DOI: 10.21917/ijsc.2026.0584

CHAOTIC BUTTERFLY OPTIMIZATION ALGORITHM FOR AUTOMATIC CLUSTERING OF MULTIDIMENSIONAL DATA

R. Jensi

Department of Computer Science and Engineering, V V College of Engineering, India

Abstract

Automatic data clustering plays a vital role in unsupervised learning, where the number of clusters and their structure are often unknown beforehand. This paper presents a novel and computationally efficient clustering algorithm based on a Chaotic Butterfly Optimization Algorithm (CBOA). The proposed approach enhances the original Butterfly Optimization Algorithm by incorporating a chaotic perturbation mechanism and adaptive movement strategies to improve convergence speed and solution diversity. Unlike traditional clustering methods, CBOA does not require prior knowledge of the number of clusters. It automatically determines the optimal cluster count using a combined fitness function based on internal cluster validity indices, including the Silhouette index, Davies-Bouldin index, and CS index. Experimental results on UCI datasets and Shape datasets demonstrate the superiority of the proposed method in terms of clustering accuracy, compactness, and separation. Comparative analysis further confirms that CBOA outperforms classical clustering algorithms and recent metaheuristic-based approaches, while significantly reducing computational time. This makes it a promising tool for large-scale and complex clustering tasks.

Keywords:

Automatic Clustering, Swarm Intelligence, Cluster Validity Indices, Chaotic Map, Cluster Structure Detection

1. INTRODUCTION

In the last decade, many nature-inspired evolutionary algorithms have been developed for solving most engineering design optimization problems which are highly nonlinear, involving many design variables and complex constraints. These metaheuristic algorithms attracted very much because of the global search capability and take less time to solve real world problems. Nature-inspired algorithms [1]-[2] imitate the behaviours of living things in nature, so they are also called Swarm Intelligence (SI) algorithms.

Evolutionary algorithms (EAs) were the initial stage of such optimization methods. Genetic Algorithm (GA) [3] and simulated annealing (SA) [4] are popular examples for EAs. The genetic algorithm was developed by John Holland in the early 1970s, inspired by biological evolution such as reproduction, mutation, recombination and selection. Simulated annealing (SA) was inspired by annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects.

Swarm Intelligence systems maintain a population of solutions which have evolved through random selection and alterations. The way they differ depends on the generation of new solutions, random selection procedure and candidate solution encoding technique. Particle Swarm Optimization (PSO) [5] developed by Kennedy and Eberhart in 1995, simulates the social behaviour of bird flock or fish school. There are several variants of the PSO algorithm developed in literature. Shi and Eberhart [6] introduced a new parameter into PSO and proved a significant impact on PSO performance. Kennedy et al. [7] investigates various population topologies. The performance of PSO algorithm was improved in [8] by introducing a new term into the particles' velocity updation and the authors proved that the proposed algorithm outperformed but still premature convergence and stuck in local optima. Mendes et al. [9] proposed a fully informed particle swarm to achieve better solutions. Ratnaweera et. al.[10] introduced time-varying initialization velocity as a robust and consistent optimization strategy with time-varying inertia weight factor in PSO and the experimental results showed faster convergence at the early stages in obtaining global optimum solutions. Parsopoulos and Vrahatis [11] introduced a new scheme in updating velocity and position of the particle for exploration and exploitation and the results proved that the proposed approach is superior compared to standard PSO algorithm. Suganthan et al. [12] proposed a new learning technique in which all other particles past best information was used to modify a particle's velocity and the experimental results showed that the proposed work achieved good performance in optimizing the problems. Tsoulos and Stavrakoudis [13] proposed a new approach which includes three modifications such as new stopping rule, similarity check and conditional rule for local search and the results showed significant improvement in speed and efficiency. Xinchao [14] proposed a new method for updating particles' position and velocity called a perturbed particle swarm algorithm. Zhan et al. [15] proposed an orthogonal learning strategy so that particles move in better directions and the results showed that it achieved faster convergence when compared to standard PSO algorithm. Wang et al. [16] introduced a self-adaptive strategy which is based on a probability model being used to update a particle. In [17] and [18], levy flight methods are combined with PSO algorithms to improve the performance of PSO algorithm. [19] - [25], PSO algorithms are hybridized with other optimization techniques in order to achieve faster convergence and to obtain solution accuracy.

The remainder of this paper is organized as follows. Section 2 briefly explains related work in clustering, the standard BOA algorithm is given in section 3, Section 4 outlines the proposed work, experimental results and its discussions are given in Section 5. Section 6 concludes the paper with fewer discussions.

2. RELATED WORK

Data clustering [31] has been a fundamental task in unsupervised learning, where the objective is to partition a dataset into distinct groups such that intra-cluster similarity is maximized while inter-cluster similarity is minimized. Traditional methods like k-means often suffer from local optima and sensitivity to initial conditions [34]. To address these limitations, a wide array of nature-inspired metaheuristic algorithms has been proposed.

One of the earliest applications of swarm intelligence in clustering was presented by Van and Engelbrecht [33], who employed Particle Swarm Optimization (PSO) to search for optimal cluster centers. Subsequently, several researchers extended the use of metaheuristics such as Simulated Annealing [35], Ant Colony Optimization (ACO) [36], and Tabu Search [37] to enhance clustering quality. Hybrid approaches soon gained popularity, combining the strengths of multiple algorithms. Kao et al. [38] proposed a hybridized approach using PSO and Nelder-Mead, demonstrating improved convergence and cluster compactness. Similarly, artificial bee colony (ABC)-based clustering has been extensively studied, with notable contributions by Zhang et al. [39], Karaboga and Ozturk [41], and Yan et al. [42], showing its suitability for high-dimensional datasets.

The Firefly Algorithm (FA) [40], Cuckoo Search (CS) with Lévy flights [44], and the Flower Pollination Algorithm (FPA) [45] have also shown promise for clustering applications. Senthilnath et al. [40] and [44] explored swarm intelligence techniques like fireflies and cuckoos for enhanced clustering results, especially in complex multimodal search spaces.

Recent years have seen the emergence of hybrid metaheuristics with improved global and local search capabilities. Notably, Jensi and Jiji [49] introduced an enhanced Krill Herd (KH) algorithm with improved global exploration for numerical optimization and clustering tasks. Bouyer and Hatamlou [50] proposed a hybrid of cuckoo optimization and modified PSO, achieving better performance in high-dimensional clustering problems.

Moreover, novel methodologies such as gravity-based clustering [51], bat algorithm-based approaches [46], and improved Cat Swarm Optimization (CSO) [48] highlight the growing interest in hybrid and adaptive metaheuristics.

Recent advancements have focused on more powerful hybrid and multi-objective algorithms. Behera et al. [53] proposed a hybrid enhanced Firefly-PSO algorithm that automatically determines the number of clusters. Similarly, Agbaje et al. [54] designed a robust FA-PSO hybrid clustering model and evaluated its performance on large-scale datasets. Zhang et al. [55] introduced a chaotic hybrid Butterfly Optimization Algorithm with PSO to address high-dimensional clustering problems.

Multi-objective and task-sharing frameworks have also gained attention. Yan et al. [56] proposed a single- and multi-objective Cat Swarm Optimization (CSO) clustering approach that effectively balances between conflicting objectives such as intra-cluster compactness and inter-cluster separation. Wang et al. [57] developed a multi-objective automatic clustering algorithm using Evolutionary Multi-Tasking Optimization (EMTO), enabling simultaneous optimization across several tasks. Furthermore, Merikhi and Soleymani [58] proposed a binary nature-inspired framework for clustering, capable of representing flexible data partitioning schemes in various applications.

These contributions highlight the growing trend towards hybrid, adaptive, and multi-objective metaheuristics for automatic clustering. Motivated by these advancements, the proposed method using a chaotic butterfly optimization algorithm combines global exploration for automatic clustering (i.e) without the prior knowledge of number of clusters, it will find optimal number clusters. The aim is to achieve superior clustering performance by addressing both solution quality and computational efficiency in an integrated framework.

3. BUTTERFLY OPTIMIZATION ALGORITHM (BOA)

The Butterfly Optimization Algorithm (BOA) is a recent nature-inspired metaheuristic proposed by Arora and Singh in 2018. It draws inspiration from the foraging behaviour of butterflies, which communicate and navigate using olfactory signals, or "fragrance." BOA has been applied successfully to a variety of real-world optimization problems due to its simplicity and the ability to balance exploration and exploitation effectively.

Each butterfly in the population represents a candidate solution in the search space. The optimization process mimics the way butterflies emit fragrances to attract others toward promising locations (global search) or explore locally based on interaction with nearby butterflies (local search). A switch probability parameter determines the transition between these two behaviors.

The fragrance intensity of a butterfly is defined as:

$$F_i = a.\ I_i^b \tag{1}$$

where, F_i is the fragrance of butterfly i, I_i is the perceived fitness (intensity) of the solution, a is the sensory modality and b is the power exponent.

As in [52], a is increased from 0.1 to 0.3 over the iterations, as shown in Eq.(2).

$$a = 0.1 + (0.3\text{-}0.1) \times (t/maxIter) \tag{2}$$

where, t is the current iteration. *maxIter* is the maximum number of iterations.

Butterflies engage in both foraging and mating behaviours across local and global search spaces; however, foraging typically dominates their activity. In the Butterfly Optimization Algorithm (BOA), the transition between global and local search is governed by a switch probability p, where $p \in [0, 1]$. A common choice in the literature is p=0.8 [52], indicating a strong preference for global search. In this phase, each butterfly moves toward the current best solution, as described by Eq.(3) and in the local search phase, the positions of individuals are updated through the solutions in the population, which can be expressed as Eq.(4).

$$X_i^{t+1} = X_i^t + F_i\left(r^2 X_g^* - X_i^t\right) \tag{3}$$

$$X_i^{t+1} = X_i^t + F_i\left(r^2 X_j^* - X_k^t\right) \tag{4}$$

where, X_g^* is the global best position found so far, X_i, X_j are two randomly chosen butterflies, r is a random number in [0,1], t is the current iteration. F_i is the fragrance of i^{th} butterfly

The pseudocode of the BOA algorithm is given in Fig.1.

Initialize parameters: population size (n), a, b, p, *max_iter*

Initialize positions of butterflies randomly

Evaluate fitness of each butterfly and compute fragrance

for t = 1 to *max_iter*

  for each butterfly i

    Generate a random number r in [0,1]

    if $r < p$ (global search)

  Move butterfly toward global best using fragrance by Eq.(3)

else (local search)
Select two butterflies randomly ($j \neq k$)
Move i toward the relative position of j and k by Eq.(4)
end
Apply boundary control (if needed)
Evaluate new position and update fragrance
end
Update global best if needed
end
Return the best solution found

4. PROPOSED ALGORITHM

The Chaotic Butterfly Optimization Algorithm (CBOA) is an enhanced variant of the standard Butterfly Optimization Algorithm (BOA), specifically designed to address the challenges of automatic data clustering. While the original BOA is effective in global optimization, it often suffers from slow convergence and limited solution diversity, especially in complex multimodal search spaces. CBOA introduces chaotic dynamics and an adaptive movement strategy to accelerate convergence, prevent premature stagnation, and improve clustering quality.

4.1 MOTIVATION

In unsupervised learning tasks like clustering, determining the optimal number of clusters and their boundaries without supervision is a major challenge. Traditional clustering algorithms assume a fixed cluster count, which limits its applicability in real-world datasets with unknown or varying structure. CBOA addresses this by:

1. Automatically detecting the optimal number of clusters during the search process.
2. Using a chaotic map to maintain population diversity and avoid local minima.
3. Incorporating internal cluster validation indices (Silhouette, Davies-Bouldin, CS index) into the objective function for quality assessment.

This integration of chaos theory and automatic cluster number estimation into a lightweight, fast-converging BOA framework marks a significant advancement over standard clustering approaches.

The proposed CBOA algorithm initializes a population of solutions, where each individual encodes a candidate set of cluster centers. Given the data set D and the random number of clusters k as input to the algorithm, the solution is represented as a row vector of size $k \times m$, where k is the number of clusters chosen ranomly and m is the number of features for the clustering problem and it is shown in Fig.2.

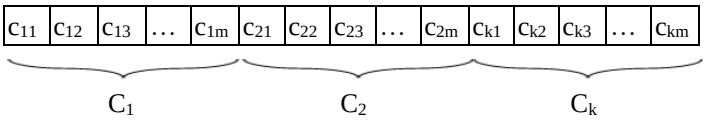


Fig.2. Representation of a candidate solution for k clusters and m features

The number of clusters per individual is varied randomly within a predefined range. At each iteration, the fitness of each butterfly is evaluated using a weighted combination of three validity indices: Silhouette Coefficient, Davies-Bouldin Index, and CS Index.

4.2 INITIALIZATION

Each butterfly represents a flattened matrix of cluster centers. The number of clusters is initialized randomly in the range [4, 20]. Positions are initialized by sampling randomly from the dataset space

4.3 FITNESS FUNCTION

The fitness function f for each butterfly is calculated as follows: If the dimension of the dataset is less than 2, then f is calculated by using Eq.(5), otherwise Eq.(6) is used. Since silhouette score requires at least two-dimensional data to produce meaningful cluster evaluations, the fitness function adapts based on input dimensionality. For $d \leq 2$, silhouette is excluded, and only DBI and CS indices are used (Eq.(5)). For $d > 2$, silhouette is included to improve clustering quality assessment (Eq.6).

$$f=\text{w\_2}\times DB+w_3\times CS \quad (5)$$

$$f=w_1(1-S)+w_2\times DB+w_3\times CS \quad (6)$$

where, S is the mean silhouette coefficient, DB is the Davis-Bouldin index, CS is the Cluster Separation index, and weights w_1=0.3, w_2=0.6 and w_3=0.4 are empirically tuned

4.4 CHAOTIC UPDATE MECHANISM

Instead of traditional attraction-based updates, chaotic scaling via the logistic map is introduced. To enhance exploration and prevent premature convergence, chaotic map based on the logistic function is employed as in Eq.(7).

$$x_{n+1}=rx_n(1-x_n) \quad (7)$$

where, r = 4 (gives fully chaotic behaviour), x_n is the value at iteration n (starting from some x_0 in (0, 1)). In this study x_0=0.7. It's sensitive to initial values and quickly generates a pseudo-random sequence that's deterministic but non-repetitive.

At each iteration, a chaotic coefficient is generated and used to modulate the position updates of butterflies. This mechanism injects deterministic chaos into the search process, improving diversity without sacrificing convergence.

This chaos coefficient modulates attraction to the global best, preventing premature convergence and increasing exploration capability. Each butterfly's position is updated using a hybrid mechanism that incorporates stochastic perturbation and guided attraction. The movement rule is given in Eq.(8).

$$X_i^{new}=X_i+f.N(0,1)+c.(X_{best}-X_i) \quad (8)$$

where, X_i - Current position of the i^{th} bufferfly (solution). X_i^{new} - Updated position of the i^{th} butterfly. f - Random stp-size scaling factor, drawn from range $[a,b]$. $N(0,1)$ - A vector of random values drawn from normal distribution. c - Chaotic coefficient, dynamically generated via a logistic map. X_{best} - Current best solution.

The position update allows simultaneous exploration and exploitation. The parameter f controls random search intensity,

while the chaotic coefficient c introduces dynamic nonlinearity, enhancing diversity and preventing premature convergence.

4.5 MEMORY AND ELITE RETENTION

A memory matrix stores elite individuals. At each iteration, the current population is compared against memory and updated based on fitness ranking. The best individual is retained as a potential global optimum. The pseudocode of the CBOA algorithm is given in Fig.3.

```
Initialize parameters: population size (n), a, b, p, max_iter
Initialize positions of butterflies randomly
Evaluate fitness of each butterfly and compute fragrance
for t = 1 to max_iter
  for each butterfly i
    Generate a random number r in [0,1]
    Evaluate fitness as in Section 4.2
Store best memory of butterflies
  end
  for each butterfly i
Generate chaotic coefficient c using logistic map by Eq.(7)
Move butterfly toward global best using fragrance by Eq.(8)
  end
    Reshape bestPosition to get best cluster centers
Assign clusters using best centers
end
  Return the best solution found
```

Fig.3. Pseudocode of the CBOA algorithm

5. EXPERIMENTAL RESULTS AND DISCUSSION

The performance of the proposed Chaotic Butterfly Optimization Algorithm (CBOA) is evaluated by comparing it with the standard Butterfly Optimization Algorithm (BOA). All experiments are conducted using MATLAB on a system equipped with an Intel Core i3-2350M processor (2.30 GHz), 4 GB RAM, and running the Windows 10 operating system.

Table.1. Parameter settings

Parameter	Values for	
	BOA	**CBOA**
Population Size (NP)	25	25
Maximum Iteration	50	50
p	0.8	-

The parameter settings used for the BOA and CBOA are detailed in Table.1. To ensure a fair comparison and account for stochastic variability, both algorithms are independently executed 20 times under identical parameter configurations.

To evaluate the performance of the proposed algorithm for data clustering, 14 datasets have been used. The datasets, namely, Iris, Wine, Glass, Wisconsin Breast Cancer (WBC), Thyroid and Vowel, are collected from Machine Learning Laboratory [30] and shape datasets are collected from [31]. The 6 UCI datasets are described in Table 2 and 8 Shape datasets are given in Table 3.

To assess the stability and effectiveness of the clustering results, the Davies-Bouldin (DB) Index and CS Index were computed for each run across 20 independent executions on benchmark UCI and Shape datasets. Lower values of both indices indicate better clustering quality—reflecting higher intra-cluster compactness and inter-cluster separation. The best, worst, average and standard deviation of DB and CS values for each algorithm for UCI datasets are summarized in Table.4. The best result values are shown in Bold.

Table.2. UCI dataset descriptions

Sl. No.	**Dataset Name**	**# of features**	**# of classes**	**# of instances (size of each class)**
1	Iris	4	3	150 (50,50,50)
2	Wine	13	3	178 (59,71,48)
3	Glass	9	6	214 (70,17,76,13,9,29)
4	WBC	9	2	683 (444,239)
5	Thyroid	5	3	215 (150,35,30)
6	Vowel	3	6	871((72, 89, 172, 151, 207, 180)

Table.3. Shape dataset descriptions

Sl. No.	**Dataset Name**	**# of features**	**# of classes**	**# of instances (size of each class)**
1	Aggregation	2	7	788 (45,170,102,273, 34, 130, 34)
2	Compound	2	6	399(50,92,38,45,158,16)
3	Path-based	2	3	300(110,97,93)
4	Spiral	2	3	312(101,105,106)
5	Flame	2	2	240(87,153)
6	Jain	2	2	373(276,97)
7	R15	2	15	600 (40 / class)
8	D31	2	31	3100 (100 / class)

An examination of the values in Table.4 reveals that CBOA significantly outperforms BOA in five out of the six evaluated datasets, demonstrating superior clustering quality in terms of both Davies-Bouldin Index and CS Index.

Table.4. DB Index and CS Index values for clustering over 20 independent runs for UCI datasets

Dataset		**DB Index**		**CS Index**	
		BOA	**CBOA**	**BOA**	**CBOA**
Iris	Best	0.6444	**0.3836**	0.3523	**0.1918**
	Worst	1.0706	**0.3836**	0.7478	**0.1918**
	Mean (std)	0.8689 (0.129)	**0.3835 (1.1390E-016)**	0.5579 (0.1147)	**0.19179 (5.695E-017)**
Wine	Best	**0.4048**	0.4496	0.2875	**0.2360**

	Worst	**0.5647**	0.6955	0.5451	**0.7463**
	Mean (std)	**0.5155 (0.042)**	0.60348 (0.0664)	0.3832 (0.0769)	**0.5259 (0.1293)**
Glass	Best	0.8134	**0.2953**	0.5444	**0.1476**
	Worst	1.4748	**0.3145**	1.6960	**0.1573**
	Mean (std)	1.2010 (0.173)	**0.2962 (0.0043)**	1.0333 (0.3481)	**0.1481 (0.0021)**
WBC	Best	1.1865	**0.7576**	0.8793	**0.3788**
	Worst	2.0880	**1.6156**	1.8265	**0.8663**
	Mean (std)	1.7132 (0.229)	**1.1554 (0.3002)**	1.259 (0.2657)	**0.5796 (0.1504)**
Thyroid	Best	0.7789	**0.7460**	0.3812	**0.3561**
	Worst	1.1089	**0.9006**	1.1152	**0.7472**
	Mean (std)	0.9413 (0.085)	**0.8297 (0.0482)**	0.7738 (0.205)	**0.6379 (0.0881)**
Vowel	Best	**0.7632**	0.8438	0.4486	**0.4453**
	Worst	**1.0503**	1.1714	**0.7325**	0.7971
	Mean (std)	**0.9076 (0.079)**	1.0257 (0.0788)	**0.55806 (0.0679)**	0.62902 (0.0886)

- **Iris**: CBOA shows remarkable consistency, with both best and worst DB Index at 0.3836, and near-zero standard deviation. CS Index is also significantly lower than CBOA (0.1918) compared to BOA (0.5579 ± 0.1147). It indicates perfectly stable convergence and superior clustering by CBOA on low-dimensional data.
- **Wine**: Although BOA slightly outperforms CBOA in terms of mean DBI and CS, the performance of CBOA remains competitive. The result suggests that for moderately complex datasets, BOA may still find good solutions, but CBOA shows higher variance and potential room for tuning.
- **Glass**: CBOA achieves a dramatic improvement with DB Index reduced from 1.2010 ± 0.173 (BOA) to 0.2962 ± 0.0043. CS Index similarly drops from 1.0333 to 0.1481, showing that CBOA is highly effective for datasets with overlapping or unevenly distributed classes.
- **WBC**: CBOA improves both DBI and CS significantly over BOA. Variance is slightly higher in DBI for CBOA (±0.3002), but still within acceptable limits, suggesting robust but adaptable convergence behaviour.
- **Thyroid**: CBOA shows consistent gains over BOA, with improvements in both metrics. Lower standard deviations indicate greater reliability in results.
- **Vowel**: BOA outperforms CBOA for both DB and CS indices on this high-dimensional, multi-class dataset. CBOA shows slightly higher DBI (1.0257) and CS Index (0.629).

To further validate the effectiveness of the proposed CBOA, a series of experiments were conducted on eight well-known synthetic shape datasets: Aggregation, Compound, Path-based, Spiral, Flame, Jain, R15, and D31. These datasets are particularly challenging due to their non-convex cluster structures, varying densities, and irregular geometries. The evaluation was based on the Davies-Bouldin Index (DBI) and CS Index, two widely used internal clustering validity metrics that quantify intra-cluster compactness and inter-cluster separation and its values are given in Table 5. Lower values of these indices indicate better clustering quality. The results, averaged over 20 independent runs, reveal that CBOA consistently outperforms BOA across all datasets.

Table.5. DB Index and CS Index values for clustering over 20 independent runs for Shape datasets

Dataset		**DB Index**		**CS Index**	
		BOA	**CBOA**	**BOA**	**CBOA**
Aggregation	Best	0.5910	**0.5021**	0.3485	**0.2775**
	Worst	0.8119	**0.7889**	0.7629	**0.5095**
	Mean (std)	0.7133 (0.0602)	**0.66885 (0.1015)**	0.4567 (0.1022)	**0.3996 (0.069)**
Compound	Best	0.5611	**0.5191**	0.3827	**0.2784**
	Worst	0.9050	**0.8334**	0.6302	**0.5421**
	Mean (std)	0.7882 (0.0869)	**0.71488(0.1005)**	0.50502 (0.0706)	**0.44870 (0.0744)**
Path-based	Best	0.6426	**0.5497**	0.3619	**0.3021**
	Worst	0.8867	**0.7388**	0.6311	**0.5132**
	Mean (std)	0.7863 (0.0621)	**0.65986(0.0609)**	0.4855 (0.0587)	**0.3805 (0.0557)**
Spiral	Best	0.8081	**0.7109**	0.4393	**0.2923**
	Worst	0.9338	**0.7980**	0.6616	**0.5384**
	Mean (std)	0.8819 (0.0249)	**0.75366(0.0255)**	0.50621 (0.0494)	**0.41905 (0.0659)**
Flame	Best	0.7107	**0.6048**	0.3377	**0.2670**
	Worst	0.8998	**0.7982**	0.5388	**0.5308**
	Mean (std)	0.79089 (0.0569)	**0.72591(0.0667)**	0.43427 (0.0659)	**0.39032 (0.07352)**
Jain	Best	0.6264	**0.6079**	0.3430	**0.3206**
	Worst	0.7465	**0.7398**	0.5185	**0.5169**
	Mean (std)	0.68511 (0.0356)	**0.67110(0.0373)**	0.42669 (0.0478)	**0.40727 (0.0547)**
R15	Best	0.6170	**0.3487**	0.2423	**0.1019**
	Worst	0.8517	**0.5093**	0.4576	**0.2924**
	Mean (std)	0.73882 (0.0639)	**0.42829(0.0563)**	0.34991 (0.05969)	**0.18587 (0.0662)**
D31	Best	0.7220	**0.6269**	0.3912	**0.3145**
	Worst	0.9067	**0.7502**	0.5700	**0.4774**
	Mean (std)	0.81212 (0.0445)	**0.69176(0.0312)**	0.4625 (0.05104)	**0.39738 (0.04967)**

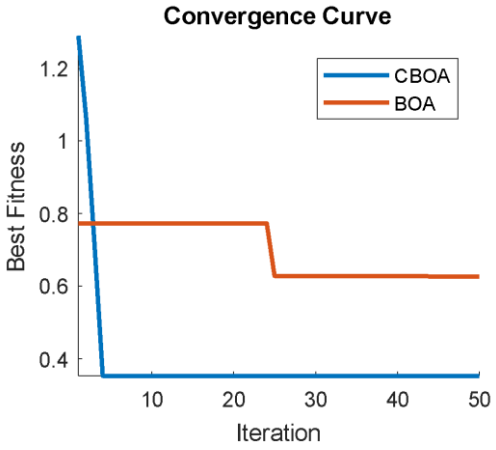
On the Aggregation dataset, CBOA achieved a lower mean DBI of 0.6689 compared to BOA's 0.7133, along with a reduced CS Index of 0.3996 against 0.4567, indicating more compact and well-separated clusters. Similar improvements were observed for the Compound dataset, where CBOA reduced the DBI from 0.7882 to 0.7149, and the CS Index from 0.5050 to 0.4487. These enhancements are especially significant in the Path-based and Spiral datasets, known for their curved and elongated clusters. For the Path-based dataset, CBOA decreased the DBI from 0.7863 to 0.6599, and CS from 0.4855 to 0.3805. In the Spiral dataset, CBOA's DBI dropped to 0.7537, and CS Index improved to

0.4191, both considerably better than the values achieved by BOA.

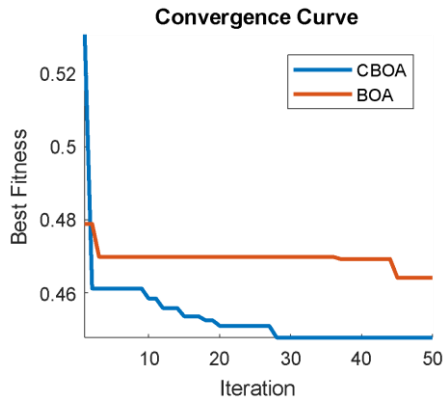
The Flame and Jain datasets further illustrate CBOA's superiority. On Flame, DBI reduced from 0.7909 to 0.7259, and CS from 0.4343 to 0.3903. On Jain, which has compact clusters with minor overlap, CBOA achieved consistent improvements in both metrics. Particularly notable is the performance on R15, a dataset with 15 well-separated clusters. CBOA significantly outperformed BOA, reducing the mean DBI from 0.7388 to 0.4283, and CS Index from 0.3499 to 0.1859, nearly halving the CS value. Finally, for the high-cluster-count D31 dataset, CBOA demonstrated strong performance with a DBI of 0.6918 compared to BOA's 0.8121, and a CS Index of 0.3974 over BOA's 0.4625.

In addition to achieving lower mean values, CBOA also demonstrated reduced standard deviations across most datasets, suggesting more stable and reliable performance over multiple runs. These results collectively validate that the chaotic update mechanism in CBOA improves exploration, avoids premature convergence, and effectively handles datasets with complex cluster topologies—making it a robust and superior alternative to traditional BOA for shape-based clustering tasks.

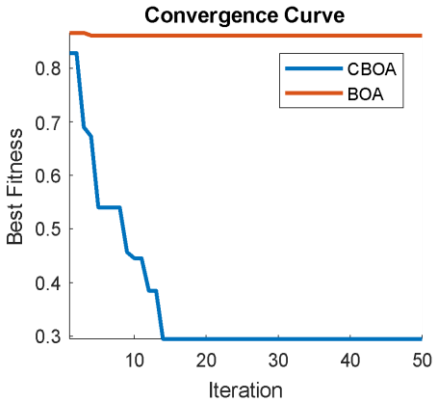
The convergence behavior of the proposed Chaotic Butterfly Optimization Algorithm (CBOA) was analyzed and compared with the standard Butterfly Optimization Algorithm (BOA) using the objective fitness values obtained across iterations for each dataset. The convergence plots for UCI datasets are shown in Fig.4, while those for the synthetic shape datasets are presented in Fig.5.



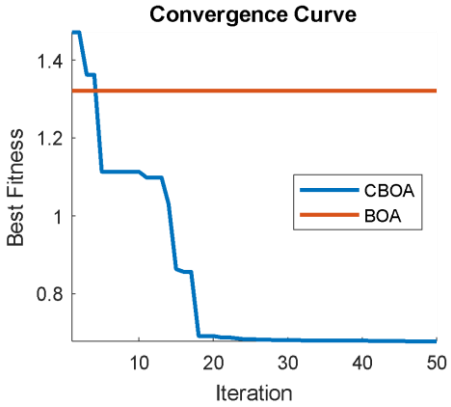
(a) Iris



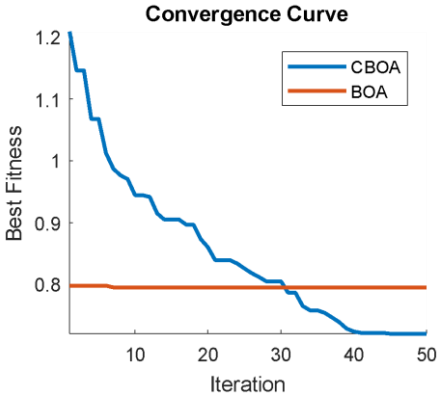
(b) Wine



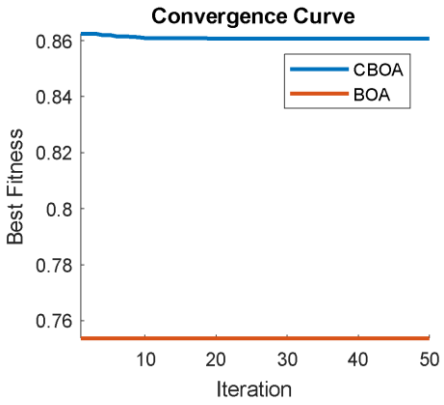
(c) Glass



(d) WBC

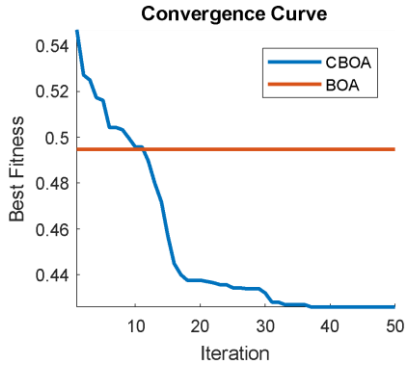


(e) Thyroid

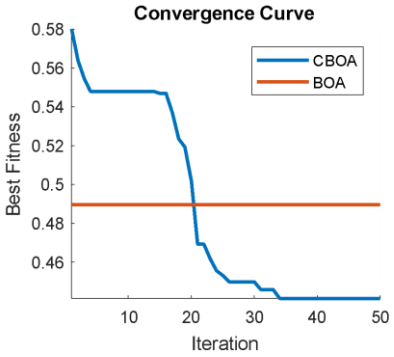


(f) Vowel

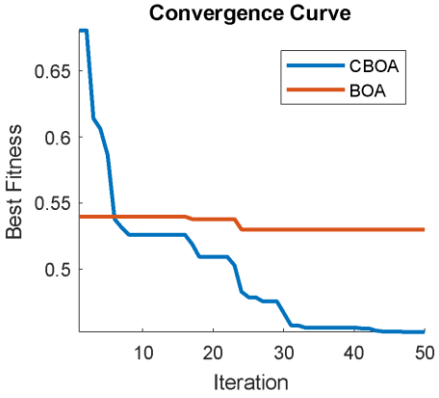
Fig.4. Convergence graph of UCI datasets



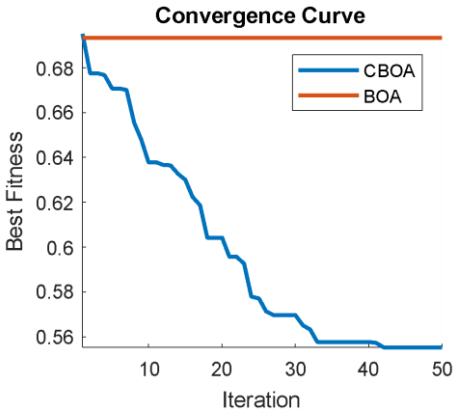
(a) Aggregation



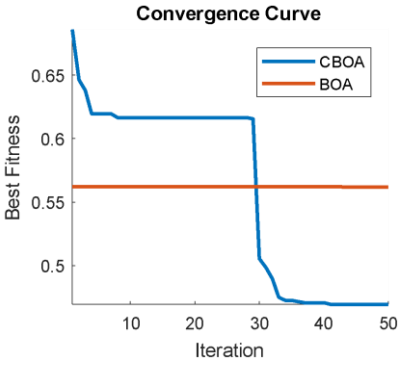
(b) Compound



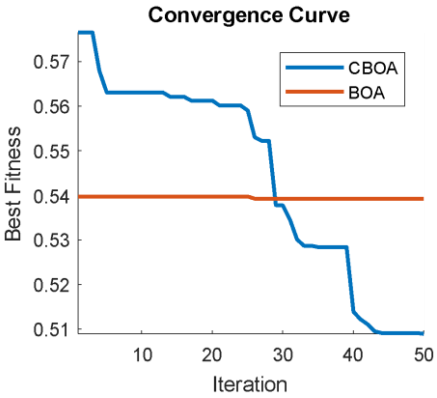
(c) Path-based



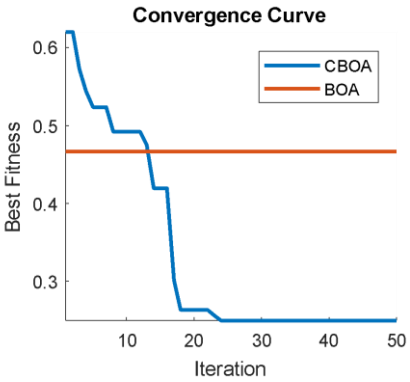
(d) Spiral



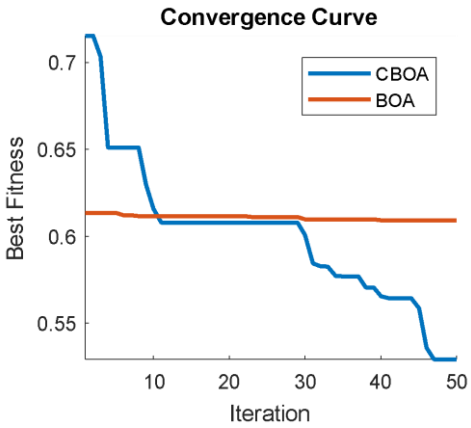
(e) Flame



(f) Jain



(g) R15



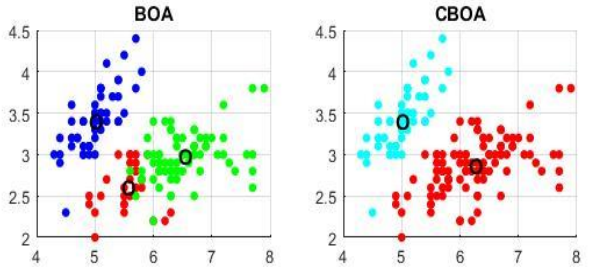
(h) D31

Fig.5. Convergence graph of Shape datasets

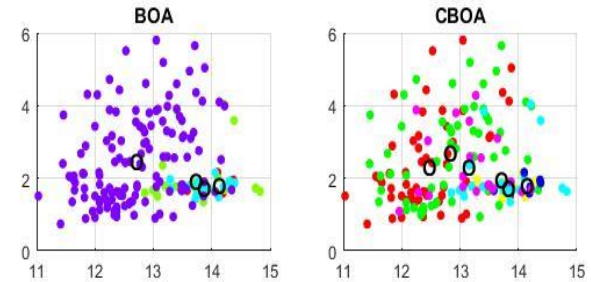
From Fig.4, it is evident that CBOA achieves faster and more stable convergence across most UCI datasets. On datasets such as Iris, Glass, and WBC, CBOA not only converges in fewer iterations but also consistently reaches a lower fitness value compared to BOA, indicating improved clustering quality. Notably, in the Iris dataset (Fig.4a), CBOA reaches its optimal fitness within the first 10-15 iterations, whereas BOA converges more slowly and stabilizes at a higher fitness value. Similar trends are observed for Glass (Fig.4c) and WBC (Fig.4d), where CBOA demonstrates a smoother and steeper descent, reflecting its ability to quickly explore and exploit the solution space.

In the case of shape datasets (Fig.5), which are inherently more complex due to irregular cluster shapes and distributions, CBOA continues to demonstrate superior convergence characteristics. For example, in R15 (Fig.5g) and D31 (Fig.5h), CBOA converges significantly faster than BOA and achieves markedly lower final fitness values. This highlights CBOA's capacity to handle multi-modal and high-cluster-count data more effectively. Similarly, in Compound (Fig.5b) and Spiral (Fig.5d) datasets, CBOA's convergence curves exhibit minimal fluctuations, indicating a more reliable and robust search behaviour.

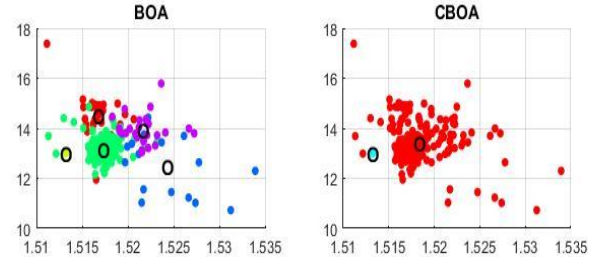
Overall, the convergence graphs confirm that the integration of the chaotic logistic map into the butterfly update mechanism enhances the algorithm's global search ability, helping it avoid premature convergence and local optima. The chaotic perturbation, combined with memory retention and dynamic step adjustment, allows CBOA to converge more quickly and stably, making it a powerful solution for unsupervised data clustering.



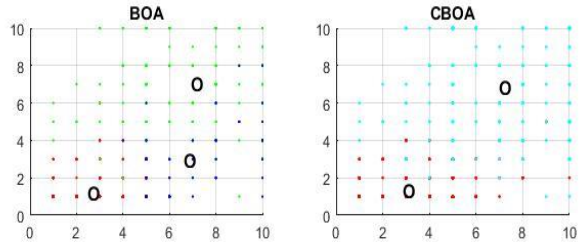
(a) Iris dataset



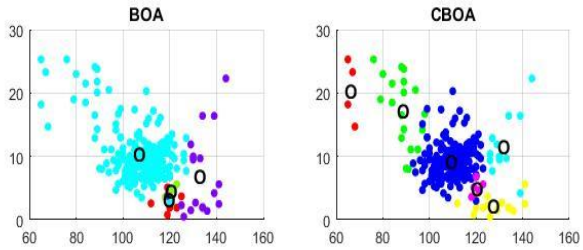
(b) Wine dataset



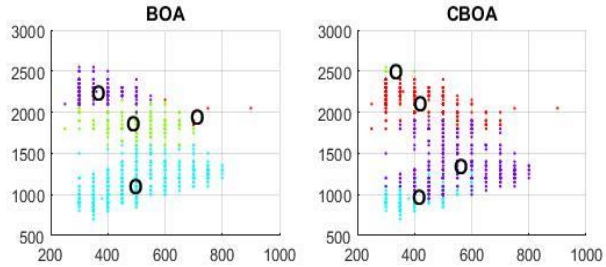
(c) Glass dataset



(d) WDBC dataset

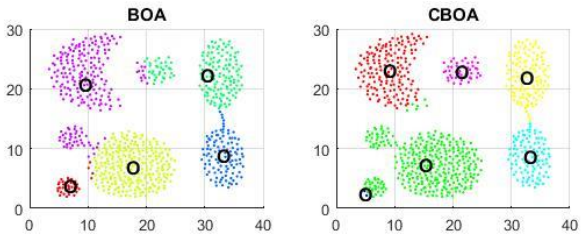


(e) Thyroid dataset

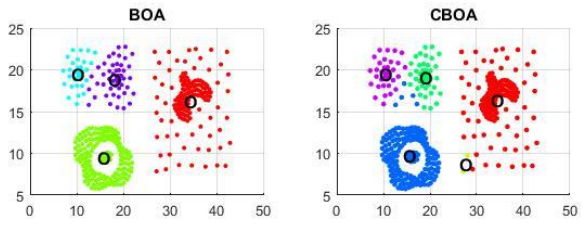


(f) Vowel dataset

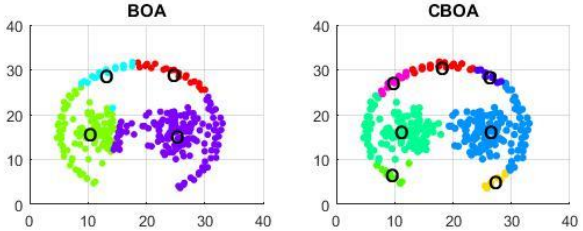
Fig.6. Clustering Illustrations for BOA and CBOA for UCI datasets



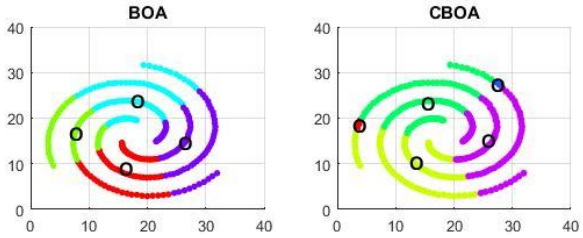
(a) Aggregation dataset



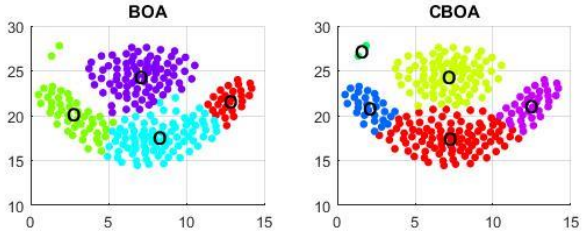
(b) Compound dataset



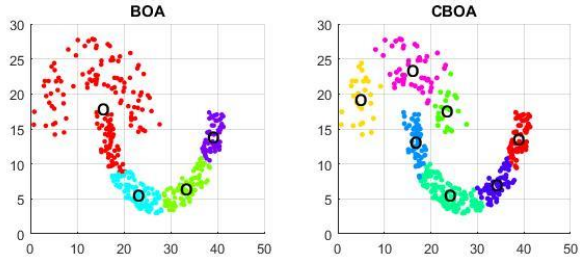
(c) Path-based dataset



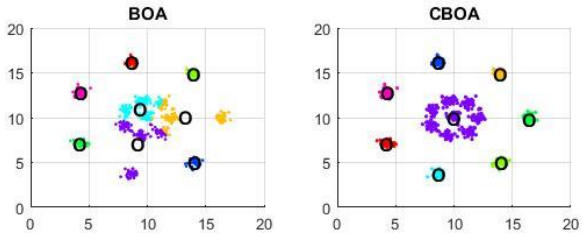
(d) Spiral dataset



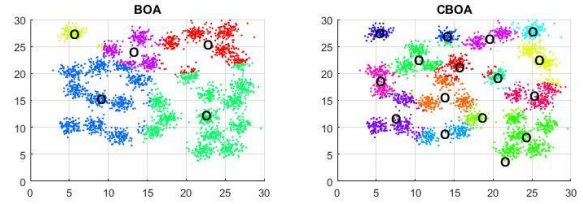
(e) Flame dataset



(f) Jain dataset



(g) R15 dataset



(h) D31 dataset

Fig.7. Clustering Illustrations for BOA and CBOA for Shape datasets

The visual comparison of clustering outputs generated by the standard Butterfly Optimization Algorithm (BOA) and the proposed Chaotic Butterfly Optimization Algorithm (CBOA) is depicted in Fig.6 (for UCI datasets) and Fig.7 (for synthetic shape datasets). These illustrations provide a qualitative assessment of how effectively each algorithm partitions the dataset into coherent clusters.

As shown in Fig.6, for the Iris, Glass, WBC, and Thyroid datasets, CBOA consistently forms clusters that are more compact, well-separated, and aligned with the true data structure compared to BOA. For example, in the Iris dataset, the CBOA-based clusters exhibit minimal overlap, capturing the underlying class boundaries more accurately, while BOA tends to produce elongated or overlapping clusters. A similar improvement is visible in the Glass dataset, where CBOA generates finer, tighter groupings that better reflect the internal variance of the data. On high-dimensional datasets like Vowel, although both algorithms show some class overlap, CBOA's clusters appear more cohesive and distinguishable, indicating better handling of complex feature interactions.

In Fig.7, the difference is even more pronounced across the shape datasets, which are known for their non-convex and intricate cluster structures. CBOA outperforms BOA in nearly every instance by effectively capturing the geometry and density of clusters. For instance, in the Path-based and Spiral datasets, BOA fails to maintain the continuity of curved clusters, often splitting them inappropriately or merging disjointed points. In contrast, CBOA preserves the natural curvature and arrangement of clusters, showcasing its robustness in modelling nonlinear boundaries. The improvement is especially significant in R15 and D31, where CBOA distinctly identifies all clusters with minimal boundary error, whereas BOA exhibits cluster merging and misclassification.

Overall, these visual results reinforce the quantitative findings from the clustering validity indices. The enhanced performance of CBOA can be attributed to its chaotic dynamics, which promote diverse exploration and precise convergence, enabling it to effectively navigate complex clustering landscapes. The illustrations clearly demonstrate that CBOA not only improves numerical metrics but also delivers visually meaningful and interpretable clustering results, critical for practical unsupervised learning tasks.

6. CONCLUSION

In this study, a Chaotic Butterfly Optimization Algorithm (CBOA) was proposed for automatic data clustering, integrating chaotic dynamics into the standard BOA framework to enhance exploration and prevent premature convergence. The proposed method dynamically determines the optimal number of clusters while simultaneously optimizing cluster compactness and separation using a composite fitness function based on internal validity indices (Davies-Bouldin Index, CS Index, and Silhouette Score).

Experimental results on a variety of benchmark datasets, including UCI and synthetic shape datasets, demonstrate that CBOA consistently outperforms the standard BOA in terms of clustering accuracy, convergence speed, and solution stability. CBOA achieved lower average values of DB and CS indices across most datasets, with significantly improved performance on complex datasets with non-convex or irregular cluster structures. The convergence analysis further confirmed that CBOA reaches optimal solutions faster and more reliably, while the visual clustering results reinforced its capability to identify well-separated and cohesive clusters.

The proposed CBOA framework offers a robust, adaptive, and efficient solution for unsupervised data clustering. Future work may explore hybridizing CBOA with deep learning techniques or extending it to dynamic data clustering and real-time applications.

REFERENCES

[1] R.C. Eberhart, Y. Shi and J. Kennedy, "*Swarm Intelligence*", 2001.

[2] X.S. Yang, "*Nature-Inspired Metaheuristic Algorithms*", 2010.

[3] D.E. Goldberg, "*Genetic Algorithms in Search, Optimization and Machine Learning*", 1998.

[4] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", *SCIENCE*, Vol. 220, pp. 671-680, 1983.

[5] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization", *Proceedings of International Conference on Neural Networks*, pp. 1942-1948, 1995.

[6] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer", *Proceedings of International Conference on Evolutionary Computation*, Vol. 8, pp. 69-73, 1998.

[7] J. Kennedy and R. Mendes, "Population Structure and Particle Swarm Performance", *Proceedings of International Congress on Evolutionary Computation*, Vol. 3, pp. 1671-1676, 2002.

[8] T. Peram, K. Veeramachaneni and C.K. Mohan, "Fitness-Distance-Ratio based Particle Swarm Optimization", *Proceedings of International Symposium on Swarm Intelligence*, Vol. 2, pp. 174-181, 2003.

[9] R. Mendes, J. Kennedy and J. Neves, "The Fully Informed Particle Swarm: Simpler, Maybe Better", *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 204-210, 2004.

[10] A. Ratnaweera, S.K. Halgamuge and H.C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients", *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 240-255, 2004.

[11] K.E. Parsopoulos and M.N. Vrahatis, "Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems", *Proceedings of International Conference on Advances in Natural Computation*, Vol. 4, pp. 582-591, 2005.

[12] J.J. Liang, A.K. Qin, P.N. Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions", *IEEE Transactions on Evolutionary Computation*, Vol. 9, pp. 281-295, 2006.

[13] I.G. Tsoulos and A. Stavrakoudis, "Enhancing PSO Methods for Global Optimization", *Applied Mathematics and Computation*, Vol. 216, pp. 2988-3001, 2010.

[14] Z. Xinchao, "A Perturbed Particle Swarm Algorithm for Numerical Optimization", *Applied Soft Computing*, Vol. 10, pp. 119-124, 2010.

[15] Z.H. Zhan, J. Zhang, Y. Li and Y.H. Shi, "Orthogonal Learning Particle Swarm Optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 15, pp. 832-846, 2011.

[16] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan and Q. Tian, "Self-Adaptive Learning based Particle Swarm Optimization", *Information Sciences*, Vol. 181, pp. 4515-4538, 2011.

[17] H. Haklı and H. Uguz, "A Novel Particle Swarm Optimization Algorithm with Levy Flight", *Applied Soft Computing*, Vol. 23, pp. 333-345, 2014.

[18] R. Jensi and G.W. Jiji, "An Enhanced Particle Swarm Optimization with Levy Flight for Global Optimization", *Applied Soft Computing*, Vol. 43, pp. 248-261, 2016.

[19] S.Z. Zhao, P.N. Suganthan, Quan-Ke Pan and M. Fatih Tasgetiren, "Dynamic Multi-Swarm Particle Swarm Optimizer with Harmony Search", *Expert Systems with Applications*, Vol. 38, pp. 3735-3742, 2011.

[20] Chen Ai-Ling, Yang Gen-Ke and Wu Zhi-Ming, "Hybrid Discrete Particle Swarm Optimization Algorithm for Capacitated Vehicle Routing Problem", *Journal of Zhejiang University Science* A, Vol. 7, No. 4, pp. 607-614, 2006.

[21] A. Kaveh and S. Talatahari, "A Hybrid Particle Swarm and Ant Colony Optimization for Design of Truss Structures", *Asian Journal of Civil Engineering (Building and Housing)*, Vol. 9, No. 4, pp. 329-348, 2008.

[22] K. Premalatha and A.M. Natarajan, "Hybrid PSO and GA for Global Maximization", *International Journal of Open Problems in Computer Science and Mathematics*, Vol. 2, No. 4, pp. 597-608, 2009.

[23] Shutao Li, Mingkui Tan, W. Ivor Tsang and James Tin-Yau Kwok, "A Hybrid PSO-BFGS Strategy for Global Optimization of Multimodal Functions", *IEEE Transactions On Systems, Man and Cybernetics-Part B: Cybernetics*, Vol. 41, No. 4, pp. 1003-1014, 2011.

[24] Mustafa Servet Kıran, Eren Ozceylan, Mesut Gunduz and Turan Paksoy, "A Novel Hybrid Approach based on Particle Swarm Optimization and Ant Colony Algorithm to Forecast Energy Demand of Turkey", *Energy Conversion and Managemen*t, Vol. 53, No. 1, pp. 75-83, 2012.

[25] A.A. Ahmed and Stan Matwin, "HPSOM: A Hybrid Particle Swarm Optimization Algorithm with Genetic Mutation", *International Journal of Innovative Computing, Information and Control*, Vol. 9, No. 5, pp. 1919-1934, 2013.

[26] C. Lin, A. Qing and Q. Feng, "A Comparative Study of Crossover in Differential Evolution", *Journal of Heuristics*, Vol. 17, No. 6, pp. 675-703, 2010.

[27] A.H. Gandomi and A.H. Alavi, "Krill Herd: A New Bio-Inspired Optimization Algorithm", *Communications in Nonlinear Science and Numerical Simulation*, Vol. 17, No. 12, pp. 4831-4845, 2012.

[28] Qiaoyong Jiang, Lei Wang, Xinhong Hei, Jiatang Cheng, Xiaofeng Lu, Yanyan Lin and Guolin Yu, "ARA e-SOM+BCO: An Enhanced Artificial Raindrop Algorithm using Self-Organizing Map and Binomial Crossover Operator", *Neurocomputing*, Vol. 275, pp. 2716-2739, 2018.

[29] Ezgi Zorarpaci, "Data Clustering using Leaders and Followers Optimization and Differential Evolution", *Applied Soft Computing*, Vol. 132, pp. 1-8, 2023.

[30] P. Fanti and S. Sieranoja, "K Means Properties on Six Clustering Benchmark Datasets", *Applied Intelligence*, Vol. 48, No. 12, pp. 4743-4759, 2018.

[31] Jiawei Han and Michelin Kamber, "*Data Mining Concepts and Techniques*", 2010.

[32] D.M. Van and A.P. Engelbrecht, "Data Clustering using Particle Swarm Optimization", *Proceedings of International Congress on Evolutionary Computation*, Vol. 2, pp. 215-220, 2003.

[33] A.K. Jain, M.N. Murty and P.J. Flynn, "Data Clustering: A Review", *ACM Computing Survey*, Vol. 31, pp. 264-323, 1999.

[34] S.Z. Selim and K.S. Al-Sultan, "A Simulated Annealing Algorithm for the Clustering Problem", *Pattern Recognition*, Vol. 24, No. 10, pp. 1003-1008, 1991.

[35] P.S. Shelokar, V.K. Jayaraman and B.D. Kulkarni, "An Ant Colony Approach for Clustering", *Analytica Chimica Acta*, Vol. 509, No. 2, pp. 187-195, 2004.

[36] Y. Liu, Z. Yi, H. Wu, M. Ye and K. Chen, "A Tabu Search Approach for the Minimum Sum-of-Squares Clustering Problem", *Information Sciences*, Vol. 178, pp. 2680-2704, 2008.

[37] Yi-Tung Kao, Erwie Zahara and I-Wei Kao, "A Hybridized Approach to Data Clustering", *Expert Systems with Applications*, Vol. 34, No. 3,pp. 1754-1762, 2008.

[38] Changsheng Zhang, Dantong Ouyang and Jiaxu Ning, "An Artificial Bee Colony Approach for Clustering", *Expert Systems with Applications*, Vol. 37, No. 7, pp. 4761-4767, 2010.

[39] J. Senthilnath, S.N. Omkar and V. Mani, "Clustering using Firefly Algorithm: Performance Study", *Swarm and Evolutionary Computation*, Vol. 1, No. 3, pp. 164-171, 2011.

[40] Dervis Karaboga and Celal Ozturk, "A Novel Clustering Approach: Artificial Bee Colony (ABC) Algorithm", *Applied Soft Computing*, Vol. 11, pp. 652-657, 2011.

[41] Xiaohui Yan, Yunlong Zhu, Wenping Zou and Liang Wang, "A New Approach for Data Clustering using Hybrid Artificial Bee Colony Algorithm", *Neurocomputing*, Vol. 97, pp. 241-250, 2012.

[42] Miao Wan, Lixiang Li, Jinghua Xiao, Cong Wang and Yixian Yang, "Data Clustering using Bacterial Foraging Optimization"**,** *Journal of Intelligent Information Systems*, Vol. 38, No. 2, pp. 321-341, 2012.

[43] J. Senthilnatha, Vipul Dasb, S.N. Omkara and V. Mani, "Clustering using Levy Flight Cuckoo Search", *Proceedings of International Conference on Bio-Inspired Computing: Theories and Applications*, Vol. 5, pp. 1-11, 2012.

[44] Xin-She Yang, "Flower Pollination Algorithm for Global Optimization", *Unconventional Computation and Natural Computation, Lecture Notes in Computer Science*, Vol. 23, pp. 240-249, 2012.

[45] P.W. Tsai, J.S. Pan, B.Y. Liao, M.J. Tsai and V. Istanda, "Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems", *Applied Mechanics and Materials*, Vol. 148, No. 134, pp. 134-137, 2012.

[46] Tunchan Cura, "A Particle Swarm Optimization Approach to Clustering", *Expert Systems with Applications*, Vol. 39, No. 1, pp. 1582-1588, 2012.

[47] Y. Kumar and G. Sahoo, "An Improved Cat Swarm Optimization Algorithm for Clustering", *Computational Intelligence in Data Mining*, Vol. 8, pp. 187-197, 2015.

[48] R. Jensi and G.W. Jiji, "An Improved Krill Herd Algorithm with Global Exploration Capability for Solving Numerical Function Optimization Problems and its Application to Data Clustering", *Applied Soft Computing*, Vol. 46, pp. 230-245, 2016.

[49] Asgarali Bouyera and Abdolreza Hatamlou, "An Efficient Hybrid Clustering Method based on Improved Cuckoo Optimization and Modified Particle Swarm Optimization Algorithm", *Applied Soft Computing*, Vol. 67,pp. 172-182, 2018.

[50] F.H. Kuwil, "A Novel Data Clustering Algorithm based on Gravity Center Methodology", *Expert Systems with Applications*, Vol. 156, pp. 1-15, 2020.

[51] Sankalap Arora and Satvir Singh, "Butterfly Optimization Algorithm: A Novel Approach for Global Optimization", *Soft Computing*, Vol. 23, pp. 715-734, 2018.

[52] M. Behera, A. Sarangi, D. Mishra, P.K. Mallick, J. Shafi, P.N. Srinivasu and M.F. Ijaz, "Automatic Data Clustering by Hybrid Enhanced Firefly and Particle Swarm Optimization Algorithms", *Mathematics*, Vol. 10, pp. 1-9, 2022.

[53] B. Moyinoluwa Agbaje, E. Absalom Ezugwu and Rosanne Els, "Automatic Data Clustering using Hybrid Firefly Particle Swarm Optimization Algorithm", *IEEE Access*, Vol. 7, pp. 184963-184984, 2019.

[54] Mengjian Zhang, Daoyin Long, Tao Qin and Jing Yang, "A Chaotic Hybrid Butterfly Optimization Algorithm with Particle Swarm Optimization for High-Dimensional Optimization Problems", *Symmetry*, Vol. 12, pp. 1-7, 2020.

[55] D. Yan, H. Cao, Y. Yu, Y. Wang and X. Yu, "Single-Objective/Multiobjective Cat Swarm Optimization Clustering Analysis for Data Partition", *IEEE Transactions on Automation Science and Engineering*, Vol. 17, pp. 1633-1646, 2020.

[56] Y. Wang, K. Dang, R. Yang, L. Li, H. Li and M. Gong, "Multi- Objective Automatic Clustering Algorithm based on Evolutionary Multi-Tasking Optimization", *Electronics*, Vol. 13, pp. 1-11, 2024.

[57] Behnaz Merikhi and M.R. Soleymani, "Automatic Data Clustering Framework using Nature-Inspired Binary Optimization Algorithms", *IEEE Access*, Vol. 9, pp. 93703-93722, 2021.