A Self-Adaption Butterfly Optimization Algorithm for Numerical Optimization Problems

YUQI FAN, JUNPENG SHAO, GUITAO SUN, AND XUAN SHAO
Key Laboratory of Advanced Manufacturing and Intelligent Technology, School of Mechanical and Power Engineering, Ministry of Education, Harbin University of Science and Technology, Harbin 150080, China
Corresponding author: Xuan Shao (xuanshao84@163.com)
This work was supported by the International Cooperation Projects under Grant 2012DFR70840.

ABSTRACT For shortcomings of poor exploration and parameter complexities of the butterfly optimization algorithm, an improved butterfly optimization algorithm based the self-adaption method (SABOA) was proposed to extremely enhance the searching accuracy and the iteration capability. SABOA has advantages of having fewer parameters, the simple algorithm structure, and the strong precision. First, a new fragrance coefficient was added to the basic butterfly optimization algorithm. Then, new iteration and updating strategies were introduced in global searching and local searching phases. Finally, this paper tested different optimization problems including low-high functions and constrained problems, and the obtained results were compared with other well-known algorithms, this paper also drew various mathematical statistics figures to comprehensively analyze searching performances of the proposed algorithms. The experimental results show that SABOA can get less number of function evaluations compared to other considered algorithms, which illustrates that SABOA has great searching balance, large exploration, and high iterative speed.

INDEX TERMS Butterfly optimization algorithm, global optimization, constrained problem.

I. INTRODUCTION
In applied mathematics and engineering fields, there are numerous optimization problems whose calculated solutions are in a large and complex searching space. Traditional optimization methods, including the steepest descent method, the conjugate gradient method, the variable scale method, and Newton method, can only deal with objective functions that are simple, continuously differentiable, and high order differentiable [1]–[3]. With the increasing of problem diversities and problem complexities, traditional optimization methods can not meet different requirements of higher calculation speed and lower average percentage error, so it is crucial to find for new optimization methods that have fast calculation speed and perfect convergence abilities [4], [5].

With the development of artificial intelligence, digitization, and computer technologies, numerous meta-heuristic optimization algorithms have been increasingly proposed and applied in science and engineering fields [6]–[8]. Meta-heuristic algorithms own characteristics of self-organizing, mutual compatibility, simplicity, parallelism, wholeness and harmony. Meta-heuristic algorithms work by that a group of random solutions carry out activities of repeating the feedback process to approach the desired goal, and the result of each iteration will become the initial value of the next iteration [9], [10]. In this searching mechanism, the algorithm only needs to know the objective function and the searching scope, and the aim solution can be obtained whether the searching scope is continuously derivable or not. Therefore, Meta-heuristic algorithms are widely applied in many fields, such as obstacle avoidance [11], automatic control [12], scheduling problems [13], image evolution [14], path planning [15], information feedback models [16] and so on. Meta-heuristic algorithms are mainly divided into three parts, including the biological evolutionary method, the species living habit, and the nature phenomena. The biological evolutionary method, such as the genetic algorithm, is inspired by strategies of organisms heredity and evolution. The species living habit approach is based on the cooperation between each individual in a group of organisms, including monarch butterfly optimization (MBO) [17], shuffled frog-leaping algorithm (SFLA) [18], elephant herding optimization (EHO) [19], krill herd (KH) [20], moth-flame Optimization Algorithm (MFO) [21], earthworm optimization algorithm (EWA) [22], grey wolf optimizer (GWO) [23], artificial bee colony algorithm (ABC) [24],
moth search algorithm (MS) [25], salp swarm algorithm (SSA) [26], pigeon-inspired optimization (PIO) [27]. The nature phenomena part is inspired by the physical rules in nature, such as artificial chemical reaction optimization algorithm (ACROA) [28], artificial flora optimization Algorithm (AF) [29], sine cosine algorithm (SCA) [30], gravitational search algorithm (GSA) [31], charged system search (CSS) [32], flower pollination algorithm (FPA) [33].

Butterfly optimization algorithm (BOA), which is inspired by swarming and mating behaviors of butterflies, is proposed by Arora and Singh [34], [35]. BOA mainly consists of two basic parts including the global search phase and the local search phase [36]. In the global search phase, disparate solutions are obtained by exploring the searching space. In the local search phase, more detailed solutions are got by exploiting information about the current optimum solution. Because of the unique optimization mechanism, many scholars are interested in BOA, and BOA has been widely introduced in the science and engineering fields [37], [38].

There are two initial parameters including the initial sensory modality coefficient and the initial power exponent coefficient. And initial parameters not only influence iteration time but also result in precocious problems. If unsuitable parameters are set in BOA before optimization operation, the BOA will have unsteady local exploitation and slow convergence speed. However, the initial sensory modality coefficient and the initial power exponent coefficient in BOA are chosen by engineers experiences, which causes a large extent of fuzziness and randomness in BOA. To improve the performance of BOA and overcome its deficiencies, this paper proposed an improved butterfly optimization algorithm based the self-adaption method (SABOA). This combination of the adaption method and basic BOA will assure that the calculated solution can reach the searching aim, also avoid local defects through parameter randomization. To achieve a more comprehensive analysis of the proposed algorithm, this paper selects different testing functions and constrained problems to verify the effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows.

Section 2 introduces the basic BOA. The proposed algorithm SABOA is presented in Section 3. In Section 4, this paper uses SABOA and comparison algorithms to calculate low dimension functions and high dimension functions. Constrained problems are discussed in Section 5. The conclusions are presented in Section 6.

### II. THE PROPOSED HEURISTIC OPTIMIZATION ALGORITHM

Butterfly optimization algorithm (BOA) is an inspired meta-heuristic algorithm based on living habits of butterflies. When a butterfly moves from one position to another position in the searching space, the butterfly can generate fragrance which will vary accordingly. Butterflies can sense the fragrance and get attracted toward the butterfly which generates the maximum concentration of the fragrance, and the propagated fragrance is sensed by butterflies and a collective fragrance network is formed. When butterflies are in the fragrance network, they will fly to the butterfly that can generate the maximum concentration of the fragrance and this phase is termed as the global search phase. When butterflies are not able to sense the fragrance network, they will fly randomly and this phase is called the local search phase. In other words, all butterflies in searching space fly to new positions and then their fitness values are calculated by functions in each iteration. So the butterfly optimization algorithm mainly consists of the global search phase and the local search phase. Searching for food and mating partners by butterflies can occur at both local and global phases.

Set the position vector of each butterfly in $T \times N$ dimensional space, where $T$ means the number of iterations and $N$ represents population size. Hence, the position vector of each individual can be represented by a multidimensional matrix in (1).

$$x_i^t = \begin{bmatrix} x_{11}^t & x_{12}^t & \cdots & x_{1N}^t \\ x_{21}^t & x_{22}^t & \cdots & x_{2N}^t \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1}^t & x_{N2}^t & \cdots & x_{NN}^t \end{bmatrix}$$ (1)

where $t$ indicates the current iteration, $x_{it}$ is the position of the $i$-th butterfly.

In the global search phase, the next position of each butterfly can be determined by (2).

$$x_{i}^{t+1} = x_{i}^t + (r^2 \times g^* - x_{i}^t) \times f$$ (2)

where $g^*$ means the best position in searching space. $r$ is a random number in the range of $[0,1]$. $f$ is the fragrance coefficient which means the perceived magnitude of the fragrance, all butterflies can send out some fragrance that makes butterflies attract each other, and $f$ can be updated in (3).

$$f = c t^a$$ (3)

where $c$ is the sensory modality coefficient, $I$ means the stimulus intensity, and $a$ is the power exponent coefficient. The parameter $a$ controls the algorithm behavior. Another parameter $c$ is also a crucial parameter in determining the speed of convergence and how the BOA algorithm behaves.

Parameters $a$ and $c$ crucially affect the convergence speed of the algorithm. BOA takes $a$ and $c$ in the range $[0,1]$. $I$ means the fitness value of the solution.

In the local search phase, the next position of a butterfly can be determined by (4).

$$x_{i}^{t+1} = x_{i}^t + (r^2 \times x_{j}^t - x_{i}^t) \times f$$ (4)

where $t$ indicates the current iteration. $x_{i}^t$ is the position of the $i$-th butterfly, $x_{j}^t$ and $x_{k}^t$ are the positions of the $j$-th and $k$-th butterflies in the searching space.

Butterflies can search for food at both local and global scales. BOA considers physical proximity and various other
factors including rain and wind, and butterflies have a significant fraction \( p \) in searching activities. So the switch probability \( p \) is applied in BOA to switch between common global searching phase to intensive local searching phase. The parameter \( p \) is in the range of \([0, 1]\).

The iterative process of the Butterfly Optimization Algorithm can be presented as follows:

Step 1: In the initialization phase, the algorithm defines the objective function and its solution space. Set the maximum number of iterations \( T \). Set \( t = 1 \). Randomly generate \( N \) butterflies positions \( x_{i}^{t} \). Determine searching range and all initial parameters including \( a \) and \( c \). Define switch probability \( p \).

Step 2: Calculate the fragrance coefficient of each butterfly using (3). Record the best butterfly position whose concentration of the fragrance is maximum.

Step 3: Randomly generate \( r \) in the range of \([0, 1]\). Contrast \( r \) with the switch probability \( p \), if \( r \) is less than \( p \), update the next position of each butterfly using (2). If \( r \) is larger or equal to \( p \), update the next position of each butterfly using (4).

Step 4: Update the power exponent \( a \) coefficient.

Step 5: Calculate each fitness value and compare all fitness values. Record and replace the best solution and the best fitness value if there is a better solution.

Step 6. Calculate \( t = t + 1 \). Judge whether \( t \) equals to \( T \). If no, return to step 2 and go on. Otherwise, stop iterative loops.

The BOA main step can be summarized in the pseudo-code shown below:

**Algorithm 1 BOA**

**Input:** Searching range. Maximum iterations \( T \). Set \( t=1 \). \( N \) positions \( x_{i}^{t} \). Function \( f(\cdot) \). Initial optimum solution \( g^{*} \). Initial optimum value \( f(g^{*}) \). Set \( p, c, a \).

**Output:** \( g^{*}, f(g^{*}) \).

while \((t < T)\)

for \( i = 1 : N \)

\( f = c.f^{a} \)

Update \( r \in [0, 1] \).

if \( 1 \) \((r < p)\)

\( x_{i}^{t+1} = x_{i}^{t} + (r^{2} \times g^{*} - x_{i}^{t}) \times f \)

else

\( x_{i}^{t+1} = x_{i}^{t} + (r^{2} \times x_{j}^{t} - x_{k}^{t}) \times f \)

end if \( 1 \)

if \( 2 \) \((f(x_{i}^{t}) \) is better than \( f(g^{*})\)

\( g^{*} = x_{i}^{t} \)

\( f(g^{*}) = f(x_{i}^{t}) \)

end if \( 2 \)

end for

\( t = t + 1 \)

end while

### III. SELF-ADAPTATION BUTTERFLY OPTIMIZATION ALGORITHM

In BOA, the initial sensory modality coefficient \( c \) and the initial power exponent coefficient of the fragrance coefficient are selected according to artificial experiences. A lot of experiments are needed before the initial \( c \) and the initial \( a \) is selected. However, many industry experiments can not be carried out many times because of the limitation of materials and the bad situation of experimental environments. In the search phase, the random number \( r \) greatly affects the searching efficiency. When \( r \) is selected too large, too large random parameter makes that BOA has high randomization, therefore, it is easy to jump from one region to another region, which causes low searching accuracy and low searching efficiency.

To obtain a better optimization result in BOA, we introduce an improved butterfly optimization algorithm based self-adaption method, and the proposed algorithm is called SABOA. To eliminate the stochastic behavior and blindness of the fragrance coefficient in BOA, the self-adaption method is added into the new fragrance coefficient. The self-adaption method is set to make the fragrance distribution range smaller and smaller so that the algorithm can search more and more carefully. When the searching process begins, the initial fragrance distribution range is large. Then, the fragrance distribution range gradually decreases as the iterative process goes on, which ensures the convergence speed and ability to jump out of the local extreme point.

The new fragrance coefficient can be updated in (5).

\[
f_{\text{new}} = u \times \left(1 - \frac{t}{T}\right) \tag{5}\]

where \( t \) indicates the current iteration, \( T \) is the maximum iteration. To create a more balanced distribution of the fragrance, \( u \) obeys the standard normal distribution using the computation program.

The ideal optimization process includes the high searching ability of the early phase and the great precision ability of the later phase. For the global search phase, SABOA works by adjusting the difference between the best solution and each butterfly position of the current iteration.

Thus, SABOA makes each individual have a more careful and thorough the searching mechanism around the best solution of the current iteration. The new position updating formula in the global search phase can be expressed as follows.

\[
x_{i}^{t+1} = g^{*} + \left(x_{i}^{t} - g^{*}\right) \times f_{\text{new}} \tag{6}\]

where \( g^{*} \) means the best position in the searching space.

For the local search phase, SABOA uses the average value of the best position and the worst position in each iteration to update the next position of each butterfly, which can reduce the blindness of randomly selecting two butterflies. The new position updating formula in the local search phase can be
expressed as follows.

\[
x_{i}^{t+1} = \frac{1}{2} \times \left( g^* + w^* \right) \times f_{i}^{new}
\]  

(7)

where \( w^* \) means the worst position in the searching space.

Step 1: In the initialization phase, the algorithm defines the objective function and its solution space. Set the maximum number of iterations \( T \). Set \( t = 1 \). Randomly generate \( N \) butterflies positions \( x_{i}^{t} \) \((i=1,2,...,N)\), \((t=1,2,...,T)\). Determine searching range. Define switch probability \( p \). Calculate each fitness value and compare all fitness values. Record the best solution \( g^* \) and the worst solution \( w^* \).

Step 2: Set \( u \) in the standard normal distribution. Calculate the fragrance coefficient of each butterfly using (5).

Step 3: Randomly update \( r \) in the range of \([0, 1]\). Contrast \( r \) with the switch probability \( p \), if \( r \) is less than \( p \), update the next position of each butterfly using (6). Otherwise, update the next position of each butterfly using (7).

Step 4: Calculate each fitness value. Record the best solution, the best fitness value, the worst solution, and the worst fitness value in the current generation. Replace the best solution \( g^* \) and the worst fitness value if there is a better solution. Replace the worst solution \( w^* \) and the worst fitness value if there is a worse solution.

Step 5: Calculate \( t = t + 1 \). Judge whether \( t \) equals to \( T \). If no, return to step 2 and go on. Otherwise, stop iterative loops.

The flow chart of SABOA implementation in Figure 1.

The SABOA main step can be summarized in the pseudo-code shown below:

SABOA uses the self-adaption method to get a better optimization result for basic BOA, and advantages of the proposed algorithm mainly include three aspect. First, the self-adaption method can make the fragrance distribution factor range smaller and smaller, so that SABOA can find the extremum more and more carefully. Second, the self-adaption method can ensure the convergence ability to jump out of the local extreme point. Three, the proposed algorithm works by adjusting the difference between the best solution and each solution, which can wander around the best solution in the current iteration. However, SABOA has more computational complexities than the basic BOA because SABOA need to calculate the best solution and the worst solution in each iteration to enhance the searching precision, which can increase the calculation amount of the proposed algorithm.

**Algorithm 2 SABOA**

Input: Searching range. Maximum iterations \( T \). Set \( t=1 \). \( N \) positions \( x_{i}^{t} \) \((i=1,2,...,N)\). Function \( f(.) \). Initial optimum solution \( g^* \). Initial optimum value \( f(g^*) \). Initial worst solution \( w^* \). Initial worst value \( f(w^*) \).

Output: \( g^* , f(g^*) \).

while \((t < T)\)

for \( i=1:N \)

Update \( u \) in the Gaussian distribution.

\[ f = u(\cdot / T) \]

Update \( r \in [0, 1] \).

if \( 1 (r < p) \)

\( x_{i}^{t+1} = g^* + \left( x_{i}^{t} - g^* \right) \times f_{i}^{new} \)

else

\( x_{i}^{t+1} = 0.5 \times \left( g^* + w^* \right) \times f_{i}^{new} \)

end if 1

if \( 2 f(x_{i}^{t}) \) is better than \( f(g^*) \)

\( g^* = x_{i}^{t} \)

\( f(g^*) = f(x_{i}^{t}) \)

end if 2

if \( 3 f(x_{i}^{t}) \) is worse than \( f(w^*) \)

\( w^* = x_{i}^{t} \)

\( f(w^*) = f(x_{i}^{t}) \)

end if 3

end for

\( t = t + 1 \)

end while

High dimension functions have many local optimization solutions, which increases the deception of problem-solving. When high dimension functions are calculated, the solution space and the complexity will increase exponentially with the increasing of dimensions. So, high dimension functions can test capabilities to solve complex functions. The low dimension function usually has a single optimization solution, and it is easy to find the best aim. In the original BOA algorithm article, literatures [34], [35] compared BOA with seven algorithms, and literatures [34], [35] showed that BOA had the best searching ability in all testing algorithms. In order to analyze the proposed algorithm comprehensively, this paper compared the proposed method with other algorithms that are different from the mentioned seven algorithms of original BOA literature. Testing algorithms in this paper include flower pollination algorithm (FPA) [33], salp swarm algorithm (SSA) [26], sine cosine algorithm (SCA) [30], lévy flight trajectory based whale optimization algorithm (LWOA) [39]. All initial parameters of testing algorithms were chosen by relevant algorithm literatures.

FPA was proposed by Xin-She Yang. There are two searching phases including biotic and cross-pollination in FPA. Biotic and cross-pollination can be seen as the global pollination process with pollen-carrying pollinators performing lévy flight trajectory. In FPA, parameter \( p \) was equal to 0.8, parameter \( \beta \) was equal to 1.5.
SSA, that imitated behaviors of predation and swarming of salp swarm, was proposed by Seyedali Mirjalili et al. in 2017. There are three parameters in SSA. Parameter $c_1 = 2e^{-(4t/T)}$, where $t$ is the current iteration, $T$ is the maximum iterations.
The parameter $c_2$ and $c_3$ were selected in the interval of $[0,1]$. Probability was selected 0.5.

SCA was proposed by Seyedali Mirjalili in 2015. There are four parameters in SCA. Parameters $r_1, r_2, r_3$, and $r_4$ influence the searching movement direction, the searching scope, the destination weight, and balanced switching, respectively. Parameter $r_1 = a - at/T$, where $t$ is the current iteration, $T$ is the maximum iterations. For SCA, $a = 2$, $r_2 = 2\pi$, $r_3$ and $r_4$ were selected in $[0, 1]$.

LWOA was proposed by Ying Ling et al. in 2017. LWOA is an improved version of the whale optimization algorithm (WOA). Lévy flight trajectory is added in WOA to improve the convergence rate and the precision of the basic WOA. LWOA has four parameters including $r$, $b$, $l$, and $p$. Parameters $r$ and $p$ were random numbers uniformly generated in the interval of $[0, 1]$. Parameter $l$ was selected in $[-1,1]$. Parameter $b$ was equal to 2. Parameter $\beta$ was equal to 1.5.

In BOA, modular modality $c$ was 0.01 and power exponent $a$ was increased from 0.1 to 0.3. The switch probability $p$ was equal to 0.8.

In SABOA, switch probability $p$ was equal to 0.8. Initial parameters of comparison algorithms were selected optimum parameters by original algorithm literatures. Operational details of comparison algorithms are in original algorithm literatures. The population size and the maximum iterations were 50 and 500, respectively. Each algorithm was independently run ten times in MATLAB.

To make a fair comparison, all algorithms were programmed in MATLAB (R2014b, The MathWorks, Inc, Natick, MA, USA). All the experiments were conducted on a laptop with Intel (R) Core (TM) i5-4210U CPU, 2.30 GHz, 4 GB RAM. All data and figures were completed in MATLAB (R2014b, The MathWorks, Inc, Natick, MA, USA).

### B. DATA DISCUSSIONS

In Table 2 and Table 3, Best, Worst, Mean, Std respectively represent the optimal fitness value, the worst fitness value, the average fitness value and standard deviation when functions are calculated ten times. Table 2 and Table 3 show the results of the algorithm in different dimensions. With the statistical results, we can make sure that the results are not generated by chance. From the results presented in Table 2 and Table 3, SABOA is able to get the best values among different testing functions in comparison with the BOA, FPA, LWOA, SCA, and SSA, and the best value is closer to the optimal point of Table 1 in each function. For BOA and SABOA, most functions solutions of BOA are unsatisfactory. SABOA can get the ideal accuracy for almost functions, it can achieve a high precision level of $f_3, f_6, f_7, f_9$, and $f_{11} - f_{14}$. With the increase of the searching dimension, the difficulty faced by the algorithm will emerge problems of low solution accuracy and solving speed slowing down. The proposed SABOA in this paper can also calculate each fitness function effectively with the best function value. Standard deviation can reflect the discrete degree of a data set, the large standard deviation performs a large difference between most calculated values and the average of all values, and the small standard deviation represents that calculated values are closer to the average of all values. To discuss and analyze the efficiency and robustness of the proposed algorithm, this paper computes the mean and standard deviation to indicate how reliable SABOA is.

We can easily find out that the SABOA algorithm has the smallest standard deviation in each function, which can show that SABOA has a steady searching ability and robustness than other algorithms in low- high dimension problems. The numerical results in Table 2 and Table 3 can demonstrate that SABOA has a consistent and reliable performance to get the global minimum, and SABOA is obviously better than the original BOA.

### C. ITERATIONS DISCUSSIONS

Iterations, whose aims are to approach desired results, are activities of repeating feedback action, and the result of each iteration will be the initial value of the next iteration. Figure 2 shows the average iterations curves of all algorithms disposing different functions under 10 independent runs. It must be mentioned here that all the curves in Figure 2 are the average convergence curves. In Figure 2, the proposed SABOA algorithm has the fastest iteration speed in all iteration curves, which shows that SABOA can enhance the diversity of achievable solutions within shorter computation times compared to other algorithms.

For BOA, the low dimension iteration speed is larger than the high dimension iteration speed. And the iteration speed of BOA in low dimension is Rapid and intense, but the iteration speed of BOA in high dimension is Slow and slow. BOA has the worst searching result in function 5. FPA only in function 5 has the good convergence curve and iteration performance, other FPA curves are all in a state of precocity and falling into the local optimum solution, especially for function 8 to function 12. LWOA has the good convergence effect, it can jump out the local optimum expect function 1 and function 8. SCA has the slow iteration speed in the early searching phase, especially for function 1 to function 4. SCA has the worst searching result in function 3, so SCA can not meet the convergence requirements under the fast convergence environment. SSA in all functions can not carefully search the optimal solution. SABOA also can overcome BOA shortcomings of premature convergence and poor local searching abilities, having the fastest convergence speed of the early phase and the best searching accuracy of the later optimization phase. Therefore, these results demonstrate that the SABOA has an outstanding ability to explore feasible regions and escaping from local optimal solution in the searching scope, and the adaption mechanism can make SABOA effectively avoid the local minimum and searching extreme points. The fastest iteration speed reflects that SABA can quickly lead each butterfly to the close-by optimal point. Considering all the advantages discussed above, the optimal solution can be got successfully by applying SABOA, which outperforms the basic BOA for all testing functions.
TABLE 2. Comparison of results for low dimension functions.

| Function | Index | Algorithm       |       |       |       |       |
|----------|-------|-----------------|-------|-------|-------|-------|
|          |       | SABOA           | BOA   | FPA   | LWOA  | SCA   | SSA   |
| $f_1$    | Best  | 3.475E-11       | 2.316E-06 | 0.4158 | 3.2088E-06 | 0.0056 | 4.321E-04 |
|          | Worst | 2.6895E-05      | 1.0522 | 3.7464 | 0.9727 | 11.9519 | 5.7323  |
|          | Mean  | 5.9834E-06      | 0.1110 | 1.3916 | 0.0981 | 1.8958  | 0.7757  |
|          | Std   | 9.9756E-06      | 0.3309 | 0.9600 | 0.3073 | 3.5564  | 1.8088  |
|          | Best  | 0.3979          | 0.3984 | 0.4282 | 0.3979 | 0.4347  | 0.3979  |
|          | Worst | 0.3979          | 0.4586 | 0.6582 | 0.4008 | 6.9115  | 1.6649  |
|          | Mean  | 0.3979          | 0.4082 | 0.5201 | 0.3982 | 1.6544  | 0.6855  |
|          | Std   | 0               | 0.0188 | 0.0730 | 9.0162E-04 | 1.9947 | 0.4026  |
| $f_2$    |      |                 |       |       |       |       |       |
|          | Best  | 7.9520E-06      | 0.0188 | 4.2294E-06 | 0.0219 | 7.9248E-04 |
|          | Worst | 8.5338E-16      | 0.1401 | 0.4953 | 3.6846E-04 | 9.4456 | 0.5257  |
|          | Mean  | 8.5343E-17      | 0.0383 | 0.2234 | 9.2367E-05 | 2.4336 | 0.1299  |
|          | Std   | 2.6986E-16      | 0.0497 | 0.1680 | 1.3988E-04 | 3.6334 | 0.1727  |
| $f_3$    |      |                 |       |       |       |       |       |
|          | Best  | 2.4272E-23      | 1.1079E-06 | 0.0525 | 2.6043E-05 | 0.0040 | 0.0033  |
|          | Worst | 5.2895E-07      | 0.0143 | 0.3076 | 2.2982E-04 | 0.9988 | 0.1169  |
|          | Mean  | 5.5282E-08      | 0.0028 | 0.1705 | 9.8445E-05 | 0.2810 | 0.0586  |
|          | Std   | 1.6649E-07      | 0.0043 | 0.0881 | 7.4350E-05 | 0.3889 | 0.0502  |
| $f_4$    |      |                 |       |       |       |       |       |
|          | Best  | -1.9132         | -1.9129 | -1.9132 | -1.9132 | -1.9130 | -1.9125  |
|          | Worst | -1.9132         | -1.7398 | -1.9129 | -1.9132 | -1.9045 | -1.8704  |
|          | Mean  | -1.9132         | -1.8565 | -1.9131 | -1.9132 | -1.9106 | -1.9018  |
|          | Std   | 4.6811E-16      | 0.0770 | 9.1253E-05 | 7.1754E-06 | 0.0028 | 0.0132  |
| $f_5$    |      |                 |       |       |       |       |       |
|          | Best  | 0               | 0.0563 | 0.0088 | 2.1906E-07 | 3.1030E-05 | 5.9629E-05 |
|          | Worst | 0               | 0.0969 | 0.2319 | 1.9211E-05 | 1.3931E-06 | 0.5547  |
|          | Mean  | 0               | 0.0788 | 0.0993 | 4.9312E-06 | 1.3931E-07 | 0.1168  |
|          | Std   | 0               | 0.0111 | 0.0918 | 6.1621E-06 | 4.4053E-07 | 0.1858  |
| $f_6$    |      |                 |       |       |       |       |       |
|          | Best  | 0               | 0.0304 | 0.0029 | 6.1734E-09 | 4.8501E-07 | 2.2065E-05 |
|          | Worst | 0               | 0.0718 | 0.0404 | 1.1168E-06 | 4.8305E-05 | 0.0151  |
|          | Mean  | 0               | 0.0565 | 0.0167 | 2.9184E-07 | 5.9441E-05 | 0.0041  |
|          | Std   | 0               | 0.0133 | 0.0148 | 3.5665E-07 | 1.5290E-05 | 0.0054  |

D. BOX- PLOT CHARTS DISCUSSIONS

The box-plot chart, that is a statistical chart, can show a set of dispersed data, it can be used in several samples to analyze the symmetry. There are five indexes in a box-plot chart, including the maximum value, the minimum value, the median, the upper quartile, and the lower quartile. The algorithm stability and robustness can be confirmed by evaluating five indexes of a box-plot chart. The box-plot chart, that is a statistical chart, can show a set of dispersed data, it can be used in several samples to analyze the symmetry. The algorithm stability and robustness can be confirmed by evaluating five indexes of a box-plot chart. All box-plot charts of different algorithms after 10 independent runs are shown in Figure 3. Figure 3 clearly shows that box-plot charts of SABOA have the narrowest form, the fewest outliers, and the weakest dispersion, which indicates that the SABOA has a balanced convergence ability and the approaching effect when calculating function problems. The median, upper quartile, and lower quantile
of the proposed algorithm are lowest in Figure 3, which also displays that SABOA has an outstanding searching capability.

### E. SEARCH PATH DISCUSSIONS

Figure 4 shows three-dimensional images of low dimension functions, the optimal BOA search path and the optimal SABOA search path of low dimension functions. In a two-dimensional plane, the optimal BOA search path, the optimal SABOA search path, and the contour plot of each function are shown separately. As can be seen from Figure 4, the optimal search path of the SABOA is much less than the search path taken by the BOA, and there are numerous short-distance, repeat, invalid, and occasional long-distance paths in BOA search path. SABOA uses the searching strategy of tightening from the periphery to the middle due to group optimal constraints. Compared with the basic BOA algorithm, the main reason that the SABOA search path is obviously increased with the increasing of iterations is that SABOA has a strong searching mechanism around the best solution. And because SABOA works by adjusting the difference between the best solution and each solution, a lot of calculating time is saved. Therefore, for the proposed algorithms, a few iterations can get a better path without too much computation, which also displays that SABOA has exceedingly higher positioning accuracy. Two sets of search path experiments show that

### TABLE 3. Comparison of results for high dimension functions.

| Function | Algorithm | SABOA | BOA  | FPA  | LWOA | SCA  | SSA  |
|----------|-----------|-------|------|------|------|------|------|
| \( f_6 \) | Best      | 0.5372| 3.0047| 14.0874| 19.4267| 3.1271| 2.0667|
|          | Worst     | 0.2418| 2.8119| 12.0057| 2.3718| 2.7821| 1.5642|
|          | Mean      | 0.1466| 0.1247| 1.8856| 6.0063| 0.2283| 0.2162|
| \( f_9 \) | Best      | 8.8818E-16| 0.3032| 1.9305| 4.8749E-04| 2.0526E-07| 0.0141|
|          | Worst     | 8.8818E-16| 0.3262| 3.1238| 0.0051| 0.0591| 0.2658|
|          | Mean      | 8.8818E-16| 0.3127| 2.6206| 0.0020| 0.0074| 0.0832|
|          | Std       | 0.0082| 0.4248| 0.0015| 0.0184| 0.0818|
| \( f_{10} \) | Best     | 8.8818E-16| 12.5270| 0.2345| 3.2637E-06| 1.7648E-09| 0.0198|
|          | Worst     | 8.8818E-16| 22.2379| 13.5343| 0.0062| 0.0712| 3.5000|
|          | Mean      | 8.8818E-16| 16.7201| 3.3805| 1.1589| 0.3391|
|          | Std       | 0.0098| 3.6998| 4.3979| 1.6600| 0.5831|
| \( f_{11} \) | Best     | 8.8818E-16| 2.4549| 3.2673E-06| 1.7648E-09| 0.0198|
|          | Worst     | 8.8818E-16| 5.9556| 0.0062| 0.0712| 3.5000|
|          | Mean      | 8.8818E-16| 3.8900| 0.0015| 0.0073| 0.8661|
|          | Std       | 0.0063| 1.0439| 0.0022| 0.0225| 1.0879|
| \( f_{12} \) | Best     | 8.8818E-16| 2.3437| 7.2403E-07| 7.7931E-07| 0.1168|
|          | Worst     | 8.8818E-16| 5.8802| 0.0021| 1.7637| 1.4650|
|          | Mean      | 8.8818E-16| 4.2898| 3.5645E-04| 0.2565| 0.4751|
|          | Std       | 0.0056| 1.3594| 6.7398E-04| 0.5606| 0.4073|
| \( f_{13} \) | Best     | 8.8818E-16| 4.7638E-06| 1.1518E-10| 7.7708E-22| 1.5254E-11|
|          | Worst     | 8.8818E-16| 4.5375E-04| 1.1492E-04| 1.7595E-07| 9.4628E-05| 7.4405E-05|
|          | Mean      | 8.8818E-16| 3.0575E-04| 3.6030E-04| 5.7206E-08| 9.4773E-06| 1.0915E-05|
|          | Std       | 0.0056| 1.3594| 6.7398E-04| 0.5606| 0.4073|
FIGURE 2. Average convergence curves of functions.
FIGURE 3. Box-plot charts of functions.
FIGURE 4. Search paths of self-adaption butterfly optimization algorithm.
compared with the BOA algorithm, the SABOA algorithm not only can give a diversity of solutions but also has more flexibility.

F. THE WILCOXON RANK SUM TEST
The rank sum test, which is a nonparametric test, does not depend on the specific form of the overall distribution, dispersed type, and known distribution, which has strong practicality. The rank sum test is to arrange all observations in order from small to large, and each observation is called the rank. The non-parametric statistical test can be applied in algorithms testing to check whether the proposed algorithm is statistically significant or not [5], [40]. However, the basic rank sum test only considers the difference sign, and ignores the absolute value difference, which can cause the rough result and the loss of some tested information. To enhance the testing performance of the basic rank sum test, Wilcoxon proposed an improved testing strategy called the wilcoxon rank sum test [41]. The wilcoxon rank sum test considers the difference direction and the difference size, can also be applied to test any difference in the distribution data, which is more effective than the basic rank sum test. The results of the wilcoxon rank sum test is called the p-values, and the p-values that is less than 0.05 indicates that there is significant difference at a level of 0.05.

To have a more comprehensive algorithm discussions, this paper used the Wilcoxon rank sum test to carry out the data analysis. The test was computed by using the SABOA results against those of other comparison algorithms at 0.05 significance level. All p-values were listed in Table 4. From Table 4 we can find that the result of LWOA in function 8 is larger than 0.05, other results are all less than 0.05. The Wilcoxon rank sum test results shows that SABOA has the highest searching accuracy, the greatest searching efficiency, and the strongest searching mechanism around the best solution in all contrast algorithms, which further demonstrates the perfect performances of SABOA.

V. CONSTRAINED PROBLEM EXPERIMENTS
Constrained optimization problem mainly consists of two parts, including objective function and constraints on variables of the objective function. The optimization process is to find the optimal solution under constraints. In many engineering design fields, there are different conditions that bring great challenges to solving the processing of algorithms. In this section, the excellence and competitiveness of the proposed algorithm are examined by solving several constrained optimization problems. In order to confirm the proposed method for constraint problems, different constrained problems are examined and then, the resulting performance of SABOA against constrained problems that are commonly used in kinds of literature were tested, and the results have been compared with another well-known algorithm. In this section, the Initial parameter of SABOA is the same as the last section. The proposed algorithm was coded in MATLAB programming software.

A. CONSTRAINED PROBLEM 1
This minimization problem, which is a constrained minimization problem, originally proposed by Bracken and McCormick. The optimum solution is obtained at $X = (0.8229, 0.9114)$ with an objective function value equal to
TABLE 4. Comparison of the wilcoxon rank sum test.

| Function | BOA | FPA | LWOA | SCA | SSA |
|----------|-----|-----|------|-----|-----|
| f_1      | 5.83E-04 | 1.80E-04 | 0.0010 | 1.83E-04 | 1.83E-04 |
| f_2      | 6.39E-05 | 6.20E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_3      | 1.81E-04 | 1.78E-04 | 1.81E-04 | 1.81E-04 | 1.81E-04 |
| f_4      | 1.83E-04 | 1.79E-04 | 1.83E-04 | 1.83E-04 | 1.83E-04 |
| f_5      | 6.39E-05 | 6.20E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_6      | 6.39E-05 | 6.16E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_7      | 6.39E-05 | 6.25E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_8      | 6.39E-05 | 1.79E-04 | 0.1212 | 1.83E-04 | 1.83E-04 |
| f_9      | 6.39E-05 | 6.29E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_10     | 6.39E-05 | 6.25E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_11     | 6.39E-05 | 6.25E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_12     | 6.39E-05 | 6.25E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_13     | 6.39E-05 | 6.20E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |
| f_14     | 6.39E-05 | 6.20E-05 | 6.39E-05 | 6.39E-05 | 6.39E-05 |

TABLE 5. Comparison of results constrained problem 1 results.

| Algorithm   | x_1  | x_2  | g_1(x) | g_2(x) | f(x)  |
|-------------|------|------|--------|--------|-------|
| GA [42]     | 0.8080 | 0.9594 | 3.7E-02 | 5.2E-02 | 1.4339 |
| EP [43]     | 0.8350 | 0.9125 | 1.05E-02 | -7.06E-02 | 1.3772 |
| HS [44]     | 0.8343 | 0.9121 | 5.06E-03 | 5.4E-03 | 1.3770 |
| SABOA       | 0.822876 | 0.911437 | 2.6E-06 | 1.0E-06 | 1.393464316 |
| optimum     | 0.8229 | 0.9114 | 7.05E-09 | 1.73E-08 | 1.393454 |

\( f(X) = 1.393454 \). The problem can then be expressed as:

\[
\begin{align*}
\min f(x) &= (x_1 - 2)^2 + (x_2 - 1)^2 \\
\text{s.t.} \quad &g_1(x) = x_1 - 2x_2 + 1 = 0 \\
&g_2(x) = -(x_2^2/4) - x_2^2 + 1 \geq 0 \\
&-10 \leq x_1, \quad x_2 \leq 10
\end{align*}
\]

Abdollah Homaifar et al. solved this problem using GA [42]. David B. Fogel compared the results of evolutionary programming (EP) with results got by GA [43]. Kang Seok Lee and Zong Woo Geem used the harmony search (HS) method and found their best solution to this problem [44].

In GA an EP, the population size equals to 400, the number of Iterations equals to 40000. HS found the best solution after approximately 40,000 searches. In SABOA, the population size and iterations respectively selected 100 and 2000. Table 5 shows the comparison of results and the corresponding calculated variables among the five algorithms. As can be seen from Table 5, the results got by SABOA is minimum, and SABOA exceeded another contrasted algorithm in terms of the final value and constraint accuracy. Although EP and HS can find the smaller solution, they only retain two decimals and three decimals respectively in constraints, their searching accuracy and the number of iterations are less than that of SABOA.

TABLE 6. Comparison of constrained problem 2 results.

| Algorithm   | Best | Worst | Mean | Std | NFEs |
|-------------|------|-------|------|-----|------|
| ABC [45]    | 0.75 | 0.75  | 0.75 | 0   | 2.4E+05 |
| ASCHEA [46] | 0.75 | NOT  | 0.75 | NOT | 1.5E+06 |
| DE [47]     | 0.74900 | 0.74900 | 0.74900 | NOT | 3E-04 |
| DELC [48]   | 0.750 | 0.750 | 0.750 | 0   | 5E-04 |
| DEDS [49]   | 0.7499 | 0.7499 | 0.7499 | 0   | 2.25E+05 |
| HEAA [50]   | 0.750 | 0.750 | 0.750 | 3.4E-16 | 2E-05 |
| HM [51]     | 0.75 | 0.75 | 0.75 | NOT | 1.4E+06 |
| IRS [52]    | 0.750 | 0.750 | 0.750 | 1.1E-16 | 137,200 |
| PSO [53]    | 0.750 | 0.998823 | 0.860530 | 8.4E-02 | 70,100 |
| PSO-DE [53] | 0.749999 | 0.750001 | 0.749999 | 2.5E-07 | 70,100 |
| SAPF [54]   | 0.749 | 0.757 | 0.751 | 2E-03 | 5E-05 |
| SMES [55]   | 0.75 | 0.75 | 0.75 | 1.52E-04 | 7.5E+04 |
| SR [56]     | 0.750 | 0.750 | 0.750 | 8E-05 | 3.5E+05 |
| SABOA       | 0.7499 | 0.7499 | 0.7499 | 0 | 100000 |

### B. CONSTRAINED PROBLEM 2

For this constrained problem 2, SABOA was compared with different algorithms [45]–[56].

\[
\begin{align*}
\min f(x) &= x_1^2 + (x_2 - 1)^2 \\
\text{s.t.} \quad &g(x) = x_2 - x_2^2 = 0 \\
&-1 \leq x_1, \quad x_2 \leq 1
\end{align*}
\]
functions, which display that the SABOA can maintain a fair balance between exploration and exploitation.

For benchmark functions, we drew iteration graphs, box-plots, and search paths to statistically test the performance of the proposed algorithm, and the results demonstrate that the proposed algorithm is undoubtedly different from compared algorithms in benchmark functions.

For three constraint Functions, we compared searching results of different literatures, and computational results based on the comprehensive comparative investigation, exhibit the attractiveness of the SABOA for handling numerous constraints. All calculated results illustrate that the SABOA can give better solutions than other compared algorithms in terms of function values and the number of function evaluations and iterations. In general, SABOA offers excellently competitive abilities compared with original BOA due to the experimental results.

In the future study, this proposed algorithm will be applied in engineering problems to find appropriate industry parameters.

### REFERENCES

[1] A. Slowik and H. Kwasnicka, “Nature inspired methods and their industry applications—Swarm intelligence algorithms,” *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1004–1015, Mar. 2018.

[2] D. Corus and P. S. Oliveto, “Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 720–732, Oct. 2018.

[3] W.-Z. Sun, J.-S. Wang, and X. Wei, “An improved whale optimization algorithm based on different searching paths and perceptual disturbance,” *Symmetry*, vol. 10, no. 6, p. 210, 2018.

[4] X. Cai, J. Zhang, H. Liang, L. Wang, and Q. Wu, “An ensemble bat algorithm for large-scale optimization,” *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 11, pp. 3099–3113, Nov. 2019.

[5] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and S. Deb, “A hybrid method based on krill herd and quantum-behaved particle swarm optimization,” *Neural Comput. Appl.*, vol. 27, no. 4, pp. 989–1006, May 2016.

[6] X. Cai, P. Wang, L. Du, Z. Cui, W. Zhang, and J. Chen, “Multi-objective three-dimensional DV-hop localization algorithm with NSGA-II,” *IEEE Sensors J.*, vol. 19, no. 21, pp. 10003–10015, Nov. 2019.

[7] S. Zhang and S. Liu, “A discrete improved artificial bee colony algorithm for 0–1 knapsack problem,” *IEEE Access*, vol. 7, pp. 104982–104991, 2019.

[8] G.-G. Wang, A. H. Gandomi, X. Zhao, and H. C. E. Chu, “Hybridizing harmony search algorithm with cuckoo search for global numerical optimization,” *Soft Comput.*, vol. 20, no. 1, pp. 273–285, Jan. 2016.

[9] S. Arora, H. Singh, M. Sharma, S. Sharma, and P. Anand, “A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection,” *IEEE Access*, vol. 7, pp. 26343–26361, 2019.

[10] X. Cai, X. Z. Gao, and Y. Xue, “Improved bat algorithm with optimal forage strategy and random disturbance strategy,” *Int. J. Bio-Inspired Comput.*, vol. 8, no. 4, pp. 205–214, 2016.

[11] A. H. Khan, S. Li, and X. Luo, “Obstacle avoidance and tracking control of redundant robotic manipulator: An RNN-based Metaheuristic approach,” *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4670–4680, Jul. 2020.

[12] R.-E. Precup, R.-C. David, and E. M. Petriu, “Grey wolf optimizer based-tuning of fuzzy control systems with reduced parameter sensitivity,” *IEEE Trans. Ind. Electron.*, vol. 64, no. 1, pp. 527–534, Jan. 2017.

[13] Z. Zhang, F. Hu, and N. Zhang, “Ant colony algorithm for satellite control resource scheduling problem,” *Int. J. Speech Technol.*, vol. 48, no. 10, pp. 3295–3305, Oct. 2018.

[14] P. Civicioglu and E. Besdok, “Bernstein-search differential evolution algorithm for numerical function optimization,” *Expert Syst. Appl.*, vol. 138, Dec. 2019, Art. no. 112831.

---

**TABLE 7. Comparison of constrained problem 3 results.**

| Algorithm | Best | Worst | Mean | Std | NFEs |
|-----------|------|-------|------|-----|------|
| ABC [45]  | $-6961.814$ | $-6961.805$ | $-6961.813$ | 2E-03 | 2.4E+05 |
| ASCHEA [46] | $-6961.81$ | NOT | $-6961.81$ | NOT | 1.5E+06 |
| DE [47]  | $-6961.814$ | $-6961.814$ | $-6961.814$ | NOT | 1.5E+04 |
| DELC [48] | $-6961.814$ | $-6961.814$ | $-6961.814$ | 7.3E-10 | 2E+04 |
| DEDS [49] | $-6961.814$ | $-6961.814$ | $-6961.814$ | 0 | 2.2E+05 |
| HEAA [50] | $-6961.814$ | $-6961.814$ | $-6961.814$ | 4.8E-12 | 2E+05 |
| HM [51]  | $-692.1$ | $5477.3$ | $6342.6$ | NOT | 1.4E+06 |
| ISR [52] | $-6961.814$ | $-6961.814$ | $-6961.814$ | 1.9E-12 | 168.800 |
| PSO [53] | $-6961.81388$ | $-6961.81388$ | $-6961.81387$ | 6.5E-06 | 1.4E+05 |
| PSO-DE [55] | $-6961.81388$ | $-6961.81388$ | $-6961.81388$ | 2.3E-09 | 1.4E+05 |
| SAPF [54] | $-6961.046$ | $6943.304$ | $-6953.061$ | 5.876 | 51.05 |
| SMES [55] | $-6961.814$ | $-6962.482$ | $-6961.284$ | 1.85 | 2.4E+05 |
| SR [56]  | $-6961.814$ | $-6550.262$ | $-6875.940$ | 160 | 1.1E+05 |
| SABOA | $-6961.8138743$ | $-6961.81382225$ | $-6961.8138658$ | 1.61E-05 | 200000 |

---

that the parameter value is not given in the original literature. From Table 6, most of the algorithms considered in this paper can find the best solution, however, SABOA obtained the best result compared to all considered algorithms in terms of NFEs as shown in Table 6. PSO and PSO-DE have less NFEs, but SABOA has higher accuracy. Table 6 shows that the proposed algorithm has a superiority factor of reaching the near-optimal value in the early iterations.

### VI. CONCLUSION

In the present article, a new SABOA algorithm, which has fewer parameters and strong exploitation, is proposed to solve the global optimization problem. The self-adaption method is introduced in the proposed algorithm to approximate the closest candidate solution and employ for the exploitation of the promising searching space. The simulation results show that SABOA performs better than BOA in optimizing 14 benchmark functions including seven low-dimensional functions and seven high-dimensional functions, and 3 constrained
S. Arora and S. Singh, “Butterfly optimization algorithm: A novel meta-heuristic algorithm with information feedback models,” IEEE Trans. Cybern., vol. 49, no. 2, pp. 542–555, Feb. 2019.

G. G. Wang, S. Deb, and Z. Cui, “Monarch butterfly optimization,” Neural Comput. Appl., vol. 31, no. 7, pp. 1995–2014, 2019.

M. Eusuff, K. Lansey, and F. Pasha, “Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization,” Eng. Optim., vol. 38, no. 2, pp. 129–154, Mar. 2006.

G. G. Wang, S. Deb, X. Z. Gao, and L. D. S. Coelho, “A new Metaheuristic optimisation algorithm motivated by elephant herding behaviour,” Int. J. Bio-Inspired Comput., vol. 8, no. 6, pp. 394–409, 2016.

A. H. Gandomi and A. H. Alavi, “Krrl herd: A new bio-inspired optimization algorithm,” Commun. Nonlinear Sci. Numer. Simul., vol. 17, no. 12, pp. 4831–4845, Dec. 2012.

S. Mirjalili, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,” Knowl.-Based Syst., vol. 89, pp. 226–249, Nov. 2015.

G. G. Wang, S. Deb, and L. D. S. Coelho, “Earthworm optimization algorithm: A bio-inspired Metaheuristic algorithm for global optimization problems,” Int. J. Bio-Inspired Comput., vol. 1, no. 1, p. 1, 2015, doi: 10.1504/IJBIC.2015.10004283.

S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” Adv. Eng. Softw., vol. 69, pp. 46–61, Mar. 2014.

D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm,” J. Global Optim., vol. 39, no. 3, pp. 459–471, Oct. 2007.

G.-G. Wang, “A bio-inspired Metaheuristic algorithm for global optimization problems,” Memetic Comput., vol. 10, no. 2, pp. 151–164, Jun. 2018.

S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Farsis, and S. M. Mirjalili, “Salp swarm algorithm: A bio-inspired optimizer for engineering design problems,” Adv. Eng. Softw., vol. 114, pp. 163–191, Dec. 2017.

Z. Cui, J. Zhang, Y. Wang, Y. Cao, X. Cai, W. Zhang, and J. Chen, “A pigeon-inspired optimization algorithm for many-objective optimization problems,” Sci. China Inf. Sci., vol. 62, no. 7, Jul. 2019.

B. Alatas, “ACROA: Artificial chemical reaction optimization algorithm for global optimization,” Expert Syst. Appl., vol. 38, no. 10, pp. 13170–13180, Sep. 2011.

L. Cheng, X.-H. Wu, and Y. Wang, “Artificial flora (AF) optimization algorithm,” Appl. Soft Comput., vol. 8, no. 5, p. 329, 2018.

S. Mirjalili, “SCA: A sine cosine algorithm for solving optimization problems,” Knowl.-Based Syst., vol. 96, pp. 120–133, Mar. 2016.

E. Rashidi, H. Nezamabadi-pour, and S. Saryazdi, “GSA: A gravitational search algorithm,” Inf. Sci., vol. 179, no. 13, pp. 2232–2248, Jun. 2009.

A. Kaveh and S. Talatahari, “A novel heuristic optimization method: Charged system search,” Acta Mechanica, vol. 213, nos. 3–4, pp. 267–289, Sep. 2010.

X.-S. Yang, M. Karamanoglu, and X. He, “Flower pollination algorithm: A novel approach for multiobjective optimization,” Eng. Optim., vol. 46, no. 9, pp. 1222–1237, Sep. 2014.

S. Arora and S. Singh, “Butterfly algorithm with Lévy flights for global optimization,” in Proc. Int. Conf. Signal Process., Comput. Control (ISPCC), Waknaghat, Town, Sep. 2015, pp. 220–224.

S. Arora and S. Singh, “Butterfly optimization algorithm: A novel approach for global optimization,” Soft Comput., vol. 23, no. 3, pp. 713–734, Feb. 2019.

G. Li, F. Shuang, P. Zhao, and C. Le, “An improved butterfly optimization algorithm for engineering design problems using the cross-entropy method,” Symmetry, vol. 11, no. 8, p. 1049, 2019.

S. Arora and P. Anand, “Binary butterfly optimization approaches for feature selection,” Expert Syst. Appl., vol. 116, pp. 147–160, Feb. 2019.

S. Arora and S. Singh, “Node localization in wireless sensor networks using butterfly optimization algorithm,” Arab J. Sci. Eng., vol. 42, no. 8, pp. 3325–3335, Aug. 2017.

Y. Ling, Y. Zhou, and Q. Luo, “Lévy flight trajectory-based whale optimization algorithm for global optimization,” IEEE Access, vol. 5, pp. 6168–6186, 2017.
JUNPENG SHAO received the B.S., M.S., and Ph.D. degrees in hydraulic drive from the Harbin Institute of Technology, in 1982, 1987, and 1991, respectively. He has been a Professor and a Ph.D. Supervisor with the Harbin University of Science and Technology. His current research interests include hydraulic drive, intelligent control, and hydraulic quadruped robots.

GUlTAO SUN received the B.S., M.S., and Ph.D. degrees in mechatronics engineering from the Harbin University of Science and Technology, in 2009, 2012, and 2015, respectively. He has been a Lecturer with the Harbin University of Science and Technology. His current research interests include hydraulic drive, intelligent control, and hydraulic quadruped robots.

XUAN SHAO received the B.S. degree from the Beijing University of Aeronautics and Astronautics, in 2008, the M.S. degree from The University of Manchester, in 2010, and the Ph.D. degree from the Harbin University of Science and Technology, in 2018. She has been a Lecturer with the Harbin University of Science and Technology. Her main research interests include hydraulic drive, intelligent control, and hydraulic quadruped robots.