Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

ACVSLA: Automatic Clustering Using Variable Structure Learning Automata

babak anari

babak.anari@iau.ac.ir

Islamic Azad University, Shabestar 53815-159, Iran

zohreh anari

Payame Noor University (PNU), Tehran, Iran

Research Article

Keywords: Data clustering, Automatic Data clustering, Learning Automata, Variable Structure Learning Automata

Posted Date: May 4th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9251725/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

ACVSLA: Automatic Clustering Using Variable Structure Learning Automata

Babak Anari[a,*], Zohreh Anari[b,*]

a) Department of Computer Engineering, Shab.C., Islamic Azad University, Shabestar 53815-159, Iran. babak.anari@iau.ac.ir
b) Department of Computer Engineering and Information Technology, Payame Noor University (PNU), Tehran, Iran. zanari323@pnu.ac.ir

[*] ***Corresponding Authors:***

Babak Anari, Zohreh Anari

Abstract

Determining the optimal number of clusters and their corresponding centroids, known as automatic data clustering, is an NP-complete problem. Most of the traditional automatic data clustering methods rely on thresholds or other user-specified parameters to determine the number of clusters, making their performance highly dependent on threshold settings and failing to work well for datasets with varying shapes, sizes, densities, and noise levels. In order to address these limitations, in this paper, we propose a novel automatic clustering method based on Variable Structure Learning Automata (VSLA). The proposed ACVSLA algorithm automatically identifies both the optimal number of clusters and their centroids from the data, without requiring threshold settings or predefined assumptions. The performance of the proposed algorithm was tested against various state-of-the-art automatic clustering methods over sixteen benchmark datasets from the UCI Machine Learning Repository. Experimental results show that ACVSLA can identify the number of clusters more accurately and is more stable and robust in producing clustering results.

Keywords: Data clustering; Automatic Data clustering; Learning Automata; Variable Structure Learning Automata.

1. Introduction

Data clustering, or data clustering analysis, is an essential technique in data mining that partitions unlabeled data objects in a dataset into groups based on inherent similarities[1]. Accordingly, objects within a cluster have a minimum distance from each other, and the maximum distance separates objects from different clusters. Data clustering algorithms are widely used across various fields, including pattern identification, web mining, data science, market research, image processing, natural language processing, software evolution, data analysis, document clustering, and machine learning[2-16].

There are two main clustering algorithms in the literature [17, 18]: hard and fuzzy. In hard clustering, each data point is either entirely in a cluster or not in any cluster. Instead of categorizing each input data point into a specific cluster, fuzzy clustering assigns each data point a probability of belonging to each cluster. Hard clustering algorithms can be classified into four distinct categories [9, 17]: 1) hierarchical, 2) partitional, 3) density-based, and 4) grid-based clustering algorithms. In addition, these categories use distinct methods to analyze data, and each method has advantages and disadvantages.

In hierarchical clustering algorithms, clusters are formed in a hierarchical manner. Hierarchical clustering provides a flexible method for determining optimal object groupings without the need to predefine the number of clusters. Hierarchical clustering algorithms can be divided into two classes [19, 20]: agglomerative and divisive. Agglomerative clustering starts by treating each object as its own cluster. In each step, clusters are merged using similarity measures. The merging process employs an iterative method until all data objects are consolidated into a single cluster. In divisive clustering, all data points are placed into a single cluster, which is then recursively divided into smaller clusters.

Partitional clustering algorithms split datasets into clusters according to specific criteria; hierarchical clustering algorithms apply this partitioning in a tree structure. One challenge with such an approach is that its results are sensitive to the initial choice of centers. Furthermore, these algorithms require a pre-defined number of clusters [21].K-means is one of the well-known algorithms in this field [22].

In density-based clustering algorithms, clusters are generated through multi-resolution grid data structures. These algorithms can significantly identify data outliers and generate clusters with non-uniform shapes, but are limited in their capability to cluster high-dimensional data and clusters with varying densities [23].

Grid-based clustering methods use a multi-resolution grid data structure, dividing the data items into a predefined number of cells to obtain a grid layout. This method has the benefit of a relatively quick processing time [24]. It is independent of the number of objects given, but is dependent on the number of cells that can fit within each dimension.

Although most data clustering algorithms have high accuracy, the number of clusters is not predetermined. This is a significant challenge, as automatically determining clusters is a difficult task [25]. As a result, automatic clustering has become a promising approach to addressing these

challenges [26]. Recently, several studies have attempted to address problems associated with automatic clustering [27-40]. Most reported methods are based on nature-inspired metaheuristic algorithms. All mentioned algorithms commonly use threshold values to identify clusters. Determining a suitable threshold value is a challenging task that can significantly affect the quality of the obtained clusters.

This paper proposes a new automatic clustering algorithm, i.e., Automatic Clustering based on Variable Structure Learning Automata (ACVSLA), to address the automatic clustering problem. Unlike many traditional automatic clustering methods that rely on a parameter or threshold settings, the proposed method is independent of threshold settings, more robust and accurately estimates the best number of clusters for any dataset size with cooperation of learning automata. The proposed method organizes cooperative automata into Level 1 and Level 2. The automata in Level 1 automatically determine the number of clusters. In contrast, the automata in Level 2 are responsible for finding the clusters corresponding to the number specified by Level 1. The cooperation between automata at these two levels works as follows: the automaton in Level 1 randomly selects an action from its available action set. It applies to the automata in Level 2. The automata in Level 2 act as a random environment for Level-1 automata. Based on the reinforcement signal received from Level 2, the automaton in Level 1 decides whether to reward or penalize its selected action. This learning process is repeated until the optimization function reaches its minimum value. Once this minimum value is achieved, the exact number of clusters and their corresponding centroids can be identified.

Our primary contributions to this paper are summarized as follows:

1. A novel framework to perform automatic data clustering based on VSLA is proposed.
2. The proposed framework does not require a predefined threshold value to identify active clusters, which improves robustness and reduces dependency on heuristic parameters.
3. Unlike most metaheuristic clustering algorithms that require multiple control parameters, the proposed ACVSLA only requires tuning the reward and penalty parameters of learning automata, which significantly reduces the complexity of parameter tuning.
4. Experimental results on benchmark datasets show that ACVSLA accurately identifies the exact number of clusters and achieves superior performance compared to existing state-of-the-art automatic clustering methods.

The remainder of this paper is organized in the following sections: Section 2 presents related work on existing methods. The ideas associated with learning automata theory and its variants are discussed in Section 3. Section 4 presents the proposed automatic data clustering algorithm using variable structure learning automata. The performance evaluation results are provided in Section 5. Finally, a summary of the main findings of this paper and some suggestions for future work are provided in Section 6.

2. Related works

This section reports and summarizes several clustering-method studies. The classification of these studies is as follows: 1) fuzzy, 2) center-based, 3) search-based, 4) graph-based, 5) density-based, 6) model-based, 7) subspace-based, 8) nature-inspired-based metaheuristic, 9) learning-based, and 10), other clustering algorithms.

2.1 *Fuzzy Clustering Algorithms*

Fuzzy clustering allows each data object to be assigned to multiple clusters rather than one. Bellman et al. [41] proposed several clustering algorithms based on fuzzy set theory. Pal and Mitra [42] presented a three-stage automatic fuzzy clustering algorithm. An automatic fuzzy clustering algorithm was proposed by Maulik and Saha [43] using a modified differential evolution method.

2.2 *Centroid-based Clustering Algorithms*

These algorithms represent each cluster with a single element, called the cluster center. A center-based algorithm seeks to minimize the objective function. One of the most popular centroid-based algorithms is K-means, which iteratively partitions a dataset into groups based on the distance to their centroids [44]. In centroid-based algorithms, clusters are typically convex, and each is defined by its center. These algorithms are especially effective for classifying high-dimensional datasets.

2.3 *Search-based Clustering Algorithms*

These clustering algorithms find globally optimal solutions by exploring the entire search space. Tabu Search (TS) and Harmony Search are the most popular optimization techniques used in these algorithms. Al-Sultan and Fedjki [45] introduced a new TS-based method to identify a global solution.

2.4 *Graph-based Clustering Algorithms*

These algorithms use an unsupervised learning approach to cluster nodes in a graph based on shared characteristics. The proposed method does not require prior knowledge of the dataset's number, size, density, or cluster structure. As a result, graph-based clustering algorithms are useful for identifying patterns in complex datasets [46]. Karypis et al. [47] presented a method that partitions graphs to group data objects in suitable sub-clusters.

2.5 *Density-based Clustering Algorithms*

This method can even identify clusters with different shapes and sizes in a large dataset, even in the presence of noise and anomalies. DBSCAN is a well-known clustering algorithm [48].

2.6 *Model-based Clustering Algorithms*

This algorithm is a statistical method that analyzes data using a mixture model. In this approach, data are generated through a combination of probabilistic features, where each cluster is represented by a normal or Gaussian distribution. Additionally, each dataset is characterized by its likelihood of belonging to a cluster [49].

2.7 *Subspace Clustering Algorithms*

A subspace clustering technique finds clusters in a dataset across different subspaces. Clustering algorithms face two primary challenges when dealing with high-dimensional datasets: they apply dimension reduction when the distances between data objects are nearly identical [50], and distinct clusters may be found in different subspaces [51]. Parsons et al. [52] proposed a study of various subspace clustering algorithms, including a classification system that groups them based on their unique characteristics.

2.8 *Nature-inspired Metaheuristic Algorithms*

Nature-inspired metaheuristic algorithms are often more effective than conventional methods for solving clustering and global optimization problems [46]. These algorithms tend to converge quickly but may stop at local minima. The local optima can influence the final optimal solution. Furthermore, they enable fast convergence in clustering algorithms. In recent years, researchers have proposed several nature-inspired metaheuristic algorithms to automatically cluster data, addressing the limitations of conventional optimization methods. Among these techniques are population-based algorithms, swarm intelligence, and evolutionary algorithms [40, 53-61] . Das et al. [62] introduced an automatic clustering technique that employs an enhanced differential evolution algorithm (ACDE). This method incorporates two novel techniques to optimize and partition the number of clusters. Their algorithm consists of two steps: the first step uses a fixed-length chromosome encoding representation and applies a threshold value to identify active clusters. The second step enhances algorithm efficiency using the differential evolution approach. Using a K-means algorithm, Kuo and Zulvia [60] proposed an improved version of artificial bee colony optimization (iABC) for automated clustering. Their enhanced iABC guides onlooker bees toward better locations, improving the behavior of these bees compared to the traditional ABC. Additionally, by providing a better initial clustering center, the upgraded ABC enables faster and more effective solution searching by onlooker bees. By providing a better initial centroid for the K-means clustering algorithm, the enhanced ABC enables onlooker bees to search for solutions more quickly and efficiently compared to their previous movement patterns. The clustering results from the iABC algorithm show that it outperforms other well-known clustering techniques in terms of stability, although it requires relatively high computational time. Das et al. [27] introduced an automatic clustering approach based on an evolutionary computing method known as the Bacterial Evolutionary Algorithm (BEA). Their simulation results showed that the proposed algorithm outperformed the comparison algorithms they evaluated. AHPSOM (Adaptive Hybrid Particle Swarm Optimization with Mutation) is an advanced algorithm designed for automatic data clustering, especially in dynamic and continuously generated data scenarios. Proposed by Sharma [63] , AHPSOM enhances the traditional Particle Swarm Optimization (PSO) method by adding a mutation operator, which helps increase the diversity of solutions and prevents

premature convergence. Ibrahim et al. [64] introduced a hybrid clustering algorithm that combines Grey Wolf Optimization (GWO) with a Trajectory-based Search (TS) algorithm to improve the clustering process. This hybrid approach aims to enhance the effectiveness of the GWO algorithm by achieving a better balance between exploration and exploitation phases. While the GWO-TS algorithm is a powerful clustering tool, it faces challenges related to computational cost, scalability, and parameter sensitivity. Hussain et al. [65] proposed an ABC optimization-based algorithm designed for clustering large, high-dimensional datasets. This method integrates co-clustering principles within the ABC framework. Instead of using Euclidean distance, the algorithm applies higher-order correlations to obtain results. It also enhances diversification by exploring the search space through three distinct approaches. The study highlights the algorithm's scalability for parallel architectures, supporting both shared memory and distributed environments. While innovative and well-suited for large, high-dimensional datasets, the algorithm has drawbacks, including higher computational complexity, resource demands, and parameter sensitivity. The Improved Artificial Bee Colony Optimization with K-means algorithm (iABC), proposed by Kuo and Zulvia [60], is an enhancement of the classical Artificial Bee Colony (ABC) algorithm. This improvement results in better initial centroids for the K-means algorithm, which accelerates and enhances the onlooker bees' ability to find solutions compared to their original random movement. The results showed that iABC outperformed other algorithms in terms of both effectiveness and stability, although it required more computational time. It was specifically designed to optimize the process of automatic data clustering, particularly for customer segmentation.

2.9 *Learning-based Clustering Algorithms*

Recently, Learning Automata (LA) has been used as an optimization tool to address the problem of automatic clustering. Anari et al. [66] developed a dynamic clustering algorithm called ACCALA to optimize the number of clusters and their corresponding centroids. The method uses cluster centroids encoding and threshold settings to determine active clusters. Hasanzadeh-Mofrad et al. [67] proposed a clustering approach called learning automata clustering (LAC) for clustering data objects. In this method, each data object is initially assigned a learning automaton. In the second step, each automaton determines the degree of membership to the desired cluster. Finally, the assignment of each cluster is determined by comparing the new and previous cluster centers using the reinforcement signal. However, this method does not perform automatic clustering and assumes that the number of clusters is predetermined.

2.10 *Other Clustering Algorithms*

In this section, several text-based clustering techniques are presented. Mehta et al. [10] examined and evaluated various proximity measures for the data clustering problem and also introduced a theoretical approach for selecting the appropriate proximity measure for clustering purposes. Additionally, Mehta et al. [11] proposed a document clustering method called WE Clustering for large text datasets. Their method combines the benefits of statistical scoring methods like TF-IDF with advanced deep learning models such as BERT. In a recent research, Mehta et al. [12] developed a semantic clustering approach that combines statistical features and semantic features in textual datasets. In another study, Mehta et al. [13] reviewed and analyzed various text clustering techniques, ranging from conventional non-semantic methods to advanced state-of-the-art

semantic text clustering approaches. Chander et al. [14] provided an overview of the taxonomy of clustering, highlighting the usage of unsupervised clustering methods and deep learning-based unsupervised clustering models. Moya et al. [15] applied clustering techniques to datasets related to traumatic brain injury (TBI) to group similar patients. The Random Swap (RS) Algorithm, introduced by Fränti and Kivijärvi [68], proposed a simple yet effective method to improve clustering performance by randomly swapping centroids and refining the solution iteratively. Zhao et al. introduced a new cluster validity index, called the WB-index, designed to evaluate the quality of clustering results, particularly for sum-of-squares based clustering algorithms like k-means. The WB-index is computationally efficient and performs well for datasets with varying cluster sizes and densities. Kärkkäinen et al. [69]proposed a Dynamic Local Search algorithm to address the clustering problem. Their approach enhances traditional clustering methods by dynamically adjusting local search operations to improve clustering accuracy and efficiency. Fränti et al.[70] investigate the performance of K-means on six benchmark datasets. The primary goal of the study is to identify the strengths and weaknesses of K-means when applied to different data structures and to propose strategies for improving its performance. Rezaei and Fränti[71] proposed the issue of determining the optimal number of clusters in clustering algorithms using external indices. The authors explore external validation indices, which leverage prior knowledge, such as ground truth labels, to assess clustering quality and determine the appropriate number of clusters. Table 1 summarizes the recommended algorithms for automatic data clustering problems, and Fig. 1 illustrates the clustering algorithm categories, organizing the concepts hierarchically to show the relationships between different clustering methods.

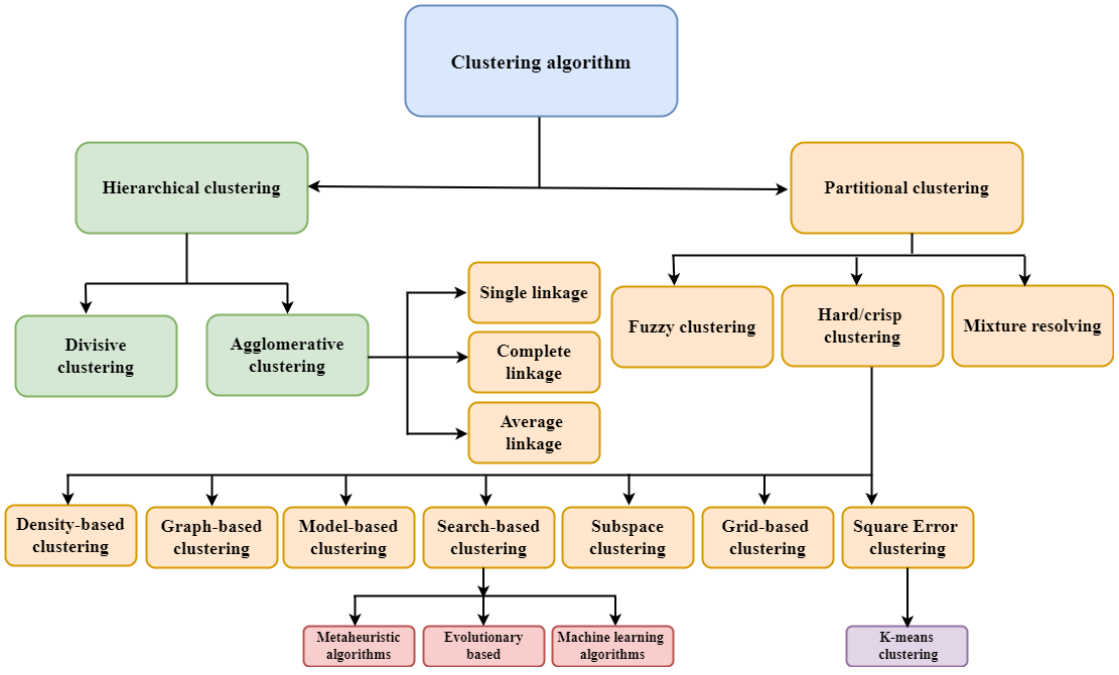


Fig. 1. Clustering algorithm categories

Table 1. Summary of recommended algorithms for automatic data clustering.

Author Name	Clustering method	Data-sets	Cluster validity index	Application Area	Used threshold
Anari et al. [74]	Continuous Action-Set Learning Automata (ACCALA)	Real life data sets	S_Dbw index	Cluster analysis and image segmentation	Yes
Alinya et al. [89]	Imperialist Competitive Algorithm for solving automatic clustering problems (AC-ICA)	Synthetic data sets	Rand Index (RI), Adjusted Rand Index (ARI), Entropy and F-measure	Real world application	Yes
Kuo et al. [56]	Artificial Bee Colony optimization algorithm and K-means algorithm (iABC)	Synthetic and real-life data sets	VI index and ARI	Customer segmentation	Yes
Sharma et al. [90]	Automatic Hybrid PSO with Mutation for generating the clusters automatically(AHPSOM)	Synthetic and real-life data sets	Intra cluster and Inter cluster distance, Adjusted Rand Index (ARI), Rand index (RI), and F-measure	Mobile network data and sensor network data	Yes
Merikhi et al. [91]	Nature-Inspired Binary Optimization Algorithms for Automatic clustering (NIBOA)	Synthetic and real life data-sets	DB-index	Cluster analysis	Yes
Harifi et al. [89]	Swarm-based Emperor Penguins Colony (EPC) algorithm for Automatic Clustering (EPC)	Real-world data sets	DB-index and CS index	Cluster analysis	Yes
Zhou et al. [47]	Automatic Data Clustering using Nature-Inspired Symbiotic Organism Search Algorithm (SOS)	Artificial data-sets and real-world data sets	Fitness function evaluation	Cluster analysis	Yes
Liu et al. [69]	Dynamic Local Search based Immune Automatic Clustering Algorithm (DLSIAC)	Real-world data-sets and artificial data sets	Reconstruction error, Minkowski Score (MS)	Cluster analysis and image segmentation	Yes
Omran et al. [23]	Dynamic Clustering using Particle Swarm Optimization (DCPSO)	Real life data sets	Dunn index (DI), Turi index, and S_Dbw index	Cluster analysis and image segmentation	Yes
Masoud et al. [30]	Dynamic Clustering using Combinatorial Particle Swarm Optimization (CPSOII)	Artificial and real-life data sets	Variance Ratio, Criterion Davies-Bouldin (DB) index	Cluster analysis and combinatorial optimization problem	Yes
Das et al. [12]	Automatic Clustering using Differential Evolution Algorithm (ACDE)	Real life data sets	DB and CS indices	Cluster analysis and image segmentation	Yes
Maulik et al. [35]	Automatic Fuzzy Clustering using Modified Differential Evolution (AFCMDE)	Real life data sets	XB index	Image segmentation	Yes
Ozturk et al. [58]	Dynamic Clustering with Improved Binary Artificial Bee Colony (DCIBABC)	Real life data sets	VI index and correct classification percentage (CCP)	Cluster analysis	Yes
Chowdhury et al. [59]	Automatic Clustering based on Invasive Weed Optimization (ACIWO)	Artificial and real-life data-sets	Sym-K index	Cluster analysis	Yes

Das et al. [22]	Bacterial Evolutionary Algorithm for Automatic Data Clustering (BEADC)	Artificial and real-life data sets	CS index	Cluster analysis	Yes
Kumar et al. [24]	Automatic Cluster Evolution using Gravitational Search (ACEGS)	Real life data sets	Point symmetry-based cluster validity index	Cluster analysis and image segmentation	Yes
Liu et al. [68]	Population-based Clustering Technique using Particle Swarm Optimization and k-means	Real life data sets	PMB index	Image segmentation	Yes
Zhou et al. [47]	Automatic Data Clustering using Nature-inspired Symbiotic Organism Search (ADCNSO)	Real life and artificial data sets	Fitness function evaluation	Cluster analysis	Yes
Pacheco et al. [60]	Ant Colony Optimization for Automatic Data Clustering Problem	Real life data sets	SI	Cluster analysis	Yes
Ezugwu et al. [57]	A Comparative Performance Study of Hybrid Firefly Algorithms for Automatic Data Clustering	Real life data sets	DB and Compact-Separated (CS)index	Cluster analysis	Yes
Nanda et al. [65]	Automatic clustering algorithm based on Multi-objective Immunized PSO (MOIMPSO)	Artificial and real data sets	Pareto Optimal Front (POF) and Minkowski Score (MS)	3D human models	Yes
Elaziz et al. [68]	Automatic Data Clustering based on Hybrid Atom Search Optimization and Sine-Cosine Algorithm (ASOCSA)	Real life data sets	Dunn, Silihouette, Davies Bouldin, and Calinski Harabasz(CH)index	Cluster analysis	Yes

3. Automatic data clustering and learning automata

This section will describe the concepts related to automatic clustering and learning automata.

3.1 Problem definition

Consider $D = \{X_1, X_2, \dots, X_n\}$ represent a dataset consists of n data objects, where each dataset d has real-value features. The i^{th} data object is denoted as: $\mathrm{x_i} = \{\mathrm{x_{i1}}, \mathrm{x_{i2}}, \dots, \mathrm{x_{ij}}, \dots, \mathrm{x_{id}}\}$, where $\mathrm{x_{ij}}$ denotes the j^{th} attribute of the data object $\mathrm{x_i}$, and d represents the number of features. The entire dataset X can be represented as a matrix in the following form.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i,1} & x_{i,j} & \dots & x_{i,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{bmatrix} \quad (1)$$

Let $\mathrm{K_{max}}$ be the maximum number of clusters in a dataset. The automatic clustering algorithm attempts to identify determine the optimal partitioning of the dataset into k clusters, represented

as $C = \{c_1, c_2, \ldots, c_k\}$, where $1 \leq k \leq K_{max}$. It optimizes cluster formation by minimizing the objective function, thereby maximizing intra-cluster cohesion and inter-cluster separation.

3.2 Clustering Validity Index

Several validation methods were used to evaluate the effectiveness of the clustering algorithm. The validity index was used to evaluate clustering quality. This paper uses the S_Dbw validity index as optimization function to evaluate clustering performance. This index aims to minimize intra-cluster variance while maximizing inter-cluster separation. Therefore, it is particularly well-suited for density-based clustering. The validity index is described as follows [72]:

$$S_Dbw = Scat(k) + Dens_bw(k) \tag{2}$$

In this equation, $Scat(k)$ represents the compactness of clusters. It is calculated as the ratio of the average intra-cluster variance to the variance of the entire dataset. Eq. (3) represents the average scattering for clusters.

$$Scat(k) = \frac{1}{k}\sum_{i=1}^{k}\frac{\sigma_i}{\delta_{data}} \tag{3}$$

Where σ_i is the variance of cluster i, σ_{data} is the variance of the entire dataset, and k is the number of clusters. $Dens_bw(k)$ measures how well-separated clusters are by examining the density of points between them. It is calculated by comparing the density at the midpoint between two clusters with the densities of the clusters themselves. A lower S_Dbw value indicates better clustering, as it reflects: 1) Higher compactness of clusters (low intra-cluster variance. 2) Better separation between clusters (low inter-cluster density). It is particularly useful in situations where other metrics like Silhouette Score or Davies-Bouldin Index might not adequately capture the cluster structure. Eq. (4) represents the inter-cluster density, Eq. (5) represents $density(C)$, Eq. (6) indicates the function $f(x, \mu)$, and Eq. (7) defines the mean standard deviation of clusters. In the equation above, $\sigma(C_i)$ denotes the variance of cluster C_i, k indicates the total number of clusters in C, $\sigma(D)$ denotes the variance of the dataset, μ denotes the centers of $C = \{c_1, c_2, \ldots, c_k\}$, and $|C|$ indicates the number of data items within cluster C.

$$Dens_bw(k) = \frac{1}{k(k-1)}\sum_{i=1}^{k}\left(\sum_{j=1}^{k}\frac{density(C_i \cup C_j)}{max\{density(C_i), density(C_j)\}}\right) \tag{4}$$

$$density(C) = \sum_{i=1}^{|C|} f(x_i, \mu) \tag{5}$$

$$f(x, \mu) = \begin{cases} 0 & \text{if } f(x, \mu) > stdev \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

$$\text{stedev} = \frac{1}{k}\sqrt{\sum_{i=1}^{k} \|\sigma(C_i)\|} \tag{7}$$

3.3 Learning Automata

A learning automaton (LA) is a self-operating machine that can be used in a random environment [73, 74]. The learning method selects the best action from a limited set of allowed actions. Each LA selects a single action from a finite set. This action is then sent to the random environment. The random environment evaluates the automaton's actions and produces a reinforcement signal. The LA uses a reinforcement learning method to determine whether the selected action results in a reward or a penalty. Each automaton's learning process is repeated until the optimal action is obtained. A LA is a machine that operates in an unpredictable environment and enhances its abilities in a specific manner [73, 75]. LA is a valuable tool and is used in numerous applications, such as computer networks [76], image processing [66], deep reinforcement learning [77], fuzzy applications [78], chaos theory [79], social networks [80], optimization problem [81], graph theory [82], clustering[83], and load balancing [84]. Fig. 2 illustrates the relationship between the LA model and the random environment.

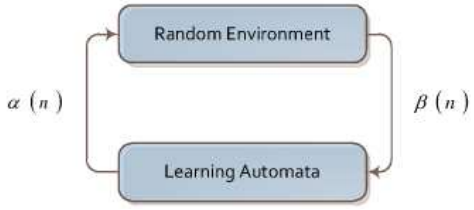


Fig. 2. Interaction between the LA model and the random environment.

The environment is described by <α, β, c>, where $\alpha \equiv \{\alpha^1, \alpha^2, \dots, \alpha_r\}$denotes the limited set of actions (inputs), $\beta \equiv \{\beta_1, \beta_2, \dots, \beta_r\}$ indicates a set of values selected from the reinforcement signal, and $c \equiv \{c_1, c_2, \dots, c_r\}$ represents a set of penalty probabilities. LA can be categorized into two types: fixed structure learning automata (FSLA) and variable structure learning automata (VSLA). In the FSLA model, the action probability remains constant, whereas the probability vector changes over time in the VSLA model. The following sections explain the concept of VSLA, which is used in this study. A VSLA is represented by the quaternion $< \alpha, \beta, p, T >$. Here, $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ denotes the allowable set of actions, $\beta = \{\beta_1, \beta_2, \dots, \beta_r\}$ denotes the input set, $sp \equiv \{p_1, p_2, \dots, p_r\}$ indicates the action probability vector. In this set, each p_i is an action probability vector that serves as the basis for the selected action α_i. Finally, T denotes the learning algorithm, and *r* represents the number of actions the LA can select. The learning algorithm is a recurrence formula employed to alter the action probability vector. When the LA selects action α_i, it receives a performance evaluation signal from a random environment. If the received signal is positive ($\beta = 0$), the action probability $p(t)$ is updated by the linear learning algorithm described by Eq. (8). If the LA obtains a negative response from the random environment ($\beta = 1$), action

α_i is penalized according to Eq. (9). In Eqs. (8) and (9), the parameters *a* and *b* represent the reward and penalty parameters, respectively. Additionally, these parameters determine the probability of increasing or decreasing actions. Suppose *a* = *b*, the formula mentioned above is called the linear reward-penalty (L_{R-P}). For $0 < b \ll a < 1$, they are known as the linear reward ε-penalty ($L_{R\varepsilon P}$). Finally, when $b = 0$, they are known as the linear reward-inaction (L_{R-I}) algorithm.

$$p_j(t+1) = \begin{cases} p_j(t) + a\left[1 - p_j(t)\right], & j = i \\ (1-a)p_j(t), & \forall j \neq i \end{cases} \tag{8}$$

$$p_j(t+1) = \begin{cases} (1-b)p_j(t), & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)p_j(t), & \forall j \neq i \end{cases} \tag{9}$$

4. The proposed ACVSLA algorithm

This section explains the proposed ACVSLA algorithm.

4.1 Proposed framework

The proposed framework for automatic data clustering is shown in Figure 3, which uses a team of learning automata.

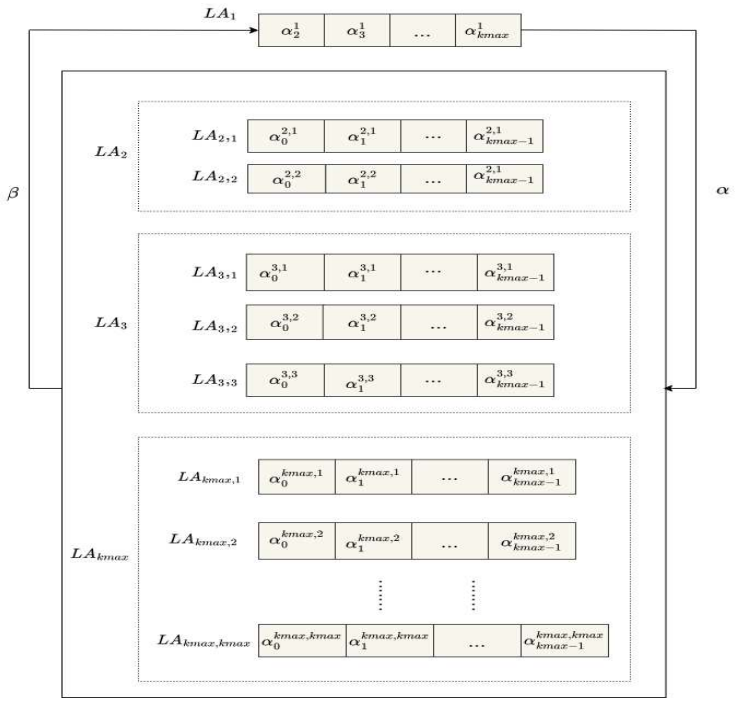


Fig. 3. The proposed framework.

Specifically, the framework employs a cooperative learning automata architecture. In this architecture, $LA_1$is determines the optimal number of clusters, while $LA_2$to LA_nact as cooperating automata that constitute the environment of LA_1. Their objective is to identify the corresponding clusters based on the action selected by LA_1. Each automaton from $LA_2$to LA_noperates within its own stochastic environment. The learning process for determining the optimal number of clusters and their associated cluster assignments proceeds as follows: First, LA_1selects an action from its action set and applies it to the environment. Based on the reinforcement signal generated by LA_1, one automaton from $LA_2$to $LA_n$is activated. Once the active automaton is identified, all cooperating automata select their respective actions. The environment then evaluates these actions and provides either rewards or penalties. Using feedback from the active automaton, LA_1updates its action by applying the appropriate reward or penalty. Meanwhile, the remaining automata continue to identify their corresponding clusters. This process is repeated iteratively until LA_1converges to the optimal number of clusters.

Let K_{max} shows the maximum number of clusters in the clustering problem. The algorithm requires $\frac{K_{max}\times(K_{max}+1)}{2}$ automaton. The action set of LA_1 is described as follows:

$$\text{actionSet}(LA_1) = \{\alpha_2^1, \alpha_3^1, \dots, \alpha_{kmax}^1\} \tag{10}$$

Each action of LA_1 is associated with a unique cluster. If LA_1 selects action α_i^1, where $i \in (2, 3, \dots, k_{max})$, the number of clusters identified will be i. $LA_{j,(2\le j\le k_{max})}$ consists of j automata, labeled as $LA_{j,1}, LA_{j,2}, \dots, LA_{j,j}$. The action set of $LA_{j,i,(1\le i\le j)}$ is described as follows:

$$\text{ActionSet}(LA_{j,i}) = \left\{\alpha_0^{j,i}, \alpha_1^{j,i}, \dots, \alpha_{k_{max}-1}^{j,i}\right\} \tag{11}$$

Each action corresponds to one cluster. Suppose an automaton in $LA_{j,i}$ chooses action $\alpha_k^{j,i}$ for some k where $0 \le k \le k_{max-1}$,it means that the automaton has selected the k^{th} cluster. For example, $\alpha_3^{j,i}$ means that the automaton $LA_{j,i}$ has selected the third cluster.

4.2 Automatic data clustering using the proposed framework

The proposed automatic clustering algorithm can be divided into three steps, which are: (1) action selection, and applying the actions to the random environment, and (2) the learning process. The flowchart of ACVSLA algorithm is shown in Fig. 5.

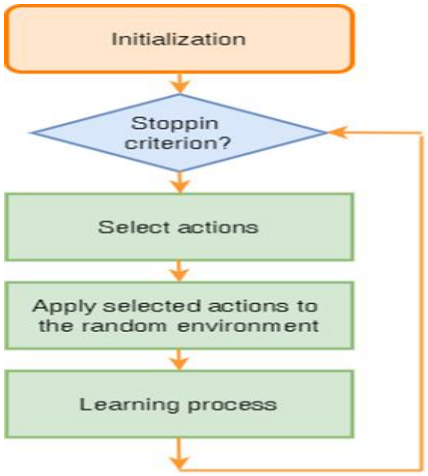


Fig. 5. Flowchart of the proposed ACVSLA algorithm.

4.2.1 Actions selection

The algorithm has no prior knowledge of the actual number of clusters. Suppose that the maximum number of clusters(k_{max}) is set to 6. The action set associated with LA_1 is $Actionset(\ LA_1) = \{\alpha_2^1\ , \alpha_3^1\ , \alpha_4^1\ , \alpha_5^1\ , \alpha_6^1\ \}$. Automaton LA_1 selects an action corresponding to the maximum number of clusters and applies it to the random environment. Depending on the selected action, one of the automata from LA_2 to LA_6 is activated. Let, the action probability vector of LA_1 be $(0.2,0.2,0.3,0.1,0.2)$. If automaton LA_1 chooses action α_4^1, this indicates that LA_1 considers 4 clusters as the optimal number of clusters. As a result, LA_4 is activated and the cooperating learning automata, denoted as $\{\mathrm{LA}_{4,1}\ , \mathrm{LA}_{4,2}\ , \mathrm{LA}_{4,3}\ ,\ \mathrm{LA}_{4,4}\}$, are activated. Let the action probability vectors of automata $\mathrm{LA}_{4,1}\ , \mathrm{LA}_{4,2}\ , \mathrm{LA}_{4,3}\ ,\ \mathrm{LA}_{4,4}$ be $(0.1,0.3,0.2,0.2,0.1,0.1)$, $(0.1,0.1,0.2,0.3,0.1,0.2)$, $(0.4,0.1,0.1,0.1,0.2,0.1)$ and $(0.1,0.1,0.5,0.2,0.05,0.05)$ respectively. These steps are shown in Figure 6. This figure shows that the automata $LA_{4,1}$, $LA_{4,2}$, $LA_{4,3}$, and $LA_{4,4}$choose clusters 1, 3, 0, and 2, respectively, for clustering with four clusters.

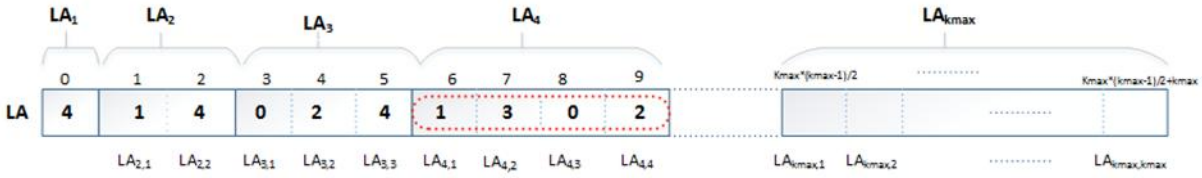


Fig. 6. An example of a VSLA implementation.

4.2.2 Apply Selected Actions to the Environment

After selecting the actions, the automaton determines the optimal number of clusters and assigns them to the data. By performing clustering on the selected clusters, the S-DbW clustering index can also be determined. For instance, in the above case, after the automaton chooses the number of clusters, four corresponding clusters, namely 1, 3, 0, and 2, are selected for clustering. The data are then clustered using these four clusters, and the S-DbW clustering index is subsequently calculated.

4.2.3 The Learning Process

Let $i \in (2, 3, \dots, k_{max})$ denote the optimal number of clusters specified by the LA_1 automaton, and let j denote the corresponding selected cluster. We save the minimum value obtained by s-dbw index for the number of clusters determined by i , with the j-th cluster in the localbest[i][j] matrix. To determine whether the selected action j has been appropriate or not, it is sufficient to compare the s-dbw index value with the value of localbest[i][j]. If the s-dbw value is lower than the value of localbest[i][j], it means that the selected action j has performed better in clustering with i clusters and can be rewarded; otherwise, action j is penalized. When action j is rewarded, the s-dbw index value is stored in localbest[i][j] as the best value found for clustering with i clusters by cluster j. Following the example above, assume the S_Dbw value is 0.6, with the previously obtained values as follows: localbest[4][1] = 1.6, localbest[4][3] = 0.9, localbest[4][0] = 0.7, and localbest[4][2] = 1.5. We find that only cluster 0 ($\alpha_0^{4,3}$) is rewarded, but the actions for clusters 1, 3, and 2 ($\alpha_1^{4,1}$, $\alpha_3^{4,2}$, $\alpha_2^{4,4}$) are penalized. We use a one-dimensional array, GlobalBest[i], to store the lowest S_Dbw index for each of the i clusters. After the S_Dbw index value is determined, it is compared with the value in GlobalBest[i]. If the S_Dbw value is lower than the value in GlobalBest[i], the action selected by automaton LA_1 is rewarded, and this new value is stored in GlobalBest[i]. Otherwise, the action selected by automaton LA_1 is penalized. In this way, automaton LA_1 learns to find the optimal action, i.e., the optimal number of clusters. Let the obtained S_Dbw value be 0.6, and suppose GlobalBest[4] = 0.4. Therefore, the action selected by the Learning Automaton LA_1 (α_1^4) is penalized. The algorithm will continuously repeat its three steps: 1) select an action, 2) execute the chosen action in the environment, and 3) the learning update until it converges to an optimal number of clusters and their corresponding assignments.

4.3 Pseudo-code of the proposed ACVSLA algorithm

The pseudocode for each phase's implementation is presented in Algorithms 1-6. The main body of the proposed algorithm is described in Algorithm 1. In Algorithm 2, the user inputs for the proposed algorithm are given. The user defines the required inputs, such as the maximum number of iterations (*Max_It*) and the number of clusters (*Kmax*). The learning automata implementation details are provided in Algorithm 3.

Algorithm 1: VSLA-Clustering

```
1 [Kmax,MaxIt,LocalBest,GlobalBest] ← Initialize()
2 Iteration ← 1
3 while Iteration ≤ MaxIt do
      // Select Action
4     [action,Actions] ← SelectActions()
      // Apply selected actions to the environment
5     S_Dbw ← CalculateClusteringMetric(action,Actions)
      // Learning Process
6     LearningProcess()
7     Iteration ← Iteration+1
8 end while
9 return [action,Actions] as a Best Solution
```

Algorithm 2: Initialize

```
   // Get maximum number of clusters as a Kmax
1  Kmax ← Input from user
   // Get maximum iterations
2  MaxIt ← Input from user
   // Build VSLA and associated actions
3  Build-VSLA(Kmax)
   // Store local and global best values
4  LocalBest[2,3,...,Kmax][0,1,2,...,Kmax] ← [∞][∞]
5  GlobalBest[2,3,...,Kmax] ← [∞]
6  return [Kmax,MaxIt,LocalBest,GlobalBest]
```

Algorithm 3: Build-VSLA(Kmax)

```
   // Create learning automaton LA1
   // Each action of LA1 represents the number of corresponding active clusters
1  Create automaton of LA1
2  Define actinoSet(LA1) = {α^1_2, α^1_3, ..., α^1_Kmax}
3  foreach i, j where i ∈ {1, ..., j} and j ∈ {2, ..., Kmax} do
4  |  Create automata of LA_{j,1}, LA_{j,2}, , ..., LA_{j,j},
5  |  Define actinoSet(LA_{j,i}) = {α^{j,i}_0, α^{j,i}_1, ..., α^{j,i}_{Kmax-1}}
6  end foreach
```

Algorithm 4 provides the implementation details for action selection. In Algorithm 4, LA_1 chooses randomly the action representing the optimal number of clusters (Line 1). The corresponding automata to this action are activated in Lines 5 to 7. The output of this function provides the optimal number of clusters and the respective cluster number for each step (Line 8). Each action for an automaton is kept in a vector called "Actions" (Line 2).

Algorithm 4: SelectActions

```
   // The action value of automaton LA1 is stored
1  action=SelectAction(LA1)
2  Actions←[0,1,...,action]
3  Actions[0] ← action
4  i ←1
5  for j = 1 : action do
6  |  Actions[i++] ← SelectAction(LA_{action,j})
7  end for
8  return (action,Actions[1:action])
```

Algorithm 5 provides a detailed implementation of clustering quality evaluation. In this phase, clustering is performed based on the action selected in the previous phase, and the S-DBW evaluation index. The Euclidean distance is used to measure the distance of each data point to the cluster centers, and the data point with the minimum distance is assigned to that cluster. The S-DBW index is then calculated to assess clustering quality.

Algorithm 5: CalculateClusteringMetric(action,Actions)

```
1  Min ← ∞
2  ClusterIndex←1
3  foreach Data Object such as obj do
4  |  for ClusterIndex=1:action do
5  |  |  Ed ← EuclideanDdistance(Actions[ClusterIndex],obj)
6  |  |  if Ed < Min then
7  |  |  |  Min ← Ed
8  |  |  |  ClusterNumber ← Actions[ClusterIndex]
9  |  |  end if
10 |  end for
11 |  Assign obj into ClusterNumber
12 |  UpdateCenters(ClusterNumber)
13 end foreach
14 S_Dbw ← CalculateS_Dbw()
15 return S_Dbw
```

Algorithm 6: LearningProcess(action,Actions)

```
1  for j = 1 : action do
2  |  if S_Dbw ≤ LocalBest[action][Actions[j] then
3  |  |  Reward action of α_{action,j} of automata of LA_{action,j}
4  |  |  LocalBest[action][Actions[j]] ← S_Dbw
5  |  |  else
6  |  |  |  penalize action of α_{action,j} of automata of LA_{action,j}
7  |  |  end if
8  |  end if
9  end for
10 if S_Dbw ≤ GlobalBest[action] then
11 |  Reward action of α^1_{action} of automata of LA1
12 |  GlobalBest[action] ← S_Dbw
13 |  else
14 |  |  Penalize action of α^1_{action} of automata of LA1
15 |  end if
16 end if
```

Once S_Dbw is calculated, the learning process begins by deciding whether to reward or penalize the selected action, as described in Algorithm 6. So, LocalBest[i][j] has the best value found for all clusters while clustering with *i* clusters. If S_Dbw is lower than the corresponding value in LocalBest, the chosen action is rewarded, and its outcome is saved; otherwise, the action is penalized (Lines 2–9). A one-dimensional array, GlobalBest[i], is also used to find the number of clusters identified by LA_1. For each of the *i* clusters, the algorithm checks whether the S_Dbw value is a new global best. If the obtained value is lower than the current GlobalBest for *i* clusters, then the choice is accepted, and the action taken by LA_1 is rewarded. Otherwise, the choice is not accepted, and the action is penalized. Thus, LA_1 learns to find the optimal number of clusters.

4.5 Complexity analysis

In this section, the time complexity of the proposed algorithm is explained. Let n denote the number of data objects, K_{max} represent the maximum number of clusters, and d denote the number of attributes. The VSLA algorithm comprises these steps: Step (1) comprises the Initialize() function. Steps (1 and 2) require O(1). Steps (3 to 5) require a maximum complexity of O($Kmax^2$). Thus, the overall complexity of the Initialize () function is O($Kmax^2$). Step (3) of the VSLA algorithm requires O(*MaxIt*). The SelectActions() function in Step 4 consists of the following substeps: Step (1) requires O(K_{max}). Lines 2- 3 and 4 - 6 also require O(K_{max}). Thus, this function requires O(K_{max}). Step 5 involves the CalculateClusteringMetric () function, which uses the S-DbW () algorithm. The complexity of this function is *O(Kmax²) + O (n* × K_{max}*)*. The maximum complexity of Step 6 in the VSLA algorithm is O($Kmax^2$). Step 7 requires O (1). Thus, the total time complexity of proposed algorithm is: O(*MaxIt*) × [O($Kmax^2$) + [O($Kmax^2$) + O (n × K_{max})] + O (1)]. By simplifying this expression, we obtain: O(*MaxIt*)× [O ($Kmax^2$) +O(*n*×K_{max})].

5. Performance evaluation and discussions

5.1 Performance metrics

All experiments were conducted using Java on a Dell laptop with an Intel Core i7-10700 U processor at 3.5 GHz, 16 GB of RAM, and Windows 11. The proposed ACVSLA algorithm was evaluated for effectiveness and compared to eight other automatic data clustering algorithms: Automatic Clustering based on Continuous Action-Set Learning Automata (ACCALA) [66], Imperialist Competitive Algorithm (ICA) to address automatic clustering issues (AC-ICA) [85], Artificial Bee Colony Optimization Algorithm and the *K*-means Algorithm (iABC)[60], Automatic Hybrid Particle Swarm Optimization with Mutation (AHPSOM)[63], Nature-Inspired Binary Optimization Algorithms (NIBOA)[39], Emperor Penguins Colony (EPC) algorithm (EPC)[86], Automatic Data Clustering using Nature-Inspired Symbiotic Organism Search algorithm (SOS)[87], HDBSCAN[88] , and Dynamic Local Search using Immune Automatic Clustering algorithm (DLSIAC)[89]. These algorithms were run independently 40 times on each of the eight datasets, and the average and standard deviation of the results were calculated. The efficiency of the ACVSLA algorithm was assessed based on various criteria, including the number of clusters generated, unpaired t-test, intra-cluster, and inter-cluster distance. In addition, the ACVSLA algorithm was assessed using the validity DB index [35], which measures the ratio of the distance between clusters to the distance within clusters.

5.2 Dataset and parameter preparation

To assess the efficiency of the proposed method, we performed tests on 16 real-world datasets obtained from the UCI Machine Learning Repository [90] and the KEEL repositories[91]. Table 4 provides an overview of the main characteristics of the datasets, including the number of data objects, attributes, and clusters. The compared algorithms are ACCALA, AC-ISA, iABC, AHPSOM, NIBOA, SOS, EPC, DLSIAC, and HDBSCAN. Additionally, each algorithm's maximum number of iterations is set to 200, and each algorithm was run independently 40 times.

Table 4. Features of the sixteen clustering datasets.

Data-set		Number of Data objects	Number of Attributes	Number of Clusters
Real-world dataset	Iris (IR)	150	4	3
	Wine (WI)	178	13	3
	Glass (GL)	214	10	6
	Breast cancer (BC)	683	9	2
	Vowel (VO)	871	3	6
	Ecoli (EC)	335	7	8
	Yeast (YE)	1484	8	10
	Soybean (SO)	47	35	4
Shape dataset	Spiral (SP)	312	2	3
	Flame (FL)	240	2	2
Higher dimensional dataset	DARWIN(DA)	174	451	2
	TUANDROMD(TU)	4464	241	2
	Dim032(DI)	1024	32	16
Synthetic dataset	S1(S1)	5000	2	15
	A1(A1)	3000	2	20

	Birch1(BI)	100000	2	100

5.3 Analysis of experimental results and evaluations

This section describes the obtained experimental tests. In this research, the simulation results of the proposed method were analyzed and compared with those of nine other automatic clustering methods.

Experiment 1.

In this experiment, we calculated the optimal number of clusters using ACVSLA and other compared algorithms, based on the S_Dbw metric. The results are presented in Table 5.

Table 5. Number of clusters obtained from the different algorithms with S_Dbw metric.

Data-set	ACVSLA	ACCALA	AC-ICA	iABC	AHPSOM	NIBOA	SOS	EPC	DLSIAC	HDBSCAN
IR	**3.1±0.38**	**2.9±0.12**	**2.73±0.25**	**3.23±0.02**	2.08±0.03	2.27±0.02	2.65±0.01	**2.9±0.18**	**3.2±0.22**	2.0±0.00
WI	**3.3±0.42**	**3.2±0.24**	**3.3±0.01**	**3.25±0.033**	**3.05±0.065**	2.9±0.03	**3.4±0.055**	**3.2±0.02**	**3.2±0.07**	2.0±0.00
GL	**5.8±0.35**	**6.0±0.33**	**6.3±0.55**	**6.33±0.02**	5.8±0.025	5.84±0.03	5.55±0.056	**5.76±0.32**	**6.1±0.22**	3.0±0.00
BC	**2.0±0.00**	**2±0.00**	**2±0.00**	**2±0.00**	**2.2±0.02**	**2.0±0.01**	**2.23±0.03**	**2.0±0.051**	**2.0±0.0**	**2.0±0.00**
VO	**6.3±0.41**	**5.75±0.25**	**5.9±0.11**	**5.89±0.18**	7.4±0.03	5.02±0.001	7.1±0.044	**6.01±0.11**	**5.98±0.21**	4.0±0.00
EC	**7.97±0.32**	**8.2±0.33**	**8.4±23**	**8.3±0.13**	7.2±0.14	7.5±0.11	**8.6±0.023**	**8.4±0.22**	**7.8±0.23**	2.0±0.00
YE	**10.1±1.02**	**10.4±2.22**	**10.24±2.16**	**10.6±1.11**	6.4±1.34	13.3±1.19	13.8±3.2	**10.8±1.78**	**10.65±1.12**	3.0±0.00
SO	**3.9±0.41**	**4.22±0.21**	**4.2±0.12**	**4.22±0.18**	3.4±0.02	3.4±0.03	3.8±0.12	**3.87±0.33**	**4.2±0.25**	2.0±0.00
SP	**2.8±0.23**	**3.3±0.17**	2.3±0.17	2.7±0.12	2.2±0.15	**2.8±0.34**	2.2±0.27	3.5±0.24	4.5±0.19	11.00±0.00
FL	**2.1±0.11**	2.9±0.02	3.9±0.18	5.3±0.11	3.5±0.03	**3.2±0.13**	2.9±0.09	4.1±0.31	6.3±0.12	6.00±0.00
DA	**2.2±0.33**	**3.9±0.11**	4.3±0.12	4.2±0.15	4.9±0.11	2.9±0.07	2.7±0.06	5.9±0.18	2.7±0.34	**2.00±0.00**
TU	**2.4±0.21**	2.7±0.21	3.1±0.26	2.8±0.21	3.1±0.01	2.6±0.61	6.1±0.31	3.3±0.21	3.1±0.41	82.00±0.00
S1	**14.7±0.37**	**13.9±0.68**	12.45±0.32	12.65±0.36	12.40±0.23	12.10±0.46	12.36±0.42	13.2±0.29	11.4±0.46	53.0±0.00
A1	**19.4±0.64**	16.4±0.17	14.4±0.17	11.4±0.17	12.4±0.13	14.4±0.25	13.4±0.36	10.4±0.42	9.4±0.66	23.0±0.00
BI	**98.9±0.94**	**98.9±0.96**	85.3±0.32	67.4±0.23	45±0.36	79.1±0.13	89.1±0.23	59.3±0.17	23.1±0.76	3575.0±0.00
DI	**16.3±0.28**	**16.5±0.48**	14.3±0.22	11.2±0.18	14.3±0.78	13.1±0.57	18.3±0.34	14.1±0.75	12.2±0.19	**16.0±0.00**

The results in Table 4 demonstrate that the algorithms EPC, AC-ICA, iABC, EPC, and DLSIAC were able to determine the optimal number of clusters for the IR, WI, GL, BC, VO, EC, YE, and SO datasets. However, non-bold results indicate cases where these algorithms failed to identify precise cluster numbers. The ACCALA algorithm can accurately determine the number of clusters for all datasets except FL, TU, and A1. NIBOA identified precise cluster numbers for BC, SP, and FL datasets. Notably, for the AHPSOM algorithm, WI and BC could accurately determine the number of clusters. The SOS algorithm for the WI, BC, and EC datasets could accurately determine the number of clusters. Finally, for the HDBSCAN algorithm, BC, DA, and DI could accurately determine the number of clusters. In contrast, the proposed algorithm consistently achieved high accuracy in identifying the number of clusters across all datasets. These results highlight the proposed algorithm's ability to precisely determine the number of clusters, particularly for large and complex-shaped datasets.

Experiment 2.

This experiment statistically evaluated the proposed ACVSLA and other clustering algorithms using the unpaired t-test. This measure identifies statistically significant relationships between the two algorithms. Table 6 provides the results obtained from the unpaired t-test, which compares ACVSLA to ACCALA. This comparison identifies any significant differences between the algorithms. The results demonstrate that ACVSLA was not statistically significant for the WI, BI, and YE datasets but was statistically significant for the remaining datasets.

Table 6. The clustering results obtained from an unpaired t-test between ACVSLA and ACCALA.

Data-set	Standard error	t	95% confidence interval	Two-tailed P	Significance
IR	0.063	3.1742	0.0746 to 0.3254	0.0022	**Very significant**
WI	0.076	1.3074	-0.0523 to 0.2523	0.1949	**Not significant**
GL	0.076	2.6295	-0.3514 to -0.0486	0.0103	**Statistically significant**
BC	-	-	-	--	**-**
VO	0.076	7.2437	0.3988 to 0.7012	<0.0001	**Extremely significant**
EC	0.073	3.1645	-0.3747 to -0.0853	0.0022	**Very significant**
YE	0.386	0.7779	-1.0677 to 0.4677	0.4390	**Not significant**
SO	0.084	3.8049	-0.4884 to -0.1516	0.0003	**Extremely significant**
SP	0.052	9.5753	-0.6045 to -0.3955	<0.0001	**Extremely significant**
FL	0.020	39.1277	-0.84093 to -0.75907	<0.0001	**Extremely significant**
DA	0.064	26.7681	-1.8271 to -1.5729	<0.0001	**Extremely significant**
TU	0.054	5.5328	-0.4085 to -0.1915	<0.0001	**Extremely significant**
DI	0.088	2.2763	-0.3749 to -0.0251	0.0256	**Statistically significant**
S1	0.065	13.8339	0.7705 to 1.0295	<0.0001	**Extremely significant**
A1	0.105	28.6528	2.7916 to 3.2084	<0.0001	**Extremely significant**
BI	0.212	0.0000	-0.4229 to 0.4229	1.0000	**Not statistically significant**

Experiment 3.

In this experiment, we compute the intra-cluster distance. The lower value represents well-compact clusters. Table 7 displays the ratios of intra-cluster distances for all the automatic clustering algorithms. The experimental results prove that the ACVSLA algorithm produced well-compact clusters on the GL, YE, DI, A1, S1, and BC datasets.

Table 7. Intra-cluster distances were calculated for each algorithm.

Data-set	ACVSLA	ACCALA	AC-ICA	iABC	AHPSOM	NIBOA	SOS	EPC	DLSIAC	HDBSCAN
IR	0.8456±0.21	**0.8322±0.43**	1.322±0.12	3.44±0.05	3.89±1.17	3.72±3.01	3.96±1.98	1.28±0.22	1.11±0.27	1.31±0.16
WI	1.2318±0.021	**1.22±0.017**	1.26±0.012	4.5±0.01	5.3±0.19	4.8±1.85	5.4±1.15	1.32±0.11	1.27±0.14	7.62±0.25
GL	**14.22±0.23**	14.42±0.12	15.85±1.22	590.26±0.16	677±7.15	670.12±4.3	685.8±18.11	17.22±1.5	15.64±0.23	0.86±0.17
BC	**4.11±0.43**	4.13±0.22	7.44±0.14	4.25±0.16	4.99±0.44	5.18±0.8	4.5±0.78	4.86±0.43	4.66±0.23	5.86±0.16
VO	301.2±1.2	291.77±1.8	**176.45±3.54**	1522.66±0.9	1578.26±4.2	1579.2±9.4	1576.25±11.8	304.8±4.5	200.13±3.11	424±2.10
EC	0.18±0.1	0.171±0.22	0.173±0.1	**0.16±0.02**	0.24±0.18	0.45±0.86	0.28±0.14	0.1697±0.19	0.17±0.12	0.35±0.41
YE	**0.157±0.01**	0.18±0.02	0.15±0.04	0.19±0.09	0.3±0.08	0.35±0.11	0.31±0.12	0.16±0.27	0.1623±0.34	0.45±0.21
SO	2.24±0.1	2.12±0.11	2.14±0.02	1.95±0.08	4.01±0.12	5.3±0.12	4.5±1.01	2.15±0.08	**2.1485±0.05**	**3.1**±0.56
SP	3.45±0.12	2.74±0.33	1.74±0.03	2.95±0.05	3.39±0.07	1.85±0.18	1.95±0.08	2.45±0.06	5.31±0.07	**0.54±0.06**
FL	4.31±0.11	5.20±0.09	3.91±0.16	5.21±0.04	4.51±0.10	3.61±0.17	2.45±0.23	1.38±0.18	6.20±0.03	**0.94±0.72**
DA	1591.26±2.9	1628.26±3.2	1563.26±3.1	**1201.26±3.5**	1378.10±4.2	1620.16±2.9	1501.54±1.2	1510.26±3.1	1433.26±2.3	1672.26±3.9
TU	1.28±0.26	0.96±0.39	1.19±0.56	2.26±0.19	0.86±1.41	0.96±2.10	2.10±0.23	1.46±0.32	2.23±0.56	**0.02±0.36**
DI	**0.09±0.38**	0.13±0.22	0.23±0.52	0.43±0.22	0.41±0.56	0.33±0.24	0.63±0.42	0.28±0.44	0.53±0.27	0.10±0.24
A1	**0.03±0.216**	0.06±0.27	0.05±0.44	0.07±0.22	0.07±0.13	0.05±0.28	0.08±0.44	0.06±0.16	0.06±0.41	0.04±0.46
S1	**0.01±0.21**	0.02±0.17	0.04±0.21	0.04±0.24	0.09±0.78	0.08±0.29	0.09±0.21	0.03±0.63	0.05±0.27	0.02±0.46
BI	0.04±0.43	0.04±0.7	0.06±0.23	0.05±0.26	0.08±0.23	0.07±0.19	0.06±0.73	0.05±0.48	0.06±0.83	**0.03±0.75**

Experiment 4.

In this experiment, we calculate the inter-cluster distance ratio. A high value for this criterion suggests that the clusters are well-separated. Table 8 displays the distances between clusters for the various clustering algorithms. According to these results, our algorithm generates distinct clusters for the EC, DI, A1, S1, and BI datasets.

Table 8. Inter-cluster distances obtained for each algorithm.

Data-set	ACVSLA	ACCALA	AC-ICA	iABC	AHPSOM	NIBOA	SOS	EPC	DLSIAC	HDBSCAN
IR	3.1±1.02	3.06±1.22	**4.43±0.3**	2.73±0.04	2.23±0.77	2.84±1.43	2.06±1.06	3.04±1.08	3.04±1.1	3.97±0.04
WI	3.65±0.19	**3.8458±0.16**	3.46±0.003	3.26±0.04	2.49±1.8	2.8±1.89	2.41±1.47	3.51±0.16	3.71±0.18	11.91±0.33s
GL	35.25±1.23	32.99±2.88	22.86±10.22	852.85±10.34	**895.38±5.16**	854.91±1.89	886.13±12.45	29.14±2.8	32.46±3.11	3.64±1.10
BC	6.11±1.2	6.10±1.65	**6.99±0.21**	3.15±0.14	2.54±0.033	2.65±1.76	2.76±1.96	6.2±0.8	6.2±0.66	6.2±0.66
VO	690.3±20.1	697±10.56	630.89±11.98	**2718±0.12**	1914.23±20.58	1962.43±18.4	2951.42±22.14	693.13±18.77	834.45±22.19	834.45±22.19
EC	**3.15±0.3**	3.11±0.23	3.10±0.44	3.14±0.71	2.66±0.44	2.27±0.43	2.1±0.21	3.11±0.61	3.9±0.16	3.9±0.16
YE	3.63±0.11	3.44±0.21	3.66±0.66	3.33±0.65	2.8±0.45	2.2±0.23	2.5±0.32	**3.69±0.89**	3.43±0.66	3.43±0.66
SO	7.8±1.1	7.99±1.2	7.6±1.3	7.7±1.09	8.01±0.56	6.56±0.85	6.89±0.9	8.02±0.12	**8.23±0.56**	8.03±0.23
SP	8.12±1.7	7.34±1.31	6.12±0.63	7.12±0.45	7.9±0.08	6.81±0.28	6.85±0.68	7.35±0.08	**10.41±0.06**	1.10±0.34
FL	8.43±0.23	9.31±0.36	7.38±0.33	8.84±0.21	9.11±0.32	6.86±0.63	6.15±0.39	3.68±0.69	**10.34±0.08**	1.61±0.42
DA	2113.36±1.8	2316.11±2.2	**2443.26±2.9**	2315.73±1.9	2264.23±3.1	2355.32±3.4	2210.32±4.3	1936.91±4.2	1835.27±4.1	4.48±0.31
TU	3.41±0.32	2.33±0.56	4.23±0.03	5.51±0.08	3.22±2.19	2.94±3.11	**6.32±0.27**	4.31±0.19	6.18±0.38	1.39±0.65
DI	**6.57±0.67**	5.32±0.42	5.26±0.35	4.86±0.92	4.41±0.72	4.26±0.37	4.86±0.12	5.77±0.38	4.26±0.82	6.36±0.32
A1	**8.94±0.25**	8.23±0.26	8.17±0.77	7.22±0.28	6.55±0.24	4.56±0.37	5.36±0.46	8.12±0.14	7.97±0.16	8.67±0.56
S1	**25.56±0.34**	20.78±0.34	19.45±0.99	24.28±0.81	18.56±0.27	22.29±0.17	20.25±0.24	19.45±0.58	19.64±0.38	23.41±0.11
BI	**33.60±0.22**	32.53±0.16	31.44±0.16	31.37±0.67	30.29±0.25	30.64±0.11	32.26±0.58	32.17±0.59	31.82±0.29	1484.15±0.71

Experiment 5.

In this experiment, we calculated the distances between clusters and compared them with the distances within each cluster. A significant amount of this feature suggests that the clusters are distinct and tightly compact. Fig. 7 illustrates the inter/intra distance ratio provided by every clustering algorithm being compared. The results show that ACVSLA provides superior clustering performance to other clustering algorithms for the Glass and Breast cancer datasets. ACCALA generates closely compact and distinctly separated clusters for the Iris and Wine datasets. DLSIAC obtained superior results on the VO and EC datasets. In addition, AC-ICA for the Yeast dataset and iABC for the Soybean dataset produced good results.

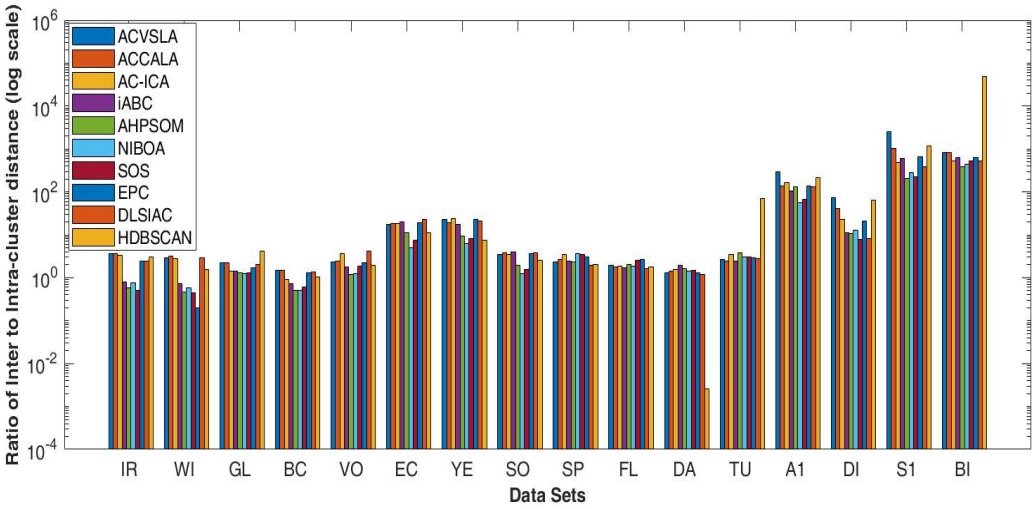


Fig. 7. The ratio of inter/intra-cluster distance.

Experiment 6.

This experiment was conducted to analyze the convergence of the proposed algorithm by examining the total number of iterations. Fig. 8 shows that the proposed ACVSLA converges to the minimum of S-Dbw in under 5,000 iterations. Fig. 9 shows the minimum value of the S-Dbw measure for the test datasets. It was observed that ACVSLA obtained the minimum S-Dbw for the Glass dataset.

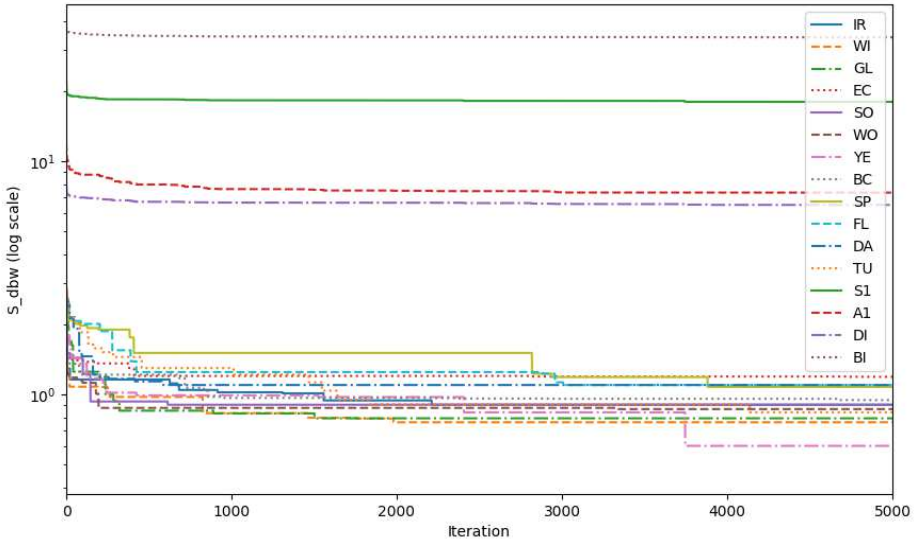


Fig. 8. Convergence of ACVSLA on the number of iterations in the test data sets.

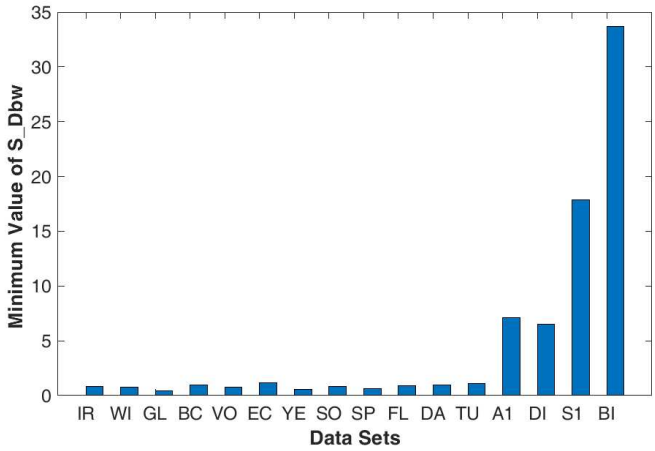


Fig. 9. Minimum value of S-Dbw in test data sets.

Experiment 7.

This experiment investigates how different values of K_{max} (i.e., 5, 10, 15, 20, 25, 30, …, 150) impact the proposed algorithm. Fig. 10 illustrates the impact of K_{max} on the resulting clusters. For the WI, IR, SP, FL, DA, TU, and Breast Cancer datasets, increasing these values did not affect the appropriate number of clusters obtained. The cluster numbers increased by 5 ≤ Kmax < 10 for the GL and SO datasets. The cluster numbers increased by 5 ≤ Kmax < 20 for the A1 datasets. They increased by 15 ≤ Kmax < 20 in the VO, DI, S1, and EC datasets. Aslo, increased by 45 ≤ Kmax < 100 in the BI dataset. Finally, they increased to 20 ≤ Kmax < 30 for the YE dataset.

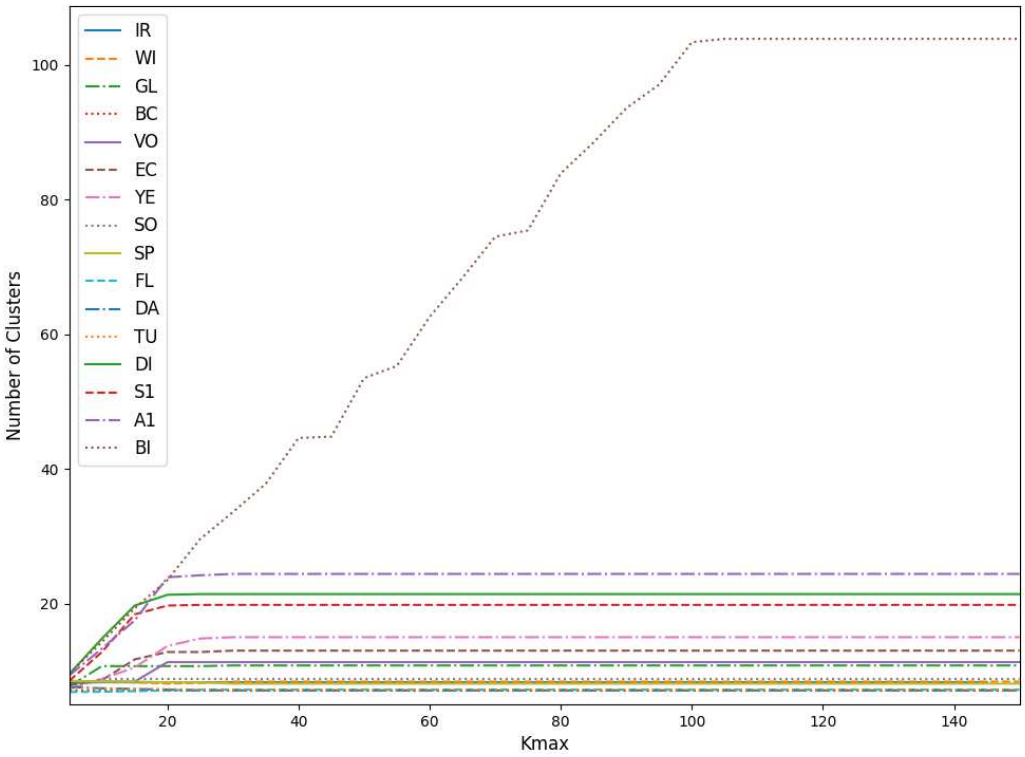


Fig. 10. The effect of K_{max} by increasing cluster numbers in the proposed ACVSLA algorithm.

6. Conclusions

This paper introduces a new method for automatic data clustering, called ACVSLA, based on Variable Structure Learning Automata (VSLA). ACVSLA uses a team of learning automata to address the challenges of automatic data clustering. Unlike existing algorithms, it autonomously finds the optimal number of clusters without requiring parameter settings and achieves robust, reliable, and high-quality clustering on datasets with varying characteistics. The performance of ACVSLA was evaluated by comparing it with several state-of-the-art automatic clustering algorithms, namely ACCALA, AC-ICA, iABC, AHPSOM, NIBOA, SOS, EPC, and HDBSCAN. The results show better intra-cluster compactness and inter-cluster separation. Statistical analyses using unpaired t-tests further confirm the significance of these performance improvements.

Experimental evaluations conducted on various benchmark datasets demonstrate that ACVSLA consistently and accurately determines the optimal number of clusters, providing stable and high-quality clustering results. In future works, we will work to generalize the method for large-scale datasets as well as to other unsupervised and optimization problems.

Reference

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR),* vol. 31, no. 3, pp. 264-323, 1999.

[2] A. Saxena *et al.*, "A review of clustering techniques and developments," *Neurocomputing,* vol. 267, pp. 664-681, 2017.

[3] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing,* vol. 300, pp. 70-79, 2018.

[4] S. Harifi, E. Byagowi, and M. Khalilian, "Comparative study of apache spark MLlib clustering algorithms," in *Data Mining and Big Data: Second International Conference, DMBD 2017, Fukuoka, Japan, July 27–August 1, 2017, Proceedings 2*, 2017: Springer, pp. 61-73.

[5] M. Flasiński and M. Flasiński, "Pattern recognition and cluster analysis," *Introduction to Artificial Intelligence,* pp. 141-156, 2016.

[6] F. Garcia-Lamont, J. Cervantes, A. López, and L. Rodriguez, "Segmentation of images by color features: A survey," *Neurocomputing,* vol. 292, pp. 1-27, 2018.

[7] L. M. Abualigah, A. T. Khader, M. A. Al-Betar, and O. A. Alomari, "Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering," *Expert Systems with Applications,* vol. 84, pp. 24-36, 2017.

[8] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A hybrid strategy for krill herd algorithm with harmony search algorithm to improve the data clustering?," *Intelligent Decision Technologies,* vol. 12, no. 1, pp. 3-14, 2018.

[9] C. C. Aggarwal and C. K. Reddy, "Data clustering," *Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra,* 2014.

[10] V. Mehta, S. Bawa, and J. Singh, "Analytical review of clustering techniques and proximity measures," *Artificial Intelligence Review,* vol. 53, pp. 5995-6023, 2020.

[11] V. Mehta, S. Bawa, and J. Singh, "WEClustering: word embeddings based text clustering technique for large datasets," *Complex & intelligent systems,* vol. 7, no. 6, pp. 3211-3224, 2021.

[12] V. Mehta, S. Bawa, and J. Singh, "Stamantic clustering: combining statistical and semantic features for clustering of large text datasets," *Expert Systems with Applications,* vol. 174, p. 114710, 2021.

[13] V. Mehta, M. Agarwal, and R. K. Kaliyar, "A comprehensive and analytical review of text clustering techniques," *International Journal of Data Science and Analytics,* pp. 1-20, 2024.

[14] B. Chander and K. Gopalakrishnan, "Data clustering using unsupervised machine learning," in *Statistical Modeling in Machine Learning*: Elsevier, 2023, pp. 179-204.

[15] A. Moya, E. Pretel, E. Navarro, and J. Jaén, "A systematic literature review of clustering techniques for patients with traumatic brain injury," *Artificial Intelligence Review,* vol. 56, no. Suppl 1, pp. 351-419, 2023.

[16] G. J. Oyewole and G. A. Thopil, "Data clustering: application and trends," *Artificial Intelligence Review,* vol. 56, no. 7, pp. 6439-6475, 2023.

[17] G. Gan, C. Ma, and J. Wu, *Data clustering: theory, algorithms, and applications*. SIAM, 2020.

[18] C. K. Reddy and B. Vinzamuri, "A survey of partitional and hierarchical clustering algorithms," in *Data clustering*: Chapman and Hall/CRC, 2018, pp. 87-110.

[19] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika,* vol. 32, no. 3, pp. 241-254, 1967.

[20] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association,* vol. 58, no. 301, pp. 236-244, 1963.

[21] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks,* vol. 16, no. 3, pp. 645-678, 2005.
[22] W.-L. Zhao, C.-H. Deng, and C.-W. Ngo, "k-means: A revisit," *Neurocomputing,* vol. 291, pp. 195-206, 2018.
[23] A. Moreira, M. Y. Santos, and S. Carneiro, "Density-based clustering algorithms–DBSCAN and SNN," *University of Minho-Portugal,* vol. 1, p. 18, 2005.
[24] N. P. Lin, C.-I. Chang, H.-E. Chueh, H.-J. Chen, and W.-H. Hao, "A deflected grid-based algorithm for clustering analysis," *WSEAS Transactions on Computers,* vol. 7, no. 4, pp. 125-132, 2008.
[25] Y. Liu, X. Wu, and Y. Shen, "Automatic clustering using genetic algorithms," *Applied mathematics and computation,* vol. 218, no. 4, pp. 1267-1279, 2011.
[26] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data," *Data Mining and knowledge discovery,* vol. 11, pp. 5-33, 2005.
[27] S. Das, A. Chowdhury, and A. Abraham, "A bacterial evolutionary algorithm for automatic data clustering," in *2009 IEEE congress on evolutionary computation*, 2009: IEEE, pp. 2403-2410.
[28] M. G. Omran, A. Salman, and A. P. Engelbrecht, "Dynamic clustering using particle swarm optimization with application in image segmentation," *Pattern Analysis and Applications,* vol. 8, pp. 332-344, 2006.
[29] V. Kumar, J. K. Chhabra, and D. Kumar, "Automatic cluster evolution using gravitational search algorithm and its application on image segmentation," *Engineering Applications of Artificial Intelligence,* vol. 29, pp. 93-103, 2014.
[30] S. Bandyopadhyay and U. Maulik, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern recognition,* vol. 35, no. 6, pp. 1197-1208, 2002.
[31] S.-M. Pan and K.-S. Cheng, "Evolution-based tabu search approach to automatic clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 37, no. 5, pp. 827-838, 2007.
[32] S. Saha and S. Bandyopadhyay, "A generalized automatic clustering algorithm in a multiobjective framework," *Applied Soft Computing,* vol. 13, no. 1, pp. 89-108, 2013.
[33] R. Liu, L. Jiao, X. Zhang, and Y. Li, "Gene transposon based clone selection algorithm for automatic clustering," *Information Sciences,* vol. 204, pp. 1-22, 2012.
[34] H. He and Y. Tan, "A two-stage genetic algorithm for automatic clustering," *Neurocomputing,* vol. 81, pp. 49-59, 2012.
[35] H. Masoud, S. Jalili, and S. M. H. Hasheminejad, "Dynamic clustering using combinatorial particle swarm optimization," *Applied intelligence,* vol. 38, pp. 289-314, 2013.
[36] J. Revathi, V. Eswaramurthy, and P. Padmavathi, "Hybrid data clustering approaches using bacterial colony optimization and k-means," in *IOP Conference Series: Materials Science and Engineering*, 2021, vol. 1070, no. 1: IOP Publishing, p. 012064.
[37] A. Karna and K. Gibert, "Automatic identification of the number of clusters in hierarchical clustering," *Neural Computing and Applications,* vol. 34, no. 1, pp. 119-134, 2022.
[38] A. E. Ezugwu, A. K. Shukla, M. B. Agbaje, O. N. Oyelade, A. José-García, and J. O. Agushaka, "Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature," *Neural Computing and Applications,* vol. 33, pp. 6247-6306, 2021.
[39] B. Merikhi and M. R. Soleymani, "Automatic data clustering framework using nature-inspired binary optimization algorithms," *IEEE Access,* vol. 9, pp. 93703-93722, 2021.
[40] A. E.-S. Ezugwu, M. B. Agbaje, N. Aljojo, R. Els, H. Chiroma, and M. Abd Elaziz, "A comparative performance study of hybrid firefly algorithms for automatic data clustering," *IEEE Access,* vol. 8, pp. 121089-121118, 2020.
[41] R. Bellman, R. Kalaba, and L. Zadeh, "Abstraction and pattern classification," *Journal of Mathematical Analysis and Applications,* vol. 13, no. 1, pp. 1-7, 1966.
[42] S. K. Pal and S. Mitra, "Fuzzy dynamic clustering algorithm," *Pattern Recognition Letters,* vol. 11, no. 8, pp. 525-535, 1990.

[43] U. Maulik and I. Saha, "Automatic fuzzy clustering using modified differential evolution for image classification," *IEEE transactions on Geoscience and Remote sensing,* vol. 48, no. 9, pp. 3503-3510, 2010.
[44] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, vol. 1, no. 14: Oakland, CA, USA, pp. 281-297.
[45] K. S. Al-Sultan and C. A. Fedjki, "A tabu search-based algorithm for the fuzzy clustering problem," *Pattern Recognition,* vol. 30, no. 12, pp. 2023-2030, 1997.
[46] B. Hufnagl and H. Lohninger, "A graph-based clustering method with special focus on hyperspectral imaging," *Analytica chimica acta,* vol. 1097, pp. 37-48, 2020.
[47] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *computer,* vol. 32, no. 8, pp. 68-75, 1999.
[48] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, 1996, vol. 96, no. 34, pp. 226-231.
[49] J. D. Banfield and A. E. Raftery, "Model-based Gaussian and non-Gaussian clustering," *Biometrics,* pp. 803-821, 1993.
[50] K. Beyer, "Goldstein. J., Ramakrishnan, R., & Shaft, U.(1999). When is "nearest neighbor" meaningful," in *Proceedings of the 7th international conference on database theory*.
[51] K. Ravi Kanth, D. Agrawal, and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," *ACM SIGMOD Record,* vol. 27, no. 2, pp. 166-176, 1998.
[52] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *Acm sigkdd explorations newsletter,* vol. 6, no. 1, pp. 90-105, 2004.
[53] A. Kaur and Y. Kumar, "A new metaheuristic algorithm based on water wave optimization for data clustering," *Evolutionary Intelligence,* vol. 15, no. 1, pp. 759-783, 2022.
[54] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, and S. Mirjalili, "Multi-verse optimizer: theory, literature review, and application in data clustering," *Nature-inspired optimizers: theories, literature reviews and applications,* pp. 123-141, 2020.
[55] S. I. Boushaki, N. Kamel, and O. Bendjeghaba, "A new quantum chaotic cuckoo search algorithm for data clustering," *Expert Systems with Applications,* vol. 96, pp. 358-372, 2018.
[56] F. S. Gharehchopogh and H. Shayanfar, "Automatic Data Clustering Using Farmland Fertility Metaheuristic Algorithm," in *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems*: Springer, 2022, pp. 199-215.
[57] M. Behera *et al.*, "Automatic data clustering by hybrid enhanced firefly and particle swarm optimization algorithms," *Mathematics,* vol. 10, no. 19, p. 3532, 2022.
[58] M. W. Ouertani, G. Manita, and O. Korbaa, "Automatic Data Clustering Using Hybrid Chaos Game Optimization with Particle Swarm Optimization Algorithm," *Procedia Computer Science,* vol. 207, pp. 2677-2687, 2022.
[59] M. B. Agbaje, A. E. Ezugwu, and R. Els, "Automatic data clustering using hybrid firefly particle swarm optimization algorithm," *IEEE Access,* vol. 7, pp. 184963-184984, 2019.
[60] R.-J. Kuo and F. E. Zulvia, "Automatic clustering using an improved artificial bee colony optimization for customer segmentation," *Knowledge and Information Systems,* vol. 57, no. 2, pp. 331-357, 2018.
[61] T. M. Pacheco, L. B. Gonçalves, V. Ströele, and S. S. R. Soares, "An ant colony optimization for automatic data clustering problem," in *2018 IEEE Congress on evolutionary computation (CEC)*, 2018: IEEE, pp. 1-8.
[62] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans,* vol. 38, no. 1, pp. 218-237, 2007.
[63] M. Sharma and J. K. Chhabra, "Sustainable automatic data clustering using hybrid PSO algorithm with mutation," *Sustainable Computing: Informatics and Systems,* vol. 23, pp. 144-157, 2019.

[64] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, and S. Mirjalili, "Clustering analysis using a novel locality-informed grey wolf-inspired clustering approach," *Knowledge and Information Systems,* vol. 62, pp. 507-539, 2020.
[65] S. F. Hussain, A. Pervez, and M. Hussain, "Co-clustering optimization using Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing,* vol. 97, p. 106725, 2020.
[66] B. Anari, J. A. Torkestani, and A. M. Rahmani, "Automatic data clustering using continuous action-set learning automata and its application in segmentation of images," *Applied Soft Computing,* vol. 51, pp. 253-265, 2017.
[67] M. Hasanzadeh-Mofrad and A. Rezvanian, "Learning automata clustering," *Journal of computational science,* vol. 24, pp. 379-388, 2018.
[68] P. Fränti and J. Kivijärvi, "Randomised local search algorithm for the clustering problem," *Pattern Analysis & Applications,* vol. 3, no. 4, pp. 358-369, 2000.
[69] I. Karkkainen and P. Franti, "Dynamic local search for clustering with unknown number of clusters," in *2002 International conference on pattern recognition*, 2002, vol. 2: IEEE, pp. 240-243.
[70] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied intelligence,* vol. 48, no. 12, pp. 4743-4759, 2018.
[71] M. Rezaei and P. Fränti, "Set matching measures for external cluster validity," *IEEE transactions on knowledge and data engineering,* vol. 28, no. 8, pp. 2173-2186, 2016.
[72] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of intelligent information systems,* vol. 17, pp. 107-145, 2001.
[73] K. S. Narendra and M. A. Thathachar, *Learning automata: an introduction*. Courier Corporation, 2012.
[74] M. Thathachar and P. S. Sastry, "A hierarchical system of learning automata that can learn die globally optimal path," *Information sciences,* vol. 42, no. 2, pp. 143-166, 1987.
[75] K. S. Narendra and M. A. Thathachar, "Learning automata-a survey," *IEEE Transactions on systems, man, and cybernetics,* no. 4, pp. 323-334, 1974.
[76] K. S. Narendra and M. A. Thathachar, "On the behavior of a learning automaton in a changing environment with application to telephone traffic routing," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 10, no. 5, pp. 262-269, 1980.
[77] F. Jazayeri, A. Shahidinejad, and M. Ghobaei-Arani, "Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach," *Journal of Ambient Intelligence and Humanized Computing,* vol. 12, pp. 8265-8284, 2021.
[78] Z. Anari, A. Hatamlou, and B. Anari, "Automatic finding trapezoidal membership functions in mining fuzzy association rules based on learning automata," 2022.
[79] B. Zarei and M. R. Meybodi, "Improving learning ability of learning automata using chaos theory," *The Journal of Supercomputing,* vol. 77, no. 1, pp. 652-678, 2021.
[80] M. Khaksar Manshad, M. R. Meybodi, and A. Salajegheh, "A new irregular cellular learning automata-based evolutionary computation for time series link prediction in social networks," *Applied Intelligence,* vol. 51, pp. 71-84, 2021.
[81] A. B. Hashemi and M. R. Meybodi, "A note on the learning automata based algorithms for adaptive parameter selection in PSO," *Applied Soft Computing,* vol. 11, no. 1, pp. 689-705, 2011.
[82] A. Yazidi, D. Silvestre, and B. J. Oommen, "Solving two-person zero-sum stochastic games with incomplete information using learning automata with artificial barriers," *IEEE Transactions on Neural Networks and Learning Systems,* 2021.
[83] B. Anari, J. Akbari Torkestani, and A. M. Rahmani, "A learning automata-based clustering algorithm using ant swarm intelligence," *Expert systems,* vol. 35, no. 6, p. e12310, 2018.
[84] A. Yazidi, I. Hassan, H. L. Hammer, and B. J. Oommen, "Achieving fair load balancing by invoking a learning automata-based two-time-scale separation paradigm," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 32, no. 8, pp. 3444-3457, 2020.

[85] Z. Aliniya and S. A. Mirroshandel, "A novel combinatorial merge-split approach for automatic clustering using imperialist competitive algorithm," *Expert Systems with Applications,* vol. 117, pp. 243-266, 2019.
[86] S. Harifi, M. Khalilian, and J. Mohammadzadeh, "Swarm based automatic clustering using nature inspired Emperor Penguins Colony algorithm," *Evolving Systems,* vol. 14, no. 6, pp. 1083-1099, 2023.
[87] Y. Zhou, H. Wu, Q. Luo, and M. Abdel-Baset, "Automatic data clustering using nature-inspired symbiotic organism search algorithm," *Knowledge-Based Systems,* vol. 163, pp. 546-557, 2019.
[88] D. M. Bot, J. Peeters, J. Liesenborgs, and J. Aerts, "FLASC: A Flare-Sensitive Clustering Algorithm: Extending HDBSCAN* for Detecting Branches in Clusters," *arXiv preprint arXiv:2311.15887,* 2023.
[89] R. Liu, B. Zhu, R. Bian, Y. Ma, and L. Jiao, "Dynamic local search based immune automatic clustering algorithm and its applications," *Applied Soft Computing,* vol. 27, pp. 250-268, 2015.
[90] A. Asuncion and D. Newman, "UCI machine learning repository," ed: Irvine, CA, USA, 2007.
[91] J. Derrac, S. Garcia, L. Sanchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Mult. Valued Logic Soft Comput,* vol. 17, pp. 255-287, 2015.