This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xx.xxxx/ACCESS.xxxx.xxxx

Improved Binary Symbiotic Organism Search Algorithm with Transfer Functions for Feature Selection

ZHI-GANG DU[1], JENG-SHYANG PAN[1, 2],(Senior Member, IEEE),SHU-CHUAN CHU[1, 3],YI-JUI CHIU[4]

[1]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2]College of Information Science and Technology, Dalian Maritime University,116026, China
[3]College of Science and Engineering, Flinders University, 1284 South Road, Clovelly Park SA 5042, Australia
[4]School of Mechanical and Automotive Engineering, Xiamen University of Technology, Xiamen, 361024, Fujian Province, China

Corresponding author: Shu-Chuan Chu (e-mail: scchu0803@gmail.com).

ABSTRACT Symbiotic Organism Search (SOS) algorithm is highly praised by researchers for its excellent convergence performance, global optimization ability and simplicity in solving various continuous practical problems. However, in the real world, there are many binary problems, which can only take values of 0 and 1, that still need to be solved. Since the original SOS algorithm cannot directly solve the binary problem, the original ASOS Binary SOS (BSOS) algorithm has the disadvantage of premature convergence. In order to improve the limitations of the ASBSOS algorithm, we propose an Improved BSOS (IBSOS) algorithm. As we all know, the transfer function is very important in the binarization of continuous optimization algorithms. Therefore, we used 9 transfer functions in the IBSOS algorithm to binarize the continuous SOS algorithm and analyzed the impact of each transfer function on the performance of the BSOS algorithm. Moreover, we use the same three biological symbiosis strategies as the continuous SOS algorithm in our proposed IBSOS algorithm to binarize the SOS algorithm to improve The diversity of the algorithm execution process and the ability to balance algorithm exploration and development. In order to verify the performance of IBSOS using different transfer functions, we use 13 benchmark functions to show the global optimization capability and convergence speed of the BSOS algorithm. Finally, we apply the algorithm to feature selection in the ten data sets of UCI. The experimental results with low classification error and few features further verify the excellent performance of the IBSOS algorithm.

INDEX TERMS Binary Symbiotic Organism Search, transfer function, swarm intelligence, feature selection

I. INTRODUCTION

IN today's huge information systems, thousands of programs are constantly generating a large number of datasets. Application personnel utilize various analysis methods to select valuable data information from the generated datasets, such as neural networks [1], data mining and analysis [2], etc. Many functions applied to pattern recognition and machine learning are redundant and unimportant. However, without prior conditions, it is difficult for us to require the computer to automatically recognize. In this case, it is crucial to select some imperative features in the dataset from the original features to reduce the dimensionality of the data, and the feature selection that requires the selected subset to have higher accuracy [3].

Feature selection algorithms are generally divided into three types: Filter, Wrapper, and Embedded [4]. The Filter method first selects features from the dataset and then trains the model [5]. The feature selection process has nothing to do with subsequent model training [6]. The Wrapper method directly uses the performance of the model to be finally used as the evaluation standard of the feature subset. The purpose of the wrapped feature selection is to select the feature subset, it is most beneficial to the performance of a given model [7]. The embedded method is a combination of the feature selection process and the model training process. Both are completed in the same optimization process, which means the feature selection is automatically performed during the model training process. In general, feature selection can be viewed as a search optimization problem. For a feature set of size n, the search space consists of $2^n - 1$ possible states.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

Davies et al. proved that the search of the smallest feature subset is an NP-hard problem, that is, in addition to an exhaustive search, it cannot guarantee to find the optimal solution. However, in practical application, when the number of features is too large to perform an exhaustive search, people will change the strategy and use a heuristic search algorithm to find the optimal alternative solution within a certain period of time. [8].

In recent decades, as the need to find the optimal alternative solution has continued to increase [9], such as supply chain problem [10–12], Traveling Salesman Problem (TSP) [13, 14], transportation planning problem, etc. Meta-heuristic evolutionary algorithms, also known as Evolutionary Computation (EC) [15], have developed rapidly. When we need the maximum or minimum value of a function, we usually apply the gradient descent method [16]. However, when the data dimension increases, the calculation is extremely time-consuming and even incomplete. Some problems in real life is not differentiable, so the method of gradient descent is not universally applicable [17–19]. Therefore, we need to use EC to obtain the maximum value of these complex functions [20]. EC is divided into three categories: Evolutionary Algorithm (EA) [21], Swarm Intelligence (SI) [22–26], and Memetic Calculation (MC). Where SI is a variety of algorithms generated through the stimulation of various natural phenomena or laws. Particle Swarm Optimization (PSO) [27] is a classic algorithm inspired by birds flying and foraging in a specific space [28–31]. Ant Colony Algorithm (ACO) is a simulation of ant colony foraging behavior widely used in path planning problems [32–34]. The QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm is inspired by the process of quasi-affine transformation in geometry [35–39]. Cat Swarm Optimization (CSO) [40–43] is inspired by the process of cats preying on prey. SI also includes Cuckoo Algorithm (CA) [44, 45], Differential Evolution (DE) [46–49], Artificial Bee Colony (ABC) [50–52] and Binary PSO(BPSO) algorithm [5], etc.

The SOS algorithm is a very promising SI algorithm[53–55]. It has state-of-the-art convergence speed and the ability of global optimization. Because of this, it has been loved by researchers recently. It was first proposed by Cheng and Prayogo in 2014 [56–58]. Many researchers have continued working to improve the performance and application scenarios of the original SOS [59–62]. For example, in 2017, Ezugwu and Adewumi proposed the discrete SOS algorithm (DSOS) for Travelling Salesman Problem (TSP) [13]. In 2020 we proposed the MQSOS algorithm Applied in Wireless Sensor Networks [63]. The original SOS was originally proposed to solve continuous problems in the real world. However, in real life, there are many binary problems that need to be solved. Therefore, we need to binarize the SOS algorithm to make full use of the outstanding performance of the original SOS algorithm, enables the SOS algorithm to solve a wider range of practical problems.

In 2019, Cao Han, Guo Zhou and Yongquan Zhou proposed the BSOS algorithm with AS of transfer method and

FIGURE 1: Mutualism relationship between small tropical fish and coral.

we call this algorithm ASBSOS in this paper, then applied it to feature selection [64]. In order to further improve the performance of the BSOS algorithm, we binarized the original SOS in this paper by using multiple types of transfer functions and observed performance analysis of our proposed IBSOS algorithm in 13 benchmark functions. We verified and analyzed the performance difference of the IBSOS algorithm using nine transfer functions. Finally, we applied the IBSOS algorithm to the practical application of feature selection.

The rest of the paper is organized as follows. Section II briefly introduces the SOS and ASBSOS algorithm. Section III mainly introduces our proposed IBSOS algorithm. Section IV, we compare the difference between the ASBSOS algorithm based on 4 transfer functions and the IBSOS algorithm based on 9 transfer functions through 13 benchmark functions. Section V, the comparison results of the IBSOS and ASBSOS algorithm after being applied to feature selection are presented. Finally, we give conclusions and further research plans in Section VI.

II. RELATED WORKS

A. SOS ALGORITHM

In this section, we introduce the original SOS algorithm, which is inspired by the three coexistence relations of Mutualism, Commensalism, and Parasitism among organisms in the ecosystem. Next, we will introduce the three phases of the SOS algorithm in detail.

1) Mutualism Phase

The mutualism phase describes the mutually beneficial symbiotic relationship between two organisms in the ecosystem. Just like the relationship between the small tropical fish and coral in FIGURE 1. Much small tropical fish use coral branches as their refuges and nests. In exchange, corals have "defence" and "housekeepers" and can get nutrients from fish excreta. We describe this relationship with eq.1 - eq.4.

$$\mathbf{X}_{i\_new} = \mathbf{X}_i^G + r_1 \times (\mathbf{X}_{gbest}^G - \mathbf{X}_{MV}^G \times b_1), \quad (1)$$

$$\mathbf{X}_{j\_new} = \mathbf{X}_j^G + r_2 \times (\mathbf{X}_{gbest}^G - \mathbf{X}_{MV}^G \times b_1), \quad (2)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

FIGURE 2: The turtle's commensalism relationship with algae.

$$\mathbf{X}_{MV}^{G} = \frac{\mathbf{X}_{i}^{G} + \mathbf{X}_{j}^{G}}{2}, \tag{3}$$

$$\begin{cases} b_1 = round(1 + rand(0,1)), \\ b_2 = round(1 + rand(0,1)), \end{cases} \tag{4}$$

Where $\mathbf{X}_i^G$ and $\mathbf{X}_j^G$ respectively represent the position states of the i-th creature and the i-th creature when iterating to the G-th generation. They update their positions to $\mathbf{X}_{i\_new}$ and $\mathbf{X}_{j\_new}$ according to the symbiotic relationship between organisms, and maintain themselves to be optimal through eq.5. r_1 and r_2 are a random variable between 0 and 1. $\mathbf{X}_{MV}^G$ is the central position state (also known as the relationship state) of $\mathbf{X}_i^G$ and $\mathbf{X}_j^G$, and b_1 and b_2 are the profit factors of the two creatures through the Mutualism stage, respectively.

$$\begin{cases} \mathbf{X}_{i}^{G+1} = \mathbf{X}_{i\_new}, & if \quad f(\mathbf{X}_{i}^{G}) > f(\mathbf{X}_{i\_new}), \\ \mathbf{X}_{j}^{G+1} = \mathbf{X}_{j\_new}, & if \quad f(\mathbf{X}_{j}^{G}) > f(\mathbf{X}_{j\_new}), \end{cases} \tag{5}$$

2) Commensalism Phase

Commensalism refers to the interaction between species that has no effect on one party but is beneficial to the other party (if it has no effect on one party and is harmful to the other party, it is called partial symbiosis). Mutual beneficial symbiosis and primitive collaboration can be called "positive interactions". The phenomenon that two kinds of independent living creatures live together in a certain relationship. As shown in FIGURE 2, algae growing on the shell of the turtle benefit from the substrate provided by the daughter turtle. The relationship between the organisms is shown in eq.6.

$$\begin{cases} \mathbf{X}_{i\_new} = \mathbf{X}_{i}^{G} + r_3 \times (\mathbf{X}_{gbest}^{G} - \mathbf{X}_{j}), \\ \mathbf{X}_{i}^{G+1} = \mathbf{X}_{i\_new}, \quad if \quad f(\mathbf{X}_{i}^{G}) > f(\mathbf{X}_{i\_new}), \end{cases} \tag{6}$$

Where r_3 is a random variable between -1 and 1, $f(x)$ is a fitness function. In the Commensalism phase, $\mathbf{X}_i^G$ and $\mathbf{X}_i^G$ interact through eq.6, but only $\mathbf{X}_i^G$ can benefit from this process. When the fitness value of $\mathbf{X}_{i\_new}^G$ is better than the $\mathbf{X}_i^G$, the creature $\mathbf{X}_i^{G+1}$ will update to the state of $\mathbf{X}_{i\_new}$.

3) Parasitic Phase

Parasites are two kinds of organisms living together. One side benefits from the relationship while the other suffers. The latter provides the former with nutrients and a place to live. The relationship between these organisms is called parasitism. Just like the new coronavirus that broke out in 2020, the new coronavirus is transmitted to humans through wild animals (such as bats, civets, and other intermediate hosts), infecting human hosts with diseases. When the human host's own immunity is strong enough, the virus will not be able to affect humans. When the human host's own immunity is weak, humans will die from viral infection. The parasitic relationship between organisms is shown in eq.7.

$$\begin{cases} \mathbf{X}_{parasite} = \mathbf{X}_{i}^{G}, \quad randn > 0.5 \\ \mathbf{X}_{parasite} = rand(0,1) \times \\ \qquad (B_{max} - B_{min}) + B_{min}, \quad else \end{cases}, \tag{7}$$

$$\mathbf{X}_{j}^{G+1} = \mathbf{X}_{parasite}, \quad if \quad f(\mathbf{X}_{parasite}) < f(\mathbf{X}_{j}^{G}), \tag{8}$$

Where X_i^G and X_j^G represent the state of the virus and the human host, $\mathbf{X}_{parasite}$ represents the state of the intermediate host, and B_{max} and B_{min} represent the range of the biological activity interval, respectively.

The creatures update their positions through Mutualism, Commensalism, and Parasitism until the maximum number of iterations is reached. The SOS algorithm pseudo-code is shown in Algorithm 1.

B. THE ASBSOS ALGORITHM

This algorithm is proposed by Cao Han, Guo Zhou and Yongquan Zhou to binarize the continuous SOS algorithm by AS transfer. Only four S-shaped transfer functions are used in the transfer process. We have a description in Section III, corresponding to its S_2, S_3, S_4 and S_5. We take the transfer function S_3 in Table 1 as an example. The transfer method of this AS is shown in the following eqs.9-11 [64].

$$AS(x_i^t) = \frac{1}{1 + e^{\frac{-x_i^t}{A}}} \tag{9}$$

$$A = (1 - \frac{t}{T}) \times T_{max} + \frac{t}{T} \times T_{min} \tag{10}$$

$$x_i^{t+1} = \begin{cases} 0, if \quad rand < AS(x_i^t) \\ 1, else \end{cases} \tag{11}$$

Where t represents the current number of iterations, and T represents the maximum number of iterations we set. In the article [64], T_{max} and T_{min} are control parameters, set to 4 and 0.01 respectively.

Algorithm 1 Shows the Pseudo Code of SOS Algorithm.

Set the size of the biological population to ps, the maximum and minimum values of the boundary to B_{max} and B_{min}, the maximum number of iterations to $iterMax$, and the current iteration $currentIter$ number is 1.
$//$**initialization** Initialize the position matrix $\mathbf{X}$ of the population and calculate the optimal value matrix $\mathbf{F}$ of each creature in the population through the fitness function, and calculate the global optimal position $Gbest$ and the global optimal value of $Gfit$.
$//$**MainLoop**
for $currentIter = 1 : iterMax$ **do**
    for $i = 1 : ps$ **do**
        /* Mutualism*/
        Randomly select a creature $\mathbf{X}_j$ to interact with $\mathbf{X}_i$ for Mutualism stage by eq 1 to 5.
        /* Commensalism*/
        Randomly select a creature $\mathbf{X}_j$ to interact with $\mathbf{X}_i$ for Commensalism stage by eq 6.
        /* Parasitism*/
        Randomly select a creature $\mathbf{X}_j$ to interact with $\mathbf{X}_i$ for Parasitism stage by eq 7 to 8.
    end for
    Calculate the current global optimal creature $Gbest$ and its optimal value $Gfit$.
end for
Ensure: The global optimum $Gbest$, global best fitness value $Gfit$.

III. OUR PROPOSED IBSOS ALGORITHM WITH MULTIPLE TYPES OF TRANSFER FUNCTIONS

In order to distinguish the binary method for the SOS algorithm, we call the original BSOS algorithm the ASBSOS algorithm in this paper. We can clearly see from Section 2.2 that in the process of binarizing the continuous SOS algorithm, the ASBSOS algorithm only relies on the position status of the organism itself for binarization. However, in the idea of the original SOS algorithm, the process of optimization relies on three coexistence relationships between living things. The coexistence of these three creatures is not well reflected in the ASBSOS algorithm. In order to make full use of the optimization ideas in the original SOS, we proposed the Improved Binary Symbiotic Organism Search (IBSOS) algorithm in this paper.

In SOS, because particles can choose any point in the search space to be the current state of the creature, the algorithm can be easily implemented. However, in the binary problem, the value of each dimension of the creature can only be 1 or 0, so the original formula cannot be used to update the position of the creature. Therefore, we will introduce the binarization of the three biological symbiotic stages of SOS.

A. MUTUALISM SYMBIOSIS STRATEGY OF BSOS ALGORITHM

The BSOS algorithm uses the same strategy to find the parameters b_1, b_2 and the intermediate position X_{mv} between the organisms through eq 3 and eq 4. Then, we obtain the value of s_i and s_i by using an S-type function, as shown in eq.12.

$$\begin{cases} s_i^d = \frac{1}{1+e^{-10(C_i^d-0.5)}} \\ s_j^d = \frac{1}{1+e^{-10(C_j^d-0.5)}} \end{cases}, \tag{12}$$

where C_i and C_j are vector of transfer variables, and d represents the value of the d-th dimension,as shown in eq.13.

$$\begin{cases} C_i^d = r_1 \times (\mathbf{X}_{gbest}^{G,d} - \mathbf{X}_{MV}^{G,d} \times b_1) \\ C_j^d = r_2 \times (\mathbf{X}_{gbest}^{G,d} - \mathbf{X}_{MV}^{G,d} \times b_2) \end{cases} \tag{13}$$

The values of $bstep_i$ and $bstep_j$ can be calculated by eqs.14-15. After the above steps, the value we get will not be a continuous value but will be converted into a binary value.

$$bstep_i^d = \begin{cases} 1 & if(s_i^d > randn) \\ 0 & else \end{cases} \tag{14}$$

$$bstep_j^d = \begin{cases} 1 & if(s_j^d > randn) \\ 0 & else \end{cases} \tag{15}$$

Where $randn$ is a random number that obeys a uniform distribution between 0 and 1. $bstep_i^d$ represents the distance that the i-th creature needs to move in the d-th dimension, so the formula for the movement of the creature in the Mutualism phase is shown in eq.16.

$$X_{i\_new}^d = \begin{cases} 1 & X_i^d + bstep_i^d \geq 1 \\ 0 & else \end{cases} \tag{16}$$

X_{i_new} is the position searched by creature X_i, and X_{j_new} can also be calculated by eq.16. We use eq.5 to keep the creature in the optimal position.

B. COMMENSALISM AND PARASITISM STRATEGIES OF BSOS ALGORITHM

In the commensalism phase, since only one creature can benefit from the interaction between the creatures, at this stage, only the creature X_i can randomly select a creature X_j in the population to interact. Then, the creature can continuously search and update to a better position in the search space. We first get the value of s_i through the sigmoid transfer function as shown in eq.17.

$$\begin{cases} s_i^d = \frac{1}{1+e^{-10(C_i^d-0.5)}} \\ C_i^d = r_3 \times (\mathbf{X}_{gbest}^{G,d} - \mathbf{X}_j^d) \end{cases}, \tag{17}$$

Where r_3 has the same meaning as r_3 in eq.6. To find the value of s_i, we can also get the value of $bstep_i$ through eq.14, and finally update the position of the creature X_i through eq.14 and eq.16.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

TABLE 1: The details of five S-type transfer functions.

Name	Transferfunction
S_1	$1/(1+e^{-10(x-0.5)})$
S_2	$1/(1+e^{-2x})$
S_3	$1/(1+e^{-x})$
S_4	$1/(1+e^{-x/2})$
S_5	$1/(1+e^{-x/3})$

In the commensalism phase, because the benefit of one organism is usually based on the harm of another organism, $X_{parasite}$ is calculated as shown in eqs.18-20.

$$\begin{cases} \mathbf{X}^d_{parasite} = \mathbf{X}^{G,d}_i & randn > 0.5 \\ \mathbf{X}^d_{parasite} = bstep^d & else \end{cases}, \tag{18}$$

$$s^d = \frac{1}{1+e^{-10((-1+2\times randn)-0.5)}} \tag{19}$$

$$bstep^d = \begin{cases} 1 & if(s^d > randn) \\ 0 & else \end{cases} \tag{20}$$

We use the three biological symbiosis relationships of the original SOS algorithm to calculate the size of the transfer function value s^d. The larger the value, the higher the probability of obtaining 1 in a certain dimension ($bstep^d$) in the solution space. When the value is smaller, the probability that $bstep^d$ takes 0 is higher. Three s^d calculation methods are used to improve the diversity of the algorithm in the solution space. In this way, compared to the original BSOS algorithm (or ASBSOS algorithm), our proposed algorithm has the advantage of jumping out of the local optimum and avoiding premature convergence of the algorithm.

C. OTHER TRANSFER FUNCTIONS

In the binary SOS algorithm, continuous values are mapped to continuous values between 0 and 1 by a transfer function and then binarized according to probability. This method retains the performance of the original SOS and enables the particle point (biological) to move in the binary space. In Section III-A and Section III-B, we used to binarize the S_1-type curve function. In this section, we will introduce the other four S-type and four V-type transfer function curves. In the original BSOS paper, the author used only the four transfer formulas S2, S3, S4, and S5 for binarization. This time we added five transfer functions, S_1 and $V_1 - V_4$, to compare the performance of BSOS.

Table 1 shows all the S-type transfer functions, and Table 2 shows all the V-type functions. At the same time, we show all the S-shaped and V-shaped function images in FIGURE 3(a) and FIGURE 3(b) respectively.

From FIGURE 3(a), we can see that when x tends to negative infinity, these S-shaped curves tend to be 0. When x tends to positive infinity, the value tends to 1, and the function near x is equal to 0. The trend of image change is rising rapidly. From FIGURE 3(b), we can see that for the V-shaped transfer function curve, when x is equal to 0, the value of the transfer function is 0, and the value of the transfer function tends to 1 when x goes to positive infinity and negative infinity.

TABLE 2: The details of four V-type transfer functions.

Name	Transferfunction
V_1	$\lvert\frac{2}{\pi}\int_{-0}^{(\sqrt{\pi}/2)x} e^{t^2}dt\rvert$
V_2	$\lvert tanh(x)/tanh(4)\rvert$
V_3	$\lvert\sqrt{17}x/(4\sqrt{1+x^2})\rvert$
V_4	$arctan(\frac{2}{\pi})/arctan(2\pi)$

The pseudo-code of the BSOS algorithm is shown in Algorithm 2, and the specific execution flow of the algorithm is shown below.

Step 1: Set the parameters used by the algorithm, such as the population size ps, the maximum number of iterations $maxIter$, and select the binary transfer function used by the algorithm.

Step 2: Initialize the biological population position $\mathbf{X}$, and calculate the optimal value $\mathbf{F}$ of each creature according to the fitness function, and set the current iteration number $Iter$ to 1.

Step 3: Randomly select creatures X_i and X_j, and perform interactions between creatures in the Mutualism stage according to eq.5 and eqs.12-20. The creatures interact in the commensalism phase according to eq.6, eq.14, and eqs.16-17. The parasitic between organisms is realized by eqs.18 - 20. Update the creature's position through the above three stages to complete one iteration.

Step 4: Repeat Step 3 until the maximum number of iterations is met.

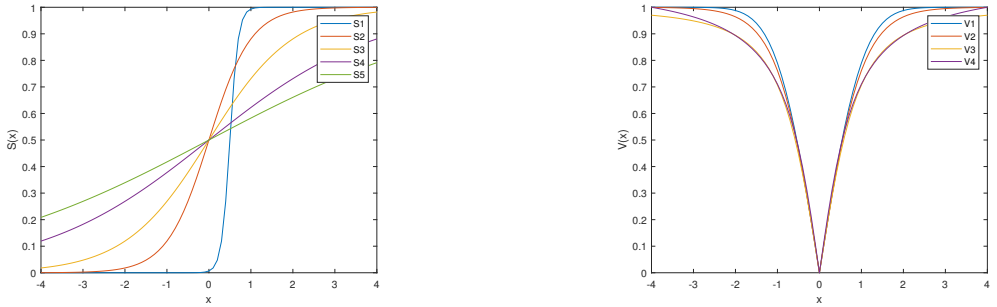
IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we will use 13 benchmark functions to verify the exploration and exploitation capabilities of our proposed BSOS algorithm with multiple types of transfer functions. Where the 13 benchmark functions, there are 7 unimodal functions, 6 multimodal functions. In the unimodal function, there is only one globally optimal solution and there is no local trap (local optimal solution). We can use this function to test the convergence speed of the algorithm. The multimodal function includes not only a globally optimal solution but also one or more local optimal solutions. We can use the multimodal function to test whether the algorithm has a strong ability to break out of local traps and avoid premature convergence. Tables 3 - 4 describe these different types of functions. In the table, "**Name**" represents the name of the function, "**No.**" represents the simplified representation of the function, "**Dim**" represents the dimension of the function, and "**Space**" represents the boundary of the search space for the function, f_{min} represents the optimal value (minimum value) of the function we know.

A. EXPERIMENTAL RESULTS

In order to verify the performance of our proposed BSOS algorithm and the effectiveness of various S-type and V-type transfer functions, we performed simulation experiments on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access



(a) The curves of S-type transfer functions.

(b) The curves of V-type transfer functions.

FIGURE 3: Transfer function curve.

TABLE 3: Unimodal benchmark functions.

No.	Name	Function	Dim	Space	f_{min}
f_1	Sphere	$\sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
f_2	Schwefel's function 2.21	$\sum_{i=1}^{n} \|x_i\| + \prod_{i=1}^{n} \|x_i\|$	30	[-10,10]	0
f_3	Schwefel's function 1.2	$\sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	[-100,100]	0
f_4	Schwefel's function 2.22	$max_i\{\|x_i\|, 1 \leqslant i \leqslant n\}$	30	[-100,100]	0
f_5	Rosenbrock	$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 - (x_i - 1)^2]$	30	[-30,30]	0
f_6	Step	$\sum_{i=1}^{n} (x_i + 0.5)^2$	30	[-100,100]	0
f_7	Dejong's noisy	$\sum_{i=1}^{n} ix_i^4 + random[0,1)$	30	[-100,100]	0

TABLE 4: Multimodal benchmark functions.

No.	Name	Function	Dim	Space	f_{min}
f_8	Schwefel	$\sum_{i=1}^{n} -x_i sin(\sqrt{\|x_i\|})$	30	[-500,500]	-125969
f_9	Rastringin	$\sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
f_{10}	Ackley	$-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}}$ $-e^{\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)} + 20 + e$	30	[-32,32]	0
f_{11}	Griewank	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x[i]}{\sqrt{i}}) + 1$	30	[-600,600]	0
f_{12}	Generalized penalized 1	$\frac{\pi}{n}\{10sin(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2$ $\times[1 + 10sin^2(\pi y_{i+1})]+$ $(y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) =$ $\begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[-50,50]	0
f_{13}	Generalized penalized 2	$0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2$ $\times[1 + sin^2(3\pi x_i + 1)]$ $+(x_n - 1)^2[1 + sin^2(2\pi x_n)]\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	30	[-50,50]	0

the algorithm, and the results are shown in Tables 6-7. Where "**Aver.**" and "**Std.**" is the average value and variance obtained by repeating the algorithm 30 times, respectively. The parameters set in the BSOS algorithm are shown in Table 5.

In the optimization process, the calculation of the optimal value is very time-consuming, and the time consumed by other calculations does not have a decisive impact on the overall performance of the algorithm. In the comparison process, we use the same number of iterations to make each algorithm have the same number of calls to the optimal value function, thereby ensuring the fairness of each algorithm in the comparison process.

B. EXPERIMENTAL ANALYSIS

In order to more intuitively reflect the performance of BSOS based on various transfer functions, we summarize Table 9 through Tables 6-7, where $'Winner'$ represents the best transfer function based on a certain test function. We can clearly see that among the S-type transfer functions, the relative performance of the S_1 transfer function is better than other S-type functions, and the V3 transfer function performs better than other V-type functions.

In order to more clearly see the convergence process of the ASBSOS and IBSOS algorithms under different transfer functions, we have given Figures 4-5. We can clearly see that the IBSOS algorithm has a stronger convergence

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

TABLE 5: The parameter values of BSOS.

parameter	value	Meaning
$iterMax$	500	Population evolution iterations
ps	50	Population size
$Renumber$	30	Repeat the number of experiments

TABLE 6: The mean and variance of the ASBSOS algorithm and IBSOS algorithm run 30 times under the f_1-f_7 benchmark function.

Function		f1	f2	f3	f4	f5	f6	f7
$ASBSOS\_S_2$	**Aver.**	6.033333333	6.266666667	332.9666667	1	0	19.9	76.56347656
	Std.	0.718395402	0.691491807	82.67947389	0	0	1.693802089	11.81378699
$ASBSOS\_S_3$	**Aver.**	5.233333333	5.466666667	232.2666667	1	0	18.03333333	65.76540921
	Std.	0.858359837	0.730296743	80.36165381	0	0	1.655363974	8.94E+00
$ASBSOS\_S_4$	**Aver.**	4.933333333	4.9	182.0666667	1	0	17.3	52.74106224
	Std.	0.583292281	0.844862772	64.83662935	0	0	1.214850636	11.26997601
$ASBSOS\_S_5$	**Aver.**	4.833333333	4.866666667	194.2333333	1	0	16.76666667	54.25444364
	Std.	0.647719252	0.628810225	60.77526909	0	0	1.337350271	9.469125989
$IBSOS\_S_1$	**Aver.**	0	0	0	0.182574186	0	0	0.014764461
	Std.	0	0	0	0.966666667	0	7.5	0.014451011
$IBSOS\_S_2$	**Aver.**	0.413840993	0.449776445	6.173357662	0	0	0.980265036	3.689960218
	Std.	0.966666667	1.066666667	5.6	1	0	9.433333333	6.004836588
$IBSOS\_S_3$	**Aver.**	0.371390676	0.571346464	4.250219737	0	0	0.8051558	4.028806751
	Std.	1	1.133333333	6.933333333	1	0	9.3	7.600074204
$IBSOS\_S_4$	**Aver.**	0.490132518	0.52083046	6.905270351	0	0	0.909717652	3.21438807
	Std.	1.033333333	1.066666667	7.8	1	0	9.5	6.017740252
$IBSOS\_S_5$	**Aver.**	0.490132518	0.413840993	6.504110106	0	0	1.04166092	3.851690809
	Std.	0.966666667	1.033333333	7.2	1	0	9.633333333	7.529215078
$IBSOS\_V_1$	**Aver.**	0.4660916	0.449776445	1.932480985	0	0	0.932183199	1.382312186
	Std.	0.3	0.266666667	1.3	1	0	8.1	1.166820522
$IBSOS\_V_2$	**Aver.**	0.406838102	0.379049022	1.07425462	0	0	0.691491807	0.728216062
	Std.	0.2	0.166666667	0.533333333	1	0	7.766666667	0.934840475
$IBSOS\_V_3$	**Aver.**	0	0	0	0.253708132	0	0	0.016430568
	Std.	0	0	0	0.066666667	29	7.5	0.015662488
$IBSOS\_V_4$	**Aver.**	0.449776445	0.305128577	0.379049022	0	0	0.813676204	1.400961368
	Std.	0.266666667	0.1	0.166666667	1	0	7.9	1.010274961

TABLE 7: The mean and variance of the ASBSOS algorithm and IBSOS algorithm run 30 times under the f_8-f_{13} benchmark function.

Function		f8	f9	f10	f11	f12	f13
$ASBSOS\_S_2$	**Aver.**	-25.24412954	6.3	1.739579026	0.201875788	2.645919146	1.35E-32
	Std.	1.08E-14	0.749712589	0.098759326	0.019329458	1.47E-01	5.57E-48
$ASBSOS\_S_3$	**Aver.**	-25.24412954	5.2	1.614150028	0.173936402	2.52025544	1.35E-32
	Std.	1.08E-14	0.71438423	0.098072064	0.02190067	1.18E-01	5.57E-48
$ASBSOS\_S_4$	**Aver.**	-25.24412954	4.833333333	1.51938329	0.145157309	2.465932067	1.35E-32
	Std.	1.08E-14	0.791477594	0.150045427	0.024205425	1.03E-01	5.57E-48
$ASBSOS\_S_5$	**Aver.**	-25.24412954	4.8	1.539086766	0.147693628	2.390446577	1.35E-32
	Std.	1.08E-14	0.71438423	0.102745763	0.02832289	0.118356171	5.57E-48
$IBSOS\_S_1$	**Aver.**	1.08E-14	0	0	0	1.13E-15	5.57E-48
	Std.	-25.24412954	0	8.88E-16	0	1.668971097	1.35E-32
$IBSOS\_S_2$	**Aver.**	1.08E-14	0.4025779	0.236930983	0.010319943	0.047113022	5.57E-48
	Std.	-25.24412954	0.9	0.664708993	0.02521479	1.821687407	1.35E-32
$IBSOS\_S_3$	**Aver.**	1.08E-14	0.449776445	0.252234199	0.013106762	0.06592209	5.57E-48
	Std.	-25.24412954	0.933333333	0.684006181	0.029291927	1.806415776	1.35E-32
$IBSOS\_S_4$	**Aver.**	1.08E-14	0.480660465	0.288197134	0.012227909	0.07102779	5.57E-48
	Std.	-25.24412954	1.1	0.669750634	0.029660133	1.783071997	1.35E-32
$IBSOS\_S_5$	**Aver.**	1.08E-14	0.484234198	0.265868834	0.014550755	0.084164042	5.57E-48
	Std.	-25.24412954	1.2	0.640804852	0.029531511	1.796598299	1.35E-32
$IBSOS\_V_1$	**Aver.**	1.08E-14	0.490132518	0.247942764	0.009621601	0.032537062	5.57E-48
	Std.	-25.24412954	0.366666667	0.095616564	0.008768877	1.682715565	1.35E-32
$IBSOS\_V_2$	**Aver.**	1.08E-14	0.449776445	0.130928372	0.007979154	0.047117725	5.57E-48
	Std.	-25.24412954	0.266666667	0.023904141	0.004722658	1.691005879	1.35E-32
$IBSOS\_V_3$	**Aver.**	1.08E-14	0	0	0	1.13E-15	5.57E-48
	Std.	-25.24412954	0	8.88E-16	0	1.668971097	1.35E-32
$IBSOS\_V_4$	**Aver.**	1.08E-14	0.379049022	0.271825237	0.007130108	0.029926327	5.57E-48
	Std.	-25.24412954	0.166666667	0.119520705	0.003498499	1.677479577	1.35E-32

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

Algorithm 2 Shows the Pseudo Code of BSOS Algorithm.

Set the size of the biological population to ps, the maximum number of iterations to $iterMax$, and the current iteration $Iter$ number is 1, the fitness function is f (x), and the transfer function is g (x).
$//\mathbf{initialization}$
for i=1:ps **do**
    /*D is the dimension of the space in which the living beings are located.*/
    for i=1:D **do**
        if rand>1 **then**
            $X_{i,j}$=1;
        else
            $X_{i,j}$=0;
        end if
    end for
end for
$\mathbf{F} = f(\mathbf{X})$;
$[Gbest, Gfit] = min(\mathbf{F})$;
$//\mathbf{MainLoop}$
for $currentIter = 1 : iterMax$ **do**
    for $i = 1 : ps$ **do**
        /* Mutualism*/
        Randomly select a creature $\mathbf{X}_j$ to interact with $\mathbf{X}_i$ for Mutualism stage by 5 and eqs.12-20.
        /* Commensalism*/
        Randomly select a creature $\mathbf{X}_j$ to interact with $\mathbf{X}_i$ for Commensalism stage by eq 6, 14, 16 and 17.
        /* Parasitism*/
        Randomly select a creature $\mathbf{X}_j$ to interact with $\mathbf{X}_i$ for Parasitism stage by eqs.18 - 20.
    end for
    Calculate the current global optimal creature $Gbest$ and its optimal value $Gfit$.
end for
Ensure: The global optimum $Gbest$, global best fitness value $Gfit$.

performance than the ASBSOS algorithm, and the ASBSOS algorithm has the shortcomings of being extremely premature and easily falling into the local optimum. Compared with the two algorithms as a whole, the IBSOS algorithm makes better use of the continuous SOS algorithm with good performance. In the nine transfer functions used in the IBSOS algorithm in this paper, we can draw the following conclusions through the experimental results and the convergence process of the algorithm in the global optimization: the S_1 and V_3 transfer functions are compared to other transfer functions In terms of achieving good global optimization and convergence performance.

From the above table, it is shown that the BSOS algorithm based on S_1, V_3 and V_4 transfer functions obtains better convergence and optimization capabilities.

V. APPLICATION FOR FEATURE SELECTION

The reason for our feature selection is to deal with the dimensional disaster. The so-called dimensional disaster is that when the feature dimension exceeds a certain limit, the performance of the classifier decreases with the increase of the feature dimension (and the higher the dimension, the greater the cost of training the model). The reason for the decline of the classifier is usually because these high-latitude features contain irrelevant and redundant features, so the main purpose of feature selection is to remove the irrelevant and redundant features. In this paper, we use the wrapper method to implement feature selection to verify our proposed algorithm.

A. TEST DATA SET DESCRIPTION

In this article, we used ten data test sets from the UCI database in machine learning. These datasets have different attributes and examples. The specific contents are shown in Table 8.

B. EXPERIMENTAL SIMULATION

Feature selection is to identify useful features and useless features from the original data set. In the simulation experiment of feature selection through the Binary SOS algorithm, we mark the useful features as 1 and the useless features as 0.

1) k-Nearest Neighbor(KNN)

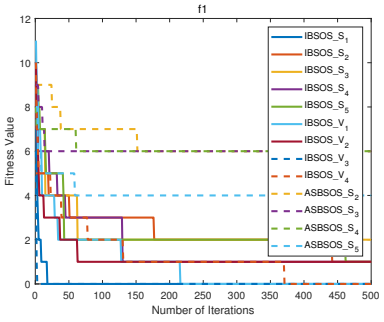
K Nearest Neighbor (KNN) is one of the relatively simple classification algorithms that are popular among researchers. The algorithm is to divide a sample data into k most similar samples in the sample space, most samples belong to a certain category, we think that the entire sample also belongs to this category. Although KNN relies on the limit theorem (based on the classification of one or more samples in the classification decision) to make decisions, it is only relevant to a very small number of adjacent samples. The KNN method is more suitable for the division or overlap of the sample set because the KNN classification method is based on a small number of adjacent samples in the periphery, rather than by dividing the category area.

In KNN, the dissimilarity index between each sample is the distance between samples. This index is to avoid the problem of matching between samples. As shown in eqs.21-22, the distance we solve generally uses two methods: Euclidean distance or Manhattan distance.

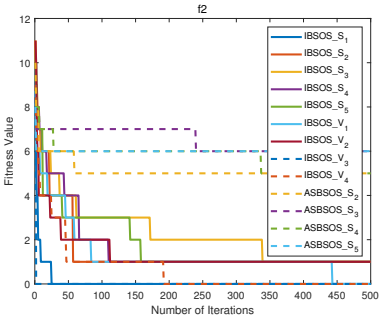
$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}, \tag{21}$$

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{k} |x_i - y_i|}, \tag{22}$$

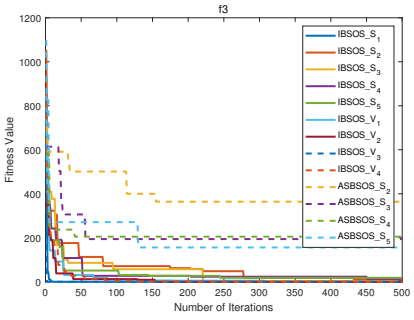
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access



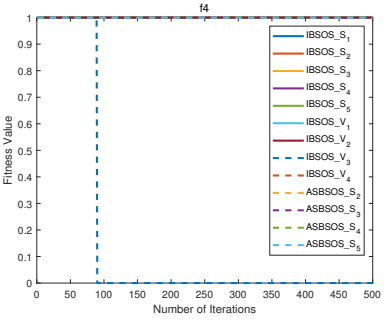
(a) Sphere



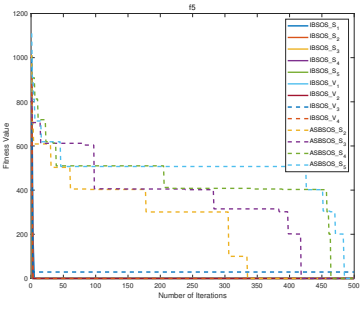
(b) Schwefel's function 2.21



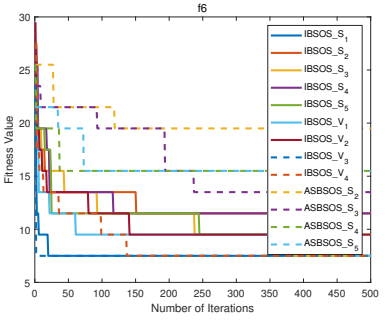
(c) Schwefel's function 1.2



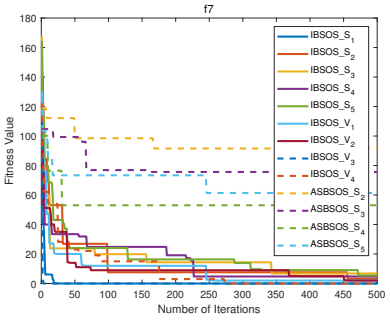
(d) Schwefel's function 2.22



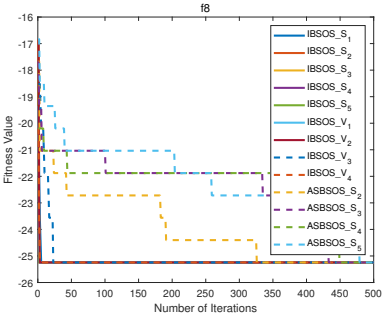
(e) Rosenbrock



(f) Step



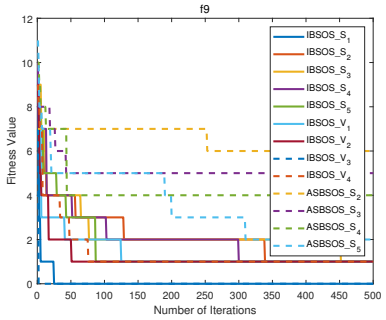
(g) Dejong's noisy



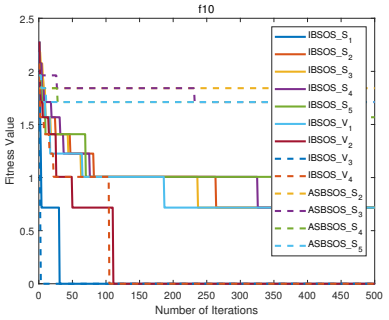
(h) Schwefel

FIGURE 4: Comparison of the convergence process of ASBSOS and IBSOS based on the f_1-f_8 test function.

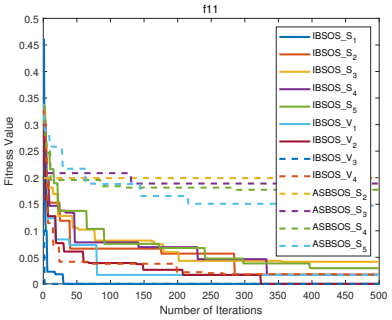
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access



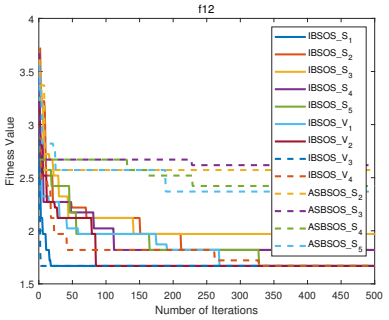
(a) Rastringin



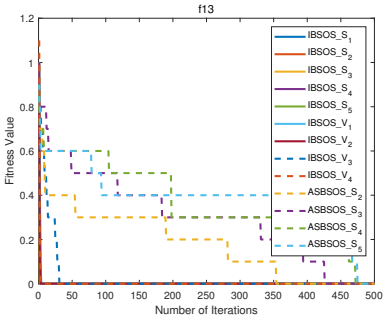
(b) Ackley



(c) Griwwank



(d) Generalized penalized 1



(e) Generalized penalized 2

FIGURE 5: Comparison of the convergence process of ASBSOS and IBSOS based on the f_9-f_{13} test function.

Where $\mathbf{x} = \{x_1, x_2, ..., x_k\}$ represents training data, $\mathbf{y} = \{y_1, y_2, ..., y_k\}$ represents test data, and k represents the number of features contained in the original data.

2) K-fold cross validation

In machine learning, we usually divide the data into a training set and a test set. The K-Fold cross-validation method is to divide the data into K groups, of which the training set used as the model is $K-1$ group, and the remaining one is used as the test set for model evaluation. K-Fold cross-validation mainly includes the following steps.

Step 1: The original data is randomly divided into K groups.

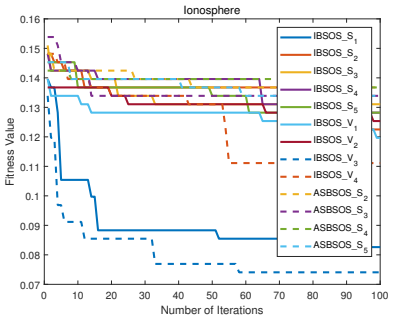
Step 2: One of the K groups is used as the test data set, and the remaining $K-1$ group is used as the training data set.

Step 3: Repeat Step 2 K times, so that each subset has a chance to be used as a test set, and the remaining opportunities are used as a training set.

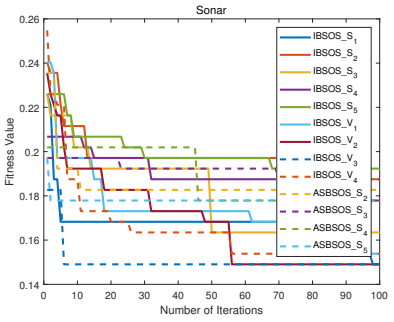
Step 4: We use the model to test the corresponding test set, and at the same time calculate and save the evaluation index value of the model.

Step 5: Calculate the average value of K sets of test results as the performance index of the model under the current k-fold cross-validation, and as an index to judge whether the model is accurate.

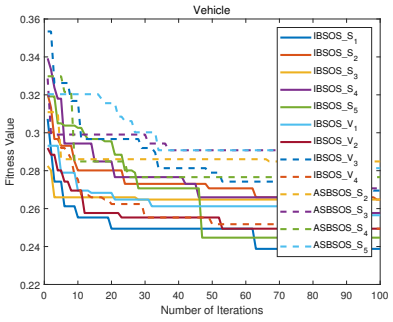
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access



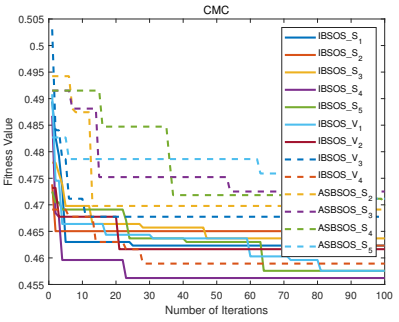
(a) Ionosphere



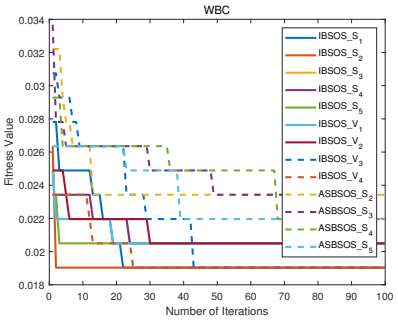
(b) Sonar



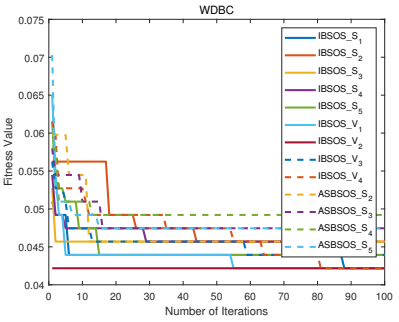
(c) Vehicle



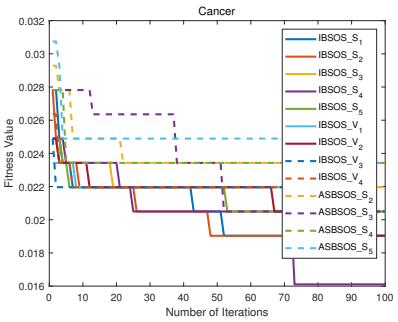
(d) CMC



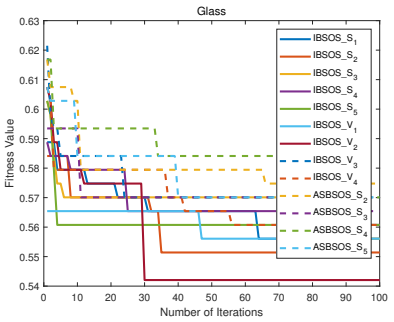
(e) WBC



(f) WDBC



(g) Cancer



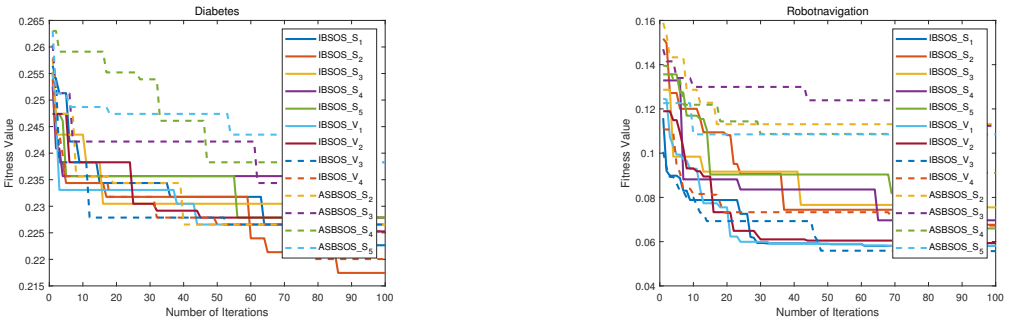
(h) Glass

FIGURE 6: Performance comparison of IBSOS and ASBSOS based on 9 test data sets in UCI

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

TABLE 8: The details of the testing datasets.

S.no.	Datasets	Instances	Number of classes (k)	Number of features (d)	Size of classes
1	Ionosphere	351	2	34	126,225
2	Sonar	208	2	60	97,111
3	Vehicle	846	3	18	199,217,218,212
4	CMC	1473	3	9	629,333,511
5	WBC	683	2	9	444,239
6	WDBC	569	2	30	357,212
7	Cancer	683	2	9	444,239
8	Glass	214	6	9	29,76,70,17,13,9
9	Diabetes	768	2	8	268,500
10	Robotnavigation	5456	4	25	82,620,972,205,329



(a) Diabetes (b) Robotnavigation

FIGURE 7: Performance comparison of IBSOS and ASBSOS based on 2 test data sets in UCI

TABLE 9: Comparison results of all transfer functions in the BSOS algorithm.

Fun.	Winner
f_1	$IBSOS\_S_1, IBSOS\_V_3$
f_2	$IBSOS\_S_1, IBSOS\_V_3$
f_3	$IBSOS\_S_1, IBSOS\_V_3$
f_4	$IBSOS\_V_3$
f_5	$IBSOS\_S_1, IBSOS\_S_2, IBSOS\_S_3, IBSOS\_S_4,$ $IBSOS\_S_5, IBSOS\_V_1, IBSOS\_V_2, IBSOS\_V_4$
f_6	$IBSOS\_S_1, IBSOS\_V_3$
f_7	$IBSOS\_S_1$
f_8	all
f_9	$IBSOS\_S_1, IBSOS\_V_3$
f_{10}	$IBSOS\_S_1, IBSOS\_V_3$
f_{11}	$IBSOS\_S_1, IBSOS\_V_3$
f_{12}	$IBSOS\_S_1, IBSOS\_V_3$
$f13$	all

TABLE 10: The average error of the nine transfer formulas of the BSOS algorithm repeated ten times under the ten test datasets of UCI.

Datasets	Ionosphere	Sonar	Vehicle	CMC	WBC	WDBC	Cancer	Glass	Diabetes	Robotnavigation
$ASBSOS\_S_2$	0.13248	0.18413	0.27293	0.46626	0.02167	0.04728	0.02138	0.56495	0.23477	0.10902
$ASBSOS\_S_3$	0.13447	0.18558	0.27506	0.46741	0.02196	0.04745	0.0224	0.57056	0.23268	0.11153
$ASBSOS\_S_4$	0.13504	0.1899	0.2766	0.47094	0.02211	0.04657	0.02225	0.56869	0.23685	0.10158
$ASBSOS\_S_5$	0.13419	0.18558	0.27612	0.47189	0.02284	0.04745	0.02284	0.56963	0.23555	0.10511
$IBSOS\_S_1$	0.08148	0.13798	0.25142	0.46158	**0.01933**	0.04323	0.0202	0.55561	0.22292	**0.0566**
$IBSOS\_S_2$	0.12678	0.18269	0.25839	0.45893	0.02035	0.04411	0.02035	0.55093	0.22878	0.08057
$IBSOS\_S_3$	0.12849	0.17837	0.25969	**0.45811**	0.02006	0.04446	0.02035	**0.5486**	0.22708	0.08032
$IBSOS\_S_4$	0.1245	0.18125	0.26052	0.46015	0.02035	0.04429	0.0202	0.55748	0.22643	0.0787
$IBSOS\_S_5$	0.12906	0.17788	0.2604	0.46117	0.01962	0.04464	**0.01991**	0.55421	0.22747	0.07859
$IBSOS\_V_1$	0.11766	0.16442	**0.24929**	0.45913	0.01977	**0.04271**	**0.01991**	0.55607	**0.2224**	0.06593
$IBSOS\_V_2$	0.11425	0.16923	0.25059	0.45879	0.01962	**0.04271**	0.0202	0.55421	0.22279	0.06692
$IBSOS\_V_3$	**0.07037**	**0.13702**	0.26478	0.46409	0.02255	0.04429	0.02138	0.56121	0.22812	0.05876
$IBSOS\_V_4$	0.11795	0.16538	0.25083	0.45961	0.01977	**0.04271**	0.0202	0.5528	0.22305	0.06585

TABLE 11: The squared difference of the nine transfer formulas of the BSOS algorithm repeated ten times under the ten test datasets of UCI.

Datasets	Ionosphere	Sonar	Vehicle	CMC	WBC	WDBC	Cancer	Glass	Diabetes	Robotnavigation
$ASBSOS\_S_2$	0.00489	0.0104	0.00518	0.00496	0.00115	0.00226	0.00102	0.01175	0.00288	0.00859
$ASBSOS\_S_3$	0.00225	0.00993	0.00566	0.00347	0.00069	0.00219	0.00121	0.01111	0.00592	0.00788
$ASBSOS\_S_4$	0.00275	0.00793	0.00489	0.00195	0.00108	0.0019	0.00093	0.01103	0.0035	0.01037
$ASBSOS\_S_5$	0.00314	0.0116	0.00794	0.00405	0.00123	0.00117	0.00102	0.00837	0.0063	0.00708
$IBSOS\_S_1$	0.00674	0.01015	0.00273	0.00308	0.00115	0.00123	0.00093	0.00465	0.00411	0.00209
$IBSOS\_S_2$	0.00726	0.00934	0.0052	0.00244	0.00046	0.0013	0.00083	0.01384	0.00222	0.00462
$IBSOS\_S_3$	0.00561	0.01025	0.00566	0.00486	0.00071	0.00119	0.00083	0.01149	0.00247	0.00485
$IBSOS\_S_4$	0.00891	0.01342	0.00611	0.00377	0.00083	0.00111	0.00062	0.00765	0.00344	0.00709
$IBSOS\_S_5$	0.00466	0.00907	0.00413	0.00189	0.00076	0.00091	0.00076	0.0083	0.00503	0.00535
$IBSOS\_V_1$	0.00871	0.00956	0.00297	0.00325	0.00077	0.00119	0.00102	0.00697	0.00486	0.00577
$IBSOS\_V_2$	0.00973	0.00779	0.0061	0.00338	0.00102	0.00085	0.00093	0.00914	0.00355	0.005
$IBSOS\_V_3$	0.00446	0.01307	0.00825	0.00292	0.00123	0.002	0.00141	0.00777	0.00442	0.00473
$IBSOS\_V_4$	0.00618	0.00823	0.00235	0.00297	0.00077	0.00085	0.00062	0.00765	0.00319	0.00304

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045043, IEEE Access

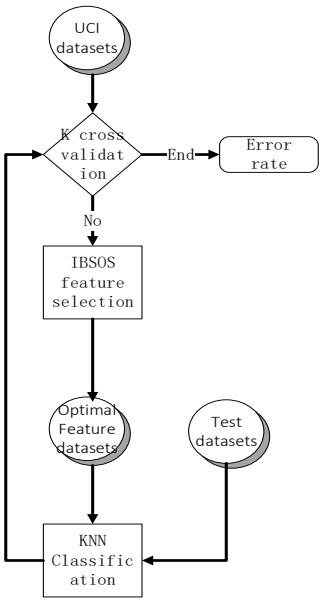


FIGURE 8: Architecture of feature selection based on IBSOS.

3) Fitness Functions

In the process of feature selection, it is very important for the classification of features. We need to calculate the classification error value and the number of feature subsets to assess the merits of classification (feature selection). In the simulation process, we can choose the following fitness function as the evaluation standard during the operation of the algorithm.

$$\mathbf{F} = M \times Kfold + N \times \frac{T}{C} \tag{23}$$

Where T represents the number of useful subsets selected from the original features, C represents the features of the original data, and $Kflod$ represents the classification error of K-fold cross validation. M and N are two coefficients, where the value of M is set to 0.99 and N is set to 0.01 in [7].

The complete system architecture based on IBSOS feature selection is shown in FIGURE 8.

C. RESULT ANALYSIS

In the simulation experiment of feature selection, we compare the advantages and disadvantages of the nine BSOS transfer functions to the performance of the BSOS algorithm through the ten test datasets of UCI. In the simulation experiment, the average value of the error we obtained by repeating the operation 10 times is shown in Table 10, and in Table 11 we give the squared difference of each transfer function repeated 10 times in each test data set. In the KNN simulation experiment, the parameter value of NumNeighbors is set to 5, and the parameter value of $KFold$ is set to 2 in cross-validation.

We can see from the above two tables that in the two test datasets of $Ionosphere$ and $Sonar$, the V_3 type transfer function has obtained relatively excellent performance. In the four test datasets of $Vehicle$, $WDBC$, $Cancer$, $Diabetes$, the V_1 type transfer obtains better performance. In the CMC and $Glass$ test datasets, the transfer function of the S_3 type obtained good convergence ability. The S_2 type transfer function obtains good feature selection capabilities based on two test datasets, WBC and $Robotnavigation$. In general, V_1 transfers have obtained better feature selection capabilities in UCI's ten test datasets.

Table 6 and Table 7 show the performance differences in feature selection for ASBSOS and IBSOS algorithms based on different transfer functions. We can see that the performance of the proposed IBSOS algorithm is better than the ASBSOS algorithm for feature selection overall. In Ionosphere, Sonar and Robotnavigation three instance test data set $IBSOS\_V_3$ obtained good convergence ability. $IBSOS\_S_1$ has a good performance in the test data set of Vehicles. $IBSOS\_S_4$ has the best performance in the CMC and Cancer test data sets. The performance of $IBSOS\_S_2$ in the test data set of WBC and Diabetes is the best. $IBSOS\_V_2$ has a better ability to select features in the two test data sets of WDBC and Glass.

VI. CONCLUSION

In this paper, we use V-type and S-type transfer functions to improve the performance of the BSOS algorithm and compare the impact of each transfer function on the BSOS algorithm. These transfer functions have different convergence capabilities for the BSOS algorithm. Experimental results show that the transfer function is very important for the performance of the algorithm. In this algorithm, we have given nine transfer functions to binarize the three stages of the original continuous SOS algorithm. We use 23 benchmark functions to verify the performance of the BSOS algorithm and the difference in the performance of the nine transfer functions for the BSOS algorithm, and through the machine learning UCI's ten test datasets, the performance of the BSOS algorithm in the practical application of feature selection to verify the practicality of the algorithm in practice, and further verify the performance difference of each transfer function for the BSOS algorithm in the feature selection experiment. In the experimental results, the V-type transfer function has shown a relatively good optimization performance in the BSOS algorithm simulation experiment or feature selection.

In future research, we will further study various optimization strategies of swarm intelligence algorithms to apply to various complex problems in real life. For example, the surrogate-assisted model cite sun2017surrogate is used to solve the extremely time-consuming and expensive optimization problem of feature selection. In addition, we will also work on applying intelligent optimization algorithms to

different scenarios in real life, such as energy supply, routing plans, and node coverage issues in wireless sensor networks.

ACKNOWLEDGMENT

This research was funded by National Natural Science Foundation of China (grant number NSF 215 61872085), Project 2018Y3001 of Fujian Provincial Department of Science and Technology and Natural Science Foundation of Fujian Province (grant number 2018J01638).

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, 2020.

[2] A. Peña-Ayala, "Educational data mining: A survey and a data mining-based analysis of recent works," Expert systems with applications, vol. 41, no. 4, pp. 1432–1462, 2014.

[3] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Computers & Electrical Engineering, vol. 40, no. 1, pp. 16–28, 2014.

[4] Y. Zhang, R. Liu, X. Wang, H. Chen, and C. Li, "Boosted binary harris hawks optimizer and feature selection," Engineering with Computers.

[5] L.-Y. Chuang, S.-W. Tsai, and C.-H. Yang, "Improved binary particle swarm optimization using catfish effect for feature selection," Expert Systems with Applications, vol. 38, no. 10, pp. 12699–12707, 2011.

[6] X. Zhao, D. Li, B. Yang, H. Chen, X. Yang, C. Yu, and S. Liu, "A two-stage feature selection method with its application," Computers & Electrical Engineering, vol. 47, pp. 114–125, 2015.

[7] P. Hu, J.-S. Pan, and S.-C. Chu, "Improved binary grey wolf optimizer and its application for feature selection," Knowledge-Based Systems, 2020.

[8] D. Moldovan, I. Anghel, T. Cioara, and I. Salomie, "Adapted binary particle swarm optimization for efficient features selection in the case of imbalanced sensor data," Applied Sciences, vol. 10, no. 4, p. 1496, 2020.

[9] J.-S. Pan, Q.-W. Chai, S.-C. Chu, and N. Wu, "3-d terrain node coverage of wireless sensor network using enhanced black hole algorithm," Sensors, vol. 20, no. 8, p. 2411, 2020.

[10] L. Nicolescu, C. Galalae, and A. Voicu, "Solving a supply chain management problem to near optimality using ant colony optimization, in an international context," Amfiteatru Economic Journal, vol. 15, no. 33, pp. 8–26, 2013.

[11] Y. Liu and L. Huang, "Supply chain finance credit risk assessment using support vector machine–based ensemble improved with noise elimination," International Journal of Distributed Sensor Networks, vol. 16, no. 1, p. 1550147720903631, 2020.

[12] P. Lenuwat and S. Boon-Itt, "Service supply chain management process capabilities: A theoretical framework and empirical study," in 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 248–252, IEEE, 2019.

[13] A. E.-S. Ezugwu and A. O. Adewumi, "Discrete symbiotic organisms search algorithm for travelling salesman problem," Expert Systems with Applications, vol. 87, pp. 70–78, 2017.

[14] Y. Zhou, R. Wang, C. Zhao, Q. Luo, and M. A. Metwally, "Discrete greedy flower pollination algorithm for spherical traveling salesman problem," Neural Computing and Applications, vol. 31, pp. 2155–2170, 2019.

[15] A. E. Eiben and M. Schoenauer, "Evolutionary computing," Information Processing Letters, vol. 82, no. 1, pp. 1–6, 2002.

[16] H. Wang, W. Wang, Z. Cui, X. Zhou, J. Zhao, and Y. Li, "A new dynamic firefly algorithm for demand estimation of water resources," Information Sciences, vol. 438, pp. 95–106, 2018.

[17] A.-Q. Tian, S.-C. Chu, J.-S. Pan, H. Cui, and W.-M. Zheng, "A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station," Sustainability, vol. 12, no. 3, p. 767, 2020.

[18] J.-S. Pan, J.-B. Li, and Z.-M. Lu, "Adaptive quasiconformal kernel discriminant analysis," Neurocomputing, vol. 71, no. 13-15, pp. 2754–2760, 2008.

[19] S.-C. Chu, X. Xue, J.-S. Pan, and X. Wu, "Optimizing ontology alignment in vector space," Journal of Internet Technology, vol. 21, no. 1, pp. 15–22, 2020.

[20] C.-l. Sun, J.-c. Zeng, and J.-s. Pan, "An improved vector particle swarm optimization for constrained optimization problems," Information Sciences, vol. 181, no. 6, pp. 1153–1163, 2011.

[21] F. Corno, M. S. Reorda, and G. Squillero, "The selfish gene algorithm: a new evolutionary optimization strategy," in Proceedings of the 1998 ACM symposium on Applied Computing, pp. 349–355, 1998.

[22] P. Hu, J.-S. Pan, S.-C. Chu, Q.-W. Chai, T. Liu, and Z.-C. Li, "New hybrid algorithms for prediction of daily load of power network," Applied Sciences, vol. 9, no. 21, p. 4514, 2019.

[23] T.-T. Nguyen, J.-S. Pan, and T.-K. Dao, "A compact bat algorithm for unequal clustering in wireless sensor networks," Applied Sciences, vol. 9, p. 1973, 2019.

[24] T.-T. Nguyen, J.-S. Pan, and T.-K. Dao, "An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network," IEEE Access, vol. 7, pp. 75985–75998, 2019.

[25] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," IEEE Transactions on Evolutionary Computation, vol. 21, no. 4, pp. 644–660, 2017.

[26] J.-S. Pan, P. Hu, and S.-C. Chu, "Novel parallel heterogeneous meta-heuristic and its communication strategies for the prediction of wind power," Processes, vol. 7,

no. 11, p. 845, 2019.

[27] H. Yu, Y. Tan, J. Zeng, C. Sun, and Y. Jin, "Surrogate-assisted hierarchical particle swarm optimization," Information Sciences, vol. 454, pp. 59–72, 2018.

[28] R. Eberhart and J. Kennedy, "Particle swarm optimization," in Proceedings of the IEEE international conference on neural networks, vol. 4, pp. 1942–1948, 1995.

[29] J.-F. Chang, J. F. Roddick, J.-S. Pan, and S. Chu, "A parallel particle swarm optimization algorithm with communication strategies," Journal of Information Science and Engineering, vol. 21, pp. 809–818, 2005.

[30] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 39, no. 6, pp. 1362–1381, 2009.

[31] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J.-S. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," Information Sciences, vol. 223, pp. 119–135, 2013.

[32] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Transactions on evolutionary computation, vol. 1, no. 1, pp. 53–66, 1997.

[33] S.-C. Chu, J. F. Roddick, and J.-S. Pan, "Ant colony system with communication strategies," Information Sciences, vol. 167, Issues 1-4, no. 1-4, pp. 63–76, 2004.

[34] X. Zhao, D. Li, B. Yang, C. Ma, Y. Zhu, and H. Chen, "Feature selection based on improved ant colony optimization for online detection of foreign fiber in cotton," Applied Soft Computing, vol. 24, pp. 585–596, 2014.

[35] Z. Meng, J.-S. Pan, and H. Xu, "Quasi-affine transformation evolutionary (quatre) algorithm: a cooperative swarm based algorithm for global optimization," Knowledge-Based Systems, vol. 109, pp. 104–121, 2016.

[36] J.-S. Pan, Z. Meng, H. Xu, and X. Li, "Quasi-affine transformation evolution (quatre) algorithm: A new simple and accurate structure for global optimization," in International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 657–667, Springer, 2016.

[37] N. Liu, J.-S. Pan, J. Wang, and T.-T. Nguyen, "An adaptation multi-group quasi-affine transformation evolutionary algorithm for global optimization and its application in node localization in wireless sensor networks," Sensors, vol. 19, no. 19, p. 4112, 2019.

[38] Z.-G. Du, J.-S. Pan, S.-C. Chu, H.-J. Luo, and P. Hu, "Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks," IEEE Access, vol. 8, 2020.

[39] S.-C. Chu, X. Xue, J.-S. Pan, and X. Wu, "Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks," journal of Internet Technology, 2020.

[40] S.-C. Chu, P.-W. Tsai, and J.-S. Pan, "Cat swarm optimization," in Pacific Rim international conference on artificial intelligence, pp. 854–858, Springer, 2006.

[41] L. Kong, J.-S. Pan, P.-W. Tsai, S. Vaclav, and J.-H. Ho, "A balanced power consumption algorithm based on enhanced parallel cat swarm optimization for wireless sensor network," International Journal of Distributed Sensor Networks, vol. 729680, no. 3, pp. 1–10, 2015.

[42] P.-W. Tsai, J.-S. Pan, S.-M. Chen, and B.-Y. Liao, "Enhanced parallel cat swarm optimization based on the Taguchi method," Expert Systems with Applications, vol. 39, no. 7, pp. 6309–6319, 2012.

[43] P.-W. Tsai, J.-S. Pan, S.-M. Chen, B.-Y. Liao, and S.-P. Hao, "Parallel cat swarm optimization," in 7th International Conference on Machine Learning and Cybernetics, vol. 6, pp. 3328–3333, IEEE, 2008.

[44] J.-S. Pan, P.-C. Song, S.-C. Chu, and Y.-J. Peng, "Improved compact cuckoo search algorithm applied to location of drone logistics hub," Mathematics, vol. 8, no. 3, 2020.

[45] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," Neural Computing and Applications, vol. 24, pp. 1659–1669, 2014.

[46] J. Tvrdík and R. Poláková, "Competitive differential evolution applied to cec 2013 problems," in 2013 IEEE Congress on Evolutionary Computation, pp. 1651–1657, 2013.

[47] Z. Meng, J.-S. Pan, and K.-K. Tseng, "Pade: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization," Knowledge-Based Systems, vol. 168, no. 9, pp. 80–99, 2019.

[48] H. Wang, S. Rahnamayan, H. Sun, and M. G. Omran, "Gaussian bare-bones differential evolution," IEEE Transactions on Cybernetics, vol. 43, no. 2, pp. 634–647, 2013.

[49] J.-S. Pan, N. Liu, and S.-C. Chu, "A hybrid differential evolution algorithm and its application in unmanned combat aerial vehicle path planning," IEEE Access, vol. 8, pp. 17691–17712, 2020.

[50] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," Applied soft computing, vol. 11, no. 1, pp. 652–657, 2011.

[51] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, and J.-S. Pan, "Multi-strategy ensemble artificial bee colony algorithm," Information Sciences, vol. 279, pp. 587–603, 2014.

[52] P.-W. Tsai, M. K. Khan, J.-S. Pan, and B.-Y. Liao, "Interactive artificial bee colony supported passive continuous authentication system," IEEE Systems Journal, vol. 8, no. 2, pp. 395–405, 2012.

[53] M. Abdullahi, M. A. Ngadi, S. I. Dishing, M. J. Usman, et al., "A survey of symbiotic organisms search algorithms and applications," Neural Computing and Applications, vol. 32, pp. 547–566, 2020.

[54] S. S. Pal, S. Samui, and S. Kar, "A new technique for time series forecasting by using symbiotic organisms

search," Neural Computing and Applications, vol. 32, pp. 2365–2381, 2020.

[55] K. H. Truong, P. Nallagownden, I. Elamvazuthi, and D. N. Vo, "An improved meta-heuristic method to maximize the penetration of distributed generation in radial distribution networks," Neural Computing and Applications, 2019.

[56] A. E. Ezugwu and D. Prayogo, "Symbiotic organisms search algorithm: theory, recent advances and applications," Expert Systems with Applications, vol. 119, pp. 184–209, 2019.

[57] A. E. Ezugwu and A. O. Adewumi, "Soft sets based symbiotic organisms search algorithm for resource discovery in cloud computing environment," Future Generation Computer Systems, vol. 76, pp. 33–50, 2017.

[58] M. Abdullahi, M. A. Ngadi, et al., "Symbiotic organism search optimization based task scheduling in cloud computing environment," Future Generation Computer Systems, vol. 56, pp. 640–650, 2016.

[59] F. Chakraborty, D. Nandi, and P. K. Roy, "Oppositional symbiotic organisms search optimization for multilevel thresholding of color image," Applied Soft Computing, vol. 82, p. 105577, 2019.

[60] K. H. Truong, P. Nallagownden, Z. Baharudin, and D. N. Vo, "A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems," Applied Soft Computing, vol. 77, pp. 567–583, 2019.

[61] Y. Zhou, F. Miao, and Q. Luo, "Symbiotic organisms search algorithm for optimal evolutionary controller tuning of fractional fuzzy controllers," Applied Soft Computing, vol. 77, pp. 497–508, 2019.

[62] A. E. Ezugwu, "Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times," Knowledge-Based Systems, vol. 172, pp. 15–32, 2019.

[63] S.-C. Chu, Z.-G. Du, and J.-S. Pan, "Symbiotic organism search algorithm with multi-group quantum-behavior communication scheme applied in wireless sensor networks," Appl. Sci., vol. 10, no. 3, p. 930, 2020.

[64] C. Han, G. Zhou, and Y. Zhou, "Binary symbiotic organism search algorithm for feature selection and analysis," IEEE Access, vol. 7, pp. 166833–166859, 2019.

ZHI-GANG DU received his B.S. degree from Qindao College, Qingdao Technological University, Qingdao, China, in 2017. He is currently pursuing a master's degree with the Shandong University of Science and Technology, Qingdao, China. His recent research interests are swarm intelligence and artificial neural networks.
Email: 1461779873@qq.com

JENG-SHYANG PAN received the B.S. degree in electronic engineering from the National Taiwan University of Science and Technology in 1986, the M.S. degree in communication engineering from National Chiao Tung University, Taiwan, in 1988, and the Ph.D. degree in electrical engineering from the University of Edinburgh, U.K., in 1996. He is currently the Director of the Fujian Provincial Key Lab of Big Data Mining and Applications, and an Assistant President with the Fujian University of Technology. He is also a Professor at the Harbin Institute of Technology. He is the IET Fellow, U.K., and has been the Vice-Chair of the IEEE Tainan Section.
Email:jspan@ieee.org

SHU-CHUAN CHU received the Ph.D. degree in 2004 from the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia. She joined Flinders University in December 2009 after 9 years at the Cheng Shiu University, Taiwan. She is the Research Fellow in the College of Science and Engineering of Flinders University, Australia from December 2009. Currently, She is the Research Fellow with PhD advisor in the College of Computer Science and Engineering of Shandong University of Science and Technology from September 2019. Her research interests are mainly in Swarm Intelligence, Intelligent Computing and Data Mining.
Email: scchu0803@gmail.com

YI-JUI CHIU obtained his Ph.D. degree in Mechanical Engineering from the National Taiwan University of Science and Technology, Taiwan in 2008. He is a Professor in Xiamen University of Technology. His current research interests are vibration and rotor dynamic.
Email: chiuyijui@xmut.edu.cn