10	10	F	T
D4 (Iris Data) [56]	150	4	3	3	T	F
D5 (Electricity Grid) [57]	10, 000	11	2	2	F	T
D6 (Forest Cover Type) [56]	100, 000	10	7	7	T	T
D7 (Network Intrusion Data) [56]	200, 000	34	23	23	T	T

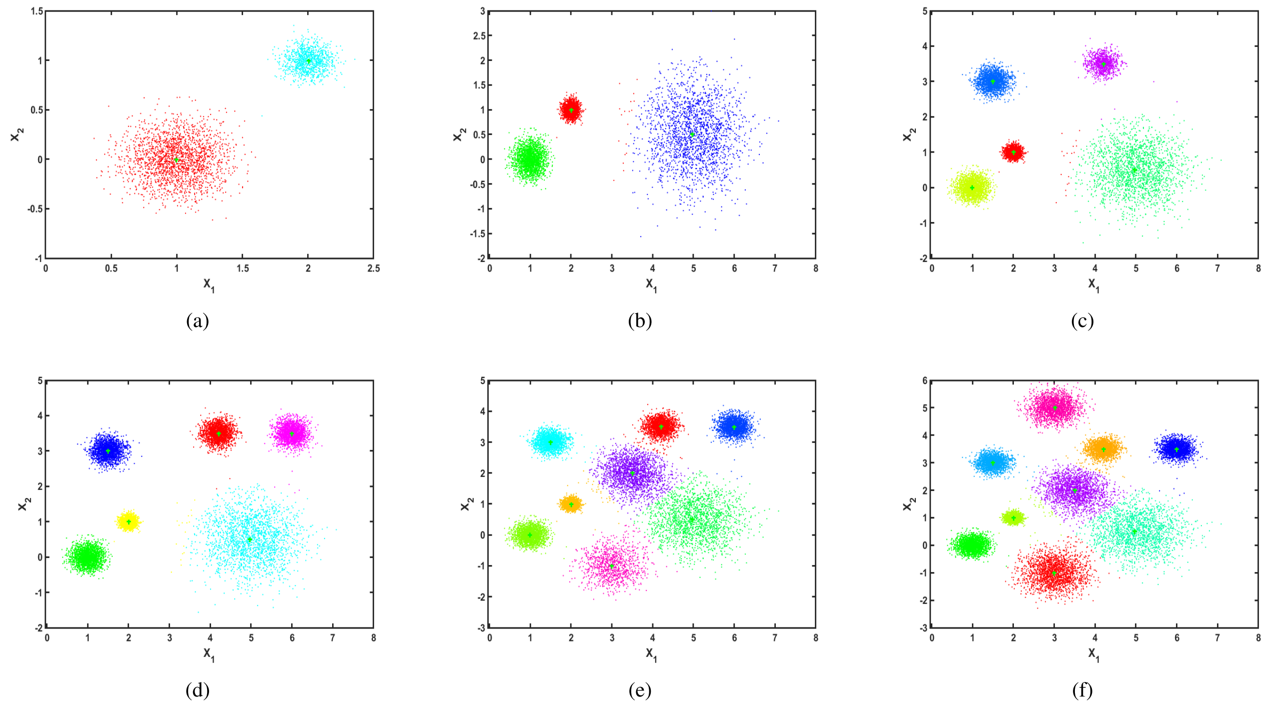


FIGURE 1. Simulation results for the synthetic dataset *D*1 (a): clusters at *t* = 1, (b): clusters at *t* = 2, (c): clusters at *t* = 3, (d): clusters at *t* = 4, (e): clusters at *t* = 5 and (f): clusters at *t* = 6.

- *D*4 is from seismic data provided by IRIS [56] and there are two highly overlapped clusters in the IRIS data set. In the simulation, it is equally divided into three data chunks, and each chunk is processed sequentially as a stream of data.
- *D*5 is collected from the stability analysis of the decentralized smart grid [56], [57]. There are two clusters in dataset *D*5, and it is partitioned into *ten* consecutive chunks with a size of 1000.
- *D*6 is a subset of the Forest Cover Type Dataset [56] and the first 100, 000 instances with ten numeric features are used here. It has seven ground truth clusters, and clusters are heavily overlapped with each other. Due to the high overlap among clusters, the cluster analysis of dataset *D*6 is very challenging. With the default experimental setting in [16], [19], [20], [24], [30], it is partitioned into 100 sequential chunks with an equal size of 1, 000.
- *D*7 is selected from the *Knowledge Discovery in Database(KDD) Cup Network Intrusion* data [56], and it is widely used in the simulations of many clustering algorithms. Regarding the large memory space, dataset *D*7 only consists of the first 200, 000 samples from the KDD Cup training dataset and all of its attributes are normalized into the interval: [0, 1]. It is divided into 200 data chunks, and each data chunk has an equal size of 1, 000, which keeps the same experimental setting in [16], [19], [20], [24], [30]. There are twenty-two different types of network attacks and one normal state in *D*7, which results in a total number of twenty-three clusters.

C. PERFORMANCE METRICS

To provide a comprehensive evaluation of the performance of the proposed algorithm, two statistical score functions known as *Rand Index* and *Kappa Index* from [58], [59] are used, respectively. The Kappa Index is also named "Adjusted Rand Index (ARI)" and it is widely used in the evaluation of many clustering methods [60]. These two parameters are based on the confusion matrix that reflects the consistency

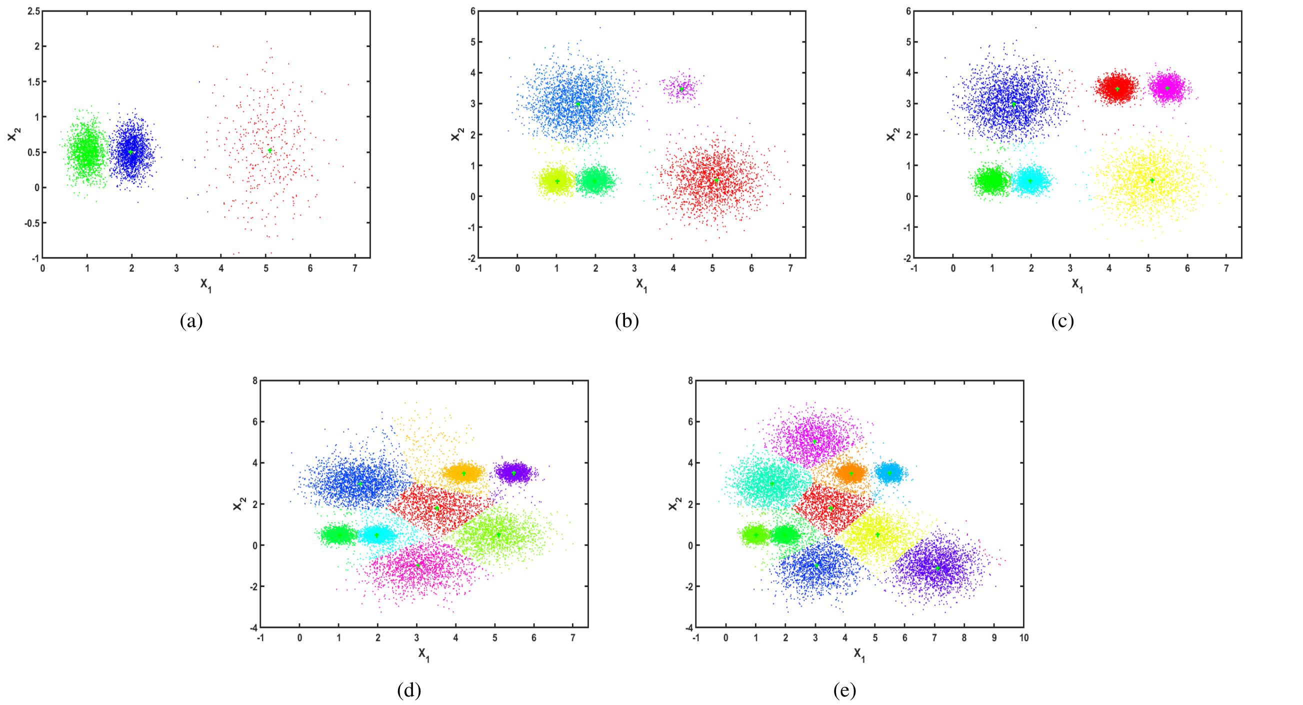


FIGURE 2. Simulation results for the synthetic dataset *D*2 (a): clusters at $t = 1$, (b): clusters at $t = 2$, (c): clusters at $t = 3$, (d): clusters at $t = 4$ and (e): clusters at $t = 5$.

TABLE 2. The comparison of *RI* between STREAM [17], [24], CluStream [16], [19], D-Stream [19], HDDStream [30] and DFPS-clustering.

Data sets	STREAM	CluStream	D-Stream	HDDStream	DBStream	DFPS-clustering
D1	0.9578	0.9667	0.9757	0.9732	0.9772	0.9885
D2	0.8314	0.8522	0.8952	0.8654	0.8874	0.9012
D3	0.9345	0.9454	0.9448	0.9656	0.9678	0.9745
D4	0.9089	0.9206	0.9153	0.9142	0.9067	0.9282
D5	0.8265	0.8312	0.8378	0.8947	0.8974	0.9280
D6	0.5987	0.6243	0.6128	0.6267	0.6278	0.6495
D7	0.8624	0.8848	0.9012	0.9034	0.9174	0.9211

TABLE 3. The comparison of KI between STREAM, CluStream, D-Stream, HDDStream, DBStream and DFPS-clustering.

Data sets	STREAM	CluStream	D-Stream	HDDStream	DBStream	DFPS-clustering
D1	0.9278	0.9357	0.9557	0.9589	0.9601	0.9782
D2	0.8156	0.8367	0.8612	0.8754	0.8597	0.8998
D3	0.9114	0.9241	0.9268	0.9358	0.9389	0.9495
D4	0.8873	0.8923	0.8894	0.8992	0.8967	0.9015
D5	0.7856	0.8024	0.8065	0.8873	0.8995	0.9192
D6	0.0604	0.0856	0.1042	0.1075	0.1120	0.1132
D7	0.8283	0.8474	0.8612	0.8781	0.8942	0.9004

between the original data clusters and the predicted results. Assume N to be the total number of data samples in the data space, and N_{ii} to be the diagonal element of the confusion matrix that defines the number of samples that are assigned to cluster i that originally belonged to cluster i. The number of samples belonging to the original cluster i can be obtained by finding the sum of each row in the confusion matrix: $N_{row_i} = \sum_j N_{ij}$. The number of samples that are classified into cluster j is the sum of each column in the confusion matrix: $N_{col_j} = \sum_i N_{ij}$. The performance indices are defined as:

1) Rand Index (RI)

$$RI = \frac{\sum_i N_{ii}}{N}. \tag{6}$$

2) Kappa Index (KI)

$$KI = \frac{N \times \sum_i N_{ii} - \sum_i N_{row_i} \times N_{col_i}}{N^2 - \sum_i N_{row_i} \times N_{col_i}}. \tag{7}$$

Based on Equations (6) and (7), the performance indices of the proposed algorithm for both synthetic and real data sets are summarized in Tables 2 and 3. The values of *RI*

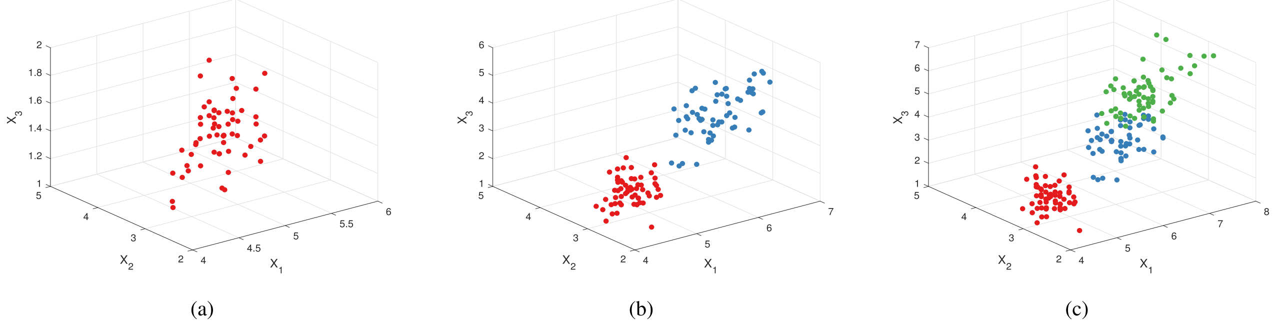


FIGURE 3. Simulation results for IRIS dataset $D4$ (a): cluster at $t = 1$, (b): clusters at $t = 2$ and (c): clusters at $t = 3$.

and KI are calculated at the arrival of the final data chunk and it includes samples from all data chunks, which reflects the change of patterns in the streaming data. The results for the STREAM, CluStream, D-Stream, HDDStream, and DBStream algorithms based on RI and KI indices are also presented in Tables 2 and 3, respectively. All of the data sets are simulated 30 times and the average values of RI and KI are presented in Tables 2 and 3.

D. RESULTS DISCUSSIONS

As shown in Table 1, the proposed algorithm can successfully capture all natural clusters in all eight data sets at the arrival of the last data chunk. For $D1$, Tables 2 and 3 demonstrate that the proposed method achieves a higher clustering accuracy than the remaining five compared existing methods. The clustering results for each data chunk in $D1$ are presented in FIGURE 1 and it shows the capability of the proposed method in tracking the change of clusters over time. In $D2$, FIGURE 2 shows that all of the overlapped clusters with uneven density values are successfully discovered in each data chunk and a final set of nine overlapped clusters are obtained after the arrival of the fifth data chunk. From Tables 2 and 3, the proposed algorithm provides a better performance than the other five clustering algorithms for $D2$ in terms of higher values of RI and KI, which indicates that the efficacy of the proposed algorithm in clustering data streams with highly overlapped clusters. In $D3$, the robustness of the proposed DFPS-clustering approach is tested in the presence of outliers and Table 1 demonstrates that all ten synthetic clusters with outliers can be discovered from the proposed DFPS-clustering approach. According to Tables 2 and 3, it is observed that the proposed approach still achieves a higher clustering accuracy than the other five compared methods, which implies the robustness of the proposed method to outliers. For real data sets, the performance of DFPS-clustering on the IRIS dataset is plotted in FIGURE 3 and the sampling time $t = 1, 2, 3$ represents the arriving time of the data chunks C_1, C_2, and C_3. As shown in FIGURE 3, the proposed algorithm can not only capture the change of clusters for streaming data in the time series, but also can distinguish the overlapped clusters. As mentioned before, $D5$ is a high-dimensional real data set that has two clusters and there are some outliers in $D5$. Compared with the other five state-of-the-art methods, Tables 2 and 3 reveal that the proposed algorithm can effectively identify two ground truth clusters in $D5$ with better clustering accuracy, which shows its effectiveness on handling the high data dimensionality and outliers. As for $D6$, it is a real data set that has both outliers and overlapped clusters. Although the proposed streaming clustering algorithm can successfully discover all seven clusters in the data stream, the accuracy is less comparable to other simulated data sets. However, compared with the other five methods, the proposed method still demonstrates better or comparable performance. As shown in Table 1, the proposed algorithm successfully discovers all of the clusters in $D7$. Furthermore, a comparable and even better performance of the proposed algorithm over the other five clustering algorithms for the real data with outliers and the overlapped clusters can be seen in Tables 2 and 3.

E. COMPLEXITY ANALYSIS

Like most of the existing clustering algorithms, one important measure to evaluate the performance of different clustering algorithms is the computational cost, especially for streaming data. Assume the chunk size is M and a percentage of l samples in the chunk is selected randomly to form the sampling set, then the number of distance calculations for the sampling set to discover N_s potential clusters is $(M + |Z_{t-1}|) \times (l \times M) \approx \left(l \times M^2\right)$, where $|Z_{t-1}|$ is the number of the previous clusters. The computational cost of cluster expansions can be divided into two parts: cluster expansions within the data chunk and clusters expansions between the previous and new clusters. For cluster expansions within the data chunk, the number of distance calculations is $N_m \times N_p \times (M + |Z_{t-1}|) \approx \left(N_m \times N_p \times M\right)$. The term N_m refers to the number of merges among the potential clusters and N_p denotes the number of pseudo points that are generated for an individual merge. On the other hand, the cluster expansions between previous $|Z_{t-1}|$ clusters and $|Z_{C_t}|$ new clusters take $|Z_{C_t}| \times N_p \times (M + |Z_{t-1}|) \approx \left(|Z_{C_t}| \times N_p \times M\right)$ distance calculations for the online update of cluster summary. Since each sample consists of d variables, the total counts of distance calculations within each data chunk become $(M + |Z_{t-1}|) \times \left(l \times M + |Z_{C_t}| \times N_p + N_m \times N_p\right) \times d \approx \left(l \times M^2 \times d\right)$.

TABLE 4. Computational costs of STREAM, CluStream, D-Stream, HDDStream, DBStream and DFPS-clustering.

Algorithms	Number of distance calculations(per Chunk)
STREAM [17], [24]	$O(M \times K_1 \times iterations)$
CluStream [16], [19]	$O(K_2 \times q \times iteration_1 + q \times M \times iteration_2)$
D-Stream [19]	$O(1)$
HDDStream [30]	$O(p_1 \times d + p_2)$
DBStream [20]	$O(M \times \log 2t_{gap} + t_{gap} \log t_{gap})$
DFPS-clustering	$O(M^2 \times d)$

The major computational cost is associated with calculating the density values for the sampling set and it is proportional to the square of chunk size M and the sample dimensionality d. Thus, the upper bound of the computational costs for the proposed method can be expressed as $O(M^2d)$. In order to evaluate the effectiveness of the proposed algorithm, the computational costs of the other five clustering algorithms in terms of Big O notation [61] are described in Table 4.

According to Table 4, the computational cost of STREAM is $K_1 \times M \times iterations$ when the number of clusters K_1 is selected properly. The term *iterations* here refers to the number of iterations it takes to converge, and a large number of iterations is required when the initial cluster centers are not selected properly. CluStream has a computational cost that is based on the number of micro-clusters: q and macro-clusters: K_2. The micro-clusters here refer to those small clusters that are obtained from each data chunk and macro-clusters are created based on the reclustering of micro-clusters. Similar to STREAM, the value of q and K_2 should be specified in advance to guarantee a good performance of CluStream with low computational cost. Also, some iterations are required for CluStream to compute the final converged clusters, which may increase the computational cost. Unlike STREAM and CluStream, the computational complexities of the D-Stream algorithm can be decomposed as attributed to the online mapping from data to grid list and the offline density-based clustering in the grid list. There is no distance calculation for the online mapping of data to grids, and thus the computational cost is $O(1)$. However, the number of grids is greatly influenced by the dimensions of data and it could take a significant amount of computations when data has a large number of dimensions. The computational cost of DBStream primarily depends on the chunk size M and the user-specified time interval t_{gap} [20]. The parameter t_{gap} denotes the duration for the cleanup of weak clusters and it needs to be selected carefully in advance. Besides, the implementation of DBStream requires several threshold values to be given by the user, such as radius, density fading factor, minimum weight and intersection factor. For HDDStream clustering method, its time complexity primarily depends on the number of potential projected core micro-clusters (p_1) and outlier micro-clusters (p_2), which results in a total cost of $O(p_1 \times d + p_2)$. Similar to other DBSCAN-based streaming clustering approaches, the HDDStream clustering algorithm requires a set of predefined parameters to guarantee a high-quality clustering performance. Compared with the other five clustering algorithms, the proposed algorithm has higher computational costs because the number of distance calculations it requires is based on $O(M^2 \times d)$.

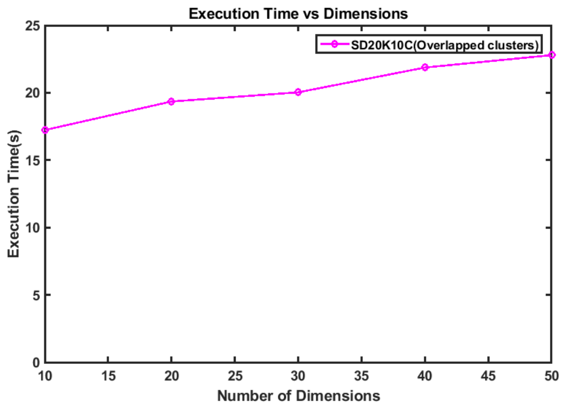
Based on the comparison from Table 4, there is a trade-off between the computational costs and the amount of prior information about data. Insufficient knowledge, or no prior knowledge, calls for a higher computational cost, which is the case of the proposed algorithm. On the other hand, sufficient prior knowledge about the data can effectively reduce the computational cost by eliminating unnecessary trails needed to guess the number of clusters and cluster radii. Therefore, compared with prior well-known clustering techniques, the proposed DFPS-clustering can provide better performance with an acceptable computational cost, which is more applicable to real-world problems.

F. SCALABILITY ANALYSIS

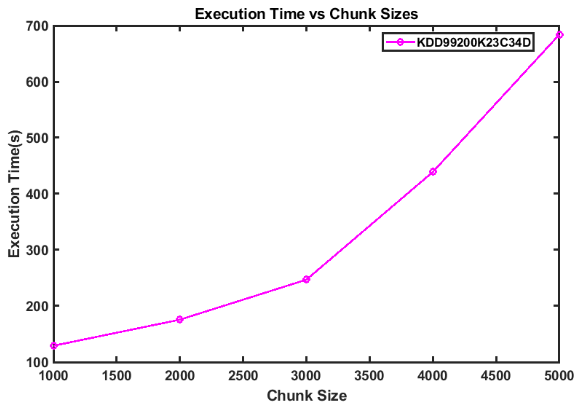
The scalability of the streaming data clustering technique is another important aspect of the efficiency evaluation. To evaluate the scalability of the proposed method, the mixture model of Gaussian distributions is used to generate several synthetic datasets with varying dimensionality. Also, the KDD Cup 99 Network Intrusion Dataset is used as the real-world benchmark for the scalable analysis. The computer used in the simulation has an Intel Core 2 Duo CPU with 3GB RAM and it uses the Microsoft Windows 7 operating system. Both the execution time and memory usage are analyzed here.

1) EXECUTION TIME ANALYSIS

As discussed in the complexity analysis, the execution time of the proposed clustering approach mainly depends on the number of dimensions and the chunk size. Hence, the analysis of the execution time with respect to the dimensionality and chunk size is conducted. There are five synthetic datasets with 20000 samples that are generated from a mixture of ten Gaussian distributions. Each Gaussian distribution forms a cluster around its mean vector and thus ten clusters exist in those synthetic datasets. All ten clusters share the same sample size in each synthetic dataset. The dimensionality of those five synthetic datasets increases from 10 to 50 with an increment of ten. A chunk size of 1000 samples is used in the simulation and the execution time of the proposed clustering method in all five synthetic datasets is obtained. The execution time measures the duration from the first arriving data chunk to the last arriving data chunk. FIGURE 4(a) shows that the execution time of the proposed method is linearly growing with the number of dimensions, which justifies the computational complexity analysis previously. Simultaneously, the real-world Network Intrusion dataset is used to evaluate the overall execution time under different chunk sizes. As mentioned previously, the first 200K samples with 34 numeric features are used in the scalability analysis. Similar to the default setting from [16], [19], [20], [24], [30], the chunk size changes from 1000 samples to 5000 samples with a step size of 1000 and the execution time is recorded here. As shown in FIGURE 4(b), the execution time increases

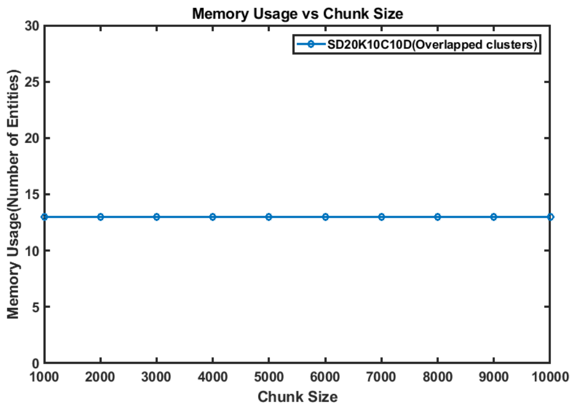


(a)

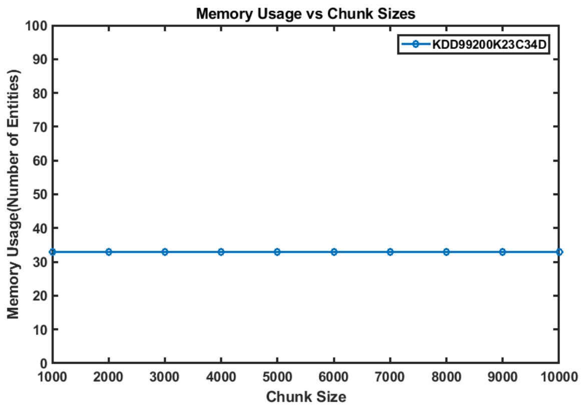


(b)

FIGURE 4. **(a): Execution time vs. Dimensions (Synthetic Datasets), (b): Execution time vs. Chunk Sizes (KDD Network Intrusion Dataset).**



(a)



(b)

FIGURE 5. **(a): Memory Space vs. Chunk size (Synthetic Dataset), (b): Memory Space vs. Chunk size (KDD Network Intrusion Dataset).**

as the chunk size grows. The execution time of the proposed method increases quadratically with the chunk size, which is also consistent with the complexity analysis.

2) MEMORY USAGE ANALYSIS

The limited memory space is another major challenge in clustering streaming data. Since the proposed algorithm only keeps the summary of optimal clusters at each time step, bounded memory complexity is guaranteed here. The memory usage of the proposed method is simulated with a synthetic dataset and the real-world KDD Cup Network Intrusion dataset. The number of entities is used to evaluate memory usage. Each entity refers to a specific cluster object that includes the cluster center vector, cluster density value, and radius. Three default entities are used to hold the global mean, variance and stabilization parameter in memory. The synthetic dataset consists of 20K samples with ten features. There are ten overlapped clusters in the synthetic dataset and each cluster has the same number of samples. Similar to the execution time analysis, the first 200K samples with 34 continuous variables are selected from the Network Intrusion dataset as the benchmark. With a similar setting in [16], [19], [20], [24], [30], the chunk size is set in the range from 1000 to 10000 with a constant increment of 1000.

As shown in FIGURE 5(a), the memory usage of the proposed streaming clustering method is fixed as the chunk size increases and it equals to the number of the natural clusters in the synthetic data stream. Similar to the synthetic data stream, FIGURE 5(a) reveals that the memory space of the proposed method in real-world data stream does not change when chunk size varies. The chunk size does not affect the memory space and the memory space completely depends on the number of natural clusters. Therefore, a bounded memory space for the proposed dynamic clustering procedure can be guaranteed.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a recursive lower bound of the Gaussian kernel function and developed a new two-phase streaming clustering algorithm named Dynamic Fitness

Proportionate Sharing clustering algorithm (DFPS-clustering) for streaming data cluster analysis. Unlike most state-of-the-art methods, our proposed method evolves the cluster structure of the data stream from scratch without the predefined cluster number or the radius. The proposed recursive lower bound of the Gaussian kernel function automatically captures the dynamic evolution of the density distribution in the data stream and a related Lemma is proposed to discriminate the outliers from the new clusters. From the scalability analysis, our proposed streaming clustering method demonstrates a bounded memory space usage as the chunk size increases. Compared with five well-known existing methods from the literature, our proposed method shows a comparable or better clustering performance without any prior knowledge about the data or user-specified parameters, especially when clusters are highly overlapped due to the nature of the data stream. Although the computational cost of the proposed algorithm is more expensive than the other five clustering algorithms, it is reasonable because it does not use any prior information about the data. Furthermore, the efficacy of the proposed method on smart grid stability data and network intrusion data suggest a promising future for the DFPS-clustering algorithm in real-world applications.

In the future, the application of the proposed method will be explored in the area of social media analysis, stock market prediction, and network intrusion detection. Besides, more efforts will be employed to enhance the computational efficiency of the proposed method.

APPENDIX

Proof: Assume that the distance from the current sample $x^i(t)$ to the rest of the samples in the data stream follows a Gaussian distribution, then the density value of the sample $x^i(t)$ can be expressed as:

$$f(x^i(t)) = \sum_{m=1}^{|D_t|} \left(e^{-\frac{\|x^i(t)-x^m(t)\|^2}{\beta_t}} \right)^{\gamma_t}, \tag{8}$$

where Table 5 provides the definitions of all the necessary parameters. The norm here refers to the Euclidean distance d_{im}, between sample $x^i(t)$ and $x^m(t)$.

The parameter $|D_t|$ is the total number of the previous and current samples that can be decomposed by the sum of the number of the previous samples $|D_{t-1}|$ and the current samples $|C_t|$, hence: $|D_{t-1}| + |C_t| = |D_t|$. Also, the number of the old samples can be further decomposed by the sum of samples in the previous clusters: $|D_{t-1}| = \sum_{j=1}^{|Z_{t-1}|} |P^j_{t-1}|$. The density value of the new sample $x^i(t)$ can be rewritten as the sum of the densities from the previous samples and the new samples :

$$f(x^i(t)) = \left[\sum_{m_1=1}^{|D_{t-1}|} \left(e^{-\frac{d^2_{im_1}}{\beta_t}} \right)^{\gamma_t} + \sum_{m_2=1}^{|C_t|} \left(e^{-\frac{d^2_{im_2}}{\beta_t}} \right)^{\gamma_t} \right], \tag{9}$$

TABLE 5. Table of parameters.

Parameter	Description
$x^i(t)$	The i_{th} sample in the current data chunk at t
C_t	$C_t = \{x^i(t), i = 1, 2, ..., M\}$
D_t	The collection of data chunks up to time t
Z_{t-1}	Cluster in the data stream D_{t-1}
P^j_{t-1}	Cluster j of data stream D_{t-1}
Z^j_{t-1}	The center of the cluster j in the data stream D_{t-1}
Z_{C_t}	The centers of new clusters in the chunk C_t
$x^k_j(t-1)$	The k_{th} sample in the cluster j at $t-1$
m_1	The index of samples in the data stream D_{t-1}
m_2	The index of samples in the chunk C_t
d_{im}	The distance from sample $x^i(t)$ to sample $x^m(t)$
d_{ijk}	The distance from $x^i(t)$ to $x^k_j(t-1)$
d_{ij}	The distance from $x^i(t)$ to Z^j_{t-1}
d_{kj}	The distance from $x^k_j(t-1)$ to Z^j_{t-1}
$\alpha_{k,1}$	The angle between the d_{ij} and $d_{kj,1}$
$\alpha_{k,2}$	The angle between the d_{ij} and $d_{kj,2}$
α_k	The angle between the d_{ij} and d_{kj}
β_t	The variance of the data stream at time D_t
β_{C_t}	The variance of the current data chunk C_t
$\hat{f}\left(x^i(t)\right)$	The estimated density value of sample x^i in the arriving data chunk C_t

where

$$\sum_{m_1=1}^{|D_{t-1}|} \left(e^{-\frac{d^2_{im_1}}{\beta_t}} \right)^{\gamma_t} = \sum_{j=1}^{|Z_{t-1}|} \left[\sum_{k=1}^{|P^j_{t-1}|} \left(e^{-\frac{d^2_{ijk}}{\beta_t}} \right)^{\gamma_t} \right]. \tag{10}$$

According to the cosine principle of the triangle $\{x^i(t), x^k_j(t-1), Z^j_{t-1}\}$ in FIGURE 6(a), the following relationship holds

$$d^2_{ijk} = d^2_{ij} + d^2_{kj} - 2d_{ij}d_{kj}cos\alpha_k, \tag{11}$$

As shown in FIGURE 6(b), each sample $x^{k,1}_j(t-1)$ in cluster $P^j(t-1)$ has a sample $x^{k,2}_j(t-1)$ that is symmetrical with respect to a line L_c that passes through the center of the cluster. Due to the symmetric property of samples in the cluster j, all samples in cluster j can be represented by $\frac{|P^j_{t-1}|}{2}$ pairs of samples in the cluster, which can be described by Equation (12).

$$\begin{aligned} \sum_{k=1}^{|P^j_{t-1}|} \left(e^{-\frac{d^2_{ijk}}{\beta_t}} \right) &= \sum_{k=1}^{|P^j_{t-1}|} \left(e^{\frac{-d^2_{ij}-d^2_{kj}+2d_{ij}d_{kj}cos\alpha_k}{\beta_t}} \right) \\ &= \sum_{k=1}^{\frac{|P^j_{t-1}|}{2}} \left[e^{\frac{\left(-d^2_{ij}-d^2_{kj1}+2d_{ij}d_{kj1}cos\alpha_{k,1}\right)}{\beta_t}} \right. \\ &\quad \left. + e^{\frac{\left(-d^2_{ij}-d^2_{kj2}+2d_{ij}d_{kj2}cos\alpha_{k,2}\right)}{\beta_t}} \right]. \end{aligned} \tag{12}$$

Since $\alpha_{k,1} + \alpha_{k,2} = 180°$ and $d_{kj,1} = d_{kj,2}$, the right hand side of Equation (12) can be rewritten as

$$\begin{aligned} &\sum_{k=1}^{\frac{|P^j_{t-1}|}{2}} e^{\frac{\left(-d^2_{ij}-d^2_{kj1}+2d_{ij}d_{kj1}cos\alpha_{k,1}\right)}{\beta_t}} + e^{\frac{\left(-d^2_{ij}-d^2_{kj1}-2d_{ij}d_{kj1}cos\alpha_{k,1}\right)}{\beta_t}} \\ &\quad = \sum_{k=1}^{\frac{|P^j_{t-1}|}{2}} \left[e^{\frac{-d^2_{ij}-d^2_{kj}}{\beta_t}} \times \left(e^{\frac{2d_{ij}d_{kj}cos\alpha_{k,1}}{\beta_t}} + e^{-\frac{2d_{ij}d_{kj}cos\alpha_{k,1}}{\beta_t}} \right) \right]. \end{aligned} \tag{13}$$

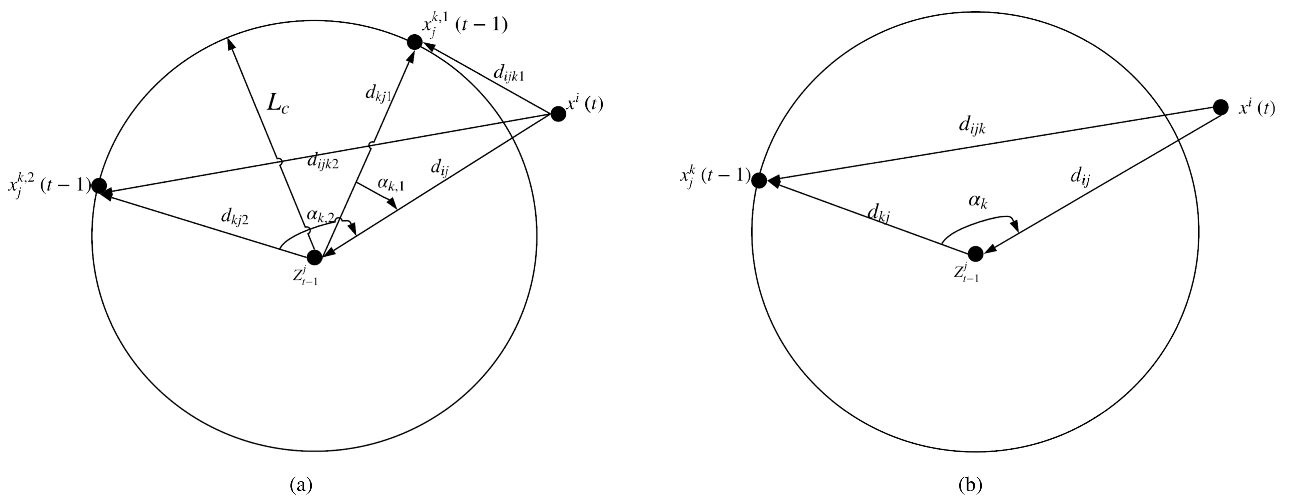


FIGURE 6. (a): The old sample $x_j^k(t-1)$ in the previous cluster j and new instance x^i, (b):The symmetric points $x_j^{k,1}(t-1), x_j^{k,2}(t-1)$ with respect to the middle line L_c in the cluster j and the new instance $x^i(t)$.

According to the inequality: $a + \frac{1}{a} \geq 2, a > 0$, the following inequality can be computed as follows

$$\left(e^{\frac{2d_{ij}d_{kj}cos\alpha_{k,1}}{\beta_t}} + e^{-\frac{2d_{ij}d_{kj}cos\alpha_{k,1}}{\beta_t}}\right) \geq 2. \tag{14}$$

Since d_{ij} is constant for all samples in the cluster P^j_{t-1}, Equation (12) can be rewritten using the inequality (14) as

$$\sum_{k=1}^{|P^j_{t-1}|}\left(e^{-\frac{d^2_{ijk}}{\beta_t}}\right) \geq e^{-\frac{d^2_{ij}}{\beta_t}} \times \sum_{k=1}^{|P^j_{t-1}|}\left(e^{-\frac{d^2_{kj}}{\beta_t}}\right). \tag{15}$$

With inequality (15), an inequality of Equation (10) can be expressed as

$$\begin{aligned}\sum_{m_1=1}^{|D_{t-1}|}\left(e^{-\frac{d^2_{im_1}}{\beta_t}}\right)^{\gamma_t} &= \sum_{j=1}^{|Z_{t-1}|}\left[\sum_{k=1}^{|P^j_{t-1}|}\left(e^{-\frac{d^2_{ijk}}{\beta_t}}\right)^{\gamma_t}\right] \\ &\geq \sum_{j=1}^{|Z_{t-1}|}\left[(e^{-\frac{d^2_{ij}}{\beta_t}})^{\gamma_t} \times \sum_{k=1}^{|P^j_{t-1}|}\left(e^{-\frac{d^2_{kj}}{\beta_t}}\right)^{\gamma_t}\right].\end{aligned} \tag{16}$$

When a new cluster appears, the variance of the data stream will change while the following relationship in Algorithm 2 holds:

$$\frac{\beta_t}{\gamma_t} = \frac{\beta_{t-1}}{\gamma_{t-1}}, \tag{17}$$

such that

$$\sum_{k=1}^{|P^j_{t-1}|}\left(e^{-\frac{d^2_{kj}}{\beta_t}}\right)^{\gamma_t} = \sum_{k=1}^{|P^j_{t-1}|} e^{-\frac{\frac{d^2_{kj}}{\beta_t}}{\gamma_t}} = \sum_{k=1}^{|P^j_{t-1}|}(e^{-\frac{d^2_{kj}}{\frac{\beta_{t-1}}{\gamma_{t-1}}}}). \tag{18}$$

Let $f\left(Z^j_{t-1}\right) = \sum_{k=1}^{|P^j_{t-1}|}\left(e^{-\frac{d^2_{kj}}{\beta_{t-1}}}\right)^{\gamma_{t-1}}$, then

$$\sum_{k=1}^{|P^j_{t-1}|}\left(e^{-\frac{d^2_{kj}}{\beta_t}}\right)^{\gamma_t} = f\left(Z^j_{t-1}\right), \tag{19}$$

With Equation (19), Equation (16) can be rewritten as

$$\sum_{m_1=1}^{|D_{t-1}|}\left(e^{-\frac{d^2_{im_1}}{\beta_t}}\right)^{\gamma_t} \geq \sum_{j=1}^{|Z_{t-1}|}\left[\left(e^{-\frac{d^2_{ij}}{\beta_t}}\right)^{\gamma_t} \times f\left(Z^j_{t-1}\right)\right]. \tag{20}$$

Hence, an inequality of Equation (9) can be derived using the inequality (20) as

$$f(x^i(t)) \geq \sum_{j=1}^{|Z_{t-1}|}\left[\left(e^{-\frac{d^2_{ij}}{\beta_t}}\right)^{\gamma_t} \times f\left(Z^j_{t-1}\right)\right] + \sum_{m_2=1}^{|C_t|}\left(e^{-\frac{d^2_{im_2}}{\beta_t}}\right)^{\gamma_t}. \tag{21}$$

Based on above Equation (21), an estimated lower boundary of the density value of sample $x^i(t)$ is defined as

$$\hat{f}\left(x^i(t)\right) = \sum_{j=1}^{|Z_{t-1}|}\left[\left(e^{-\frac{d^2_{ij}}{\beta_t}}\right)^{\gamma_t} \times f\left(Z^j_{t-1}\right)\right] + \sum_{m_2=1}^{|C_t|}\left(e^{-\frac{d^2_{im_2}}{\beta_t}}\right)^{\gamma_t}, \tag{22}$$

where $f(x^i(t)) \geq \hat{f}(x^i(t))$.

Therefore, based on Equation (22), a recursive lower bound of the estimated density values is derived and Lemma 1 is defined to check the occurrence of new clusters as the variance of the data stream varies. Otherwise, samples in the new data chunk are directly assigned to the existing cluster. ■

ACKNOWLEDGMENT
The authors would like to thank the Air Force Research Laboratory and Office of the Secretary of Defense. The authors would also like to thank the NASA Langley Research Centers.

REFERENCES

[1] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*. Berlin, Germany: Auerbach, 2016.

[2] J. Gama and M. M. Gaber, *Learning From Data Streams Processing Techniques in Sensor Networks*. Berlin, Germany: Springer, 2007.

[3] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: A review," *ACM SIGMOD Rec.*, vol. 34, no. 2, pp. 18–26, Jun. 2005.

[4] Q. Su and L. Chen, "A method for discovering clusters of e-commerce interest patterns using click-stream data," *Electron. Commerce Res. Appl.*, vol. 14, no. 1, pp. 1–13, 2015.

[5] S. Geisler, C. Quix, S. Schiffer, and M. Jarke, "An evaluation framework for traffic information systems based on data streams," *Transp. Res. C, Emerg. Technol.*, vol. 23, pp. 29–55, Aug. 2012.

[6] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez, "Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study," *IEEE Syst. J.*, vol. 9, no. 1, pp. 31–44, Mar. 2015.

[7] X. Li, B. Plale, N. Vijayakumar, R. Ramachandran, S. Graves, and H. Conover, "Real-time storm detection and weather forecast activation through data mining and events processing," *Earth Sci. Inform.*, vol. 1, no. 2, pp. 49–57, 2008.

[8] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski, "Open challenges for data stream mining research," *ACM SIGKDD Explor. Newslett.*, vol. 16, no. 1, pp. 1–10, 2014.

[9] S. Mansalis, E. Ntoutsi, N. Pelekis, and Y. Theodoridis, "An evaluation of data stream clustering algorithms," *Stat. Anal. Data Mining The ASA Data Sci. J.*, vol. 11, no. 4, pp. 167–187, 2018.

[10] 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.

[11] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in *Proc. 41st Annu. Symp. Found. Comput. Sci.*, Nov. 2000, pp. 359–366.

[12] A. Workineh and A. Homaifar, "A fitness proportionate reward sharing: A viable default hierarchy formation strategy in LCS," in *Proc. Int. Conf. Genetic Evol. Methods (GEM)*, 2012, p. 1.

[13] D.-X. Chang, X.-D. Zhang, C.-W. Zheng, and D.-M. Zhang, "A robust dynamic niching genetic algorithm with niche migration for automatic clustering problem," *Pattern Recognit.*, vol. 43, no. 4, pp. 1346–1360, 2010.

[14] J. Gan and K. Warwick, "Dynamic niche clustering: A fuzzy variable radius niching technique for multimodal optimisation in GAs," in *Proc. Congr. Evol. Comput.*, vol. 1, May 2001, pp. 215–222.

[15] 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.*, Feb./Mar. 2002, pp. 685–694.

[16] 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*, 2003, pp. 81–92.

[17] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, pp. 103–114, Jun. 1996.

[18] D. K. Tasoulis, N. M. Adams, and D. J. Hand, "Unsupervised clustering in streaming data," in *Proc. 6th IEEE Int. Conf. Data Mining—Workshops*, Dec. 2006, pp. 638–642.

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

[20] 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.

[21] J. Gama, *Knowledge Discovery From Data Streams*. Boca Raton, FL, USA: CRC Press, 2010.

[22] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: Indexing micro-clusters for anytime stream mining," *Knowl. Inf. Syst.*, vol. 29, no. 2, pp. 249–272, 2011.

[23] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, "Maintaining variance and K-medians over data stream windows," in *Proc. 22nd ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, 2003, pp. 234–243.

[24] S. Guha and N. Mishra, "Clustering data streams," in *Data Stream Management*. Washington, DC, USA: Springer, 2016, pp. 169–187.

[25] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding windows," *Knowl. Inf. Syst.*, vol. 15, no. 2, pp. 181–214, 2008.

[26] 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*, 2006, pp. 328–339.

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

[28] S. Laohakiat, S. Phimoltares, and C. Lursinsap, "Hyper-cylindrical micro-clustering for streaming data with unscheduled data removals," *Knowl.-Based Syst.*, vol. 99, pp. 183–200, May 2016.

[29] N. Wattanakitrungroj, S. Maneeroj, and C. Lursinsap, "BEstream: Batch capturing with elliptic function for one-pass data stream clustering," *Data & Knowl. Eng.*, vol. 117, pp. 53–70, Sep. 2018.

[30] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel, "Density-based projected clustering over high dimensional data streams," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 987–998.

[31] A. A. A. Alazeez, S. Jassim, and H. Du, "EDDS: An enhanced density-based method for clustering data streams," in *Proc. 46th Int. Conf. Parallel Process. Workshops (ICPPW)*, Aug. 2017, pp. 103–112.

[32] 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.

[33] S. Gong, Y. Zhang, and G. Yu, "Clustering stream data by exploring the evolution of density mountain," *Proc. VLDB Endowment*, vol. 11, no. 4, pp. 393–405, 2017.

[34] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[35] A. Amini and T. Y. Wah, "DENGRIS-Stream: A density-grid based clustering algorithm for evolving data streams over sliding window," in *Proc. Int. Conf. Data Mining Comput. Eng.*, 2012, pp. 206–210.

[36] C. Jia, C. Tan, and A. Yong, "A grid and density-based clustering algorithm for processing data stream," in *Proc. 2nd Int. Conf. Genetic Evol. Comput.*, Sep. 2008, pp. 517–521.

[37] V. Bhatnagar, S. Kaur, and S. Chakravarthy, "Clustering data streams using grid-based synopsis," *Knowl. Inf. Syst.*, vol. 41, no. 1, pp. 127–152, Oct. 2014.

[38] 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.

[39] U. Kokate, A. Deshpande, P. Mahalle, and P. Patil, "Data stream clustering techniques, applications, and models: Comparative analysis and discussion," *Big Data Cogn. Comput.*, vol. 2, no. 4, p. 32, 2018.

[40] 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 Data (TKDD)*, vol. 3, no. 3, p. 14, 2009.

[41] A. Amini and T. Y. Wah, "Leaden-stream: A leader density-based clustering algorithm over evolving data stream," *J. Comput. Commun.*, vol. 1, no. 5, p. 26, 2013.

[42] P. Angelov, *Autonomous Learning Systems: From Data Streams To Knowledge In Real-Time*. Hoboken, NJ, USA: Wiley, 2012.

[43] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst., Man, Cybern., B, (Cybern.)*, vol. 34, no. 1, pp. 484–498, Feb. 2004.

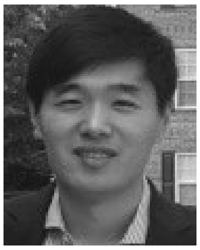
[44] X. Gu and P. P. Angelov, "Autonomous data-driven clustering for live data stream," in *Proc. IEEE Int. Conf. Syst. Man, Cybern. (SMC)*, Oct. 2016, pp. 001128–001135.

[45] A. Hinneburg and H.-H. Gabriel, "DENCLUE 2.0: Fast clustering based on Kernel density estimation," in *Proc. Int. Symp. Intell. Data Anal.* Berlin, Germany: Springer, 2007, pp. 70–80.

[46] K.-L. Wu and M.-S. Yang, "Mean shift-based clustering," *Pattern Recognit.*, vol. 40, no. 11, pp. 3035–3052, 2007.

[47] J. Chen, Y. Wu, X. Lin, and Q. Xuan, "DOE-AND-SCA: A novel SCA based on DNN with optimal eigenvectors and automatic cluster number determination," *IEEE Access*, vol. 6, pp. 20764–20778, 2018.

[48] M.-S. Yang and K.-L. Wu, "A similarity-based robust clustering method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 434–448, Apr. 2004.
[49] L. A. Zadeh, "Similarity relations and fuzzy orderings," *Inf. Sci.*, vol. 3, no. 2, pp. 177–200, Apr. 1971.
[50] R. R. Yager and D. P. Filev, "Approximate clustering via the mountain method," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 24, no. 8, pp. 1279–1284, Aug. 1994.
[51] A. Workineh and A. Homaifar, "Fitness proportionate niching: Maintaining diversity in a rugged fitness landscape," in *Proc. Int. Conf. Genetic Evol. Methods (GEM). Steering Committee World Congr. Comput. Sci. Comput. Eng. Appl. Comput. (WorldComp)*, 2012, p. 1.
[52] X. Yan, A. Homaifar, S. Nazmi, and M. Razeghi-Jahromi, "A novel clustering algorithm based on fitness proportionate sharing," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Oct. 2017, pp. 1960–1965.
[53] D. Chang, Y. Zhao, L. Liu, and C. Zheng, "A dynamic niching clustering algorithm based on individual-connectedness and its application to color image segmentation," *Pattern Recognit.*, vol. 60, pp. 334–347, Dec. 2016.
[54] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. 4th. Int. Conf. Knowl. Discovery. Data Mining*, vol. 98. Aug. 1998, pp. 58–65.
[55] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. W. Hamlen, and N. C. Oza, "Facing the reality of data stream classification: Coping with scarcity of labeled data," *Knowl. Inf. Syst.*, vol. 33, no. 1, pp. 213–244, Oct. 2012.
[56] M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[57] B. Schäfer, C. Grabow, S. Auer, J. Kurths, D. Witthaut, and M. Timme, "Taming instabilities in power grid networks by decentralized control," *Eur. Phys. J. Special Topics*, vol. 225, no. 3, pp. 569–582, 2016.
[58] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. Amer. Statist. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971.
[59] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.
[60] M. J. Warrens, "On the equivalence of cohen's Kappa and the Hubert-Arabie adjusted rand index," *J. Classification*, vol. 25, no. 2, pp. 177–183, 2008.
[61] M. Ghesmoune, M. Lebbah, and H. Azzag, "State-of-the-art on clustering data streams," *Big Data Anal.*, vol. 1, no. 1, p. 13, 2016.

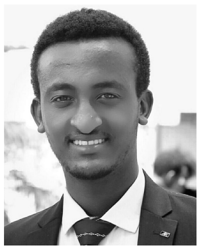


XUYANG YAN received the B.S. degree from North Carolina Agricultural and Technical State University (NC A&T) and Henan Polytechnic University, in 2016, and the M.S. degree from NC A&T, in 2018, all in electrical engineering, where he is currently pursuing the Ph.D. degree in electrical engineering. His current research interests include extracting knowledge from streaming data, analyzing the emergent behaviors of large-scale autonomous systems, and the application of machine learning techniques in robotics.

MOHAMMAD RAZEGHI-JAHROMI received the B.S. degree from the Amirkabir University of Technology, Tehran, Iran, in 1997, the M.S. degree from the University of Tehran, Tehran, in 2000, and the Ph.D. degree from the University of Rochester, Rochester, NY, USA, in 2016, all in electrical engineering. He has been a Research Scientist with ABB Corporate Research United States (USCRC), Raleigh, NC, USA, since 2017. His current research interests include networked control systems, control systems theory, stochastic control and stochastic differential equations, Markov jump linear systems, machine learning techniques, and convex optimization.

ABDOLLAH HOMAIFAR received the B.S. and M.S. degrees from the State University of New York at Stony Brook, in 1979 and 1980, respectively, and the Ph.D. degree from the University of Alabama, in 1987, all in electrical engineering. He is currently the NASA Langley Distinguished Professor and the Duke Energy Eminent Professor with the Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University (NCA&TSU). He is the Director of the Autonomous Control and Information Technology Institute and the Testing, Evaluation, and Control of Heterogeneous Large-scale Systems of Autonomous Vehicles (TECHLAV) Center, NCA&TSU. His current research interests include machine learning, unmanned aerial vehicles (UAVs), testing and evaluation of autonomous vehicles, optimization, and signal processing. He is a member of the IEEE Control Society, Sigma Xi, Tau Beta Pi, and Eta Kapa Nu. He serves as an Associate Editor of the *Journal of Intelligent Automation and Soft Computing*. He is a Reviewer of the IEEE Tbioscransactions on F bioscuzzy S bioscystems, M bioscan M bioscachines and C bioscybernetics, and *Neural Networks*.

BERAT A. EROL received the B.Sc. degree in mathematics from Kocaeli University, in 2007, the M.Sc. degree in software engineering from St. Mary's University, in 2012, and the Ph.D. degree in electrical engineering from The University of Texas at San Antonio, in 2018. His dissertation and research activities with UTSA's Autonomous Control Engineering Laboratories have been sponsored by several research grants and contracts that he contributed to, including the US DoD, Bank of America, 80|20 foundation, UTSA Lutcher Brown Endowed Chair, and UTSA Open Cloud Institute. He is currently a Postdoctoral Fellow with the ACIT Institute, North Carolina Agricultural and Technical State University. His current research interests include autonomous systems, human-robot interactions, manned-unmanned teaming, visual SLAM, machine learning, and the Internet of Robotic Things (IoRT). He is a member of AIAA and IEEE Eta Kappa Nu honor society.



ABENEZER GIRMA received the B.Sc. degree in electrical and electronics engineering from the Addis Ababa Institute of Technology. He is currently pursuing the direct Ph.D. degree in electrical engineering with North Carolina Agricultural and Technical State University. His current research interests include machine learning, big data analysis, and data driven algorithms in autonomous vehicles and connected cars.

EDWARD TUNSTEL received the B.S. and M.Eng. degrees from Howard University, in 1986 and 1989, respectively, all in mechanical engineering, and the Ph.D. in electrical engineering from The University of New Mexico, in 1996. He was a Senior Robotics Engineer with the NASA Jet Propulsion Laboratory, Pasadena, CA, USA, from 1989 to 2007, a Space Robotics and Autonomous Control Lead and Senior Roboticist with the Johns Hopkins Applied Physics Laboratory, Laurel, MD, USA, from 2007 to 2017. He is currently an Associate Director of robotics with the United Technologies Research Center, East Hartford, CT, USA. He serves as the President of the IEEE Systems, Man, and Cybernetics Society, from 2018 to 2019. His current research interests include mobile robot navigation, autonomous control, cooperative multirobot systems, human-collaborative robotics, robotic systems engineering, and applications of soft computing to autonomous systems.