Received: May 31, 2022. Revised: June 25, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Dynamic Evolving Cauchy Possibilistic Clustering Based on the Self-Similarity Principle (DECS) for Enhancing Intrusion Detection System

Suha Mohammed Hadi[1] Ali Hakem Alsaeedi[2]* Mohammed Iqbal dohan[2]
Riyadh Rahef Nuiaa[3] Selvakumar Manickam[3] Ali Saeed D. Alfoudi[4]

[1]*Informatics Institute for Postgraduate Studies, Iraqi Commission for Computer and Informatics, Iraq*
[2]*College of computer science and information technology, University of Al-Qadisiyah, Iraq.*
[3]*National Advanced IPv6 Centre (NAv6), University Sains Malaysia, Malaysia*
[4]*Liverpool John Moores University, College of Computer Science, United Kingdom*
* Corresponding author's Email: ali.alsaeedi@qu.edu.iq

Abstract: Unsupervised machine learning plays a critical role in improving the security level of applications and systems. The cyberattack floods the network with data streams to deny services or destroy the network infrastructure. In this paper, a new development strategy (dynamic evolving cauchy possibilistic clustering based on the self-similarity principle (DECS)) is proposed to optimize the data stream clustering model based on the self-similarity principles (inter-cluster and intra-cluster). It is based on computing the self-similarity principles for data between and within clusters. The proposed system mitigates the dependence on the centre of the cluster to produce clusters with highly correlated data and minimal errors within the same cluster. The DECS consists of two phases: In the first phase, the data are clustered based on density points to generate clusters, and in the second phase, the clusters are evolved according to the inter-and intra-distance points to/in the clusters. The proposed model is tested on well-known intrusion datasets such as UNSW-NB15, KDD99 and NSL-KDD. In the experiments, DECS efficiently clustered the data with fewer errors and an optimal number of clusters compared to other models, it has achieved minimum error, high silhouette coefficient and an optimal number of clusters. The implementation results show that the average error of DECS over eight datasets is (0.45) with an average silhouette coefficient (0.63). At the same time, the pure cauchy possibilistic clustering obtains on average error (0.69) and silhouette coefficient (0.44).

Keywords: Data mining, Stream clustering, Anomaly intrusion detection system, Self-similarity, Evolving model.

1. Introduction

Given the many points of attack and breakthroughs in hacking computers, cybersecurity has become an important area for protecting information and computing resources. Cybersecurity aims to minimize the attack vectors/points because it is difficult to protect every attack point[1]. Therefore, the patterns of cyber-attacks happen to be from different classes. To overcome these challenges, all evolving methods to add, merge, and delete clusters should be implemented [2, 3]. At the same time, the system should learn and adapt to unknown situations and detect potential temporal shifts and data drifts [4]. Cauchy density assigns new samples to clusters or forms new clusters. The threshold for assignment depends on the statistical cluster distribution [5]. Moreover, it is crucial to utilize clustering by ignoring outdated stored knowledge that affects the tracking of dynamic changes without gaining complete control over the classification due to the absence of essential gained knowledge[6].

1.1 Motivation

Classification [7–9], regression [10], and clustering [4] models have been used to improve the performance and increase the detection rate of intrusion detection systems (IDS). Intrusion data is knowns as stream data with large amounts of data generated at high speed in networking [7]. The IDS

usually suffer from a low detection rate when dealing with data streams (attack and normal traffic). The stream clustering algorithms are efficient data mining tools that process and analyse streaming traffic [8] for detecting attacks and normal data. Most of these algorithms are facing weaknesses in terms of computation complexity and performance. According to this context, we introduce a new evolving strategy based on dynamic analysis of the geometric distribution and self-similarity to enhance Cauchy possibilistic clustering for data in the same cluster. The enhancement of the Cauchy possibilistic clustering algorithm consists of two main activities: clustering the data and evolution of the outcome clusters by splitting and margining. In general, standard stream clustering techniques such as Cauchy possibilistic potentially have the following drawbacks:

1. **High complexity**: They suffer from high complexity in clustering of the data stream, which could address in terms of the time and space complexity [7]. Generally, the stream clustering techniques produce a high and sparse number of clusters. This requires more extensive memory usage and costs much processing time.
2. **Inefficient prediction of clusters**: A traditional stream clustering produces a high number of clusters potentially resulting in an inefficient prediction of clusters. The stream data are likely to be clustered without having a high close proximity degree with other data points in the same cluster. Moreover, the clustering models which produced a high number of clusters do not work efficiently [8, 9].
3. **Huge sparse clusters**: The stream clustering models significantly suffer from producing a high number of clustering data that might degrade the performance of the clustering performance [10].

1.2 Contribution

In this paper, we propose a new evolving strategy to optimize the stream data clustering model based on the self-similarity principles (inter and intra clusters). The average similarity distance between inter-cluster elements represents the inter-self-similarity of cluster elements, while the intra-self-similarity refers to the average distance between the outer point and cluster. The power of self-similarity measures the clustering fragment for regular and irregular geometric objects [11, 12]. Self-similarity utilizes to satisfy two objects: obtaining clusters with high similarity content and the optimal number of clusters. According to the affirmation, the main contributions of the proposed model are:

1. **Low complexity**: Self-similarity is developed to design a new development strategy to improve the performance of the stream clustering model. These similarity measurements are based on the calculation of the distance between the data points of the cluster. The proposed technique enables clustering with low complexity compared to iterative models. The proposed technique potentially requires less time for clustering compared to other iterative clustering models.
2. **Efficient prediction of clusters**: The proposed self-similarity is introduced to compute the homogeneity of each cluster's data points and show how the cluster elements are connected. The proposed technique aims to create clusters with a very high degree of similarity by grouping them dynamically, and the proposed technique does not require a predefined number of clusters. The accurate predictions of the proposed technique can achieve better compression ratios than other stream clustering models.
3. **Optimal number of clusters**: The proposed system achieved the optimal number of clusters.

1.3 Evaluation strategies

To evaluate the performance of the proposed system, recent work on compression develops Cauchy clustering. Five datasets of attacks are used to evaluate the performance of the proposed model: CICDDoS2019, DRDoS_DNS, KDD99, NSLKDD, and UNSW-NB15, and three Red Wine Quality, keystroke, and SCADI

Three metrics for unsupervised performance factors are used: local error (i.e., cluster error rates), silhouette coefficient, and an optimal number of clusters. In addition, the accuracy and processing time of the proposed model are investigated and compared with the recent works on possibilistic Cauchy clustering.

1.4 Paper organization

The remaining sections of this paper are as follows: Section 2 explains related works; Intrusion detection system (IDS) and machine learning are presented in section 3; section 4 contains the research method; the experiments and result analysis are presented in section five; conclusion and future work are shown in section 6.

2. Related works

To develop the evolving mechanism of clustering techniques of the data stream. There were several studies have been proposing different methods for evolving the final clusters of the clustering algorithm. Most of the researchers in this field had applied the distance between the point and centre of the cluster to measure the degree of the belonging of the cluster (point-to-pint). Moreover, the other studies utilized the centre of the cluster to represent the major attribute of the clustering system. These studies produced several limitations in terms of performance and computation complexity (time, space).

Škrjanc *at. el*. [13] suggested cosine-based Cauchy clustering to monitor cyberattacks. The proposed technique is helpful when the data had noisy, and outliers occur frequently, like with cyberattacks. To process massive data sets, the algorithm evolved. The presented clustering technique merely adjusts a few parameters, like maximum density. This research implements the proposed strategy on the DARPA dataset. The proposed system had a weakness as shown by the encouraging results, the system reduces the dynamic behaviour of stream clustering.

Andonovski *at. el* [14] developed Cauchy possibilistic clustering to be suitable for large-scale cyberattack monitoring. The new strategy for discovering neuro-fuzzy models, based on filtered recursive least squares and gradually developing Gaussian clustering, shows promise for nonlinear dynamic processes. The model is generated online, and the algorithm has only four easy-to-understand parameters. Fuzzifying the data space can be done using numerous criteria and has many uses. On a simulated Hammerstein-type nonlinear dynamic process and a genuine plate heat exchanger, the algorithm was evaluated. Hammerstein-type identification works for many real-world processes. The limitation of the proposed method the fuzzy system significantly suffers from being capable to learn.

The authors in [11] proposed a new method to enhance the evolving concept of Cauchy possibilistic clustering. They evolve the algorithm based on the threshold, where a new point (x) joins to cluster if it lies in low as possible as the distance between the centre of the cluster and point x. Only half the distance between point x and centre of cluster remove other points in opposite side of point. Therefore, they reduce affects the points perhaps it closest from those points between the centre and point x.

According to [15] the authors suggest a new evolving stream clustering that consists of four phases: creating, splitting, merging, and removing. The creation is based on close local distance. The merging is based on high correlation points with the centre and nearest point. The model consists of two clustering algorithms: the creation of a clustering algorithm by assigning all points to a given cluster. The evolving cluster algorithms produce the optimal number of clusters.

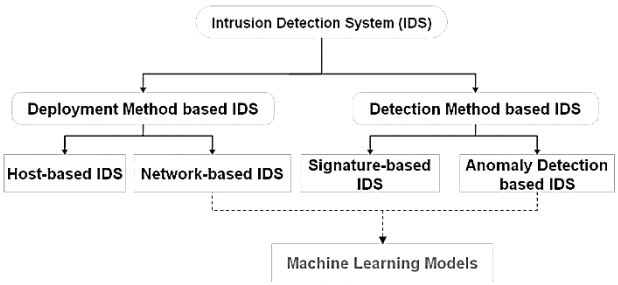

Figure. 1 Intrusion detection systems taxonomy structure

3. Intrusion detection system (IDS) and machine learning

IDS is a system that monitors network traffic to detect any suspicious behaviour and known threats [15]. When such activity is identified, it may also notify the administrator. Various machine learning (ML) techniques can be utilized to effectively manage and classify attacks. IDS can be a hardware or software system that automatically monitors, identifies the attack or intrusion, and notifies the computer or network [16–18]. This alert report assists the administrator or user in locating and resolving any vulnerabilities present on the system or network. Common methods for detecting intrusion include anomaly-based detection, signature-based detection, and hybrid detection [19–22].

Fig. 1 illustrates the classification of IDS models and which models can use machine learning.

Machine-learning algorithms have become increasingly popular as a method for detecting and identifying a wide variety of threats. These algorithms operate by first determining if a network packet is malicious or normal, and then classifying the packets accordingly [23, 24].

A network-based intrusion detection system observes network traffic for any suspicious, atypical, or unauthorized activities that may result in a cyberattack [25]. The basis of network-based intrusion detection is “acquired knowledge”. Either signature (misuse detection) of well-known assaults or Behaviour-based techniques are utilized to detect these behaviours (anomaly detection). Behaviour-based systems identify deviations from the typical behaviour profile and can detect unknown attacks,

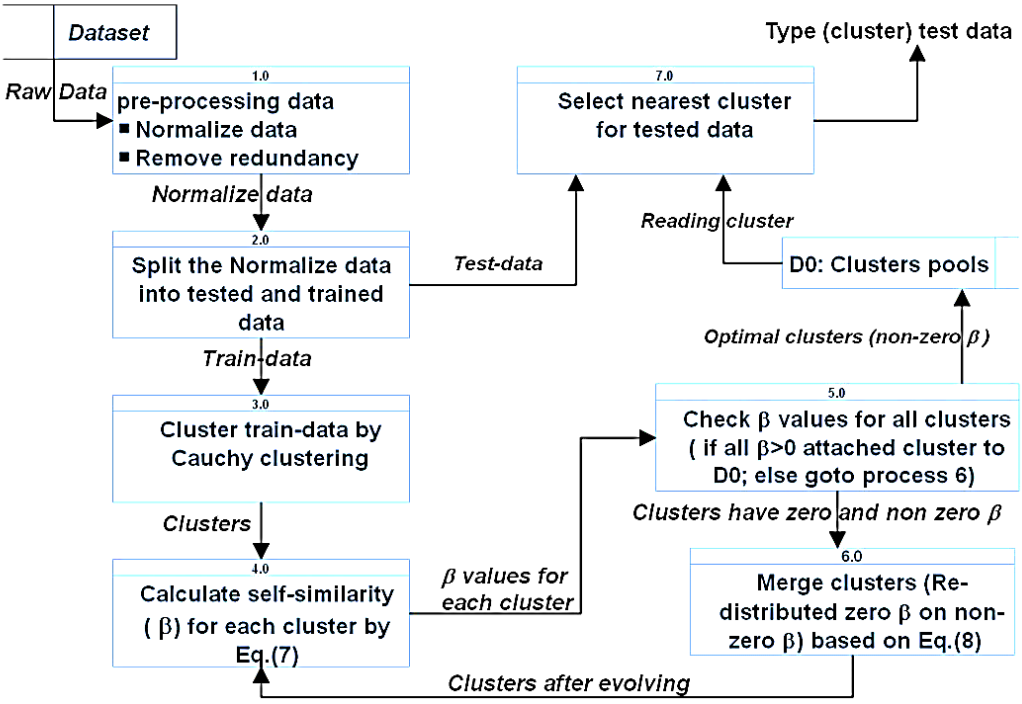

Figure. 2 Main steps of proposed DECS

whereas signature-based systems can only detect attacks for which they are trained to notify [26,27].

Over standard channels, all network traffic is expressly routed to the network intrusion detection system (NIDS) [28].

Implementation of an anomaly intrusion detection system to detect attacks based on recorded normal behaviour. This form of the intrusion detection system is commonly utilized because it can detect new types of intrusions by comparing the present real-time data to previously recorded regular real-time traffic [29].

4. Research method

This section explains the main steps of dynamic evolving Cauchy possibilistic clustering based on the self-similarity principle (DECS). Fig. 2 shows the steps of the proposed DECS.

4.1 Cluster trained data by Cauchy clustering

In this step, there are two activities add a new cluster and update the existing cluster.

- **Add a new cluster**

In begging the Cauchy density clustering calculates the density (γ_i^j) of pint $z(i)$ for cluster j - j is an index of each cluster in data at a time add point i- where the value of local density (γ_i^j) compared with the maximum-density threshold (Γ_{max})[2]. The algorithm generates a new cluster in case the γ_i^j greater than or equal the Γ_{max} otherwise the algorithm add point into cluster generate with a cluster maximum γ. According to [13], Eq. (8) calculates the local density of new data points:

$$\gamma_i^j = \frac{1}{1+\frac{1}{\sigma_t^2}(z(i)-\mu^j)^T(\Sigma^j)^{-1}(z(k)-\mu^j)+\frac{q\,(M-1)}{\sigma_t^2\,M^j}} \quad (1)$$

where μ^j defines the centre of the j^{th} cluster of dimensions $m \times 1$, and the lower and upper index at z_i^j define the i^{th} sample in the j^{th} cluster. The notation of cluster centre can also be written as $\mu_{M_j}^j$ to explicitly express that the j^{th} cluster consists of M^j samples. Eq. (2) calculates the μ^j.

$$\mu^j = \frac{1}{M^j}\sum_{i=1}^{M^j} z_i^j \quad (2)$$

Σ^j denotes the covariance matrix, of dimension $m \times m$, of the jth cluster. The covariance matrix is calculated in Eq. (3):

$$\Sigma_{M^j}^j = \frac{1}{M^j-1}\sum_{i=1}^{M^j}\left(z_i^j-\mu_{M^j}^j\right)\left(z_i^j-\mu_{M^j}^j\right)^T \quad (3)$$

- **Update cluster**

The cluster is updated when the density condition

is satisfied ($\gamma_1^j > \Gamma_{\text{mar}}$). The new data point adds to a cluster that achieved high cluster density. After adding a new point into the cluster, the algorithm (DECS) updates all cluster parameters. The number of elements in a cluster (M) increases by one. Eq. (4) calculates the new centre of the data.

$$\mu_{M^j+1}^j = \mu_{M^j}^j + \frac{1}{M^j+1}(z(k) - \mu_j) \quad (4)$$

The states of the non-normalized covariance matrix (S parameter) are updated according to Eq. (5).

$$S_{M^j+1}^j = S_{M^j}^j + (z(k) - \mu_j)\left(z(k) - \mu_{M^j+1}^j\right)^T \quad (5)$$

The covariance matrix is then obtained by Eq. (6).

$$\Sigma_{M^j+1}^j = \frac{1}{M^j} S_{M^j+1}^j \quad (6)$$

4.2 Evolving clusters and self-similarity

In this step, the proposed algorithm divides the clusters into two groups negative and positive. The positive group represent the clusters that have elements with a high correlation. Whenever the data in the same cluster are close to each other than they are to other clusters, those clusters are classified as positive membership or otherwise negative membership. Eqs. (7) and (8) calculates the membership of the self-similarity function of each cluster.

$$\omega_{x_i} = \begin{cases} 1 & if \quad c_{ir} - \frac{\sum_{j=1}^n d(x_i,x_j)}{n-1} \geq 0 \\ -1 & otherwise \end{cases} \quad (7)$$

Where: $j \neq i$ the index of the point in the same cluster, c_{ir} the nearest point to x_i in another cluster

$$\beta_j = \begin{cases} 1 & if \quad \frac{1}{m}\sum_{i=1}^n \sum_{j=1}^m \omega_{x_j} \geq 0 \\ 0 & otherwise \end{cases} \quad (8)$$

4.3 Marge cluster

The β values according to Eq. (8) are either 0 or 1. The clusters that have nonzero β-values (1-values) are passed on to the next generation of a clustering lifetime, while the clusters that have β-values equal to zero (0-values) must be redistributed to clusters that are not zero. Each point in zero-β clusters is reassigned to the corresponding non-zero β cluster. The proposed model calculates the average distances (δ) between the point to be redistributed and all points in the cluster (non-zero β cluster). Eq. (9) calculates

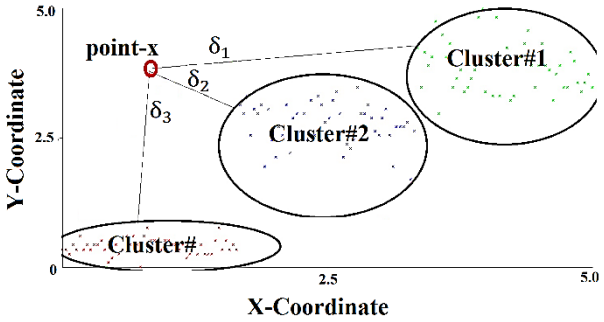

Figure. 3 δ between point (x) and three clusters

Table1. Dataset descriptions

Sq.	Dataset name	No. Instances	No. Attributes
1	CICDDoS2019	207673	77
2	DRDoS_DNS	33926	16
3	KDD99	5,209,460	41
4	NSLKDD	125,973	41
5	UNSW-NB15	2,540,044	45
6	Red Wine Quality	1600	12
7	keystroke	1600	8
8	SCADI	70	206

δ for each point in zero β clusters (x_j) and cluster k.

$$\delta = \frac{1}{m}\sum_{j=1}^m \|x_j - x_i\|^2 \quad (9)$$

Fig. 3 shows an example of the values of δ between point (x) and three clusters.

4.4 Select optimal cluster to test data

The test teat assigns to the appropriate cluster according to the closest corresponding cluster. Eq. (10) calculates the membership function to find the optimal test cluster for test data.

$$D_{j,j} = (x_i - m_j)^T \cdot C_j^{-1} \cdot (x_i - m_j) \quad (10)$$

Where $D_{j,j}$ is the square of the Mahalanobis distance, x_i is the vector of the observation (row in a dataset), m_j is the vector of mean values of independent variables (mean of each column), C_j^{-1} is the inverse covariance matrix of independent variables. The cluster corresponds minimum D between point x and cluster j.

5. Experiments and result analysis

In this section discusses three aspects dataset,

system performance over the dataset, and a comparison with other related works.

5.1 Dataset

The dataset section describes the dataset that was used to test the performance of the proposed system. Table 1 illustrates the dataset.

The dataset_3 (KDD99) we take 5 % randomly (260,473) from the entire dataset to evaluate the performance of the proposed model, and from dataset_5 (UNSW-NB15) we take 10 % randomly (254,004) from the entire dataset. All above eight datasets are split into 70 % train and 30% test.

5.2 Performance evaluation metrics

Three major clustering evaluation matrices are used to evaluate the clustering performance of the proposed clustering model: errors evaluation metrics, time, and a number of produced clusters. The error in the clustering process represents by the mean square error and silhouette. Mean square error (MSE) indicates the average sum of the euclidean distances between each point (x_j) and its cluster centre (μ_i.). Eq. (11) calculates the MSE of n points in the dataset.

$$MSE = \frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{m}\|x_j - \mu_i\|^2 \quad (11)$$

The silhouette (SIL) refers to a way of interpreting and validating consistency inside data clusters. The method generates a simple graphical depiction of how successfully each item was categorised. Eq. (12) calculates the average SIL for entire clusters.

$$SIL = \frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{m}\frac{x_j-\mu_k}{max\{x_j,\mu_k\}} \quad (12)$$

Where: k number of closures, m size of cluster i, x_j point ∈ cluster i , μ_k is nearest cluster centre in clusters pool has the smallest distance to point x_j.

5.3 Experimental results

To test the performance of our proposed framework, this subsection examines the MSE, the SIL factor, the number of clusters, and the computation time of the algorithms. Table 2 shows the comparative result between the pure Cauchy and the proposed DECS.

From Table 2, we can see that the MSE of the proposed DECS is lower (better) than that of the pure Cauchy used on all datasets. Reducing the number of clusters of DECS accompanied by minimizing the MSE. In dataset KDD99 the number of clusters is 318 reduced to 7 by enhancing the MSE from 1.32 in pure Cauchy to 0.9 in proposed DECS. The highest enhancement when proposed DECS used on CICDDoS2019 dataset, the MSE decreased the most by nearly 0.5 (from 0.96 to 0.48), and on keystroke, KDD99, NSL-KDD, and DRDoS_DNS by about 0.61,0.38, and 0.31 respectively. The lowest improvement when using the proposed model on the UNSW-NB15, Red Wine Quality, and SCADI the average error rate also decreased by about from 0.07 to 0.19.

Table 2. Result comparisons of MSE and SIL between pure Cauchy and proposed DECS

Sq.	Dataset name	Pure Cauchy		Proposed DECS	
		MSE	SIL	MSE	SIL
1	CICDDoS2019	0.96	0.39	0.48	0.64
2	DRDoS_DNS	0.35	0.72	0.26	0.76
3	KDD99	1.32	0.66	0.90	0.71
4	NSL-KDD	0.68	0.53	0.42	0.95
5	UNSW-NB15	0.72	0.64	0.58	0.87
6	Red Wine Quality	0.18	0.27	0.16	0.28
7	keystroke	0.76	0.02	0.29	0.42
8	SCADI	0.56	0.33	0.52	0.39

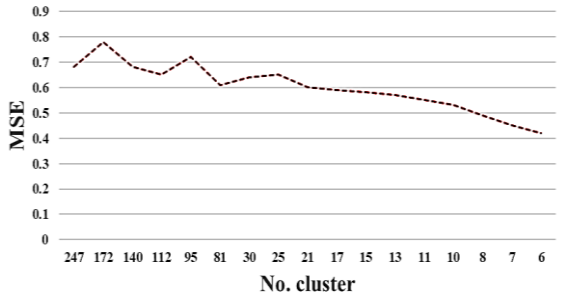

Figure. 4 MSE of proposed DECS on NSL-KDD dataset

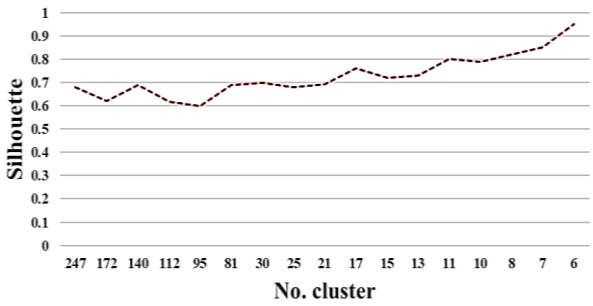

Figure. 5 SIL of proposed DECS on NSL-KDD dataset

Fig. 4 shows the MSE has been improving over evolving progress of proposed model.

The SIL rate was improved on all datasets with the highest improving percentage 0.4 in NSL-KDD and lowest red wine quality dataset 0.03. Fig. 5 shows

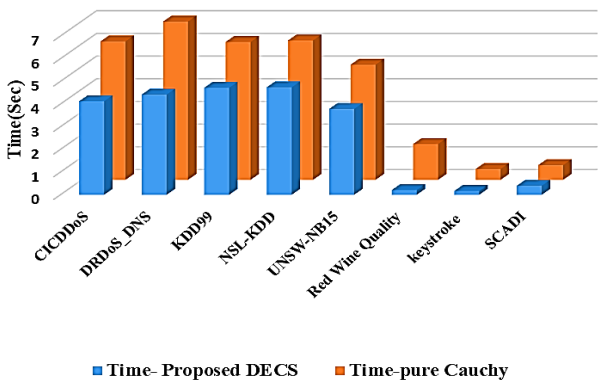

Figure. 6 SIL of proposed DECS on NSL-KDD dataset

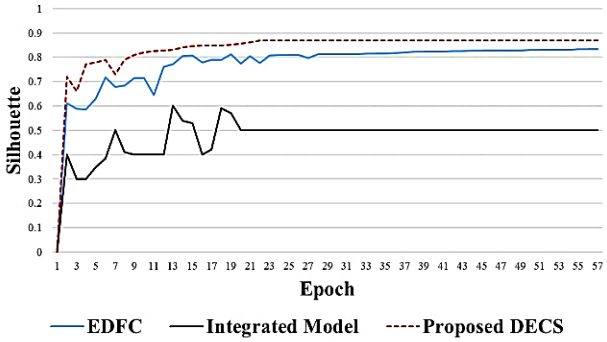

Figure. 7 SIL performance of proposed DECS, EDFC [15], and Integrated model [30]

Table 3. The silhouette value for the DECS model compared to silhouette values and the number of clusters of other models

Sq.	Dataset name	Reference			Proposed DECS	
		Reference No.	No. cluster	SLL	No. cluster	SIL
1	Keystroke	[11]	5	0.35	2	0.53
	NSL-KDD (1000)		3	0.35	9	0.42
2	UNSW-NB15	[30]	15	0.6	8	0.87
3	DRDoS_DNS	[15]	13	0.76	13	0.76
	UNSW-NB15		11	0.7	8	0.87
	NSL-KDD		2	0.93	6	0.95
	KDD99		3	0.66	7	0.68

the SIL has been improving over evolving progress of the proposed model.

The SIL and MSE give an indicator that the clusters that are produced by the proposed DECS have a high correlation with high similarity groups. The UNSW-NB15 and NSL-KDD are complex datasets, the proposed model overcomes the challenges in this dataset by grouped data that have a high similarity- low SIL and MSE-.

The processing time to assign the test point into the corresponding cluster is minimised by the proposed algorithm. The time required is different according to the size of the dataset used to train the model. The big volume of data the attached test point needs is approximately by nearly 4 sec, while the small size of the dataset required less than 1 sec.

Fig. 6 shows the processing time needed for each point time required.

The proposed DECS compare with three recent evolving strategies of cluster streaming data.

Fig. 7 shows the compared performance among proposed DECS, EDFC[15], and integrated model [30] on UNSW-NB15.

From Fig. 7, the proposed DECS and integrated model finished the evolving process after 25 epochs, while the EDFC continued until 57 epochs.

5.4 Comper the proposed DECS with other studies

This section compares the proposed system with existing techniques applied to the same dataset. The first study in [11] shows that the evolving clustering system depends on two aspects: removing and merging. The removing technically dependent on the threshold. At the same time, merging is based on the minimum average distance of the points within half the distance between point x and the centre of the cluster. The second technique in comparison is [30], which constructed clusters to improve the detection of attacks. The proposed model reduced the data by selecting 15 sets by trial and error. In the study [15], the cluster model grouped the data based on the least distance between points in the data pool. The evolving process separates the clusters into positive and negative sets based on thresholds, and the last one is redistributed to the positive clusters based on another threshold. Table 3 illustrates the comparison between our proposed system and methods in [11], [15], and [30] according to silhouette officiants and the number of clusters.

In summary, the proposed model can effectively accelerate the convergence speed and reduce the number of clusters in the clustering process. The MSE and SIL of the proposed algorithm improve significantly. From the above analysis, we can conclude that using the proposed framework can effectively improve the performance of the DECS algorithm.

6. Conclusion

The major challenge in clustering stream data is the number of clusters. The streaming of data over a

network increases the difficulty of processing and distinguishing between normal and abnormal traffic. The merging and splitting of data according to self-similarity principles inter or intra clusters points to enhance the performance of stream clustering. The time required to attach the test points into suitable clusters depends essentially on the number of clusters; therefore, reducing the processing time done by reducing clusters enhances the correlation of data points in the same cluster. The requirement of segregation of clusters into optimal clusters a robust correlation metrics of entire data in the same cluster. The centre of a cluster often does not a representation of entire cluster precisely, so all cluster members have to use it to describe the cluster. The future work is to implement the DECS model on a real online dataset with a zero-day attack.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The paper's background work, conceptualization, methodology and result analysis and comparison have been done by the first and second authors. Dataset collection, the implementation by third and fourth authors. The preparation and editing draft and visualization have been done by the fifth and sixth authors. The supervision, review of work and project administration, have been done by the first author.

References

[1] E. Alothali, H. Alashwal, and S. Harous, "Data stream mining techniques: A review", *Telkomnika (Telecommunication Comput. Electron. Control*, Vol. 17, No. 2, pp. 728–737, 2019, doi: 10.12928/TELKOMNIKA.v17i2.11752.

[2] D. B. Rawat, R. Doku, and M. Garuba, "Cybersecurity in Big Data Era: From Securing Big Data to Data-Driven Security", *IEEE Trans. Serv. Comput.*, Vol. 14, No. 6, pp. 2055–2072, 2021, doi: 10.1109/TSC.2019.2907247.

[3] I. Škrjanc, S. Ozawa, T. Ban, and D. Dovžan, "Large-scale cyber attacks monitoring using Evolving Cauchy Possibilistic Clustering", *Appl. Soft Comput. J.*, Vol. 62, pp. 592–601, 2018, doi: 10.1016/j.asoc.2017.11.008.

[4] A. S. Choudhary, P. P. Choudhary, and S. Salve, "A Study on Various Cyber Attacks and A Proposed Intelligent System for Monitoring Such Attacks", In: *Proc. of 3rd Int. Conf. Inven. Comput. Technol. ICICT 2018*, pp. 612–617, 2018, doi: 10.1109/ICICT43934.2018.9034445.

[5] D. Leite, I. Škrjanc, and F. Gomide, "An overview on evolving systems and learning from stream data", *Evol. Syst.*, Vol. 11, No. 2, pp. 181–198, 2020, doi: 10.1007/s12530-020-09334-5.

[6] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A Survey", *Inf. Sci. (Ny).*, Vol. 490, pp. 344–368, 2019, doi: 10.1016/j.ins.2019.03.060.

[7] A. Adnan, A. Muhammed, A. A. A. Ghani, A. Abdullah, and F. Hakim, "Hyper-heuristic framework for sequential semi-supervised classification based on core clustering", *Symmetry (Basel).*, Vol. 12, No. 8, 2020, doi: 10.3390/SYM12081292.

[8] A. Amini, T. Y. Wah, M. R. Saybani, and S. R. A. S. Yazdi, "A study of density-grid based clustering algorithms on data streams", In: *Proc. of 2011 8th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2011*, Vol. 3, pp. 1652–1656, 2011, doi: 10.1109/FSKD.2011.6019867.

[9] A. H. Alsaeedi, "Evolving Dynamic Fuzzy Clustering (EDFC) to Enhance DRDoS_DNS Attacks Detection Mechnism", *Int. J. Intell. Eng. Syst.*, Vol. 15, No. 1, 2022, doi: 10.22266/ijies2022.0228.46.

[10] P. Somasekaram, R. Calinescu, and R. Buyya, "High-availability clusters: A taxonomy, survey, and future directions", *J. Syst. Softw.*, Vol. 187, 2022, doi: 10.1016/j.jss.2021.111208.

[11] H. A. A. A. Khamees, N. A. A'araji, and E. S. A. Shamery, "Data Stream Clustering Using Fuzzy-based Evolving Cauchy Algorithm", *Int. J. Intell. Eng. Syst.*, Vol. 14, No. 5, pp. 348–358, 2021, doi: 10.22266/ijies2021.1031.31.

[12] D. A. Shammary and I. Khalil, "Dynamic fractal clustering technique for SOAP web messages", In: *Proc. of 2011 IEEE Int. Conf. Serv. Comput. SCC 2011*, pp. 96–103, 2011, doi: 10.1109/SCC.2011.15.

[13] I. Skrjanc, S. Ozawa, D. Dovˇ, B. Tao, J. Nakazato, and J. Shimamura, "Evolving Cauchy Possibilistic Clustering and Its Application to Large-Scale Cyberattack Monitoring", In: *Proc. of 2017 IEEE Symposium Series on Computational Intelligence*, pp. 1–7, 2018, doi.org/10.1109/SSCI.2017.8285203.

[14] G. Andonovski, E. Lughofer, and I. Skrjanc, "Evolving Fuzzy Model Identification of Nonlinear Wiener-Hammerstein Processes", *IEEE Access*, Vol. 9, pp. 158470–158480, 2021, doi: 10.1109/ACCESS.2021.3130678.

[15] R. R. Nuiaa, A. H. Alsaeedi, S. Manickam, and D. E. J. A. Shammary, "Evolving Dynamic Fuzzy Clustering (EDFC) to Enhance DRDoS_DNS Attacks Detection Mechnism", *Int. J. Intell. Eng. Syst.*, Vol. 15, No. 1, pp. 509–519, 2022, doi: 10.22266/IJIES2022.0228.46.

[16] J. Zala, A. Panchal, A. Thakkar, B. Prajapati, and P. Puvar, "Intrusion Detection System using Machine Learning", *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, No. Icosec, pp. 61–71, 2020, doi: 10.32628/cseit2062166.

[17] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. K. A. A. Khan, "Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review", *Procedia Comput. Sci.*, Vol. 171, No. 2019, pp. 1251–1260, 2020, doi: 10.1016/j.procs.2020.04.133.

[18] C. Xu, J. Shen, X. Du, and F. Zhang, "An Intrusion Detection System Using a Deep Neural Network with Gated Recurrent Units", *IEEE Access*, Vol. 6, pp. 48697–48707, 2018, doi: 10.1109/ACCESS.2018.2867564.

[19] Amrita and P. Ahmed, "A hybrid-based feature selection approach for IDS", *Lect. Notes Electr. Eng.*, Vol. 284 LNEE, pp. 195–211, 2014, doi: 10.1007/978-3-319-03692-2_16.

[20] N. Shahadat, I. Hossain, A. Rohman, and N. Matin, "Experimental analysis of data mining application for intrusion detection with feature reduction", In: *Proc. of ECCE 2017 - Int. Conf. Electr. Comput. Commun. Eng.*, pp. 209–216, 2017, doi: 10.1109/ECACE.2017.7912907.

[21] R. B. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, "Towards detecting and classifying network intrusion traffic using deep learning frameworks", *J. Internet Serv. Inf. Secur.*, Vol. 9, No. 4, pp. 1–17, 2019, doi: 10.22667/JISIS.2019.11.30.001.

[22] A. Tajbakhsh, M. Rahmati, and A. Mirzaei, "Intrusion detection using fuzzy association rules", *Appl. Soft Comput. J.*, Vol. 9, No. 2, pp. 462–469, 2009, doi: 10.1016/j.asoc.2008.06.001.

[23] S. Aljawarneh, M. Aldwairi, and M. Bani, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model", *J. Comput. Sci.*, Vol. 25, pp. 152–160, 2018, doi: 10.1016/j.jocs.2017.03.006.

[24] S. Khan, K. Kifayat, A. K. Bashir, A. Gurtov, and M. Hassan, "Intelligent intrusion detection system in smart grid using computational intelligence and machine learning", *Trans. Emerg. Telecommun. Technol.*, Vol. 32, No. 6, p. e4062, 2021.

[25] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset", *IEEE Access*, Vol. 9, pp. 142206–142217, 2021.

[26] S. Kumar, A. Viinikainen, and T. Hamalainen, "A network-based framework for mobile threat detection", In: *Proc. of 2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 2018, pp. 227–233.

[27] S. Kumar, A. Viinikainen, and T. Hamalainen, "Machine learning classification model for network based intrusion detection system", In: *Proc. of 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2016, pp. 242–249.

[28] M. O. Okay, R. Samet, Ö. Aslan, and D. Gupta, "A Comprehensive Systematic Literature Review on Intrusion Detection Systems", *IEEE Access*, 2021.

[29] I. S. Thaseen, B. Poorva, and P. S. Ushasree, "Network intrusion detection using machine learning techniques", In: *Proc. of 2020 International Conference on Emerging Trends in Information Technology and Engineering (IC-ETITE)*, 2020, pp. 1–7.

[30] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset", *Cluster Comput.*, Vol. 23, No. 2, pp. 1397–1418, 2020, doi: 10.1007/s10586-019-03008-x