International Journal of Intelligent Computing and Information Science

SCLUSTREAM: AN EFFICIENT ALGORITHM FOR TRACKING CLUSTERS OVER SLIDING WINDOW IN BIG DATA STREAMING

Doaa.Sayed
Computer Science Department, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
Doaa.ahmed74@yahoo.com

Sherine.Rady
Information Systems Department, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
srady@cis.asu.edu.eg

Mostafa.Aref
Computer Science Department, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
Mostafa.aref@cis.asu.edu.eg

Abstract: *Mining in data streams has been a hot research topic in the recent time. A main challenge in data stream mining lies in extracting knowledge in real time from a massive, dynamic data stream in only a single scan. Data stream clustering presents an important role in data stream processing. This paper proposes SCluStream an algorithm for tracking clusters over a sliding window to handle such challenges. The algorithm is an enhancement over CluStream which does not involve this sliding window concept. In the sliding window model, only the most recent data is used while the old data is eliminated, which allows for faster execution. A better clustering technique is also involved which managed to contribute to accuracy enhancement. The proposed algorithm has been tested on a dataset for Intrusion detection and the results showed that comparing SCluStream to CluStream has proven that the former algorithm is more efficient for online clusters generation for big data streaming in regard of the accuracy as well as the utilized time and memory resources.*

Keywords: Data stream mining; Data stream clustering; Time series in big data; Window models; sliding window.

1 Introduction

Data streams are infinite data that continuously evolves over time. Huge amounts of data streams are currently generated in real time through many applications. Financial transactions, sensor networks, telecommunications, website analysis, weather monitoring, and e-business are examples of applications that generate huge volume of data streams. Data streams show a unique set of several challenging characteristics such as: the features of data streams evolve over time and data streams arrive in continuously in limited time, which require a fast real-time response. Data streams are also generated rapidly, which both require theoretically unlimited memory. Data streams need to be processed in only one scan because data streams can be used mostly only once after which it is eliminated. Clustering is one of the most important data mining techniques, which aims to analyze and discover knowledge from continuous data streams. Data stream clustering presents some challenges; to meet real-time requirements. The clustering needs to be executed in a short time frame with limited memory using a single-scan process. So tracking clusters in the sliding window is a main target to handle in data streams and remains a challenge to handle while considering the restricted time and memory resources.

Clustering in a stream environment depends on the moments they are determined as well as on the time window or horizon over which they are measured. Windowing [1] is one popular processing method that is used with streaming of data. Data stream can be split into finite sets, or windows, based on the time dimension. There are three different kinds of window models for tracking evolving data streams [1]: 1) Landmark Window Models, 2) Damped Window Models and 3) Sliding Window Models. In Landmark window the window is determined by a specific time point called landmark and the present. It is used for mining over the entire history of the data streams. This model is not suitable for applications where recent information is more important than the past information. Data stream clustering algorithms that are implemented under the landmark model are CluStream ,HPStream , StreamingKM++ and BIRCH [15]. In the damping model, weights are assigned to data, where higher weights are given to most recent data items and less weight are given to the older data. This means that the model assumes the most recent data as more important than the older data. Data stream clustering algorithms that are implemented under the damped model are DenStream, HDenStream, D-Stream, E-Stream and HUE-Stream [15]. Sliding window models assume that recent data is more relevant than older data by using accordingly, a maintained window or horizon. The most recent instances falling within the window of the data stream are kept while discarding older data. The need to use the sliding window rather than landmark or damped model is often critically required for other purposes, such as to save more time or memory resource. Data stream clustering algorithms that are implemented under the sliding model are SWClustering [16] and SDStream [15].

This paper proposes SCluStream, an improvement of CluStream algorithm, for tracking clusters. SCluStream tracks clusters in three phases online, expiring and offline. In the online phase, SCluStream tracks clusters over a sliding window for focusing on the most recent data to handle the pre-mentioned data streams challenges. In other words, SCluStream can handle and analyze data streams in relatively short time. In addition, SCluStream can track well the evolving of data over time. CluStream uses landmark window, where the equal importance is given to the whole data, so apparently it is incapable of handling data streams challenges in regard of time and memory. In the expiring phase of the proposed SCluStream, the old data is eliminated so less memory is consumed. In the offline phase, CluStream uses k-mean algorithm, which needs initialization and mostly may lead to produce inaccurate results as reported by some works [14]. SCluStream uses k-means++ instead of k-means used by CluStream in order to improve the accuracy of results.

The organization of this paper is as follows: in section 2 related works about data streams clustering algorithms are discussed. In section 3, SCluStream algorithm, an improvement for CluStream is discussed. In section 4 the experimental study and evaluations are discussed. In the last section conclusion and future work are highlighted.

2 Related work

This section discusses the state of the art of the data stream clustering algorithms. A clustering algorithm aims at finding the number of clusters ‘k’ that mostly represents the data stream distribution. Among those clustering algorithms are Clustream, HPStream, STREAMKM++, DenStream, SDStream, HDenStream, BIRCH, E-Stream, HUE-Stream, D-Stream.

CluStream [2] is popular algorithm in data stream as it employs a very unique online-offline framework. CluStream is divided into two phase’s strategy. In the online- phase, the framework analyzes the streaming data and stores its summary statistics using micro-clusters. Micro-clusters are the continuous evolving of the clustering stored with two parameters: time and square of time. Micro-clusters are saved periodically at time snapshots which usually take pyramidal pattern. In the offline phase, the framework uses the data summary in micro-clusters to deduce the final clusters. CluStream algorithm can determine clusters over time horizon, since it detects the evolution of data at any time. This algorithm uses landmark window so its biggest limitation is that the radius of clustering continuously increases with the streaming of data, and as it doesn't eliminate “old data” online, more and more data will increase over time [3].

HPStream [4] was developed as an extension to CluStream. This algorithm is a projected clustering for high-dimensional streaming data. The primary motivation behind this extension is that CluStream does not perform efficiently when applied to a high-dimensional data stream. However the HP-Stream algorithm is capable of handling high-dimensional cases, the difficulty lies in obtaining an appropriate average projection dimension [5]. STREAMKM++ [18] is a type of k-means algorithm that is suitable for clustering data streams from an Euclidean space. In terms of running time, this algorithm is faster than CluStream but CluStream is preferable than STREAMKM++ in terms of quality [5].

DenStream [6] is an algorithm based on a density method .This algorithm uses two-phases similar to CluStream. In the first phase, the algorithm uses the fading window model to create a synopsis of the data. Then, in the second phase, the stored synopsis of the data is utilized by DBSCAN [17] in an offline phase to provide the final clustering result. The limitation of this algorithm is the consumed time in the pruning phase for deleting outliers and noises.

rDenStream [7] is the extension to DenStream algorithm. It consists of three phases; the first two phases are the same as DenStream but a new phase is added, called outlier retrospect. rDenStream means DenStream with retrospect. In DenStream outlier micro clusters are neglected while in rDenStream they are stored in a historical buffer. In the retrospect phase, these outlier micro clusters are re- identified. If the weight of some outlier micro clusters increases with time then retrospect phase gives a chance to form potential micro clusters from these outliers’ clusters. However, if the weight of the outlier micro cluster is more than a certain threshold, then they are discarded. This algorithm is more accurate than DenStream, but needs more time for processing historical data and more memory for storing the historical outlier buffer.

HDenStream [9] is another algorithm based on density methods. It is an extension for DenStream for clustering high dimensional data streams. This algorithm has the same DenStream algorithm phases. In the online phase the summary of the data is kept and in offline phase defines the final cluster based on given projected cluster. The distances between each pair categorical attributes and continuous attributes are computed separately. The algorithm suffers from keeping categorical features for data streams in an effective way. SDStream [8] is another algorithm similar to DenStream but uses a sliding window model to define the clusters in the most recent data. The old data in sliding window are deleted.

BIRCH [10] was originally designed to mine traditional data. BIRCH has two steps: in the first step, the algorithm scans the data and then creates a tree consisting of information regarding data clusters. In the second step, BIRCH cleans the tree by eliminating sparse nodes (outliers) and generates new original clusters. The algorithm can deal with large amount of data. The major limitation of Birch is the defined size of the tree by user in advance. In addition, this algorithm can't detect concept drift (the characteristics of data may change over time.).

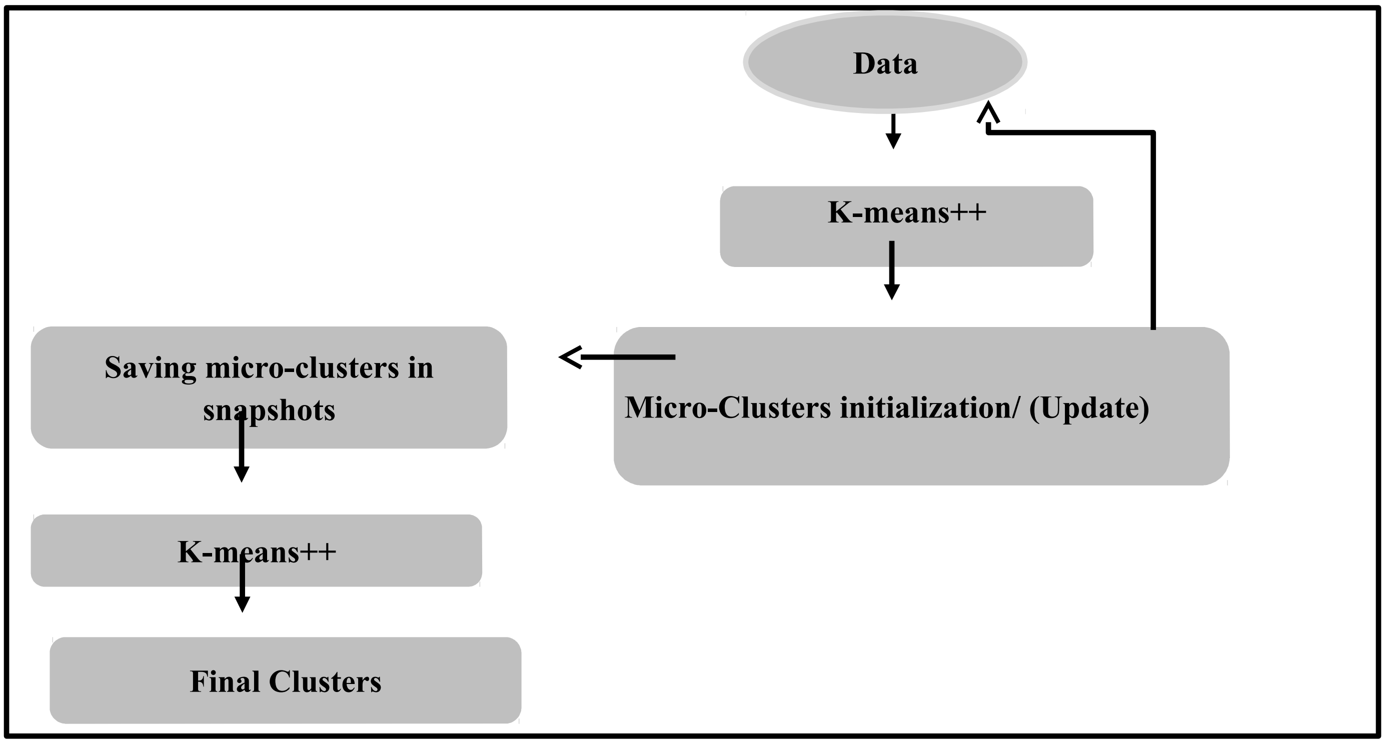
The E-Stream [11] is an evolutionary algorithm, that deals with different kinds of evolution: (i) the appearance of a new cluster by agglomerating enough points in an area, (ii) the disappearance of existing clusters by considering the fading structure, (iii) the evolution of a cluster by changing the behaviour of data, (iv) the merging of a pair of similar clusters, and (v) the splitting of a cluster into two sub-clusters. The E-Stream has a polynomial runtime (O(k2)) with regard to the number of clusters in the merging process.

HUE-Stream [12] is an improvement of E-Stream for clustering high dimensional streaming data. Uncertainty is handled by a distance function with a probability distribution of two data objects. It also merges and splits data like E-Stream the merging of clusters is done by using the proposed distance function and the closest cluster of a given incoming data is determined. Splitting clusters is done by using a suggested histogram management for categorical data. A fading function is used by E-Stream to minimize the weight of old data over time.

D-Stream [13] is an algorithm which uses online component and offline component similar to CluStream. In the online component, each incoming data point is mapped into a grid which corresponds to the dimension of the data. In the offline component the grids are clustered based on their density after the density of each grid is computed. The algorithm uses a decay factor to determine which data are recent (more important) and which are old (less important). This is sensitive towards handling high dimensional data but it is able to handle outliers and noises.

3 SCluStream algorithm

SCluStream is a proposed improvement for CluStream algorithm. The letter 'S' in SCluStream refers to an introduced sliding window concept. Besides, the sliding window, SCluStream proposes some improvements for the online and offline phases of CluStream to contribute to saving in memory and time resources and increasing the accuracy of clustering. In the online phase, saving in space and time are achieved by clustering data over a sliding window, additionally, an expiring phase is included where expiring snapshots are deleted from sliding window frame. In the offline phase, k-mean++ clustering algorithm is used rather than traditional k-means to generate more accurate results. Initial centroids are determined based on systematic method using probability distribution. The block diagram in figure 1 represents the principal blocks of SCluStream and the relationships between these blocks. The representation of sequence steps are described by the pseudo code shown in figure 2. SCluStream consist of 3 phases described in the following:



Data
K-means++
Micro-Clusters initialization/ (Update)
Saving micro-clusters in snapshots
K-means++
Final Clusters

Figure 1: Block diagram illustrating how SCluStream works.

Inputs

Micro-clusters number *q*, initial number of data points *n*, number of numerical attributes *d*, time window w_t

Output *K* final clusters from micro clusters

```
/* Initialization */
1: Initialize number of micro –clusters
/*online phase*/
2: for each data point $x$ arrive within time $w_t$ from data stream
3:   Find nearest micro clusters $c$
4:      if $D$ (distance from $x$ to $c$) $\leq$ Maximum boundary of micro-cluster.
6:              Merge $x$ to $c$
7:     else
8:        Create new micro –cluster to absorb $x$
9:       micro –clusters are stored in snapshots of clusters which change when new points arrive.
/* expiring phase*/
10:  Check if exist expiring snapshots from sliding window
11:    Delete all expiring snapshots from expiring window $w_t$ .
/*Offline phase*/
13:   Determine number of final clusters $K$, time to get micro-clusters from snapshots which stored
        within specific time
14:     Apply k-means ++
15:           choose first center $c$ as a closest center randomly from data stream
16:           for i=2 to $K$
16:            compute distance from a data point to closest center $D(x)$.
17:            choose the next center $c_i$ using a weighted probability distribution where a point $x$ is
               chosen with probability proportional to the square distance from point to first center $D(x)^2$.
18:         end for
19:       apply standard k-means to generate final clusters
```

```
18:         end
```

Figure 2: Pseudo code for SCluStream.

➢ **Online phase**

In this phase, the size of window is determined. The window time w_t or horizon is maintained for keeping the most recent instances falling within the window size of the data stream while discarding old data. Statistical information about the data locality in terms micro-clusters are maintained. The micro-cluster structure is a temporal extension of the cluster feature vector (CF). A micro-cluster is a tuple having the form {N, LS, SS, LST, SST} where {N, LS, SS} are the three components of the CF vector, namely, the number of data points in the cluster, the linear sum of the N data points, and the squared sum of the N data points. The two other LST and SST are the sum and the sum of the squares of the time stamps of the N data points. The incremental and additive properties of the micro-clusters make them a natural choice for the data stream problems. By incrementally, it means that the CF is updated by adding a new arriving data point x, while by additive it means that two disjoint CFs can be merged into a new CF by adding their components. Figure 3 shown an illustration how a micro cluster structured.

Incrementally	Additively
LS=LS+ x	LS= LS_1+LS_2
SS=SS+ x^2	SS= SS_1+SS_2
N=N+1	N= N_1+N_2

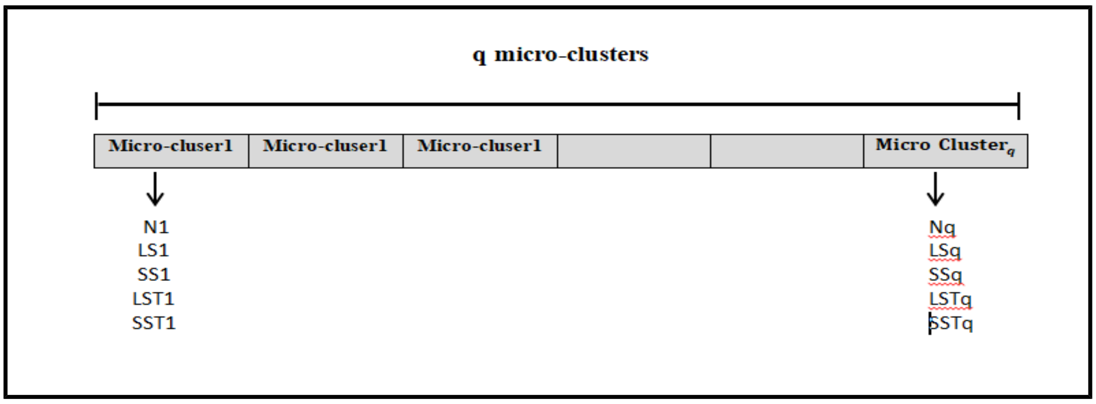


Figure 3: Micro cluster structure

In the online phase, and at the start of the algorithm execution, the k-means ++ is executed to create q initial micro-clusters from the stream files arriving within the time window w_t . Upon new data point arrival one of the following two alternatives occurs:-

i) The data point is absorbed by one of the existing micro-clusters. This absorption is based on the closeness of the cluster to the data point; this is generally calculated using a distance metric between the centroid of the micro-cluster and the data point. Accordingly, the data point is absorbed to the nearest micro cluster using the additive property.

ii) The data point is placed in its own new micro-cluster, but with the constraint that the number of micro clusters remains fixed. Thus, the number of the existing clusters needs to be reduced by one, which can be achieved by either deleting one of the older micro-clusters or merging two closest micro clusters together.

➢ **Expiring Phase**

The micro clusters are stored at snapshots at every time slot within a time window w_t. Figure 4 illustrates an example which shows several files at different time slot. Expiring snapshots are those snapshots, whose time stamp is less than the current time minus window time and those expiring from the sliding window will be eliminated.

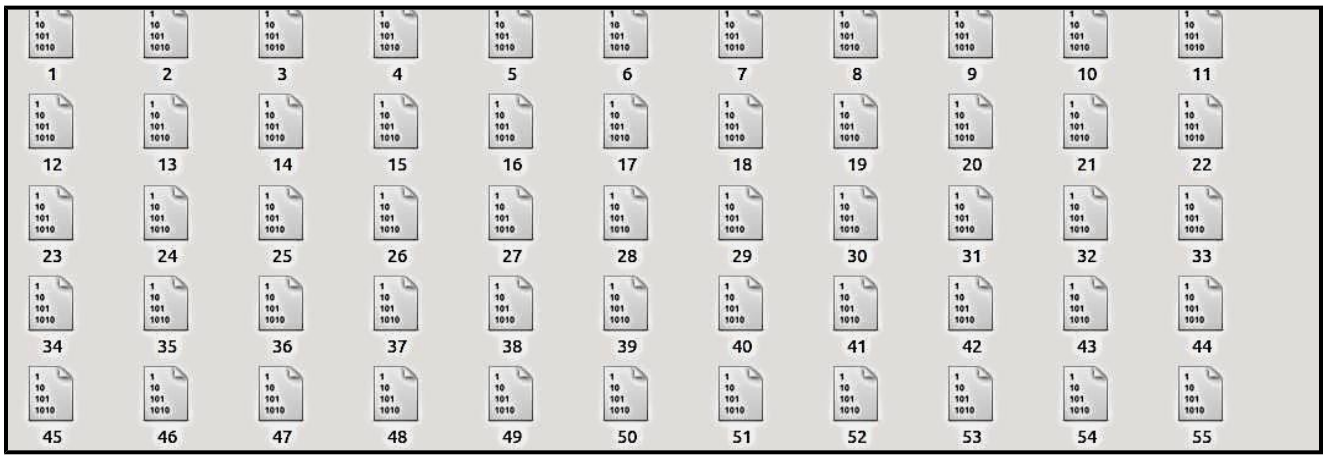


Figure4: Snapshots

➢ **Offline Phase**

The synopsis information stored in the micro-clusters is used by the offline phase to deduce the final macro clusters. The offline phase gets the micro clusters from saved snapshots within the time window w_t. The offline phase is dependent on many user inputs which are defined in advance such as the number of macro-clusters and time horizon of length (h), the micro- clusters which are used are specific to that time-horizon. The final macro clusters are defined by using k-means++ rather than traditional k-means. In k-means++ the first centroid is chosen uniformly at random from data points , after the next centroids from data point with probability proportional to its squared distance from the point's closest existing center $D(x)^2$.

4 Experimental study

4.1 Preprocessing steps

Before running algorithm, some preprocessing steps are executed which includes the following:

- ✓ Only numerical data are extracted from files because SCluStream works with numerical attributes only.
- ✓ A big data set file is divided into different chunks files.
- ✓ Number of attributes (dimensions) d is defined.
- ✓ Number of data points n to start the online phase is defined.
- ✓ Number of initial micro-clusters q is defined.

4.2 Experimental setup

All experiments have been run on a laptop with a processor core I5, 8G memory and using (Ubuntu 64-bit). The implementation was done in apache spark using scala 2.10. The environment is prepared such that spark streaming reads data streams from incoming files sequences and produces results in the file system (in the form of snapshots). The dataset used for testing is the network intrusion detection data set (KDD-CUP'99) which consists of 494,021 records and 43 attributes. The network intrusion detection data is a series of TCP connection records within specific period from local area network. Each record either belongs to a normal connection or an intrusion or attack. In the experiments, only the numerical attributes are used (31 attributes). The experiment aims at evaluating the clustering quality and performance of the proposed modified version of CluStream algorithm (SCluStream) for intrusion detection. The traditional CluStream is implemented in apache spark to compare SCluStream with traditional CluStream on the same platform. The experimentation has been done by using the same parameter settings as in the reference paper [2]. The initial number of points n is 2000; the number of micro clusters q is set to 50 and the number of dimensions is set to 31 attributes. The number of micro-clusters q and the number of dimensions d will change during experiments for test that stability and scalability of results.

4.3 Clustering Evaluation

A) Clustering Quality

The quality of clustering is measured by using the sum of squared distance (SSQ). The distance D between data point x_i and the nearest centroid C_{xi} is calculated $D(x_i, C_{xi})$. The SSQ is calculated as equal to the sum of $D^2 = (x_i, C_{xi})$ for all points in current window.

$$x$$

$$¿$$

$$(¿i,¿C_{xi})$$

$$D^2¿$$

$$SSQ=\sum_{i=0}^{n}¿$$

To confirm the accuracy of the results, the clustering quality of SCluStream is compared with that of CluStream for different window sizes, different files size, and different number of micro clusters.

B) Clustering Speed

The speed is measured by the execution time, which is the elapsed time for processing batch of data during window time w_t .

C) Clustering Scalability

The scalability is evaluated by two different measures 1) Execution time 2) Memory usage with respect to several factors, which are: the number of micro clusters, data dimensionality and number of data points.

- ✓ **Execution time**

The execution time of SCluStream is evaluated on data stream with various dimensionality and different number of micro clusters.

- ✓ **Memory usage**

The memory usage is measured by the size of snapshots from micro clusters which are stored in disk. SCluStream uses sliding window so expiring snapshots from window size are deleted, which is not the case with CluStream where no elimination is done for snapshots.

4.3 Experimental results

Figure 5 illustrates the comparison between the clustering quality (SSQ) of SCluStream and that of CluStream for different files size at the same window time (70 *s*) with the same parameters in experimental setting (Initial points 2000, No.of micro clusters (*q*=50)). The figure obviously shows that SCluStream is more accurate than CluStream at all the experimented different files sizes. The average SSQ of SCluStream is about five orders of magnitude smaller than that of CluStream.

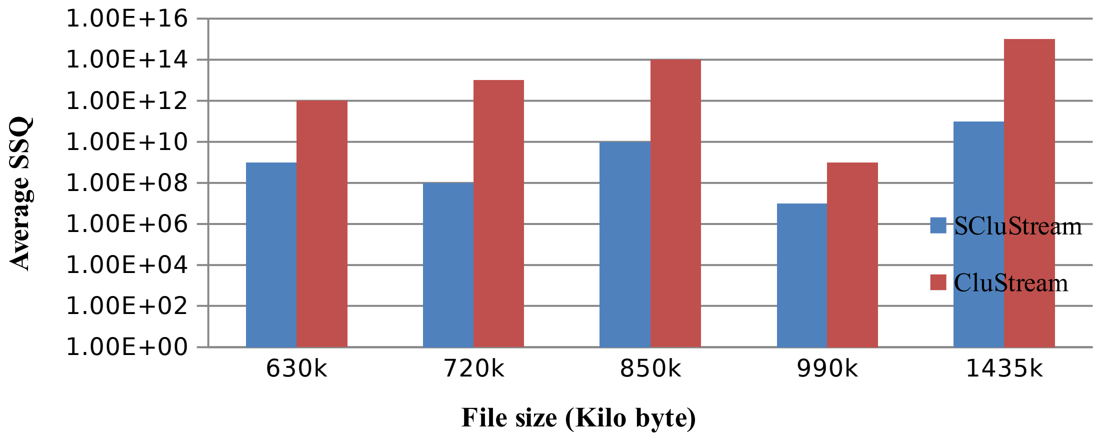


Figure 5: Clustering Quality (SSQ) of SCluStream and CluStream at different file sizes.

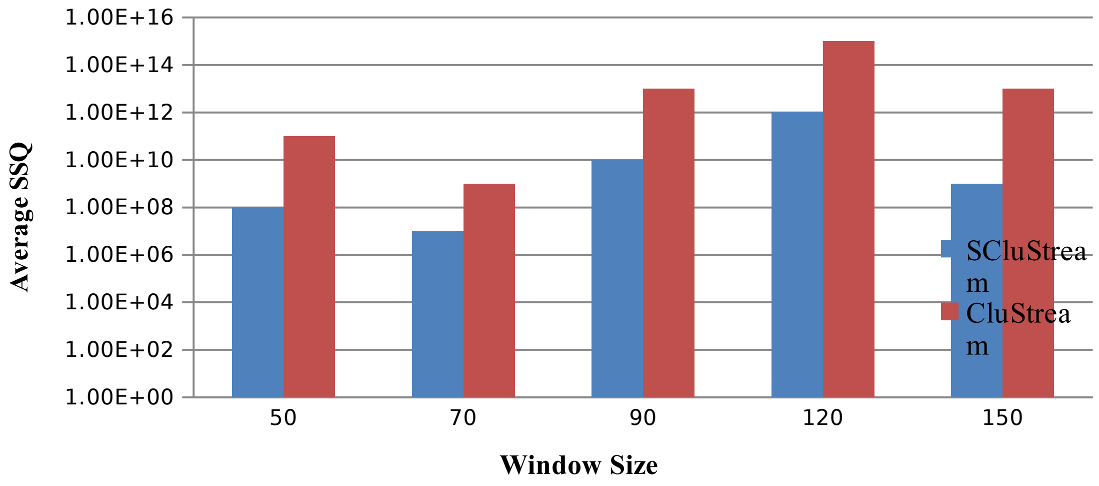


Figure 6: Clustering Quality (SSQ) of SCluStream and CluStream at different window sizes

Figure 6 illustrates the clustering quality (SSQ) of SCluStream and that of CluStream for different window sizes w_t with the same file size and while fixing initial number of points and number of micro clusters. The figure shows that the SSQ of SCluStream is better than that of CluStream at all different window sizes with the same stream speed (Number of points in batch). The average SSQ of SCluStream is about four orders of magnitude smaller than that of CluStream.

Figure 7 illustrate the clustering quality (SSQ) of SCluStream and that of CluStream with for different number of clusters and while fixing the stream speed (number of points per batch) and the window size w_t . The figure also shows that the clustering quality for SCluStream is better than that of CluStream, and the average SSQ of SCluStream is about four orders of magnitude smaller than that of CluStream.

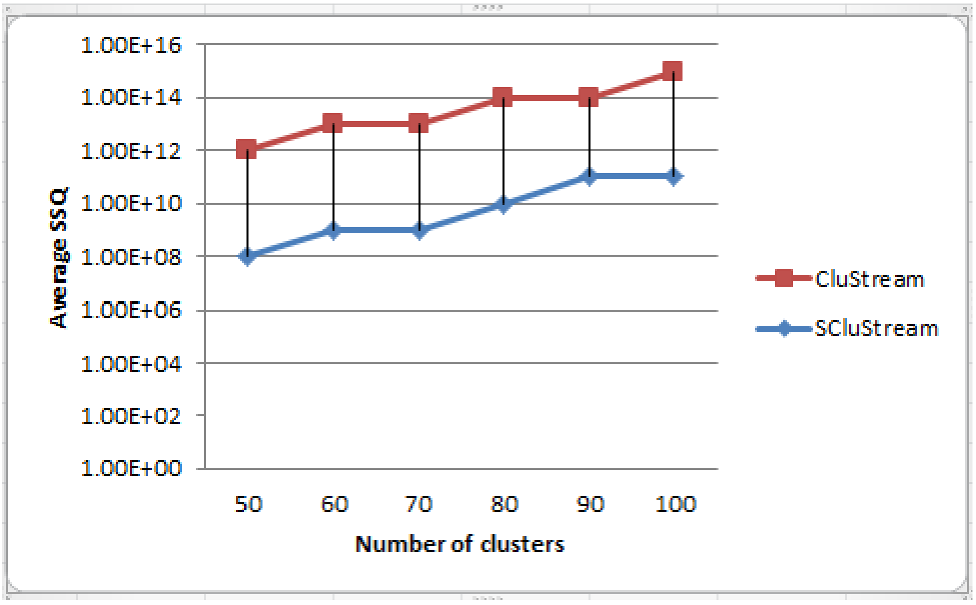


Figure 7: Clustering Quality (SSQ) of SCluStream and CluStream with different number of clusters.

Obviously, all the previous comparisons for the clustering quality of SCluStream and that of traditional CluStream using different parameters, indicates that SCluStream is better than CluStream.

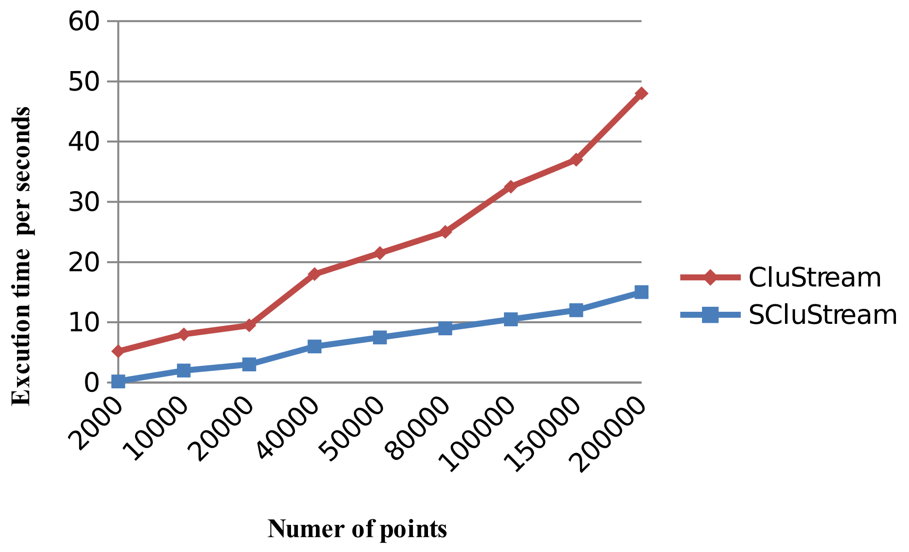


Figure 8: Execution time of SCluStream and CluStream for processing batch of data.

Figure 8 shows the execution time in seconds for both algorithms SCluStream and CluStream versus different number of data points. The figure shows that the elapsed time for processing 2000 points almost takes no time by SCluStream. Processing 200000 points by SCluStream takes 17 seconds in contrary to CluStream which takes 49 seconds. This means that SCluStream is approximately 3 times faster than CluStream. Accordingly, it is concluded that the elapsed time for processing the data in SCluStream is faster than the traditional CluStream, which makes SCluStream suitable for using in big data streaming.

Figures 9-11 shows the scalability results of SCluStream in regard of the execution time and memory usage versus different parameters. Figure 9 illustrates that the execution time increases slowly with the increase of number of micro clusters while fixing the number of points and number of dimensions. The execution time increases by only 6 seconds when the number of clusters updated from 20 to 140. Figure 10 shows that the execution time grows very slowly with the varying of the number of dimensions and fixing the number of clusters and number of points. The execution time only increase by 6 seconds when the numbers of dimensions change from 5 to 25.

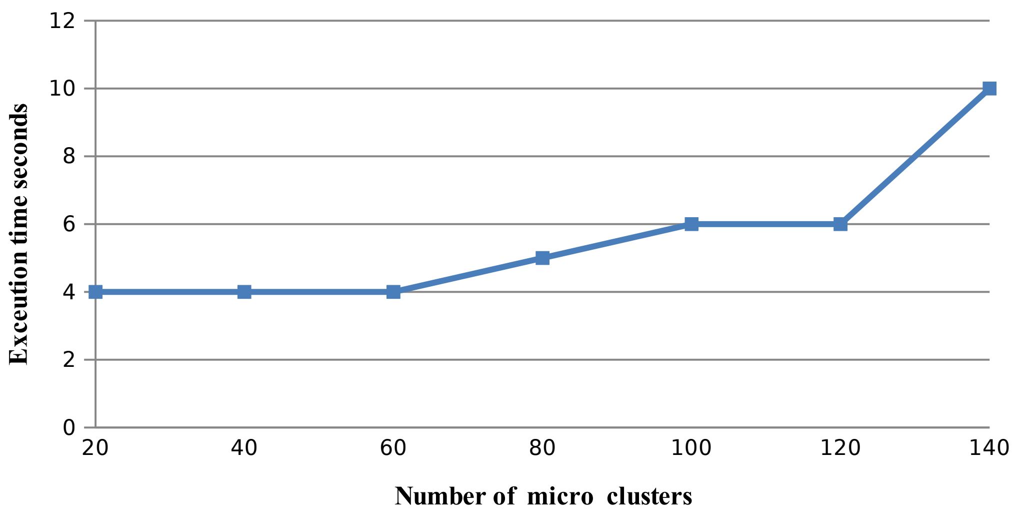


Figure 9: Scalability: Execution time of SCluStream with varying the number of micro clusters.

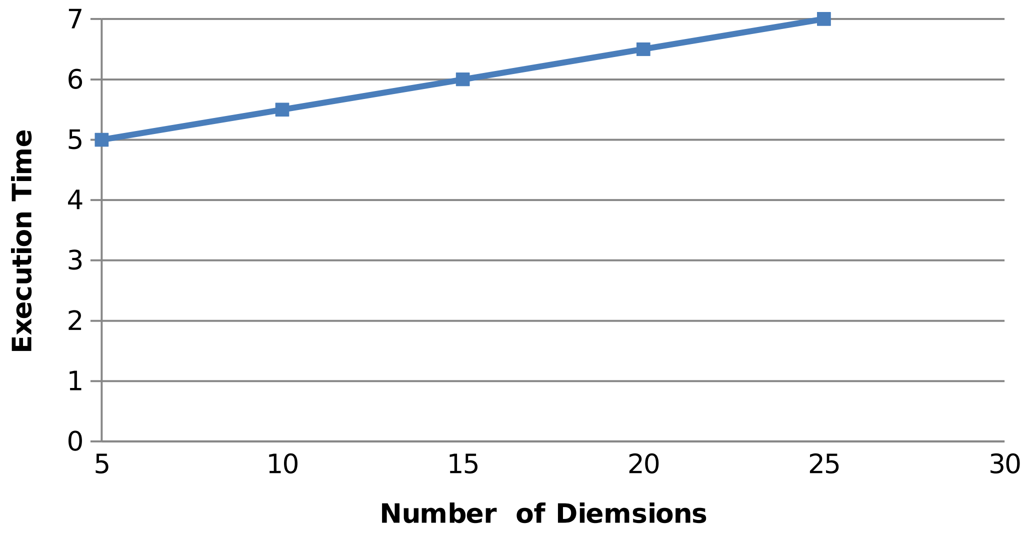


Figure 10: Scalability: Execution time of SCluStream with varying the number of dimensions.

Figure 11 illustrates the memory usage for both SCluStream and CluStream with the increase of the number of data points, which indicates a linear behavior but at different rates. It is obviously shown that the memory usage of SCluStream is limited as the streams length increases. For example, the memory only increases by 7KB when the length of stream increases from 80000 to 100000 in SCluStream, while in the CluStream the memory increases by 15KB when the length of stream increases from 80000 to 100000. This means that the memory consumption in SCluStream is far 2 times less than traditional CluStream, which makes SCluStream more suitable for dealing with big data streaming.

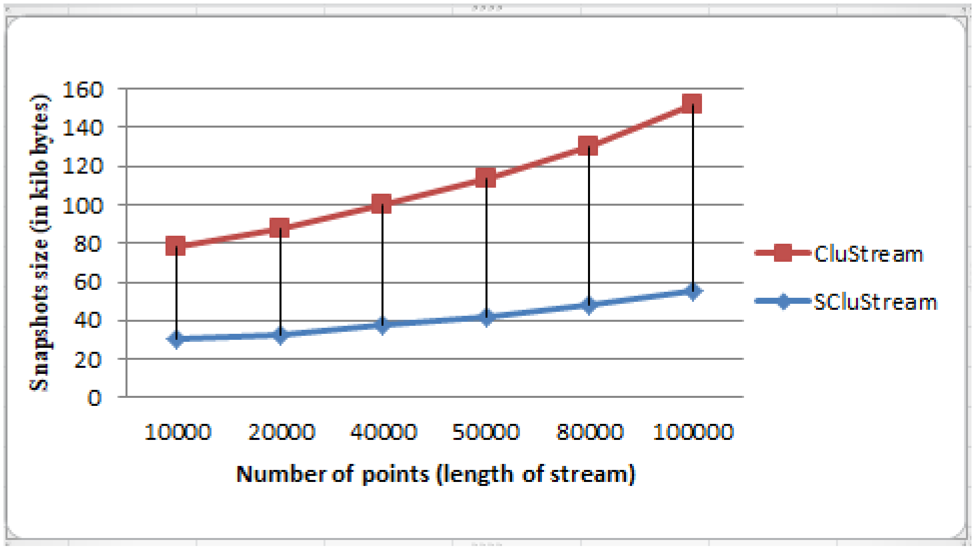


Figure 11: Scalability of SCluStream: Memory usage with varying stream length

4.4 Results Discussion

The previous experimental results have shown that the introduced proposed enhancements in SClustream managed to outperform the classical CluStream for the different parameter studied settings when testing on the real data set, network intrusion detection. Using k-means ++ rather than k-means in maintaining micro clusters and final clusters manages to improve both accuracy of clustering quality and speed of updates and generation of micro clusters, as well as the final clusters convergence. Thanks to SCluStream maintaining for snapshots only within a window size and deletion of expiring snapshots, the algorithm provides efficient real time performance. SCluStream is an appropriate extension for CluSream for use while clustering big data streams because SCluStream can overcome the main challenges in big data streams: - time and memory resources. SCluStream uses sliding window for tracking clusters for most recent data, so SCluStream saves more time when compares to classical CluStream which focus on the whole data and define the clusters for the - whole data. SCluStream also overcomes the memory constraint by deleting all snapshots expiring from window time, so SCluStream is an efficient implementation for clustering big data streams.

5 Conclusion and future work

In this paper, an effective algorithm SCluStream is proposed for clustering big data online streams. The proposed algorithm is an extension for the CluStream, which suggests tracking clusters over a sliding window and integrating more efficient clustering technique. CluStream is not practical when using big data, because all data is saved and processed and no elimination for old data takes place. SCluStream has an expiring phase to delete snapshots expiring from the window size; saving memory and focusing on generate final clusters from most recent data. Experimental evaluation over a real data set for intrusion detection has ensured that SCluStream outperforms CluStream, Especially in terms of clustering quality and scalability.

The future work will include the improvement of SCluStream to deal with all data types rather than only numerical data. The dynamic determination of all parameters settings. Use of data index instead of the synopsis is also suggested. Comparisons between SCluStream and Other data streaming clustering algorithms such as StreamingK++ and SWClustering .etc with different real data sets are also suggested to confirm the quality and scalability for SCluStream compare to other algorithms.

References

[1]Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, Andre C. P. L. F. DE Carvalho and Joao Gama," Data Stream Clustering: A Survey," ACM Computing Surveys, Vol. 46, No. 1, Article 13, Publication date: October 2013.

[2]C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in Proceedings of the 29th international conference on Very large data bases-Volume 29, 2003, pp. 81-92.

[3]Chonghuan Xu, " A Novel Spatial Clustering Method based on Wavelet Network and Density Analysis for Data Stream," in Journal Of Computers, Vol. 8, No. 8, August 2013.

[4]C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for projected clustering of high dimensional data streams," in Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, 2004, pp. 852-863.

[5]Maryam Mousavi, Azuraliza Abu Bakar, and Mohammadmahdi Vakilian,"Data Stream Clustering Algorithms: A Review," Int. J. Advance Soft Compu. Appl, Vol. 7, No. 3, November 2015 ISSN 2074-8523

[6]F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in Proceedings of the 2006 SIAM International Conference on Data Mining, 2006, pp. 328-339.

[7]L. Li-xiong, K. Jing, G. Yun-fei, and H. Hai, "A three-step clustering algorithm over an evolving data stream," in Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on, 2009, pp. 160-164.

[8]Jiadong Ren ; Ruiqing Ma, " Density-Based Data Streams Clustering over Sliding Windows," in Sixth International Conference on Fuzzy Systems and Knowledge Discovery, April 2009, 978-0-7695-3735-1.

[9]Lin, J. and H. Lin, "A Density-Based Clustering over Evolving Heterogeneous Data Stream". Vol. 5. 2009. 275-277.

[10] Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases." ACM Sigmod Record. Vol. 25. No. 2. ACM, 1996.

[11] Udommanetanakit, Komkrit, Thanawin Rakthanmanon, and Kitsana Waiyamai. "E-stream: Evolution-based technique for stream clustering." International Conference on Advanced Data Mining and Applications. Springer, Berlin, Heidelberg, 2007.

[12] Meesuksabai, W.T.Kangkachit, and K.Waiyamai. "Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty." International Conference on Advanced Data Mining and Applications. Springer, Berlin, Heidelberg, 2011.

[13] Tu, Li, and Yixin Chen. "Stream data clustering based on grid density and attraction." ACM Transactions on Knowledge Discovery from Data (TKDD) 3.3 (2009): 12.

[14] D.Arthur and S.Vassilvitskii. "K-means++: The Advantages of Careful Seeding." Proceedings of the eighteenth annual ACM, 2007.

[15] J.Youn, J.Shim and S.Lee. “Efficient Data Stream Clustering With Sliding Windows Based on Locality-Sensitive Hashing." IEEE Access. Vol 6, 2018.

[16] A.Zhou ,F.Cao ,W.Qian and C.Jin. “Tracking clusters in evolving data streams over sliding windows." Knowledge and Information Systems (2007).

[17] Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd. Vol. 96. No. 34. 1996.

[18] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: A clustering algorithm for data streams," Journal of Experimental Algorithmics (JEA), vol. 17, p. 2.4, 2012.