Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.053627

ARTICLE

# A Feature Selection Method Based on Hybrid Dung Beetle Optimization Algorithm and Slap Swarm Algorithm

**Wei Liu*** **and Tengteng Ren**

School of Information Science and Engineering, Shenyang Ligong University, Shenyang, 110158, China

*Corresponding Author: Wei Liu. Email: liuwei19781020@126.com

Received: 06 May 2024 Accepted: 11 July 2024 Published: 15 August 2024

**ABSTRACT**

Feature Selection (FS) is a key pre-processing step in pattern recognition and data mining tasks, which can effectively avoid the impact of irrelevant and redundant features on the performance of classification models. In recent years, meta-heuristic algorithms have been widely used in FS problems, so a Hybrid Binary Chaotic Salp Swarm Dung Beetle Optimization (HBCSSDBO) algorithm is proposed in this paper to improve the effect of FS. In this hybrid algorithm, the original continuous optimization algorithm is converted into binary form by the S-type transfer function and applied to the FS problem. By combining the K nearest neighbor (KNN) classifier, the comparative experiments for FS are carried out between the proposed method and four advanced meta-heuristic algorithms on 16 UCI (University of California, Irvine) datasets. Seven evaluation metrics such as average adaptation, average prediction accuracy, and average running time are chosen to judge and compare the algorithms. The selected dataset is also discussed by categorizing it into three dimensions: high, medium, and low dimensions. Experimental results show that the HBCSSDBO feature selection method has the ability to obtain a good subset of features while maintaining high classification accuracy, shows better optimization performance. In addition, the results of statistical tests confirm the significant validity of the method.

**KEYWORDS**

Feature selection; dung beetle optimization; KNN; transfer function; HBCSSDBO

## 1 Introduction

With the development of the information industry, science and technology, and the gradual maturity of modern collection technology, thousands of applications have produced a huge amount of data information, and various data information also provides a guarantee for the development of technology. At the same time, the data analysis and processing are also faced with a big challenge, because the data dimension is too large to get more valuable information. Therefore, dimension reduction is very important in the process of data pre-processing [1].

FS extracts a significant portion of features from the original dataset and uses predetermined assessment metrics to eliminate characteristics that are unnecessary, redundant, or noisy in order to reduce dimension. In contrast to Feature Extraction, which also performs the function of dimension reduction, Feature Extraction accomplishes this through the creation of new feature combinations,

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

while FS does not need to modify original features, so FS has been widely used in text mining and other fields [2]. FS can be divided into three types: filter, wrapper, and embedded. When selecting features, the filter method can select important features based on the internal relationship between features without combining the learning method. Although this method is simple, it usually has low classification accuracy [3]. The wrapper approach evaluates the quality of selected features by combining them with a machine learning model, and the evaluation process is achieved by searching for subsets of features repeatedly until certain stopping conditions are met or the desired optimal subset of features is obtained. This method can obtain higher classification accuracy, but at the same time has greater computational complexity. The embedded method is to embed feature selection into learner training. Compared with the wrapper, it saves more time, but easily increases the training burden of the model.

An increasing number of algorithms have been used in the field of FS as a result of the ongoing development of meta-heuristic algorithms [4]. The meta-heuristic algorithm is an optimization algorithm generated by simulating physical phenomena or biological behaviors in nature. Thanks to the advantages of the simple model, easy implementation, and high robustness, meta-heuristics have been successfully implemented in several optimization domains. Nevertheless, as stated by the No Free-Lunch (NFL) theorem [5], it is impossible for any method to successfully solve every optimization issue, which also encourages researchers to innovate and improve the algorithm constantly, so as to apply the optimization algorithm to a broader field. In the field of FS, Song et al. [6] considered that the particle updating mechanism and population initialization strategy adopted by the traditional Particle Swarm Optimization (PSO) did not consider the characteristics of the FS problem itself, which limited its performance in dealing with high-dimensional FS problems. So they proposed a Bare-Bones Particle Swarm Optimization (BBPSO) based on mutual information. By combining with the K nearest neighbor (KNN) classifier, the effectiveness of the proposed algorithm was verified on multiple datasets, and the classification accuracy was greatly improved. A better binary Salp Swarm Algorithm (SSA) for FS was developed by Faris et al. [7]. To enhance the algorithm's exploration performance, crossover operators were added and continuous SSA was transformed into binary form using various transfer functions. The suggested method's higher performance was confirmed when compared to the other 5 algorithms over 22 UCI (University of California Irvine) datasets. Emary et al. [8] proposed a binary Grey Wolf Optimization for FS, which binarized the continuous algorithm through the S-type transfer function and verified the method on multiple datasets. It has provided an important reference value for the subsequent improvement of Grey Wolf Optimization for FS. Mafarja et al. [9] proposed two improved Whale Optimization Algorithm for FS, which demonstrated better classification performance through verification on UCI datasets and comparison with other algorithms. Researchers have also proposed many hybrid optimization algorithms and applied them to FS problems. Enhanced Chaotic Crow Search and Particle Swarm Optimization (ECCSPSOA), a hybrid binary introduced by Adamu et al. [10] accomplished the organic merger of Particle Swarm Optimization and Crow Search Algorithm and included the opposition-based learning technique. The algorithm's search space was enlarged, and several chaotic initialization procedures were employed to enhance its optimization efficiency, resulting in significant improvements. For high-dimensional imbalanced data containing missing values, a particle swarm optimization-based fuzzy clustering FS algorithm (PSOFS-FC) was proposed by Zhang et al. [11]. It shows superiority over state-of-the-art FS methods in handling high-dimensional unbalanced data with missing values by exhibiting remarkable classification performance on several public datasets. To solve high-dimensional FS issues, Song et al. [12] suggested a variable-size cooperative coevolutionary particle swarm optimization method (VS-CCPSO) and evaluated it against six other algorithms on twelve datasets. The experimental results demonstrated that the algorithm is more advantageous in handling high-dimensional FS problems. In addition, literature [13–15] also proposed different hybrid algorithms for FS, provides new ideas and directions for the application of swarm intelligence optimization algorithms in feature selection.

Dung Beetle Optimization (DBO) [16], as a new meta-heuristic algorithm, shows strong optimization ability in the benchmark function, but there are also problems such as easy to entrap in local optima and exploration and exploitation ability imbalance. Therefore, a hybrid Dung Beetle Optimization is proposed in this paper. The original rolling dung beetle update is replaced with the salp swarm method, which has a unique leader and follower mechanism and enhances the algorithm's capacity to explore the solution space. Subsequently, the population is initialized using chaotic mapping, augmenting the DBO's population diversity. Finally, the mutation operator is added to stealing population position updates in the DBO to improve the ability of the algorithm to jump from local optima. By converting the improved DBO into binary form through transfer function, a Hybrid Binary Chaotic Salp Swarm Dung Beetle Optimization (HBCSSDBO) is proposed and applied to FS. The main motivation for the decision to merge DBO and SSA was the strong merit-seeking capability of DBO and the unique leader, and follower search mechanism of SSA. The complementary advantages of the algorithms are realized. The goal of merging these two algorithms is to create a hybrid feature selection algorithm that is more effective. The following highlights this paper's primary contributions:

- Introducing a unique feature selection approach for dung beetle optimization in hybrid salp swarms. Combining the salp swarm optimization method with the dung beetle optimization algorithm makes good use of the features that make each approach distinct.
- Using this hybrid algorithm combined with a KNN classifier greatly improves classification accuracy.
- The effectiveness of the hybrid algorithm as applied to the FS problem is explored by comparing it with four better-known meta-heuristics on 16 UCI benchmark test datasets, and by employing a variety of evaluation criteria as well as multiple perspectives from high, medium, and low dimensionality.
- The Friedman test was employed to see whether there was a significant difference between the outcomes produced from the comparison approach and the suggested hybrid feature selection method.

## 2 Relate Works

### *2.1 Dung Beetle Optimization*

Dung Beetle Optimization (DBO) is a new swarm intelligence optimization algorithm inspired by simulating the social behavior of dung beetles in nature. In this algorithm, the original author divided the dung beetle population into four different agents (ball-rolling dung beetles, brood balls, small dung beetles, and stealing dung beetles) according to a certain proportion to simulate different dung beetle behaviors. The specific proportion division is shown in Fig. 1.

#### *2.1.1 Dung Beetle Ball Rolling Behavior*

To keep the ball moving in a straight course, dung beetles in the wild must pay attention to astronomical signals. The locations of the beetles are continually shifting as they roll. The mathematical model of dung beetles' ball-rolling is shown in Eq. (1):

$$x(t+1) = x_i(t) + \alpha \times k \times x_i(t-1) + b \times \Delta x,$$
$$\Delta x = | x_i(t) - X^w | \quad (1)$$

where $x_i(t)$ is the position of the $i$-th dung beetle in the $t$-th iteration, $\alpha$ is a natural coefficient, assigned $-1$ or 1, $k$ represents the deflection coefficient and $k \in (0, 0.2)$, $b$ represents a random constant and $b \in (0, 1)$. $X^w$ denotes the global worst position.

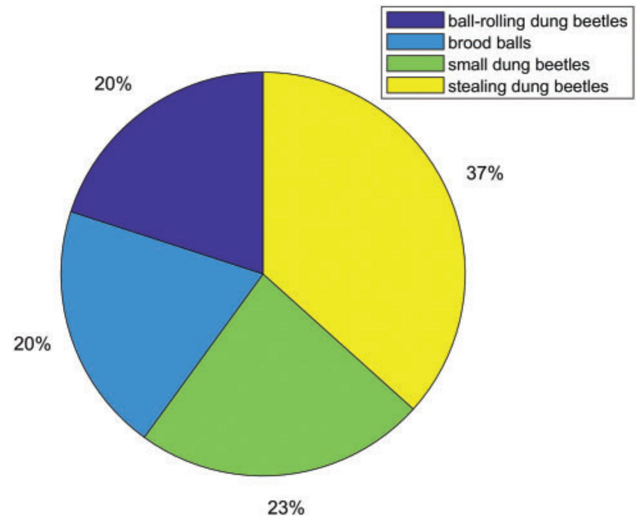

**Figure 1:** Proportion division of dung beetle population

A dung beetle must dance to figure out its new course when it comes across an obstruction that prevents it from moving ahead. This dancing behavior is described as follows:

$$x_i(t+1) = x_i(t) + \tan(\theta)|x_i(t) - x_i(t-1)| \tag{2}$$

from $|x_i(t) - x_i(t-1)|$ in the above formula, we can see that the position update of dung beetles is affected by current and historical data. Where $\theta \in [0, \pi]$, it is important to note that the dung beetle's location will not be updated if $\theta$ is 0, $\pi/2$ or $\pi$.

### 2.1.2 Brood Balls

Dung beetles roll their balls to a secure spot, conceal them, and then lay their eggs to create a safer habitat for their larvae. In order to replicate the locations where female dung beetles deposit their eggs, the following boundary selection approach is suggested:

$$Lb^* = max(X^* \times (1-R),\ Lb),$$
$$Ub^* = min(X^* \times (1+R),\ Ub) \tag{3}$$

where $X^*$ represents the local optimal value, the problem's lower and upper limits are denoted by $Lb$ and $Ub$, respectively. $R = 1 - t/Tmax$, and $Tmax$ is the maximum number of iterations. $Lb^*$ and $Ub^*$ are the lower and upper boundaries of the spawning area, respectively. As can be seen from the above formula, the spawning boundary is dynamically determined by the $R$ value, so the position of the brood ball is also dynamically updated, as shown below:

$$X_i(t+1) = X^* + b1 \times (X_i(t) - Lb^*) + b2 \times (X_i(t) - Ub^*) \tag{4}$$

where $b1$ and $b2$ represent two independent $1 \times D$ random vectors, $D$ is the dimension of the optimization problem.

### 2.1.3 Small Dung Beetles

The ideal foraging region is first determined to direct tiny dung beetles to forage, and the foraging border is specified as follows in order to mimic the foraging behavior of these insects:

$$Lb^b = max(X^b \times (1 - R),\ Lb),$$
$$Ub^b = min(X^b \times (1 + R),\ Ub) \tag{5}$$

where $X^b$ represents the global optimal position, $Lb^b$ and $Ub^b$ represent the lower and upper optimal foraging boundary, respectively. Thus, the location of the little dung beetle is updated as follows:

$$x_i(t+1) = x_i(t) + C1 \times (x_i(t) - Lb^b) + C2 \times (x_i(t) - Ub^b) \tag{6}$$

where $C1$ is a random number that follows a normal distribution, $C2$ represents a random number belonging to (0, 1).

### 2.1.4 Stealing Dung Beetles

Certain dung beetles in the wild take dung balls from other dung beetles. The following is an updated position of the thief:

$$x_i(t+1) = X^b + S \times g \times (|x_i(t) - X^*| + |x_i(t) - X^b|) \tag{7}$$

where $S$ is a constant, $g$ is a $1 \times D$ random vector.

## 2.2 Salp Swarm Algorithm (SSA)

In 2017, Mirjalili et al. introduced a novel intelligent optimization technique called the Salp Swarm Algorithm (SSA) [17]. SSA, a relatively new swarm intelligence optimization method, mimics the swarm behavior of salps traveling and feeding in a chain in the deep sea. They often exhibit their chain activity as discrete units that are joined end to end and move sequentially. The first person in the salp chain is the only one who is considered the leader; the others are followers. A leader directs the followers to walk in a chain behavior toward food at each iteration. The situation of slipping into local optimum is much reduced while traveling since the leader performs global exploration while the following completely conducts local exploitation. The leader's position is updated as follows:

$$x_j^1 = \begin{cases} F_j + c_1((ub_j - lb_j) \times c_2 + lb_j)c_3 \le 0 \\ F_j - c_1((ub_j - lb_j) \times c_2 + lb_j)c_3 > 0 \end{cases} \tag{8}$$

where $F_j$ represents the position of the food, $ub_j$ and $lb_j$ are the upper and lower bounds, respectively, $c_2$ and $c_3$ are two random number between [0, 1], $x_j^1$ is the position of the leader. $c_1$ is a very important parameter in the algorithm, which controls the balance between global exploration and local exploitation, and its expression is as follows:

$$c_1 = 2e^{-\left(\frac{4t}{t_{max}}\right)^2} \tag{9}$$

where $t$ is the current iteration number, $t_{max}$ is the maximum iteration. After the leader's position is updated, the followers' position is updated as follows:

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}) \tag{10}$$

where $x_j^i$ is the position of the $j$-th individual in the $i$-th iteration, and the value of $i$ is greater than 1.

## 3 The Proposed Algorithm

The FS problem is a binary problem with an upper bound of 1 and a lower bound of 0. Obviously, the continuous DBO cannot deal with this problem, so we convert the continuous algorithm into binary form through Eq. (11) [18,19], so as to search in the solution space.

$$x_i(t+1) = \begin{cases} 1 & if \dfrac{1}{1+e^{\left(-10*\left(x_i(t)-0.5\right)\right)}} > r \\ 0 & otherwise \end{cases} \tag{11}$$

where $r$ is the random number in [0, 1].

To address the imbalance between exploration and exploitation and the tendency for local optima to easily stagnate in the current DBO, this paper adopts the following strategies to improve the original DBO.

### *3.1 Chaotic Mapping*

Chaos is an aperiodic phenomenon with asymptotic self-similarity and orderliness. Due to its unique randomness, ergodicity, and complexity, it is widely used as a global optimization processing mechanism to effectively avoid the dilemma of local optima in the process of data search optimization in the field of decision system design [20]. In recent years, chaotic strategies have been widely used in intelligence optimization algorithms.

In order to solve problems such as low population diversity and unsatisfactory search results in the random initialization stage of DBO, the Bernoulli chaotic mapping method is introduced. It makes the dung beetle populations traverse solution space better in the initialization phase, and improves the overall optimal search performance. The histogram of the Bernoulli chaotic mapping sequence is depicted in Fig. 2, and its expression is as follows:

$$x_{i+1} = \begin{cases} x_i/(1-a), & x_k \in (0, 1-a] \\ (x_i - 1 + a)/a, & x_i \in (1-a,\ 1) \end{cases} \tag{12}$$

where $a$ is control factor, 0.479 is used to achieve better mapping results in this paper.

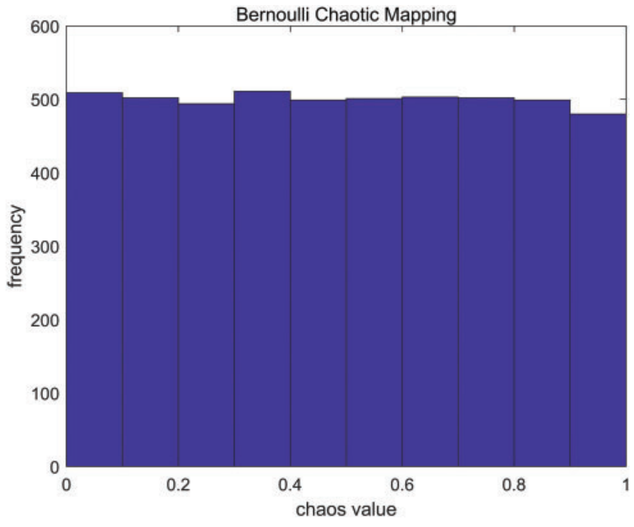

**Figure 2:** Histogram of the Bernoulli chaotic mapping sequence

### 3.2 *Improved Ball-Rolling Dung Beetles*

The ball-rolling dung beetle plays an important role in the DBO, and its updated position has a crucial impact on the global exploration of the algorithm in the solution space. Moreover, SSA has a unique leader and follower mechanism, strong global exploration ability, and a simple structure to implement. Thus, in order to enhance the algorithm's capacity for global exploration, we include SSA into DBO to update the location of the ball-rolling dung beetle. Simultaneously, the original SSA is improved upon and the Levy flight strategy is introduced to the leader position update in order to disrupt the ideal position, hence facilitating a larger exploration of the solution space by the algorithm. Levy flight strategy [21] is a random behavior strategy used to simulate step size and direction during a random walk or search. The leader's position in the improved SSA is updated as follows:

$$x_j^1 = \begin{cases} Levy. * F_j + c_1((ub_j - lb_j) \times c_2 + lb_j)c_3 \leq 0 \\ Levy. * F_j - c_1((ub_j - lb_j) \times c_2 + lb_j)c_3 > 0 \end{cases} \tag{13}$$

The aforementioned variables have been explained above, where the calculation formula of Levy flight operator is shown as follows:

$$Levy = (s_1,\ s_2, \cdots, s_n), s_i = \frac{\mu}{|v|^{\frac{1}{\lambda}}}, i = 1,\ 2, \cdots, n \tag{14}$$

where $v$ follows the standard normal distribution (the mean is 0 and the variance is 1), $\mu$ follows Gaussian distribution (the mean is 0 and the variance is $\sigma_\mu$), and $\sigma_\mu$ is calculated as Eq. (15):

$$\sigma_\mu = \left[\frac{\Gamma(1+\lambda) \times sin\left(\pi \times \frac{\lambda}{2}\right)}{\Gamma\left(\frac{1+\lambda}{2}\right) \times \lambda \times 2^{\frac{\lambda-1}{2}}}\right]^{\frac{1}{\lambda}} \tag{15}$$

where λ falls in the (1, 3), and the value in this paper is 1.5, Γ(x) is the gamma function.

### 3.3 *Mutation Operator*

The behavior of the stolen dung beetle is precisely around the dung ball (global optimal location), which aims to address the issue that DBO is prone to be trapped in local optima. As a result, it plays a significant role in the capacity of DBO to exploit situations locally. In order to prevent the stealing dung beetle from stopping forward search due to finding the local optimal position, a mutation operator is added to the position update of the stealing dung beetle. After each stealing dung beetle position update is completed, mutation operation is carried out with a certain probability $p$, which can effectively ensure that the DBO is not easily trapped in local optima, and enhance the local exploitation ability of the DBO. When $p >$ rand, the inverse operation is performed on the current individual, otherwise it remains unchanged, as shown in Fig. 3. The probability $p$ is calculated as follows:

$$p = 0.9 + \frac{-0.9 * (t-1)}{M-1} \tag{16}$$

where $t$ and $M$ represent the current and maximum iterations, respectively.

$$1\ 0\ 1\ 0\ 1\ 0\ 1 \rightarrow 1\ 0\ 0\ 0\ 1\ 0\ 1$$

**Figure 3:** Mutation of stealing dung beetle in DBO

By enhancing the DBO using the aforementioned tactics, the algorithm's capacity to break out of local optima is effectively increased, its scope for global exploration is broadened, and a healthy balance between its capabilities for local exploitation and global exploration is struck. The flow chart applied to FS specifically is shown in Fig. 4. The following is the specific process of HBCSSDBO feature selection methods:

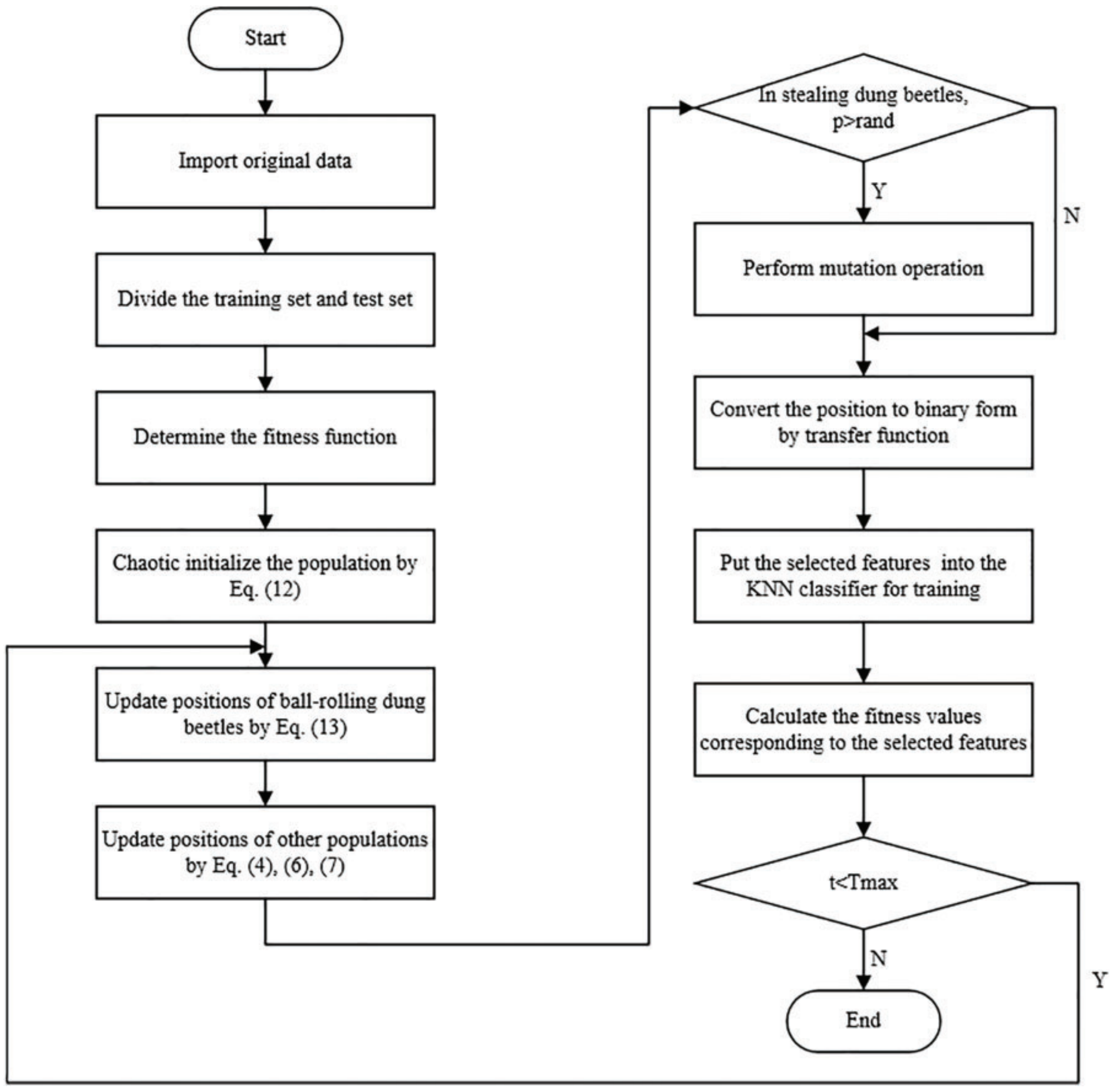
Start
Import original data
Divide the training set and test set
Determine the fitness function
Chaotic initialize the population by Eq. (12)
Update positions of ball-rolling dung beetles by Eq. (13)
Update positions of other populations by Eq. (4), (6), (7)
In stealing dung beetles, p>rand
Y
N
Perform mutation operation
Convert the position to binary form by transfer function
Put the selected features into the KNN classifier for training
Calculate the fitness values corresponding to the selected features
t<Tmax
N
Y
End

**Figure 4:** Flow chart

1. Import the data and divide it into training and test sets. Determine the fitness function and the parameter initialization settings.

2. Initialization of chaotic mapping for dung beetle populations according to Eq. (12).
3. Using an enhanced SSA to update a rolling dung beetle's location within a DBO.
4. The $p$-value at the moment of the stealing dung beetle's position update determined whether to carry out a mutation operation. The locations of breeding dung beetles, tiny dung beetles, and stealing dung beetles were updated in accordance with Eqs. (4), (6), and (7), respectively.
5. The characteristics that are iteratively chosen are supplied into the KNN classifier for training after the dung beetle population is binarized based on the transfer function.
6. Calculate the fitness value corresponding to the selected feature, if $t < \text{Tmax}$, then return to step three.
7. Output results.

### *3.4 Fitness Function*

Choosing the lowest feature subset from the original features and increasing classification accuracy are the two goals of the multi-objective optimization problem known as the FS problem. Based on the foregoing, the fitness function of the FS issue is determined to be as follows in order to attain a balance between the two objectives [22].

$$fitness = \alpha\rho_R(D) + \beta\frac{|X|}{|N|} \tag{17}$$

where $\rho_R(D)$ represents the classification error rate by KNN classifier, $|X|$ represents the number of feature subsets obtained from algorithm optimization, the total number of features in the dataset is represented by the symbol $|N|$. $\alpha$ and $\beta$ are used to achieve the balance between classification accuracy and the number of feature subsets, and $\alpha \in [0, 1]$, $\beta = 1 - \alpha$.

## 4 Experimental Results

### *4.1 Datasets*

The performance of the HBCSSDBO method for FS is verified in this study using 16 standard datasets that are all taken from the UCI Machine Learning Repository [23]. Table 1 displays each dataset's parameters, which are categorized into three groups based on the number of features in each dataset: low, medium, and high dimensions.

**Table 1:** Datasets parameters

| No. | Dataset | No. of instances | No. of features | Dimension |
|---|---|---|---|---|
| 1 | Breastcancer | 699 | 9 | Low |
| 2 | BreastEW | 569 | 30 | High |
| 3 | CongressEW | 435 | 16 | Medium |
| 4 | Exactly | 1000 | 13 | Low |
| 5 | Exactly2 | 1000 | 13 | Low |
| 6 | HeartEW | 270 | 13 | Low |
| 7 | IonosphereEW | 351 | 34 | High |
| 8 | KrvskpEW | 3196 | 36 | High |
| 9 | M-of-n | 1000 | 13 | Low |
| 10 | SonarEW | 208 | 60 | High |

(Continued)

**Table 1 (continued)**

| No. | Dataset | No. of instances | No. of features | Dimension |
|---|---|---|---|---|
| 11 | SpectEW | 267 | 22 | Medium |
| 12 | Tic-tac-toe | 958 | 9 | Low |
| 13 | Vote | 300 | 16 | Medium |
| 14 | WaveformEW | 5000 | 40 | High |
| 15 | WineEW | 178 | 13 | Low |
| 16 | Zoo | 101 | 16 | Medium |

### *4.2 Parameter Settings*

One supervised learning technique that classifies new instances based on how far they are from the training set is the K nearest neighbor (KNN) approach [24]. It is widely used in pattern recognition, artificial intelligence, and other fields, so this paper selects KNN as a classifier combined with HBCSSDBO to apply to the FS problem and K = 5. Experiments have shown that when K is set to 5, better classification results can be obtained on numerous datasets [25]. By applying K-fold cross-validation, the dataset is divided into K segments. The test set is comprised of one segment, while the training set is made up of the other (K-1) segment, whose data segments are randomly shuffled.

Intel(R) CoreTMi7-6700 machine with 3.4 GHz CPU (Central Processing Unit) and 8 GB of RAM (random access memory) was used to run the HBCSSDBO algorithm and other comparative algorithms. The comparison algorithms include the original DBO [16], SSA [17], hybrid Gray Wolf Particle Swarm Optimization (BGWOPSO) [26], and Gravity Search Algorithm (GSA) [27]. The parameters in each algorithm can be found in Table 2, and the comparison algorithm is converted into binary form by Eq. (11) to ensure the comparability and fairness of the experiment to the greatest extent possible. It is noteworthy that every algorithm's population number in this work is set consistently at 30, the number of iterations at 100, the upper and lower bounds of the search at 1, and the number of independent operations at 10. For the value of the weight parameter $\alpha$ in the fitness function is 0.99 [28].

**Table 2:** The configuration parameters of different methods

| Methods | Parameters | Values |
|---|---|---|
| HBCSSDBO | $c_2$, $c_3$, $s$ | [0, 1], [0, 1], 0.5 |
| DBO | $k$, $b$, $s$ | 0.1, 0.3, 0.5 |
| SSA | $c_2$, $c_3$ | [0, 1], [0, 1] |
| BGWOPSO | $c_1$, $c_2$, $c_3$ | 0.5 |
| GSA | $G_0$ | 100 |

### *4.3 Experimental Evaluation Indices*

To validate the suggested approach, a variety of evaluation indices are used in this study, including average classification accuracy (in %), mean fitness value, best and worst fitness values, fitness value variation, average number of picked features, and average computational time. The following formula is used to determine the index:

(1) Average classification accuracy

$$Acc = \frac{1}{N}\sum\nolimits_{k=1}^{N} F_K \tag{18}$$

where $F_K$ is the highest classification accuracy for the $K$-th run, $N$ is the number of runs.

(2) Mean fitness value

$$Fit = \frac{1}{N}\sum\nolimits_{k=1}^{N} F_K^* \tag{19}$$

where $F_K^*$ is the best fitness value for the $K$-th run.

(3) The best fitness value

$$Max = \min_{1 \le K \le N_r} F_K^* \tag{20}$$

(4) The worst fitness value

$$Min = \max_{1 \le K \le N_r} F_K^* \tag{21}$$

(5) Fitness value variance

$$STD = \sqrt{\frac{1}{N-1}\sum\nolimits_{K=1}^{N}(F_K^* - Fit)^2} \tag{22}$$

(6) Average number of selected features

$$SIZE = \frac{1}{N}\sum\nolimits_{k=1}^{N} T_K^* \tag{23}$$

where $T_K^*$ is the number of selected features at the $K$-th run.

(7) Average computational time

$$Avgtime = \frac{1}{N}\sum\nolimits_{k=1}^{N} time_K \tag{24}$$

where $time_K$ is the time of running the algorithm for the $K$-th time.

## 5 Experimental Results and Discussion

The outcomes of using the HBCSSDBO algorithm on 16 datasets are displayed in Table 3. The computed time, mean fitness, average accuracy (in%), and feature selection are among the outcomes. Ten runs of the method were performed for each dataset. Among these, the algorithm gets the best accuracy of 100% on the datasets M-of-n and Exactly. On the WineEW and Zoo datasets, it also achieves more than 99% accuracy, followed by more than 98% accuracy on the Breastcancer and CongressEW datasets. With respect to the total number of characteristics picked, the BreastEW dataset has the fewest features chosen—just three. Secondly, KrvskpEW and WaveformEW have the highest amount of features picked (13.9 and 17.6, respectively), while WineEW has the fewest features selected (3.3). In terms of computational time, WineEW, SonarEW, and SpectEW datasets spend the least time, with 13.73, 13.76, and 13.78 s, respectively.

**Table 3:** The optimization results of the HBCSSDBO algorithm

| No. | Dataset | Average accuracy (%) | Mean fitness | Feature selected | Computational time (s) |
|---|---|---|---|---|---|
| 1 | Breastcancer | 98.92 | 0.02 | 4.20 | 15.43 |
| 2 | BreastEW | 96.90 | 0.03 | 3.00 | 15.69 |
| 3 | CongressEW | 98.39 | 0.02 | 5.30 | 14.60 |
| 4 | Exactly | 100.00 | 0.00 | 6.00 | 16.03 |
| 5 | Exactly2 | 78.65 | 0.22 | 6.90 | 16.17 |
| 6 | HeartEW | 90.56 | 0.10 | 4.00 | 14.42 |
| 7 | IonosphereEW | 96.71 | 0.03 | 4.50 | 15.04 |
| 8 | KrvskpEW | 98.69 | 0.02 | 13.90 | 32.05 |
| 9 | M-of-n | 100.00 | 0.00 | 6.00 | 16.37 |
| 10 | SonarEW | 97.56 | 0.03 | 8.10 | 13.76 |
| 11 | SpectEW | 89.81 | 0.10 | 6.10 | 13.78 |
| 12 | Tic-tac-toe | 84.82 | 0.16 | 7.00 | 15.85 |
| 13 | Vote | 97.83 | 0.02 | 4.00 | 14.04 |
| 14 | WaveformEW | 86.02 | 0.14 | 17.60 | 58.34 |
| 15 | WineEW | 99.43 | 0.01 | 3.30 | 13.73 |
| 16 | Zoo | 99.50 | 0.01 | 5.20 | 14.06 |
|  | Average | 94.61 | 0.06 | 6.57 | 18.71 |

The original DBO and the original SSA are compared, respectively, to confirm the efficacy of fusion, and this process is done to validate the hybrid algorithm's usefulness. To confirm the algorithm's improved performance, the hybrid Gray Wolf Particle Swarm Optimization method and the Gravity Search algorithm are chosen for comparison. The running results of each algorithm are shown in Tables 4 to 10. The results are average accuracy, average number of selected features, average fitness value, average best fitness value, average worst fitness value, variance of fitness value, and average computational time, respectively, and the optimal value is bolded. Each algorithm runs independently 10 times.

As shown in Table 4, the classification accuracy of HBCSSDBO on 14 datasets is higher than that of other comparative algorithms, and the average classification accuracy on all datasets is as high as 94.61%, which is 1.18%, 1.09%, 0.88%, and 0.88% higher than that of BDBO (Binary Dung Beetle Optimization), BSSA (Binary Salp Swarm Algorithm), BGWOPSO (Binary Grey Wolf Optimization Particle Swarm Optimization), and BGSA (Binary Gravitational Search Algorithm), respectively. The proposed hybrid method, original DBO, and SSA all have 100% classification accuracy on the M-of-n dataset; on the Vote dataset, the hybrid approach and SSA have 97.83% classification accuracy. In addition to the above two datasets, the classification accuracy of the HBCSSDBO is superior to the two original algorithms. The feasibility and effectiveness of the hybrid algorithm are proven. The accuracy of HBCSSDBO, BGWOPSO, and BGSA is 100% on Exactly and M-of-n datasets, and the proposed algorithm only lags behind these two algorithms on Tic-tac-toe and Vote datasets. To sum up, HBCSSDBO has higher classification accuracy.

**Table 4:** Average classification accuracy of each algorithm (in %)

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | **98.92** | 97.99 | 98.27 | 98.27 | 98.42 |
| BreastEW | **96.90** | 96.55 | 96.02 | 95.13 | 95.66 |
| CongressEW | **98.39** | 97.24 | 97.93 | 98.05 | 98.16 |
| Exactly | **100.00** | 97.15 | 96.95 | **100.00** | **100.00** |
| Exactly2 | **78.65** | 77.55 | 77.85 | 78.00 | 78.15 |
| HeartEW | **90.56** | 87.41 | 88.70 | 87.41 | 87.41 |
| IonosphereEW | **96.71** | 96.29 | 94.71 | 94.43 | 93.43 |
| KrvskpEW | **98.69** | 98.34 | 98.15 | 98.64 | 98.50 |
| M-of-n | **100.00** | **100.00** | **100.00** | **100.00** | **100.00** |
| SonarEW | **97.56** | 96.59 | 96.59 | 95.85 | 95.61 |
| SpectEW | **89.81** | 88.30 | 88.68 | 88.87 | 89.43 |
| Tic-tac-toe | 84.82 | 84.14 | 83.82 | **85.45** | 83.66 |
| Vote | 97.83 | 97.50 | 97.83 | 98.00 | **98.33** |
| WaveformEW | **86.02** | 85.34 | 84.81 | 85.58 | 85.99 |
| WineEW | **99.43** | 98.00 | 97.43 | 98.00 | 98.86 |
| Zoo | **99.50** | 96.50 | 98.50 | 98.00 | 98.00 |
| Average | **94.61** | 93.43 | 93.52 | 93.73 | 93.73 |

Table 5 shows that, out of all the comparable methods, the suggested approach ranks highest with an average of 6.57 selected features across all datasets. Its number of selected features on only 5 datasets lags behind other comparative algorithms and is in a leading position on the other 11 datasets. Combined with the analysis of the FS accuracy, it has been demonstrated that the suggested algorithm performs better while using FS.

**Table 5:** Average number of selected features of each algorithm

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | 4.20 | **3.50** | 3.8 | 4.30 | 4.60 |
| BreastEW | **3.00** | 3.50 | 3.60 | 4.20 | 8.70 |
| CongressEW | 5.30 | 4.80 | **4.50** | 5.50 | 5.10 |
| Exactly | **6.00** | 6.20 | 6.10 | **6.00** | **6.00** |
| Exactly2 | 6.90 | **5.90** | 6.50 | 6.50 | 6.20 |
| HeartEW | **4.00** | **4.40** | 4.40 | 4.60 | 4.70 |
| IonosphereEW | **4.50** | **4.50** | 8.10 | 10.50 | 10.20 |
| KrvskpEW | **13.90** | 15.60 | 18.40 | 21.00 | 20.80 |
| M-of-n | **6.00** | **6.00** | **6.00** | **6.00** | **6.00** |
| SonarEW | **8.10** | 9.00 | 16.70 | 20.80 | 26.00 |
| SpectEW | **6.10** | 7.50 | 9.10 | 8.80 | 9.60 |

(Continued)

**Table 5 (continued)**

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Tic-tac-toe | 7.00 | 7.10 | **6.30** | 7.90 | 7.10 |
| Vote | 4.00 | **3.60** | 5.20 | 5.60 | 5.50 |
| WaveformEW | **17.60** | 22.20 | 22.20 | 27.80 | 23.50 |
| WineEW | **3.30** | 3.60 | 4.30 | 4.20 | 3.80 |
| Zoo | **5.20** | 5.80 | 6.30 | 6.20 | 6.60 |
| Average | **6.57** | 7.03 | 8.19 | 9.37 | 9.65 |

When comparing the average fitness, the best and worst fitness, and the variance of different algorithms, the results are displayed using scientific counting, and the optimal results are bolded in order to prevent data differences from being too minor and challenging to see. Table 6 shows that compared to other comparison algorithms, HBCSSDBO has a higher average fitness value across all datasets. Based on the aforementioned analysis, the proposed algorithm shows stronger optimization ability. However, in Table 7, the best fitness value found by the HBCSSDBO on all datasets is superior to other algorithms, which shows that the algorithm has a stronger optimization breadth and verifies the improvement of the global exploration ability in the algorithm improvement strategy. Moreover, this is still the case in Table 8, where the worst fitness value found by the HBCSSDBO is smaller than other algorithms on 12 datasets. It is proved that the HBCSSDBO is less prone to stagnate in local optima. Combined with the variance of fitness value in Table 9, the variance obtained on half of the datasets is smaller than other algorithms. For datasets Exactly, KrvskpEW, SonarEW, WineEW, etc., although the variance of the proposed algorithm is not optimal, it is still at the forefront of many algorithms, which shows that HBCSSDBO has better robustness.

**Table 6:** Mean fitness value of each algorithm

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | **1.535E-02** | 2.383E-02 | 2.132E-02 | 2.187E-02 | 2.078E-02 |
| BreastEW | **3.166E-02** | 3.533E-02 | 4.062E-02 | 4.959E-02 | 4.583E-02 |
| CongressEW | **1.924E-02** | 3.031E-02 | 2.330E-02 | 2.278E-02 | 2.139E-02 |
| Exactly | **4.615E-03** | 3.298E-02 | 3.489E-02 | **4.615E-03** | **4.615E-03** |
| Exactly2 | **2.167E-01** | 2.268E-01 | 2.243E-01 | 2.228E-01 | 2.211E-01 |
| HeartEW | **9.658E-02** | 1.281E-01 | 1.152E-01 | 1.282E-01 | 1.283E-01 |
| IonosphereEW | **3.385E-02** | 3.809E-02 | 5.471E-02 | 5.825E-02 | 6.806E-02 |
| KrvskpEW | **1.688E-02** | 2.076E-02 | 2.339E-02 | 1.931E-02 | 2.065E-02 |
| M-of-n | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** |
| SonarEW | **2.550E-02** | 3.530E-02 | 3.659E-02 | 4.452E-02 | 4.780E-02 |
| SpectEW | **1.036E-01** | 1.192E-01 | 1.162E-01 | 1.142E-01 | 1.090E-01 |
| Tic-tac-toe | 1.581E-01 | 1.649E-01 | 1.672E-01 | **1.529E-01** | 1.696E-01 |
| Vote | 2.395E-02 | 2.700E-02 | 2.470E-02 | 2.330E-02 | **1.994E-02** |
| WaveformEW | **1.428E-01** | 1.507E-01 | 1.559E-01 | 1.497E-01 | 1.446E-01 |
| WineEW | **8.196E-03** | 2.257E-02 | 2.876E-02 | 2.303E-02 | 1.424E-02 |

(Continued)

**Table 6 (continued)**

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Zoo | **8.200E-03** | 3.828E-02 | 1.879E-02 | 2.368E-02 | 2.393E-02 |
| Average | **5.687E-02** | 6.867E-02 | 6.816E-02 | 6.646E-02 | 6.652E-02 |

**Table 7:** The best fitness value of each algorithm

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | **1.046E-02** | 1.157E-02 | **1.046E-02** | **1.046E-02** | 5.556E-03 |
| BreastEW | 9.761E-03 | 1.852E-02 | **1.000E-03** | 1.819E-02 | 2.052E-02 |
| CongressEW | 1.875E-03 | 1.325E-02 | **6.250E-04** | **6.250E-04** | 2.500E-03 |
| Exactly | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** |
| Exactly2 | **2.049E-01** | 2.106E-01 | 2.175E-01 | 2.099E-01 | 2.133E-01 |
| HeartEW | **5.808E-02** | 5.885E-02 | **5.808E-02** | 9.397E-02 | 5.962E-02 |
| IonosphereEW | 1.532E-02 | **1.176E-03** | 1.650E-02 | 3.123E-02 | 3.123E-02 |
| KrvskpEW | 1.191E-02 | 1.219E-02 | 1.358E-02 | 1.696E-02 | **1.175E-02** |
| M-of-n | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** |
| SonarEW | 1.167E-03 | **8.333E-04** | 2.500E-03 | 3.667E-03 | 3.667E-03 |
| SpectEW | **5.922E-02** | 7.790E-02 | 7.926E-02 | 7.972E-02 | 6.058E-02 |
| Tic-tac-toe | 1.374E-01 | 1.344E-01 | 1.188E-01 | 1.240E-01 | **1.137E-01** |
| Vote | 2.500E-03 | **6.250E-04** | 2.500E-03 | 2.500E-03 | 2.500E-03 |
| WaveformEW | **1.238E-01** | 1.414E-01 | 1.443E-01 | 1.350E-01 | 1.320E-01 |
| WineEW | **1.538E-03** | **1.538E-03** | 2.308E-03 | **1.538E-03** | 2.308E-03 |
| Zoo | **3.125E-03** | 3.750E-03 | **3.125E-03** | **3.125E-03** | **3.125E-03** |
| Average | **4.064E-02** | 4.349E-02 | 4.249E-02 | 4.625E-02 | 4.197E-02 |

**Table 8:** The worst fitness value of each algorithm

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | **2.581E-02** | 4.006E-02 | 3.182E-02 | 4.940E-02 | 4.940E-02 |
| BreastEW | 6.233E-02 | **5.323E-02** | 7.985E-02 | 7.985E-02 | 7.342E-02 |
| CongressEW | 4.864E-02 | 5.877E-02 | 6.190E-02 | 5.052E-02 | **2.776E-02** |
| Exactly | **4.615E-03** | 2.883E-01 | 2.917E-01 | **4.615E-03** | **4.615E-03** |
| Exactly2 | **2.315E-01** | 2.433E-01 | 2.365E-01 | 2.433E-01 | **2.315E-01** |
| HeartEW | **1.314E-01** | 1.704E-01 | 1.688E-01 | 1.887E-01 | 1.872E-01 |
| IonosphereEW | **5.834E-02** | **5.834E-02** | 1.014E-01 | 1.019E-01 | 1.164E-01 |
| KrvskpEW | **2.320E-02** | 3.289E-02 | 4.036E-02 | 2.709E-02 | 2.935E-02 |
| M-of-n | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** | **4.615E-03** |
| SonarEW | **7.344E-02** | 9.775E-02 | 7.527E-02 | 7.694E-02 | 1.013E-01 |
| SpectEW | 1.704E-01 | 1.882E-01 | 1.531E-01 | **1.358E-01** | 1.535E-01 |

(Continued)

**Table 8 (continued)**

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Tic-tac-toe | **1.829E-01** | 2.036E-01 | 1.922E-01 | 1.914E-01 | 1.922E-01 |
| Vote | 6.788E-02 | 5.200E-02 | 5.388E-02 | 7.038E-02 | **3.988E-02** |
| WaveformEW | **1.547E-01** | 1.793E-01 | 1.686E-01 | 1.602E-01 | 1.619E-01 |
| WineEW | **3.136E-02** | 5.888E-02 | 6.042E-02 | 5.965E-02 | 5.811E-02 |
| Zoo | **5.200E-02** | 1.021E-01 | 1.015E-01 | 5.388E-02 | 5.450E-02 |
| Average | **8.270E-02** | **1.145E-01** | **1.139E-01** | **9.364E-02** | **9.285E-02** |

**Table 9:** Variance of adaptation values for each algorithm

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | **5.090E-03** | 1.113E-02 | 8.035E-03 | 1.360E-02 | 1.239E-02 |
| BreastEW | 1.669E-02 | **7.226E-03** | 1.945E-02 | 1.659E-02 | 1.888E-02 |
| CongressEW | 1.655E-02 | 1.721E-02 | 1.850E-02 | 1.476E-02 | **8.155E-03** |
| Exactly | **9.143E-19** | 8.971E-02 | 9.037E-02 | **9.143E-19** | **9.143E-19** |
| Exactly2 | 7.619E-03 | 1.217E-02 | **5.533E-03** | 9.600E-03 | 6.656E-03 |
| HeartEW | **2.334E-02** | 3.207E-02 | 3.179E-02 | 3.683E-02 | 4.328E-02 |
| IonosphereEW | **1.772E-02** | 1.898E-02 | 2.490E-02 | 1.928E-02 | 2.415E-02 |
| KrvskpEW | 3.799E-03 | 6.492E-03 | 8.353E-03 | **2.818E-03** | 5.216E-03 |
| M-of-n | **9.143E-19** | **9.143E-19** | **9.143E-19** | **9.143E-19** | **9.143E-19** |
| SonarEW | 2.532E-02 | 3.454E-02 | 3.055E-02 | **2.311E-02** | 3.569E-02 |
| SpectEW | 3.416E-02 | 3.109E-02 | 2.496E-02 | **2.044E-02** | 3.716E-02 |
| Tic-tac-toe | **1.531E-02** | 2.308E-02 | 2.116E-02 | 2.295E-02 | 2.243E-02 |
| Vote | 1.885E-02 | 1.811E-02 | 1.889E-02 | 2.682E-02 | **1.601E-02** |
| WaveformEW | 9.910E-03 | 1.107E-02 | 8.185E-03 | **7.578E-03** | 9.770E-03 |
| WineEW | **1.203E-02** | 2.298E-02 | 2.092E-02 | 1.919E-02 | 1.928E-02 |
| Zoo | **1.540E-02** | 3.274E-02 | 3.291E-02 | 2.559E-02 | 2.565E-02 |
| Average | **1.386E-02** | 2.304E-02 | 2.278E-02 | 1.620E-02 | 1.779E-02 |

Table 10 illustrates this, even though the HBCSSDBO only obtains a faster time on six datasets., such as Exactly, KrvskpEW, SpectEW, Vote, WaveformEW, and WineEW, the average time of HBCSSDBO running in 16 datasets is ahead of other algorithms. Furthermore, in other datasets, the computing speed of HBCSSDBO is also at the forefront of many algorithms. The hybrid algorithm is composed of the original DBO and the original SSA, but it does not consume more computing time, which proves the rationality of the proposed algorithm.

The suggested technique performs better than other comparable algorithms in each of the aforementioned seven assessment indices. When compared to the BDBO feature selection approach, the HBCSSDBO feature selection method improves the average prediction accuracy and average number of chosen features on the entire dataset by 1.18% and 7.2%, respectively. It is evident from the aforementioned study of the average classification accuracy and the average number of features chosen that the suggested approach is capable of selecting the best subset of characteristics while also guaranteeing a high classification accuracy. The suggested method has significantly improved in terms of depth of optimization search and exploration ability as well as robustness, according to the examination of the variance, the best fitness value, and the optimal fitness. These results justify the fusion algorithm, i.e., the embedded SSA algorithm leader and follower mechanisms effectively help the dung beetle individuals better explore the solution space, while the mutation operator and chaotic initialization mapping population ensure that the algorithm does not easily fall into the local optimum and guarantee the diversity of the population, respectively. Furthermore, the hybrid algorithm is guaranteed to be efficient because the suggested approach does not result in any extra time overhead when compared to the two original methods in terms of running time.

**Table 10:** Computational time of each algorithm

| Dataset | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | 15.43 | 15.10 | 16.78 | 17.48 | **14.10** |
| BreastEW | 15.69 | 16.14 | **14.86** | 15.67 | 15.10 |
| CongressEW | 14.60 | **14.55** | 15.37 | 15.09 | 14.60 |
| Exactly | **16.03** | 16.45 | 17.75 | 16.88 | 16.39 |
| Exactly2 | 16.17 | **15.83** | 16.97 | 17.05 | 16.35 |
| HeartEW | 14.42 | 14.45 | 14.49 | 14.66 | **14.26** |
| IonosphereEW | 15.04 | 15.90 | 15.33 | 14.57 | **13.78** |
| KrvskpEW | **32.05** | 34.24 | 37.72 | 38.94 | 33.91 |
| M-of-n | 16.37 | 16.26 | 17.22 | 16.88 | **13.84** |
| SonarEW | 13.76 | 14.35 | 14.80 | 15.17 | **13.43** |
| SpectEW | **13.78** | 14.37 | 15.48 | 14.90 | 15.35 |
| Tic-tac-toe | 15.85 | 16.70 | 16.64 | 17.75 | **13.36** |
| Vote | **14.04** | 14.38 | 14.51 | 14.69 | 14.48 |
| WaveformEW | **58.34** | 69.26 | 66.25 | 88.58 | 66.79 |
| WineEW | **13.73** | 14.34 | 14.41 | 14.35 | 14.02 |
| Zoo | 14.06 | 14.70 | 14.53 | 14.50 | **13.98** |
| Average | **18.71** | **19.81** | **20.19** | **21.70** | **18.98** |

The datasets are split into three datasets of different dimensions based on the varying number of features, namely high dimensional datasets, medium dimensional datasets, and low dimensional datasets, in order to further assess the effectiveness of the suggested approach. Specific corresponding information is given in Table 1. In order to more intuitively verify the superiority of the proposed algorithm, the average classification accuracy (in %) and the average number of selected features obtained by each algorithm are presented in the form of bar charts according to three different dimensions, and the specific results are shown in Figs. 5–7. As can be seen from Fig. 5, on high dimensional datasets, HBCSSDBO is ahead of other comparative algorithms in terms of classification accuracy and number of selected features, which also indicates that HBCSSDBO can use fewer features to obtain higher classification accuracy, showing strong optimization ability. Fig. 6a shows that, for the medium-dimensional datasets, HBCSSDBO's classification accuracy trails BGSA only in the Vote dataset. Fig. 6b shows that, in the Vote and CongressEW datasets, the number of selected features is marginally higher than the original DBO. It is still in the leading position compared with other algorithms. Fig. 7a shows that all methods work well and achieve high classification accuracy for low-dimensional datasets. Still, it is evident that HBCSSDBO outperforms other algorithms in the majority of datasets. However, as Fig. 7b illustrates, the suggested technique does not choose as many features as other algorithms do—at least not as much as it does for large dimensional datasets.

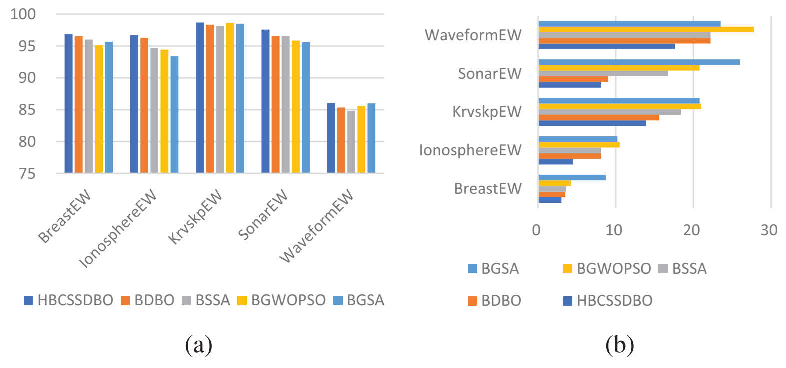

**Figure 5:** Average classification accuracy (a) and number of selected features (b) for high-dimensional datasets

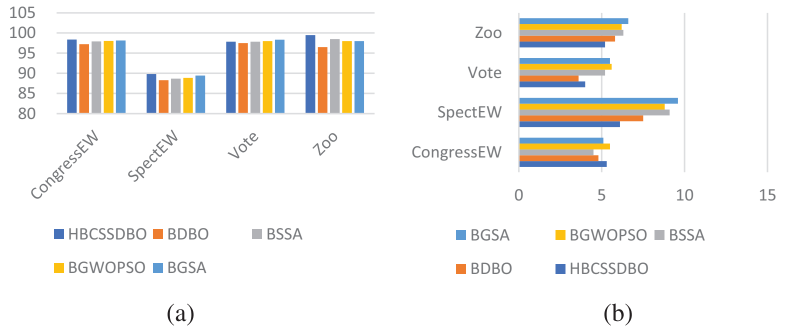

**Figure 6:** Average classification accuracy (a) and number of selected features (b) for medium dimensional datasets

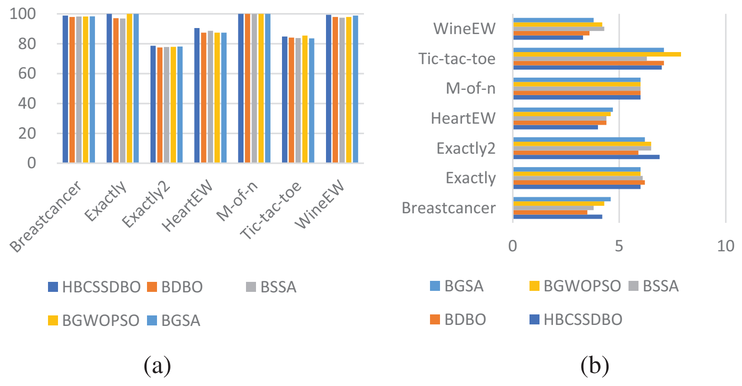

**Figure 7:** Average classification accuracy (a) and number of selected features (b) for low dimensional datasets

To sum up, HBCSSDBO can obtain higher classification accuracy when applied to FS, can extract feature subsets more effectively from original datasets, especially in medium and high dimensional datasets, and does not perform poorly on low dimensional datasets. Overall, it is still ahead of the comparative algorithms.

To assess the statistical significance of the earlier obtained results, the experimental data were subjected to a Friedman test [11]. Where 5% was used as the significance level for this test. Table 11 displays the test's findings.

**Table 11:** The Friedman ranking, statistical test, critical value, and *p*-value of the studied algorithm with BDBO, BSSA, BGWOPSO, and BGSA

| | HBCSSDBO | BDBO | BSSA | BGWOPSO | BGSA |
|---|---|---|---|---|---|
| Breastcancer | 1 | 5 | 3 | 4 | 2 |
| BreastEW | 1 | 2 | 3 | 5 | 4 |
| CongressEW | 1 | 5 | 4 | 3 | 2 |
| Exactly | 1 | 4 | 5 | 2 | 3 |
| Exactly2 | 1 | 5 | 4 | 3 | 2 |
| HeartEW | 1 | 3 | 2 | 4 | 5 |
| IonosphereEW | 1 | 2 | 3 | 4 | 5 |
| KrvskpEW | 1 | 4 | 5 | 2 | 3 |
| M-of-n | 1 | 2 | 3 | 4 | 5 |
| SonarEW | 1 | 2 | 3 | 4 | 5 |
| SpectEW | 1 | 5 | 4 | 3 | 2 |
| Tic-tac-toe | 4 | 1 | 2 | 3 | 5 |
| Vote | 5 | 4 | 1 | 3 | 2 |
| WaveformEW | 1 | 5 | 4 | 2 | 3 |
| WineEW | 1 | 5 | 2 | 4 | 3 |
| Zoo | 1 | 3 | 4 | 5 | 2 |
| Sum | 23 | 57 | 52 | 55 | 53 |
| Average | 1.375 | 3.75 | 3.6875 | 3.125 | 3.0625 |
| F (t) | 25.57 | | | | |
| *p*-value | 3.87E-05 | | | | |
| Critical value | 9.487729 | | | | |

A pairwise Benjamin-Hochberg post hoc test with adjusted p value using HBCSSDBO as a control algorithm was significant for HBCSSDBO *vs.* BDBO ($p = 0.000107511$, adj $= 0.0125$), HBCSSDBO *vs.* BSSA ($p = 0.000107511$, adj $= 0.025$), HBCSSDBO *vs.* BGSA ($p = 0.001340641$, adj $= 0.0375$) and HBCSSDBO *vs.* BGWOPSO ($p = 0.007526315$, adj $= 0.05$). Specific results are shown in Table 12. Consequently, it is evident from the statistical analysis findings above that there is a substantial difference between the other analyzed methods and the suggested approach.

**Table 12:** Benjamini–Hochberg test

| Algorithm | $p$-value | Rank | adj $\alpha$ |
|---|---|---|---|
| BDBO | 0.0001075 | 1 | 0.0125 |
| BSSA | 0.0001075 | 2 | 0.025 |
| BGSA | 0.0013406 | 3 | 0.0375 |
| BGWOPSO | 0.0075263 | 4 | 0.05 |

## 6 Conclusions

In this paper, HBCSSDBO, a new hybrid algorithm, is proposed to solve the feature selection problem. The method combines the respective features of DBO and SSA, realizes the organic integration of the two algorithms, and effectively solves the feature selection challenge. The average fitness value, average classification accuracy, average number of selected features, variance of fitness value, average running time, and optimal and worst fitness values are the seven-evaluation metrics that are used to assess the performance and stability of the proposed method on 16 UCI datasets. In addition, HBCSSDBO is compared with four meta-heuristic feature selection methods, which include BDBO, BSSA, BGWOPSO, and BGSA. Based on several assessment criteria, the trial findings indicate that HBCSSDBO is the best option. Its average classification accuracy is 94.61%, and it has an average of 6.57 chosen features. In the analysis of the method in terms of the optimal and worst fitness values as well as the variance of the fitness values, we can see that the hybrid method has a great improvement in both the optimization searching accuracy and robustness. In addition, the hybrid method does not incur more time loss in terms of running time. These results show that HBCSSDBO is more competitive in solving the feature selection problem. The final statistical test verifies the significant effectiveness of the method.

In our future research, we will explore the application of the proposed method to the feature selection problem in different domains such as data mining, medical applications, engineering applications, and so on. We will try to combine machine learning algorithms other than KNN to study the performance of the HBCSSDBO method.

**Acknowledgement:** We thank the Liaoning Provincial Department of Education and Shenyang Ligong University for financial support of this paper.

**Funding Statement:** This research was funded by the Short-Term Electrical Load Forecasting Based on Feature Selection and optimized LSTM with DBO which is the Fundamental Scientific Research Project of Liaoning Provincial Department of Education (JYTMS20230189), and the Application of Hybrid Grey Wolf Algorithm in Job Shop Scheduling Problem of the Research Support Plan for Introducing High-Level Talents to Shenyang Ligong University (No. 1010147001131).

**Author Contributions:** The authors confirm contribution to the paper as follows: study conception and design: Wei Liu; data collection: Wei Liu; analysis and interpretation of results: Tengteng Ren; draft manuscript preparation: Tengteng Ren. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** Due to the nature of this research, participants of this study did not agree for their data to be shared publicly, so supporting data is not available.

**Ethics Approval:** This article does not contain any research involving humans or animals.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] T. Dokeroglu, A. Deniz, and H. E. Kiziloz, "A comprehensive survey on recent metaheuristics for feature selection," *Neurocomputing*, vol. 494, pp. 269–296, 2022. doi: 10.1016/j.neucom.2022.04.083.

[2] T. H. Pham and B. Raahemi, "Bio-inspired feature selection algorithms with their applications: A systematic literature review," *IEEE Access*, vol. 11, pp. 43733–43758, 2023. doi: 10.1109/ACCESS.2023.3272556.

[3] N. Khodadadi *et al.*, "BAOA: Binary arithmetic optimization algorithm with K-nearest neighbor classifier for feature selection," *IEEE Access*, vol. 11, pp. 94094–94115, 2023. doi: 10.1109/ACCESS.2023.3310429.

[4] M. Rostami, K. Berahmand, E. Nasiri, and S. Forouzandeh, "Review of swarm intelligence-based feature selection methods," *Eng. Appl. Artif. Intell.*, vol. 100, pp. 104210, 2021. doi: 10.1016/j.engappai.2021.104210.

[5] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997. doi: 10.1109/4235.585893.

[6] X. Song, Y. Zhang, D. Gong, and X. Y. Sun, "Feature selection using bare-bones particle swarm optimization with mutual information," *Pattern Recognit.*, vol. 112, pp. 107804, 2021. doi: 10.1016/j.patcog.2020.107804.

[7] H. Faris *et al.*, "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowl. Based Syst.*, vol. 154, pp. 43–67, 2018. doi: 10.1016/j.knosys.2018.05.009.

[8] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, 2016. doi: 10.1016/j.neucom.2015.06.083.

[9] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," *Appl. Soft Comput.*, vol. 62, pp. 441–453, 2018. doi: 10.1016/j.asoc.2017.11.006.

[10] A. Adamu, M. Abdullahi, S. B. Junaidu, and I. H. Hassan, "An hybrid particle swarm optimization with crow search algorithm for feature selection," *Mach. Learn. Appl.*, vol. 6, pp. 100108, 2021. doi: 10.1016/j.mlwa.2021.100108.

[11] Y. Zhang, Y. H. Wang, D. W. Gong, and X. Y. Sun, "Clustering-guided particle swarm feature selection algorithm for high-dimensional imbalanced data with missing values," *IEEE Trans. Evol. Comput.*, vol. 26, no. 4, pp. 616–630, 2021. doi: 10.1109/TEVC.2021.3106975.

[12] X. F. Song, Y. Zhang, Y. N. Guo, X. Y. Sun, and Y. L. Wang, "Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data," *IEEE Trans. Evol. Comput.*, vol. 24, pp. 882–895, 2020. doi: 10.1109/TEVC.2020.2968743.

[13] R. A. Ibrahim, A. A. Ewees, D. Oliva, M. Abd. Elaziz, and S. Lu, "Improved salp swarm algorithm based on particle swarm optimization for feature selection," *J. Ambient Intell. Humaniz. Comput.*, vol. 10, pp. 3155–3169, 2019. doi: 10.1007/s12652-018-1031-9.

[14] M. Abdel-Basset, W. Ding, and D. El-Shahat, "A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection," *Artif. Intell. Rev.*, vol. 54, no. 1, pp. 593–637, 2021. doi: 10.1007/s10462-020-09860-3.

[15] A. A. Abdelhamid *et al.*, "Innovative feature selection method based on hybrid sine cosine and dipper throated optimization algorithms," *IEEE Access*, vol. 11, pp. 79750–79776, 2023. doi: 10.1109/ACCESS.2023.3298955.

[16] J. Xue and B. Shen, "Dung beetle optimizer: A new meta-heuristic algorithm for global optimization," *J. Supercomput.*, vol. 79, no. 7, pp. 7305–7336, 2023. doi: 10.1007/s11227-022-04959-6.

[17] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, 2017. doi: 10.1016/j.advengsoft.2017.07.002.

[18] S. Wang, Q. Yuan, W. Tan, T. Yang, and L. Zeng, "SCChOA: Hybrid sine-cosine chimp optimization algorithm for feature selection," *Comput. Mater. Contin.*, vol. 77, no. 3, pp. 3057–3075, 2023. doi: 10.32604/cmc.2023.044807.

[19] E. S. M. El-Kenawy, M. M. Eid, M. Saber, and A. Ibrahim, "MbGWO-SFS: Modified binary grey wolf optimizer based on stochastic fractal search for feature selection," *IEEE Access*, vol. 8, pp. 107635–107649, 2020. doi: 10.1109/ACCESS.2020.3001151.

[20] B. Alatas, E. Akin, and A. B. Ozer, "Chaos embedded particle swarm optimization algorithm," *Chaos Solit. Fractals*, vol. 40, no. 4, pp. 1715–1734, 2009. doi: 10.1016/j.chaos.2007.09.063.

[21] H. Haklı and H. Uğuz, "A novel particle swarm optimization algorithm with levy flight," *Appl. Soft Comput.*, vol. 23, pp. 333–345, 2014. doi: 10.1016/j.asoc.2014.06.034.

[22] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary ant lion approaches for feature selection," *Neurocomputing*, vol. 213, pp. 54–65, 2016. doi: 10.1016/j.neucom.2016.03.101.

[23] A. Frank, "UCI machine learning repository," 2010. Accessed: Oct. 01, 2023. [Online]. Available: http://archive.ics.uci.edu/ml

[24] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1774–1785, 2017. doi: 10.1109/TNNLS.2017.2673241.

[25] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, 2017. doi: 10.1016/j.neucom.2017.04.053.

[26] Q. Al-Tashi, S. J. A. Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Binary optimization using hybrid grey wolf optimization for feature selection," *IEEE Access*, vol. 7, pp. 739496–739508, 2019. doi: 10.1109/ACCESS.2019.2906757.

[27] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009. doi: 10.1016/j.ins.2009.03.004.

[28] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang and S. Mirjalili, "Asynchronous accelerating multi-leader salp chains for feature selection," *Appl. Soft Comput.*, vol. 71, pp. 964–979, 2018. doi: 10.1016/j.asoc.2018.07.040.