Neural Computing and Applications (2024) 36:9567–9589
https://doi.org/10.1007/s00521-024-09581-6

ORIGINAL ARTICLE

An improved binary snake optimizer with Gaussian mutation transfer function and hamming distance for feature selection

Xinyu Bao[1] · Hui Kang[1,2] · Hongjuan Li[1]

Received: 12 January 2023 / Accepted: 5 February 2024 / Published online: 7 March 2024
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2024

Abstract
The snake optimizer (SO) is a highly efficient bio-inspired algorithm for solving continuous optimization problems. This algorithm mathematically simulates the unique foraging and mating behaviors observed in snake populations in nature. However, this algorithm cannot be directly applied to solve binary optimization problems, such as feature selection. Feature selection is a significant data preprocessing step in data mining, aimed at reducing data dimensionality, lowering computational costs in terms of time and space, and improving the predictive accuracy of classifiers. To address this limitation, an improved binary version of SO (IBSO) is proposed for feature selection, which incorporates the concept of hamming distance and introduces a novel mutation transfer function (MTF). IBSO extends the application of the conventional SO to binary optimization problems by introducing hamming distance and presents a new binary position update strategy. Furthermore, IBSO utilizes a MTF based on a Gaussian distribution. The MTF not only transforms each dimension of each individual in the population into binary space but also enhances the local random search capability and increases the population diversity of the algorithm. Finally, the experiment is conducted on 27 standard benchmark datasets from UC Irvine Machine Learning Repository and IBSO is compared with several state-of-the-art binary swarm intelligence algorithms to analyze the effectiveness and efficiency of IBSO. The results show that IBSO can obtain the best fitness values with less CPU time. Besides, to evaluate the validity of the proposed MTF, IBSO is compared with other binary versions of SO with different well-known transfer functions.

Keywords Feature selection · Classification · Bio-inspired computing · Snake optimizer

1 Introduction

With the rapid advancement of databases and information technology, our ability to gather data and information from various fields and of different types has significantly improved. The availability of vast amounts of data has greatly facilitated the development and application of artificial intelligence technologies in numerous domains, including bioinformatics [1, 2], earth system science [3], and computer vision [4, 5]. However, many datasets contain hundreds or even thousands of features, which poses significant challenges to methods in machine learning and data mining [6]. Additionally, these datasets often include redundant, irrelevant and noisy features. Training machine learning models using such features not only imposes computational burdens but also adversely impacts the performance of machine learning algorithm [7].

Feature selection is an essential preprocessing step in data mining, aiming to eliminate irrelevant features from a datasets using suitable methods [8]. The primary objective of feature selection is to reduce the number of features in the datasets while improving the performance of machine learning model [9]. Feature selection techniques have the potential to improve the performance of machine learning

✉ Hui Kang
Kanghui@jlu.edu.cn

Xinyu Bao
Baoxy1820@mails.jlu.edu.cn

Hongjuan Li
lihongjuan1205@foxmail.com

1 College of Computer Science and Technology, Jilin University, Changchun 130012, China

2 Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

model while reducing computational complexity, computation time and storage overhead. They effectively address the problem of the "curse of dimensionality" by eliminating irrelevant and redundant features [10].

Feature selection methods can be classified into three major categories: filter methods, wrapper methods and embedded methods [11]. Filter methods in feature selection refer to a category of techniques that assess the relevance of features based on their individual characteristics. These methods use statistical measures, such as correlation or information gain, to rank or score features. Filter methods have the advantage of being computationally efficient since they do not require training a learning algorithm. However, they may overlook interactions or dependencies between features, as they evaluate features independently [12]. Wrapper methods employ a specific learning model to assess feature subsets. They create multiple subsets of features, train a model on each subset, and assess their performances using cross-validation or other evaluation metrics. Wrapper methods can provide more accurate feature selection but are computationally expensive due to the repeated training of models [13]. In addition, feature selection methods include the embedded method, which combines feature selection with model training. In this approach, feature selection is integrated as part of the model training process. This method is powerful and flexible, but it may involve relatively higher computational costs [14].

The feature selection issue can be viewed as an NP-hard global combinatorial optimization problem [15], so that the computing complexities of the exhaustive search approaches are usually unacceptable, especially for the datasets with huge feature space. The heuristic search approaches are more efficient and simpler, thereby widely conducted to tackle feature selection problems [16]. Swarm intelligence optimization algorithms, as an important branch of heuristic search methods, have been widely applied to address feature selection problems. Many researchers employ swarm intelligence optimization algorithms such as dragonfly algorithm (DA) [17], grey wolf optimization (GWO) [18], bat algorithm (BA) [19] and particle swarm optimization (PSO) [20] as the core search strategies in the wrapper-based feature selection approach.

Inspired by the special behaviors of snakes in nature, Hashim and Hussien [21] developed an optimization algorithm called snake optimizer (SO). This algorithm simulates various special behaviors of snakes, including foraging and mating. Through comparative experiments with other state-of-the-art swarm intelligence optimization algorithms, SO has demonstrated its effectiveness in solving continuous optimization problems and several real-world engineering optimization problems. However, according to the no free lunch (NFL) theorems [22], no universally best method can deal with all optimization problems properly. Besides, SO may have certain defects. For example, the exploitation capability of SO relies on the mating mode which may be inadequate for the huge binary search spaces of feature selection issues. Therefore, a reptile search algorithm-snake optimizer (RSA-SO) has been proposed in [23], which introduces a parallel mechanism to avoid to stuck in local optima, to cope with the feature selection problems. However, RSA-SO adopted a parallel mechanism which may cause more computational cost. In addition, the conventional SO cannot be directly used for binary optimization problems like feature selection. Therefore, we are motivated to enhance and modify the SO to make it more suitable for handling feature selection problems by the aforementioned circumstances. The main conclusions and contributions of this paper are listed as follows:

- Formulating a linear weighted sum fitness function to recast the bi-objective optimization problem as a single objective problem.
- Proposing an improved binary snake optimizer (IBSO) as the major search strategy of the wrapper-based approach to tackle the feature selection issues.
- Proposing a binary position update strategy based on hamming distance to extend the use of the SO for feature selection.
- Proposing a novel mutation transfer function based on the Gaussian distribution to enhance the performance of IBSO.
- Using 27 benchmark datasets from the UC Irvine Machine Learning Repository for the experiments and comparing the results obtained by IBSO with 7 state-of-the-art binary swarm intelligence algorithms to evaluate the effectiveness and efficiency of IBSO in dealing with feature selection problems.
- Investigating the effectiveness of the proposed novel Gaussian mutation transfer function for IBSO by comparing with other well-known and widely used transfer functions.

The rest of this paper is organized as follows. Section 2 provides a review of swarm intelligence optimization algorithms and their applications in feature selection. Section 3 reviews the conventional snake optimization algorithm. Section 4 introduces a fitness function based on the linear weighting method and provides detailed explanations of its various components. Section 5 proposes the IBSO. Section 6 presents the experiment results and analysis. Section 7 offers the conclusion of this work and introduces the future work.

2 Related work

In recent years, there has been an increasing attention and research focus on bio-inspired heuristic algorithms due to their remarkable performances in solving complex optimization problems [24]. Swarm intelligence optimization algorithms are one of the significant branches of the bio-inspired heuristic algorithms. These algorithms take inspiration from the collective behavior of biological populations observed in nature [25].

Over the past few decades, there has been a continued emergence and enrichment of swarm intelligence optimization algorithms, which are being applied to diverse optimization problems from various domains. For instance, by observing the special hunting behavior of harris hawks in nature, Heidari et al. [26] proposed a novel swarm intelligence optimization algorithm called harris hawks optimization (HHO) to address a wide range of optimization problems. Moreover, Braik [27] designed a chameleon swarm algorithm (CSA), which mimics the hunting behavior of chameleons. Balani et al. [28] presented a novel swarm intelligence optimization algorithm called golden eagle optimizer (GEO) and the key inspiration for the algorithm is the hunting procedure of golden eagles. Mirjalili et al. [29] proposed a salp swarm algorithm (SSA) and the base inspiration for SSA is the navigation and foraging behavior of salps in oceans. Su et al. [30] presented a novel dove swarm optimization (DSO) which simulates the unique strategies of doves for searching and obtaining food. In addition, Kaur et al. [31] described a sandpiper optimization algorithm (SOA) which imitates the attacking and migration behaviors of the sandpipers in nature. More swarm intelligence-based bio-inspired algorithms can be found in literature [32–35].

Most of the swarm intelligence algorithms are proposed for optimization problems with continuous search space and cannot be directly used for binary optimization problems. Therefore, many researchers adapt these algorithms to binary ones and propose feature selection methods based on the binary variants of these algorithms [36]. Some of the representative previous works are reviewed as follows.

Eluri and Devarakonda [37] described a wrapper-based approach for dimensionality reduction depending on a binary version of golden eagle optimizer. Zhang et al. [38] proposed a binary particle swarm optimization (BPSO) and employed BPSO in a wrapper-based feature selection approach for the spam detection. Rodrigues et al. [39] designed a feature selection method based on the binary cuckoo search (BCS) and utilized BCS for the theft detection in power distribution systems. Emary et al. [40] exploited a novel binary gray wolf optimization (BGWO) for feature selection in high-dimensional datasets. The authors of [41] presented a feature selection approach for biomedical data classification with a binary chimp optimization algorithm. Moreover, Shekhawat et al. [42] proposed a new binary salp swarm algorithm (BSSA) and used BSSA to select the best feature set from large datasets. In addition, Arora and Anand [43] proposed a binary butterfly optimization algorithm for feature selection to enhance the performance of the machine learning classifier.

Recently, the enhanced versions of swarm intelligence algorithms with certain improved factors have increasingly been applied for feature selection. For example, in order to continually improve the quality of the population, Chen et al. [44] suggested a feature selection approach based on particle swarm optimization which used a correlation-guided update strategy and a particle selection strategy. Moreover, Haouassi [45] used a novel binary grasshopper optimization algorithm (NBGOA) for feature selection and introduced some simple operators to NBGOA to extend the use in binary space. An improved binary dragonfly algorithm called IBDA was presented by Li et al. [46]. A crossover operator and an evolutionary population dynamics were employed in IBDA to improve the performance for feature selection. An enhanced binary pigeon-inspired optimization with a novel transfer function was designed by Pan et al. [47] and utilized for feature selection. Houssein et al. [48] presented an enhanced version of sooty tern optimization algorithm, which was named mSTOA, to solve the feature selection problems. A feature selection method based on the improved binary sparrow search algorithm (iBSSA) was also suggested in the reference [49]. In addition, Zhang et al. [50] proposed a novel wrapper-based method, which integrated the brain storm optimization (BSO) with fresh approaches for individual updating and fresh ways for individual clustering, for feature selection.

Many swarm intelligence optimization algorithms have been proposed and applied to feature selection problems. However, the performance of optimization algorithms varies when faced with different optimization problems. There is no single algorithm that can perfectly solve all feature selection problems. Therefore, this motivates us to design a new approach named IBSO to address more feature selection tasks.

3 Snake optimization algorithm

Hashim and Hussien proposed a novel swarm intelligence optimization algorithm called snake optimizer. This algorithm is inspired by certain peculiar behaviors observed in snake populations in nature, including foraging, mating and fight, as well as their connection to environmental temperature.

In nature, when snakes lack food, they employ random movements to search for sustenance. However, when food is abundant, snakes exhibit different behavioral patterns depending on the level of environmental temperature. When the environmental temperature is relatively high, snakes move toward the direction of food. On the other hand, when the temperature is low, snakes enter the mating phase. During this phase, snakes engage in two types of behavior. The first is combat, where snakes fight each other to compete for mates. The second is mating, which occurs between snakes of different genders. After mating, the female snake lays eggs, and if the eggs successfully hatch, new snake individuals are produced.

The aforementioned description provides a brief overview of the unique behaviors exhibited by snakes in their natural habitat. The detailed steps of the SO are as follows.

The first step is initializing the population by the following equation:

$$X_i = X_{\min} + r \times (X_{\max} - X_{\min}) \tag{1}$$

where X_i refers to the position of *ith* individual, X_{max}, X_{min} are the upper and lower bounds defined in the optimization problem and r is a random number in uniform distribution $r \in U(0,1)$. After initialization process of each agent, the snakes are divided into two equal groups: male group and female one.

$$N_m \approx N/2 \tag{2}$$

$$N_f = N - N_m \tag{3}$$

where N is the number of snakes in the population, N_m represents the number of male individuals and N_f is the number of female ones. In each iteration, SO evaluates each group to find the best male ($f_{\text{best},m}$), the best female ($f_{best,f}$) and the global best agent (f_{food}).

As mentioned above, there are two important factors in SO that are the ambient temperature Temp and the quantity of the food (Q), which are mathematically given by:

$$\text{Temp} = \exp\left(\frac{-t}{T}\right) \tag{4}$$

$$Q = c_1 \times \exp\left(\frac{t-T}{T}\right) \tag{5}$$

where t represents the current iteration number, T represents the maximum number of iterations and c_1 is a constant equal to 0.5.

When $Q < \text{Threshold}_{\text{food}}$ ($\text{Threshold}_{\text{food}}$ = 0.25), the snakes move to any random position to search for food and update their position as follows.

$$\begin{aligned} X_i(t+1) = & X_{\text{rand}}(t) \pm c_2 \times A \\ & \times ((X_{\max} - X_{\min}) \times rand + X_{\min}) \end{aligned} \tag{6}$$

where,

$$A = \exp\left(\frac{-f_{\text{rand}}}{f_i}\right) \tag{7}$$

where X_i is the location of the *ith* agent in male or female group, X_{rand} is the location of a random individual from the same gender population, c_2 is a constant equal to 0.05, $rand \in U(0,1), f_{rand,m}$ is the value of the fitness function of the X_{rand} and the f_i is the value of the fitness function of the X_i.

In the exploitation phase, SO has two stages to search for the optimal solution. If $\text{Temp} > \text{Threshold}_{\text{temp}}$ ($\text{Threshold}_{\text{temp}}$ = 0.6), whether male or female, they only move toward the position of food:

$$\begin{aligned} X_i(t+1) = & X_{\text{food}} \pm c_3 \times \text{Temp} \times \text{rand} \\ & \times (X_{\text{food}} - X_i(t)) \end{aligned} \tag{8}$$

where X_i is the location of the male or female agent, X_{food} is the best position achieved so far by the snake swarm and c_3 is a constant equal to 2.

If $\text{Temp} < \text{Threshold}_{\text{temp}}$, there are two behavior modes of the snakes which are the fight mode and the mating mode.

For the fight mode:

$$\begin{aligned} X_i(t+1) = & X_i(t) + c_3 \times F \times rand \\ & \times (Q \times X_{\text{best}} - X_i(t)) \end{aligned} \tag{9}$$

where,

$$F = \exp\left(\frac{-f_{best}}{f_i}\right) \tag{10}$$

where X_i represents the location of the *ith* individual in male or female group and X_{best} is the position of the best individual in another group.

Mating mode:

$$\begin{aligned} X_i(t+1) = & X_i(t) + c_3 \times M \times \text{rand} \\ & \times \left(Q \times X_{i,\text{another}}(t) - X_i(t)\right) \end{aligned} \tag{11}$$

where,

$$M = \exp\left(\frac{-f_{i,another}}{f_i}\right) \tag{12}$$

where X_i is the location of the *ith* male or female and $X_{i,another}$ is the *ith* individual in another group. Specially, if mating occurs, the snake eggs will be laid and if the eggs hatch, the newly born snake will replace the worst one X_{worst} in males and females:

$$X_{\text{worst}} = X_{\min} + \text{rand} \times (X_{\max} - X_{\min}) \tag{13}$$

Through this process, the worst one in the current population is erased and replaced by a newly born individual.

4 Fitness function

The fitness function plays a crucial role in determining the optimization result achieved by the algorithm. Feature selection aims to improve the classification accuracy of the classifier while reducing the number of features in the datasets. Therefore, feature selection can be viewed as a bi-objective optimization problem.

To enable the use of swarm intelligence optimization algorithms for solving this problem, we have employed a linear weighted method to integrate the two objectives into a single fitness function, as shown below:

$$Fitness = \alpha \times (1 - acc) + (1 - \alpha) \times \frac{F_s}{F_{all}} \tag{14}$$

where acc is the prediction accuracy of machine learning classifier so that $(1 - acc)$ represents the error rate. Additionally, F_{all} and F_s can be used to describe the entire number of features and the number of features that were chosen by the feature selection methods. Besides, α is a weight parameter and $\alpha \in [0, 1]$.

Due to the difference in magnitude between the error rate and the ratio of selected features to the total number of features, it is necessary to set an appropriate value for α in order to optimize both objectives simultaneously. Moreover, adjusting the value of α allows for the relative importance of these two objectives to be tuned. Specifically, when α is large, we prioritize achieving higher classification accuracy and can tolerate a larger number of selected features. Conversely, when α is small, we aim to reduce the dimensionality of the datasets as much as possible, even if it means accepting a loss in accuracy.

Indeed, computational cost is a significant concern in wrapper methods. It is generally advisable to avoid using classifiers with high computational overhead in the context of wrapper feature selection. This is because wrapper methods involve performing multiple evaluations of the classifier, each time with a different feature subset. Therefore, classifiers with high computational complexity can significantly slow down the feature selection process and make it impractical for large-scale datasets.

Choosing classifiers with lower computational cost, such as K-nearest neighbors (KNN), can help mitigate this problem. KNN is known for its simplicity and efficiency, as it primarily involves distance calculations between data points. This makes it a suitable choice for wrapper feature selection, where the classifier needs to be evaluated multiple times. It is simple, intuitive, and easy to implement, which makes it widely used in relevant research. By selecting this computationally inexpensive classifier, we can reduce the overall computational burden and facilitate the efficient exploration of the feature space in wrapper methods. In binary classification problems, the accuracy of KNN can be represented as:

$$acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{15}$$

where acc is the prediction accuracy and TP, TN, FP and FN are the rate of true positives, true negatives, false positives and false negatives, respectively. More generally, when dealing with n-class classification tasks using the KNN algorithm, the accuracy can be calculated as follows:

$$acc = \frac{TP_1 + TP_2 + \ldots + TP_n}{N} \tag{16}$$

where TP_1 to TP_n represent the true positives for each class, and N represents the total number of samples.

5 The proposed binary snake optimization algorithm

SO has been verified as an effective optimization algorithm for continuous optimization problems. However, no one method can perfectly solve all optimization tasks. In some cases, traditional SO may exhibit certain limitations.

Many real-world optimization problems often have a discrete search space. For example, feature selection can be considered as a discrete optimization problem due to its binary search space. In continuous optimization problems, the solution is a real number that lies within a continuous domain. However, in the binary optimization problems, the solution is a binary vector in N-dimensional space $X = (x^1, x^2 .. x^N)$ and each dimension takes binary value: $x^j \in \{0, 1\}$. Hence, SO cannot be directly applied to solve binary optimization tasks. Furthermore, since SO was originally designed to tackle continuous optimization problems, certain stages and strategies within SO may not be suitable for handling binary optimization issues.

These conditions motivate us to expand the application of the SO to binary optimization problems like feature selection. To improve SO and make it more useful and efficient for feature selection, we modify some of the phases of the traditional SO and add some new, improved factors. The process of the proposed IBSO for feature selection is presented in Algorithm 1, and the flowchart of the algorithm is offered in Fig. 2.

5.1 Binary encoding for feature selection

In conventional SO, the operators and update strategies are defined in continuous space and the position of the *ith* snake in the group can be expressed as a real number within the upper and lower bounds and thereby cannot be used in binary problems. In the aim of tackling this issue, we present a binary version of SO, where the position of each snake can be expressed as a binary vector $X_i = \left(x_i^1, x_i^2 \ldots x_i^{dim}\right)$ where *dim* denotes the dimension of the binary search space and the total number of features in the datasets. Moreover, for each dimension of this vector, the value can only be taken as a binary value: 0 or 1.

When a dimension is 0, it indicates that the feature corresponding to that dimension is not selected. Conversely, when a dimension is 1, it signifies that the feature corresponding to that dimension is selected. For example, the current position of a snake is shown as [1, 0, 0, 1, 1, 0, 1, 0, 0], it means that the $F1$, $F4$, $F5$ and $F7$ are selected among the original 9 features, as shown in Fig. 1. Therefore, the snake swarm of the IBSO can be expressed as follows:

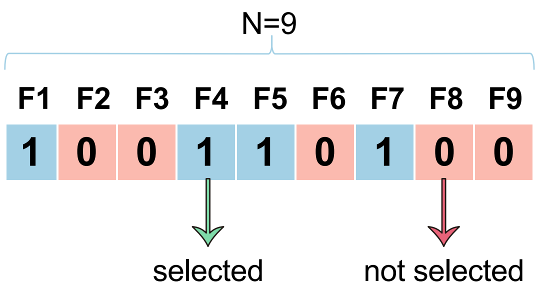

Fig. 1 An example of snake position encoding in feature selection problems where 1 and 0 represent whether the corresponding feature is selected or not, respectively

$$Swarm = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^{dim} \\ x_2^1 & x_2^2 & \cdots & x_2^{dim} \\ \vdots & \vdots & & \vdots \\ x_N^1 & x_N^2 & \cdots & x_N^{dim} \end{bmatrix} \tag{17}$$

```
Require: Number of snakes ($N$), Maximum number of iterations for optimization ($Iteo_{max}$) and
    dimension of datasets ($Dim$).
Ensure: Optimal position of the snake.
 1: Initialize the population and the problem setting
 2: Divide population N to 2 equal groups: male group and female group
 3: $t = 0$
 4: while $t < Iteo_{max}$ do
 5:     Calculate the fitness values of male and female snake groups.
 6:     Update the best male snake, the best female snake and global optimal individual
 7:     Define $Temp$ and food quantity $Q$.
 8:     if $Q < 0.25$ then
 9:         Update each dimension of each snake by using Eq. (19)
10:     else
11:         if $Temp > 0.6$ then
12:             Update each dimension of each snake by using Eq. (24)
13:         else
14:             if $rand > 0.6$ then
15:                 Snakes are in fight mode and update their positions according to Eq. (26)
16:             else
17:                 Snakes are in mating mode and update their positions according to Eq. (27)
18:                 If the eggs are successfully hatched, replace the worst individuals by using Eq. (28)
19:             end if
20:         end if
21:     end if
22:     Convert each dimension of each individual in the snake group to 0 or 1 by using Eq. (29)
23:     t=t+1.
24: end while
```

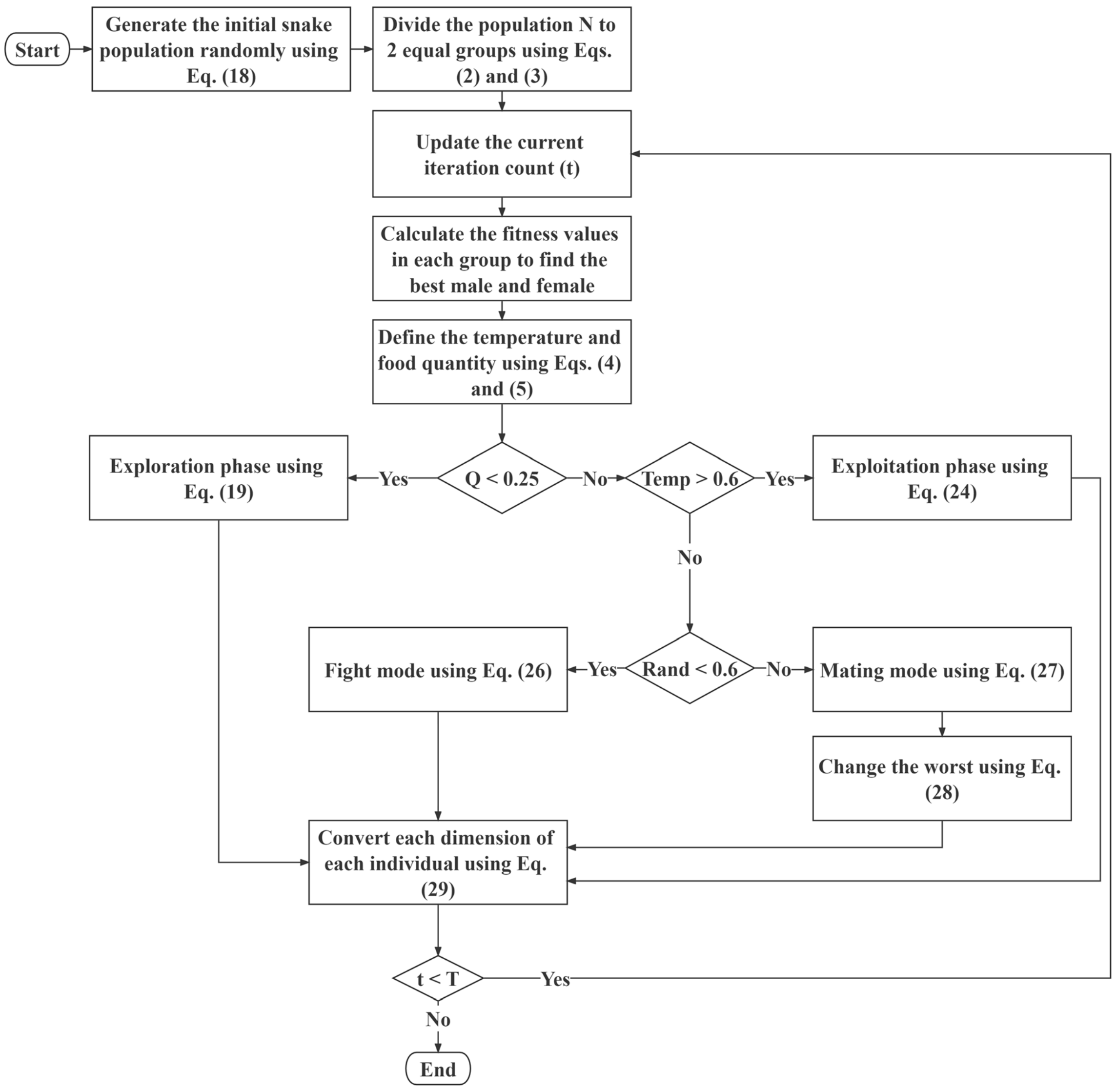
Start
Generate the initial snake population randomly using Eq. (18)
Divide the population N to 2 equal groups using Eqs. (2) and (3)
Update the current iteration count (t)
Calculate the fitness values in each group to find the best male and female
Define the temperature and food quantity using Eqs. (4) and (5)
Exploration phase using Eq. (19)
Yes
Q < 0.25
No
Temp > 0.6
Yes
Exploitation phase using Eq. (24)
No
Fight mode using Eq. (26)
Yes
Rand < 0.6
No
Mating mode using Eq. (27)
Change the worst using Eq. (28)
Convert each dimension of each individual using Eq. (29)
t < T
Yes
No
End

Fig. 2 A detailed flowchart of the IBSO, which gives each process and the formulas used therein

5.2 Initialization of the population

Like the conventional SO, IBSO begins the optimization process by generating a random population. Nevertheless, the initialization of the population in continuous space follows Eq. (1) which is unsuitable for feature selection. Therefore, in this work, we initialize the population in binary space by using the following equation.

$$x_i^j(0) = \begin{cases} 1, & \text{rand} > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where $x_i^j(0)$ is the initial value of the *jth* dimension of the *ith* individual in the snake swarm and $rand \in U(0,1)$. Then, the definition and initialization of N_f, N_m, $Temp$ and Q follow Eqs. (2), (3), (4) and (5).

5.3 Exploration phase in binary space

In the exploration stage of conventional SO, each individual randomly moves closer to or farther away from another random individual within the population. This design significantly enhances the capacity for global exploration of the algorithm.

Considering the feature selection problem, especially when dealing with large-scale and high-dimensional datasets, there exists a tremendous search space. Therefore, it is essential to enhance the global exploration capability of IBSO so that it can find a more promising position during the exploration phase, thereby enabling the discovery of the optimal solution in subsequent exploitation stage. Hence, IBSO extends the position update strategy of the exploration phase in the conventional SO to binary space by

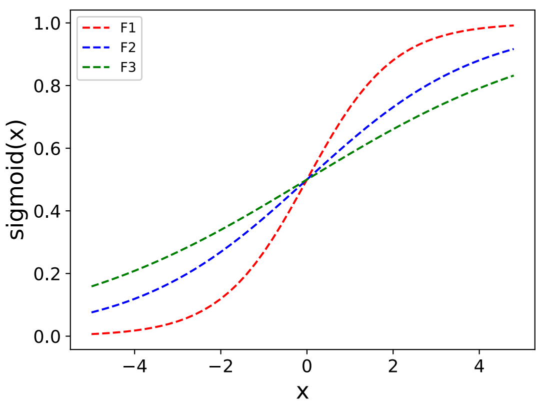

Fig. 3 Three examples of s-shaped transfer functions (γ in $F1$, $F2$ and $F3$ are $1, \frac{1}{2}$ and $\frac{1}{3}$), and by comparing these functions we can see how the value of γ affects the function

calculation of each dimension. For the *ith* agent in the male or female group, it updates each dimension of its position by the following equation.

$$x_i^j(t+1) = x_{rand}^j(t) \pm c_2 \times A \times rand \tag{19}$$

where x_i^j is the *jth* dimension of the *ith* male or female, x_{rand}^j is the value of the *jth* dimension of a random individual in the same group, $rand \in U(0,1)$ and the calculation of A follows Eq. (7). In this phase, we maximize the global exploration capability of IBSO with updating the position of each agent by dimension.

5.4 Exploitation phase with hamming distance

When food is abundant and the environmental temperature is at a high level, snakes tend to focus on searching for food. During this search process, each snake moves toward or keeps away from the food position, which is the best position achieved so far by the snakes. Each snake updates the position by adding or subtracting a value which is related to the distance between the location of the snake and the food location, based on Eq. (8). However, the method for calculating the distance between the snakes and the position of the food is particularly different when updating the positions of the snakes in binary space as opposed to continuous space.

In continuous search space, it can be simply defined as the difference between the two real numbers $\left(X_{food} - X_{i,j}(t)\right)$. In binary solution space, however, each agent is specified as an N-dimensional vector, which seems to be a coordinate point in the N-dimensional space and the search space can be thought of as an N-dimensional hypercube. It is difficult to quantify the distance between each agent and the food position. As a result, we provide a novel position update approach in this work that is based on hamming distance in binary space.

5.4.1 Hamming distance

The hamming distance is a metric used to quantify the dissimilarity between two strings of equal length. Calculating the hamming distance is a straightforward process: comparing the two strings and determining the number of differing characters in corresponding positions. In other words, the hamming distance measures the differences between two strings by comparing their characters at corresponding positions.

For the *ith* and *jth* agents in binary space, their hamming distance $D_h(X_i, X_j)$ can be defined as:

$$D_h(X_i, X_j) = \sum_{k=1}^{dim} d_h\left(x_i^k, x_j^k\right) \tag{20}$$

where d_h can be defined as follow:

$$d_h\left(x_i^k, x_j^k\right) = \begin{cases} 1, & x_i^k \neq x_j^k \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

where *dim* is the dimension of the vector and d_h is the hamming distance between the same dimension of two snakes. The continuous optimization issues allow for the representation of the distances of any two snakes as a real number. However, describing the distance between two positions in high-dimensional space is challenging. To represent the distance between any two positions in binary space, we therefore introduce the hamming distance. In high-dimensional binary space, the distance between two locations can be expressed using a nonnegative integer in this manner.

Our proposed binary version updates the position of each agent by dimension and each dimension $x_i^j \in \{0,1\}$. However, the hamming distance $D_h() \in [0, dim]$, where *dim* is the dimension of the datasets which is far greater than 1. Thus, it is requisite to employ a transfer function to transform D_h from the region [0, *dim*] to a new value which is in the range of [0, 1].

5.4.2 S-shape transfer function

In our study, we introduce the most commonly and widely used sigmoid function to IBSO to transform the hamming distance to a new value in the interval [0, 1], and the detailed description of the sigmoid function is as follows.

$$sigmoid(x) = \frac{1}{e^{-\gamma \times x} + 1} \tag{22}$$

where the γ can adjust the shape of the function as shown in Fig. 3

where γ in $F1$, $F2$ and $F3$ are $1, \frac{1}{2}$ and $\frac{1}{3}$, correspondingly. Hence for the same value of x, if the γ is set to a lower value, the mapping value of x will become lower as well.

5.4.3 Novel position update strategy

This stage, which comes after the global search phase, serves as a transition stage between the global search phase and the local random search phase. To ensure a good convergence rate, we propose a novel position update strategy in this process.

First, we propose a distance transfer function $S()$ based on the sigmoid function as follow:

$$S(x) = \frac{1}{e^{-\frac{1}{Q}\times x} + 1} \tag{23}$$

where Q is the food quantity defined in Eq. (5). This indicates that in the early iterations of the algorithm, the transfer function $S()$ tends to map to a larger value. As the algorithm continues to iterate over time, $S()$ gradually tends to generate a smaller value. By obtaining larger values during the early iteration phase, the algorithm is able to explore the problem space more extensively. Conversely, in the later iterations, the transfer function $S()$ tends to produce smaller values, indicating a more focused local exploitation by the algorithm. This strategy of balancing exploration and exploitation phases enables a smooth transition from the exploration phase to the exploitation phase of the algorithm.

Then, we design the novel position update strategy as follows.

$$x_i^j(t+1) = \begin{cases} x_i^j(t), & d_h(x_{food}^j, x_i^j(t)) = 0 \\ x_{food}^j \pm H, & \text{otherwise} \end{cases} \tag{24}$$

where,

$$H = c_3 \times Temp \times rand \times S(D_h(X_{food}, X_i)) \tag{25}$$

where x_i^j is the jth dimension of the ith agent in the male snakes or female ones, x_{food}^j is the jth dimension of the food location and the $D_h(X_{food}, X_i)$ is the hamming distance between the food position and the ith agent. Specifically, we hold the same dimensions between the agent and the food position. Additionally, because Q influences the value of $S(D_h(X_{food}, X_i))$, the value of H keeps decreasing continuously. The shift from the exploration stage to the local random search stage is facilitated by these factors.

The next part of IBSO is an extension of SO. When the temperature reaches a sufficiently low level, the snake population enters the breeding phase. During this period, snakes exhibit two possible behaviors: fight or mating.

For the fight mode:

$$\begin{aligned} X_i^j(t+1) = &X_i^j(t) + c_3 \times F \times rand \\ &\times \left(Q \times X_{best}^j - X_i^j(t)\right) \end{aligned} \tag{26}$$

where x_i^j is the jth dimension of the ith male or female, $x_{best,f}^j$ is the jth dimension of the best individual in another group and F follows Eq. (10).

Mate mode:

$$\begin{aligned} X_i^j(t+1) = &X_i^j(t) + c_3 \times M \times rand \\ &\times \left(Q \times X_{i,another}^j(t) - X_i^j(t)\right) \end{aligned} \tag{27}$$

where the calculation of M follows Eq. (12)

After the mating behavior, the snakes will lay eggs and if the egg is successfully hatched, select the weakest members in the snake swarm and replace them by using the following equation:

$$x_{\text{worst}}^j = \begin{cases} 1, & \text{rand} > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

After these steps, each dimension of each agent is a continuous value, so we need to transform each dimension of them to binary space appropriately.

5.5 Gaussian mutation transfer function

The mutation is a vital part of the genetic algorithm (GA) which can enhance the local random search capability and increase the population diversity of GA. Many related studies, which were also influenced by GA, use mutation processes to enhance the performance of swarm intelligence-based bio-inspired algorithms [51–55]. Moreover, a fundamental component of the binary bio-inspired algorithm is the transfer function, which can map each dimension of each agent from the continuous space to binary space.

In this study, we propose a mutation transfer function with Gaussian distribution as follows:

$$x_i^j = \begin{cases} 1, & \text{Gauss} < x_i^j \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

where $Gauss$ is a random number generated from a Gaussian distribution with a mean of 0.5 and a standard deviation of 0.25, $Gauss \sim N(0.5, 0.25^2)$, x_i^j is the jth dimension of the ith snake.

In this way, we can convert the value of x_i^j to binary space in each iteration so that making it suitable for the computation of the proposed fitness function and hamming distance. Besides, random numbers are generated from a Gaussian distribution with a mean set to 0.5. In normal circumstances, x_i^j can be mapped from continuous space to binary space based on the value of x_i^j. However, in rare cases, the Gaussian distribution may generate an extremely large or small random number, forcing x_i^j to be mapped to 0 or 1, simulating the process of mutation.

Table 1 Detailed information of benchmark datasets from the UC Irvine Machine Learning Repository for experiments

No.	Datasets	No. of features	No. of instances	No. of classes
D1	Arrhythmia	279	452	13
D2	BreastEW	30	596	2
D3	Clean1	166	476	2
D4	Clean2	166	6598	2
D5	CNAE-9	856	1080	9
D6	Congress	16	435	2
D7	Flags	30	194	7
D8	German	24	1000	2
D9	HeartEW	13	270	2
D10	Hepatitis	19	155	2
D11	Hillvalley	100	606	2
D12	Horse-colic	27	368	2
D13	Image-segmentation	19	2310	7
D14	Ionosphere	34	351	2
D15	Krvskp	36	3196	2
D16	Libras	91	360	15
D17	Parkinsons	23	197	2
D18	Parkinsonspeech	754	756	2
D19	PenglungEW	325	73	7
D20	Semeion	256	1593	10
D21	Sonar	60	208	2
D22	Spect	22	267	2
D23	Spectf	44	267	2
D24	Triazines	60	186	2
D25	Vehicle	18	846	4
D26	Vote	16	300	2
D27	WDBC	31	569	2

Table 2 Comparison of the performance for IBSO with different key parameters values (c_1, c_2 and c_3)

Metrics		(c_1, c_2, c_3)	
	(0.35, 0.04, 1.6)	(0.5, 0.05, 2)	(0.8, 0.07, 2.5)
$avg_{fitness}$	0.2588	**0.2476**	0.2499
$std_{fitness}$	0.0209	**0.0191**	0.0251
$avg_{accuracy}$	0.7419	**0.7521**	0.7496
$std_{accuracy}$	0.0209	**0.0191**	0.0245
$avg_{feature}$	9.10	6.20	**5.47**
$std_{feature}$	4.07	**3.50**	4.46
$avg_{CPUtime}$	8.4	**8.3**	**8.3**

Bold highlights the optimal value within each row (evaluation metric) to underscore the parameter combination yielding the best performance in the considered metric

Through this method, we can increase the population diversity and boost the exploration capability of the algorithm. In addition, compared to other methods, we introduce the mutation process into the algorithm without significant additional computational burden. This is because we combine the mutation process with the transfer function through Gaussian distribution.

5.6 Complexity analysis of IBSO

The most time-consuming step in wrapper feature selection methods based on swarm intelligence algorithms can be the calculation of the fitness function value. The calculation time of this step is significantly larger than that of other steps by several orders of magnitude. So, we ignore other steps in this analysis.

The calculation process of fitness function value has been carried out $Iteo_{max} \times N$ times, where $Iteo_{max}$ is the maximum number of iterations and N is the number of snakes. Therefore, the complexity of IBSO is the same as the conventional SO. However, the proposed strategies may lead to unpredictable additional computing time. Thus, in Sect. 6, this work evaluates the computational complexity of IBSO by the average CPU time.

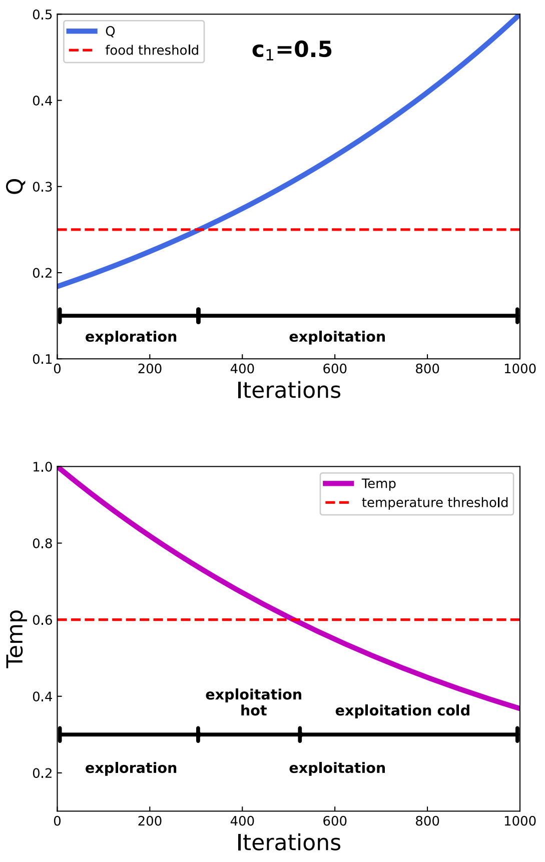

Fig. 4 The variation of the quantity of the food (Q) and the temperature (Temp) parameters of IBSO with iteration and the thresholds of the food (Q) and the temperature (Temp) are marked

6 Experiment and analysis

In this section, we evaluate the performance of IBSO in handling feature selection problems by conducting comparative experiments with other state-of-the-art binary swarm intelligence optimization algorithms.

6.1 Datasets for experiments

27 datasets from the UC Irvine Machine Learning Repository were chosen for this study. Table 1 shows the detail information of them. These datasets are sourced from various domains and have different total numbers of features, ranging from tens to hundreds.

6.2 Parameter tuning

During our research, we recognized that the parameters c_1, c_2, c_3, temperature threshold and food threshold play a crucial role in balancing the exploration and exploitation phases of IBSO. The balance between exploration and exploitation is critical for the effectiveness of swarm intelligence optimization algorithms. Hence, it is essential

Table 3 Key parameter settings for IBSO and the compared algorithms used in the experiment

Algorithm	Parameters
BHHO [26]	$\beta = 1.5,\ q, r1, r2, r3, r4 \in U(0,1)$
	$E_0 \in U(-1,1)$
BSSA [29]	$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2},\ c_2 \in U(0,1),\ c_3 \in U(0,1)$
BGWO [40]	$\alpha \in [2,0]$
BBA [61]	$A = 0.25,\ Q_{max}$=2, Q_{min}=0
BPSO [60]	$c_1 = 2.0,\ c_2 = 2.0$
BCS [39]	$\alpha = 1, \lambda = 1.5, p_a = 0.25$
BGWOPSO [62]	$c_1 = 0.5, c_2 = 0.5, c_3 = 0.5,$
	$w = 0.5 + rand()/2, l \in [1,0]$
IBSO	$c_1 = 0.5,\ c_2 = 0.05,\ c_3 = 2$

to fine-tune these parameters to ensure the optimal performance of IBSO. However, due to the presence of five major parameters, it is not feasible to individually optimize each parameter. Moreover, there exist inherent interdependencies among many parameters. For instance, c_1, temperature threshold and food threshold jointly govern the exploration and exploitation phases of the algorithm.

Taking into consideration the aforementioned circumstances, based on the parameter tuning process outlined in [56] for the snake optimization algorithm, we investigated several key parameters within the algorithm and conducted experiments with several sets of parameters used in [56]. Additionally, according to the NFL theory, it is recommended to adjust the parameters of the algorithm based on different optimization problems to achieve optimal performance.

In this work, we utilized a total of 27 datasets that represent various real-world feature selection problems. However, tuning the parameters on all 27 datasets would require a significant amount of effort and resources. Based on the parameter optimization process outlined in [46], we conducted the optimization of the major parameters using a datasets of intermediate dimensions, which named Flags.

To select the best values of parameters in the snake optimization algorithm, we considered the different combinations of values based on the idea from [21] and [56]. In the interest of fairness, each combination was independently executed 30 times with population size and iteration rounds set at 30 and 100, respectively. Table 2 records the average fitness function values, average number of selected features, average classification accuracy, their respective standard deviations, and average CPU time for different parameters values for IBSO. Apart from having a higher average number of selected features compared to combination (0.8, 0.07, 2.5), combination (0.5, 0.05, 2) demonstrates significant advantages in all other evaluation

Table 4 The average fitness function values and standard deviations obtained from running different algorithms 30 times

	BHHO		BSSA		BGWO		BBA		BPSO		BCS		BGWOPSO		IBSO	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
D1	0.2762	0.0148	0.3026	0.0111	0.3228	**0.0062**	0.3140	0.0066	0.3302	0.0092	0.3377	0.0089	0.3132	0.0076	**0.2201**	0.0224
D2	0.0176	0.0042	0.0166	0.0048	0.0208	0.0030	0.0202	**0.0023**	0.0253	0.0044	0.0261	0.0037	0.0195	0.0031	**0.0129**	0.0039
D3	0.0837	0.0114	0.0800	0.0150	0.1023	0.0070	0.0976	0.0070	0.1079	0.0145	0.1210	**0.0047**	0.0953	0.0121	**0.0446**	0.0147
D4	0.0265	0.0019	0.0253	0.0022	0.0284	**0.0010**	0.0274	0.0011	0.0288	0.0019	0.0317	0.0012	0.0274	0.0014	**0.0202**	0.0020
D5	0.0889	0.0082	0.0922	0.0141	0.0946	**0.0067**	0.1165	0.0075	0.0840	0.0105	0.1010	0.0085	0.1034	0.0084	**0.0700**	0.0096
D6	0.0225	0.0057	0.0187	**0.0043**	0.0195	0.0046	0.0211	0.0051	0.0325	0.0099	0.0361	0.0084	0.0201	0.0049	**0.0183**	0.0050
D7	0.2918	0.0192	0.3187	0.0274	0.3291	**0.0169**	0.3362	0.0171	0.3417	0.0272	0.3603	0.0329	0.3318	0.0228	**0.2492**	0.0306
D8	0.1906	0.0100	0.1821	0.0102	0.1901	**0.0067**	0.1927	0.0082	0.2034	0.0091	0.2134	0.0115	0.1935	0.0080	**0.1785**	0.0078
D9	0.1105	0.0117	0.1179	0.0146	0.1175	0.0135	0.1126	0.0105	0.1356	**0.0009**	0.1337	0.0100	0.1130	0.0116	**0.1100**	0.0158
D10	0.0621	0.0182	0.0611	0.0208	0.0599	**0.0141**	0.0616	0.0154	0.0752	0.0248	0.0913	0.0174	0.0604	0.0179	**0.0522**	0.0205
D11	0.3179	0.0111	0.3157	0.0128	0.3354	0.0077	0.3306	0.0077	0.3340	0.0125	0.3496	0.0079	0.3341	**0.0058**	**0.2886**	0.0094
D12	0.0720	0.0081	0.0797	0.0106	0.0823	0.0067	0.0815	0.0073	0.0922	0.0097	0.1020	0.0103	0.0816	**0.0057**	**0.0634**	0.0081
D13	0.0598	0.0028	0.0579	0.0031	0.0631	0.0025	0.0618	0.0025	0.0702	0.0041	0.0702	0.0034	0.0613	0.0026	**0.0560**	**0.0012**
D14	0.0405	0.0119	0.0634	0.0124	0.0832	0.0071	0.0760	**0.0060**	0.0992	0.0092	0.1015	0.0133	0.0779	0.0076	**0.0309**	0.0084
D15	0.0330	0.0040	0.0267	0.0055	0.0329	0.0032	0.0353	0.0047	0.0362	0.0053	0.0384	0.0050	0.0340	0.0035	**0.0178**	**0.0001**
D16	0.2043	0.0139	0.2074	0.0140	0.2268	**0.0054**	0.2198	0.0074	0.2343	0.0103	0.2384	0.0073	0.2220	0.0074	**0.1743**	0.0156
D17	0.0171	0.0120	0.0169	0.0141	0.0209	0.0115	0.0181	0.0120	0.0326	0.0119	0.0371	0.0103	0.0213	0.0111	**0.0075**	**0.0099**
D18	0.0733	0.0081	0.0785	0.0105	0.0904	0.0053	0.0858	0.0048	0.0824	0.0116	0.1016	0.0050	0.0878	**0.0046**	**0.0437**	0.0089
D19	0.0358	0.0329	0.1208	0.0405	0.1506	0.0260	0.1365	**0.0002**	0.2015	0.0118	0.1938	0.0221	0.1394	0.0117	**0.0072**	0.0198
D20	0.0818	0.0051	0.0828	0.0057	0.0826	0.0035	0.0859	0.0036	0.0753	0.0070	0.0865	**0.0022**	0.0844	0.0038	**0.0729**	0.0056
D21	0.0238	0.0131	0.0198	0.0127	0.0361	0.0107	0.0344	0.0101	0.0510	0.0127	0.0550	0.0113	0.0299	0.0117	**0.0028**	**0.0041**
D22	0.1827	0.0210	0.1653	0.0248	0.1862	0.0134	0.1808	0.0171	0.2133	0.0221	0.2425	0.0183	0.1863	**0.0123**	**0.1483**	0.0231
D23	0.0700	0.0125	0.0585	0.0163	0.0669	**0.0107**	0.0703	0.0112	0.0711	0.0139	0.0875	0.0130	0.0693	0.0119	**0.0416**	0.0168
D24	0.1822	0.0180	0.1806	0.0150	0.1980	0.0142	0.1980	0.0129	0.2324	0.0248	0.2397	0.0194	0.1955	**0.0112**	**0.1430**	0.0183
D25	0.1877	0.0066	0.1827	0.0063	0.1845	**0.0061**	0.1871	0.0068	0.1988	0.0127	0.1989	0.0087	0.1833	0.0064	**0.1781**	0.0062
D26	0.0201	0.0057	0.0197	0.0046	0.0195	0.0046	0.0201	0.0047	0.0317	0.0114	0.0362	0.0112	0.0201	0.0048	**0.0186**	**0.0044**
D27	0.0169	0.0043	0.0164	0.0040	0.0214	**0.0022**	0.0190	0.0035	0.0255	0.0040	0.0261	0.0041	0.0200	0.0030	**0.0119**	0.0033

Bold formatting to denote the optimal values within each row (dataset), drawing attention to the algorithm demonstrating superior performance within the respective datasets

Table 5 The average classification accuracy and standard deviations obtained from running different algorithms 30 times

	BHHO		BSSA		BGWO		BBA		BPSO		BCS		BGWOPSO		IBSO	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
D1	0.7220	0.0148	0.6989	0.0112	0.6795	**0.0064**	0.6875	0.0068	0.6729	0.0093	0.6648	0.0089	0.6886	0.0077	**0.7788**	0.0225
D2	0.9857	0.0042	0.9871	0.0049	0.9836	0.0030	0.9833	**0.0026**	0.9801	0.0045	0.9798	0.0040	0.9845	0.0037	**0.9895**	0.0042
D3	0.9189	0.0113	0.9239	0.0151	0.9025	0.0072	0.9060	0.0072	0.8975	0.0146	0.8849	**0.0047**	0.9088	0.0123	**0.9572**	0.0149
D4	0.9770	0.0021	0.9792	0.0022	0.9773	**0.0008**	0.9773	0.0011	0.9779	0.0018	0.9744	0.0013	0.9775	0.0014	**0.9826**	0.0022
D5	0.9181	0.0081	0.9120	0.0142	0.9108	**0.0069**	0.8873	0.0076	0.9225	0.0106	0.9069	0.0085	0.9014	0.0086	**0.9350**	0.0097
D6	0.9828	0.0057	0.9862	**0.0046**	0.9858	0.0049	0.9839	0.0056	0.9728	0.0101	0.9705	0.0082	0.9851	0.0053	**0.9866**	0.0052
D7	0.7068	**0.0184**	0.6829	0.0277	0.6735	0.0174	0.6650	0.0174	0.6615	0.0276	0.6436	0.0333	0.6701	0.0236	**0.7504**	0.0303
D8	0.8132	0.0104	0.8215	0.0107	0.8197	**0.0070**	0.8108	0.0080	0.8017	0.0092	0.7920	0.0111	0.8103	0.0082	**0.8243**	0.0078
D9	0.8920	0.0118	0.8852	0.0147	0.8858	0.0136	0.8901	0.0106	0.8704	**0.0008**	0.8716	0.0095	0.8901	0.0116	**0.8926**	0.0161
D10	0.9419	0.0194	0.9430	0.0215	0.9452	**0.0148**	0.9430	0.0160	0.9301	0.0251	0.9151	0.0176	0.9441	0.0185	**0.9516**	0.0216
D11	0.6815	0.0112	0.6857	0.0129	0.6669	0.0080	0.6708	0.0077	0.6694	0.0124	0.6543	0.0074	0.6675	**0.0059**	**0.7107**	0.0098
D12	0.9297	0.0081	0.9234	0.0107	0.9221	0.0067	0.9221	0.0076	0.9131	0.0097	0.9041	0.0101	0.9221	**0.0057**	**0.9383**	0.0083
D13	0.9426	0.0029	0.9450	0.0031	0.9405	0.0021	0.9413	0.0023	0.9354	0.0035	0.9356	0.0027	0.9418	0.0024	**0.9470**	**0.0017**
D14	0.9604	0.0121	0.9390	0.0122	0.9205	0.0071	0.9267	**0.0061**	0.9052	0.0094	0.9033	0.0126	0.9252	0.0080	**0.9700**	0.0085
D15	0.9734	0.0035	0.9781	0.0050	0.9732	0.0030	0.9697	0.0049	0.9709	0.0052	0.9697	0.0052	0.9713	0.0039	**0.9845**	**0.0006**
D16	0.7963	0.0136	0.7949	0.0142	0.7764	**0.0055**	0.7824	0.0075	0.7694	0.0105	0.7662	0.0072	0.7805	0.0075	**0.8259**	0.0155
D17	0.9855	0.0127	0.9863	0.0144	0.9829	0.0121	0.9855	0.0127	0.9718	0.0121	0.9684	0.0108	0.9821	0.0118	**0.9949**	**0.0103**
D18	0.9280	0.0078	0.9256	0.0105	0.9146	0.0055	0.9183	0.0049	0.9238	0.0117	0.9046	**0.0047**	0.9166	**0.0047**	**0.9576**	0.0087
D19	0.9644	0.0333	0.8822	0.0410	0.8533	0.0267	0.8667	**0.0000**	0.8022	0.0120	0.8089	0.0227	0.8644	0.0120	**0.9933**	0.0200
D20	0.9242	0.0051	0.9215	0.0058	0.9229	0.0036	0.9184	0.0036	0.9311	0.0070	0.9217	**0.0022**	0.9203	0.0039	**0.9316**	0.0054
D21	0.9794	0.0134	0.9841	0.0128	0.9690	0.0109	0.9698	0.0105	0.9548	0.0128	0.9516	0.0115	0.9746	0.0122	**0.9992**	**0.0043**
D22	0.8189	0.0216	0.8371	0.0251	0.8170	0.0139	0.8214	0.0174	0.7899	0.0222	0.7610	0.0184	0.8164	**0.0128**	**0.8535**	0.0237
D23	0.9340	0.0133	0.9457	0.0165	0.9383	**0.0110**	0.9340	0.0114	0.9346	0.0141	0.9191	0.0131	0.9352	0.0124	**0.9617**	0.0172
D24	0.8180	0.0184	0.8216	0.0150	0.8054	0.0146	0.8045	0.0134	0.7712	0.0249	0.7649	0.0199	0.8072	**0.0115**	**0.8568**	0.0187
D25	0.8158	0.0068	0.8207	0.0063	0.8193	**0.0061**	0.8162	0.0070	0.8061	0.0126	0.8061	0.0085	0.8203	0.0064	**0.8250**	0.0066
D26	0.9851	0.0060	0.9854	0.0051	0.9858	0.0049	0.9851	0.0053	0.9739	0.0115	0.9701	0.0117	0.9851	0.0053	**0.9862**	**0.0046**
D27	0.9860	0.0043	0.9871	0.0044	0.9830	**0.0022**	0.9845	0.0037	0.9798	0.0040	0.9798	0.0046	0.9839	0.0033	**0.9904**	0.0035

Bold formatting to denote the optimal values within each row (dataset), drawing attention to the algorithm demonstrating superior performance within the respective datasets

metrics. Therefore, in our research, we utilized combination (0.5, 0.05, 2) as the value of parameters c_1, c_2 and c_3. Note that the optimal values are bolded to emphasize the parameter combination with the best performance in the considered metric.

We also conducted a more in-depth study on the parameters of the IBSO. For most swarm intelligence optimization algorithms, achieving a good balance between exploration and exploitation is vital for their effectiveness. The IBSO algorithm consists of three distinct phases: the exploration phase when the food quality is below the food threshold, the first exploitation phase when the food quality is above the threshold and the temperature is above the temperature threshold, and the second exploitation phase when the temperature is below the temperature threshold. Therefore, parameter settings should take into account the balance among these three phases as well as the balance between exploration and exploitation.

The food quantity (Q) and temperature (Temp) parameters of IBSO are evaluated during iterations using Eqs. 4 and 5, respectively. The evaluation results are displayed in Fig. 4. From Fig. 4, it is evident that when c_1 is set to 0.5, the temperature threshold of 0.6 and the food threshold of 0.25 effectively balance the exploration and exploitation phases, as well as the two parts within the exploitation phase: exploitation when the temperature is high and exploitation when the temperature is low. Therefore, in the IBSO, we set the food threshold and temperature threshold to 0.25 and 0.6, respectively.

6.3 Experiment setups

We implement the simulation experiment on a PC with an Intel Core i7-10875 H, 32 GB RAM, 2.30 GHz and Python 3.7. Moreover, we utilize the KNN ($k = 5$) as the classifier. Setting k to 5 allows for a good balance between local and

Table 6 The average number of features selected and standard deviations obtained from running different algorithms 30 times

	BHHO		BSSA		BGWO		BBA		BPSO		BCS		BGWOPSO		IBSO	
	N_{avg}	N_{std}	N_{avg}	N_{std}	N_{avg}	N_{std}	N_{avg}	N_{std}	N_{avg}	N_{std}	N_{avg}	N_{std}	N_{avg}	N_{std}	N_{avg}	N_{std}
D1	**26.20**	18.62	119.03	7.88	145.07	**7.76**	131.07	43.13	167.80	7.92	154.50	30.25	129.93	8.10	27.67	8.40
D2	10.27	3.05	11.67	2.17	13.80	1.62	15.80	2.43	16.70	2.13	18.27	3.33	12.37	2.26	**7.40**	**1.25**
D3	57.80	21.00	77.27	6.80	95.27	5.77	84.90	6.36	107.90	**5.37**	118.03	25.82	83.50	7.09	**37.37**	8.26
D4	61.97	17.19	78.23	7.24	97.60	6.83	82.27	6.35	115.40	**5.32**	105.07	20.66	85.87	6.33	**49.03**	8.21
D5	665.93	66.98	434.53	13.72	539.73	16.67	**427.70**	20.38	622.13	**11.84**	759.83	43.73	491.43	27.32	486.57	72.92
D6	8.77	1.69	**8.13**	0.62	8.77	0.76	**8.03**	2.12	8.97	1.68	11.00	1.69	8.53	1.18	**8.03**	**0.55**
D7	**4.33**	4.10	13.40	2.36	16.33	2.61	14.13	2.31	18.57	**1.93**	20.93	2.69	14.43	3.27	5.90	3.64
D8	13.50	2.80	13.00	2.25	15.17	**1.65**	11.87	2.20	16.87	**1.65**	18.07	1.93	13.80	1.97	**11.00**	1.81
D9	**4.63**	1.45	5.53	1.43	5.83	1.29	6.00	2.08	9.43	1.20	8.57	1.89	5.47	1.33	4.80	**0.83**
D10	8.80	2.36	8.93	1.84	10.70	1.64	9.70	2.15	11.40	1.84	13.60	1.93	9.57	**1.54**	**8.23**	1.91
D11	26.03	12.84	45.07	4.89	57.03	5.33	49.10	4.82	67.30	**4.45**	73.47	13.61	49.63	6.00	**22.80**	7.74
D12	6.13	3.36	9.70	2.37	12.90	1.89	11.97	2.26	15.43	1.67	17.57	2.65	11.23	**1.58**	**5.77**	2.58
D13	**5.37**	1.54	6.30	**0.69**	7.70	1.27	8.83	1.97	11.37	1.56	11.70	1.88	6.67	1.60	6.47	1.02
D14	4.60	1.11	10.30	2.00	15.20	2.48	17.33	3.20	18.40	2.29	19.83	5.34	13.17	1.81	**4.13**	**0.72**
D15	24.07	5.35	18.07	3.69	22.80	3.80	18.33	3.25	26.83	**2.38**	30.13	3.51	20.03	3.30	**8.70**	2.38
D16	23.70	9.13	38.87	3.84	48.57	5.11	43.77	5.65	54.47	**3.42**	62.87	13.26	42.87	4.56	**17.47**	5.98
D17	6.03	1.89	7.33	1.68	8.73	1.65	10.97	2.61	10.30	1.88	12.77	2.39	7.80	1.54	**5.33**	**1.01**
D18	156.23	79.06	365.20	**12.89**	442.17	30.04	378.23	17.36	526.63	15.13	540.90	117.77	392.90	23.77	**132.97**	61.72
D19	**18.40**	7.88	135.97	10.07	174.43	16.41	162.63	8.84	186.77	5.46	150.43	17.42	169.97	14.33	18.47	**5.21**
D20	174.30	36.73	**129.60**	**6.73**	160.90	6.86	129.73	10.56	183.33	7.30	231.57	15.84	140.66	10.21	130.80	26.74
D21	20.33	7.55	24.67	2.95	32.80	3.91	30.33	4.01	37.03	4.07	42.23	8.32	28.67	4.19	**12.30**	**2.34**
D22	7.43	2.46	8.83	**1.42**	11.07	1.91	10.90	2.21	11.67	1.66	12.97	2.30	9.87	1.98	**7.07**	1.93
D23	20.33	6.40	20.77	3.63	25.40	3.45	21.93	3.19	27.87	**3.01**	32.77	5.02	22.70	3.31	**16.27**	3.96
D24	12.43	6.71	24.10	3.54	32.37	3.57	30.23	3.75	35.27	2.49	41.30	6.97	27.80	3.13	**7.17**	**1.61**
D25	9.57	1.61	9.43	**1.05**	10.23	1.12	**8.47**	2.36	12.37	1.20	12.60	1.17	9.80	1.28	8.90	1.54
D26	8.50	1.06	8.40	1.05	8.70	0.97	7.93	1.69	9.43	1.76	10.63	2.04	8.43	0.96	**7.87**	**0.34**
D27	8.97	2.85	11.00	2.00	13.73	1.44	15.83	2.50	16.57	2.19	18.40	3.62	12.33	1.56	**7.07**	**1.12**

Bold formatting to denote the optimal values within each row (dataset), drawing attention to the algorithm demonstrating superior performance within the respective datasets

Table 7 Number of remaining features as a percentage of the total number of features in the datasets obtained from running the algorithm 30 times on the datasets (in percentage units)

	BHHO	BSSA	BGWO	BBA	BPSO	BCS	BGWOPSO	IBSO
D1	**9.4**	42.7	52.0	47.0	60.1	55.4	46.6	9.9
D2	34.2	38.9	46.0	52.7	55.7	60.9	41.2	**24.7**
D3	34.8	46.5	57.4	51.1	65.0	71.1	50.3	**22.5**
D4	37.3	47.1	58.8	49.6	69.5	63.3	51.7	**29.5**
D5	77.8	50.8	63.1	**50.0**	72.7	88.8	57.4	56.8
D6	54.8	50.8	54.8	**50.2**	56.1	68.8	53.3	**50.2**
D7	**14.4**	44.7	54.4	47.1	61.9	69.8	48.1	19.7
D8	56.3	54.2	63.2	49.5	70.3	75.3	57.5	**45.8**
D9	**35.6**	42.5	44.8	46.2	72.5	65.9	42.1	**36.9**
D10	46.3	47.0	56.3	51.1	60.0	71.6	50.4	**43.3**
D11	26.0	45.1	57.0	49.1	67.3	73.5	49.6	**22.8**
D12	22.7	35.9	47.8	44.3	57.1	65.1	41.6	**21.4**
D13	**28.3**	33.2	40.5	46.5	59.8	61.6	35.1	34.1
D14	13.5	30.3	44.7	51.0	54.1	58.3	38.7	**12.1**
D15	66.9	50.2	63.3	50.9	74.5	83.7	55.6	**24.2**
D16	26.0	42.7	53.4	48.1	59.9	69.1	47.1	**19.2**
D17	26.2	31.9	38.0	47.7	44.8	55.5	33.9	**23.2**
D18	20.7	48.4	58.6	50.2	69.8	71.7	52.1	**17.6**
D19	**5.7**	41.8	53.7	50.0	57.5	46.3	52.3	**5.7**
D20	68.1	**50.6**	62.9	50.7	71.6	90.5	54.9	51.1
D21	33.9	41.1	54.7	50.6	61.7	70.4	47.8	**20.5**
D22	33.8	40.1	50.3	49.5	53.0	59.0	44.9	**32.1**
D23	46.2	47.2	57.7	49.8	63.3	74.5	51.6	**37.0**
D24	20.7	40.2	54.0	50.4	58.8	68.8	46.3	**12.0**
D25	53.2	52.4	56.8	**47.1**	68.7	70.0	54.4	49.4
D26	53.1	52.5	54.4	49.6	58.9	66.4	52.7	**49.2**
D27	28.9	35.5	44.3	51.1	53.5	59.4	39.8	**22.8**

Bold formatting to denote the optimal values within each row (dataset), drawing attention to the algorithm demonstrating superior performance within the respective datasets

global information, while maintaining manageable computational costs while obtaining reliable results. Moreover, we have set the value of α in the fitness function to 0.99, which is a commonly used value in related studies and can effectively balance the two optimization objectives of feature selection [57–59].

We use binary particle swarm optimization (BPSO) [60], binary grey wolf optimization (BGWO) [40], binary salp swarm algorithm (BSSA) [29], binary bat algorithm (BBA) [61], binary cuckoo search (BCS) [39], binary harris hawk optimization (BHHO) [26] and binary hybrid GWO with PSO (BGWOPSO) [62] as the compared algorithms and the parameters of each are listed in Table 3. To facilitate a meaningful comparison, the parameters listed in the table are all proven in relevant literature to be effective in achieving good performance of the algorithm. In this work, the swarm size and total number of iterations for each method are respectively set to 30 and 100. To offset experimental random bias, we independently execute each method 30 times on each datasets. Additionally, this work uses 80% of the instances for training and the remaining 20% for testing, which is a standard setup in earlier researches.

6.4 Feature selection results and analysis

The findings of the average fitness function values, average number of selected features, average classification accuracy, their respective standard deviations, average CPU time and convergence rate as determined by different algorithms are shown in this subsection. The best values are additionally bolded in Tables 4–8 for emphasis. It's worth noting that the averages and the standard deviations are calculated based on the 30 results obtained from 30 independent runs of each algorithm.

The average fitness function value and standard deviation (*std*) produced by different algorithms are shown in Table 4 as numerical statistics results. Whether the datasets is small, with dozens of features, or large, with hundreds or thousands of features, IBSO performs better than the

Table 8 The average CPU time obtained from running different algorithms 30 times

	BHHO	BSSA	BGWO	BBA	BPSO	BCS	BGWOPSO	IBSO
D1	32.0	36.7	41.0	29.9	**26.9**	29.9	42.0	27.1
D2	23.1	16.9	18.1	18.4	15.9	22.4	21.1	**13.5**
D3	36.9	31.0	32.2	26.6	**23.4**	26.9	34.7	24.1
D4	267.4	182.2	184.0	177.8	170.0	211.2	222.0	**167.8**
D5	102.4	91.1	95.1	68.9	67.5	93.6	111.6	**60.7**
D6	14.8	9.4	9.3	9.3	9.0	13.1	9.8	**8.7**
D7	10.2	9.1	9.3	9.0	7.6	15.7	12.1	**6.6**
D8	42.6	26.6	25.6	24.7	24.6	32.0	28.5	**24.4**
D9	9.6	6.5	6.3	**5.9**	6.1	7.0	6.8	**5.9**
D10	8.6	5.2	5.2	4.9	4.9	6.8	5.9	**4.7**
D11	35.4	28.7	28.9	25.5	25.0	27.4	31.7	**24.9**
D12	15.7	10.5	10.3	10.7	9.1	14.4	13.7	**8.8**
D13	73.5	45.7	47.6	49.2	48.0	66.7	52.1	**40.3**
D14	13.7	14.6	14.3	17.8	11.3	14.4	20.3	**9.2**
D15	172.3	108.8	100.3	114.9	**99.3**	106.7	108.9	105.9
D16	30.9	25.9	25.9	22.9	21.3	29.1	27.2	**21.0**
D17	9.1	6.0	6.0	5.9	5.1	8.0	7.5	**5.0**
D18	63.7	75.5	78.9	55.5	**45.9**	60.9	59.6	47.7
D19	21.3	31.0	32.1	22.0	**17.2**	21.5	27.1	18.5
D20	95.0	69.9	70.4	62.5	**58.5**	70.0	74.0	59.0
D21	19.8	18.0	18.3	16.4	15.6	19.2	19.5	**13.1**
D22	10.7	7.1	7.1	6.9	6.8	8.8	8.4	**6.1**
D23	22.8	17.5	18.2	16.3	15.9	19.7	18.9	**13.2**
D24	17.5	17.9	18.6	16.0	15.0	18.0	19.2	**12.0**
D25	28.2	17.0	18.7	16.3	17.4	18.2	17.2	**16.2**
D26	14.4	9.4	9.3	8.9	8.9	11.6	9.2	**8.3**
D27	23.2	16.7	22.9	18.3	18.7	21.4	20.9	**13.5**

Bold formatting to denote the optimal values within each row (dataset), drawing attention to the algorithm demonstrating superior performance within the respective datasets

compared algorithms, as seen by the best average fitness function values it can obtain across 27 datasets. Moreover, on 6 datasets, IBSO has the best standard deviation of the average fitness function values, and on the remaining datasets, the standard deviation is acceptable. This shows that IBSO has a stable performance when dealing with feature selection problems. Additionally, IBSO outperforms competitors in datasets with a large number of characteristics, demonstrating that it is capable of more comprehensive global exploration. The reasons may be that the proposed mutation transfer function enhances the exploration ability of IBSO.

It should be noted that Sect. 6.5 further evaluates and discusses the efficiency of various transfer functions. Therefore, it can be verified that IBSO is a reliable and effective binary optimization algorithm with good stability for issues involving feature selection.

The average classification accuracy attained by different algorithms are displayed in Table 5. As we can see, out of 27 datasets, IBSO achieves the highest average classification accuracy. Besides, IBSO, BGWO and BGWOPSO demonstrate excellent performance in terms of standard deviation of accuracy. This indicates that IBSO exhibits good stability in terms of accuracy compared to existing algorithms. Hence IBSO performs better on these datasets in terms of classification accuracy. Moreover, Table 6 displays the average quantity of the features that were chosen and the standard deviation produced by different algorithms. In addition, to provide a more visual representation of the dimensionality reduction achieved through feature selection, we recorded in Table 7 the average percentage of selected features obtained from running each algorithm 30 times, relative to the total number of features in the datasets.

Table 6 shows that IBSO chooses the fewest features on average across 19 datasets and the fewest standard deviations on 10 datasets, which means IBSO has better data dimension reduction capability and the stability is also at a better level on most of the datasets. The data presented in Table 7 provide a more visual representation of the

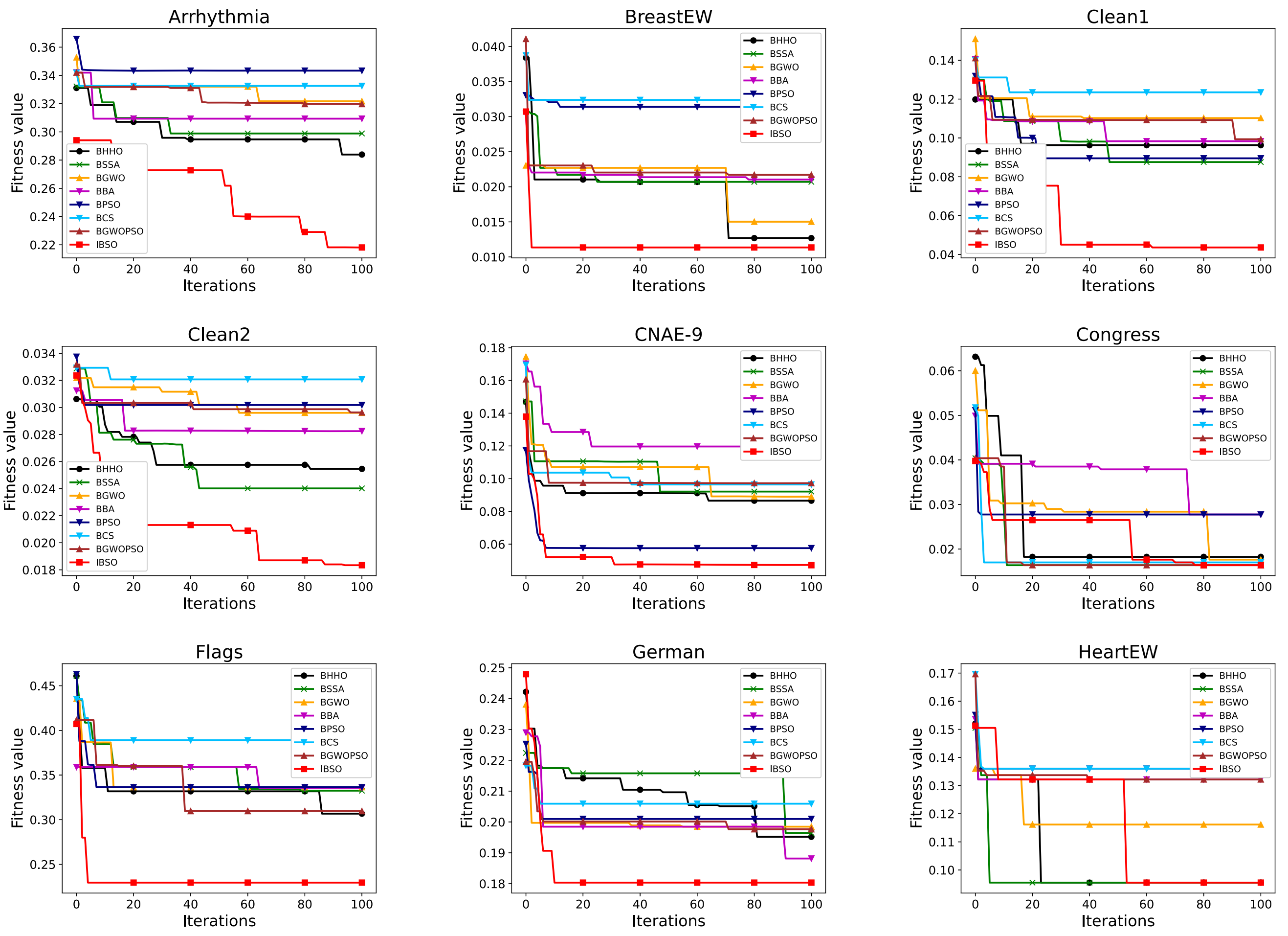

Fig. 5 The convergence curves of IBSO and other algorithms from the 15*th* run on each datasets (Datasets D1 to D9)

dimensionality reduction achieved by different algorithms. Among them, IBSO demonstrates the best dimensionality reduction effect by minimizing the number of features while maintaining accuracy.

It is noteworthy that there are trade-offs between the accuracy of the feature selection and the quantity of features chosen, making it potentially challenging to meet both of these goals for any feature selection approaches. Our experimental results show that IBSO can use fewer features while yet achieving improved feature selection accuracy and stability while taking all the factors mentioned above into account.

The average CPU time for different algorithms is shown in Table 8, where all results are presented in seconds as the unit of measurement. As we can see that BPSO and IBSO achieve the least average CPU time. The calculation of fitness function value has been carried out $Iteo_{max} \times N$ times of IBSO, which is the same as other methods. However, IBSO proposes a mutation transfer function, which combines mutation operation with binary transfer function, so that it can reduce computational complexity on the basis of improving performance compared with other methods. In addition, in the exploitation stage, it does not calculate individuals whose hamming distance is 0 from the food location and the dimensions where the agent and the food location have the same values, which reduces the time cost and keeps its running time at a relatively short level.

Figures 5, 6 and 7 display the convergence rates experienced during the running process of the 15*th* test. Due to space limitations, the figures have been divided into these three parts. The convergence curves of IBSO are all marked in red.

We can observe that the red curves achieve the best convergence performance. It can be confirmed that IBSO achieved the best convergence curves on all 27 datasets used. It is worth noting that many algorithms tend to converge prematurely and get trapped in local optima in the later stages of iteration. Especially for datasets with high dimensions, many algorithms tend to get trapped in

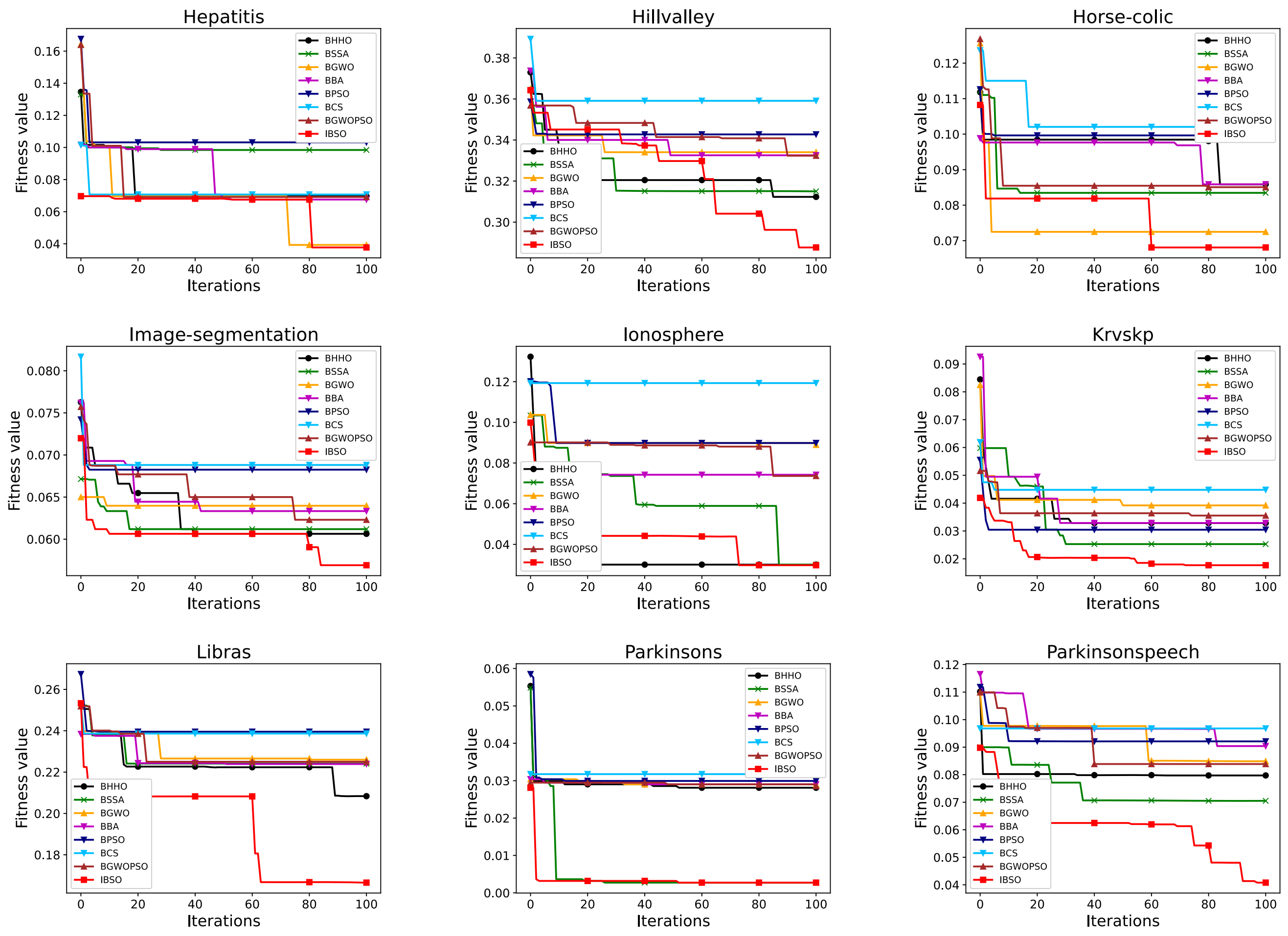

Fig. 6 The convergence curves of IBSO and other algorithms from the 15*th* run on each datasets (Datasets D10 to D18)

local optima and struggle to escape. This issue is largely attributed to the inadequate balance between the exploration and exploitation phases of the algorithms. However, by examining the convergence curves of IBSO on the 27 datasets, it can be observed that IBSO consistently finds better solutions during the later stages of algorithm iteration, specifically in the exploitation phase.

This demonstrates the superiority of the exploitation phase strategy of IBSO, highlighting its powerful exploitation capability and ability to escape local optima. Furthermore, this also demonstrates that the parameters used in IBSO effectively balance the exploration and exploitation phases of the algorithm, enabling it to simultaneously explore the global search space and exploit local areas, leading to the attainment of superior solutions. This is also attributed to the mutation transfer function IBSO employs and we further investigated the effectiveness of the mutation transfer function through experiments in Sect. 6.5.

In order to further analyze the results, it is necessary to perform a statistical significance analysis on the experimental outcomes to demonstrate that the observed differences are not due to randomness. In our study, we conducted a statistical significance analysis on the fitness function values obtained from 30 independent runs of each algorithm on the same datasets.

Considering relevant studies and the specific requirements of our statistical significance analysis task, we chose to employ the Wilcoxon statistical test to examine the obtained average fitness function values at a significance level of 5%. Table 9 displays the p-values obtained from conducting the Wilcoxon test between IBSO and other compared algorithms. We have marked the p-values greater than 0.05 in bold. It can be observed that the majority of p-values are less than 5%, providing strong evidence that IBSO exhibits significant differences compared to other algorithms on most datasets. The superiority of IBSO is attributed to significant performance differences rather than random factors.

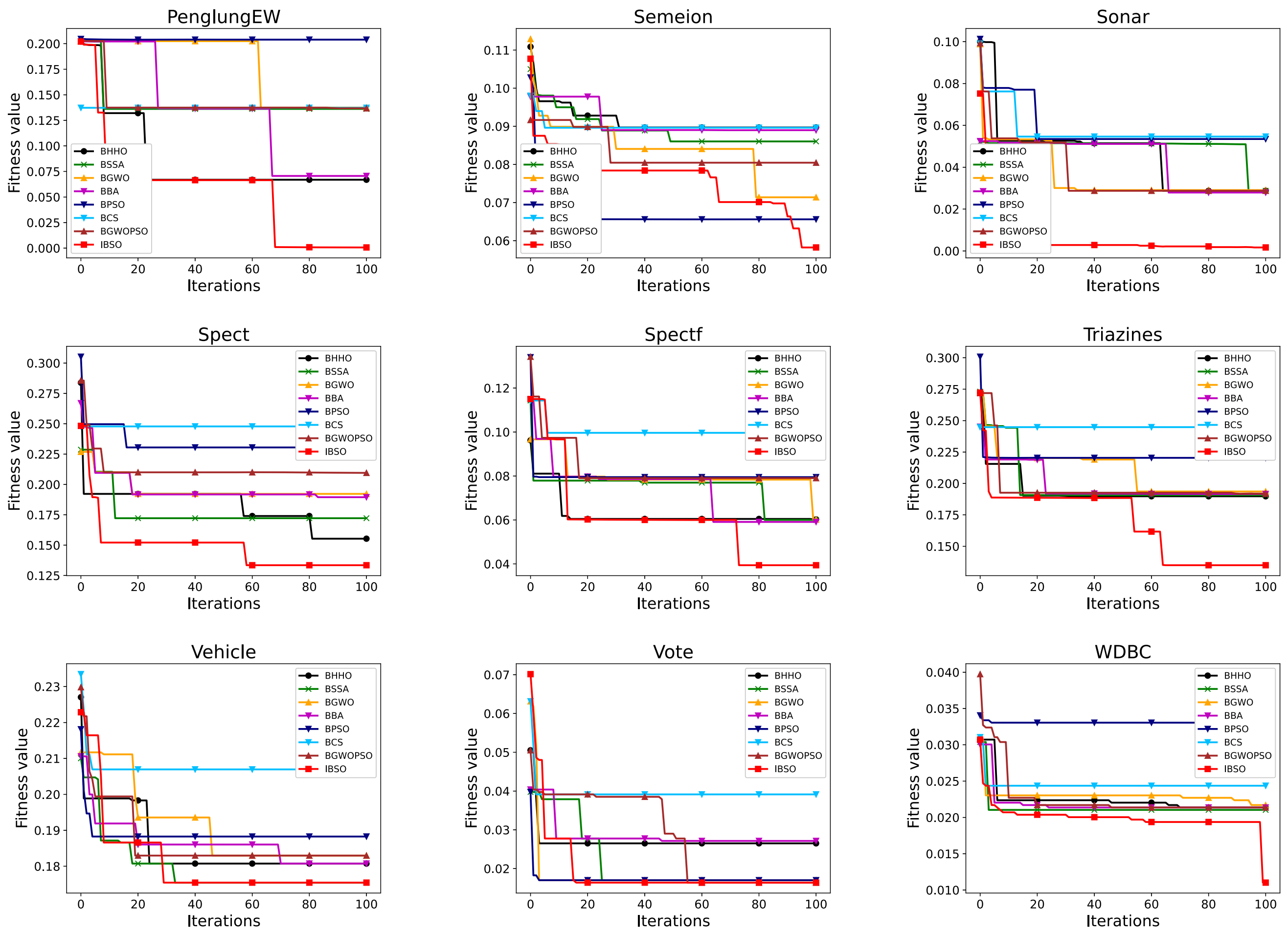

Fig. 7 The convergence curves of IBSO and other algorithms from the 15*th* run on each datasets (Datasets D19 to D27)

6.5 Effectiveness of the proposed mutation transfer function

In this subsection, we compared the performance of four different versions of binary SO: IBSO, BSO-S, BSO-V1 and BSO-V2 to verify the effectiveness of proposed mutation transfer function. IBSO employs the proposed mutation transfer function based on Gaussian distribution and the other three employ the commonly used state-of-the-art transfer functions: S-shape function and V-shape function.

BSO-S adopts a sigmoid transfer function $T_s()$ as in Eq. (30) to transform each dimension of the agents to the interval [0, 1].

$$T_s\left(x_i^j(t)\right) = \frac{1}{1 + e^{-x_i^j(t)}} \tag{30}$$

where $x_i^j(t)$ is the *jth* dimension of the *ith* agents in the *tth* iteration. Then the x_i^j in the next iteration is updated by Eq. (31).

$$x_i^j(t+1) = \begin{cases} 1, & rand < T_s(x_i^j(t)) \\ 0, & otherwise \end{cases} \tag{31}$$

where $rand \in U(0,1)$ and $x_i^j(t+1)$ is the *jth* dimension of the *ith* agents in the $(t+1)th$ iteration. Moreover, Eq. (32) is utilized by BSO-V1 and Eq. (33) is employed by BSO-V2 to transform each dimension of the agents to the interval [0, 1] and updates the x_i^j in the next iteration by Eq. (34).

$$T_{v1}\left(x_i^j(t)\right) = \left| \frac{2}{\pi}\arctan\left(\frac{\pi}{2}\left(x_i^j(t)\right)\right) \right| \tag{32}$$

$$T_{v2}\left(x_i^j(t)\right) = \left| \tanh\left(x_i^j(t)\right) \right| \tag{33}$$

$$x_i^j(t+1) = \begin{cases} \neg x_i^j(t), & rand < T_v(x_i^j(t)) \\ x_i^j(t), & otherwise \end{cases} \tag{34}$$

$T_s()$ is one of the most widely used transfer function for binary swarm intelligence algorithms. Moreover, in particular, there are many different versions of the V-shape

Table 9 The p-value obtained from conducting the Wilcoxon test between IBSO and other control algorithms ($p \geq 0.05$ are bolded)

	BHHO	BSSA	BGWO	BBA	BPSO	BCS	BGWOPSO
D1	1.07E−09	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D2	2.49E−06	2.58E−07	4.40E−10	2.26E−10	5.77E−11	2.87−11	3.06E−09
D3	7.03E−11	1.48E−09	2.87E−11	2.87E−11	3.18E−11	2.87E−11	5.77E−11
D4	5.23E−11	5.84E−10	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D5	8.12E−09	1.31E−07	7.03E−11	2.87E−11	4.27E−06	1.15E−10	5.23E−11
D6	5.43E−05	**2.09E−01**	4.00E−04	3.00E−04	2.60E−08	2.37E−10	5.80E−03
D7	1.00E−04	2.03E−09	6.07E−11	7.39E−11	2.87E−11	3.88E−11	1.86E−10
D8	1.25E−05	7.40E−03	2.78E−07	4.88E−08	1.61E−10	1.04E−10	5.09E−08
D9	**7.84E−01**	2.19E−02	2.11E−02	**3.95E−01**	3.18E−11	3.82E−09	**2.37E−01**
D10	7.00E−04	1.38E−02	7.74E−06	9.79E−05	3.96E−07	1.62E−09	6.00E−04
D11	2.05E−10	8.49E−10	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D12	3.00E−04	7.44E−09	7.39E−11	7.38E−10	3.51E−11	3.18E−11	1.21E−10
D13	1.31E−07	2.60E−03	3.64E−10	2.74E−10	2.87E−11	2.87E−11	2.13E−09
D14	1.00E−04	6.07E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D15	2.87E−11	4.08E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D16	1.01E−08	5.84E−10	2.87E−11	3.18E−11	2.87E−11	2.87E−11	2.87E−11
D17	4.35E−05	3.45E−06	9.67E−09	5.53E−08	9.92E−11	2.87E−11	5.00E−09
D18	8.56E−11	4.73E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D19	4.70E−03	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D20	3.39E−07	1.02E−07	5.71E−09	2.05E−10	**1.39E−01**	3.51E−11	1.02E−09
D21	1.27E−10	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D22	1.87E−07	1.00E−03	6.12E−10	3.65E−08	3.51E−11	2.87E−11	5.32E−10
D23	3.06E−09	1.29E−05	1.94E−09	1.41E−09	8.49E−10	3.51E−11	3.20E−09
D24	1.39E−10	1.04E−10	2.87E−11	2.87E−11	2.87E−11	2.87E−11	2.87E−11
D25	1.19E−06	3.10E−03	3.83E−05	3.70E−06	3.47E−10	1.54E−10	5.00E−04
D26	6.50E−03	6.20E−03	2.60E−03	3.30E−03	2.95E−08	2.74E−10	1.40E−03
D27	7.48E−06	2.19E−08	6.37E−11	5.32E−10	2.87E−11	4.28E−11	2.05E−10

Values exceeding the 0.05 (threshold set in this work) are marked in bold to signal instances where discrepancies in experimental results may be attributable to chance

Table 10 The average fitness function values and average classification accuracy obtained from using binary snake optimization algorithms with different transfer functions

	IBSO		BSO-S		BSO-V1		BSO-V2	
	FIT	ACC	FIT	ACC	FIT	ACC	FIT	ACC
Vote	**0.0168**	**0.9881**	0.0193	0.9858	0.0183	0.9866	0.0203	0.9843
Vehicle	**0.1769**	**0.8264**	0.1839	0.8201	0.1830	0.8201	0.1838	0.8193
Hillvalley	**0.2774**	**0.7231**	0.3350	0.6675	0.2972	0.7017	0.2965	0.7030
PenglungEW	**0.0028**	**0.9978**	0.1440	0.8600	0.0090	0.9911	0.0201	0.9800
Sonar	**0.0020**	**1.0000**	0.0329	0.9722	0.0051	0.9968	0.0058	0.9960
Ionosphere	**0.0290**	**0.9719**	0.0836	0.9200	0.0346	0.9662	0.0328	0.9681
Clean1	**0.0379**	**0.9649**	0.1023	0.9025	0.0668	0.9347	0.0672	0.9347
German	**0.1696**	**0.8340**	0.1889	0.8153	0.1923	0.8107	0.1897	0.8132
WDBC	**0.0108**	**0.9915**	0.0208	0.9833	0.0117	0.9906	0.0147	0.9877
Semeion	**0.0650**	**0.9392**	0.0827	0.9227	0.0836	0.9201	0.0854	0.9185
Spectf	**0.0314**	**0.9722**	0.0679	0.9370	0.0641	0.9383	0.0605	0.9420
CNAE-9	**0.0540**	**0.9502**	0.0890	0.9165	0.1074	0.8961	0.1105	0.8932

Bold formatting to denote the optimal values within each row (dataset), drawing attention to the algorithm demonstrating superior performance within the respective datasets

transfer function. $T_{v2}()$ is utilized in many previous works. In [63], through comparative experiments, the author compared many different versions of the V-shape transfer function, and finally proved that $T_{v1}()$ has merit for use in binary algorithms. Therefore, in this work, we chose these three different transfer functions as comparisons.

We conduct the comparative experiment on 12 datasets of different sizes. Moreover, we compare the average fitness values (FIT) and average accuracy (ACC) obtained by IBSO with those of the other three algorithms in order to confirm its efficacy, and the experimental settings used are the same as those described in Sect. 6.3. The average fitness values attained by SO with various transfer functions are displayed in Table 10. We can observe that IBSO has the best performance on 12 datasets, and the best values are bolded for emphasis. The table shows that the algorithm using our proposed mutation transfer function based on Gaussian distribution is more successful at selecting features.

7 Conclusion

In our research, we address the problem of feature selection by introducing the improved binary snake optimizer (IBSO). We formulate the feature selection problem using a fitness function through the linear weighting method, with the optimization objective set as the value of this fitness function. To extend the application of the conventional SO to binary space, IBSO introduces hamming distance and a mutation transfer function based on a Gaussian distribution. Additionally, this Gaussian-based mutation transfer function enhances population diversity within the algorithm and improves its ability to avoid local optima. To evaluate the effectiveness and efficiency of IBSO, we employ 27 benchmark datasets with varying dimensionalities and compare IBSO with seven well-known binary swarm intelligence algorithms. From our results, it is evident that IBSO outperforms the other algorithms. Besides, we assess the effectiveness of the proposed mutation transfer function. Comparing it with state-of-the-art transfer functions, the results indicate that the proposed mutation transfer function effectively enhances the performance of IBSO.

In the future work, we intend to further improve IBSO to extend the use of the IBSO in dealing with other science and engineering optimization problems. Moreover, we plan to combine the mutation transfer function with other optimization algorithms to enhance their capacity to resolve binary problems.

Author contributions Bao Xinyu contributed to writing—original draft, investigation, formal analysis, and methodology. Kang Hui contributed to conceptualization, methodology, resources, supervision, project administration, and funding acquisition. Li Hongjuan performed writing—review and editing.

Funding This work is supported in part by the National Key Research and Development Program of China (2022YFB4500600), in part by the National Natural Science Foundation of China (62002133, 61872158, 61806083), in part by the Science and Technology Development Plan Project of Jilin Province 20200201166JC, 20220101101JC), in part by Youth Science and Technology Talent Lift Project of Jilin Province (QT202013), and in part by the Excellent Young Talents Program for Department of Science and Technology of Jilin Province (Grant 20190103051JH), and in part by Graduate Innovation Fund of Jilin University (2022028, 2022155).

Data availability All data that support the findings of this study are available by contact with the corresponding author upon reasonable request.

Code availability Not applicable.

Declarations

Conflict of interest The authors have no conflict of interest to declare. The authors have no relevant financial or non-financial conflicts of interest to declare.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Elmarakeby H, Hwang J, Arafeh R, Crowdis J, Gang S, Liu D, AlDubayan SH, Salari K, Kregel S, Richter C, Arnoff TE, Park J, Hahn WC, Allen EMV (2021) Biologically informed deep neural network for prostate cancer discovery. Nature 598(7880):348–352. https://doi.org/10.1038/s41586-021-03922-4
2. Senior AW, Evans R, Jumper J, Kirkpatrick J, Sifre L, Green T, Qin C, Zídek A, Nelson AWR, Bridgland A, Penedones H, Petersen S, Simonyan K, Crossan S, Kohli P, Jones DT, Silver D, Kavukcuoglu K, Hassabis D (2020) Improved protein structure prediction using potentials from deep learning. Nature 577(7792):706–710. https://doi.org/10.1038/s41586-019-1923-7
3. Reichstein M, Camps-Valls G, Stevens B, Jung M, Denzler J, Carvalhais N (2019) Prabhat: deep learning and process understanding for data-driven earth system science. Nature 566(7743):195–204. https://doi.org/10.1038/s41586-019-0912-1
4. He C, Li K, Zhang Y, Xu G, Tang L, Zhang Y, Guo Z, Li X (2023) Weakly-supervised concealed object segmentation with sam-based pseudo labeling and multi-scale feature grouping. CoRR **abs/2305.11003**https://doi.org/10.48550/arXiv.2305.11003
5. He C, Li K, Zhang Y, Tang L, Zhang Y, Guo Z, Li X (2023) Camouflaged object detection with feature decomposition and edge reconstruction. In: IEEE conference on computer vision and pattern recognition (CVPR)
6. Rostami M, Berahmand K, Nasiri E, Forouzandeh S (2021) Review of swarm intelligence-based feature selection methods.

Eng Appl Artif Intell 100:104210. https://doi.org/10.1016/j.engappai.2021.104210
7. Abualigah LM, Diabat A (2022) Chaotic binary group search optimizer for feature selection. Expert Syst Appl 192:116368. https://doi.org/10.1016/j.eswa.2021.116368
8. Chandrashekar G, Sahin F (2014) A survey on feature selection methods. Comput Electr Eng 40(1):16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024
9. Zhang Y, Gong D, Gao X, Tian T, Sun X (2020) Binary differential evolution with self-learning for multi-objective feature selection. Inf Sci 507:67–85. https://doi.org/10.1016/j.ins.2019.08.040
10. Bolón-Canedo V, Alonso-Betanzos A (2019) Ensembles for feature selection: a review and future trends. Inf Fusion 52:1–12. https://doi.org/10.1016/j.inffus.2018.11.008
11. Karlupia N, Abrol P (2023) Wrapper-based optimized feature selection using nature-inspired algorithms. Neural Comput Appl 35(17):12675–12689. https://doi.org/10.1007/s00521-023-08383-6
12. Lee J, Choi IY, Jun C (2021) An efficient multivariate feature ranking method for gene selection in high-dimensional microarray data. Expert Syst Appl 166:113971. https://doi.org/10.1016/j.eswa.2020.113971
13. Zhou P, Li P, Zhao S, Wu X (2021) Feature interaction for streaming feature selection. IEEE Trans Neural Netw Learn Syst 32(10):4691–4702. https://doi.org/10.1109/TNNLS.2020.3025922
14. Remeseiro B, Bolón-Canedo V (2019) A review of feature selection methods in medical applications. Comput Biol Med 112:103375. https://doi.org/10.1016/j.compbiomed.2019.103375
15. Ouadfel S, Elaziz MA (2022) Efficient high-dimension feature selection based on enhanced equilibrium optimizer. Expert Syst Appl 187:115882. https://doi.org/10.1016/j.eswa.2021.115882
16. Chen K, Xue B, Zhang M, Zhou F (2022) Evolutionary multitasking for feature selection in high-dimensional classification via particle swarm optimization. IEEE Trans Evol Comput 26(3):446–460. https://doi.org/10.1109/TEVC.2021.3100056
17. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073. https://doi.org/10.1007/s00521-015-1920-1
18. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
19. Mirjalili S, Mirjalili SM, Yang X (2014) Binary bat algorithm. Neural Comput Appl 25(3–4):663–681. https://doi.org/10.1007/s00521-013-1525-5
20. Ji B, Lu X, Sun G, Zhang W, Li J, Xiao Y (2020) Bio-inspired feature selection: an improved binary particle swarm optimization approach. IEEE Access 8:85989–86002. https://doi.org/10.1109/ACCESS.2020.2992752
21. Hashim FA, Hussien AG (2022) Snake optimizer: a novel meta-heuristic optimization algorithm. Knowl Based Syst 242:108320. https://doi.org/10.1016/j.knosys.2022.108320
22. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893
23. Al-Shourbaji I, Kachare PH, Alshathri S, Duraibi S, Elnaim B, Abd Elaziz M (2022) An efficient parallel reptile search algorithm and snake optimizer approach for feature selection. Mathematics 10(13):1013. https://doi.org/10.3390/math10132351
24. Tian D, Hu J, Sheng Z, Wang Y, Ma J, Wang J (2016) Swarm intelligence algorithm inspired by route choice behavior. J Bionic Eng 13(4):669–678. https://doi.org/10.1016/S1672-6529(16)60338-4
25. Fan X, Sayers W, Zhang S, Han Z, Ren L, Chizari H (2020) Review and classification of bio-inspired algorithms and their applications. J Bionic Eng 17(3):611–631
26. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja MM, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872. https://doi.org/10.1016/j.future.2019.02.028
27. Braik MS (2021) Chameleon swarm algorithm: a bio-inspired optimizer for solving engineering design problems. Expert Syst Appl 174:114685. https://doi.org/10.1016/j.eswa.2021.114685
28. Balani AM, Nayeri MD, Azar A, Yazdi MRT (2021) Golden eagle optimizer: a nature-inspired metaheuristic algorithm. Comput Ind Eng 152:107050. https://doi.org/10.1016/j.cie.2020.107050
29. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
30. Su M, Chen J, Utami AM, Lin S, Wei H (2022) Dove swarm optimization algorithm. Access 10:46690–46696. https://doi.org/10.1109/ACCESS.2022.3170112
31. Kaur A, Jain S, Goel S (2020) Sandpiper optimization algorithm: a novel approach for solving real-life engineering problems. Appl Intell 50(2):582–619. https://doi.org/10.1007/s10489-019-01507-3
32. Tu J, Chen H, Wang M, Gandomi AH (2021) The colony predation algorithm. J Bionic Eng 18(3):674–710
33. Braik M, Hammouri AI, Atwan J, Al-Betar MA, Awadallah MA (2022) White shark optimizer: a novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowl Based Syst 243:108457. https://doi.org/10.1016/j.knosys.2022.108457
34. Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175. https://doi.org/10.1016/j.swevo.2018.02.013
35. Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23(3):715–734. https://doi.org/10.1007/s00500-018-3102-4
36. Macedo M, Siqueira H, Figueiredo EMN, Jro CJS, Lira RC, Gokhale A, Filho CJAB (2021) Overview on binary optimization using swarm-inspired algorithms. IEEE Access 9:149814–149858. https://doi.org/10.1109/ACCESS.2021.3124710
37. Eluri RK, Devarakonda N (2022) Binary golden eagle optimizer with time-varying flight length for feature selection. Knowl Based Syst 247:108771. https://doi.org/10.1016/j.knosys.2022.108771
38. Zhang Y, Wang S, Phillips P, Ji G (2014) Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. Knowl Based Syst 64:22–31. https://doi.org/10.1016/j.knosys.2014.03.015
39. Rodrigues D, Pereira LAM, Almeida TNS, Papa JP, de Souza AN, Ramos CCO, Yang X (2013) BCS: A binary cuckoo search algorithm for feature selection. In: 2013 IEEE international symposium on circuits and systems (ISCAS2013), Beijing, China, May 19-23, pp. 465–468. https://doi.org/10.1109/ISCAS.2013.6571881
40. Emary E, Zawbaa HM, Hassanien AE (2016) Binary grey wolf optimization approaches for feature selection. Neurocomputing 172:371–381. https://doi.org/10.1016/j.neucom.2015.06.083
41. Pashaei E, Pashaei E (2022) An efficient binary chimp optimization algorithm for feature selection in biomedical data classification. Neural Comput Appl 34(8):6427–6451. https://doi.org/10.1007/s00521-021-06775-0
42. Shekhawat SS, Sharma H, Kumar S, Nayyar A, Qureshi B (2021) bssa: binary salp swarm algorithm with hybrid data

transformation for feature selection. IEEE Access 9:14867–14882. https://doi.org/10.1109/ACCESS.2021.3049547
43. Arora S, Anand P (2019) Binary butterfly optimization approaches for feature selection. Expert Syst Appl 116:147–160. https://doi.org/10.1016/j.eswa.2018.08.051
44. Chen K, Xue B, Zhang M, Zhou F (2022) Correlation-guided updating strategy for feature selection in classification with surrogate-assisted particle swarm optimization. IEEE Trans Evol Comput 26(5):1015–1029. https://doi.org/10.1109/TEVC.2021.3134804
45. Haouassi H, Merah E, Rafik M, Messaoud MT, Chouhal O (2022) A new binary grasshopper optimization algorithm for feature selection problem. J King Saud Univ Comput Inf Sci 34(2):316–328. https://doi.org/10.1016/j.jksuci.2019.11.007
46. Li J, Kang H, Sun G, Feng T, Li W, Zhang W, Ji B (2020) IBDA: improved binary dragonfly algorithm with evolutionary population dynamics and adaptive crossover for feature selection. IEEE Access 8:108032–108051. https://doi.org/10.1109/ACCESS.2020.3001204
47. Pan J, Tian A, Chu S, Li J (2021) Improved binary pigeon-inspired optimization and its application for feature selection. Appl Intell 51(12):8661–8679. https://doi.org/10.1007/s10489-021-02302-9
48. Houssein EH, Oliva D, Çelik E, Emam MM, Ghoniem RM (2023) Boosted sooty tern optimization algorithm for global optimization and feature selection. Expert Syst Appl 213(2):119015. https://doi.org/10.1016/j.eswa.2022.119015
49. Gad AG, Sallam KM, Chakrabortty RK, Ryan MJ, Abohany AA (2022) An improved binary sparrow search algorithm for feature selection in data classification. Neural Comput Appl 34(18):15705–15752. https://doi.org/10.1007/s00521-022-07203-7
50. Zhang W, Zhang Y, Peng C (2019) Brain storm optimization for feature selection using new individual clustering and updating mechanism. Appl Intell 49(12):4294–4302. https://doi.org/10.1007/s10489-019-01513-5
51. Rajammal RR, Mirjalili S, Ekambaram G, Palanisamy N (2022) Binary grey wolf optimizer with mutation and adaptive k-nearest neighbour for feature selection in parkinson's disease diagnosis. Knowl Based Syst 246:108701. https://doi.org/10.1016/j.knosys.2022.108701
52. Abdel-Basset M, El-Shahat D, El-Henawy IM, de Albuquerque VHC, Mirjalili S (2020) A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. Expert Syst Appl 139:112824. https://doi.org/10.1016/j.eswa.2019.112824
53. Guo W, Xu P, Dai F, Hou Z (2022) Harris hawks optimization algorithm based on elite fractional mutation for data clustering. Appl Intell 52(10):11407–11433. https://doi.org/10.1007/s10489-021-02985-0
54. Duan X, Zhang X (2022) A hybrid genetic-particle swarm optimizer using precise mutation strategy for computationally expensive problems. Appl Intell 52(8):8510–8533. https://doi.org/10.1007/s10489-021-02828-y
55. Ewees AA, Mostafa RR, Ghoniem RM, Gaheen MA (2022) Improved seagull optimization algorithm using lévy flight and mutation operator for feature selection. Neural Comput Appl 34(10):7437–7472. https://doi.org/10.1007/s00521-021-06751-8
56. Janjanam L, Saha SK, Kar R (2023) Optimal design of hammerstein cubic spline filter for nonlinear system modeling based on snake optimizer. IEEE Trans Ind Electron 70(8):8457–8467. https://doi.org/10.1109/TIE.2022.3213886
57. Sun L, Si S, Zhao J, Xu J, Lin Y, Lv Z (2023) Feature selection using binary monarch butterfly optimization. Appl Intell 53(1):706–727. https://doi.org/10.1007/s10489-022-03554-9
58. Faris H, Mafarja MM, Heidari AA, Aljarah I, Al-Zoubi AM, Mirjalili S, Fujita H (2018) An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowl Based Syst 154:43–67. https://doi.org/10.1016/j.knosys.2018.05.009
59. Emary E, Zawbaa HM, Hassanien AE (2016) Binary ant lion approaches for feature selection. Neurocomputing 213:54–65. https://doi.org/10.1016/j.neucom.2016.03.101
60. Lin S, Ying K, Chen S, Lee Z (2008) Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst Appl 35(4):1817–1824. https://doi.org/10.1016/j.eswa.2007.08.088
61. Liu F, Yan X, Lu Y (2020) Feature selection for image steganalysis using binary bat algorithm. IEEE Access 8:4244–4249. https://doi.org/10.1109/ACCESS.2019.2963084
62. Al-Tashi Q, Abdulkadir SJ, Rais HM, Mirjalili S, Alhussian H (2019) Binary optimization using hybrid grey wolf optimization for feature selection. IEEE Access 7:39496–39508. https://doi.org/10.1109/ACCESS.2019.2906757
63. Mirjalili S, Lewis A (2013) S-shaped versus v-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14. https://doi.org/10.1016/j.swevo.2012.09.002

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.