Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 7796696, 19 pages
https://doi.org/10.1155/2021/7796696

Research Article

A Feature Selection Method by using Chaotic Cuckoo Search Optimization Algorithm with Elitist Preservation and Uniform Mutation for Data Classification

Le Wang,[1] Yuelin Gao,[2] Jiahang Li,[1] and Xiaofeng Wang[1]

[1]*School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China*
[2]*Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, Yinchuan 750021, China*

Correspondence should be addressed to Yuelin Gao; gaoyuelin@263.net

Received 26 April 2021; Revised 14 May 2021; Accepted 8 June 2021; Published 21 June 2021

Academic Editor: Rui Wang

Feature selection is an essential step in the preprocessing of data in pattern recognition and data mining. Nowadays, the feature selection problem as an optimization problem can be solved with nature-inspired algorithm. In this paper, we propose an efficient feature selection method based on the cuckoo search algorithm called CBCSEM. The proposed method avoids the premature convergence of traditional methods and the tendency to fall into local optima, and this efficient method is attributed to three aspects. Firstly, the chaotic map increases the diversity of the initialization of the algorithm and lays the foundation for its convergence. Then, the proposed two-population elite preservation strategy can find the attractive one of each generation and preserve it. Finally, Lévy flight is developed to update the position of a cuckoo, and the proposed uniform mutation strategy avoids the trouble that the search space is too large for the convergence of the algorithm due to Lévy flight and improves the algorithm exploitation ability. The experimental results on several real UCI datasets show that the proposed method is competitive in comparison with other feature selection algorithms.

1. Introduction

Data processing and data mining has become a significant area of research for academics, and how to process data is a very complex and challenging task. Datasets often have many attributes and features, and when using the data, not every feature is helpful for the dataset, and some features are redundant and irrelevant. In the data processing phase, features are processed in two ways: feature selection and feature extraction [1]. The feature extraction method maps high-dimensional features to a low-dimensional space, reducing the time required for model training; feature selection differs from feature extraction in that it does not spatialize the features but selects some of the features in a more extensive feature set for training. In contrast to feature extraction, feature selection is a quantitative solution to the problem of too many features. The feature selection problem is a very challenging task in the field of machine learning. In machine learning classification tasks, it is necessary to select a subset of features that can make the classifier accurate and efficient at the same time.

Feature selection as the first step in data preprocessing can be divided into three types: filter, wrapper, and embedded [2]. For the first type, the use of information processing methods such as information gain, information entropy, Pareto analysis, T-tests, and mutual information has been used to solve feature selection problems [3–5], where the principle is to use correlation between features and attributes to select the subset of features with the strongest correlation. The wrapper approach consists of two phases: the feature selection phase, and the other is the training phase of the machine learning classifier. The selection of feature subsets depends on the classification algorithm, and the feature subsets that are selected have a higher accuracy of the classifier. Regarding the last type, the embedded approach tries to combine the abovementioned

two types. The feature selection method proposed in this paper is based on the wrapper approach.

The wrapper method uses learner performance as the evaluation criterion for a subset of features. Traditional wrapper methods are based on improvements of machine learning classifiers, greedy search, and other methods [6], but these methods tend to fall into local optima and are computationally expensive. Nowadays, nature-inspired algorithms are widely used to solve optimization problems, which are inspired by nature and are increasingly used by scholars to find the optimal value in the problem due to their conceptual simplicity and ease of implementation [7]. A large number of contributions have been made using nature-inspired algorithms to solve feature selection problems. These approaches include genetic algorithms [8–10], particle swarm algorithms [11–14], artificial bee colony algorithms [15], grey wolf optimization algorithms [16], brainstorming algorithms [17], firefly algorithm [18], bat algorithms [19], and butterfly algorithms [20]. Nature-inspired algorithms are committed to the study of exploration and exploitation capabilities. In fact, the two strategies of efficiency and effectiveness are contradictory in solving the feature selection problem [21], and the search operator in a single algorithm is slightly thin in solving the problem, so a multistrategy multioperator hybrid algorithm is proposed. In 2015, Ali [22] combined the cuckoo search algorithm with a genetic algorithm. In 2017, Mafarja et al. [23] mixed the whale optimization algorithm with the cooling annealing strategy in the simulated annealing algorithm, which accepts suboptimal solutions with a certain probability, thereby improving the local search capability of the whale algorithm. Elgamal et al. [24] and Abdel-Basset et al. [25] similarly added the simulated annealing cooling process to the Harris Hawk optimizer. In 2019, Moslehi and Haeri [26] mixed filter and wrapper feature selection methods while allowing genetic algorithms and particle swarm algorithms to update the population, increasing the diversity of the population. In 2020, Tubishat et al. [27] improved the salp swarm algorithm by incorporating a reversal science strategy and combining it with a local search algorithm to solve the feature selection problem.

In 2009, Yang and Deb [28] first proposed a cuckoo search algorithm, which describes the parasitic behavior of cuckoos in nature and generalizes this behavior into an optimization algorithm. Yang's team applied this algorithm to the engineering optimization problem. Phogat et al. [29] combined mutual information with cuckoo to classify complex diseases. In 2021, the adaptive cuckoo search algorithm was used by Salgotra et al. [30]. In 2010, Lin and Ramli [31] first proposed a discrete form of the cuckoo search algorithm. Shirin et al. [32] proposed a binary cuckoo search algorithm to solve the discrete 0-1 backpack problem. Later, the discretized cuckoo search algorithm was used to solve the feature selection problem [33]. Hamidzadeh et al. [34] incorporated opposite learning and destruction operators into the cuckoo search algorithm to solve the feature selection problem, but this algorithmic strategy is too tedious and ignores the complexity of the calculation. Sadegh Salesi and Cosma [35] proposed a new cuckoo search algorithm for solving discrete feature selection problems, and the drawback of the algorithm is that the randomness of the proposed strategy is too great, and the strategy is not conducive to the convergence of the algorithm.

The cuckoo search algorithm is used to solve all kinds of optimization problems due to its simple structure and easy-to-implement parameters, but less research work has been carried out on the feature selection problem, so there is a lot of room for exploration. The current cuckoo search has the following limitations when tackling feature selection problems:

(1) Nature-inspired algorithms need to be initialized when they iterate to find the optimal solution to solve continuous or discrete problems, which means that the initial populations are dependent on subsequent algorithms. An excellent initial population plays a role in the convergence and iteration of the algorithm. Binary cuckoo algorithms initialized too randomly and blindly primarily affect the preservation of quality features, and the diversity of the population is not guaranteed.

(2) The heavy-tailed distribution of the Lévy flight means that great values can be taken with a significant probability. The iterations of the formula depend on two standard random numbers, and the random numbers cause the cuckoo to search for a path on each flight randomly that can be large or small or positive or negative, so it is easy to jump from one region to another, jumping at local locations with significant probability, jumping out of the local optimum, and thus, expanding the search. This updated path of the Lévy flight would theoretically prevent the algorithm from converging to the optimum.

(3) Sadegh Salesi and Cosma [35] proposed a new cuckoo search algorithm for solving discrete feature selection problems, which adds a pseudoneighborhood mutation strategy to the cuckoo search algorithm, in which individuals are randomly selected to mutate the feature sequences during the iteration of the algorithm. However, the number of random variations is too large, which is likely to cause good individuals to be mutated and deficient individuals to be retained to the next generation instead. This cycle increases the computational effort and is not conducive to the convergence of the algorithm.

To address the abovementioned algorithms' shortcomings, we propose a new multistrategy integration cuckoo search algorithm to improve the performance of the cuckoo algorithm in solving the feature selection problem in this paper. The main contributions are as follows:

(1) A new feature selection method is proposed based on the cuckoo search algorithm (CBCSEM). The population is initialized using different chaotic maps to ensure the diversity of the population, and uniformly initialized individuals also form the basis for the convergence of the algorithm.

(2) To address the drawbacks of the overly random nature of the strategy proposed in literature [36], in this paper, a two-population elite preservation strategy is proposed. Firstly, the two populations are initialized to calculate their individual fitness, unlike the random selection of population individuals strategy, where individuals with high fitness are directly retained into the next generation. Secondly, the more deficient individuals within the two populations are retained, and these individuals are used as the candidate set for performing the mutation operation. The uniform mutation strategy used in this paper differs from random threshold mutation in that, on the one hand, the operation can reduce the randomness; on the other hand, avoiding the large search space caused by Levy flight strategy, the algorithm cannot reach the state of convergence.

(3) The CBCSEM is defined as a wrapped supervised feature selection method that aims to reduce the number of features selected in order to increase the accuracy of the classifier, and a comparison with five recent algorithms is proposed to solve the feature selection problem shows that the CBCSEM ranks first.

The rest of this paper is organized as follows. Section 2 details the theory and methods mentioned in this paper, and the proposed feature selection method is provided in Section 3. In Section 4, we evaluate the performance of the strategies on real data and compare the algorithm proposed in this paper with the remaining five feature selection methods. The last section summarizes the paper and gives an outlook.

2. Theory and Method

In this part, Section 2.1 introduces the feature selection problem, Section 2.2 briefly introduces the chaotic function, Section 2.3 introduces the cuckoo search algorithm, Section 2.4 introduces the Lévy flight, and Section 2.5 introduces the elitist preservation.

2.1. Feature Selection. If a dataset contains n features, it will have 2^n feature subset selection. When the value of n is large enough, how to select a subset of these feature combinations that make the machine model training more efficient is the primary problem to solve, so the feature selection problem gradually evolves into a class of optimization problems.

Let M be a dataset of K samples with D features, assuming that the full set F of features contains D-dimensional features. Feature selection is to select d-dimensional subset f of features from the full set F of features, where $d \le D$, $f \le F$. The objective function, $f(X)$, i.e., classification accuracy. The purpose of feature selection is to minimize the number of features, when the classifier performs best (at this point, $f(X)$ is the optimal solution).

The feature selection problem differs from traditional optimization problems. It is identified as a discrete binary problem where the search space is an n-dimensional lattice space of Boolean type, and the solution to feature selection is to display and update at each corner of the hypercube [35].

$$X = (x_1, x_2 \ldots x_i..x_D), \quad x_i \in \{0, 1\}, \tag{1}$$

where $x_i = 1$ represents the jth feature is selected into the feature subset X, whereas $x_i = 0$ means this feature is not selected.

Thus, the feature selection problem can be formulated as the following optimization problem:

$$\begin{cases} \max & f(X) \\ \text{s.t.} & X = (x_1, x_2, \ldots . x_D), \quad x_i \in \{0, 1\}, j = 1, 2, 3, \ldots, D \\ & 1 \le \{X\} \le D, \end{cases} \tag{2}$$

where $\{X\}$ represents the number of the features in X, i.e., a subset of features.

2.2. Chaotic Function. Chaotic maps are generally used to generate chaotic sequences and random sequences generated by a simple deterministic system that is nonlinear, ergodic, stochastic, and overall stable locally unstable [36]. Chaos expresses the amount of initial state of a nonlinear system, and even slight differences in the initial state of the system can lead to different state development changes. It is based on such a theory that chaotic maps generated by chaotic systems can be used as random number generators to generate chaotic numbers between 0 and 1, which can be used as initial populations in optimization algorithms. In this paper, six of the 12 chaotic map functions are selected for experimentation. The maps range of all six mapping functions is between 0 and 1, and the chaotic maps are shown in Table 1.

2.3. Cuckoo Search Algorithm. Cuckoo search (CS) is an algorithm proposed by Yang and Deb in 2009 [28]. The algorithm forms a biologically inspired heuristic search algorithm based on a summary of the parasitic and reproductive behavior of cuckoos in nature. For successive optimization problems, we can define each egg in the nest as a solution, then the cuckoo egg is defined as a completely new solution, and the new solution is used to replace the not-so-good solution in the nest when solving the specific optimization problem. In traditional CS algorithms, both the cuckoo's egg and the host's nest can be used as solutions to the CS algorithm. However, in the binary cuckoo algorithm, the host nest is defined as an individual of the population, and the nest allows the cuckoo to place one or even more eggs. As the number of iterations increases, the nests with high fitness values are retained, which means that the good eggs are retained later in the iteration of the algorithm, and thus, the good features are retained.

The continuous CS algorithm mapping to binary space relies on the update formula of the binary particle swarm algorithm [37], which uses a sigmoid function to map vectors in continuous space into two dimensions, with the following mapping formula:

Table 1: The six chaotic maps used in this paper.

aNo.	Chaotic map	Definition	Range
1	Logistic	$X_{i+1} = aX_i(1 - X_i)$	(0, 1)
2	Singer	$X_{i+1} = \mu(7.86X_i - 23.31X_i^2 + 28.75X_i^3 - 13.302875X_i^4)\mu = 1.07$	(0, 1)
3	Sinusoidal	$X_{i+1} = aX_i^2 \sin(\pi X_i)\, a = 2.3$	(0, 1)
4	Iterative	$X_{i+1} = \sin(a\pi/X_i)\, a = 4$	(−1, 1)
5	Sine	$X_{i+1} = (a/4)\sin(\pi X_i)\, a = 4$	(0, 1)
6	Gauss/mouse	$X_{i+1} = \begin{cases} 1 & X_i = 0 \\ (1/\mathrm{mod}(X_i, 1)) & \text{otherwise} \end{cases}$	(0, 1)

$$S\left(X_{ij}^t\right) = \frac{1}{1 + e^{-X_{ij}^t}}, \tag{3}$$

$$X_{ij}^{t+1} = \begin{cases} 1, & \text{if } S\left(X_{ij}^t\right) > \sigma, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where σ is a random number between $[0, 1]$, X_{ij}^t is the j dimension of the i nest in the t generation, where $i = 1, 2, 3, \ldots, n$, $j = 1, 2, 3, \ldots, D$, and X_{ij}^{t+1} is the j dimensional feature of the i nest at generation $t + 1$.

2.4. Lévy Flight. Lévy flight is used to characterize objects whose steps obey a heavy-tailed distribution when they perform random wanderings. Dr. Yang, the proposer of the CS algorithm, used in the specific implementation of CS the formula proposed by Mantegna in 1994 [38] for modeling Lévy flight jump paths.

$$s = \frac{\mu}{|\nu|^{1/\lambda}}, \tag{5}$$

where s is the Lévy flight jump path and the parameters μ, ν are normally distributed random numbers that obey a normal distribution, and the corresponding standard deviation σ_μ, σ_ν of the normal distribution takes the following values:

$$\begin{cases} \mu \sim N\left(0, \sigma_\mu^2\right), \\ \nu \sim N\left(0, \sigma_\nu^2\right), \end{cases}$$

$$\begin{cases} \sigma_\mu = \left\{ \dfrac{\Gamma(1 + \lambda)\sin(\pi\lambda/2)}{\Gamma(\lambda/2)2^{\lambda/2}} \right\}^{1/\lambda}, \\ \sigma_\nu = 1. \end{cases} \tag{6}$$

2.5. Elitist Preservation. Genetic algorithms, one of the classical algorithms in heuristics, is a biologically inspired learning method that was proposed by Holland in 1975 [39]. The standard genetic algorithm simulates the evolutionary process of organisms in nature. The most crucial problem of genetic algorithms is to solve the convergence problem of the optimal global solution. Rudolph used Markov chain theory to prove that standard genetic algorithms cannot converge to the optimal global solution by selection, crossover, and mutation operators only. Simple crossover mutation will, to some extent, lead to the destruction of good gene combinations, so individuals with high fitness values do not need to undergo crossover mutation operation and are directly retained to the next generation, which means that the locally optimal solution is most likely to be the optimal global solution, which is the elitist preservation of genetic algorithms proposed by Zhou and Sun in his Ph.D. thesis [40]. This is described as [40] Algorithm 1.

From the abovementioned pseudocode, the population iterates to generation t, the best individual in the population is $a(t)$, and $A(t + 1)$ is the new generation population. If there are no individuals in the new population that are better than individual $a(t)$, then individual $a(t)$ is added to $A(t + 1)$ as $n + 1$ individuals in $A(t + 1)$ or the worst individual in $A(t + 1)$ is replaced in order to keep the size of the population unchanged.

3. Proposed Cuckoo Search Algorithm with Chaotic Function, Lévy Flight, and Elitist Preservation (CBCSEM)

3.1. The Improved Cuckoo Search Algorithm. In this paper, a binary chaotic cuckoo search algorithm that mixes Lévy flight and new elitist preservation and mutation strategies are proposed.

The components of the new cuckoo search algorithm (CBCSEM) are as follows.

3.1.1. Cuckoo Nest. In the traditional cuckoo search algorithm, the cuckoo is updated at any position in the space, which is called the continuous space, while in solving the feature selection problem, the solution of the problem is restricted to the interval $[0, 1]$. For a population of size N, the number of cuckoos in the population is N, and the attributes carried by each individual are M. This means that the search range for each individual is an $N * M$ matrix. A schematic diagram of how the binary cuckoo search algorithm is encoded to solve the feature selection problem is shown in Figure 1. As shown above, the initialization in each nest generates a different binary string, with each bit representing a different feature, where a 1 for that bit means that the feature corresponding to that bit is selected and a 0 for that bit means that the feature is not selected.

(1) Initialize a population of size n, $P(t) = \{a_1(t), a_2(t), \ldots, a_n(t)\}$
(2) $f_{\max}(t) = \max\{f(a_1(t)), f(a_2(t)), \ldots, f(a_n(t))\}$
(3) $f_{\max}(t+1) = \max\{f(a_1(t+1)), f(a_2(t+1)), \ldots, f(a_n(t+1))\}$
(4) **if** $f_{\max}(t) > f_{\max}(t+1)$ **then**
(5) replicate $\{a'_k(t)\} = \{f(a_k(t)) > f_{\max}(t+1), a_k(t) \in P(t)\}$
(6) $k = 1$ or $k = 1, 2, 3, \ldots, K$
(7) **if** $f_{\max}(t) > f_{\max}(t+1)$ **then**
(8) replace randomly $\{a_j(t+1) \in P(t+1)\}$ with $\{a'_k(t)\}$
(9) $j = 1$ or $j = 1, 2, 3, \ldots, K$
(10) **else**
(11) replace the worst ones of $\{a_j(t+1) \in P(t+1)\}$ with $\{a'_k(t)\}$
(12) **end**
(13) **end**

Algorithm 1: Elitist preservation.

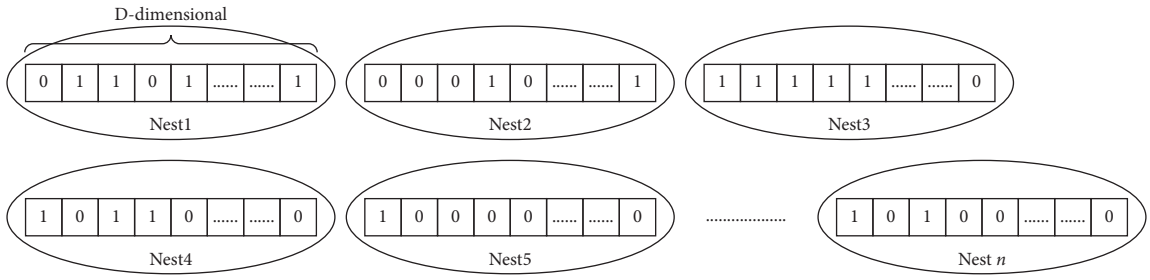

Figure 1: Feature selection problem coding method.

3.1.2. Chaotic Map. Chaos theory was first proposed to study atmospheric flow patterns, revealing that the chaotic state of chaos is somewhat structured. Small changes in initial test conditions can produce significant changes in subsequent behavior. Therefore, this paper incorporates a chaotic map in the initialization phase to provide a basis for convergence of the algorithm on the one hand and to increase the diversity of the population during the initialization phase on the other. All the chaotic maps used in this paper are shown in Table 1, and the logistic function is described as follows:

$$x_i = \text{rand}(0, 1), \tag{7}$$

$$x_{i+1} = a x_i (1 - x_i). \tag{8}$$

In the abovementioned equation, x_i is a random number, and the sequence is fully chaotic when the value of $a = 4$.

3.1.3. Lévy Flight. In order to better find a suitable nest in the next generation, the cuckoo needs to search a large area and then find an optimal solution. The Lévy flight size step search is applied to the CBCSEM algorithm to increase its search range within the effective range and improve the global search capability of the algorithm. The step size is calculated as follows:

$$\text{step} = \alpha \times \text{Levy}(\lambda) = \alpha \times s, \tag{9}$$

where step is the cuckoo's path in the solution space from the last old nest location to the random search for a new nest location. At each iteration of the algorithm, the Lévy flight formula and the Lévy flight-based nest position update formula are shown as follows:

$$X_{ij}^{(t+1)} = X_{ij}^{t} + \alpha \oplus \text{Levy}(\lambda). \tag{10}$$

The formula shows that X_{ij}^{t} is the value of the ith individual in the jth dimension in the tth generation, $X_{ij}^{(t+1)}$ is the individual resulting from the Lévy flight update, and α is a constant representing the step scaling factor. The value of α in the CBCSEM is 1 to ensure that the step size of the algorithm is not overstepped.

3.1.4. Uniform Mutation. A good optimization algorithm needs to take into account both exploration and exploitation capabilities. Lévy flight determines the optimal solution area of the algorithm. We need to find a new strategy to find the optimal solution or close to the optimal solution in this area. In 2017, Salesi and Cosma [35] proposed a random mutation strategy, and the mutation formula is as follows:

$$\begin{aligned} X_i^j &\in [0, 1], \\ \sigma &\in [0, 1], \\ X_i^j &= \begin{cases} 0, & X_i^j \geq \sigma, \\ 1, & \text{otherwise.} \end{cases} \end{aligned} \tag{11}$$

It can be seen from the above mentioned formula that both individual and threshold are randomly generated, and the individuals in the bird's nest have both all possible mutations

and zero mutations. This is the drawback of the mutation operation of the algorithm. In order to avoid this drawback, this paper proposes the uniform mutation operation performing the uniform mutation method. The schematic diagram of the transformation is shown in Figure 2. It is seen from this figure, after the binary encoding of the feature sequence within the nest, a 0-1 transformation to achieve a complete update of this sequence, avoiding the transformation of individual randomness due to the influence of random thresholds. Algorithm 2 is the pseudocode of mutation operation.

3.1.5. New Elitist Preservation. In addition to the decision to mutate or not based on the threshold size, Sadegh Salesi also proposed a random selection of individuals to perform mutation operations, which is mainly capable of changing the structure of the population. However, the variance of random selection is too large and faces the risk of good individuals undergoing mutation and more deficient individuals being retained. Section 2.5 of the article introduces the elite preservation strategy in genetic algorithms. The CBCSEM improves on this by introducing a two-population strategy, where after initializing two populations, the fitness value of each population is calculated, the worse individual in both populations is taken to perform the mutation operation described above. The fitness value of the individual is then calculated after mutation to replace the worse individual in the original population. Algorithm 3 is a pseudocode for the new elitist preservation. Lines 1 to 4 initialize the two populations and calculate the fitness of each population. To find the worst individual in each population for the next mutation operation, line 5 to line 10 traverse the two populations to obtain their minimum values, and the minimum values of the two populations are then compared and stored in an array for the next operation.

3.1.6. Objective Function. Using a wrapper approach to algorithmically optimized feature selection, the accuracy of the classifier becomes the criterion by which the best individuals in each generation are judged. In the proposed CBCSEM, the Random Forest algorithm becomes the evaluation algorithm for fitness. The data column and label column in the data set are taken out, respectively, to form the data matrix and the label matrix and find the dimension index with the value of 1 in each bird's nest and store it in the feature_set. We use this small sample to map the key position of the total sample, use the Random Forest classifier model to predict the sub_samples label to get the predict_label to compare with the actual_label, and return an ACC, which is the fitness of the algorithm. The value of ACC in the CBCSEM is determined by the Random Forest integrated classifier, and the number of decision trees is specified as 10, described as follows:

$$\begin{aligned} \mathrm{ACC} = {} & \text{Random Forest Classifier}(n\_\text{estimators} \\ & = 10, \text{random\_state} = 0). \end{aligned} \tag{12}$$

Algorithm 4 is the pseudocode for CBCSEM. Figure 3 shows the internal selection mechanism of the CBCSEM, from which it can be seen that features can be real-number or binary encoding, the difference between the two is whether or not a binary mapping is required, and the features encoded in real are converted to binary and then optimized by the algorithm to select the features corresponding to the binary encoding of 1. Figure 4 shows the framework flow of the optimization problem using the CBCSEM.

3.1.7. Computational Complexity. The computational complexity of the algorithm depends on several components, including the cuckoo algorithm (CS), chaotic maps (CM), Lévy flights (LF), elite preservation strategies (EP), and mutation (M). Therefore, the calculation of CBCSEM complexity is given by the following equations:

$$\begin{aligned} O(\mathrm{CS}) &= O(t * (\dim \times N + C \times N + N \log N)), \\ O(\mathrm{CM}) &= O(t \times N), \\ O(\mathrm{LF}) &= O(t * (\dim \times N + C \times N)), \\ O(\mathrm{EP}) &= O(2(t * N)), \\ O(M) &= O(\dim \times N), \\ O(\mathrm{CBCSEM}) &= N \times O(\mathrm{CS}) + N \times O(\mathrm{CM}) + N \times O(\mathrm{LF}) + \mathrm{Sub}\, N \times O(\mathrm{EP}) + O(M). \end{aligned} \tag{13}$$

In the abovementioned equation, t is the number of iterations of the algorithm, dim is the feature dimension, N is the population size, sub N is the number of subpopulations selected by the feature, and C is the objective function cost. Combining the pseudocode of Algorithm 4, we can see that the cuckoo search optimization runs throughout the algorithm, both in the initialization phase and in the main loop, so that $O(\mathrm{CS})$ includes steps completed in linear time as well as in nonlinear time [34]. Lines 4 to 6 of Algorithm 4 initialize the population with chaotic maps, whose computational complexity is determined by the population size N; lines 13 to 15 of the algorithm perform the Lévy flight, a part of the algorithm whose complexity depends not only on the population size N but likewise on the dimensionality dim of the features carried by the individuals, whose objective function cost C is computed for each generation of

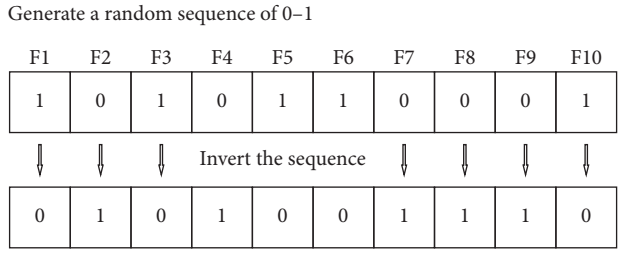

Figure 2: Example of uniform mutation.

Input: mut_pop1, mut_pop2
Output: Population 1 after mutation replacement, Population 2 after mutation replacement
(1) **for** *each nest i in* mut_pop 1 **do**
(2) **for** *j in nest_i* **do**
(3) *a* = random [0,1, size = feature_num] **if** *j* = = *0* **then**
(4) nest[*j*] = 1−a[*j*]
(5) **end**
(6) new_nest = [ ], cur = 0 **for** *j in nest_i* **do**
(7) **if** *j* = = *1* **then**
(8) new_nest.append(cur)
(9) **end**
(10) **end**
(11) **end**
(12) **return** *nest, new_nest*
(13) **end**
(14) Calculate the mut_fitness
(15) mut_pop 2 also performs the above operation
(16) **return** *mut_fitness,nest, new_nest*

Algorithm 2: Pseudocode of mutation.

Input: number of dimensions(features) *d*, Lb, Ub, Boolean Boundary sigma
Output: The poorer individual of the two populations: mut_pop1, mut_pop2
(1) pop 1 = $\{a_1(t), a_2(t), a_3(t), \ldots, a_n(t)\}$
(2) pop 2 = $\{b_1(t), b_2(t), b_3(t), \ldots, b_n(t)\}$
(3) $f_{\text{pop}\,1}(t) = \{f(a_1(t)), f(a_2(t)), f(a_3(t)), \ldots, f(a_n(t))\}$
(4) $f_{\text{pop}\,2}(t) = \{f(b_1(t)), f(b_2(t)), f(b_3(t)), \ldots, f(b_n(t))\}$
(5) **if** $f_{\min 1}(t) = \{\{a'_k(t)\} | a'_k(t) \in \text{pop}\,1\}$ **then**
(6) save $f_{\min 1}(t)$
(7) **end**
(8) **if** $f_{\min 2}(t) = \{\{b'_k(t)\} | b'_k(t) \in \text{pop}\,2\}$ **then**
(9) save $f_{\min 2}(t)$
(10) **end**
(11) **if** $f_{\min 1}(t) < f_{\min 2}(t)$ **then**
(12) mut_pop1[]++
(13) **end**
(14) **else**
(15) mut_pop2[] ++
(16) **end**
(17) **return** mut_pop 1, mut_pop 2

Algorithm 3: Pseudocode of new elitist preservation.

Input: labelled data, max_iterations or stop criteria, number of nests(m), number of dimensions(features) d, CS parameters, Lb, Ub, classifer models.
Output: Best fitness, best features.
(1) n_nests = 50, n_features = dimensions(d), nest_fitness[] = −10,
(2) Initialize two populations performs the following operations:
(3) **for** each nest in population **do**
(4) **for** each dimension in one nest **do**
(5) Randomly assign values to two populations of individuals based on [Lb, Ub]. Updating populations based on chaotic maps equations (7) and (8)
(6) **end**
(7) Convert two populations to binary using equations (3) and (4)
(8) Calculate the fitness for individuals of two populations by equation (12)
(9) Update the nest_fitness
(10) **end**
(11) Both populations perform Algorithm 3
(12) **while** $t \leq$ *max_iterations or stop criteria* **do**
(13) **for** *each nest in populations* **do**
(14) perform levy flights to generate new populations use equation (9) and (10)
(15) **end**
(16) **for** *each nest in populations* **do**
(17) ▷ **Rectification procedure**
(18) Normalize the population value generated by Lévy flight
(19) Binary conversion
(20) **end**
(21) Then execute Algorithm 2
(22) **end**
(23) **if** *nest_fitness* < *mut_fitness* **then**
(24) repalce the nest with mut_pop(i)
(25) **end**
(26) **return** *mut_fitness*

Algorithm 4: Pseudocode of CBCSEM based on feature selection.

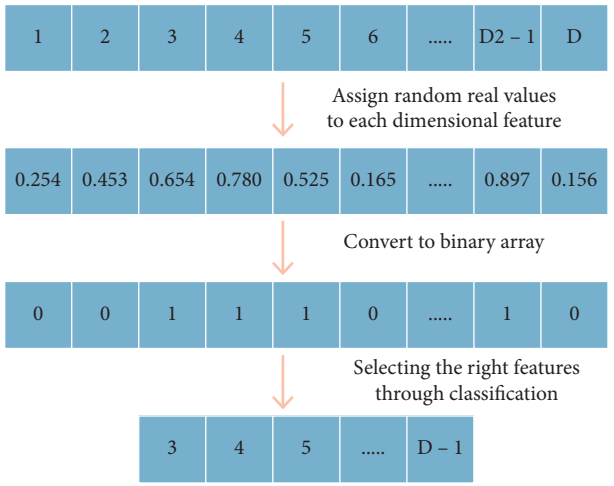

Figure 3: Internal mechanism for feature selection of CBCSEM.

individuals. Next, we consider Algorithms 2 and 3, which propose that the new elite preservation strategy considers two populations, so that $O(\mathrm{EP})$ is twice the product of the number of iterations t and the number of individuals N. The uniform mutation changes the value of each feature bit of an individual, and the computational complexity of this part is determined by the feature dimension dim and the number of individuals N. The computational complexity of CBCSEM shows that the complexity of all the strategies proposed in this paper can be accomplished in linear time, even though there is an elite preservation strategy with two-population competition, but it does not have a catastrophic impact on the complexity, and the number of individuals in the population and the feature dimension are the main factors affecting the computational complexity.

4. Experimental

In this section, we conduct comparative experiments on the algorithms, which consist of different classification datasets compared to the algorithm proposed in this paper and other algorithms. First, Section 4.1 gives a brief introduction to the dataset; Section 4.2 presents the experimental setup and discussion of the parameters; and finally, Section 4.3 presents a methodological evaluation of the performance of the proposed method relative to other methods.

4.1. Datasets. The dataset for solving Android malware classification in this paper is divided into two parts: one is obtained by decompiling, with a total of 2720 samples, 532 malicious application samples from the Canadian Institute of Network Security [41], which contain typical Android platform malware, such as ransomware malicious applications, threatening SMS applications, and advertising

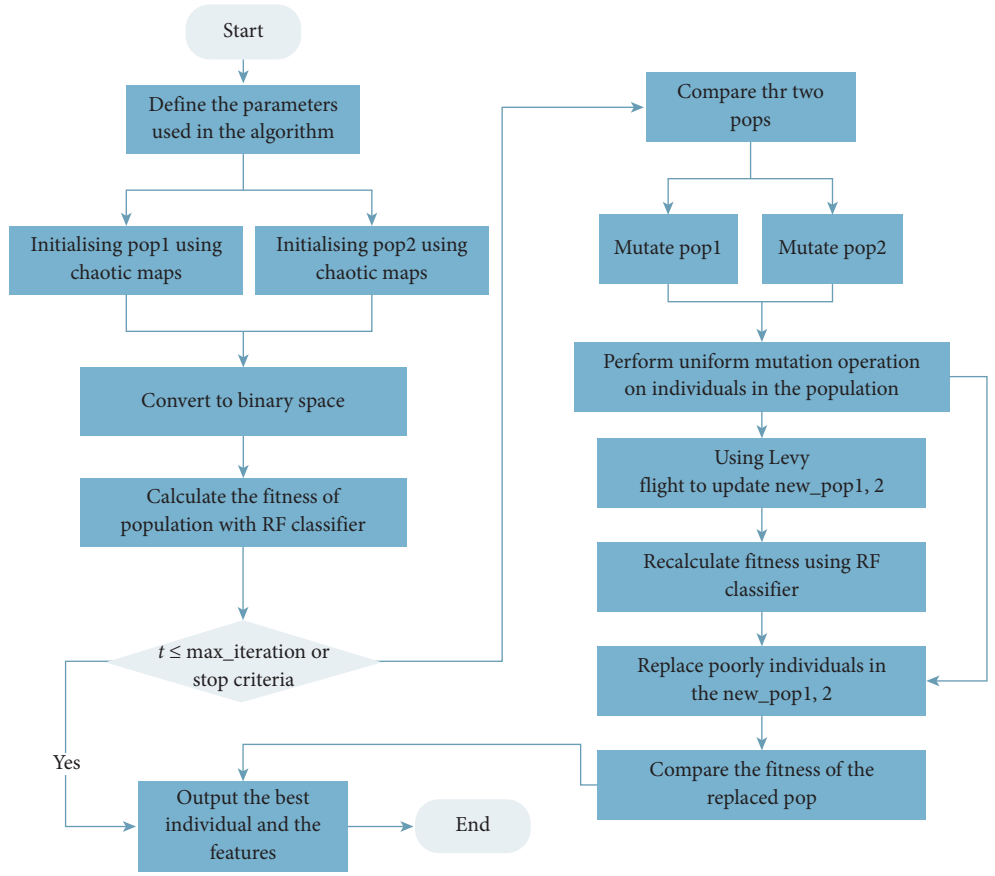

Figure 4: Algorithm flow chart.

applications, and 400 malicious samples from Dr. Wang's repository dataset (http://infosec.bjtu.edu.cn/wangwei/?page_id=85); benign apps were mainly downloaded through Google Play Python crawlers to obtain a total of 188 APKs from the Xiaomi App Market, which were released to the App Market after security testing. Besides, there are 1600 benign samples from Dr. Wang's database; second, all were from Dr. Wei Wang's database, which we selected 1200 benign software from Google market and 3200 malware from the shared malware database VirusShare. The rest of the comparison datasets are from the UCI database [42] real dataset. The attributes of the dataset samples are shown in Table 2.

Table 2: Datasets.

No.	Dataset	Number of data samples	Number of features
D1	Permission B + M	4407	88
D2	Extrapermission	2720	88
D3	SPECT Heart	267	22
D4	Sonar	208	60
D5	Diabetes	768	8
D6	Brest	683	9
D7	Clean	476	166
D8	Kr-vs-kp	3196	36
D9	Spambase	4601	57
D10	Mushroom	5644	22

4.2. Experimental Setup and Parameters Setup

4.2.1. Equipment Requirements. The experiment was conducted on the Intel core i7 computing platform with 8 GB RAM, 2.60 GHz frequency. The programming environment is python professional 2019.3.3 in the Windows 10 operating system.

4.2.2. Classifier Evaluation Indicators. The feature selection problem is essentially the data preprocessing part of solving the classification problem, so the classifier's accuracy evaluates the selection of features, and the evaluation formulas for all the classifiers used in this paper are given below.

ACC (Sample Accuracy) represents the percentage of the overall dataset that is correctly classified. The higher the ACC value, the better the classification effect. It is defined as follows:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})}, \tag{14}$$

where TP: samples that are actually positive are predicted to be positive, TN: samples that are actually negative are predicted to be negative, FP: samples that are actually negative are predicted to be positive, and FN: samples that are actually positive are predicted to be negative.

4.2.3. Comparing Algorithm. To verify the superiority of CBCSEM, we compare with other metaheuristic algorithms, such as genetic algorithm (GA) [43], grey wolf algorithm (GWO) [44], ant colony algorithms (ACO) [45], and whales optimization algorithm (WOA) [46]. In addition, we focus on a comparison with the cuckoo search algorithm with neighbourhood mutation (EBCS) [35] to further illustrate the superiority of the algorithm proposed in this paper through experimental data. The parameter settings for the comparison algorithm are given in Table 3.

In order to ensure the reliability and stability of the training model of the method proposed in this paper, the k-fold crossover is carried out for all datasets. In this paper, a ten-fold crossover is applied for validation, where the data and scores are divided into ten equal-sized copies, and the last copy is used for testing; for the training part of the classifier model, 80% of the dataset is used for training and 20% for testing the classification model.

4.3. Analysis of the Proposed Strategy. In order to verify the good or bad performance of the strategy proposed in this paper, in this section, we analyze the improved algorithm in detail. For the algorithm, the population size is set to be 50, the number of runs is equal to be 10, and the algorithm iterates 100 times.

4.3.1. Chaotic Maps and Lévy Flight. To achieve the aim of increasing population diversity, this paper proposes using different chaotic functions to initialize populations. In this section, we choose one of the chaotic maps logistic functions to compare the results of population initialization for different datasets. A graphical representation of the initialization results is shown in Figure 5.

As can be seen from the graph, the distribution of individuals initialized by chaos is more evenly distributed than that of randomly distributed individuals, with distributions in every region between 0 and 1. When the dataset has a large number of features, such as D4 and D9, the diversity of chaotic initials is not very different from random initials. In contrast, on datasets with a smaller number of feature dimensions, such as D5 and D8, we can intuitively see that the individuals initialized by the chaotic function fill the gap of random initials, and the diversity of individual distributions is greatly enhanced in the limited 0-1 search space.

Lévy flight is a random walk process, and this process is a combination of small and large step sizes in which there are significant jump events. A line graph of the fitness of individuals updated by Lévy flight compared to randomly initialized individuals is shown in Figure 6. We chose four datasets, D1, D2, D3, and D4, from which we can see that the individual fitness of the Lévy flight update spans a relatively large range, which undoubtedly increases the search space of the population and improves the exploration capacity of the algorithm in terms of scope.

4.3.2. New Elitist Preservation and Uniform Mutation. From the elite preservation strategy for two populations proposed in Section 3, it is clear that the subsequent uniform mutation operation is performed based on the elite preservation strategy; therefore, in that section, we first evaluate the effectiveness of CBCSEM without and with the mutation operation.

Table 3: State-of-the-art contrast algorithm parameter settings.

Algorithm	Parameter	Value
GA	Cross rate	0.6
	Mutation rate	0.001
	Iteration	20
	Population	50
EBCS	Nest_mut	10
	Population	50
	Iteration	20
WOA	Iteration	20
	Individuals	10
	Classifier (KNN)	$k = 5$
GWO	Iteration	20
ACO	α	1.0
	β	3.0
	Iteration	20
	ant_num	20
CBCSEM	Population	50
	Iteration	20

As can be seen from Figure 7, the CBCSEM with a uniform mutation operator shows good performance on data sets D1, D2, D3, D4, D5, D6, D7, D8, D9, and D10. For these ten datasets, the algorithm with the proposed uniform mutation all obtains the best classification accuracy.

This paper is based on the EBCS for improvement [35], so we further compared the elite preservation strategy with uniform mutation of the CBCSEM with the randomly selected individual mutation strategy of the EBCS by iterative curves. Figure 8 shows the experimental results of the two algorithms. It can be seen from Figure 8 that the overall performance of CBCSEM is better than that of the EBCS. For D1, D3, D8, and D10, the classification accuracy of the two algorithms is not much different, but the former has faster convergence rates. For D2, D4, D5, D6, D7, and D9, the classification accuracy of the CBCSEM is far better than the EBCS, which proves that the two-population elite preservation strategy proposed in this paper performs well.

4.4. Experimental Results. In this section, we analyze the experimental results of the CBCSEM proposed in this paper against other comparative algorithms. To make the results fairer and well convincing, the results of all experiments are averaged over 10 independent experiments. In performing CBCSEM, the paper specifies that the population size is equal to 50 and the algorithm iterates 20 times. We use the Random Forest algorithm for classification to find the accuracy of the algorithm, in which the number of subdecision trees is set to 10.

To show that our proposed algorithm's performance is significantly better than that of the comparison algorithms, we used a nonparametric statistical test: the Wilcoxon rank-sum test with a significance level of $\alpha = 0.05$. The null hypothesis is that the proposed algorithm is not significantly

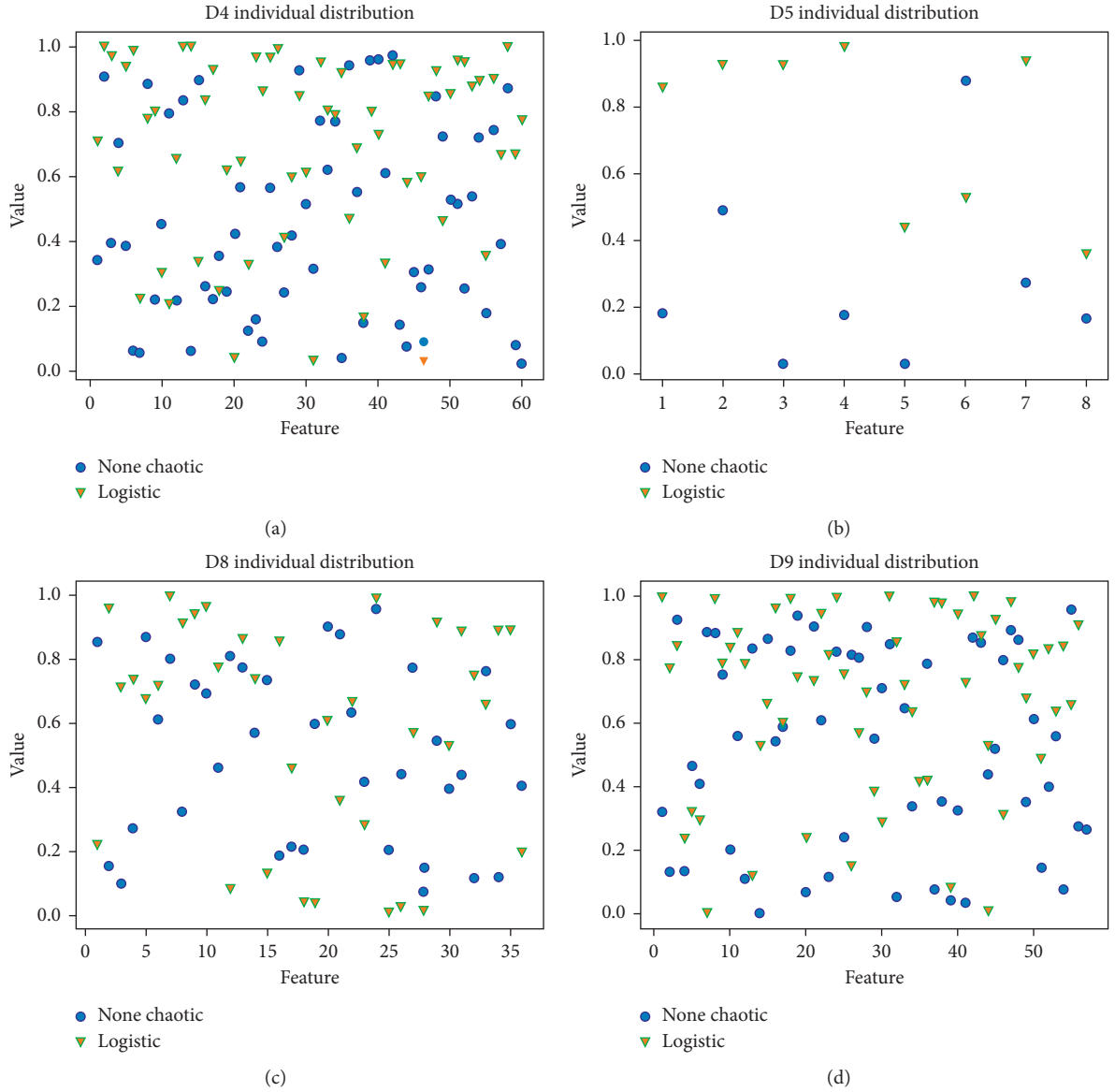

Figure 5: Initialization of different datasets. (a) Initialization of the D4 dataset. (b) Initialization of the D5 dataset. (c) Initialization of the D8 dataset. (d) Initialization of the D9 dataset.

different from the comparison algorithm, and the alternative hypothesis is that the proposed algorithm is significantly different from the comparison algorithm. We use the symbols $+, =, -$ to indicate that the proposed algorithm's performance is significantly better, has no significant difference, and is significantly inferior to the corresponding comparison algorithm.

The results of different chaotic maps on different datasets are shown in Table 4 where SF is denoted as the percentage of features selected by the algorithm and ACC results from averaging the ten-fold cross validation of the classifier. As can be seen from the data in Table 4, each chaotic map, except the Sine map, showed some advantages on different datasets. Overall, logistic function showed higher accuracy on the D3, D4, D5, and D10 datasets than the other maps. Therefore, we chose to use the CBCSEM with the logistic map compared to the other algorithms.

The comparison of the CBCSEM with the GA, WOA, GWO, and ACO algorithms is shown in Table 5. The data in the table are the results of ten independent runs of each algorithm. In comparison with the four state-of-the-art algorithms, it can be seen that the GWO algorithm performance does not show a significant advantage in terms of ACC or the number of features selected, the GA algorithm has a high accuracy of the classifier on D2, D4, and D9, the WOA algorithm has a high ACC value on the D1 dataset, and the ACO algorithm has a high ACC value on D6. In addition to the advantages of the comparison

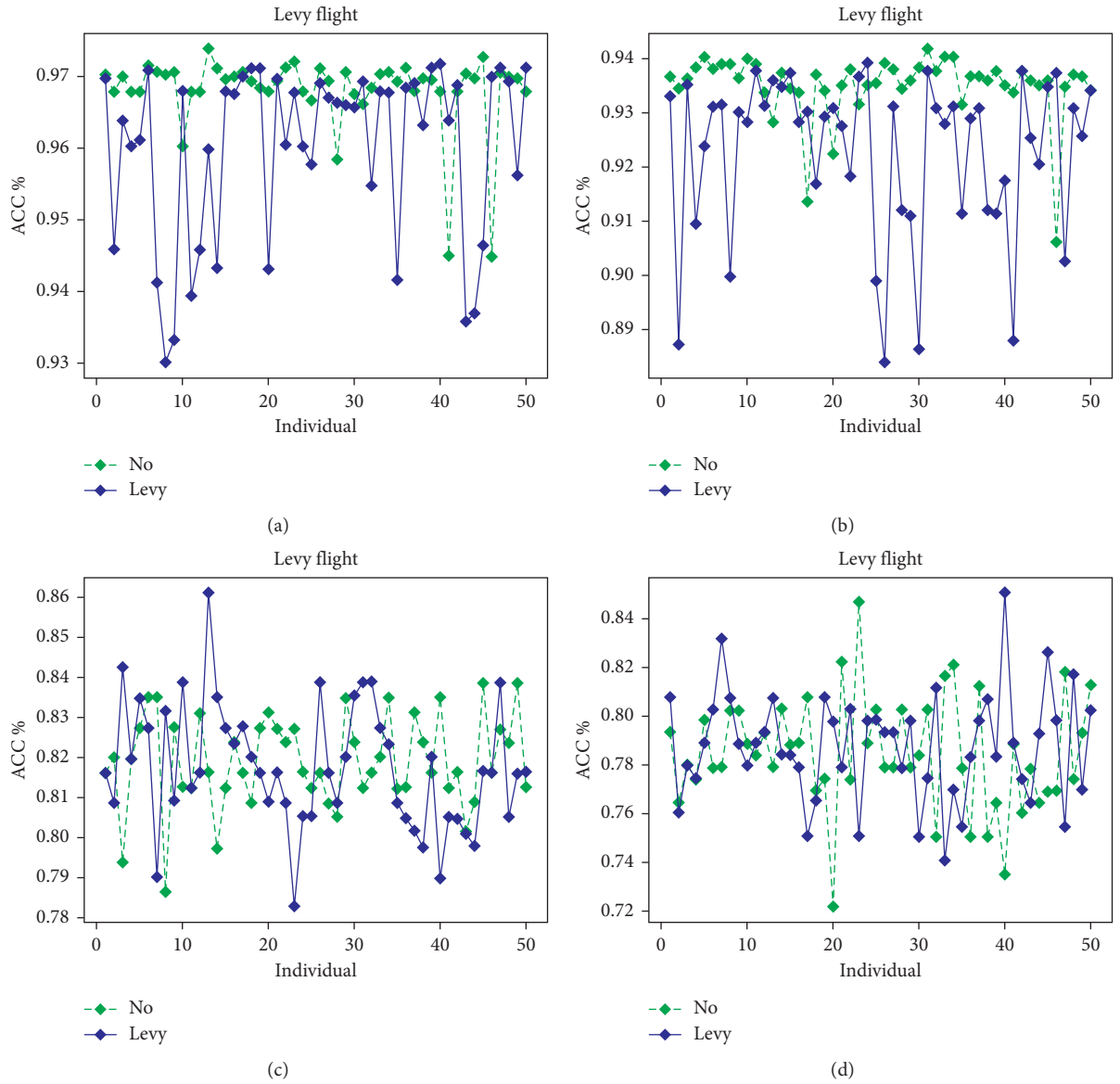

Figure 6: The impact of Lévy flight on the ACC. (a) Fitness for the Lévy update on the D1. (b) Fitness for the Lévy update on the D2. (c) Fitness for the Lévy update on the D3. (d) Fitness for the Lévy update on the D4.

algorithm, CBCSEM has the highest ACC values on half of the datasets, for example, D3, D5, D7, D8, and D10, and even though the accuracy is not as high as the comparison algorithm, the SF metrics account for nearly half or less than half of the complete set of features. A graphical representation of the ACC values is shown in Figures 9(a) and 10(a).

In this paper, CBCSEM is inspired by the EBCS to improve the shortcomings of the EBCS and improve the performance of the algorithm, so the paper focuses on comparing the two algorithms. In Table 6, we compare the ACC values, SF values, and the iteration times of the algorithms for both algorithms.

According to the data in Table 6, we can see that the two algorithms have their advantages in terms of accuracy and the number of feature subsets selected. A graphical depiction of the comparison results is shown in Figures 9(b) and 10(b). The ACC values of the EBCS are generally higher than those of the CBCSEM on D1, D2, D5, D6, D9, and D10, even though the difference between the CBCSEM and the former does not exceed 0.3% above and below, which is negligible when the number of selected features is reduced. On the D3, D4, D7, and D8 datasets, the CBCSEM has higher values than the EBCS for both accuracy metrics and feature share values. In addition to the two hard metrics, we find that the improved algorithm takes much less time to select features than the former EBCS, a total ten-fold speedup that significantly improves algorithm improvement. Combined with the ten different dataset properties given in Section 4.1, we can conclude that the CBCSEM performs well on smaller datasets with sample sizes up to 500.

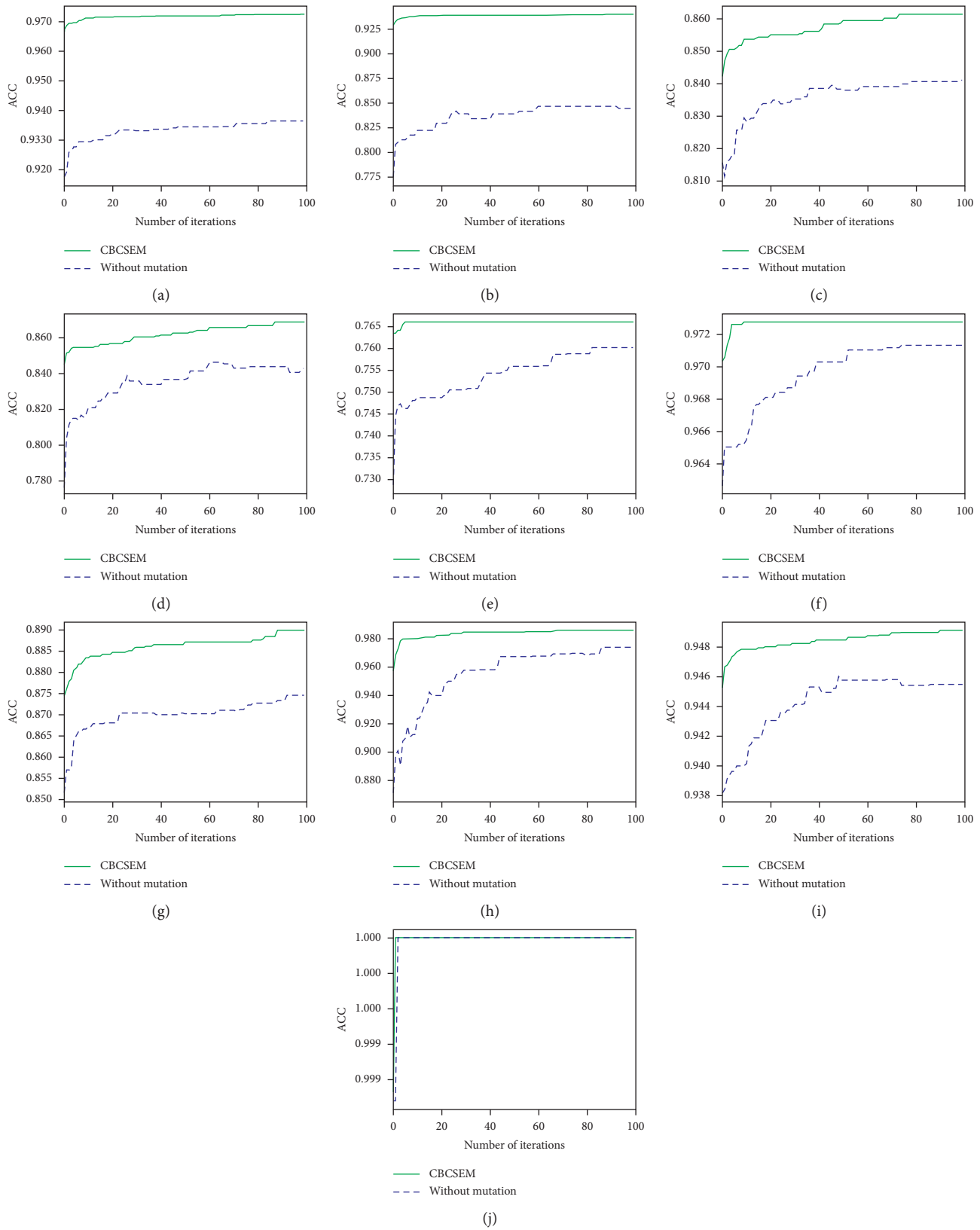

Figure 7: Experimental results of CBCSEM in the following two cases: with uniform mutation and without mutation. (a) D1, (b) D2, (c) D3, (d) D4, (e) D5, (f) D6, (g) D7, (h) D8, (i) D9, and (j) D10.

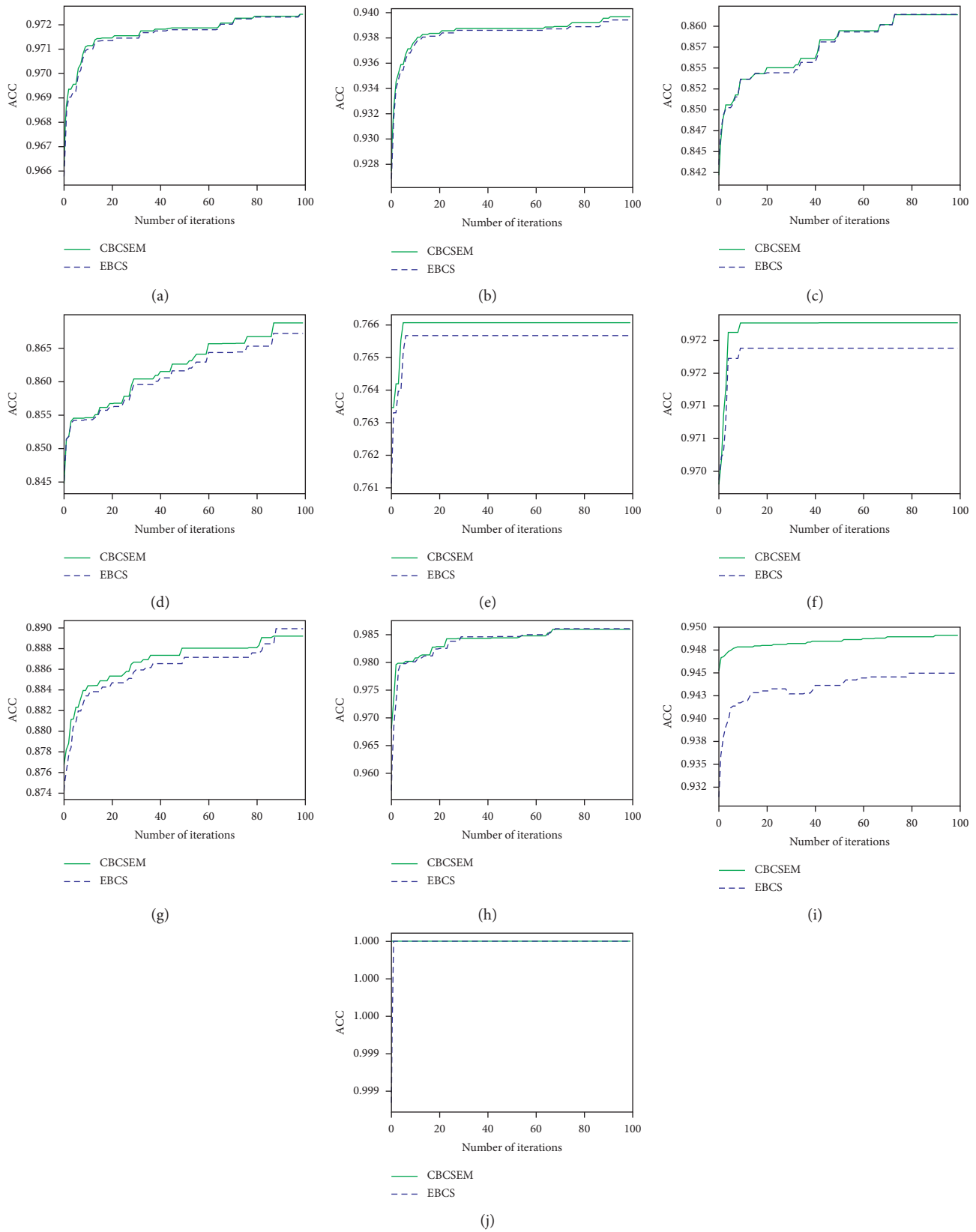

Figure 8: The evolutionary curves of CBCSEM and EBCS. (a) D1, (b) D2, (c) D3, (d) D4, (e) D5, (f) D6, (g) D7, (h) D8, (i) D9, and (j) D10.

Table 4: Results of different chaos maps on different datasets.

No.	Chaotic maps	D1		D2		D3		D4		D5		D6		D7		D8		D9		D10	
		ACC	SF	ACC	SF	ACC	SF	ACC	SF	ACC	SF	ACC	SF	ACC	SF	ACC	SF	ACC	SF	ACC	SF
1	Logistic	0.9021	53.06	0.9530	57.84	**0.8945**	53.63	**0.8995**	56.67	**0.8016**	61.25	0.9731	85.56	0.8839	52.59	0.9837	75.28	0.9468	64.49	**1.0000**	68.64
2	Singer	**0.9350**	66.93	0.9696	67.72	0.8830	54.09	0.8602	54.50	0.7610	68.75	0.9719	72.22	0.8830	50.48	0.9798	78.05	0.9466	69.82	1.0000	66.82
3	Sinusoidal	0.9330	63.63	**0.9706**	58.52	0.8869	49.54	0.8529	52.17	0.7633	73.75	0.9707	67.77	**0.8858**	51.08	0.9799	78.05	**0.9472**	62.63	1.0000	59.90
4	Iterative	0.9319	59.43	0.9696	56.25	0.8533	49.55	0.8582	52.83	0.7630	75.00	**0.9733**	78.88	0.8850	50.66	0.9803	65.00	0.9468	60.52	1.0000	50.90
5	Sine	0.9303	71.59	0.9688	61.70	0.8807	52.27	0.8600	56.00	0.7626	72.50	0.8600	56.00	0.7626	72.50	0.8440	55.38	0.9315	93.33	0.9000	58.00
6	Gauss/mouse	0.9320	84.65	0.9686	73.75	0.8500	56.37	0.8255	58.00	0.7537	77.50	0.9704	78.88	0.8702	55.30	**0.9840**	87.77	0.9470	87.36	1.0000	100

The bold values show the best result of the chaotic maps in different datasets.

Table 5: Comparison of ACC and SF results for different algorithms.

Dataset	GA		WOA		GWO		ACO		CBCSEM	
	ACC	SF	ACC	SF	ACC	SF	ACC	SF	ACC	SF
D1	0.9114	54.77	**0.9237**	64.31	0.8533	44.48	0.8942	53.63	0.9022	**53.07**
D2	**0.9729**	48.86	0.9561	62.61	0.9243	**51.59**	0.9484	54.65	0.9531	57.84
D3	0.8582	48.18	0.8530	42.22	0.7670	56.63	0.8005	62.22	**0.8945**	**53.64**
D4	**0.9023**	52.50	0.8825	**50.00**	0.7786	52.66	0.8843	53.17	0.8996	**50.67**
D5	0.7922	41.25	0.7346	37.50	0.7124	47.36	0.7596	55.00	**0.8017**	61.25
D6	0.9730	61.11	0.9659	54.44	0.9597	53.33	**0.9894**	83.33	0.9731	85.56
D7	0.8708	49.04	0.8734	75.48	0.8298	48.67	0.8632	49.22	**0.8839**	**52.59**
D8	0.9828	48.33	0.9514	71.94	0.8597	51.67	0.9641	40.00	**0.9837**	75.28
D9	**0.9512**	50.88	0.9059	60.00	0.9245	49.47	0.9239	63.68	0.9468	64.49
D10	1.0000	57.73	0.9999	38.64	0.9758	49.09	0.9813	60.00	**1.0000**	68.64

The best results among different algorithms have been shown in bold.

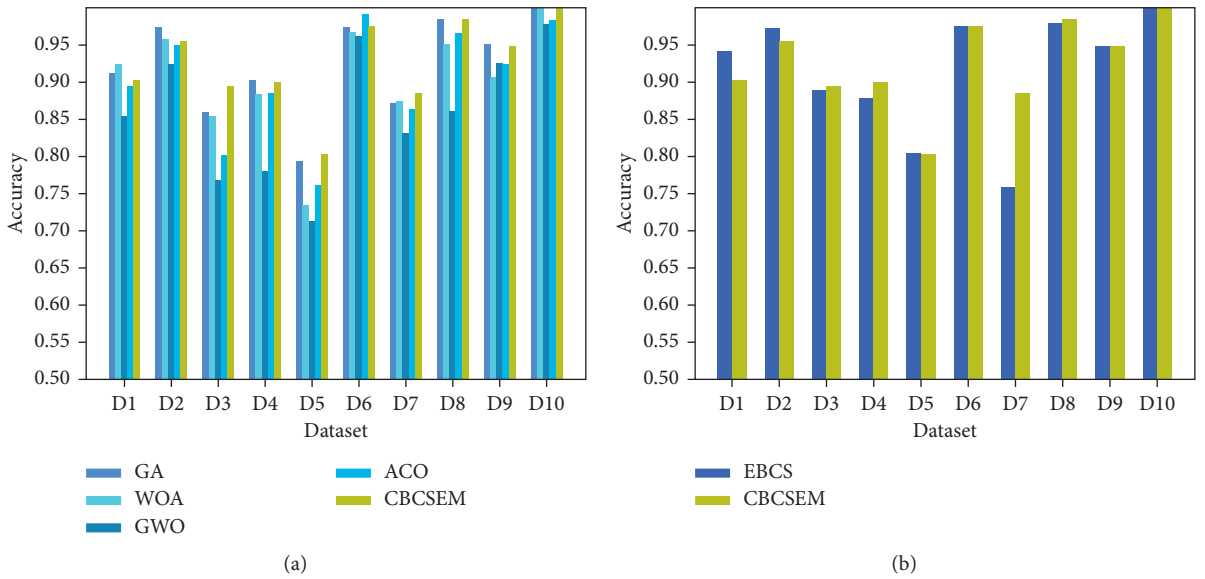

Figure 9: Comparison of accuracy results. (a) Performance comparison of CBCSEM with state-of-the-art algorithms. (b) Performance comparison between CBCSEM and EBCS.

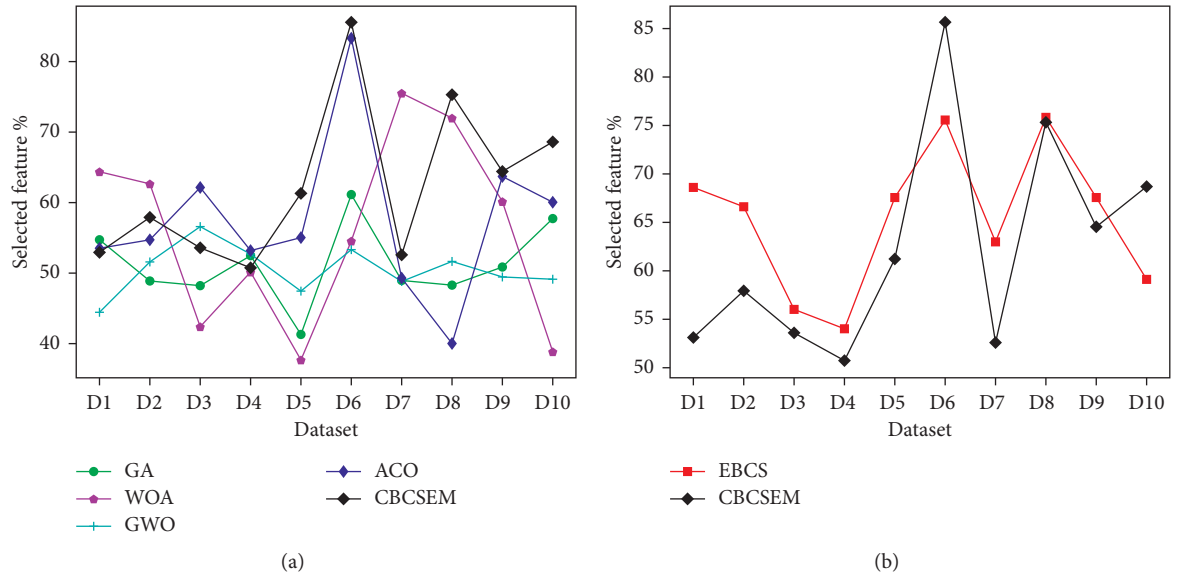

Figure 10: Comparison of SF results. (a) Performance comparison of CBCSEM with state-of-the-art algorithms. (b) Performance comparison between CBCSEM and EBCS.

Table 6: Comparison of EBCS and CBCSEM.

Dataset	Indicators	EBCS	CBCSEM
D1	AVG-ACC	**0.9393**	0.9022
	AVG-SF	68.52	**53.06**
	AVG-TIME	402.74	**28.85**
D2	AVG-ACC	**0.9710**	0.9531
	AVG-SF	66.59	**57.84**
	AVG-TIME	373.78	**24.40**
D3	AVG-ACC	0.8885	**0.8945**
	AVG-SF	56.00	**53.63**
	AVG-TIME	145.05	**8.74**
D4	AVG-ACC	0.8766	**0.8996**
	AVG-SF	54.00	**50.66**
	AVG-TIME	137.18	**11.37**
D5	AVG-ACC	**0.8038**	0.8017
	AVG-SF	67.50	**61.25**
	AVG-TIME	143.76	**14.02**
D6	AVG-ACC	**0.9732**	0.9731
	AVG-SF	75.56	85.56
	AVG-TIME	161.17	**10.78**
D7	AVG-ACC	0.7588	**0.8839**
	AVG-SF	63.00	**52.59**
	AVG-TIME	217.16	**22.41**
D8	AVG-ACC	0.9790	**0.9837**
	AVG-SF	75.83	**75.28**
	AVG-TIME	383.23	**20.73**
D9	AVG-ACC	**0.9469**	0.9468
	AVG-SF	67.54	**64.49**
	AVG-TIME	518.87	**51.02**
D10	AVG-ACC	**1.0000**	**1.0000**
	AVG-SF	**59.09**	68.64
	AVG-TIME	961.81	**19.63**

The bold values show the best result in the algorithm comparison.

Table 7: Mean and standard deviation results of acc on CBCSEM and its competitors for ten datasets.

	CBCSEM Mean (std)	EBCS Mean (std)	GA Mean (std)	WOA Mean (std)	GWO Mean (std)	ACO Mean (std)
D1	90.22 (**0.33**)	**93.93** (6.59) −	91.14 (0.54) =	92.37 (5.18) =	85.33 (0.50) +	89.42 (4.34) =
D2	95.31 (**0.25**)	97.10 (1.08) =	**97.29** (0.30) −	95.61 (2.69) −	92.43 (0.34) +	94.84 (3.51) +
D3	**89.45** (2.04)	88.85 (**1.35**) =	85.82 (1.69) +	85.30 (4.22) +	76.70 (5.86) +	80.05 (2.04) +
D4	89.96 (1.71)	87.66 (2.61) +	**90.23** (3.10) =	88.25 (2.18) =	77.86 (**1.30**) +	88.43 (1.73) +
D5	80.17 (2.10)	**80.38** (1.01) =	79.22 (**0.74**) =	73.46 (4.46) +	71.24 (1.72) +	75.96 (1.48) +
D6	97.31 (0.26)	97.32 (0.35) =	97.30 (0.40) =	96.59 (1.38) +	95.97 (**0.12**) +	**98.94** (0.27) -
D7	**88.39** (**0.66**)	75.88 (0.88) +	87.08 (1.07) +	87.34 (1.53) =	82.98 (0.70) +	86.32 (0.75) +
D8	**98.37** (1.05)	97.90 (0.57) =	98.28 (0.85) =	95.14 (8.33) +	85.97 (**0.28**) +	96.41 (0.44) +
D9	94.68 (0.21)	94.69 (0.22) =	**95.12** (0.44) −	90.59 (1.01) +	92.45 (0.88) +	92.39 (**0.17**) +
D10	**100.00** (**0.00**)	**100.00** (**0.00**) =	**100.00** (0.04) =	99.99 (3.44) =	97.58 (1.61) +	98.13 (**0.00**) +
+/=/−		2/7/1	2/6/2	5/4/1	10/0/0	8/1/1

The bold values show the best mean and standard deviation results.

In Table 7, it can be seen that the outcomes of the mean and standard deviation are reported for all algorithms. In ten datasets, the mean and std values of accuracy found by CBCSEM are bigger than the best results obtained by other metaheuristics. It can be observed that CBCSEM has a high accuracy when dealing with D3, D7, D8, and D10. Furthermore, it indicates that the CBCSEM is more stable for solving these datasets. Moreover, CBCSEM finds significantly accurate solutions than WOA, GWO, and ACO on more than half of the datasets. Compared with GA, CBCSEM has significantly improved performance on 2 datasets. Compared with EBCS, although CBCSEM has significantly reduced performance on 1 dataset, however, CBCSEM has significantly improved performance by 2 datasets than this algorithm. In other words, CBCSEM performs much better than all other competitors.

5. Conclusions

In this paper, we propose a new feature selection algorithm, called CBCSEM, and successfully applied it to solve the feature selection problem. The proposed algorithm extends EBCS by including three new strategies, namely, chaotic maps, Lévy flight, and a two-population elite preservation strategy with uniform mutation, which are very effective when dealing with feature selection problems. To evaluate the performance of the proposed algorithm, we implemented the algorithm on ten real datasets. In future work, other strategies can be incorporated to reduce the time complexity of the CBCSEM. Currently, this paper defines the feature selection problem as a single-objective optimization problem, which can later be defined as a multiobjective optimization problem, using different constrained optimization methods to solve the feature selection problem.

Data Availability

The data sets cited in the article are all public data sets and are quoted in the original text.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant 11961001, the Construction Project of First-Class Subjects in Ningxia Higher Education (NXYLXK2017B09), the Major Proprietary Funded Project of North Minzu University (ZDZX201901), and Postgraduate Innovation Project Funding of Northern University for Nationalities (YCX20087).

References

[1] Y. Zhang, S. Wang, P. Phillips, and G. Ji, "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection," *Knowledge-Based Systems*, vol. 64, pp. 22–31, 2014.

[2] X. Bing, M. Zhang, W. N. Browne, and Y. Xin, "BA survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 606–626, 2016.

[3] D. B. Osteyee and I. J. Good, "Information, weight of evidence, the singularity between probability measures and signal detection,information, weight of evidence, the singularity between probability measures and signal detection," *Information, Weight of Evidence, the Singularity between Probability Measures and Signal Detection*, vol. 376, 1974.

[4] A. Jhoseph Jesus, A. Anne Canuto, and B. Daniel Araujo, "An exploratory analysis of data noisy scenarios in a Pareto-front based dynamic feature selection method," *Applied Soft Computing*, vol. 100, Article ID 106951, 2020.

[5] C. Lei, C. S. Gates, S. Luo, and N. Li, "A probabilistic discriminative model for android malware detection with decompiled source code," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, pp. 400–412, 2015.

[6] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 80, 1997.

[7] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, "Metaheuristic algorithms on feature selection: a survey of one decade of research (2009–2019)," *IEEE Access*, vol. 9, pp. 26766–26791, 2021.

[8] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Pattern Recognition*, vol. 33, no. 1, pp. 25–41, 2000.

[9] A. K. Shukla, P. Singh, and M. Vardhan, "A new hybrid feature subset selection framework based on binary genetic algorithm and information theory," *International Journal of Computational Intelligence and Applications*, vol. 18, no. 3, 2019.

[10] A. C. Vieira, P. Souza, J. Ueyama, G. Pessin, and Pabon, "Rec,Improving flood forecasting through feature selection by a genetic algorithm -experiments based on real data from an Amazon rainforest river," *Earth Science Informatics*, vol. 51, 2021.

[11] A. Xfs, Z. A. Yong, B. Dwga, and A. Xys, "Feature selection using bare-bones particle swarm optimization with mutual information," *Pattern Recognition*, vol. 98, Article ID 106794, 2020.

[12] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective particle swarm optimization for feature selection with fuzzy cost," *IEEE Transactions on Cybernetics*, vol. 51, no. 2, 2020.

[13] M. A. Shoorehdeli, M. Teshnehlab, and H. A. Moghaddam, "Feature subset selection for face detection using genetic algorithms and particle swarm optimization," *IEEE International Conference on Networking*, vol. 14, 2006.

[14] X. F. Song, Y. Zhang, D. W. Gong, and X. Z. Gao, "A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data," *IEEE Transactions on Cybernetics*, vol. 99, pp. 1–14.

[15] S. Palanisamy and S. Kanmani, "Artificial bee colony approach for optimizing feature selection," *International Journal of Computer Science Issues*, vol. 9, pp. 432–438, 2012.

[16] P. N. S. Ninu, G. Brammya, R. Ramya, S. Praveena, and B. R. Rajakumar, "Grey wolf optimization-based feature selection and classification for facial emotion recognition," *IET Biom*, vol. 7, no. 5, 2018.

[17] Yu Xue, Y. Zhao, and A. Slowik, "Classification based on brain storm optimization with feature selection," *IEEE Access*, vol. 9, pp. 16582–16590, 2020.

[18] Y. Zhang, X.-f. Song, and D.-w. Gong, "A return-cost-based binary firefly algorithm for feature selection," *Information Sciences*, vol. 419, pp. 561–574, 2017.

[19] Y. M. N. Rodrigo, L. A. M. Pereira, K. A. P. Costa, R. Douglas, J. P. Papa, and X.-S. Yang, "BBA: a binary bat algorithm for feature selection," in *Proceedings of the 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pp. 291–297, Ouro Preto, Brazil, August 2012.

[20] Z. Sadeghian, E. Akbari, and H. Nematzadeh, "A hybrid feature selection method based on information theory and binary butterfly optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 97, Article ID 104079, 2021.

[21] Tein, H. Lim, and R. Ramli, "Translational impact," *Disease Models & Mechanisms*, vol. 3, no. 5-6, p. 395, 2010.

[22] S. M. Ali, "Hybrid cuckoo search-genetic algorithm (CSGA): an approach to solve some combinatorial problems," *Journal of Advanced Computer Science and Technology Research*, vol. 5, pp. 123–132, 2015.

[23] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, 2017.

[24] Z. M. Elgamal, N. B. M Yasin, M. Tubishat, M. Alswaitti, and S. Mirjalili, "An improved Harris hawks optimization algorithm with simulated annealing for feature selection in the medical field," *IEEE Access*, vol. 8, pp. 186638–186652, 2020.

[25] M. Abdel-Basset, W. Ding, and D. El-Shahat, "A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection," *Artificial Intelligence Review*, vol. 54, no. 1, pp. 593–637, 2021.

[26] F. Moslehi and A. Haeri, "A novel hybrid wrapper-filter approach based on genetic algorithm, particle swarm optimization for feature subset selection," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 3, pp. 1105–1127, 2020.

[27] M. Tubishat, N. Idris, L. Shuib, M. A. M. Abushariah, and S. Mirjalili, "Improved salp swarm algorithm based on opposition based learning and novel local search algorithm for feature selection," *Expert Systems with Applications*, vol. 145, Article ID 113122, 2020.

[28] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," *World Congress on Nature & Biologically Inspired Computing (NaBIC)*, vol. 9, pp. 210–214, 2009.

[29] M. Phogat and D. Kumar, "Classification of complex diseases using an improved binary cuckoo search and conditional mutual information maximization," *Computacion Y Sistemas*, vol. 24, no. 3, 2020.

[30] R. Salgotra, U. Singh, S. Saha, and A. H. Gandomi, "Self adaptive cuckoo search: analysis and experimentation," *Swarm and Evolutionary Computation*, vol. 60, Article ID 100751, 2020.

[31] H. T. Lim and R. Ramli, "Recent advancements of nurse scheduling models and a potential path," 2010.

[32] Shirin, Nozarian, Hodais et al., "A binary model on the basis of cuckoo search algorithm in order to solve the problem of knapsack 1-0," in *Proceedings of the 2012 International Conference on System Engineering and Modeling(ICSEM 2012)*, Beijing, China, April 2012.

[33] D. Rodrigues, L. A. M. Pereira, T. N. S. Almeida, J. P. Papa, and X. S. Yang, "BCS: a binary cuckoo search algorithm for feature selection," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Beijing, China, May 2013.

[34] J. Hamidzadeh and others, "Feature selection by using chaotic cuckoo optimization algorithm with levy flight, opposition-based learning and disruption operator," *Soft Computing*, vol. 25, no. 25, pp. 2911–2933, 2021.

[35] S. Salesi and G. Cosma, "A novel extended binary cuckoo search algorithm for feature selection," in *Proceedings of the International Conference on Knowledge Engineering and Applications*, London, UK, October 2017.

[36] Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, and F. Yuan, "CBSO: a memetic Brain Storm optimization with chaotic local search," *Memetic Computing*, vol. 99, 2018.

[37] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, Washington, DC, USA, October 1997.

[38] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes," *Physical Review E*, vol. 49, no. 5, pp. 4677–4683, 1994.

[39] J. Holland, "Adaptation in natural and artificial systems: an introductory analysis with application to biology," *Control and Artificial Intelligence*, vol. 211, 1975.

[40] M. Zhou and S. Sun, "Principles and applications of genetic algorithms," 1999.

[41] L. Taheri, A. Kadir, and A. H. Lashkari, "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls," in *Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST)*, Chennai, India, October 2019.

[42] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[43] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," *IEEE Intelligent Systems and Their Applications*, vol. 13, pp. 44–49, 2002.

[44] A. Sm, B. Smm, and A. Al, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 46, pp. 56–61, 2014.

[45] Y. Wan, M. Wang, Z. Ye, and X. Lai, "A feature selection method based on modified binary coded ant colony optimization algorithm," *Applied Soft Computing*, vol. 49, pp. 248–258, 2016.

[46] A. G. Hussien, A. E. Hassanien, E. H. Houssein, S. Bhattacharyya, and M. Amin, "S-shaped binary whale optimization algorithm for feature selection," *Recent Trends in Signal and Image Processing*, vol. 9, pp. 79–87, 2019.