Applied Intelligence (2021) 52:11606–11637
https://doi.org/10.1007/s10489-021-03020-y

Gradient-based elephant herding optimization for cluster analysis

Yuxian Duan[1,2] · Changyun Liu[1] · Song Li[1] · Xiangke Guo[1] · Chunlin Yang[2,3]

Accepted: 15 November 2021 / Published online: 28 January 2022

Abstract

Clustering analysis is essential for obtaining valuable information from a predetermined dataset. However, traditional clustering methods suffer from falling into local optima and an overdependence on the quality of the initial solution. Given these defects, a novel clustering method called gradient-based elephant herding optimization for cluster analysis (GBEHO) is proposed. A well-defined set of heuristics is introduced to select the initial centroids instead of selecting random initial points. Specifically, the elephant optimization algorithm (EHO) is combined with the gradient-based algorithm GBO for assigning initial cluster centers across the search space. Second, to overcome the imbalance between the original EHO exploration and exploitation, the initialized population is improved by introducing Gaussian chaos mapping. In addition, two operators, i.e., random wandering and variation operators, are set to adjust the location update strategy of the agents. Nine datasets from synthetic and real-world datasets are adopted to evaluate the effectiveness of the proposed algorithm and the other metaheuristic algorithms. The results show that the proposed algorithm ranks first among the 10 algorithms. It is also extensively compared with state-of-the-art techniques, and four evaluation criteria of accuracy rate, specificity, detection rate, and F-measure are used. The obtained results clearly indicate the excellent performance of GBEHO, while the stability is also more prominent.

Keywords Cluster analysis · Metaheuristic algorithm · Elephant herding optimization · Real-world datasets

1 Introduction

Machine learning can be divided into supervised and unsupervised learning, depending on whether the examples are trained with or without labels [1]. Supervised learning focuses on prediction, mainly by considering the model complexity as well as the variance and bias between samples. The main task is to obtain the corresponding response variables when observations are made on the predictor variables. In contrast, unsupervised learning focuses on observation. Unlike supervised learning, response variables are not available in unsupervised learning. Consequently, the chief task is to determine the underlying characteristics of the input variables. Specifically, a cluster analysis is a representative technique used in unsupervised learning.

✉ Song Li
polpxin@163.com

1 Air and Missile Defense College, Air Force Engineering University, Xi’an 710051, China

2 Graduate College, Air Force Engineering University, Xi’an 710051, China

3 Air Traffic Control and Navigation College, Air Force Engineering University, Xi’an 710051, China

1.1 Literature review

The concept of a cluster analysis was first introduced by Driver and Kroeber in 1932 [2]. Later, Zubin and Tryon brought it to the field of psychology. Clustering techniques are currently widely used in many fields, such as data mining [3], image segmentation [4], wireless communication [5], outlier detection [6], agricultural production [7], and e-commerce [8]. Different from classification techniques, the classes to be divided in clustering are unknown. The correlation, distribution, and variability among the data need to be analyzed from the sample data. In other words, the process divides the samples into different groups by weighing the similarity measures between them, where each group is called a cluster [9]. Homogeneity and separability are two important metrics used in cluster analyses. The former indicates the similarity between objects in the same cluster, and the latter implies the difference of objects between different clusters. The purpose of clustering is to maximize the homogeneity of the same cluster and the heterogeneity of different clusters [10]. Driven by these two concepts, various types of clustering methods have been introduced. Interestingly, different conclusions can be drawn depending on the method used.

Clustering algorithms can be broadly classified into two categories, namely, hierarchical and partitional methods [11]. Hierarchical clustering methods assume a hierarchical structure between clusters and recursively find nested clusters. Their advantage lies in that the entire clustering process can be completed at once without requiring a priori knowledge. However, this approach is computationally intensive. The main methods include DIANA, BIRCH, CURE, and CHAMELEON. The partitional method, on the other hand, simulates the lookup of all clusters as a partition of the data instead of imposing a hierarchical structure [12]. Specifically, the dataset is divided into a fixed number of clusters based on a specific criterion. These clusters are disjointed, i.e., each object belongs to only one cluster. This type of method is characterized by insensitivity to the input dataset and is easy to operate. In addition, it is computationally simple. However, the scalability is poor, and most methods fall into local optima when the dimensions of the data objects increase [13].

Due to the ease of implementation, simplicity, and efficiency, k-means clustering has become one of the most widely used clustering methods [14]. It separates all samples into the closest clusters by minimizing the sum of squared errors to find the approximate solution in a greedy manner. However, due to the nature of the gradient descent, k-means often converges to a local minimum of the objective function. For the same reason, the quality of k-means for solving clustering problems depends heavily on the initial solution [15]. If not chosen properly, the algorithm can converge slowly, and may produce empty clusters. Under this circumstance, the probability of falling into a local optimum will increase [16]. With further research, many k-means variants have emerged to overcome this problem. For example, Bortoloti et al. [17] proposed supervised kernel-density-estimation k-means, called SKDEKMeans. The kernel destiny was used to estimate a better representation of the distribution so that the balance between majority and minority clusters was achieved. A k-centroid initialization algorithm (PkCIA) was then proposed by Manochandar et al. [18]. The eigenvector of the new matrix was adopted as an index for computing initial cluster centroids. On this basis, the problem that the original algorithm is highly sensitive to the initial solution can be solved. An I-k-means-plus was proposed by Hassan [19]. According to his philosophy, the quality of the solution can be improved by removing or splitting the class clusters in each iteration. It was experimentally demonstrated that the clustering process was accelerated with a relatively higher accuracy. Huang et al. [20] developed a robust deep k-means model to learn the hidden attributes. The objective function is derived to a more trackable form to tackle the optimization problem more easily while obtaining the final robust results. An entropy-based initialization algorithm was proposed by Chowdhury et al. [21]. In their method, an entropy-based objective function was defined to finish the initialization process. Meanwhile, by using a number of cluster validity indexes, the proper number of clusters for different datasets can be calculated. Therefore, the performance of the proposed algorithm was enhanced. Zhao et al. [22] proposed another novel variant of k-means to perform top-down hierarchical clustering. It exhibited a faster speed while maintaining a lower clustering error.

Data clustering has been widely used in the real world for mining valuable information. It has long been applied in such areas such as object detection and segmentation patterns, medical risk assessment, energy exploration and development, IoT applications and anomaly detection [23].

In the real world, datasets are mostly vague, complex, and large. Meanwhile, their labels and attributes are often difficult to access smoothly. In particular, it is almost impossible to cluster the data with varied shapes, sizes, and densities [24]. In this case, an accurate and efficient estimation of the initial centroids without a priori information is urgently required [25].

Since the aims of clustering are to maximize the similarity within the same cluster and dissimilarity across clusters, it can be considered an optimization problem [26]. In optimization problems, it is often necessary to maximize or minimize some objective function (the function used to evaluate the quality of the solution). In the entire process, various difficulties, such as constraints, multiple objectives, uncertainties, and local optimum traps, need to be solved. Optimization algorithms are one of the most powerful tools to address these problems. These methods treat the problem as a black box and search for the best solution through predefined steps. Traditional optimization methods include the dynamic programming algorithm (DPA), stochastic search, steepest descent, and Newton's method [27]. The drawback of these methods is that they are usually limited by the size of a particular problem and the given dataset.

Inspired by natural phenomena and biological evolutionary behaviors, many simple and easy-to-implement metaheuristic algorithms have emerged in recent years for solving global optimization problems, such as Monarch Butterfly Optimization (MBO) [28], Slime Mould Algorithm (SMA) [29], Moth Search Algorithm (MSA) [30], Hunger Games Search (HGS) [31], Harris Hawks Optimization (HHO) [32], and others. The recent emergence of metaheuristic algorithms has developed a simple yet powerful data abstraction and analysis tool for researchers [33].

Currently, the popular research trend is to combine clustering algorithms with metaheuristics, thus ensuring a greater probability of achieving optimal clustering [34]. Chen et al. [35] proposed a new algorithm called QALO-K. k-means was optimized with a quantum-inspired ant lion to enhance the clustering performance and reach the global optimum. In addition, three clustering algorithms, GA-PFKM, PSO-PFKM, and SCA-PFKM, were proposed by Kuo et al. [36] to address the problem where fuzzy k-mode algorithms are sensitive to the initial solution. Nayak et al. [37] combined fuzzy c-means (FCM) with chemical reaction optimization (CRO) to achieve the global best solution. Aggarwal and Singh [38] introduced a nature-inspired algorithm for optimizing the k-means++ algorithm, aimed at overcoming the tendency to fall into local optima. Lakshmi et al. [39] mixed the crow search algorithm (CSA) with k-means, and the quality of the solutions obtained on the benchmark dataset was significantly improved. Due to the defect that traditional clustering methods usually perform poorly when dealing with high-dimensional optimization problems, Yang and Sutrisno [40] proposed a clustering-based SOS (CSOS) algorithm. The combination of local and global searches was achieved through cross-cluster interactions between elite individuals, thus enhancing the clustering efficiency. Note that Fuzzy C-means (FCM) tends to fall into local minima when facing complex problems. Verma et al. [41] proposed hybrid FCM and particle swarm optimization (PSO) algorithms (Hybrid FCM- PSO), while the global optimization property of PSO is used to search for cluster centers. In [42], an Automatic Clustering Local Search HMS (ACLSHMS) algorithm was proposed for image segmentation, incorporating a local search operator in the algorithm aimed at optimizing the cluster configuration of the clusters. In addition, given the effectiveness of unsupervised learning for medical image diagnosis, Mittal et al. [43] proposed a novel k-means-based improved gravitational search algorithm clustering (KIGSA-C) method for diagnosing medical images of coronavirus (COVID-19).

Considering the relevance of clustering methods to most real-world problems, there is a need to modify the current algorithms to improve the clustering performance and expand the range of applications. Cluster analysis is an open field. Researchers [2, 25, 44] encourage the application of new metaheuristic algorithms in combination with traditional clustering methods to efficiently solve complex clustering problems.

Elephant herding optimization (EHO) [45] is a novel metaheuristic algorithm proposed by Wang et al. in 2016. The algorithm has a strong global optimization capability and few control parameters [46]. Consequently, it is simple and efficient for clustering. Unfortunately, EHO still has defects, such as a lack of exploitation ability, slow convergence, and an ease of falling into local optimality. Li et al. [47] proposed an improved EHO algorithm (IMEHO) that introduced a global speed strategy and a novel learning strategy to update the speed and position of search agents. Experiments showed that the algorithm can find a better solution. Ismaeel et al. [48] proposed three EHO variants, EEH015, EEH020, and EEH025, based on the γ-value. The purpose was to overcome the problem of an unreasonable convergence to the origin. Huseyin [49] proposed a binary version of EHO. Mostafa et al. [50] presented a study of parameters in EHO. Three versions of EHO with cultural-based, alpha-tuning, and biased initialization were proposed to ameliorate the exploration and exploitation capabilities. However, none of the above algorithms are involved in the field of clustering, and their performance in clustering analysis has not been verified.

According to the no free lunch (NFL) theorem, a metaheuristic algorithm that performs well on one specific problem cannot be adapted to all optimization problems [51]. This allows researchers to add new modules and mechanisms to enhance the performance of metaheuristic algorithms. It has been determined that these hybrid algorithms can obtain a global optimal solution more efficiently than a single metaheuristic algorithm [52]. In summary, the research in this paper has a strong relevance. Inspired by this, a gradient-based elephant herding optimization for cluster analysis (GBEHO) is proposed in this paper for cluster analyses. EHO is combined with a gradient-based optimizer (GBO) [53] to further improve the convergence efficiency and exploitation capability. In addition, random wandering and variational operators are introduced to improve the ability of the algorithm to jump out of the local optimum and increase the convergence accuracy.

1.2 Contribution and organization of the paper

Overall, although many researchers have made great contributions to enhance the performance of clustering algorithms, there are still limitations. The paper contributes with six main aspects:

1. A novel hybrid metaheuristic algorithm, GBEHO, is proposed for the cluster analysis, which can automatically determine the best cluster centers.
2. Certain modifications are made to easily address the problem of falling into the local optimum. First, Gaussian chaotic mapping is introduced for initialization to generate high-quality initial populations. Second, a random wandering operator is designed to optimize the update strategy of the patriarch position. Third, a mutation operator is adopted to change the update strategy of other agents in the EHO. This prevents premature convergence and enhances the ability of the algorithm to jump out of the local optimum.

3. To prevent premature convergence and enhance the balance between exploration and exploitation, EHO is combined with GBO. A framework is developed to fuse the advantages of both algorithms, and the resulting clustering centers are evaluated using a greedy selection strategy.
4. A set of ablation experiments are designed to verify the effect of the variational probability PSR on the performance of the algorithm. The experiments are conducted on 23 recognized benchmark functions and tested statistically. The results show that the newly added operators are emphatic for the improvement of EHO, and that the optimization is most effective when $PSR = 0.2$.
5. The analysis for the different modules illustrates that the combined strategy is effective. Experiments are carried out on synthetic and real-world datasets. GBEHO is compared with nine other metaheuristics and clustering algorithms, including k-means, particle swarm optimization (PSO), differential evolution (DE), genetic algorithm (GA), cuckoo search algorithm (CS), gravitational search algorithm (GSA), bat algorithm (BA), a quantum-inspired ant lion optimized hybrid k-means algorithm (QALO-K), hybrid grey wolf optimizer and a tabu search (GWOTS). The experimental results show that GBEHO has a superior clustering accuracy and higher stability.
6. Comparative experiments are conducted with four other state-of-the-art techniques on five datasets, including CSOS, Hybrid FCM-PSO, ACLSHMS, and KIGSA-C. A variety of measures, namely, accuracy rate, specificity, detection rate, and F-measure, are adopted to evaluate the clustering effect. The experiments proved that GBEHO is an effective algorithm for clustering analysis.

The structure of this paper is shown as follows: Section 2 briefly introduces the principles of cluster analysis, EHO, and GBO. Section 3 provides a specific description of the novel concepts and design process. Section 4 conducts the experiment and analyzes the results. Discussions are given in Section 5. Finally, conclusions are summarized, and future research directions are proposed in Section 6.

2 The basic theory

2.1 Principle of clustering

Clustering is the process of organizing datasets and objects into different clusters based on certain rules [54]. In short, all data points are clustered into different clusters by comparing their similarity. Suppose there exists a set of objects $U = \{x_1, x_2, \ldots\ldots, x_n\}$ in an argument space U, where $U \in R^{n*m}$. The hard assignment follows the principle of dividing objects into K clusters $C = \{C_1, C_2, \ldots\ldots, C_K\}$. No intersection is allowed between any two clusters. This can be expressed as follows:

$$C_i \neq \emptyset, i = 1, 2, \ldots\ldots, K \tag{1}$$

$$C_i \cap C_j = \emptyset, i, j = 1, 2, \ldots\ldots K, i \neq j \tag{2}$$

$$\cup_{i=1}^{K} C_i = \{x_1, x_2, \ldots\ldots x_n\} \tag{3}$$

$$sim(X_1, X_2) > sim(X_1, Y_1), X_1, X_2 \in C_i \text{ and } Y_1 \in C_j, i \neq j \tag{4}$$

During this process, the similarity between objects in a cluster plays the most significant role in the clustering result [55]. The main way to measure the similarity in clusters is to calculate the distance between data points, such as the Mahalanobis distance [56], cosine distance [57], Pearson correlation measure [58], Jaccard measure [59], or Dice coefficient measure [60]. The most common is the Euclidean distance [61]. For two data points $x_i = \{x_{i1}, x_{i2}, \ldots\ldots x_{im}\}$ and $x_j = \{x_{j1}, x_{j2}, \ldots\ldots x_{jm}\}$ in m dimensions, the Euclidean distance is shown as follows:

$$d(x_i, x_j) = \sqrt{\sum_{n=1}^{m} (x_{in} - x_{jn})^2} \tag{5}$$

Generally, the smaller the intracluster distance or the larger the intercluster distance, the better the clustering performance [62]. In this paper, the sum of squared errors (SSE) is chosen as the objective function. SSE should be minimized in each iteration, which can be expressed as follows:

$$\min \quad SSE = \sum_{i=1}^{k} \sum_{x \in c_i} d(x, g_i)^2 \qquad \text{where} \quad g_i = \frac{\sum_{x \in c_i} x}{|c_i|} \tag{6}$$

where $d(x, g_i)^2$ denotes the squared distance from the sample point x to the center of mass g_i of cluster c_i.

2.2 EHO

EHO is a population-based algorithm proposed to simulate the nomadic life characteristics of elephants. In EHO, three principles are followed. (i) The population of all agents is divided into a specific number of clans. (ii) Each clan is led by a female individual, called a matriarch, representing the best-positioned agent in each iteration. (iii) The worst agent in each iteration represents the male elephant, who, once reaching adulthood, leaves its clan to live alone. EHO sets up the clan operator and the separation operator to model the above behavior.

2.2.1 The clan operator

For the search agent j in clan ci, its position must be modified according to the relationship with the clan leader, which can be expressed by:

$$x_{new,ci,j} = x_{ci,j} + \alpha \times (x_{best,ci} - x_{ci,j}) \times rand \tag{7}$$

where $x_{best,ci}$ is the position of the best agent in clan ci, $x_{ci,j}$ and $x_{new,ci,j}$ are the current and new positions of the search agent j in clan ci, respectively, and α and $rand$ are both random numbers between [0,1]. Unlike other member position updates, the position of the clan leader is adjusted based on the current position of all agents in the clan. This can be modeled by (8).

$$x_{new,ci,j} = \beta \times x_{center,ci} \tag{8}$$

where x_{center,c_i} denotes the central position of all agents in clan ci, which is calculated by:

$$x_{center,ci} = \frac{1}{n_{ci}} \times \sum_{j=1}^{n_{ci}} x_{ci,j} \tag{9}$$

where β affects the extent to which $x_{center,ci}$ acts on $x_{new,ci,j}$, $\beta \in [0, 1]$, and n_{ci} is the number of all agents in clan ci.

2.2.2 Separating operator

The separating operator imitates the life characteristics of male elephants. When adults, male elephants leave their current clan, represented by the following equation:

$$x_{worst,ci} = x_{\min} + (x_{\max} - x_{\min} + 1) \times rand \tag{10}$$

where r is a random number between [0,1], and $x_{\max}$ and $x_{\min}$ are the upper and lower bounds of the individual position, respectively.

2.2.3 Elitism strategy

To protect the best elephant individuals from being ruined, EHO sets an elitism strategy. At the beginning of the algorithm, the best m elephant individuals are saved. After an iteration is completed, the fitness values of the worst m elephants are compared with the best elephant individuals that were saved before, and the better agents have the opportunity to be preserved. In this way, it is ensured that the quality of the latter agents is not worse than the quality of the former agents.

2.2.4 Pseudocode of EHO

Based on the above description, the process of EHO can be summarized, and the pseudocode is shown in Algorithm 1.

Algorithm 1 EHO

```
Input:
    Initialize the maximum generation $t_{\max}$, the size of population $N$, the number of clans $c$, and the upper and lower bounds $x_{\max}$ and $x_{\min}$.
Output:
    The best solution $x_{best}$.
 1:
 2: Initialize population and parameters $\alpha$, $\beta$, $r$.
 3: Calculate and sort the fitness of every initialized agents.
 4: Save the best $m$ elephants.
 5: Divide the initial population into $c$ clans.
 6: while $t < t_{\max}$ do
 7:     for $ci = 1 : c$ do
 8:         Generate $x_{new,ci,j}$ and update $x_{ci,j}$ based on (8) to (9).
 9:         for $j = 2 : n_{ci}$ (the number of elephants in clan $ci$) do
10:             Generate $x_{new,ci,j}$ and update $x_{ci,j}$ based on (7).
11:         end for
12:     end for
13:     for $ci = 1 : c$ do
14:         Replace the agent with the worst fitness $x_{worst,ci}$ based on (10).
15:     end for
16:     Calculate and update the fitness values according to each position.
17:     Sort the entire population according to fitness.
18:     Replace the worst agents with the $m$ generated agents' elites.
19:     $t = t + 1$
20: end whilereturn $x_{best}$
```

2.3 GBO

GBO is a population-based algorithm solved by the gradient method. In GBO, the search direction is controlled by Newton's method. Additionally, two main operators and a set of vectors are adapted to explore the search space.

2.3.1 Gradient search rule (GSR)

The gradient search rule (GSR) is extracted from Newton's method to control the direction of the vector search. To ensure a balance between exploration and exploitation during the iterations and accelerate the convergence, a series of vectors are introduced as follows:

$$\rho_1 = 2 \times rand \times \alpha - \alpha \tag{11}$$

$$\alpha = \left|\beta \times \sin\left[\frac{3\pi}{2} + \sin(\beta \times \frac{3\pi}{2})\right]\right| \tag{12}$$

$$\beta = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \times \left[1 - (\frac{m}{M})^3\right]^2 \tag{13}$$

where $\beta_{\max}$ and $\beta_{\min}$ are taken as 1.2 and 0.2, respectively, m and M represent the current and the maximum number of iterations, respectively, and $rand$ denotes a random number between [0,1]. The value of α varies with the iterations and can be used to control the convergence rate. Early in the iteration, the value of α is large, thus allowing the algorithm to increase the diversity and converge quickly to the region where it hopes to find the optimal solution. Later in the iteration, the value decreases. Therefore, the algorithm can better exploit the explored regions. On this basis, the expression of GSR is as follows:

$$GSR = rand \times \rho_1 \times \frac{2\Delta x \times x_n}{(x_{worst} - x_{best} + \varepsilon)} \tag{14}$$

where x_{worst} and x_{best} represent positions of the worst and the best agents, and ε is a small number in the range of [0, 0.1]. The proposed GSR is capable of a random search, which enhances the exploration ability of GBO and the ability to jump out of the local optimum. Δx is calculated by the following expression:

$$\Delta x = rand(1:N) \times |step| \tag{15}$$

$$step = \frac{(x_{best} - x_{r1}^m) + \delta}{2} \tag{16}$$

$$\delta = 2 \times rand \times \left(\left|\frac{x_{r1}^m + x_{r2}^m + x_{r3}^m + x_{r4}^m}{4} - x_n^m\right|\right) \tag{17}$$

where $rand(1 : N)$ denotes N random numbers between [0,1] and $step$ is the step size. x_{best} represents the global optimal agent, and x_n^m denotes the m_{th} dimension of the n_{th} agent. $r1$, $r2$, $r3$, $r4$ are different integers randomly selected from [1, N].

Moreover, a motion parameter DM is set for a local search to improve the exploitation capabilities. The expression is shown as follows:

$$DM = rand \times \rho_2 \times (x_{best} - x_n) \tag{18}$$

$rand$ denotes a random number between [0,1], and ρ_2 is the parameter that controls the step size and is represented as follows:

$$\rho_2 = 2 \times rand \times \alpha - \alpha \tag{19}$$

Ultimately, the current location of the search agent (x_n^m) can be updated by GSR and DM in the following way:

$$X1_n^m = x_n^m - GSR + DM \tag{20}$$

According to 14 and 18, (20) can also be expressed as follows:

$$X1_n^m = x_n^m - rand \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(yp_n^m - yq_n^m + \varepsilon)} + rand \times \rho_2 \times (x_{best} - x_n^m) \tag{21}$$

where $yp_n^m = y_n^m + \Delta x$, $yq_n^m = y_n^m - \Delta x$, and y_n^m is a newly generated variable determined by the average of x_n^m and z_{n+1}^m. According to Newton's method, z_{n+1}^m is formulated by:

$$z_{n+1}^m = x_n^m - randn \times \frac{2\Delta x \times x_n^m}{(x_{worst} - x_{best} + \varepsilon)} \tag{22}$$

where Δx is specified by (15), and x_{worst} and x_{best} denote the current worst and best agents, respectively. After replacing the current vector x_n^m in (21) with x_{best}, a new vector $X2_n^m$ can be obtained with the following expression.

$$X2_n^m = x_{best} - rand \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(yp_n^m - yq_n^m + \varepsilon)} + rand \times \rho_2 \times (x_{r1}^m - x_{r2}^m) \tag{23}$$

Based on 21 and 23, the new solution x_n^{m+1} can be expressed as:

$$x_n^{m+1} = r_a \times [r_b \times X1_n^m + (1 - r_b) \times X2_n^m] + (1 - r_a) \times X3_n^m \tag{24}$$

$$X3_n^m = x_n^m - \rho_1 \times (X2_n^m - X1_n^m) \tag{25}$$

where r_a and r_b are random numbers between [0,1].

2.3.2 Local escaping operator (LEO)

The local escaping operator (LEO) is set to retune the resulting solution so that the algorithm can move away from local optima, improving the probability of finding the optimal solution. A solution with superior performance (X_{LEO}^m) is introduced in the LEO, which is represented as:

$$\begin{aligned}
&if \quad rand < pr \\
&X_{LEO}^m = \begin{cases} X_n^{m+1} + f_1 \times (u_1 \times x_{best} - u_2 \times x_k^m) + f_2 \times \rho_1 \times [u_3 \times (X2_n^m - X1_n^m) \\ + u_2 \times (x_{r1}^m - x_{r2}^m)]/2 & rand < 0.5 \\ x_{best} + f_1 \times (u_1 \times x_{best} - u_2 \times x_k^m) + f_2 \times \rho_1 \times [u_3 \times (X2_n^m - X1_n^m) \\ + u_2 \times (x_{r1}^m - x_{r2}^m)]/2 & otherwise \end{cases} \\
&end
\end{aligned} \tag{26}$$

pr is a predetermined threshold, where $pr = 0.5$. f_1 is a random number between [-1,1], and f_2 is a random number that conforms to the standard normal distribution. u_1, u_2, u_3 are respectively represented by:

$$u_1 = L_1 \times 2 \times rand + (1 - L_1) \tag{27}$$

$$u_2 = L_1 \times rand + (1 - L_1) \tag{28}$$

$$u_3 = L_1 \times rand + (1 - L_1) \tag{29}$$

where L_1 is a binary parameter of 0 or 1, and μ_1 is a random number between [0,1]. When $\mu_1 < 0.5$, $L_1 = 1$; otherwise,

$L_1 = 0$. In summary, the resulting solution x_k^m is expressed as follows:

$$x_k^m = L_2 \times x_p^m + (1 - L_2) \times x_{rand} \tag{30}$$

where x_p^m is a randomly selected solution from the population, $p \in [1, 2, \ldots\ldots N]$. L_2 is a binary parameter of 0 or 1, and μ_2 is a random number between [0,1]. When $\mu_2 < 0.5$, $L_2 = 1$; otherwise, $L_2 = 0$. x_{rand} is the newly generated solution in the following manner.

$$x_{rand} = X_{\min} + rand \times (X_{\max} - X_{\min}) \tag{31}$$

3 The proposed algorithm

3.1 Motivation

Traditional clustering algorithms (e.g., k-means), whose degree of validity depends on the initial solution, may fall into local optima when dealing with complex problems. Therefore, in this paper a new method is developed for data clustering. The method applies the concept of metaheuristics to automatically estimate the initial clustering centers and enhance the ability of the algorithm to escape from local optima.

The ability to balance exploration and exploitation is the concern of all metaheuristic algorithms [63]. The analysis of EHO reveals that the worst positioned agents are only randomly modified by (10). This kind of approach lacks some variation mechanism, which makes the exploitation capacity insufficient and thus leads to a slow convergence. Furthermore, the best-positioned agents are adjusted by (8). This would be useless once the population has fallen into a local optimum while reducing the diversity of the population. In addition, the capability of exploitation of EHO is relatively weak, which increases the probability of falling into a local optimum [64]. By combining with GBO, the search direction during the iteration can be guided to avoid trapping in a local optimum, resulting in a better solution. The local escape operator (LEO) in GBO can improve the diversity of the population and avoid excessive stagnation. In this case, the proposed algorithm can make full use of the gradient information so that the search efficiency of the algorithm can be improved. [65].

Based on the above reasons, several modifications are performed. First, Gaussian chaos mapping is introduced to initialize the population, thus increasing the diversity and traversal of the initial population. Next, two operators, random wandering and variation operators, are adopted to optimize the position of the agents. The aim is to achieve a better balance between exploration and exploitation. Furthermore, EHO is combined with GBO to enhance the exploitation capability by introducing GSR and LEO operators. In summary, the authors believe that this kind of modification is quite interesting.

3.2 Methodology

Since the algorithm is based on a metaheuristic, the search agents need to be represented first. Depending on the specificity of the clustering problem, the representation of the individuals is supposed to be changed. If the input dataset $U=\{x_1, x_2, \ldots\ldots, x_n\}$ includes n agents, then each object with m features can be represented as $x_i=\{x_{i1}, x_{i2}, \ldots\ldots x_{im}\}$, $i \in [1, n]$. Since one or more initial clustering centers are generated, the dimensionality dim of the algorithm will change based on the number of clusters k, i.e., dim $= m \times k$. Therefore, each candidate solution C_j denotes a set of cluster centers, which can be represented by:

$$C_j = \{c_{11}, c_{12}, \ldots\ldots c_{1m}, c_{21}, \ldots\ldots c_{\dim}\} \tag{32}$$

The solution for the initial iteration is irrelevant to the clustering problem and is randomly generated based on the available dataset. To complete the initialization process, upper and lower bounds must be determined for each feature. Namely, the lower bound is represented as $c_{\min} = \{c_{l1}, c_{l2}, \ldots\ldots c_{lm}\}$, where $c_{lm} = \min\{c_{1m}, c_{2m}, \ldots\ldots c_{nm}\}$. Similarly, the upper bound is determined as $c_{\max} = \{c_{u1}, c_{u2}, \ldots\ldots c_{um}\}$, where $c_{um} = \max\{c_{1m}, c_{2m}, \ldots\ldots c_{nm}\}$.

3.2.1 Initialization

It is noted that a strong connection exists between the quality of the initial population and the efficiency of the metaheuristic algorithm. Under this circumstance, it is necessary to improve the initialization by suitable methods to obtain a higher quality initial population. In the original EHO, the search process starts from a randomly generated initialized population. Based on that, a priori knowledge of the objective function or constraints is not required. However, it lacks ergodicity and diversity. It has been experimentally demonstrated that chaotic maps have similar properties to randomness but possess better statistical and dynamic properties [66]. Therefore, it is advantageous to use chaotic maps for population initialization in GBEHO.

In this paper, a pre-programmed Gaussian sequence [67] is selected to replace the conventional random number generator, which is represented as follows.

$$\eta(t+1) = \begin{cases} 0 & \eta(t) = 0 \\ \frac{1}{\eta(t)} - [\frac{1}{\eta(t)}] & otherwise \end{cases} \tag{33}$$

$\eta(t)$ and $\eta(t+1)$ denote the numbers of chaotic maps generated in the current and next generations, respectively. The initialized population is generated by the Gaussian chaos mapping function, which can explore the space more extensively to obtain better exploration results.

3.2.2 Random wandering operator

It should be emphasized that in the original EHO, the position of the patriarch is determined by the position of all members in the same clan. Once the algorithm has fallen into a local optimum, the quality of the best solution is difficult to modify. As a result, the populations generated by the clan operator are prone to wandering in place. This makes the algorithm somewhat lack the ability to jump out of the local optimum. In our consideration, as the best-positioned agent in each clan, the update strategy of the patriarch should be pioneering and innovative.

One of the most significant rules of metaheuristic algorithms is to maintain a balance between exploration (diversification) and exploitation (intensification). In the pre-exploration stage, agents need to explore the search space sufficiently to identify promising regions for exploitation. During this phase, individuals should have a better stochastic search ability; otherwise, it will lead to premature convergence. In the exploitation phase, agents focus on discovering better solutions in the explored regions. Therefore, the accuracy of individuals in finding the best solution should be optimized so that the algorithm converges to a feasible local or global optimal solution in a limited time. Based on this consideration, the update strategy of the patriarch is adapted as follows:

$$x_{best,ci}^{t+1} = \begin{cases} x_{best,ci}^{t} + C(\sigma) & it/Maxiter < 0.5 \\ x_{a,ci}^{t} + 2(rand - 0.5)(x_{b,ci}^{t} - x_{c,ci}^{t}) & otherwise \end{cases} \tag{34}$$

where $x_{best,ci}^{t}$ and $x_{best,ci}^{t+1}$ denote the current and latest positions of the patriarch in clan ci, respectively, $x_{a,ci}, x_{b,ci}, x_{c,ci}$ denote individuals randomly selected from clan ci, respectively, $rand$ is a random number between [0,1], and $C(\sigma)$ denotes the Cauchy distributed random number. It has been proven that a Cauchy distribution-based random walk could contribute to global exploration [68]. The Cauchy distribution function is defined as

$$F(\sigma; a, b) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{\sigma - a}{b}) \tag{35}$$

where a is the location parameter and b is the scale parameter. In the standard Cauchy distribution, $a = 0, b = 1$. Meanwhile, the Cauchy density function is as follows

$$f_{C(a,b)}(\sigma) = \frac{b}{\pi(b^2 + \sigma^2)} \tag{36}$$

The Cauchy distributed random number $C(\sigma)$ generated by (35) can be expressed by

$$\sigma = \tan\left(\pi\left(F(\sigma; a, b) - \frac{1}{2}\right)\right) \tag{37}$$

It should be noted that the random wandering operator based on the Cauchy distribution replaces the original strategy of updating based on the mean value in GBEHO. Under this circumstance, it is beneficial for agents to expand the search area, bringing an increase in diversity. For the algorithm to run smoothly, bounds should be checked to prevent crossing them. Once out of range, the Cauchy mutation is repeated several times until the new solution lies within the specified range. As the iterations continue to run, it is actually a process of decreasing the step size. Later in the algorithm, GBEHO moves to exploitation. At that time, the clan leader is modified by the position of three random individuals in the population, which contributes to improving the accuracy of discovering the globally optimal solution.

3.2.3 Mutation operator

Another deficiency of EHO is the lack of a variation mechanism, which is reflected in the following two points. First, most of the agents in a population, excluding the worst individual, are updated based on the relationship with the clan leader, and the sense of independence is poor. This type of mechanism is not conducive to enhancing the diversity. For instance, once the algorithm is caught in a local optimum, it is difficult to have the opportunity to continue exploring. Second, during the search process, a few agents broke away from the group led by the female matriarch. These agents obviously have a more prominent sense of independence and are able to perform a random search in the search space. However, their sense of following the matriarch is still relatively weak in terms of the whole clan. Unless most of them explore the wrong search direction, it will slow down the convergence speed of the algorithm and affect the search efficiency. In the original algorithm, the position of the worst individual is adjusted by the random nature, making it difficult to ensure that the search agent is updated to a better position [69].

Similar to mutations in chromosomes, mutation strategies have been widely used through genetic algorithms [70], the aim of which is to increase the diversity of the population. To ensure that most agents have the opportunity to mutate, a variance probability (PSR) is set. This parameter should take a value between (0,1) to avoid exceeding the population size boundary. If PSR is less than 0.2, it means that fewer individuals undergo mutation, and it is difficult for the experiment to have a substantial effect. If PSR is greater than 0.8, then the algorithm will determine that most of the individuals will participate in the mutation, which is contrary to the original intention of the setting. Therefore, for the purpose of maintaining a balance between exploration and exploitation while meeting the diversity enhancement requirements, the magnitude of the variance probability PSR is proposed to be experimentally tested in order to determine the optimal clustering effect. It has been experimentally verified that this module has a positive impact on the performance of the algorithm. The ablation experiments will be presented in the next section. In GBEHO, the mutation operator is set as shown below:

$$x_{worst,ci} = x_{worst,ci} + \delta_m r_1 + K \tag{38}$$

$$K = u_1 e^{\frac{-2t}{Maxiter}} \tag{39}$$

$$x_i^{t+1} = \begin{cases} x_i^t + r_2\left(\frac{x^t_{pbest}+x_{Gbest}}{2} - x_i^t\right) + r_3\left(\frac{x^t_{pbest}-x_{Gbest}}{2} - x_i^t\right) & rand < PSR \\ x_i^t + \alpha r_4 (x^t_{best,ci} - x_i^t) & otherwise \end{cases} \tag{40}$$

where $x_{worst,ci}$ represents the position of the agent to be modified and δ is the variation factor. In this paper, $\delta = 0.1 * (X_{\max} - X_{\min})$. r_1, r_2, r_3, and r_4 are random numbers uniformly distributed from 0 to 1. u_1 is a random variable of $[-1, 1]$, and t and $Maxiter$ represent the current and maximum number of iterations, respectively. x^t_{pbest} is the optimal solution at the t_{th} iteration, and x_{Gbest} stands for the global optimal solution.

3.2.4 Greedy selection strategy

When designing a hybrid framework, there are two critical issues [71]. One is to combine two or more methods into one framework, and the other is to evaluate the best solution from the iterations. In this paper, EHO is set as the basic algorithm because of its ease of implementation and certain exploration capability. The obtained solutions are then updated via GBO to enhance the diversity of the population. Compared to EHO, GBO is more advantageous in terms of its exploitation capability due to GSR and LEO. Finally, the solutions provided by the search agents are evaluated by a reedy selection strategy. If the fitness generated by the new agent is better than the current one, it is replaced and involved in a new round of iteration processes. The purpose is to ensure the convergence of GBEHO.

$$GBestX = \begin{cases} x_k^i & f(x_k^i) < f(GBestX) \\ GBestX & else \end{cases} \tag{41}$$

where $GBestX$ represents the global optimal agent, and x_k^i represents the k_{th} agent generated in the i_{th} iteration.

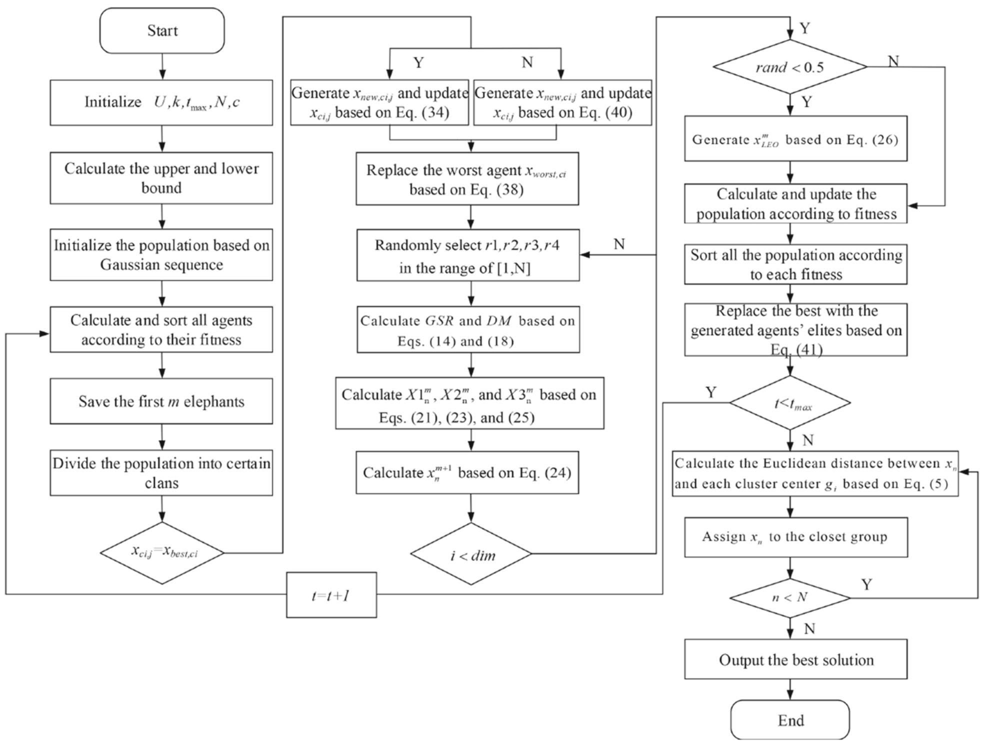
3.3 Pseudocode of GBEHO

According to the above adjustments, the pseudocode of GBEHO is shown in Algorithm 2. The initialization is performed in line 4 by means of the introduced chaotic mapping. The EHO phase is then completed in lines 7 to 16. In detail, the two proposed operators are applied in lines 11 and 15. In the second stage, the algorithm performs the gradient search rule (GSR) and local escaping operator (LEO) operators, which are shown in lines 17-28. Finally, the clustering process is completed based on the searched clustering centers in lines 33-36. In addition, the flow chart of GBEHO is given in Fig. 1.

Algorithm 2 GBEHO

```
Input:
    The original dataset $U = \{x_1, x_2, \ldots\ldots, x_N\}$, with $m$ dimensions and $k$ clusters, initializes the maximum generation $t_{\max}$, the size of population $N$, the number of clans $c$, $pr = 0.5$.
Output:
    The optimal solution $X = \{x_1, x_2, \ldots\ldots x_N\}$ in $k$ clusters.
1:
2:  Randomly generate the parameters $\alpha$, β, $r$.
3:  dim $= m \times k$.
4:  Calculate the upper and lower bound $x_{\max}$ and $x_{\min}$.
5:  Initialize the population based on the Gaussian chaotic map.
6:  Calculate and sort the fitness of every initialized agent.
7:  Divide the initial population into $c$ clans.
8:  while $t < t_{\max}$ do
9:      for $ci = 1 : c$ do
10:         Generate $x_{new,ci,j}$ and update $x_{ci,j}$ based on (34).
11:         for $j = 2 : n_{ci}$ (the number of elephants in clan $ci$) do
12:             Generate $x_{new,ci,j}$ and update $x_{ci,j}$ based on (40).
13:         end for
14:     end for
15:     for $ci = 1 : c$ do
16:         Replace the agent with the worst fitness $x_{worst,ci}$ based on (38).
17:     end for
18:     for $n = 1 : N$ do
19:         for $i = 1 :$ dim do
20:             Randomly select $r1, r2, r3, r4$ in the range of [1, N].
21:             Calculate $GSR$ and $DM$ based on (14) and (18), respectively.
22:             Calculate $X1_n^m$, $X2_n^m$, $X3_n^m$ based on (21), (23), and (25), respectively.
23:             Calculate $x_n^{m+1}$ based on (24).
24:         end for
25:         if $rand < pr$ then
26:             Generate $X^m_{LEO}$ based on (26).
27:         end if
28:         Calculate and update the fitness according to each position.
29:     end for
30:     Sort all the population according to each fitness.
31:     Replace the best with the generated agents' elites based on (41).
32:     $t = t + 1$
33: end while
34: for $n = 1 : N$ do
35:     Calculate the Euclidean distance between $x_n$ and each cluster center $g_i$ based on (5).
36:     Assign $x_n$ to the closet group which has the closet cluster center.
37: end forreturn $X = \{x_1, x_2, \ldots\ldots x_N\}$
```

Fig. 1 Flowchart of GBEHO


Start
Initialize U, k, t_{max}, N, c
Calculate the upper and lower bound
Initialize the population based on Gaussian sequence
Calculate and sort all agents according to their fitness
Save the first m elephants
Divide the population into certain clans
$x_{ci,j} = x_{best,ci}$
$t=t+1$
Y
N
Generate $x_{new,ci,j}$ and update $x_{ci,j}$ based on Eq. (34)
Generate $x_{new,ci,j}$ and update $x_{ci,j}$ based on Eq. (40)
Replace the worst agent $x_{worst,ci}$ based on Eq. (38)
Randomly select $r1, r2, r3, r4$ in the range of [1,N]
Calculate GSR and DM based on Eqs. (14) and (18)
Calculate $X1_n^m$, $X2_n^m$, and $X3_n^m$ based on Eqs. (21), (23), and (25)
Calculate x_n^{m+1} based on Eq. (24)
$i < dim$
$rand < 0.5$
Generate x_{LEO}^m based on Eq. (26)
Calculate and update the population according to fitness
Sort all the population according to each fitness
Replace the best with the generated agents' elites based on Eq. (41)
$t < t_{max}$
Calculate the Euclidean distance between x_n and each cluster center g_i based on Eq. (5)
Assign x_n to the closet group
$n < N$
Output the best solution
End

3.4 Time complexity

The time complexity of the algorithm can reflect the magnitude of the running time variation with an increase in the input size [72]. The time complexity of the proposed GBEHO is bounded by the number of search agents N, the dimensions of the problem D, and the maximum number of iterations T.

In general, the time complexity of GBEHO can be divided into the following parts: chaos initialization, random wandering, mutation, and the GBO strategy. First, the time spent initializing the population using Gaussian chaos mapping is $O(N)$. Next, the main loop phase with a maximum number of iterations of T is executed. Random wandering with a Cauchy distribution takes $O(TN)$, and the execution of the mutation operator takes $O(TN)$. In addition, the GBO strategy costs $O(TND)$, so the computational complexity of GBEHO is $O(TDN + TN)$.

4 Experiments and analysis

In this section, experiments are conducted to verify the validity of the GBEHO. All simulations are implemented on a Windows 10 operating system computer with an Intel(R) Core (TM) i5-9300H (2.40 GHz) processor, 16 GB of RAM and the MATLAB R2019b platform.

4.1 Influence of the parameters

In Section 3, the variation probability PSR is introduced into GBEHO. To verify the sensitivity of the controlled parameters, four versions of GBEHO were developed to test the performance under different parameters on a set of 23 recognized functions [73]. The values of PSR vary in the range of [0.2, 0.8] with a step size of 0.1. For the sake of convenience, these sub-algorithms are named GB2, GB3, GB4, GB5, GB6, GB7, and GB8, corresponding to PSR values of 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8, respectively. The test functions include 7 unimodal benchmark functions, 6 expandable multimodal functions, and 10 multimodal functions with fixed dimensions. The basic information of the benchmark functions is listed in Table 1. In this subsection, PSR is the only parameter that changes across GBEHO versions. For the sake of validating the parameters, the final clustering part of the original algorithm was excluded, and only the final fitness values were calculated. Furthermore, the number of clans c in GBEHO is set to 5. The maximum number of iterations $t_{\max}$ is set to 500, and the size of population N is set to 10.

To evaluate the performance of each variant, several measurement terms were invoked, including the mean and standard deviation values (std). Given the randomized nature of the heuristic algorithm, it was necessary to compare the experimental results via statistical tests in order

Table 1 Details of nine benchmark functions

No.	Function	Dimension	Range	f_{min}
F1	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
F2	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
F3	$f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	[-100,100]	0
F4	$f_4(x) = \min_i\{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	[-100,100]	0
F5	$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
F6	$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[-100,100]	0
F7	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[-1.28,1.28]	0
F8	$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500,500]	$-418.9829 \times \text{dim}$
F9	$f_9(x) = \sum_{i=1}^{\text{n}} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
F10	$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) -$ $\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	[-32,32]	0
F11	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600,600]	0
F12	$f_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4} u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[-50,50]	0
F13	$f_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[-50,50]	0
F14	$f_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1}$	2	[-65,65]	1
F15	$f_{15}(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	[-5,5]	0.00030

Table 1 (continued)

No.	Function	Dimension	Range	f_{min}
F16	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.0316
F17	$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	[-5,5]	0.398
F18	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3
F19	$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2]$	3	[1,3]	-3.86
F20	$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	[0,1]	-3.32
F21	$f_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
F22	$f_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + C_i]^{-1}$	4	[0,10]	-10.4028
F23	$f_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5363

to test the validity of the experimental data [74]. Therefore, the Wilcoxon rank-sum test and the Friedman test with a significance level of 5% were adopted, where the p value is an important indicator of the confidence of the results. When $p < 0.05$, it was determined that there was a statistically significant difference between the two groups of results. In addition, bold indicates the best candidate solution obtained in each function, and NaN indicates that the algorithm performs best on the current function. Moreover, the ranks of the results obtained by different algorithms on each function were also compared. The results obtained when each experiment was completed 30 times are shown in Table 2.

The box plots represent discrete information of a set of data, which can detect outliers and data skewness and can be used to differentiate the ability of algorithms in terms of data symmetry and dispersion [75]. The height in each boxplot reflects the stability, with narrower heights representing less noise and outliers and more stable results. The aggregation of the solution is an important factor in assessing the performance of the algorithm. If an algorithm falls into a local optimum, it will lead to premature convergence, and the quality of the solution will be degraded. 6 representative functions are selected from unimodal benchmark functions, multimodal functions, and fixed-dimension functions, with the box plots of the 7 variants and the original EHO plotted in Fig. 2. As can be observed from the results in the figure, the graphics of GB2 are relatively lower and narrower. Considering the mean, best value, worst value, and standard deviation, GBEHO performs better in different functions when $PSR = 0.2$.

The results of the 8 algorithms on 23 benchmark functions are represented in Table 2. As clearly shown in the table, the different GBEHO variants achieved higher ranks than EHO. The results show that the quality of the candidate solutions is significantly strengthened by two operators and the combination with GBO. Indeed, GBEHO obtains a lower mean and standard deviation on most functions, especially GB2. Meanwhile, the performance of the algorithm gradually decreases as the probability of variation increases. Specifically, GB2 surpasses the other algorithms on F1 to F4, F9 to F11, F13 to F15, and F17 to F19 and obtains the highest ranking. GB3 performs best on F10 and F16 and ranks second among all algorithms. Comparatively, the improvement of GBEHO with PSR greater than 0.6 is less pronounced. These results show that the newly added operators further promote the capability of local exploitation. Meanwhile, the balance between exploration and exploitation is well achieved through the combined framework. However, the performance of variants on parts of the functions is relatively insignificant, e.g., F5, F6, F8, F12, F22, and F23. It is obvious that the best solutions on those functions are achieved by the original

Table 2 Comparison results on 23 benchmark functions of EHO and GBEHO with PSR varying from 0.2 to 0.8

Datasets		EHO	GB2	GB3	GB4	GB5	GB6	GB7	GB8
F1	Mean	1.51E-03	**8.45E-281**	1.25E-265	4.64E-234	1.25E-212	5.16E-190	4.80E-169	2.19E-138
	Std	3.27E-03	**9.68E-280**	7.12E-264	8.95E-233	1.75E-211	3.96E-190	8.49E-168	1.00E-137
	p-value	2.95E-11	**NaN**	1.51E-05	6.41E-09	3.61E-11	2.95E-11	2.95E-11	2.95E-11
	rank	8	1	2	3	4	5	6	7
F2	Mean	9.46E-03	**3.09E-239**	1.38E-224	5.33E-202	4.70E-188	9.46E-162	4.66E-146	7.61E-126
	Std	1.15E-02	**1.29E-238**	7.23E-223	2.95E-202	5.66E-187	5.18E-161	1.79E-145	3.49E-125
	p-value	3.02E-11	**NaN**	5.09E-08	3.34E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
	rank	8	1	2	3	4	5	6	7
F3	Mean	1.35E+00	**1.39E-214**	1.69E-191	2.52E-179	1.98E-163	3.79E-145	9.17E-119	4.25E-103
	Std	2.91E+00	**4.52E-213**	7.69E-190	4.12E-178	8.95E-162	2.07E-144	5.03E-118	2.31E-102
	p-value	3.02E-11	**NaN**	3.03E-03	1.78E-10	3.34E-11	3.02E-11	3.02E-11	3.02E-11
	rank	8	1	2	3	4	5	6	7
F4	Mean	4.90E-03	**2.68E-240**	9.25E-218	1.19E-200	1.47E-180	1.13E-163	4.59E-144	6.00E-123
	Std	7.23E-03	**4.67E-239**	1.92E-217	5.46E-198	6.74E-179	3.87E-162	2.48E-143	3.29E-122
	p-value	3.02E-11	**NaN**	3.35E-08	6.70E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
	rank	8	1	2	3	4	5	6	7
F5	Mean	**1.15E-01**	4.23E-01	6.02E-01	7.21E-01	5.02E-01	6.13E-01	4.49E-01	4.22E-01
	Std	**1.87E-01**	4.45E-01	1.27E+00	6.84E-01	4.69E-01	1.09E+00	9.84E-01	8.43E-01
	p-value	**NaN**	1.32E-04	3.15E-05	2.50E-05	2.38E-05	3.50E-05	9.47E-05	1.68E-05
	rank	1	3	6	8	5	7	4	2
F6	Mean	**1.80E-03**	4.80E-02	6.71E-02	1.13E-01	9.93E-02	1.02E-01	9.42E-02	1.12E-01
	Std	**2.78E-03**	5.32E-02	5.82E-02	8.24E-02	9.56E-02	7.94E-02	7.56E-02	8.11E-02
	p-value	**NaN**	1.70E-08	1.33E-10	6.12E-10	8.15E-11	2.61E-10	4.08E-11	4.98E-11
	rank	1	2	3	8	5	6	4	7
F7	Mean	2.11E-03	3.83E-04	4.83E-04	5.70E-04	3.54E-04	**3.32E-04**	4.51E-04	5.20E-04
	Std	1.75E-03	3.25E-04	3.85E-04	4.48E-04	2.70E-04	**2.51E-04**	3.42E-04	3.94E-04
	p-value	1.19E-06	6.10E-03	**1.45E-01**	2.42E-02	**7.62E-01**	**NaN**	**2.71E-01**	4.84E-02
	rank	8	3	5	7	2	1	4	6
F8	Mean	**-1.26E+04**	-8.17E+03	-7.80E+03	-8.15E+03	-8.49E+03	-8.19E+03	-8.52E+03	-8.68E+03
	Std	**6.45E+00**	1.25E+02	6.25E+02	6.48E+02	9.94E+01	1.04E+02	5.92E+01	5.91E+01
	p-value	**NaN**	3.02E-11	3.02E-11	3.02E-11	3.34E-11	3.02E-11	3.02E-11	3.02E-11
	rank	1	6	8	7	4	5	3	2
F9	Mean	3.22E-02	**0.00E+00**	5.35E-28	2.68E-22	1.89E-20	4.21E-18	6.41E-15	5.47E-13
	Std	5.88E-02	**0.00E+00**	3.98E-27	5.91E-21	6.19E-19	9.13E-17	3.49E-14	6.77E-12
	p-value	1.21E-12	**NaN**	8.23E-04	5.31E-05	5.31E-05	5.31E-05	5.31E-05	5.31E-05
	rank	8	1	2	3	4	5	6	7
F10	Mean	8.34E-04	**8.88E-16**	**8.88E-16**	2.13E-13	7.64E-11	3.86E-09	5.56E-08	1.69E-06
	Std	1.12E-04	**0.00E+00**	**0.00E+00**	5.31E-12	2.19E-10	1.51E-08	8.82E-07	4.42E-06
	p-value	1.21E-12	**NaN**	**NaN**	5.36E-07	8.91E-08	1.21E-12	1.21E-12	1.21E-12
	rank	8	1	1	3	4	5	6	7

Table 2 (continued)

Datasets		EHO	GB2	GB3	GB4	GB5	GB6	GB7	GB8
F11	Mean	4.77E-04	**0.00E+00**	4.52E-20	8.52E-17	2.39E-15	1.13E-11	6.39E-09	5.61E-07
	Std	4.91E-04	**0.00E+00**	5.39E-19	2.98E-16	4.33E-14	2.56E-10	2.99E-08	4.11E-07
	p-value	1.21E-12	**NaN**	9.14E-08	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12
	rank	8	1	2	3	4	5	6	7
F12	Mean	**6.98E-05**	1.78E-03	2.66E-03	2.65E-03	2.91E-03	3.86E-03	4.03E-03	4.25E-03
	Std	**1.96E-04**	1.21E-03	1.54E-03	1.93E-03	1.89E-03	2.51E-03	2.51E-03	2.44E-03
	p-value	**NaN**	2.87E-10	4.08E-11	3.16E-10	9.92E-11	3.47E-10	3.69E-11	4.50E-11
	rank	1	2	4	3	5	6	7	8
F13	Mean	8.61E-04	**1.87E-05**	1.24E-04	9.09E-04	2.88E-03	6.42E-03	1.44E-02	2.29E-02
	Std	1.91E-03	**2.84E-05**	1.86E-04	1.73E-03	3.94E-03	8.54E-03	2.83E-02	3.56E-02
	p-value	2.77E-05	**NaN**	2.60E-05	7.62E-03	1.87E-05	3.37E-05	3.03E-03	3.16E-05
	rank	3	1	2	4	5	6	7	8
F14	Mean	9.92E-01	**9.99E-01**	9.98E-01	9.98E-01	9.94E-01	9.92E-01	9.93E-01	9.93E-01
	Std	1.44E-04	**1.52E-07**	5.24E-06	2.43E-05	8.74E-04	2.43E+00	9.65E-02	1.21E+00
	p-value	5.36E-09	**NaN**	3.67E-08	2.23E-09	3.26E-07	3.26E-07	6.77E-07	7.70E-07
	rank	7	1	2	3	4	8	5	6
F15	Mean	1.65E-03	**3.80E-04**	1.16E-03	4.83E-04	1.18E-03	1.01E-03	1.35E-03	1.95E-03
	Std	9.23E-04	**3.10E-05**	3.66E-03	2.43E-04	3.64E-03	1.10E-03	3.67E-03	5.14E-03
	p-value	4.50E-11	**NaN**	**1.54E-01**	1.71E-02	2.71E-03	3.37E-04	6.97E-03	3.03E-02
	rank	7	1	4	2	5	3	6	8
F16	Mean	-3.35E-01	-1.03E+00	**-1.03E+00**	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00
	Std	3.98E-01	8.15E-08	**1.03E-08**	2.36E-08	1.19E-07	2.40E-07	8.48E-07	1.19E-07
	p-value	3.02E-11	4.94E-05	**NaN**	**3.55E-01**	3.51E-02	2.89E-03	1.03E-06	1.86E-06
	rank	8	2	1	3	4	5	7	6
F17	Mean	4.81E-01	**3.98E-01**	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01
	Std	9.93E-02	**5.17E-09**	3.80E-08	8.81E-08	5.20E-08	2.28E-07	1.60E-07	4.78E-08
	p-value	3.02E-11	**NaN**	3.83E-06	**2.34E-01**	3.16E-10	6.35E-04	**5.75E-02**	2.32E-02

Table 2 (continued)

Datasets		EHO	GB2	GB3	GB4	GB5	GB6	GB7	GB8
	rank	8	1	2	5	4	7	6	3
F18	Mean	2.89E+01	**3.00E+00**	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	Std	7.40E+00	**1.75E-07**	2.11E-06	1.53E-06	1.50E-06	7.29E-05	4.43E-05	1.48E-05
	p-value	3.02E-11	**NaN**	2.12E-05	9.06E-08	2.17E-05	7.62E-05	**7.24E-02**	1.41E-04
	rank	8	1	4	3	2	7	6	5
F19	Mean	-2.89E+00	**-3.86E+00**	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00
	Std	5.82E-01	**1.61E-04**	1.94E-03	9.56E-04	1.14E-03	5.48E-04	8.71E-04	5.01E-04
	p-value	3.02E-11	**NaN**	**6.84E-02**	4.20E-03	**7.28E-01**	1.67E-04	2.84E-02	**7.39E-01**
	rank	8	1	7	5	6	3	4	2
F20	Mean	-1.57E+00	-3.27E+00	-3.27E+00	-3.27E+00	**-3.29E+00**	-3.29E+00	-3.27E+00	-3.28E+00
	Std	5.30E-01	7.45E-02	7.90E-02	7.62E-02	**6.36E-02**	7.08E-02	7.85E-02	7.04E-02
	p-value	3.02E-11	5.19E-04	7.06E-03	3.40E-02	**NaN**	**7.17E-01**	1.15E-02	2.52E-03
	rank	8	5	7	4	1	2	6	3
F21	Mean	-1.01E+01	-9.97E+00	-9.64E+00	-9.81E+00	-9.87E+00	-9.64E+00	-9.81E+00	**-1.02E+01**
	Std	1.76E-02	1.22E-01	1.55E+00	1.29E+00	1.29E+00	1.56E+00	1.45E+00	**1.69E-03**
	p-value	2.38E-03	7.60E-07	1.62E-02	7.06E-04	9.12E-03	**6.15E-02**	**1.91E-01**	**NaN**
	rank	2	3	7	6	4	8	5	1
F22	Mean	**-1.04E+01**	-1.02E+01	-9.34E+00	-9.87E+00	-9.65E+00	-9.51E+00	-9.52E+00	-9.26E+00
	Std	**2.27E-02**	1.83E-01	2.16E+00	1.62E+00	1.97E+00	2.01E+00	2.01E+00	2.35E+00
	p-value	**NaN**	3.51E-02	7.96E-04	3.79E-05	1.19E-05	1.05E-05	1.02E-04	2.12E-03
	rank	1	2	7	3		6	5	8
F23	Mean	**-1.05E+01**	-1.04E+01	-9.81E+00	-1.04E+01	-9.63E+00	-9.81E+00	-9.86E+00	-9.99E+00
	Std	**1.20E-02**	1.65E-01	3.27E+00	9.87E-01	2.05E+00	1.82E+00	2.14E+00	2.13E+00
	p-value	**NaN**	2.24E-02	**3.63E-01**	2.05E-03	**9.93E-02**	**6.79E-02**	3.92E-02	3.15E-02
	rank	1	2	7	3	8	6	5	4
	Average Rank	5.6087	1.8696	3.8696	4.1304	4.1818	5.2609	5.4783	5.5652
	All Rank	8	1	2	3	4	5	6	7

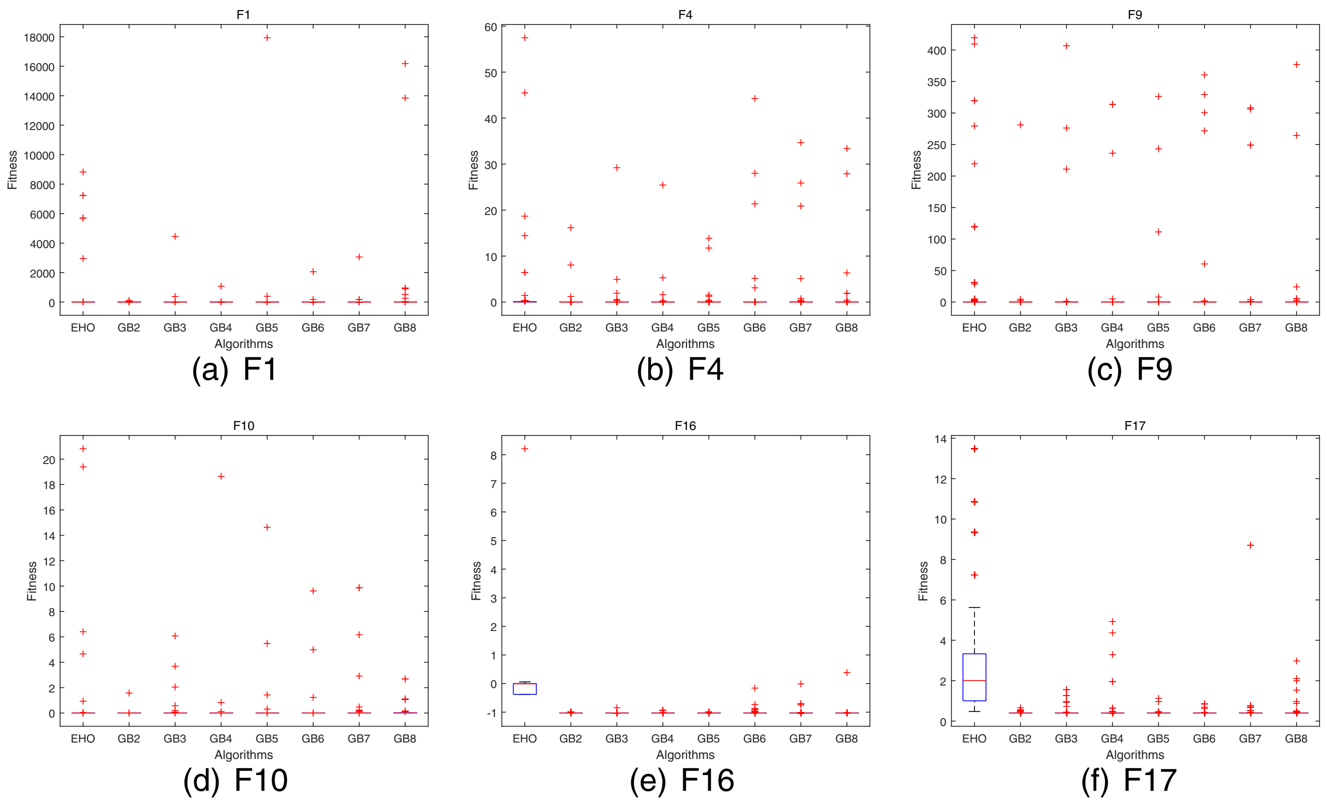


Fig. 2 Comparison of the box plots for different algorithms

EHO. This is probably due to the unique characteristics of the different functions that make the modifications of EHO inapplicable. In conclusion, the variation operator can promote the performance of the original algorithm, and the improvement is most pronounced when $PSR = 0.2$. Therefore, PSR is set to 0.2 in the subsequent experiments.

4.2 Analysis of the modifications

To investigate the impact of the modifications on the performance of the algorithm, a set of comparison experiments is conducted. In this subsection, in addition to GBEHO and EHO, three other methods, namely, Gaussian sequence + EHO (GEHO), mutation operator + EHO (MEHO), and random wandering operator + EHO (RWEHO) are designed. The above three strategies are also the core modules of the modifications. 6 representative functions are selected to verify the performance of the different variants. The size of population N is set to 30, the maximum number of iterations $t_{\max}$ is 500, and the number of clans c is 5. Other than that, other parameters are kept consistent. To reduce the effects of errors and instabilities, all algorithms are subjected to 30 experiments. The final results are based on an average of 30 experiments.

Figure 3 shows the convergence curves of 5 algorithms on the 30-dimensional functions. The convergence efficiency of the other 4 variants is significantly better than the original EHO. This indicates that the modifications of Gaussian mapping, random wandering, and mutation operators can indeed all improve the convergence efficiency of the EHO. Specifically, the Gaussian sequence improves the initialization, which leads to an increased efficiency in the early stages of the algorithm. In addition, the global optimal solutions achieved by MEHO and RWEHO are superior to EHO and GEHO. It is therefore proven that random wandering and mutation operators enhance the diversity of the population. Consequently, exploration and exploitation are promoted, leading to a higher convergence accuracy. In comparison, GBEHO has the best convergence performance. The global optimum is attained around the 300th and 10th generations on the F2 and F9 functions, respectively. The convergence rate on the F11, F14, F15, and F20 functions is also the fastest among several algorithms. These results provide strong evidence that the combined effect of modifications has led to further improvements in the search accuracy and breadth of GBEHO.

In addition, the average, standard deviation, best, and worst values of different variants on the six benchmark

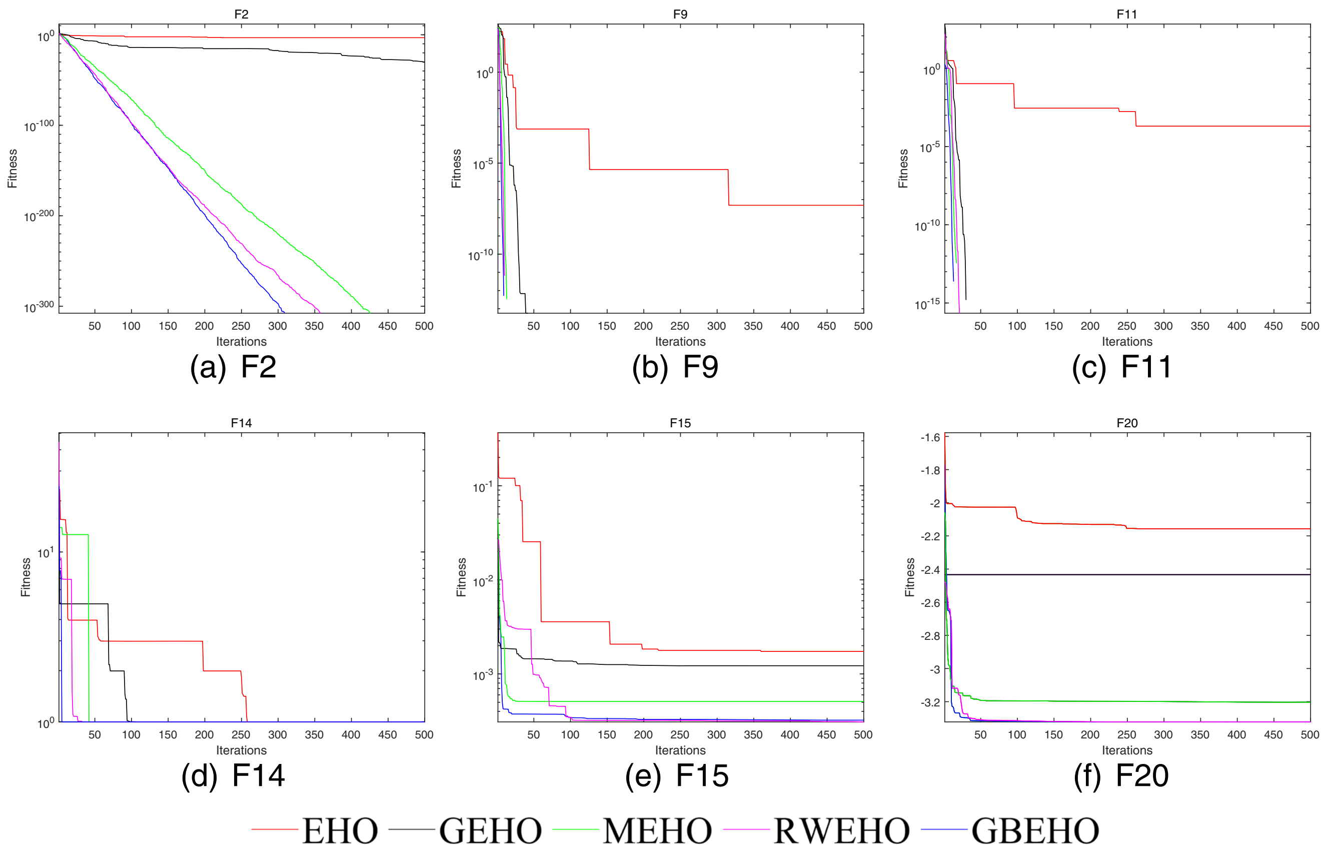


Fig. 3 Comparison of box plots for different algorithms

functions are recorded and reported in Table 3. The results in the table are the average results obtained after 30 runs of each algorithm, and the best results on each function are shown in bold. It is obvious that all variants perform better than the original EHO algorithm, indicating that the strategy of Gaussian sequence, the random wandering, and the mutation operators are efficient, respectively. Besides, it is also worth noting that GBEHO achieves the most desirable performance overall, with the best average and standard deviation results on F2, F9, F11, F14, and F15. This indicates that the combination of different strategies is effective in a way that can significantly improve exploration and exploitation. In summary, it can be concluded that the modifications of EHO are convincing.

4.3 Comparison with other metaheuristic algorithms

To further verify the effectiveness of the algorithm, GBEHO was compared with nine other metaheuristics, namely, k-means, particle swarm optimization (PSO) [76], differential evolution (DE) [77], genetic algorithm (GA) [70], cuckoo search algorithm (CS) [78], gravitational search algorithm (GSA) [79], bat algorithm (BA) [80], a quantum-inspired ant lion optimized hybrid k-means algorithm (QALO-K) [35], hybrid grey wolf optimizer and a tabu search (GWOTS) [81].

4.3.1 Parameter settings

Under the consideration of fairness, parameters within the selected algorithm are preset, which are shown in Table 4. It should be noted that the parameters in the table are set according to the recommendations in the above work. Except for the parameters in the table, the other parameters are kept consistent. Furthermore, the maximum number of iterations $t_{\max}$ is set to 200, and the size of population N is set to 10. The number of clans c in GBEHO is set to 5.

4.3.2 Datasets

Adán et al. [34] stated that the evaluation of a complete clustering algorithm should include both synthetic and standard real-world datasets. The datasets chosen for the experiments are from the University of California, Irvine

Table 3 Comparison results on 6 benchmark functions of different variants

Datasets		EHO	GEHO	MEHO	RWEHO	GBEHO
F2	Mean	3.42E-03	1.53E-24	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	4.83E-03	8.37E-24	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Best	2.12E-05	4.15E-45	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Worst	2.46E-02	4.58E-23	**0.00E+00**	**0.00E+00**	**0.00E+00**
F9	Mean	6.44E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	1.59E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Best	3.28E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Worst	6.33E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F11	Mean	1.20E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	1.47E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Best	5.39E-06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Worst	3.88E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F14	Mean	9.98E-01	9.98E-01	9.98E-01	9.98E-01	**9.98E-01**
	Std	2.21E-04	3.44E-04	4.31E-07	2.83E-09	**4.55E-15**
	Best	9.98E-01	9.98E-01	9.98E-01	9.98E-01	**9.98E-01**
	Worst	9.99E-01	9.99E-01	9.98E-01	9.98E-01	**9.98E-01**
F15	Mean	7.59E-02	2.95E-02	6.34E-03	5.95E-04	**3.72E-04**
	Std	2.76E-03	2.34E-04	3.62E-03	4.19E-04	**6.70E-05**
	Best	5.35E-03	9.09E-03	2.08E-03	3.12E-04	**3.08E-04**
	Worst	1.70E-02	1.60E-02	9.04E-03	7.05E-04	**5.72E-04**
F20	Mean	−2.19E+00	−3.24E+00	−3.32E+00	**−3.32E+00**	−3.32E+00
	Std	7.85E-01	5.48E-02	5.51E-02	2.86E-02	**5.37E-03**
	Best	−2.87E+00	−3.32E+00	−3.32E+00	**−3.32E+00**	−3.32E+00
	Worst	−2.04E+00	−3.15E+00	−3.27E+00	−3.28E+00	**−3.29E+00**

(UCI) machine learning repository [82] and include Iris, Wine, Seeds, Breast, Heart, CMC, and Vowel. The synthetic dataset consists of two artificial datasets: two-moon and aggregation [83]. The basic information of the datasets is shown in Table 5.

4.3.3 Comparison of the experimental results

In this section, the various algorithms are compared based on the experimental values of SSE. Each algorithm is run 30 times separately, and the obtained results are shown in Table 6. Best, Worse, Mean, and Std. denote the best, worst, mean, and standard deviation of all the results, respectively. Obviously, it can be seen that the algorithms produce separate values due to the complexity of the dataset. GBEHO can provide the lowest solutions in most datasets. Compared to the basic k-means algorithm, GBEHO achieves better mean values in all cases.

For 9 datasets, GBEHO can provide the lowest mean SSE results for 7 datasets: Wine, Seeds, Breast, Heart, CMC, Vowel, and Aggregation. In particular, GBEHO achieves the lowest best and worst values on these datasets. However, due to the inability to accurately identify the manifold structure, GBEHO performed poorly in the Two-moon dataset. The standard deviations of GBEHO are smaller than those of the other algorithms, indicating that the algorithm is more stable in its operation. In general, GBEHO could obtain more satisfactory results than the other 9 algorithms. Consequently, these results provide strong proof for GBEHO to solve the clustering problem effectively.

Figure 4 shows the box plots obtained by 9 algorithms on the different datasets. It is observed that the box plots of GBEHO are the narrowest among all data sets. Obviously, GBEHO has a more stable clustering ability, and the population diversity is ameliorated by using the strategy of mixing EHO and GBO. In addition, GBEHO produced the fewest outlier points, which indicates that GBEHO has a strong robustness. These facts indicate that the proposed algorithm can effectively circumvent local minima.

4.3.4 Convergence analysis

Iteration is the act of repeating a set of procedures to achieve the best solution. When all procedures of an algorithm are repeated once, this is called one iteration. The results of

Table 4 Parameter settings of the different algorithms

Algorithm	Parameter	Range
GBEHO	c	5
	pr	0.5
	PSR	0.2
PSO	c_1	2
	c_2	2
	$\omega_{\max}$	0.9
	$\omega_{\min}$	0.2
DE	Differential weight	0.8
	Crossover probability	0.2
GA	Crossover rate	0.7
	Mutation rate	0.2
	Selection rate	0.8
CS	P	0.25
GSA	G_0	100
	a	20
BA	α	0.9
	τ	0.9
	$f_{\min}$	0
	$f_{\max}$	2
QALO-K	F	0.2
	CR	0.5
GWOTS	α	[2,0]
	bw	0.01
	$Tabusize$	5

each iteration provide the initial value for the next iteration [84]. The convergence curve can reflect the convergence rate and the global search ability during the iteration of the algorithm.

The comparison of convergence curves on different datasets is shown in Fig. 5. All curves are generated synthetically after 30 independent runs of the different algorithms. GBEHO reaches stability at the 20th generation on the Iris, Wine, Seeds, Breast, Heart, and Aggregation datasets. Despite the fact that GBEHO converges more slowly on the Vowel dataset, the quality of the solutions found is higher. The results verify that GBEHO has relatively faster convergence and a superior global search capability. Compared with GBEHO, the performances of metaheuristics for PSO, DE, GA, CS, GSA, and BA are slightly less.

4.3.5 Statistical analysis

In the proceeding experiments, there are inevitable chance factors that affect the experimental results. To test the variability between different algorithms, further statistical analysis of the obtained results is needed to obtain more reliable data. Nonparametric tests can be used in the field of mathematics to check the performance of the algorithms [85]. The Wilcoxon signed-rank test [86] and Friedman test [87] are two well-known techniques. Both can be used on data distributions, statistically examining whether a difference exists between two groups. The experiments in this paper are performed at the 5% significance level.

Table 7 reports the results for the comparison of GBEHO with PSO, GBEHO with DE, GBEHO with GA, GBEHO with CS, GBEHO with GSA, GBEHO with BA, GBEHO with QALO-K, GBEHO with GWOTS, and GBEHO with k-means on the nine groups. If the p-value is less than 0.05, then the result is significantly different. The bold values in the table indicate values greater than 0.05. As observed from the table, except for the values obtained for GBEHO vs. CS on Iris and GBEHO vs. PSO on the Heart dataset, which are greater than 0.05, all other values are less than 0.05, which provides valid evidence against the null hypothesis. The results suggest that the excellent performance of GBEHO is statistically significant, and not achieved by chance.

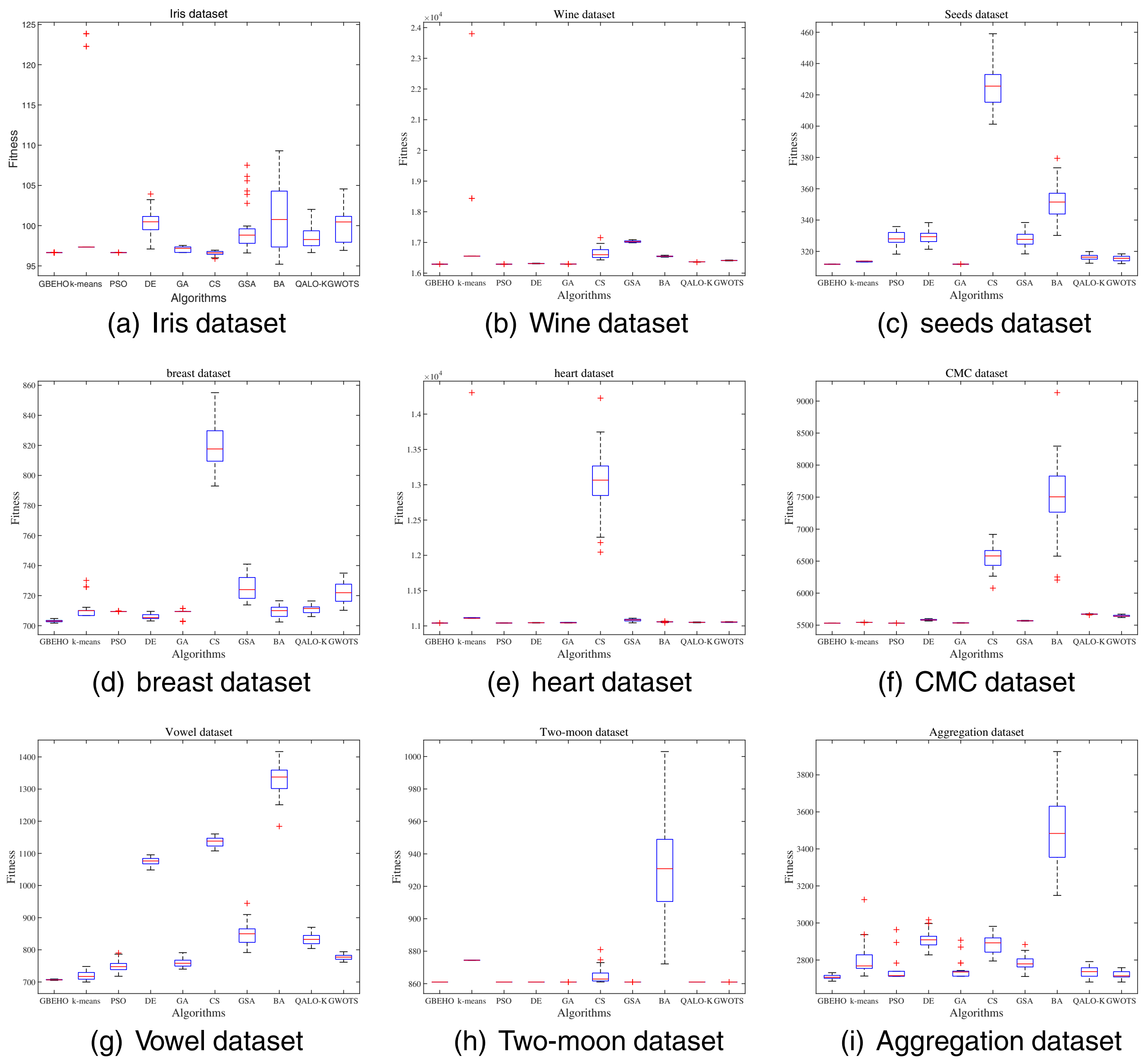
The results of the Friedman test are shown in Table 8. The obtained values are the average ranking of all algorithms

Table 5 Basic information of the datasets

Title	Number	Attribute	Cluster
Two-moon	1502	2	2
Aggregation	788	2	7
Iris	150	3	4
Wine	178	3	13
Seeds	210	3	7
breast	277	9	2
heart	270	13	2
CMC	1473	9	3
Vowel	874	3	6

Table 6 Comparison results on different datasets

Datasets		GBEHO	k-means	PSO	DE	GA	CS	GSA	BA	QALO-K	GWOTS
Iris	Mean	96.6555	101.65	96.6555	100.5246	97.0601	**96.5923**	99.6906	102.4521	98.4628	99.9655
	Std	**6.42E-10**	9.8172	2.14E-09	1.4261	0.3509	0.2451	2.9389	3.5164	1.2551	2.1593
	Best	96.6555	97.3259	96.6555	97.0982	96.6611	95.8766	96.6088	**95.5132**	96.6555	96.9264
	Worst	**96.6555**	123.8497	**96.6555**	103.9348	97.2282	96.9519	107.5079	108.9816	102.0196	104.5636
Wine	Mean	**16292.1797**	16922.5807	16292.4966	16313.0385	16295.4036	16652.5692	17035.6734	16551.9985	16368.5625	16413.1972
	Std	**0.1128**	1383.6805	0.5118	5.6243	1.8804	169.3483	27.3914	15.1421	3.5621	9.3362
	Best	**16290.8845**	16555.6794	16292.1876	16304.4056	16292.672	16432.0892	16989.6022	16522.4516	16357.4776	16399.2559
	Worst	**16293.1294**	23800.1739	16293.9858	16324.7719	16302.228	17155.5986	17090.6633	16582.9983	16374.3042	16429.2935
Seeds	Mean	**311.7978**	313.5273	311.7979	328.3513	311.8186	425.4047	327.9767	353.1714	316.3263	315.4146
	Std	8.24E-07	0.2578	**0**	4.8276	0.0196	13.5094	4.5986	14.5277	2.0355	1.8022
	Best	**311.7978**	313.2168	**311.7978**	321.3633	311.8018	401.2814	318.404	331.5712	312.4252	312.1504
	Worst	**311.7978**	313.7343	311.7981	338.3638	311.8956	459.032	338.4472	379.0951	319.8667	318.4113
breast	Mean	**702.5287**	710.8969	709.2409	705.9913	708.3464	820.0942	725.4121	707.2217	711.8086	721.7778
	Std	**0.5271**	6.6782	1.3015	1.7923	2.8458	16.253	8.1136	5.8082	2.5663	6.4781
	Best	**701.8935**	706.7196	708.8748	703.1575	702.8753	792.9774	713.8192	702.2515	706.0635	710.2501
	Worst	**704.2451**	730.1364	709.4936	709.5189	711.4823	855.0989	740.9746	717.5345	716.4412	735.0085
heart	Mean	**11040.586**	11218.3322	11041.2554	11044.8615	11045.75	13057.0111	11076.9611	11058.7853	11050.7003	11054.8329
	Std	**9.43E-06**	582.7805	0.3517	1.917	3.8177	456.0667	20.5362	8.7843	2.5672	4.1267
	Best	11040.586	11108.7834	**11039.8159**	11041.0695	11041.4541	12044.4999	11043.4089	11046.6782	11044.2447	11048.511
	Worst	**11040.5861**	14303.8768	11043.2214	11048.5204	11051.4777	14226.2367	11109.0933	11068.5732	11056.5577	11060.7088
CMC	Mean	**5531.4566**	5543.9922	5532.0356	5582.5772	5535.1876	6544.9993	5567.4964	7513.5361	5671.2282	5642.1387
	Std	**7.12E-05**	1.522	0.0018	10.2814	2.2088	182.9644	3.2124	602.5733	4.7362	13.4821
	Best	**5531.1835**	5539.8891	5532.1851	5565.5292	5532.4447	6078.6424	5562.3148	6205.1573	5658.0608	5619.5534
	Worst	**5532.1124**	5547.2265	5532.1921	5601.7021	5540.069	6918.2407	5575.149	9133.2466	5678.3702	5672.4519
Vowel	Mean	**706.2537**	718.0558	749.5712	1074.9043	759.6974	1135.8144	858.994	1333.5731	827.0123	776.8145
	Std	**0.8085**	12.6938	15.2169	12.7467	12.8204	15.229	35.7316	45.6782	17.3256	7.9426
	Best	**705.2218**	706.8515	718.1412	1048.3263	739.9638	1107.727	791.5659	1182.7827	804.0304	761.5453
	Worst	**708.0811**	748.2787	779.1062	1095.4346	791.1734	1160.4399	944.5888	1418.2864	869.9818	794.1206
Two-moon	Mean	861.5887	874.4292	**861.0022**	861.0024	861.0027	864.9261	861.0024	935.3676	861.0048	861.0025
	Std	**2.87E-15**	3.47E-13	1.63E-11	3.05E-04	1.24E-03	4.7591	2.49E-04	35.7313	7.83E-05	2.85E-07
	Best	861.5887	874.4292	**861.0022**	**861.0022**	**861.0022**	861.0596	**861.0022**	871.0532	861.0046	861.0025
	Worst	861.5887	874.4292	862.0022	861.0033	861.0088	881.0101	861.003	1005.6893	861.0049	**861.0025**
Aggregation	Mean	**2713.3419**	2801.593	2741.3152	2912.3635	2741.3672	2883.8272	2783.4515	3482.1218	2737.9821	2719.3046
	Std	**8.1782**	88.7827	56.7838	46.2044	44.5043	48.0931	35.6736	176.3351	32.1578	21.2672
	Best	**2687.1285**	2713.3486	2713.7931	2827.733	2712.2452	2794.5799	2710.9933	3142.4736	2695.249	2691.1079
	Worst	**2730.0819**	3126.1613	2967.2318	3017.472	2906.9183	2981.8651	2883.6941	3928.2248	2791.453	2758.5424



(a) Iris dataset (b) Wine dataset (c) seeds dataset

(d) breast dataset (e) heart dataset (f) CMC dataset

(g) Vowel dataset (h) Two-moon dataset (i) Aggregation dataset

Fig. 4 Comparison of box plots for different algorithms

when conducting the experiments. According to the results, the algorithm with the lower ranking is considered to be the most efficient algorithm. Obviously, a better average ranking of GBEHO proves that the proposed algorithm has a more competitive advantage. At the same time, it makes the series of experiments more convincing.

4.3.6 Analysis of the clustering process

In this subsection, three datasets, Iris, Seeds and Aggregation, are selected for visualization and presentation. The original distributions are shown in Fig. 6. Figures 7, 8 and 9 display the clustering visualization results. We know that GBEHO and PSO are the two best algorithms on the Iris dataset. It can be observed in Fig. 7 that both algorithms accurately divide the dataset into three distinct clusters, and both achieve relatively better solutions. In comparison, the centroids found by GBEHO are significantly closer to the real scenario than PSO. This suggests that GBEHO has a better performance. In terms of the iterations, the centroids found by GBEHO are relatively stable at the 20th generation. This indicates that GBEHO has a faster convergence

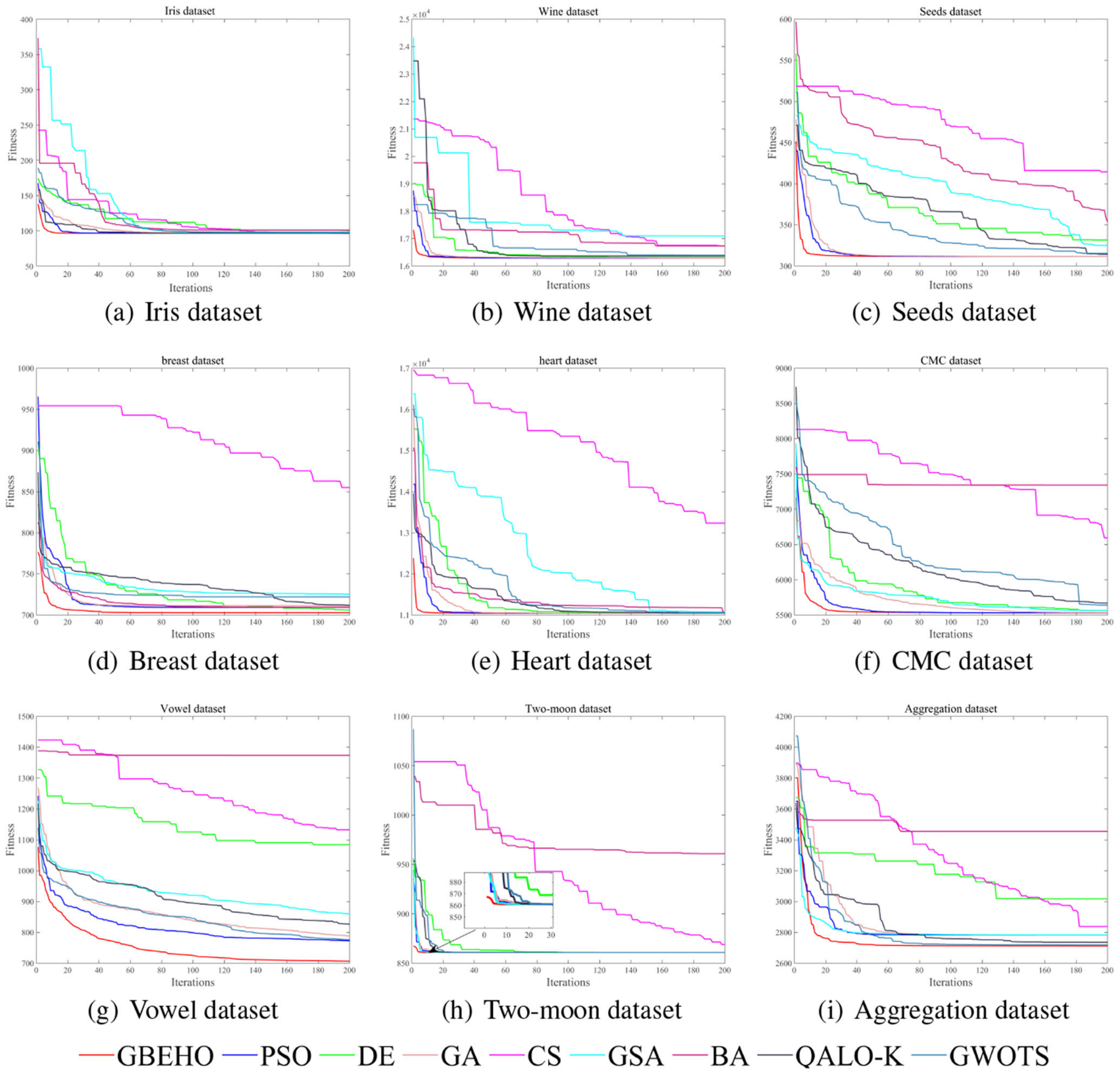


Fig. 5 Convergence curves of different algorithms

rate and stability. Figure 8 compares the clustering results on the Seeds dataset, where GBEHO and GA are the two superior algorithms. Apparently, GBEHO achieves better positions of centroids in the 20th generation and in the final results. In the 20th generation, GBEHO is able to extract centroids of the bottom leftmost cluster, while GA is unable to. It is clear that GBEHO is able to distinguish blue and green clusters more accurately than GA. The performance on the Aggregation dataset is shown in Fig. 9. For the two clusters on the top left and top, GBEHO obtains more precise clustering centroids. Both GWOTS and GBEHO find the exact centroids on the upper and lower right clusters. However, GBEHO's delineation in the bottommost cluster is more obvious. Although the black and magenta clusters in Fig. 6c are not accurately distinguished, this is due to the shortcomings of the traditional Euclidean distance. In terms of the overall convergence rate and clustering accuracy, GBEHO is relatively superior.

Table 7 Results of p-values obtained by the Wilcoxon signed-rank test

Datasets	PSO	DE	GA	CS	GSA	BA	QALO-K	GWOTS	k-means
Iris	3.25E-02	9.17E-12	9.17E-12	**6.54E-01**	1.97E-10	4.34E-05	9.17E-12	9.17E-12	5.77E-12
Wine	1.76E-02	3.02E-11	6.07E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.16E-12
Seeds	1.01E-11	1.01E-11	1.01E-11	1.01E-11	1.01E-11	1.01E-11	1.01E-11	1.01E-11	4.14E-12
breast	3.02E-11	1.17E-09	2.78E-07	3.02E-11	3.02E-11	2.67E-09	3.02E-11	3.02E-11	2.57E-11
heart	**1.85E-01**	2.68E-11	2.68E-11	2.68E-11	2.68E-11	2.68E-11	2.68E-11	2.68E-11	1.41E-11
CMC	2.88E-11	2.88E-11	2.88E-11	2.88E-11	2.88E-11	2.88E-11	2.88E-11	2.88E-11	2.88E-11
Vowel	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.56E-04
Two-moon	1.92E-09	1.95E-09	1.93E-10	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.15E-12	1.69E-14
Aggregation	3.71E-05	3.02E-11	1.49E-06	3.02E-11	9.92E-11	3.02E-11	1.49E-04	6.38E-03	1.40E-09

Table 8 Results of ranks obtained by the Friedman test

Datasets	GBEHO	k-means	PSO	DE	GA	CS	GSA	BA	QALO-K	GWOTS
Iris	1.00	10.00	2.00	5.00	4.00	3.00	8.00	9.00	6.00	7.00
Wine	1.00	10.00	2.00	4.00	3.00	9.00	8.00	7.00	5.00	6.00
Seeds	1.00	3.00	6.00	7.00	2.00	10.00	8.00	9.00	5.00	4.00
breast	1.00	7.00	2.00	3.00	4.00	10.00	9.00	6.00	5.00	8.00
heart	1.00	10.00	2.00	3.00	4.00	9.00	8.00	7.00	5.00	6.00
CMC	1.00	4.00	2.00	6.00	3.00	9.00	5.00	10.00	8.00	7.00
Vowel	1.00	2.00	3.00	8.00	4.00	9.00	7.00	10.00	6.00	5.00
Two-moon	2.00	8.00	1.00	5.00	7.00	9.00	4.00	10.00	6.00	3.00
Aggregation	1.00	9.00	6.00	8.00	5.00	7.00	4.00	10.00	3.00	2.00
Avg.rank	1.11	7.00	2.89	5.44	4.00	8.33	6.78	8.67	5.44	5.33
Overall rank	1.00	8.00	2.00	5.00	3.00	9.00	7.00	10.00	5.00	4.00

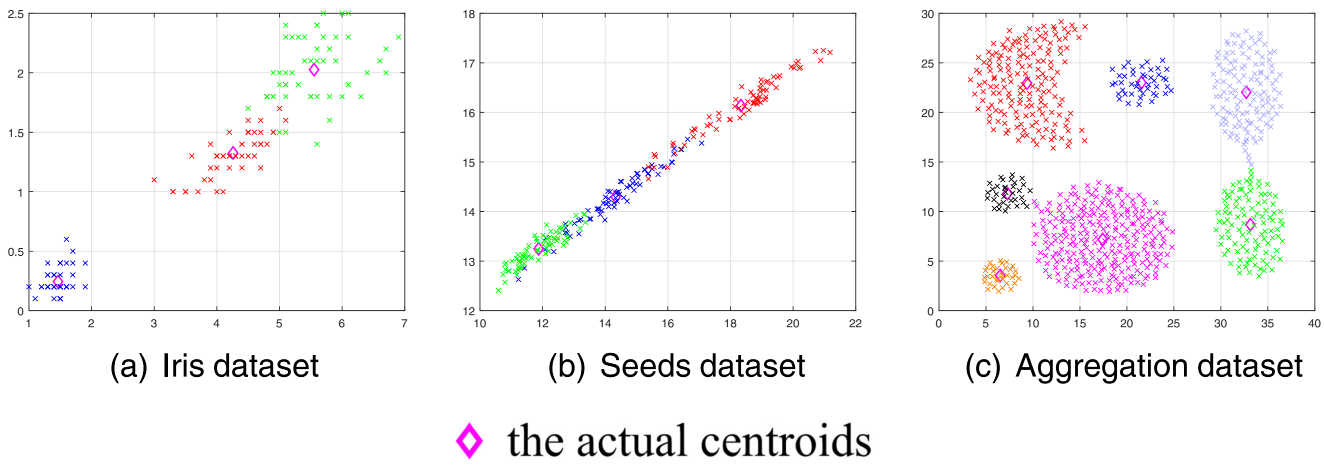


Fig. 6 The original distribution

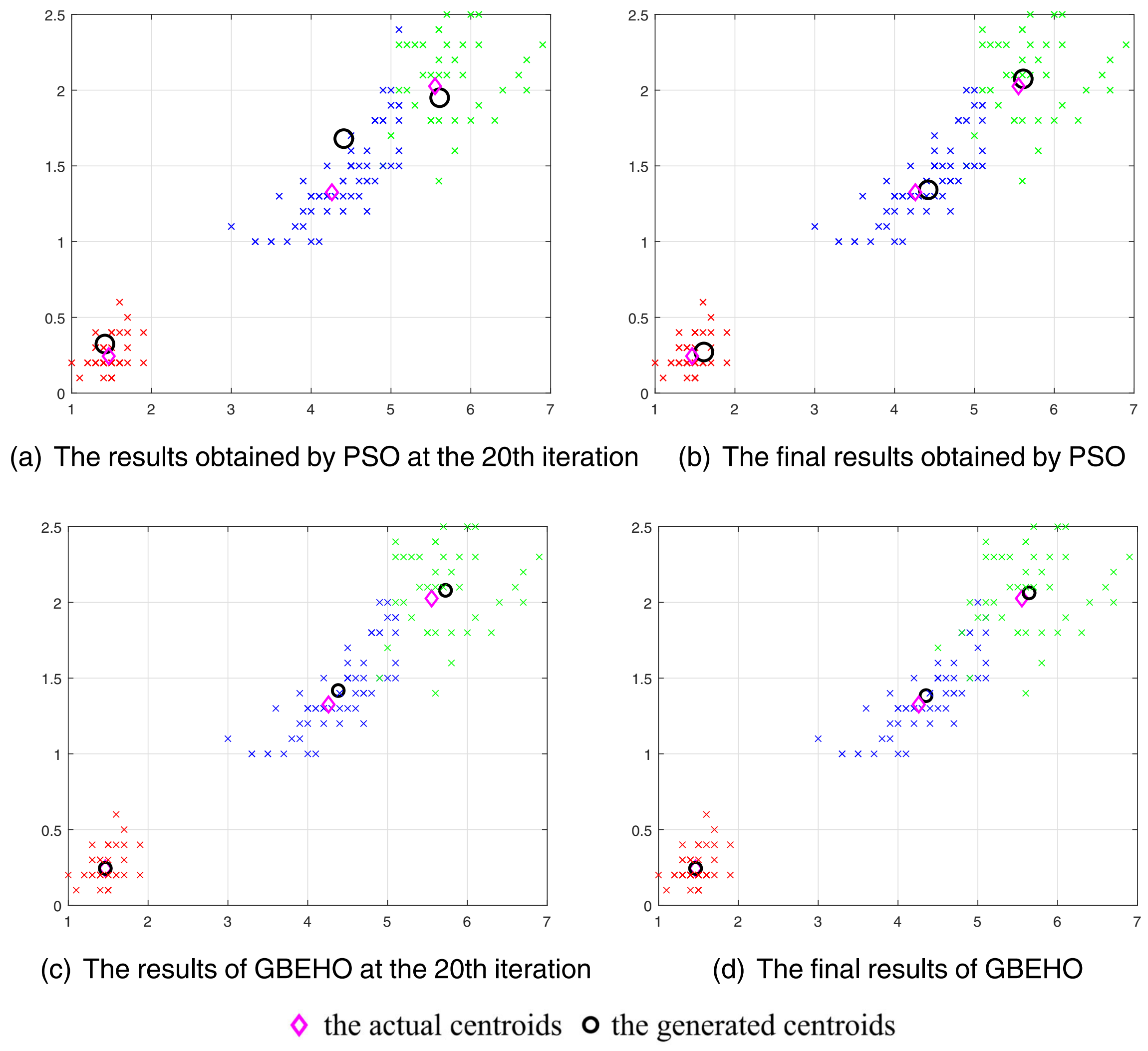


Fig. 7 Comparison of the clustering results on the Iris dataset

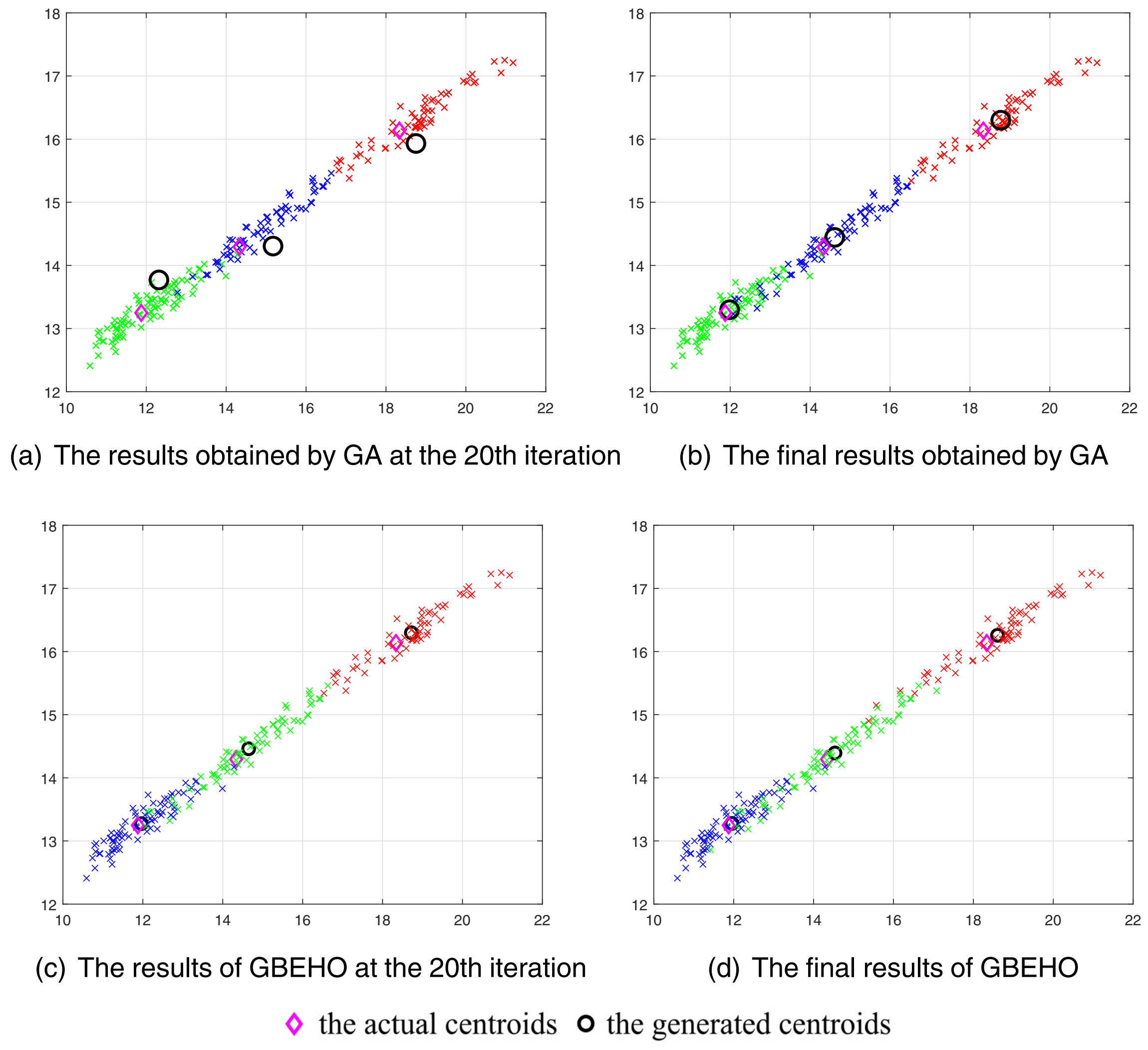


Fig. 8 Comparison of clustering results on the Seeds dataset

4.4 Comparison experiments with state-of-art techniques

In this subsection, extra experiments are conducted to further validate the performance of the proposed algorithm.5 UCI datasets, namely, Wine, Breast, CMC, Heart, and Vowel, are chosen to evaluate the significance of GBEHO with $PSR = 0.2$ versus the reported results of four other recently proposed algorithms, such as CSOS, Hybrid FCM-PSO, ACLSHMS, and KIGSA-C. Table 9 shows the values of the experimental parameters for the different algorithms. The maximum number of iterations $Maxit$ is set to 500, and the size of population N is set to 30. To eliminate the influence of uncontrollable factors to the greatest extent possible, all algorithms were run 30 times, and the average value was adopted as the final result for comparison.

When completing the clustering of the dataset, attention needs to be given to the degree of adaptation of the clusters to the input data. Therefore, it is necessary to validate via certain evaluation criteria, which is a fundamental aspect of data clustering. The metrics for evaluating the clustering results are broadly classified into three categories, namely, external metrics, internal metrics, and relative validation [25]. Four evaluation metrics are invoked in the experiments to quantitatively compare the clustering performance, namely, accuracy rate (AR), specificity (SP), detection rate (DR), and F-measure (F_1), which are defined in 42-45.

$$AR = \frac{TP + TN}{TP + TN + FN + FP} \tag{42}$$

$$SP = \frac{TN}{TN + FP} \tag{43}$$

$$DR = \frac{TP}{TP + FN} \tag{44}$$

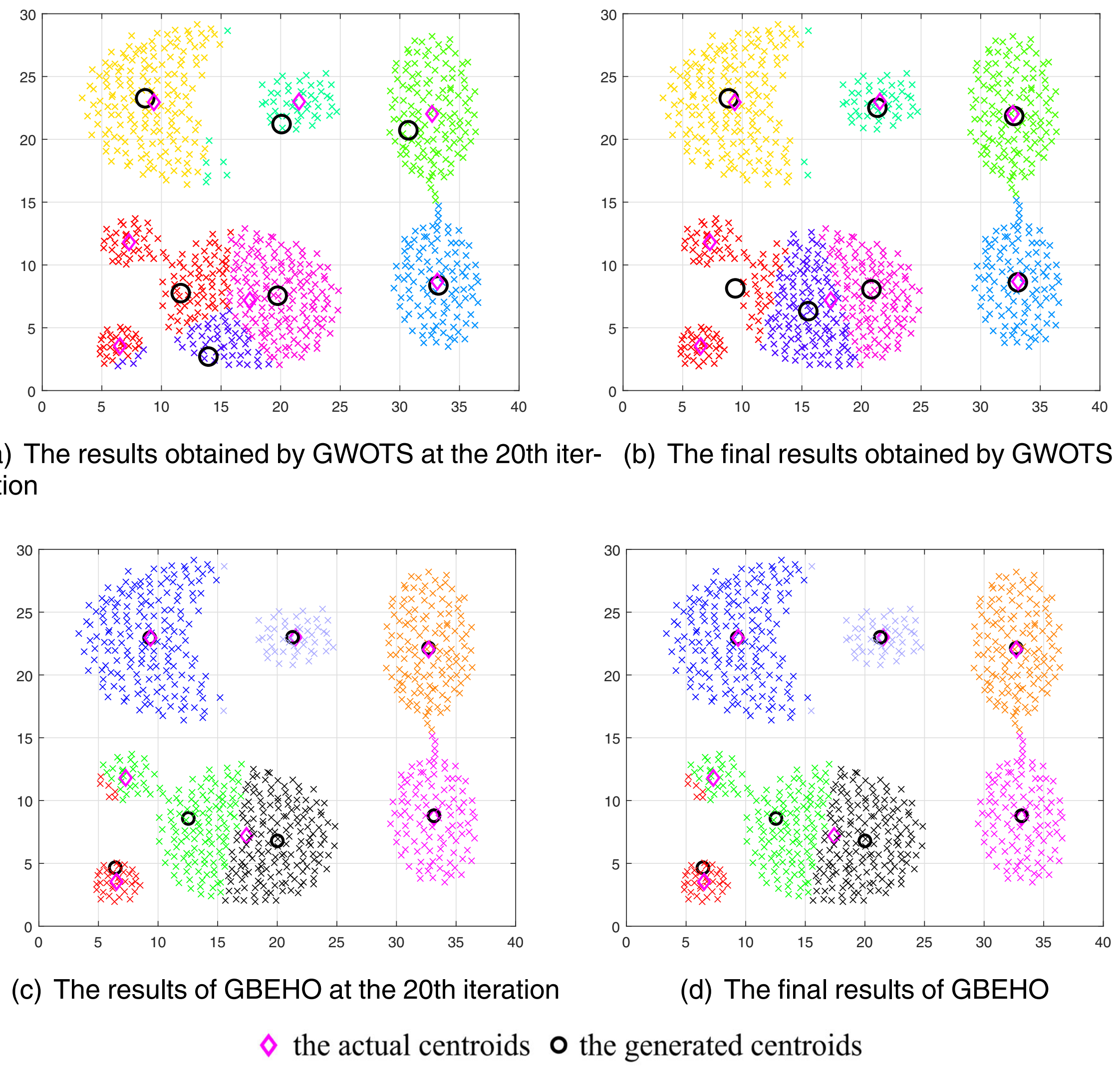


Fig. 9 Comparison of clustering results on the Aggregation dataset

$$F_1 = \frac{\left(b^2+1\right)\cdot precision \cdot recall}{b^2 \cdot precision + recall} \tag{45}$$

where TP is true positive, TN is true negative, FP is false positive, FN is false negative in classification, $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$ and $b = 1$.

The obtained results are shown in Table 10. Figure 10 shows the comparison of the evaluation metrics of the five algorithms on different datasets. On the Wine dataset, all five algorithms achieve satisfactory results. The reason lies in the simpler structure of the Wine dataset. Therefore, the different algorithms are able to achieve more accurate identification. In contrast, on the breast and Vowel datasets, several algorithms do not perform well due to the more complex structure of the clusters. Specifically, for AR, GBEHO achieves the best performance on Wine and CMC, and ranks 2nd, 3rd and 4th on breast, Vowel and heart, respectively. As for SP, GBEHO ranks 1st on Wine and CMC, and ranks 3rd, 3rd and 4th on breast, Vowel and heart datasets, respectively. For DR, GBEHO ranks first on Wine and CMC datasets, and 2nd, 2nd and 3rd on breast, Vowel and heart datasets, respectively. For F1, GBEHO ranks 2nd on Wine, breast and CMC datasets, and 4th on heart and Vowel. Overall, GBEHO performs the best on Wine, CMC. The performance on breast is located at 2nd, which is not as good as KIGSA-C. While for the heart dataset, CSOS performs the best, Hybrid FCM-PSO is second, and GBEHO is able to achieve a tie with KIGSA-C. On the Vowel dataset, GBEHO performs second only to KIGSA-C and ranks second. From the above analysis, it can be concluded that GBEHO's performance is competitive and convincing in the comparison of the state-of-art techniques.

Table 9 Parameter settings of the different algorithms

Algorithm	Parameter	Range
CSOS	N	30
	$Maxit$	500
Hybrid FCM-PSO	m	2
	ω	1
	c_1	2
	c_2	2
	N	30
	$Maxit$	500
ACLSHMS	k	5
	C	1
	N	30
	$Maxit$	500
KIGSA-C	G_0	100
	N	30
	$Maxit$	500

Legend:
N: The size of population, $Maxit$: Maximum number of iterations,
ω: Inertia weight, c_1: Cognitive coefficient,
c_2: Social coefficient, m: Fuzzifier constant,
k: Number of clusters in bid grouping, C: Moving coefficient, G_0: Gconstant

In general, it is proved that GBEHO provides a better choice of clustering. Therefore, GBEHO can be regarded as a powerful and effective clustering algorithm.

5 Discussions

Overall, the experimental results are consistent with the hypothesis. The introduction of the two operators and GBO improves the performance of the original EHO. Experiments on benchmark functions and datasets with different types prove that the improvement is significant. The proposed GBEHO is proven to have a higher clustering accuracy by evaluating four metrics, namely, accuracy rate, specificity, detection rate, and F-measure. Therefore, it can be concluded that GBEHO is an effective clustering method that can be used for a cluster analysis of different datasets.

Compared with other algorithms based on metaheuristics that are used for clustering, GBEHO shows a more competitive and superior performance and provides more desirable clustering results. GBEHO inherits all the advantages of traditional EHO, such as a superior global exploration capability. Meanwhile, the clan operator and separating operator in the original EHO are improved by the random wandering operator and mutation operator

Table 10 Results of GBEHO with $PSR = 0.2$ versus others state-of-the-art techniques

		Algorithm				
Dataset	Measure	GBEHO	CSOS	Hybrid FCM-PSO	ACLSHMS	KIGSA-C
Wine	AR	0.8263	0.7191	0.6707	0.8041	0.7866
	SP	0.7528	0.6895	0.6512	0.7352	0.7194
	DR	0.8536	0.7618	0.8098	0.8412	0.7985
	F1	0.8254	0.6575	0.7418	0.7882	0.8583
breast	AR	0.3588	0.3276	0.3012	0.3367	0.3693
	SP	0.2353	0.2112	0.2454	0.2336	0.3518
	DR	0.5637	0.5098	0.4839	0.5567	0.6147
	F1	0.4412	0.2786	0.3108	0.4148	0.4685
CMC	AR	0.5688	0.5174	0.5236	0.5472	0.5334
	SP	0.5712	0.5568	0.5384	0.5611	0.5598
	DR	0.4396	0.3748	0.3982	0.4157	0.4049
	F1	0.4936	0.3415	0.4165	0.4577	0.5103
heart	AR	0.6111	0.6457	0.6329	0.584	0.6382
	SP	0.5823	0.6212	0.6035	0.5426	0.6096
	DR	0.6533	0.7083	0.6726	0.6121	0.6231
	F1	0.6512	0.6788	0.6812	0.5987	0.6989
Vowel	AR	0.1875	0.1126	0.1667	0.2012	0.2379
	SP	0.1812	0.1297	0.1482	0.2139	0.2854
	DR	0.625	0.4631	0.5052	0.4667	0.6898
	F1	0.1664	0.1258	0.1895	0.1978	0.2512

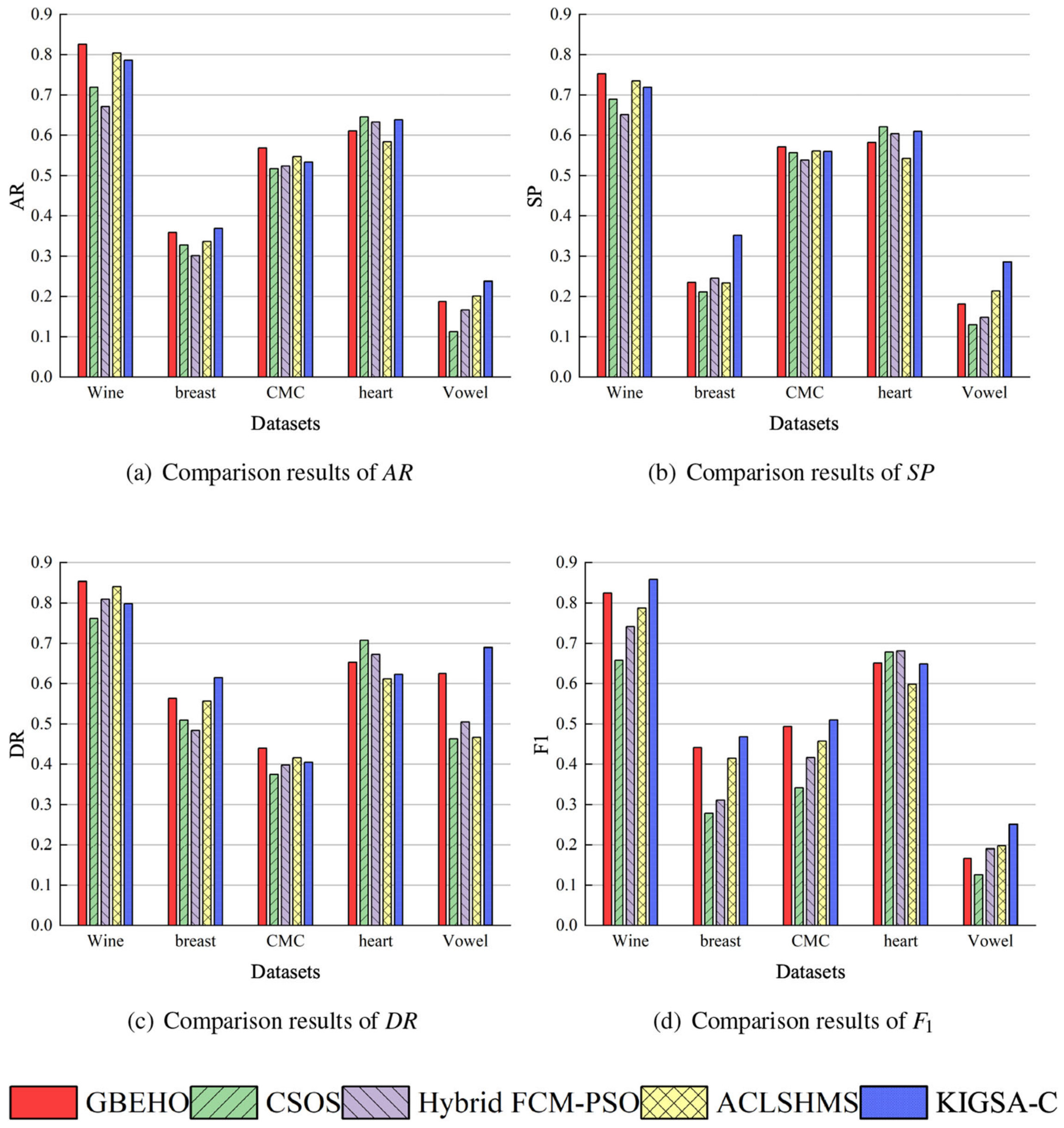


Fig. 10 Comparison results of 5 algorithms on different datasets

so GBEHO is better equipped with a stronger local exploitation compared with PSO, DE, GA, etc. Compared with BA and CS, it has a better exploration, and thus better avoids falling into the local optimum trap. In that case, the convergence rate is optimized. Moreover, GBEHO provides more accurate clustering results than the state-of-art algorithms. However, we observe that GBEHO is subject to several problems as follows. First, the time complexity of GBEHO is too high compared to other classical algorithms, which is caused by the newly added mechanism. The enhancement in clustering accuracy leads to an increase in the complexity of GBEHO. Second, with the increase in dimensionality, some of the metaheuristic algorithms suffer from a weakened stability. A scalability test with expandable dimensions is not performed, so the adaptability of GBEHO to multiple dimensions needs to be further examined. However, based on the No Free Lunch (NFL) theorem [51], there is no perfect optimization method, so we do not intend to claim that GBEHO is the best method in the world. The famous k-means algorithm has gained widespread use and attention since its inception, but it does not mean that k-means is without flaws. On the contrary, k-means is still limited to dependence on the initial solution and the tendency to fall into a local optimum. For our proposed method, we are more concerned with the accuracy of clustering rather than the time. As the research work goes further and becomes more detailed, the authors believe that there will be more techniques for improving operational efficiency in the future, such as parallel computing, which will provide better technical support for GBEHO.

6 Conclusions and future work

Traditional clustering methods easily fall into local optima, and the initialization of the center of mass position is a prominent problem. In this paper, an improved version of EHO is proposed for clustering analysis. Chaotic mapping based on Gaussian sequences improves the ergodicity and diversity of the initialized populations. Two operators, random wandering and mutation, are presented to optimize the strategy of updating positions in EHO, thus promoting the population diversity and the ability to jump out of the local optimum. Among them, the former improves the diversity of the population as well as the global exploration ability, and the latter promotes local exploitation at a later stage. In addition, GBO operators contribute to further balancing exploration and exploitation for the sake of determining the best center of mass more accurately. More suitable variable parameters are determined through ablation experiments.

Experiments on artificial and real-world datasets indicated that GBEHO has a better clustering performance than the other metaheuristic algorithms and their variants. The obtained intracluster variance was compared with classical k-means, PSO, DE, and GA algorithms to show superiority. By analyzing box plots and convergence curves, it was shown that GBEHO has a greater stability and faster convergence. The numerical data were confirmed by statistical analysis. Nonparametric tests were performed to verify significant differences between GBEHO and other algorithms. The visualization graphs of the clustering process demonstrated that GBEHO can find more accurate centroids at a faster iteration rate. Compared with the other state-of-art algorithms, GBEHO achieves more realistic results on accuracy rate, specificity, detection rate, and F-measure on five UCI datasets. Taken together, these results confirmed that GBEHO is an effective tool for data clustering.

In future research, we plan to reduce the time complexity of GBEHO through further design and experimentation. GBEHO can also be extended to several application areas, such as intrusion detection, image segmentation, and route planning. In addition, the performance of the hybrid algorithm will continue to be optimized to address sophisticated problems faced in practical engineering. The authors believe that this is an algorithm with great potential, and its application effect is worthy of expectation.

Declarations

Conflict of Interests The authors declare that they have no conflicts of interest.

References

1. Gambella C, Ghaddar B, Naoum-Sawaya J (2020) Optimization problems for machine learning: A survey. Eur J Oper Res 290(3):807–828
2. Zhou Y, Wu H, Luo Q, Abdel-Baset M (2019) Automatic data clustering using nature-inspired symbiotic organism search algorithm. Knowl-Based Syst 163:546–557
3. Zhang C, Hao L, Fan L (2019) Optimization and improvement of data mining algorithm based on efficient incremental kernel fuzzy clustering for large data. Clust Comput 22(2):3001–3010
4. Mousavirad SJ, Ebrahimpour-Komleh H, Schaefer G (2019) Effective image clustering based on human mental search. Appl Soft Comput 78:209–220
5. Maheshwari P, Sharma AK, Verma K (2021) Energy efficient cluster based routing protocol for wsn using butterfly optimization algorithm and ant colony optimization. Ad Hoc Netw 110:102317
6. Zhang J, Yu X, Xun Y, Zhang S, Qin X (2017) Scalable mining of contextual outliers using relevant subspace. IEEE Transactions on Systems, Man, and Cybernetics: Systems 50(3):988–1002
7. Maione C, Barbosa RM (2019) Recent applications of multivariate data analysis methods in the authentication of rice and the most analyzed parameters: A review. Critical reviews in food science and nutrition 59(12):1868–1879
8. Li H-J, Bu Z, Wang Z, Cao J (2019) Dynamical clustering in electronic commerce systems via optimization and leadership expansion. IEEE Transactions on Industrial Informatics 16(8):5327–5334
9. Huang D, Wang C-D, Wu J-S, Lai J-H, Kwoh C-K (2019) Ultra-scalable spectral clustering and ensemble clustering. IEEE Trans Knowl Data Eng 32(6):1212–1226
10. Saeed MM, Al Aghbari Z, Alsharidah M (2020) Big data clustering techniques based on spark: a literature review. PeerJ Computer Science 6:e321
11. Naouali S, Ben Salem S, Chtourou Z (2020) Clustering categorical data: A survey. International Journal of Information Technology & Decision Making 19(01):49–96
12. Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern recognition letters 31(8):651–666
13. Liu Y, Liu J, Jin Y, Li F, Zheng T (2020) An affinity propagation clustering based particle swarm optimizer for dynamic optimization. Knowl-Based Syst 195:105711
14. Bu Z, Li H-J, Zhang C, Cao J, Li A, Shi Y (2019) Graph k-means based on leader identification, dynamic game, and opinion dynamics. IEEE Trans Knowl Data Eng 32(7):1348–1361
15. Capó M, Pérez A, Lozano JA (2020) An efficient k-means clustering algorithm for tall data. Data mining and knowledge discovery 34(3):776–811
16. Tian K, Li J, Zeng J, Evans A, Zhang L (2019) Segmentation of tomato leaf images based on adaptive clustering number of k-means algorithm. Comput Electron Agric 165:104962
17. Bortoloti FD, de Oliveira E, Ciarelli PM (2021) Supervised kernel density estimation k-means. Expert Syst Appl 168:114350
18. Manochandar S, Punniyamoorthy M, Jeyachitra RK (2020) Development of new seed with modified validity measures for k-means clustering. Computers & Industrial Engineering 141:106290

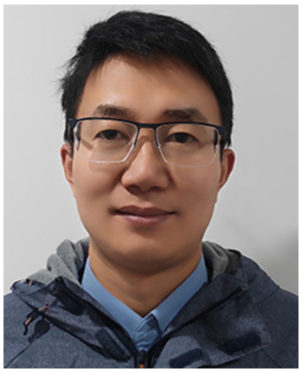
19. Ismkhan H (2018) Ik-means-+: An iterative clustering algorithm based on an enhanced version of the k-means. Pattern Recogn 79:402–413
20. Huang S, Kang Z, Xu Z, Liu Q (2021) Robust deep k-means: An effective and simple method for data clustering. Pattern Recogn 117:107996
21. Chowdhury K, Chaudhuri D, Pal AK (2021) An entropy-based initialization method of k-means clustering on the optimal number of clusters. Neural Comput & Applic 33(12):6965–6982
22. Zhao W-L, Deng C-H, Ngo C-W (2018) k-means: A revisit. Neurocomputing 291:195–206
23. Mahmoudi MR, Akbarzadeh H, Parvin H, Nejatian S, Rezaie V, Alinejad-Rokny H (2021) Consensus function based on cluster-wise two level clustering. Artif Intell Rev 54(1):639–665
24. Dutta D, Sil J, Dutta P (2019) Automatic clustering by multi-objective genetic algorithm with numeric and categorical features. Expert Syst Appl 137:357–379
25. Ezugwu AE, Shukla AK, Agbaje MB, Oyelade ON, Jose-Garcia A, Agushaka JO (2020) Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature. Neural Comput & Applic, pp 1–60
26. Zhu S, Xu L, Goodman ED (2020) Evolutionary multi-objective automatic clustering enhanced with quality metrics and ensemble strategy. Knowl-Based Syst 188:105018
27. Gupta S, Deep K, Mirjalili S (2020) An efficient equilibrium optimizer with mutation strategy for numerical optimization. Appl Soft Comput 96:106542
28. Wang G-G, Deb S, Cui Z (2019) Monarch butterfly optimization. Neural computing and applications 31(7):1995–2014
29. Li S, Chen H, Wang M, Heidari AA, Mirjalili S (2020) Slime mould algorithm: A new method for stochastic optimization. Futur Gener Comput Syst 111:300–323
30. Wang G-G (2018) Moth search algorithm: a bio-inspired meta-heuristic algorithm for global optimization problems. Memetic Computing 10(2):151–164
31. Yang Y, Chen H, Heidari AA, Gandomi AH (2021) Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. Expert Syst Appl 177:114864
32. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: Algorithm and applications. Future generation computer systems 97:849–872
33. Hussain K, Salleh MNM, Cheng S, Shi Y (2019) Metaheuristic research: a comprehensive survey. Artif Intell Rev 52(4):2191–2233
34. José-García A, Gómez-Flores W (2016) Automatic clustering using nature-inspired metaheuristics: A survey. Appl Soft Comput 41:192–213
35. Chen J, Qi X, Chen L, Chen F, Cheng G (2020) Quantum-inspired ant lion optimized hybrid k-means for cluster analysis and intrusion detection. Knowl-Based Syst 203:106167
36. Kuo RJ, Zheng YR, Nguyen TPQ (2021) Metaheuristic-based possibilistic fuzzy k-modes algorithms for categorical data clustering. Inf Sci 557:1–15
37. Nayak J, Naik B, Behera HS, Abraham A (2017) Hybrid chemical reaction based metaheuristic with fuzzy c-means algorithm for optimal cluster analysis. Expert Syst Appl 79:282–295
38. Aggarwal S, Singh P (2019) Cuckoo, bat and krill herd based k-means++ clustering algorithms. Clust Comput 22(6):14169–14180
39. Lakshmi K, Visalakshi NK, Shanthi S (2018) Data clustering using k-means based on crow search algorithm. Sādhanā 43(11):1–12
40. Yang C-L, Sutrisno H (2020) A clustering-based symbiotic organisms search algorithm for high-dimensional optimization problems. Appl Soft Comput 97:106722
41. Verma H, Verma D, Tiwari PK (2021) A population based hybrid fcm-pso algorithm for clustering analysis and segmentation of brain image. Expert Syst Appl 167:114121
42. Mousavirad SJ, Ebrahimpour-Komleh H, Schaefer G (2020) Automatic clustering using a local search-based human mental search algorithm for image segmentation. Appl Soft Comput 96:106604
43. Mittal H, Pandey AC, Pal R, Tripathi A (2021) A new clustering method for the diagnosis of covid19 using medical images. Appl Intell 51(5):2988–3011
44. Kuo R-J, Zulvia FE (2020) Multi-objective cluster analysis using a gradient evolution algorithm. Soft Comput 24(15):11545–11559
45. Wang G-G, Deb S, Gao X-Z, Coelho LDS (2016) A new meta-heuristic optimisation algorithm motivated by elephant herding behaviour. International Journal of Bio-Inspired Computation 8(6):394–409
46. Muthusamy H, Ravindran S, Yaacob S, Polat K (2021) An improved elephant herding optimization using sine–cosine mechanism and opposition based learning for global optimization problems. Expert Syst Appl 172:114607
47. Li W, Wang G-G, Alavi AH (2020) Learning-based elephant herding optimization algorithm for solving numerical optimization problems. Knowl-Based Syst 195:105675
48. Ismaeel AlaaAK, Elshaarawy IA, Houssein EH, Ismail FH, Hassanien AE (2019) Enhanced elephant herding optimization for global optimization. IEEE Access 7:34738–34752
49. Hakli H (2020) Bineho: a new binary variant based on elephant herding optimization algorithm. Neural Comput & Applic 32(22):16971–16991
50. Elhosseini MA, El Sehiemy RA, Rashwan YI, Gao XZ (2019) On the performance improvement of elephant herding optimization algorithm. Knowl-Based Syst 166:58–70
51. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE transactions on evolutionary computation 1(1):67–82
52. Dokeroglu T, Sevinc E, Kucukyilmaz T, Cosar A (2019) A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering 137:106040
53. Ahmadianfar I, Bozorg-Haddad O, Chu X (2020) Gradient-based optimizer: A new metaheuristic optimization algorithm. Inf Sci 540:131–159
54. Purushothaman R, Rajagopalan SP, Dhandapani G (2020) Hybridizing gray wolf optimization (gwo) with grasshopper optimization algorithm (goa) for text feature selection and clustering. Appl Soft Comput 96:106651
55. Saxena A, Prasad M, Gupta A, Bharill N, Patel OP, Tiwari A, Er MJ, Ding W, Lin C-T (2017) A review of clustering techniques and developments. Neurocomputing 267:664–681
56. Sokal RR (1966) Numerical taxonomy. Sci Am 215(6):106–117
57. Xu R, Wunsch D (2005) Survey of clustering algorithms. IEEE Transactions on neural networks 16(3):645–678
58. Pearson K, Lee A (1900) Mathematical contributions to the theory of evolution. viii. on the inheritance of characters not capable of exact quantitative measurement. part i. introductory. part ii. on the inheritance of coat-colour in horses. part iii. on the inheritance of eye-colour in man. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 195:79–150
59. Strehl A, Ghosh J, Mooney R (2000) Impact of similarity measures on web-page clustering. In: Workshop on artificial intelligence for web search (AAAI 2000), vol 58, p 64
60. Dice LR (1945) Measures of the amount of ecologic association between species. Ecol 26(3):297–302
61. Ramos-Guajardo AB, Ferraro MB (2020) A fuzzy clustering approach for fuzzy data based on a generalized distance. Fuzzy Sets Syst 389:29–50

62. Taib H, Bahreininejad A (2021) Data clustering using hybrid water cycle algorithm and a local pattern search method. Adv Eng Softw 153:102961
63. Bhadoria A, Marwaha S, Kamboj VK (2021) A solution to statistical and multidisciplinary design optimization problems using hgwo-sa algorithm. Neural Comput & Applic 33(8):3799–3824
64. Khalilpourazari S, Doulabi HH, Çiftçioğlu AO, Weber G-W (2021) Gradient-based grey wolf optimizer with gaussian walk: Application in modelling and prediction of the covid-19 pandemic. Expert Syst Appl, pp 114920
65. Hassan MH, Houssein EH, Mahdy MA, Kamel S (2021) An improved manta ray foraging optimizer for cost-effective emission dispatch problems. Eng Appl Artif Intell 100:104155
66. Singh NJ, Singh S, Chopra V, Aftab MA, Hussain SM, Ustun TS (2021) Chaotic evolutionary programming for an engineering optimization problem. Appl Sci 11(6):2717
67. Gandomi AH, Yang X-S (2014) Chaotic bat algorithm. Journal of Computational Science 5(2):224–232
68. James JQ, Lam AYS, Li VOK (2012) Real-coded chemical reaction optimization with different perturbation functions. In: 2012 IEEE Congress on Evolutionary Computation, IEEE, pp 1–8
69. Li W, Wang G-G (2021) Elephant herding optimization using dynamic topology and biogeography-based optimization based on learning for numerical optimization. Engineering with Computers, pp 1–29
70. Holland JH (1975) Adaptation in natural and artificial systems. ann arbor 18(3):529–530
71. Agbaje MB, Ezugwu AE, Els R (2019) Automatic data clustering using hybrid firefly particle swarm optimization algorithm. IEEE Access 7:184963–184984
72. Li W, Wang G-G (2021) Improved elephant herding optimization using opposition-based learning and k-means clustering to solve numerical optimization problems. Journal of Ambient Intelligence and Humanized Computing, pp 1–32
73. Yousri D, Mirjalili S, Machado JAT, Thanikanti SB, Fathy A et al (2021) Efficient fractional-order modified harris hawks optimizer for proton exchange membrane fuel cell modeling. Eng Appl Artif Intell 100:104193
74. Jia H, Sun K, Zhang W, Leng X (2021) An enhanced chimp optimization algorithm for continuous optimization domains. Complex & Intelligent Systems, pp 1–18
75. Fan Y, Shao J, Sun G, Shao X (2020) A modified salp swarm algorithm based on the perturbation weight for global optimization problems. Complexity, 2020
76. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4, IEEE, pp 1942–1948
77. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4):341–359
78. Yang X-S, Deb S (2009) Cuckoo search via lévy flights. In: 2009 World congress on nature & biologically inspired computing (NaBIC), Ieee, pp 210–214
79. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) Gsa: a gravitational search algorithm. Information sciences 179(13):2232–2248
80. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, pp 65–74
81. Aljarah I, Mafarja M, Heidari AA, Faris H, Mirjalili S (2020) Clustering analysis using a novel locality-informed grey wolf-inspired clustering approach. Knowl Inf Syst 62(2):507–539
82. Asuncion A, Newman D (2007) Uci machine learning repository. Irvine, CA, USA
83. Duan Y, Liu C, Li S (2021) Battlefield target grouping by a hybridization of an improved whale optimization algorithm and affinity propagation. IEEE Access 9:46448–46461
84. Tu J, Chen H, Liu J, Heidari AA, Zhang X, Wang M, Ruby R, Pham Q-V (2021) Evolutionary biogeography-based whale optimization methods with communication structure: towards measuring the balance. Knowl-Based Syst 212:106642
85. Ouaar F, Boudjemaa R (2021) Modified salp swarm algorithm for global optimisation. Neural Comput & Applic, pp 1–26
86. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information sciences 180(10):2044–2064
87. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1(1):3–18

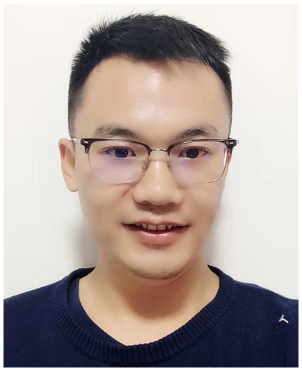
Yuxian Duan was born in Weihai, Shandong, China, in 1992. He is currently pursuing the M.S. degree with the College of Air and Missile Defense, Air Force Engineering University. His research interests include command and control system, situation awareness, and intelligent information processing.

Changyun Liu was born in 1973. He received the B.S. and M.S. degrees from Air Force Engineering University (AFEU), in 1996 and 2002, respectively, and the Ph.D. degree in information and communication engineering from Xidian University, in 2014. He is currently a Professor with the College of Air and Missile Defense, AFEU. His research interests include signal processing, target tracking, and time-frequency analysis.

Song Li was born in Anhui, China, in 1977. He received the Ph.D. degree from Air Force Engineering University (AFEU). He is currently an Associate Professor with the College of Air and Missile Defense, AFEU. His main research interests include command automation information processing and air defense auxiliary decision.



Xiangke Guo was born in 1980. He received the Ph.D. degree from Beijing University of Aeronautics and Astronautics (BUAA) in 2010. He is currently an associate professor at Air Force Engineering University. His research focuses on data mining and intelligent information processing.



Chunlin Yang is currently pursuing the M.S. degree with the College of Air Traffic Control and Navigation, Air Force Engineering University. His research focuses on intelligent information processing.