i-CODAS: An Improved Online Data Stream Clustering in Arbitrary Shaped Clusters

Md Kamrul Islam, Md Manjur Ahmed, and Kamal Zuhairi Zamli

Abstract—**Nowadays a lot of IT-based applications are generating huge data streams continuously, and clustering of these streams provide many advantages in data mining. In the field of clustering of data stream, density-based technique is the most popular as it is able to generate arbitrary shaped cluster with high cluster quality in a noisy environment. However, most of the existing density-based algorithms for data stream clustering are either offline or hybrid of offline and online phase or can handle only hyper-elliptical clusters. But offline algorithms are not good choice for data stream clustering as storing the data stream is impractical and often the shape of the cluster is arbitrary rather than regular in data space. Recently, an online clustering method called CODAS has been proposed where the generated clusters are arbitrary in shape. However, like other existing density based clustering algorithms, the radius of all micro-cluster in CODAS is global and constant. But it is really hard to set the optimal value of micro-cluster radius in practical, and a global radius may not be optimal for each micro-cluster. An erroneous choice of radius decreases the clustering quality remarkably. In this paper, we present an improved version of CODAS called i-CODAS based on the concept of maintaining local radius for each micro-cluster independently. The radius is updated in an online manner towards its local optimal value as new data sample lies in the cluster. The data samples are summarized in a metadata, called micro-cluster. The micro-clusters are presented in a clustering graph based on the connectivity among micro-clusters. The clustering graph is finally used to generate arbitrary shaped clusters. The performance of the proposed i-CODAS is measured and compared with other density-based clustering algorithms. The experimental result proves the superiority of i-CODAS over other clustering algorithms in terms of noise sensitivity, accuracy, purity, processing speed and scalability.**

Index Terms—**data stream, online clustering, arbitrary shape, Euclidean distance, cluster graph.**

I. Introduction

THE fast development of information technology (IT) leads to generate a massive amount of data continuously from several emerging applications known as the data stream. The example of such data stream sources include sensor networks, anomaly detection, financial transactions, call records, social data, multimedia data, advertising, etc. Modern societies are increasingly interested to exploit and utilize these data stream to solve social problems [1]. The knowledge inside these data streams are discovered using data mining technique. Clustering is one of the vital tasks in data mining to identify the abstract of hidden themes in a coherent manner those characterize the stream [2]. Clustering of data stream is defined as the technique for partitioning the data in data stream into clusters where the similar data are placed in the same cluster, and dissimilar data are placed in another cluster [3], [4], [5]. It is an unsupervised learning that has become a useful, ubiquitous and essential tool in machine learning, data mining, bio-informatics, image processing, and other fields, in data stream analysis increasingly [6], [7], [8]. There are a lot of research on clustering of static datasets in offline mode, but they cannot be applied on data stream due to its uncertainty in volume, arrival speed and gradual change with time [9], [10]. This storing problem is solved by providing online clustering algorithms where only the data point summarization (or cluster) is stored and updated as new data arrives [11]. The tendency for developing online clustering algorithm is a growing and challenging phenomenon to deal with data stream [12]. Many clustering algorithms exist in literature those are categorized into five categories and they are partitioning, hierarchical, density-based, grid-based, and model-based methods [13]. Among them, density-based clustering has been found a natural and most attractive clustering technique as it has the ability to generate arbitrarily shaped clusters in dense areas and to detect noises and act accordingly in noisy environment [14].

In recent years, researchers have proposed many density-based clustering algorithms for data stream. DBSCAN is considered to be the primitive density-based clustering algorithm that generates arbitrarily shaped clusters in an incremental manner [15]. However, it is not preferable for high dimensional data set as it suffers from the so-called ”curse of dimensionality” [16]. Recently, DBSCAN is enhanced in GISN-DBSCAN by enabling it to process complex data stream, detect noise, and reduce the computational time [17]. However, the cluster quality degrades in GISN-DBSCAN. Two density-based clustering algorithms; CluStream [18] and DenStream [19] summarizes the data stream by storing the temporal locality of data which is termed as micro-cluster. Though, they improve cluster quality, but CluStream is limited to generate only spherical shaped cluster and DenStream suffers from the time-consuming task of pruning the outlier micro-cluster. C-DenStream [20], rDenStream [21], SDStream [22], HDenStream [23], HDDStream [24], VDStream [25] are all improvement of DenStream. The semi-supervised C-DenStream, retrospect learning-based rDenStream and SNN (Shared Nearest Neighbor)-based VDStream focuses on improving cluster quality. However, they require high processing time and memory space, and they are not applicable to high dimensional data stream. For handling the evolving nature, sliding window based SDStream and HDenStream are introduced. SDStream can detect the noise efficiently, but not scalable to high dimensional data stream.

Manuscript received on October 11, 2018; revised on May 23, 2019.

Md Kamrul Islam is a postgraduate student in the faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, 26300 Gambang, Pahang, Malaysia. He is also with Jashore University of Science and Technology, Bangladesh. (e-mail: polash2k48@gmail.com)

Md Manjur Ahmed is with the department of Computer Science and Engineering, University of Barisal, Kornokathi, Patuakhali Highway, Barisal 8200, Bangladesh. (Corresponding author, phone: +880-1851-924944; e-mail: manjur_39@yahoo.com).

Kamal Zuhairi Zamli is with the faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, 26300 Gambang, Pahang, Malaysia. (e-mail: kamalz@ump.edu.my)

HDenStream can process heterogeneous data stream, but does not describe the idea to store the categorical attributes in an efficient way. On the other hand, HDDStream is able to handle high-dimensional data stream and generate cluster with high quality, but the clusters do not evolve. The efficiency of HDDStream was further improved in the PreDeConStream [26] by adapting the fading function [19] in order to detect the evolving nature in data stream. However, PreDeConStream suffers from its time consuming pruning phase. SOStream [27] is another density-based clustering approach that achieves high clustering quality with occupying less memory, but the processing time is quite high.

Based on affinity propagation clustering [28], two density-based clustering algorithms were proposed, namely APDenStream [29] and ADStream [30]. They improve the cluster quality. However, APDenStream is not suitable for high dimensional data stream and ADStream is not suitable for noisy data stream. The bio-inspired model like FlockStream [31] and ACSC [32] are fast and generate high quality clusters. However FlockStream does not clarify about removing the outliers from memory and ACSC suffering from time consuming cluster merging operation.

The aforementioned algorithms are either online-offline hybrid or incremental clustering process. Baruah and Angelov developed two online evolving clustering algorithm called ELM [33] and DEC [34]. They provide high cluster purity and low processing time but are unable to generate arbitrary shaped clusters. An adaptive clustering technique was introduced in [35] for dynamic IoT data streams. The technique shows good cluster quality, but it requires predetermining the number of clusters in the system and cannot generate arbitrarily shaped clusters. Recently, an online clustering algorithm, CODAS [11] has been proposed which generates arbitrary shaped clusters from data streams. The purity and accuracy of CODAS are quite high and it shows good performance in terms of processing time and memory requirement. However, the problems of setting the micro-cluster radius are still unsolved in the field of density-based clustering of data stream. Using constant and global radius of micro-clusters creates two problems. Firstly, it is tough to set the constant value of optimal radius of micro-cluster prior to the execution due to variable density in the data space, and an erroneous choice of radius degrades the clustering performance remarkably. Secondly, a global value of radius may cause inefficient clustering as some micro-clusters may contain more sparse areas than other micro-clusters in the system and a unique radius may not be a good choice for both of them. In this paper, we present an online clustering algorithm called i-CODAS (improved CODAS) for data stream that maintains the local radius of each micro-cluster independently. In i-CODAS, rather than predicting a global and constant radius, a range of radius is set during the initialization of micro-cluster. The maximum radius confirms the cluster separation and minimum radius confirms the formation of the cluster. Moreover, the radius is then updated recursively towards its local optimal based on spatial information of the data sample. The metadata of micro-cluster is updated as data sample falls into it. The circular shaped micro-clusters forms a cluster graph based on their connectivity. The arbitrary shaped clusters are generated from the cluster graph.

Section 2 describes the proposed i-CODAS (improved CODAS) algorithm in details. After that, the performance of i-CODAS has been measured and compared with the existing data stream clustering algorithms (Section 3). The measured performance shows that the proposed i-CODAS algorithm outperforms other existing clustering algorithms including CODAS. Finally, Section 4 concludes the article and describes future research directions.

II. The Proposed Approach

The proposed i-CODAS clustering algorithm is an improved version of the existing CODAS algorithm. i-CODAS is a fully online algorithm for clustering of data stream that reduces the dependency on user in setting the optimal value of algorithm parameter. Contrast to traditional offline clustering methods, the online i-CODAS algorithm stores the metadata of data samples. Similar to other density-based clustering algorithms, the proposed i-CODAS algorithm maintains the metadata in the forms of micro-clusters and macro-clusters or simply clusters. The data structure of micro-cluster is shown in Fig.1.

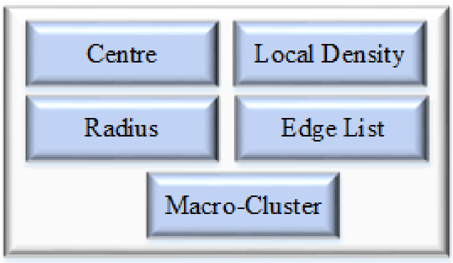


Fig. 1: Structure of Micro-cluster in i-CODAS

The data structure of micro-cluster includes the following information

1) Centre: The centre of a micro-cluster is computed as the mean of the data samples in the micro-cluster. It shows the position of a micro-cluster in the data space.
2) Radius: The radius of the micro-cluster describes the spread of the micro-cluster from the centre. The outer region which is covered by half of the radius is known as shell region whereas the inner half part is known as the kernel region.
3) Local density: Local density of a micro-cluster is simply the number of data samples within a micro-cluster.
4) Edge list: If the kernel region of a micro-cluster intersects with shell or kernel region of another micro-cluster then they are said to be edged or intersected. The intersected micro-clusters of a micro-cluster are collectively build the edge list of the micro-cluster. An empty edge list of a micro-cluster indicates that no micro-cluster intersect with that micro-cluster.
5) Macro-cluster: The intersecting micro-clusters form a single macro-cluster. Other words, intersecting micro-clusters belong to the same macro-cluster. A micro-cluster with the local density more than threshold but with no intersecting micro-cluster forms a macro-cluster itself. The macro-cluster is also called as the cluster.

i-CODAS algorithm defines the term density threshold to differentiate between the true clusters from background noise in the data stream. Based on the threshold value, i-CODAS maintains the following two types of metadata.

1) Micro-cluster: A metadata with local density above or equal to the density threshold is identified as a micro-cluster. They actively participate in the cluster graph for generating the micro-cluster/clusters.

2) Outlier: The metadata with the local density less than the density threshold is said to be an outlier. Outliers are actually the noisy samples. They dont participate in cluster graph for cluster generation.

A. Description of i-CODAS algorithm

The proposed i-CODAS algorithm requires three parameters to be set prior to the execution of the algorithm. The parameters are set based on expert knowledge about the application. The parameters include the following constants.

1) Radius Growth factor (G_r): The radius growth factor describes that how fast the radius of micro-cluster grows to its optimal value. The parameter contributes to increasing the radius of the micro-cluster recursively every time a data sample falls in the shell region of the micro-cluster.
2) Maximum Radius (R_{max}) and Minimum Radius (R_{min}): Maximum radius is the maximum distance between data so that any data outside this radius belongs to a different micro-cluster. Whereas, minimum radius is the lowest value of the distance between data so that any data outside this radius belongs to a different micro-cluster. A radius more than the maximum radius, affects the cluster separation and smoothness of the micro-cluster. On the other hand, a radius less than the minimum radius, prevents the formation of micro-cluster in dense region with enough samples.
3) Density Threshold ($Threshold_{min}$): The minimum number of data samples that is required to form a micro-cluster is referred to as the density threshold. This value separates the micro-clusters from background noise in data stream.

The i-CODAS algorithm composed of the following four distinct sub-algorithms excluding parameter setting.

i Micro-cluster Initiation (Algorithm 2.2)
ii Search the Micro-Cluster (Algorithm 2.3)
iii Update Micro-Clusters (Algorithm 2.4)
iv Update Cluster Graph (Algorithm 2.5)

A data stream (X) is defined as a series of data samples X $= \{X_0, X_1, X_2, ..., X_i, X_{i+1}, ...\}$, where X_i is the i^{th} data sample [36]. As a data sample comes from the data stream, these four sub-algorithms are executed under the supervision of the master algorithm (Algorithm 2.1) of i-CODAS. The master algorithm (Algorithm 2.1) is executed till samples are coming from the data stream.

Algorithm 2.1: i-CODAS
This algorithm supervises the whole clustering process. It calls the other four sub-algorithms to cluster the data stream and update the cluster metadata.
Input: Data Stream, X $= \{X_0, X_1, .., X_i, X_{i+1}, ..\}$, R_{min}, R_{max},$Threshold_{min}$.

```
Step 1: Repeat from Step 2 to Step 5 for each data
        sample, $X_i$ in data stream $X$
Step 2: Call Algorithm 2.4 to check whether the
        current sample resides in any micro-cluster.
Step 3: If Algorithm 2.4 returns false then
            Go to Step 4.
        Else
            Go to Step 5.
        End If
Step 4: Create new micro-cluster using Algorithm 2.2
Step 5: Call Algorithm 2.5 to update the cluster graph
        and assign macro-cluster number.
Step 6: Exit
```

For each data sample, the algorithm tries to find the desired micro-cluster using micro-cluster searching sub-algorithm (Algorithm 2.3). If no such micro-cluster exists then the master algorithm creates a new micro-cluster using the micro-cluster creation sub-algorithm (Algorithm 2.2). If the data sample lies in a micro-cluster, then the metadata of the micro-cluster is updated (Algorithm 2.4). Finally, the cluster graph is updated if any the edge list of any micro-cluster is changed or a new micro-cluster is created in the graph. The macro-cluster number is updated for each micro-cluster from the cluster graph using (Algorithm 2.5). In this way, a new data sample comes, find its cluster and update the cluster information and remove the data sample. The process eliminates the need to store the data sample which is the requirement to create an online clustering algorithm.

i Micro-cluster Initiation

The newly arrived data sample from the data stream generally falls into either in the empty space or within a micro-cluster. In case of empty space, the data sample itself creates a new micro-cluster using algorithm 2.2.

Algorithm 2.2: New Micro-cluster Initiation
This sub-algorithm initializes a new micro-cluster structure for any sample which is not inside any existing micro-clusters.
Input: Sample X_i , Minimum radius (R_{min}), Micro-cluster set, MC

```
Step 1: Set N=Maximum macro-cluster number in
        graph $MC$
Step 2: Create new micro-cluster($C$)
        C.Centre = $X_i$
        C.Radius= $R_{min}$
        C.Count = 1
        C.Macro = N+1
        C.Edge = {}
Step 3: Add $C$ to the micro-cluster set $MC$
Step 4: Exit.
```

In the micro-cluster initialization algorithm 2.2, the sample works as the Centre of the micro-cluster. As this is the first sample, so the local density variable Count is initialized to 1. The radius of micro-cluster is set to minimum radius. The identification number of the newly generated micro-cluster is considered as the macro-cluster number (or Macro) of this micro-cluster. Initially, the newly generated micro-cluster is not a stable micro-cluster at all. Thus, the micro-cluster has no intersecting micro-cluster and it does not participate in the cluster-graph structure. The micro-cluster is added to the micro-cluster set in the system.

ii Search the Micro-Cluster

When a new data sample comes from the data stream, i-CODAS searches for the micro-cluster where the data sample resides. The micro-cluster searching operation is written as the following sub-algorithm (Algorithm 2.3).

Algorithm 2.3: Micro-cluster Search
This sub-algorithm searches the desired micro-cluster for a new data sample. If the search operation is successful then it updates the micro-cluster information. **Input:**Sample X_i and micro-cluster set MC

```
Step 1: Set $d_{min}$ = the distance from sample $X_i$ to
        its nearest micro-cluster ($C$) centre
Step 2: If $d_{min}$ is less than radius of $C$ then
        Update the information of micro-cluster $C$
        by algorithm 2.3, and return true.
        End If
Step 3: Return false.
```

Initially, the Euclidean distance from the data sample to each micro-cluster centre is computed and the micro-cluster with minimum distance is also searched. If the minimum distance is less than the radius of the micro-cluster then the data sample falls in the micro-cluster with minimum distance. In case of multiple micro-clusters with minimum distance exist; one of them is selected randomly. The metadata of the micro-cluster is updated using the micro-cluster updating sub-algorithm (Algorithm 2.3). Otherwise, the sub-algorithm (Algorithm 2.4) returns false to the master algorithm (Algorithm 2.1) for notifying that the data sample does not belong to any micro-clusters.

iii Update Micro-Clusters

If the data sample falls within any micro-cluster region, then metadata of the mapped micro-cluster is updated in a fully online manner. The local density of the micro-cluster is incremented is simply by 1 as in Eq. (1). However, the micro-cluster centre is updated using Eq. (2) in case data sample falls into shell region as the data sample in kernel region has little effect on shifting the micro-cluster centre. If the data sample lies in the shell region then the micro-cluster radius is updated using Eq. (3). Suppose that, at t^{th} time instant, the local density/count, centre and radius of a cluster k are N_t^k, C_t^k and R_t^k respectively. Then at $(t+1)^{th}$ time instant, micro-cluster count, centre, and radius are updated using the recursive Eq. (1), Eq. (2) and Eq. (3) respectively.

$$N_{t+1}^k = N_t + 1 \tag{1}$$

$$C_{t+1}^k = \frac{(N_{t+1}-1)\times C_t^k + X_{t+1}^k}{N_{t+1}^k} \tag{2}$$

$$R_{t+1}^k = min\Bigg(\Big[R_t^k + \{\frac{2\times d_{X_{t+1}^k, C_{t+1}^k}}{R_t^k} - 1\}\times G_r\Big], R_{max}\Bigg) \tag{3}$$

where $G_r$is the radius growth factor.

The procedure for updating the metadata of the micro-cluster is written in the following sub-algorithm(Algorithm 2.4).

Algorithm 2.4: Update Micro Cluster
When a new data sample belongs to an existing micro-cluster, this sub-algorithm updates the information of micro-cluster.
Input: Sample, X_i and micro-cluster C

```
Step 1: Increase the sample count of the
        micro-cluster, $C$ using Eq. (1).
Step 2: If $X_i$ is in shell region of $C$ then
        Update the centre of micro-cluster $C$
        using Eq. (2).
        Update the radius of micro-cluster $C$
        using Eq. (3).
        End If
Step 3: Exit
```

Every time a new data sample falls in a micro-cluster, the micro-cluster information is updated. In the updating process, the local density or sample count of the micro-cluster is incremented using Eq. (1). The centre the micro-cluster are recursively updated if the data sample falls into the shell region of the micro-cluster using Eq. (2). The micro-cluster radius is updated based on the spatial information of data sample. The updating process is an online operation as written in Eq. (3). The micro-cluster is checked whether it has the local density equal or more than the density threshold. The micro-cluster with local density equal or above the density threshold is added to micro-cluster set and is considered for macro-cluster or cluster generation process. The edge list and macro-cluster number of the micro-cluster is updated.

iv Update Cluster Graph

Every time a new micro-cluster is added to the micro-cluster set, the cluster graph is updated by adding the new micro-cluster into graph. The cluster graph is also updated in case of relocating the centre of a micro-cluster. The procedure for updating the clustering graph is given in the following sub-algorithm (Algorithm 2.5).

Algorithm 2.5: Update Cluster Graph
This sub-algorithm executes when a new micro-clusters local density meets the minimum density threshold to become a micro-cluster to participate in cluster graph or the centre of micro-cluster is relocated.
Input: A micro-cluster,C that has been modified, Micro-cluster set, MC

```
Step 1:Update the edge list of $C$ and any
       micro-cluster intersecting $C$
Step 2:If any micro-cluster edge list is changed then
            Set a new macro-cluster number
            throughout the graph
       End If
Step 3: Exit
```

From the cluster graph updating procedure, initially the edge list of the modified or newly added micro-cluster is computed and updated based on the intersection between the micro-cluster and all other micro-cluster. Two micro-clusters are said to be intersected if the shell region intersects with the kernel region of another micro-cluster. The intersected micro-clusters are assigned to the same macro-cluster. Thus, if the edge list of any micro-cluster is modified then new macro-cluster number is assigned throughout the cluster graph with the help of Breadth First Search (BFS) algorithm.

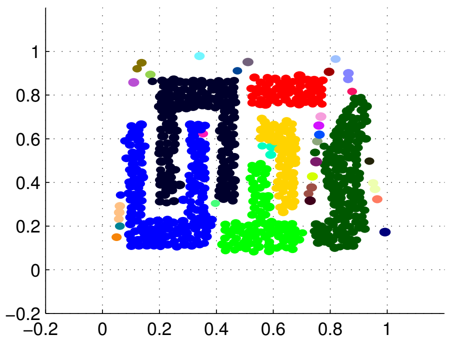
III. Experimental Result and Discussion

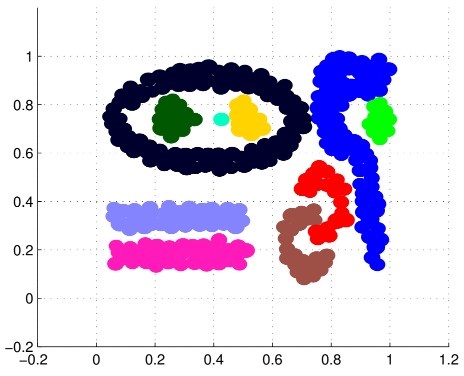
The performance of the proposed i-CODAS (improved CODAS) algorithm is analyzed and compared with existing popular clustering algorithms in this section. To analyze the performance, the i-CODAS has been coded in MATLAB R2014a and run on a run on a Core i7 processor with 8GB primary memory environment.

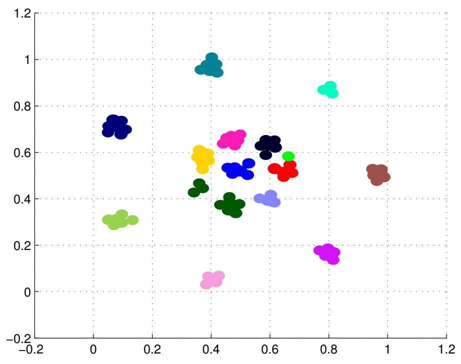
In sub-section 3.A, we demonstrate the ability of i-CODAS algorithm in generating arbitrary shaped clusters. We evaluate the noise sensitivity of i-CODAS and compare with CODAS in sub-section 3.B. In sub-section 3.C, we compute the cluster accuracy and purity, and compare with other popular clustering algorithms like ADStream, DenStream, CODAS. The memory efficiency of the proposed algorithm is also measured in sub-section 3.D. Finally, we measure the sample processing time of i-CODAS for low to high dimensional data stream and compared with other alternative clustering algorithms in sub-section 3.E. The scalability property of i-CODAS is also described in this sub-section.

A. Cluster Formation

To evaluate the ability of i-CODAS in forming clusters in dense areas, it is executed on four data sets namely DS1, DS2, R15 and Spiral data sets. The data sets are benchmark data sets which are used in several clustering algorithms [37], [38]. DS1, DS2, R15 and Spiral data sets contain 8000, 10000, 600 and 312 data samples respectively. DS1 and DS2 contain 10% noise. Naturally, DS1 has 6 clusters, DS2 has 9 clusters, R15 has 15 clusters and Spiral data set has 3 clusters. The clustering results from i-CODAS are visualized in Fig. 2(a)-2(d).


(a) DS1 data set


(b) DS2 data set


(c) R15 data set

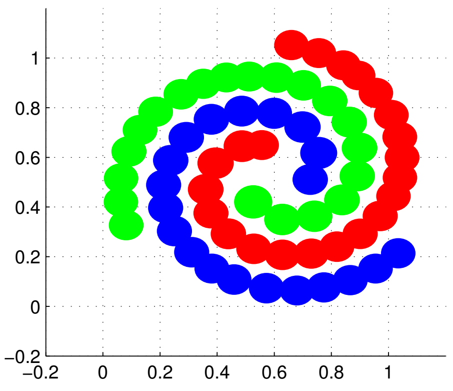

(d) Spiral data set

Fig. 2: i-CODAS clustering result on different test data sets

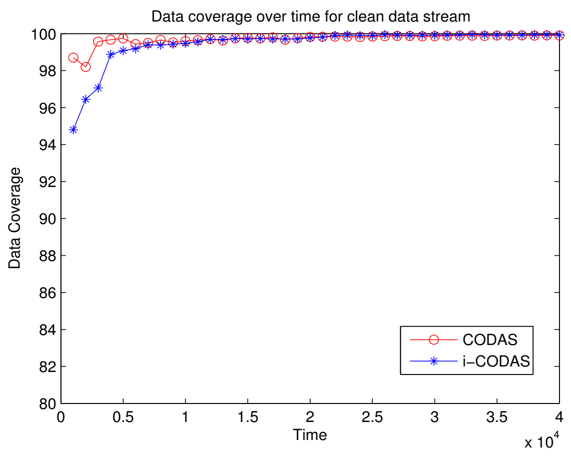
The small circles represent the micro-clusters and they joined with each other for forming clusters. In Fig. 2, the micro-clusters with same color belong to a single cluster.The data samples from each data set come sequentially to simulate an online execution of i-CODAS and after clustering, each sample is removed immediately. Fig. 2(a)-2(d) illustrates the ability of the proposed algorithm in identifying all the clusters accurately as present naturally in all the experimental data sets. In Fig. 2(a) and Fig. 2(b), some outliers are also visualized which represent the noisy samples in data sets. The figures clearly show the existence of minimum distance among the sample from a single clusters and the maximum distance among the samples from other clusters in all data sets. The well separation among clusters is clearly visible in all clustering result in Fig. 2. Though, the micro-clusters are spherical in shaped, they connects each other based on the cluster graph to form arbitrary shaped clusters. Thus, Fig. 2 confirms the formation of clusters in a natural way and the detection of noises in data set.

B. Noise Sensitivity

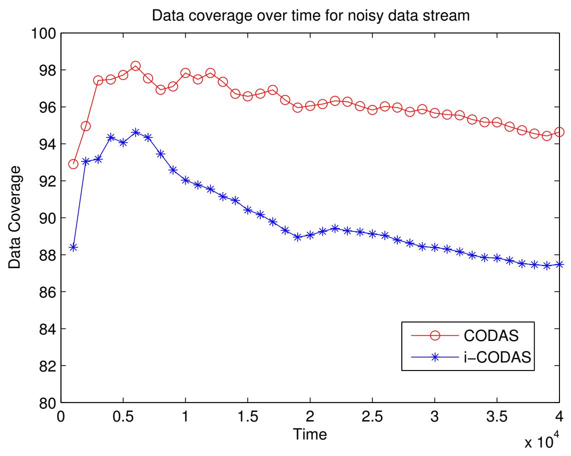
To evaluate the behavior of the proposed i-CODAS algorithms in a noisy environment, we execute it on a benchmark data stream called MacKey-Glass data stream [39], [40], [41]. Mackey-Glass data stream is a synthetic three dimensional (3D) time series which is generated using the following differential Eq. (4).

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1 + x(t-\tau)^{10} - bx(t)} \tag{4}$$

This equation is solved using 4^{th} order RungeKutta numerical method and the data stream is generated using different values for a and b for different time period t. Furthermore, the noisy Mackey-Glass data stream is generated by replacing every 5th data sample with a noisy data sample. Thus the noisy Mackey-Glass data stream contains 20% noisy data samples. To measure the noise sensitivity, we compare the data coverage or data point assignment of i-CODAS and CODAS in clean and noisy Mackey-Glass data stream. The data coverage is defined as the percentage of data samples in data stream those reside in a cluster [11]. In this experiment, the algorithm parameters namely density threshold, radius growth rate, minimum and maximum radius are set to 15, 0.001, 0.03 and 0.07 respectively. Fig. (3), shows the data coverage for clean and noisy Mackey-Glass data stream using the CODAS and i-CODAS algorithms.
In Fig. (3), the proposed i-CODAS shows an initial data coverage of approximately 95% while the existing CODAS algorithm shows nearly 98% data coverage. As new data sample lies in a micro-cluster, the micro-cluster radius is updated in an online manner towards its local optimal value. This online operation creates the fact that the micro-clusters are getting stable in terms of their radius. The stability is reflected in Fig. 3(a) where the data coverage is increased continuously and finally reaches to around 100% in the following time periods. The data coverage stays close to 100% for both CODAS and i-CODAS algorithms. In other words, all data samples are identified as true samples. On the other hand, Fig. 3(b) shows the data coverage for noisy Mac-key Glass data stream by both i-CODAS and CODAS. Unlike Fig. 3(a), both of CODAS and i-CODAS show much less data coverage as noisy Mackey-Glass data stream contains



(a) Clean data stream



(b) Noisy data stream

Fig. 3: Data coverage over time in Mac-key Glass data stream

noisy data samples with clean samples. The data coverage by i-CODAS stays below the data coverage by CODAS in all the time periods. Though the data coverage is nearly 94%, it goes below 90% as time progresses and micro-cluster radius is updated. CODAS algorithm shows over 95% data coverage in the noisy environment whereas the proposed i-CODAS shows less than approximately 88% data coverage at most of the time periods. Thus the proposed i-CODAS detects more than 60% noisy data samples whereas CODAS algorithm detects only 25% of noisy data samples. Hence, the proposed i-CODAS clustering algorithm is more efficient to detect the noisy samples than CODAS in the noisy environment.

C. Cluster Quality

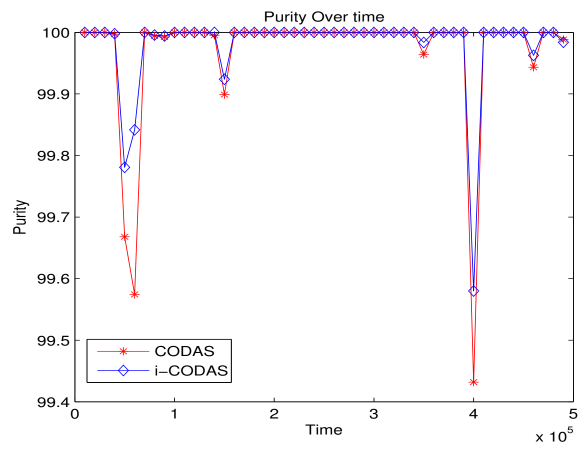
The quality of cluster is defined in terms of two parameters, cluster accuracy and purity [11]. High accuracy means high correctness whereas high purity means low errors in data sample assignment during clustering process [3]. Cluster purity cannot express the cluster quality alone. If in an application the number of low-density clusters is much higher than the high-density clusters then overall purity is high which creates the misinterpretation about clustering. If there are i n data samples in a cluster and among them, d i n samples lies in the dominant cluster then for N such clusters, the mean purity and accuracy is measured using Eq. (5) and

Eq. (6) respectively.

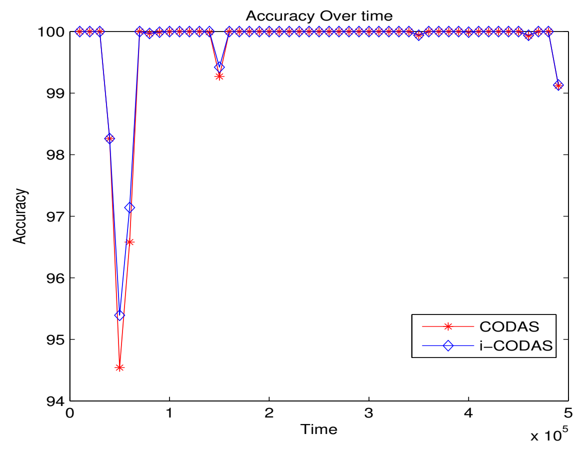
$$Purity = \frac{\sum_{i=1}^{N} n_i^d}{n_i} \tag{5}$$

$$Accuracy = \frac{\sum_{i=1}^{N} n_i^d}{\sum_{i=1}^{N} n_i} \tag{6}$$

The KDDCup99 [42] is a popular benchmark data stream for evaluating quality of clusters in the field of clustering. This data stream consists of network intrusion data packets with 41 features and 1 truth cluster. KDDCup99 data stream contains 22 types of network attacks with normal network traffic data sets. In this experiment, we use the KDDCup99 data stream to measure the cluster accuracy and purity of clusters using Eq. (5) and Eq. (6). In this experiment, the algorithm parameters namely density threshold, radius growth rate, minimum radius and maximum radius are configured as 4, 0.001, 0.07 and 0.12 respectively. The experimental results are evaluated in a window of 10000 data samples. Fig. 4 compares the mean purity and accuracy of i-CODAS with exiting CODAS clustering algorithms as CODAS is a fully online and arbitrary shaped cluster generating algorithm that outperformed the existing algorithms. In Fig. 4(a) and Fig. 4(b), both of CODAS and i-CODAS shows impressive cluster purity and accuracy.



(a) Cluster Purity



(b) Cluster Accuracy

Fig. 4: Cluster Purity and Accuracy in KKDCup99 data stream

In Fig. 4(a), the purity is 100% for both algorithms in almost all the time period except some time periods such as 50K, 170K, 400K. However, in those time periods, i-CODAS clearly produce more pure clusters than CODAS. The fact behind this improvement is that i-CODAS maintains the individual micro-cluster radius and updates it to its local optimal value while CODAS maintains a global radius for all micro-cluster that is not optimal for some micro-clusters. Following this fact, more samples are miss clustered, and as a result the purity is degraded. Similarly, the accuracy in i-CODAS algorithm is better than in CODAS algorithm. It is also seen that the accuracy is fall down to 94.54% at 50K time period, whereas the purity is 99.668% in that time period. In that time period, the number of low-density clusters is much higher than the high-density clusters and thus the overall purity is high.

To measure the numerical result, the clustering results of i-CODAS are evaluated in 50K data samples window. In each window, the purity and accuracy is computed and tabulated in Table I. The cluster purity and accuracy are compared with popular data stream clustering algorithms, ADStream, and DenStream.

TABLE I: Purity and Accuracy for KDDCup99 data stream by different algorithms

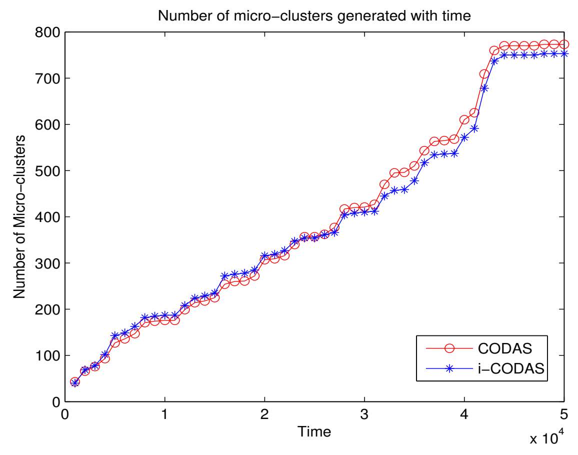
Purity(%) / Accuracy(%)		ADStream	DenStream	CODAS	i-CODAS
Time	50K	98.91	97.89	99.668	99.781
		93.65	91.43	94.54	95.39
	100K	96.37	92.43	100	100
		91.99	89.93	100	100
	150K	94.58	91.25	99.899	99.924
		93.23	90.67	99.27	99.42
	200K	93.83	89.39	100	100
		92.14	87.18	100	100
	250K	94.76	92.33	100	100
		91.27	88.41	100	100
	300K	99.21	97.89	100	100
		89.62	87.28	100	100
	350K	98.23	97.53	99.964	99.984
		93.38	88.82	99.94	99.94
	400K	90.77	89.68	99.432	99.58
		90.43	90.16	99.99	99.99
	450K	92.81	91.23	99.943	99.963
		91.78	88.57	99.93	99.94

Similar to Fig. 4, the accuracy and purity is 100% for the time period of 100K and from 200K to 300K. In other time periods, the clustering accuracy and purity value stay very close to 100% except the time period of 50K. At the time from 0 to 50K, the micro-cluster radius is developing and much drifting in data stream is found. These two facts together degrade the cluster accuracy in i-CODAS. However, the clustering accuracy is still better than the other comparing algorithms. From Table I, it is clearly found that both of the cluster purity and accuracy are highest for the proposed i-CODAS in all time period due to maintaining local optimal radius. It is clear from Fig. 3 and Table I that i-CODAS shows better performance than other algorithms in terms of cluster purity and accuracy.

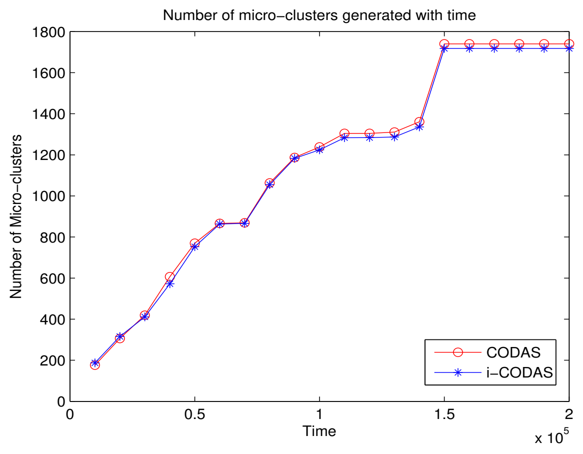
D. Memory Efficiency

The memory efficiency is measured as the storage required when clustering the data stream. The memory is required to store the cluster information, application parameters (radius and density threshold). As the radius is constant, so the memory usage is directly proportional to the number of micro-cluster.

To measure the memory usage by the clustering algorithm, the same KKDCup99 [42] data stream as Section 3.B was used. The clustering parameters are configured with the same constant as described in Section 3.B. The number of micro-clusters is computed in window of 10K data sample basis. We consider only the micro-clusters with density above the density threshold. Fig. 5(a) and Fig. 5(b) show the memory uses by CODAS and i-CODAS in terms of total number of micro-clusters for medium and long time stamp.



(a) Medium time period



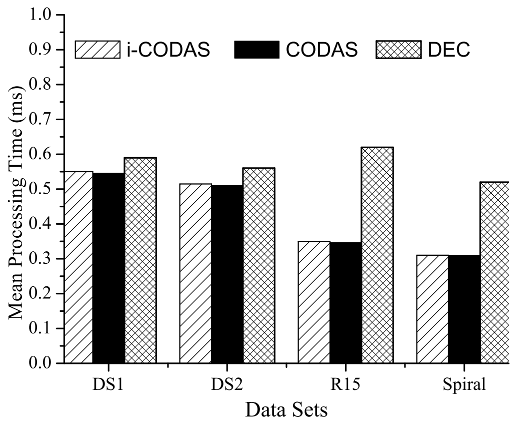
(b) Long time period

Fig. 5: Memory usage by clustering algorithm in KKDCup99 data stream

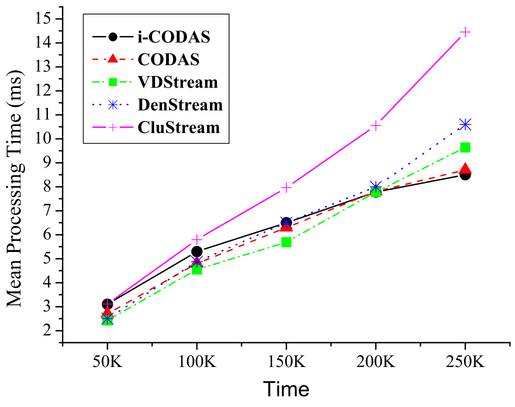
Fig. 5(a) describes that initially the number of micro-cluster is more in i-CODAS than the CODAS by a small amount as many micro-clusters have the radius lower than the optimal radius. With time the amount of micro-clusters rises for both algorithms. As time progresses new data samples fall into the micro-clusters and their radii get close to their local optimal radius and new micro-clusters are generated by a low amount. This fact is reflected in Fig. 5(a), where the number of micro-clusters in i-CODAS is lower than the micro-cluster in CODAS algorithm. The trend exists in the following time periods. The same trend is found for the micro-cluster amount measuring for long time period. It is seen from Fig. 5(b) that in some period like from 60K to 70K no new micro-cluster is generated as no drifting is detected in this window and the data samples fall into a micro-cluster in the current system. On the other hand, rapid growth in micro-cluster amount is found in some time period like from 150K to 160K where frequent drifting in data stream is detected and many new micro-clusters are generated. It is also clear from Fig. 5(b) that the number of micro-cluster is lower in i-CODAS than CODAS for long time periods.

E. Speed and dimensionality

1) Processing Time: To evaluate the processing speed, we record the total clustering time and compute the mean processing time for the data sets (DS1, DS2, R15, Spiral) used in visualizing the cluster formation (Section 3.A) and for KDDCup99 data stream. We execute these experiments for five times and compute their mean processing time. Fig. 6 compares the mean sample processing time of the proposed i-CODAS algorithm with other popular data stream clustering algorithm.



(a) DS1, DS2, R15 and Spiral data sets



(b) KDDCup99 data stream

Fig. 6: Mean processing time by different clustering algorithms

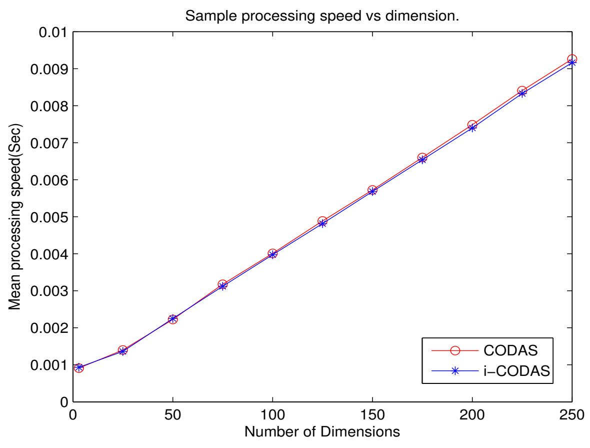
Fig. 6(a) compares the processing time with two online clustering algorithms for data stream namely CODAS and DEC. Though the data sets contains small amount of data samples, the data samples are processed in fully online manner and removed immediately processing. DEC clustering algorithm is popular for low processing time. However, it is seen from the Fig. 6(a) that the mean processing time of i-CODAS and CODAS is less than the processing time of DEC.

The data sets in Fig. 6(a) contains small amount of data samples. To measure the processing speed on data stream containing big amount of data samples, we use the KDDCup99 data stream and record the processing time in a 50000 samples per window. The mean sample processing time of i-CODAS is computed and compared with other clustering algorithms namely CODAS, VDStream, DenStream, CluStream in Fig. 6(b). In Fig. 6(b), the processing time rises as time progresses. For first 50K data samples, the processing time of i-CODAS and CluStream algorithms are higher than the processing time of VDStream, DenStream and CODAS. In this window, the radii of micro-clusters are updating in an online way, but some of them are still less than their optimal value. In the next window period the processing time of i-CODAS is near to the lowest time period which is shown by VDStream and DenStream as many of the micro-clusters has already their optimal radii. At the time period of 200K, the sample processing time of i-CODAS is lowest among the algorithms. And the processing time of i-CODAS remains lowest in the following time windows. Initially VDStream shows good processing speed, but it is not fully online algorithm. The most noticeable trend from the figure is that the mean processing time increasing rate is lower for i-CODAS algorithm than other comparing algorithm in this experiment. Thus, it can be concluded from Fig. 6 that the proposed i-CODAS algorithm.

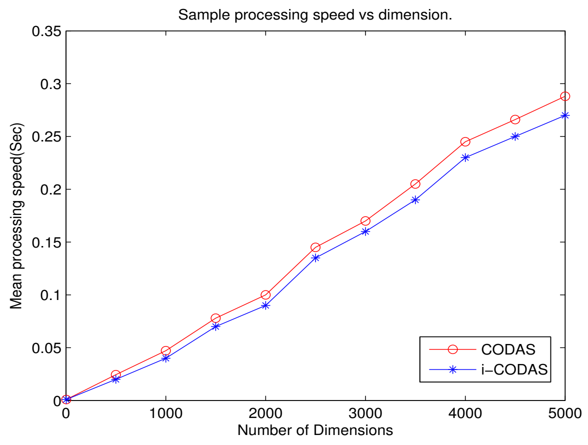
2) Scalability: To evaluate the behavior of proposed i-CODAS algorithm from low to high dimensional data stream, we use the popular helical data streams [43]. This experiment describes the scalability characteristics of i-CODAS algorithm to high dimensional data stream. The original helical data stream consists of three helical data series and generated using Eq. (7).

$$\begin{aligned} X &= rsin(t) \\ Y &= rcos(t) \\ Z &= ct \end{aligned} \tag{7}$$

The helical data stream is generated using the above time series equation for different values of t and the constant value of c. Then, the data series are moved into higher dimensional data space by adding additional data coordinates. In this application, the density threshold, radius growth rate, minimum and maximum radius are set to 4, 0.001, 0.03 and 0.07 respectively. The mean sample processing time for the low and high dimensional helical data stream is shown in Fig. 7(a) and Fig. 7(b) respectively. In the experiment, data stream with less than 250 dimensions is referred to as low dimensional data stream where as more than 250 dimensions size is referred to as high dimensional data stream. For high dimensional data stream, 5000 is considered as the maximum number of dimensions. There is a very small processing time penalty for recursively updating the micro-cluster radius in i-CODAS until the radius reaches its optimal. As seen in Fig. 4(a) and Fig. 4(b) in Section 3.C, the number of micro-clusters are more in i-CODAS than CODAS initially. However, the trend reverses after some



(a) Low dimensional data stream



(b) High dimensional data stream

Fig. 7: Mean sample processing time over low to high dimensional data stream

time where the number of micro-clusters is more in CODAS than i-CODAS. Thus, the mean sample processing time in CODAS exceeds the processing time in i-CODAS. From Fig. 7(a), the mean sample processing time is less by a small amount in i-CODAS than CODAS for low dimensional data stream. As the dimension size increases, so the mean processing time increases. The difference between the sample processing time in CODAS and i-CODAS is not significant for low dimensional data stream. However, the average sample processing time is larger by a significant amount in CODAS than i-CODAS for high dimensional data stream. As the dimension increases, this difference is noticeable in Fig. 7(b). This cause behind this trend is that number of micro-cluster is more in CODAS than i-CODAS, though the number of macro-cluster is quite same in both algorithms. Though this time penalty does not affect in CEDAS so much for the low dimensional data stream, they are significant in the high dimensional data stream. Thus, for the high dimensional data stream, the mean sample processing time is more in CODAS than i-CODAS. This experiment explains the good scalability property of the proposed i-CODAS algorithm.

IV. Conclusion

Data stream clustering has become one of the hot topics to research for scientist recently. In this article, we present an improved version of recently proposed CODAS clustering algorithm. The proposed i-CODAS algorithm offers a fully online clustering of data stream. i-CODAS algorithm removes the requirement of setting the global and optimal micro-cluster radius. Instead, it maintains the local optimal radius for each micro-cluster independently. The micro-cluster radius is updated towards its local optimal value using an online operation. In Section 2, the proposed algorithm is described in details. We evaluate i-CODAS clustering algorithm to demonstrate its ability to form arbitrary shaped clusters and detect noise. The algorithm is compared with other clustering algorithms with respect to cluster quality, memory usage, processing speed and scalability. The experimental result shows that i-CODAS can detect arbitrary shaped cluster in highly dense region which are separated by sparse regions. The algorithm shows superior performance for identifying the noisy sample than the existing algorithms in noisy environment. The efficient formation of micro-cluster contributes to show excellent cluster quality in terms of purity and accuracy comparing to other popular data stream clustering algorithms. The memory efficiency experiment shows that i-CODAS require less memory to complete its clustering process comparing to other clustering algorithm. The proposed algorithm shows more sample processing speed and more scalability performance when comparing to other clustering algorithms like VDStream, DenStream, DEC, CODAS.

In this study, the proposed clustering algorithm is an online approach for clustering the data stream into arbitrarily shaped with high accuracy, purity and noise sensitivity. This algorithm is scalable to high-dimensional data stream. Future research could reduce the memory requirement by a significant amount without deteriorating other clustering performance criteria.

Acknowledgment

The work reported in this paper is funded by Fundamental Research Grant from Ministry of Higher Education (MoHE), Malaysia titled ”A Reinforcement Learning Sine Cosine based Strategy for Combinatorial Test Suite Generation” (Grant no RDU170103). We thank MoHE, Malaysia for the contribution and support.

References

[1] D. Y. Kim, S. T. Park and M. H. Ko, ”A study on the analysis of IT-related occupational cluster using big data,” *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 7-11, 2018.

[2] K. Abdalgader, ”Clustering short text using a centroid-based lexical clustering algorithm,” *IAENG International Journal of Computer Science*, vol. 44, no. 4, pp. 523-536, 2017.

[3] Q. Duan, Y. L. Yang and Y. Li, ”Rough k-modes clustering algorithm based on entropy,” *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 13-18, 2017.

[4] M. Ghesmoune, M. Lebbah and H. Azzag, ”State-of-the-art on clustering data streams,” *Big Data Analytics*, vol. 1, no. 1, pp. 13, 2016.

[5] F. Zhao, Y. Yang and W. Zhao, ”Adaptive clustering algorithm based on max-min distance and bayesian decision theory,” *IAENG International Journal of Computer Science*, vol. 44, no. 2, pp. 180-187, 2017.

[6] L. F Zhu and J. S Wang, ”Data clustering method based on bat algorithm and parameters optimization,” *Engineering Letters*, vol. 27, no. 1, pp. 241-250, 2019.

[7] S. Sen, S. Narasimhan and A. Konar, ”Biological data mining for genomic clustering using unsupervised neural learning,” *Engineering Letters*, vol. 14, no. 2, pp. 61-71, 2007.

[8] K. Deshmukh and G. Shinde, ”Adaptive color image segmentation using fuzzy min-max clustering,” *Engineering Letters*, vol. 13, no. 2, pp. 57-64, 2006.

[9] S. Mansalis, E. Ntoutsi, N. Pelekis and Y. Theodoridis, ”An evaluation of data stream clustering algorithms,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 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 Computing Surveys*, vol. 46, no. 1, pp. 13, 2013.

[11] R. Hyde and P. Angelov, ”A new online clustering approach for data in arbitrary shaped clusters,” in *Proc.* 2^nd *IEEE International Conference on Cybernetics 2015*, pp. 228-223.

[12] H. Li, R. Liu, J. Wang and Q. Wu, ”An enhanced and efficient clustering algorithm for large data using MapReduce,” *IAENG International Journal of Computer Science*, vol. 46, no. 1, pp. 61-67, 2019.

[13] H. L. Nguyen, Y. K. Woon and W. K. Ng, ”A survey on data stream clustering and classification,” *Knowledge and Information Systems*, vol. 45, no. 3, pp. 535-569, 2015.

[14] A. Amini, T. Y. Wah and H. Saboohi, ”On density-based data streams clustering algorithms: A survey,”*Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 116-141, 2014.

[15] 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.* 2^{nd} *International Conference on Knowledge Discovery and Data Mining 1996*, pp. 226-231.

[16] T. Ali, S. Asghar and N. A. Sajid,”Critical analysis of DBSCAN variations,” in *Proc. International Conference on Information and Emerging Technologies 2010*, pp. 1-6.

[17] J. Yang, Q. Wu, Z. Qu and Z. Liu, ”An enhanced density clustering algorithm for datasets with complex structures,” *IAENG International Journal of Computer Science*, vol. 44, no. 2, pp. 150-156, 2017.

[18] C. C. Aggarwal, S. Y. Philip, J. Han and J. Wang, ”A framework for clustering evolving data streams,” in *Proc.* 29^{th} *International Conference on Very Large Databases 2003*, pp. 81-92.

[19] F. Cao, M. Estert, W. Qian and A. Zhou, ”Density-based clustering over an evolving data stream with noise,” in *Proc. SIAM International Conference on Data Mining 2006*, pp. 328-339.

[20] C. Ruiz, E. Menasalvas and M. Spiliopoulou, ”C-DENSTREAM: Using domain knowledge on a data stream,” in *Proc. International Conference on Discovery Science”*, pp. 287-301.

[21] L. X. Liu, Y. F. Guo, J. Kang and H. Huang, ”A three-step clustering algorithm over an evolving data stream,” in *Proc. IEEE International Conference on Intelligent Computing and Intelligent Systems 2009*, pp. 160-164.

[22] J. Ren and R. Ma, ”Density-based data streams clustering over sliding windows,” in *Proc.* 6^th *International Conference on Fuzzy Systems and Knowledge Discovery 2009*, pp. 248-252.

[23] J. Lin and H. Lin, ”A density-based clustering over evolving heterogeneous data stream,” in *Proc. ISECS International Colloquium on Computing, Communication, Control, and Management 2009*, pp. 275-277.

[24] I. Ntoutsi, A. Zimek, T. Palpanas, P. Krger and H. P. Kriegel, ”Density-based projected clustering over high dimensional data streams,” in *Proc. SIAM International Conference on Data Mining 2012*, pp. 987-998.

[25] N. Su, J. Liu, C. Yan, T. Liu and X. An, ”An arbitrary shape clustering algorithm over variable density data streams,” *Journal of Algorithms and Computational Technology*, vol. 11, no. 1, pp. 93-99, 2017.

[26] M. Hassani, P. Spaus, M. M. Gaber and T. Seidl, ”Density-based projected clustering of data streams,” in *Proc. International Conference on Scalable Uncertainty Management 2012*, pp. 311-324.

[27] C. Isaksson, M. H. Dunham and M. Hahsler, ”SOStream: Self organizing density-based clustering over data stream,” in *Proc. International Workshop on Machine Learning and Data Mining in Pattern Recognition 2012*, pp. 264-278.

[28] B. J. Frey and D. Dueck, ”Clustering by passing messages between data points,” *Science*, vol. 315, no. 5814, pp. 972-976, 2007.

[29] J. P. Zhang, F. C. Chen, L. X. Liu and S. M. Li, ”Online stream clustering using density and affinity propagation algorithm,” in *Proc.* 4^{th} *Proc. IEEE International Conference on Software Engineering and Service Science 2013*, pp. 828-832.

[30] S. Ding, J. Zhang, H. Jia and J. Qian, ”An adaptive density data stream clustering algorithm,” *Cognitive Computation*, vol. 8, no. 1, pp. 30-38, 2016.

[31] A. Forestiero, C. Pizzuti and G. Spezzano, ”A single pass algorithm for clustering evolving data streams based on swarm intelligence,” *Data Mining and Knowledge Discovery*, vol. 26, no. 1, pp. 1-26, 2013.

[32] C. Fahy, S. Yang and M. A. Gongora, ”Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams,” *IEEE Transactions on Cybernetics*, vol. 49, no. 6, pp. 2215 - 2228, 2018.

[33] R. D. Baruah and P. Angelov, ”Evolving local means method for clustering of streaming data,” in *Proc. IEEE International Conference on Fuzzy Systems 2012*, pp. 1-8.

[34] R. D. Baruah and P. Angelov,”DEC: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models,” *IEEE Transactions on Cybernetics*, vol. 44, no. 9, pp. 1619-1631, 2014.

[35] D. Puschmann, P. Barnaghi and R. Tafazolli, ”Adaptive clustering for dynamic IoT data streams,” *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 64-74, 2017.

[36] V. S. Moertini, G. W. Suarjana, L. Venica and G. Karya, ”Big data reduction technique using parallel hierarchical agglomerative clustering,” *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 188-205, 2018.

[37] J. Hou and A. Zhang, ”Enhanced dominant sets clustering by cluster expansion,” *IEEE Access*, vol. 6, pp. 8916-8924, 2018.

[38] L. Jiang and D. Xie, ”An efficient differential memetic algorithm for clustering problem,” *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 118-129, 2018.

[39] M. K.Islam, M. M. Ahmed and K. Z. Zamli, ”A buffer-based online clustering for evolving data stream,” *Information Sciences*, vol. 489, pp. 113-135, 2019.

[40] L. Glass and M. Mackey, ”Mackey-glass equation,” *Scholarpedia*, vol. 5, no. 3, pp. 6908, 2010.

[41] R. Hyde, P. Angelov and A. R. MacKenzie, ”Fully online clustering of evolving data streams into arbitrarily shaped clusters,” *Information Sciences*, vol. 382, pp. 96-114, 2017.

[42] S. D. Bay, D. Kibler, M. J. Pazzani and P. Smyth, ”The UCI KDD archive of large data sets for data mining research and experimentation,” *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 2, pp. 81-85, 2000.

[43] H. Steinhaus, *Mathematical snapshots*, 3rd ed., New York: Dover: Courier Corporation, 1999.