Received December 19, 2019, accepted February 20, 2020, date of publication February 24, 2020, date of current version March 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976101

An Opposition-Based Chaotic Salp Swarm Algorithm for Global Optimization

XIAOQIANG ZHAO, FAN YANG, YAZHOU HAN, AND YANPENG CUI
School of Communication and Information, Xi'an University of Posts and Telecommunications, Xi'an, 710121, China
Shaanxi Key Laboratory of Information Communication Network and Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Corresponding author: Fan Yang (kbean_yang@163.com)

This work was supported in part by the Yalong River Joint Funds of the National Natural Science Foundation of China under Grant U1965102, in part by the Science and Technology Innovation Team for Talent Promotion Plan of Shaanxi Province under Grant 2019TD-028, in part by the Special Planned Project for Serving Local Areas of Education Department of Shaanxi Provincial Government under Grant 18JC029, in part by the Science and Technology Program of Xi'an under Grant 201806117YF05NC13-2, and in part by the Science and Technology Achievement Promotion Project of Shaanxi Province under Grant 2018CG-007.

ABSTRACT The salp swarm algorithm (SSA) is a bio-heuristic optimization algorithm proposed in 2017. It has been proved that SSA has competitive results compared to several other well-known meta-heuristic algorithms on various optimization problem. However, like most meta-heuristic algorithms, SSA is prone to problems such as local optimal solution and a slow convergence rate. To solve these problems, a chaotic salp swarm algorithm based on opposition-based learning (OCSSA) is proposed. The application of opposition-based learning (OBL) guarantees a better convergence speed and better develops the search space. The chaotic local search (CLS) method is also introduced, which can improve the performance of the algorithm to obtain the global optimal solution. The performance of OCSSA is compared with that of the original SSA and some other meta-heuristic algorithms on 28 benchmark functions with unimodal or multimodal characteristics. The experimental results show that the performance of OCSSA, with an appropriate chaotic map, is better than or comparable with the SSA and other meta-heuristic algorithms.

INDEX TERMS Salp swarm algorithm, global optimization, meta-heuristic algorithms, opposition-based learning, chaotic local search.

I. INTRODUCTION

Meta-heuristic algorithms have become popular due to their advantages of simple and easy implementation, effective avoidance of local optimization, and good scalability. Many meta-heuristic algorithms have shown efficient and powerful performance in solving high-dimensional and nonlinear optimization problems [1].

Without considering its structure, a meta-heuristic algorithm can to some extent be divided into the two main phases of exploration and exploitation [2]. In the exploration phase, algorithms conduct random expansion exploration on the whole search space to increase the diversity of solutions. Following this, the exploitation phase aims to improve the quality of the solution by performing local searches around promising areas that have been identified during the exploration phase. It is important to maintain a good balance between exploration and exploitation to avoid a suboptimal solution that is locally optimal.

Meta-heuristic algorithms can be divided into evolutionary and swarm intelligence algorithms. They are designed in accordance with the collective and intelligent behavior of insects, animals, humans, and other social creatures. Among the most prominent are particle swarm optimization (PSO) [3], the whale optimization algorithm (WOA) [4], the artificial bee colony algorithm (ABC) [5], and the grey wolf optimizer (GWO) [6], [7]. Meta-heuristic algorithms that have emerged in recent years include the butterfly optimization algorithm (BOA) [8], the vampire bat optimizer (VBO) [9], and the salp swarm algorithm (SSA) [10].

Mirjalili et al. recently proposed a bio-inspired meta-heuristic algorithm, SSA, that mimics salp swarm mechanisms. It has the advantages of simple implementation, few parameters, and low computational cost. In terms of optimal solution accuracy and convergence rate, SSA provides better results than common methods such as PSO,

The associate editor coordinating the review of this manuscript and approving it for publication was Huiping Li.

genetic algorithm (GA) [11], firefly algorithm (FA) [12], and bat algorithm (BA) [13]. One of the key works on SSA was conducted by Sayed et al. in [14]. The authors proposed a new chaotic SSA (CSSA) to deal with feature selection tasks. The simulation result demonstrated that the CSSA can be regarded as a good optimizer compared to some previous methods. Asaithambi and Rajappa [15] integrated SSA with sin cosine algorithm (called HSSASCA) to improve the convergence performance with the exploration and exploitation stage. The simulation results revealed that the proposed algorithm achieves the best accuracy with least runtime in comparison with other meta-heuristics. SSA was also applied in electric engineering, Singh *et al.* [16] proposed a hybrid SSA to optimize the sizing of a CMOS differential amplifier and the comparator circuit. The experiment showed that the proposed SSA with CMOS analog IC designs outperformed other existing methods. However, SSA still has its own limitations, it is more prone to exploitation phase, so it cannot always conduct global search well and, in some cases, cannot find the global optimal solution [14]. Furthermore, although SSA has a competitive performance in single- objective problems, it still has room for improvement in dealing with multi-objective problems.

The chaotic local search is one of the most common method employed to boost the performance of meta-heuristic algorithms [17]. Many researchers have applied chaotic local search techniques to different optimization algorithms [18], [20]. Arora and Anand [18] introduced chaotic local search into the grasshopper optimization algorithm, which effectively balances development and exploration, and reduces the rejection or attraction between grasshoppers in the optimization process. Similarly, Kohli and Arora [19] added many chaotic maps to the grey wolf algorithm, adjusting key parameters to control the exploitation and exploration phases in the optimization process. Jordehi [20] combined with chaos theory and bat swarm optimization algorithm, using the ergodicity and non-repetition of chaotic functions diversified the bat population and mitigated the problem of premature convergence. Additionally, the theory of opposition-based learning has been confirmed by Tizhoosh [21] that the opposite number is closer than a random number to the optimal value and can enhance the search ability and accelerate convergence. This mathematical method was widely used by many researchers in different meta-heuristic algorithms [22], [25]. Kang *et al.* [22] introduced the opposition-based learning in his letter, aiming to solve the problem of premature convergence and low population diversity in traditional PSO. Zhang *et al.* [23] used the elite opposition-based learning to improve the performance of original gray wolf optimizer, the experiment results show the efficiency of their proposed algorithm compared with the original GWO and other meta-heuristic algorithms in terms of convergence rate and search ability.

The major contributions of this work are as follows.

1) A hybridization approach based on SSA, chaotic local search, and opposition-based learning is proposed.
2) Ten widely used chaotic maps are integrated into SSA, and their performance is compared with the original SSA.
3) The performance of the best opposition-based learning chaotic SSA is compared with various metaheuristics that have shown excellent performance on various benchmark functions.

The rest of the paper is arranged as follows: Section 2 presents the original SSA algorithm, mainly introducing its mathematical model. Section 3 describes the detail of proposed algorithm OCSSA. Also, the concepts of opposition-based learning and chaotic local search are illustrated in this section. Section 4 discusses the experimental results on global benchmark problems. Finally, a brief conclusion and recommendations for future work are offered in Section 5.

II. AN OVERVIEW OF SSA

SSA is one of several random population-based algorithms proposed in 2017, based on the population mechanism of a salp swarm foraging in the ocean. In the deep sea, a salp swarm usually forms a long chain. At the front of chain is the leader, whereas the rest of salps are considered as followers.

A. MATHEMATICAL MODEL OF SSA

The optimization process of SSA is determined by the three steps of population initialization, leader position updating, and follower position updating, which basically mimic the real clustering behavior of a salp swarm in the ocean. The operation of SSA is discussed in the next three sections.

1) POPULATION INITIALIZATION

Let the predation space be an $N \times D$ dimensional Euclidean space, where N is the scale of the salp swarm, and D is the spatial dimension. There is a food $F = [F_1, F_2, \cdots, F_D]^T$ in space, and the salp position can be expressed as $x_n = [x_{n,1}, x_{n,2}, \cdots, x_{n,D}]^T, n = 1, 2, \cdots, N$. The upper bound of the search space is expressed as $ub = [ub_1, ub_2, \cdots, ub_D]^T$, and the lower bound is $lb = [lb_1, lb_2, \cdots, lb_D]^T$. We randomly initialize the population:

$$X_{N\times D} = rand\,(N, D) \times (ub - lb) + lb \times ones\,(N, D) \quad (1)$$

In the population, the states of leaders and followers in $d - th$ dimension are $x_{1,d}$ and $x_{m,d}$, respectively, where $m = 2, 3, \cdots, N$.

2) LEADER POSITION UPDATE

The leader of a salp swarm is responsible for searching for food in the environment and guiding the movement of the whole group. Its position is updated randomly by

$$x_{1,d} = \begin{cases} F_d + c_1\left((ub_d - lb_d)\,c_2 + lb_d\right), & c_3 \geq 0.5 \\ F_d - c_1\left((ub_d - lb_d)\,c_2 + lb_d\right), & c_3 < 0.5, \end{cases} \quad (2)$$

where c_2 and c_3 are random numbers in the interval [0, 1]. The parameters enhance the randomness of leader movement, the global search ability, and the individual diversity, and c_1

is the main parameter in (2) which exists in all meta-heuristic algorithms and often called the convergence factor. It balances the exploration and exploitation ability of the algorithm in the iterative process. When the convergence factor is greater than 1, the algorithm performs global exploration. In contrast, when the factor is less than 1, the algorithm starts to develop the local part and obtains the accurate estimation value. To make the algorithm search globally in the first half of the iteration and develop accurately in the second half of the iteration, the value of the convergence factor is usually a decreasing number from 2 to 0. The expression of the convergence factor c_1 in SSA is

$$c_1 = 2e^{-\left(\frac{4l}{l_{\max}}\right)^2}, \tag{3}$$

where l is the current iteration number, and $l_{\max}$ is the maximum number of iterations.

3) FOLLOWER POSITION UPDATE

In SSA, there is no random movement of followers; they follow in a chain sequence. Therefore, the position of followers is only related to their initial position, motion speed, and acceleration. The motion mode conforms to Newton's law of motion, so the motion distance R of followers can be expressed as

$$R = \frac{1}{2}at^2 + v_0 t. \tag{4}$$

Because the time t is called iteration in optimization process, the discrepancy between iterations is equal to 1, i.e. $t = 1$. v_0 is the follower speed, which is 0 at the beginning of each iteration; and a is the acceleration of the followers between the beginning and end of an iteration, i.e. $a = \left(v_{final} - v_0\right)/t$, since a follower only follows the salp immediately in front of it. So, the movement speed $v_{final} = \left(x^l_{m-1,d} - x^l_{m,d}\right)/t$, where $t = 1$, $v_0 = 0$, hence

$$R = \frac{1}{2}\left(x^l_{m-1,d} - x^l_{m,d}\right), \tag{5}$$

and the update equation of follower location is

$$x^{l+1}_{m,d} = x^l_{m,d} + R = \frac{1}{2}\left(x^l_{m,d} + x^l_{m-1,d}\right), \tag{6}$$

where $x^l_{m,d}$ is the d dimensional position of the $m-th$ follower in the $l-th$ iteration, and $x^{l+1}_{m,d}$ is the follower location in the $(l+1)-th$ iteration. Algorithm 1 provides the pseudo-code of the standard SSA.

III. THE PROPOSED OCSSA

This section introduces opposition-based learning (OBL) and chaotic local search (CLS), which are used to enhance SSA algorithm performance, and then the improved OCSSA is described.

Algorithm 1 Salp Swarm Algorithm

```
1. Initialize the randomly generated population of the salp
   swarm $X_i$ $(i = 1, 2, \ldots, n)$.
2. Calculate the fitness value of each salp.
3.    $X^*$ = the best search agent.
4. while stopping criteria not reached
5.       Update $c_1$ by (3)
6.       for each salp
7.           if($n == 1$)
8.               Update the position of the leader salp by (2)
9.           else
10.              Update the position of the follower salp
                 by (6)
11.          end if
12.  end for
13.  Compute the fitness value of every salp.
14.  Update $X^*$ if there is a better solution.
15. end while
16. return the best solution $X^*$ and its fitness value.
```

A. OPPOSITION-BASED LEARNING

Traditional meta-heuristic algorithms start the search process by using a set of randomly generated numbers as the initial solution. The convergence rate of the algorithm is not stable and will be slow in most cases. To avoid these problems, opposition-based learning is introduced, and both randomly generated and reverse solutions are considered. The OBL properties are defined as follows. Let $P = (y_1, y_2, \cdots, y_D)$ be a point in D space, where $y_1, y_2, \cdots, y_D \in R$, $y_i \in [a_i, b_i]$, $\forall i \in \{1, 2, \cdots, D\}$. Then the opposite point of P is $OP = (oy_1, oy_2, \cdots, oy_d)$, where $oy_i = a_i + b_i - y_i$.

B. CHAOTIC LOCAL SEARCH

To improve the performance of SSA in obtaining the global optimal solution, a chaotic local search method based on search strategy is introduced, which accelerates the search process and forces it to advance to a region where the optimal solution is more likely to be obtained, enhancing the ability of algorithm exploitation [26]. CLS ends when a better solution or local search termination condition is reached.

Chaos is a common phenomenon in nonlinear systems in nature, and its ergodic property, namely traversing all states within a certain range without repetition, is frequently used as an optimization mechanism to escape from local optimum. In this paper, 10 chaotic maps, as shown in Table 1, are used to generate corresponding chaotic sets. The initial point of these chaotic mappings can be any number between 0 and 1. The initial value of the chaotic map used here is set to 0.7. We adopted the initial values as in [27].

C. PROPOSED OCSSA

OCSSA adds chaotic local search and opposition-based learning to solve the problem of local optimal solution and low convergence speed. Because chaotic local search has the characteristics of not repeatedly traversing the search

TABLE 1. Details of chaotic maps applied on OCSSA.

No.	Map Name	Map Equation
1	Logistic map	$c_{k+1} = 4c_k(1-c_k)$
2	Cubic map	$c_{k+1} = 2.59c_k(1-c_k^2)$
3	Sine map	$c_{k+1} = \sin(\pi c_k)$
4	Sinusoidal map	$c_{k+1} = 2.3c_k^2 \sin(\pi c_k)$
5	Singer map	$c_{k+1} = 1.073(7.86c_k - 23.31c_k^2 + 28.75c_k^3 - 13.302875c_k^4)$
6	Tent map	$c_{k+1} = \begin{cases} c_k/0.4, & 0 < c_k \le 0.4 \\ (1-c_k)/0.6, & 0.4 < c_k \le 1 \end{cases}$
7	Gaussian map	$c_{k+1} = \begin{cases} 0, & c_k = 0 \\ (1/c_k)\bmod(1), & c_k \ne 0 \end{cases}$
8	Chebyshev map	$c_{k+1} = \cos(0.5\cos^{-1} c_k)$
9	Bernoulli map	$c_{k+1} = \begin{cases} c_k/0.6, & 0 < c_k \le 0.6 \\ (c_k-0.6)/0.4, & 0.6 < c_k \le 1 \end{cases}$
10	Circle map	$c_{k+1} = c_k + 0.5 - \frac{1.1}{\pi}\sin(2\pi c_k)\bmod(1)$

area and opposition-based learning can bring the algorithm closer to the global optimal solution, the performance and convergence speed of OCSSA will be improved into various degrees compared to SSA. The next two sections describe the improvement scheme of parameter initialization and a leader position update stage.

1) IMPROVED POPULATION INITIALIZATION

SSA uses the random generation of salp population position for population initialization; hence, its performance is unstable. If the generated initial population position is close to that of the global optimal solution, then the convergence speed of the algorithm and its ability to obtain the global optimal solution will be good. But randomly generated initial solution positions are rarely ideal. Therefore, the OCSSA algorithm adds chaotic local search, which can provide a more reliable initial population position when the population of salps is initialized, so as to ensure that the convergence speed of the algorithm will not fluctuate greatly, and to improve the performance of the algorithm to a certain extent. The salp population X is built by the chaotic local search using the following equation:

$$x_{ij} = l_{ij} + ch_{ij} * \left(u_{ij} - l_{ij}\right) \tag{7}$$

where $x_i \in X$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, d$, l_i and u_i represent the lower boundary and the upper boundary of the salp $x_i \in X$,respectively. The ch_{ij} is the chaotic map value constructed using the equation listed in Table 1. In addition, adding opposition-based learning in the population initialization can make the function of chaos more powerful. By comparing the fitness value of the initial population position before and after the change of chaotic local search, the better individual position can be selected to improve the performance of the algorithm in terms of convergence speed.

Algorithm 2 OCSSA

```
1. Initialize the randomly generated population of the salp swarm $X_{ini}$ $(ini = 1, 2, \ldots, n)$.
2. Using chaotic maps to form new population of the salp swarm $X_i$ $(i = 1, 2, \ldots, n)$.
3. Calculate opposite point $X_{oi}$ of $X_i$.
4. Calculate the fitness value $f(X_i), f(X_{oi})$ of $X_i$ and $X_{oi}$.
5. if $f(X_{oi}) \le f(X_i)$
6. $X_i = X_{oi}$
7. end if
8. Calculate the fitness value of $X_i$.
9. $X^* =$ the best search agent.
10. while stopping criteria not reached
11.     Update $c_1$ by (3)
12.     for each salp
13.       if($n == 1$)
14.     Update the position of the leader salp by (2)
15.  else
16.     Update the position of the follower salp by (6)
17. end if
18. end for
19. Compute the fitness value of every salp $X_i$ $(i = 1, 2, \ldots, n)$.
20. Update $X^*$ if there is a better solution.
21. Using chaotic maps and opposition-based learning to form another new solution $X^*_{new}$.
22. Calculate the fitness value $f(X^*), f(X^*_{new})$ of $X^*$ and $X^*_{new}$.
23. if$f(X^*_{new}) \le f(X^*)$
24. $X^* = X^*_{new}$.
25. end if
26. end while
27. Return the best solution $X^*$ and its fitness value $f(X^*)$.
```

2) IMPROVED LEADER POSITION UPDATE

Equation 2 shows that the position of leaders in the population changes according to the position of food, while the position of followers is adjusted according to the position of leaders. So, the position of the leader is critical to the algorithm. If the algorithm has ideal exploration ability, the leader's position is equal to the food position, i.e., the algorithm achieves the global optimal solution. Food processing also differs significantly between OCSSA and SSA.

To optimize the position of leaders, OCSSA introduces chaotic local search and opposition-based learning. If the leader position remains the same or changes only slightly, the algorithm will fall into a local optimal solution. OCSSA uses chaotic local search to escape this issue. At this stage we set a threshold to control the number of CLS, and use the equation 7 to change the position of food. When a better position is found or the search upper limit is reached, the algorithm ends the chaotic search and performs the OBL phase. X^*_{new} updates its position according to the description in this paragraph. If the algorithm is always in a

TABLE 2. The definition of CEC2005 benchmark.

Function	Range	GM
Unimodal benchmark functions		
$F_1(x)=\sum_{i=1}^{n} x_i^2$	[-100,100]	0
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[-10,10]	0
$F_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
$F_4(x)=\max\left(\lvert x_i\rvert, 1\le i\le n\right)$	[-100,100]	0
$F_5(x)=\sum_{i=1}^{n-1}[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2]$	[-30,30]	0
$F_6(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	[-100,100]	0
$F_7(x)=\sum_{i=1}^{n} i x_i^4+random[0,1)$	[-1.28,1.28]	0
Multimodal benchmark functions		
$F_8(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500,500]	-8379.658
$F_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	[-5.12,5.12]	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	[-32,32]	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600,600]	0
$F_{12}(x)=\frac{\pi}{n}\left\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{n-1}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+\left(y_n-1\right)^2\right\}+\sum_{i=1}^{n}u\left(x_i,10,100,4\right)$ $y_i=1+\frac{x_i+1}{4}$	[-50,50]	0
$u\left(x_i,a,k,m\right)=\begin{cases}k\left(x_i-a\right)^m, & x_i>a\\ 0, & -a<x_i<a\\ k\left(-x_i-a\right)^m, & x_i<-a\end{cases}$		0
$F_{13}(x)=0.1\left\{\sin^2\left(3\pi x_1\right)+\sum_{i=1}^{n}\left(x_i-1\right)^2\left[1+\sin^2\left(3\pi x_i+1\right)\right]+\left(x_n-1\right)^2\left[1+\sin^2\left(2\pi x_n\right)\right]\right\}+\sum_{i=1}^{n}u\left(x_i,5,100,4\right)$	[-50,50]	0

nonconvergent state, it can more quickly reach the position near the global optimal solution, to improve the convergence speed and accuracy. Algorithm 2 is the pseudo-code of the improved salp swarm algorithm.

IV. EXPERIMENTAL RESULTS

A. EXPERIMENT PLATFORM

All the algorithms compared in this section are performed on MATLAB 2017 installed over Windows 10 (64bit) operating system that runs on a core i5 personal computer with 8GB RAM.

B. INTRODUCTION OF BENCHMARK FUNCTIONS

1) THE CLASSICAL CEC2005 BENCHMARK FUNCTIONS

We selected a set of 13 functions from the CEC2005 benchmark functions to compare the performance of the proposed algorithm with original SSA and other well-known algorithms. This benchmark functions can be divided into

TABLE 3. The definition of CEC2015 benchmark.

No.	Types	Name	Optimum
F1(CEC)	Unimodal functions	Rotated high conditioned elliptic function	100
F2(CEC)		Rotated bent cigar function	200
F3(CEC)	Simple multimodal functions	Shifted and rotated HGBat function	300
F4(CEC)		Shifted and rotated Weierstrass function	400
F5(CEC)		Shifted and rotated Ackleys function	500
F6(CEC)		Shifted and rotated HappyCat function	600
F7(CEC)		Shifted and rotated Griewanks function	700
F8(CEC)		Shifted and rotated Rastrigins function	800
F9(CEC)		Shifted and rotated Sphere function	900
F10(CEC)	Hybrid functions	Hybrid function 1 (N=3)	1000
F11(CEC)		Hybrid function 2 (N=4)	1100
F12(CEC)		Hybrid function 3 (N=5)	1200
F13(CEC)	Composition functions	Composition function 1 (N=3)	1300
F14(CEC)		Composition function 2 (N=3)	1400
F15(CEC)		Composition function 3 (N=5)	1500
Search range:	[-100,100]	Dimension: Dim=50	

two types; one is called the unimodal functions (F1-F7) which can be used to examine the optimization accuracy and convergence rate of an algorithm. These functions have only one extreme value in the search area. The other is multimode functions (F8-F13) and they are used to evaluate the ability of an algorithm to avoid local optimal solutions. They have more than one extreme value in the given area domain. The mathematical formulas and related properties of these functions are listed in Table 2. The dimension of each function is set to 20.

2) THE CEC2015 BENCHMARK FUNCTIONS

From the CEC2015 functions, we selected a set of 15 functions as a second benchmark to test the performance of algorithms which are used in this paper. They can be used to deal with the competition on single objective optimization problems. In addition to the 2005 test functions, these test functions include some new features, such as new basic problems, the shifted and rotated problems. A brief description of these benchmark problems is listed in Table 3.

C. PARAMETER SETTING

All algorithms used in the paper have a population of 30 and the maximum number of iterations is 500. For the statistical analysis, each benchmark function is carried out for 50 independent runs to minimize the statistical error of the results.

The relevant parameters in WOA, GWO and ABC algorithms adopt the values set in the original algorithm. The mutation probability in DE is 0.5 and the weight factor value is 0.9. The learning factor value of PSO is 2, the inertia factor is 0.6, and the maximum velocity of particle is 10, which is as same as OCSSA's step distance.

D. COMPARISON OF OCSSA AND ORIGINAL SSA

In this part, we compare the SSA and the proposed OCSSA algorithm on 13 classical benchmark functions in terms of numerical characteristics (mean, standard deviation, and statistical best), algorithm diversity and algorithm computational complexity and runtime. The specific information is described in the following sections.

1) NUMERICAL CHARACTERISTICS

A set of 13 functions from CEC2005 benchmark is used in this section to evaluate the performance of original SSA and the proposed algorithm OCSSA. OCSSA1, OCSS2, . . ., OCSSA10 correspond to the 10 chaotic maps listed in Table 1. The comparison results for original SSA and ten versions of OCCSA using the unimodal and the multimodal functions are shown in Tables 4-5.

Table 4 displays the results of the original SSA and different chaotic versions of OCSSA on the unimodal functions. The simulation results in this table show that OCSSA4 has the best mean value and statistical best among 6

TABLE 4. Results of unimodal benchmark functions.

F1	Best	Mean	Std	**F2**	Best	Mean	Std
SSA	2.95E-09	6.63E-09	1.83E-09	SSA	2.91E-04	1.87E-01	2.57E-01
OCSSA1	4.97E-47	6.10E-42	2.80E-41	OCSSA1	9.27E-25	3.14E-09	1.75E-08
OCSSA2	1.12E-52	1.20E-38	6.67E-38	OCSSA2	2.08E-27	3.71E-20	2.06E-19
OCSSA3	2.56E-49	1.22E-41	3.79E-41	OCSSA3	2.16E-25	2.43E-17	1.35E-16
OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	OCSSA4	**2.01E-230**	**1.57E-179**	**0.00E+00**
OCSSA5	3.09E-116	8.12E-28	4.51E-27	OCSSA5	9.18E-55	1.64E-05	9.09E-05
OCSSA6	1.42E-110	6.19E-102	2.12E-101	OCSSA6	1.45E-57	1.54E-31	8.57E-31
OCSSA7	2.95E-108	4.48E-94	2.40E-93	OCSSA7	9.05E-58	1.18E-51	5.29E-51
OCSSA8	0.00E+00	4.29E-34	2.31E-33	OCSSA8	3.96E-197	3.18E-06	1.77E-05
OCSSA9	3.57E-110	6.08E-98	1.83E-97	OCSSA9	2.80E-55	1.73E-23	9.64E-23
OCSSA10	1.05E-10	1.24E-09	1.39E-09	OCSSA10	6.61E-06	1.27E-04	4.28E-04
F3	Best	Mean	Std	**F4**	Best	Mean	Std
SSA	1.02E+01	1.09E+02	1.06E+02	SSA	1.78E-01	2.80E+00	3.51E+00
OCSSA1	2.74E-11	5.39E-02	1.55E-01	OCSSA1	6.69E-13	3.42E-12	2.80E-12
OCSSA2	8.60E-10	1.13E-02	2.48E-02	OCSSA2	2.50E-20	1.82E-18	1.97E-18
OCSSA3	2.03E-20	2.59E-02	7.42E-02	OCSSA3	1.79E-13	1.42E-12	1.96E-12
OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	OCSSA4	**8.75E-235**	**4.21E-208**	**0.00E+00**
OCSSA5	1.25E-09	9.04E-01	2.56E+00	OCSSA5	1.20E-63	9.58E-47	2.91E-46
OCSSA6	2.92E-52	6.97E-10	1.90E-09	OCSSA6	1.29E-47	4.89E-44	1.40E-43
OCSSA7	7.86E-71	2.20E-10	5.68E-10	OCSSA7	9.40E-42	1.45E-37	4.58E-37
OCSSA8	4.49E-241	6.23E-03	1.73E-02	OCSSA8	1.41E-233	3.19E-198	0.00E+00
OCSSA9	1.77E-145	3.49E-85	1.00E-84	OCSSA9	3.31E-39	9.29E-38	1.09E-37
OCSSA10	1.07E-09	6.21E+00	1.70E+01	OCSSA10	5.26E-06	8.12E-06	1.75E-06
F5	Best	Mean	Std	**F6**	Best	Mean	Std
SSA	1.73E+01	5.55E+01	1.70E+03	SSA	3.85E-09	8.30E-09	3.02E-09
OCSSA1	1.86E+01	1.86E+01	3.04E-02	OCSSA1	6.21E-10	2.97E-09	1.44E-09
OCSSA2	1.85E+01	1.86E+01	3.71E-02	OCSSA2	1.32E-09	4.43E-09	1.63E-09
OCSSA3	1.86E+01	1.86E+01	**2.45E-02**	OCSSA3	3.71E-10	2.20E-09	1.06E-09
OCSSA4	**9.41E-09**	**1.52E+01**	6.95E+00	OCSSA4	**3.26E-14**	**1.04E-10**	**1.81E-10**
OCSSA5	2.28E-07	1.57E+01	6.34E+00	OCSSA5	1.35E-09	5.24E-09	1.75E-09
OCSSA6	1.86E+01	1.86E+01	3.11E-02	OCSSA6	3.75E-10	2.24E-09	1.24E-09
OCSSA7	1.86E+01	1.86E+01	3.19E-02	OCSSA7	1.12E-11	1.76E-09	1.56E-09
OCSSA8	1.85E-07	1.75E+01	3.34E+00	OCSSA8	8.43E-11	4.05E-09	1.87E-09
OCSSA9	1.86E+01	1.87E+01	2.64E-02	OCSSA9	8.50E-12	1.20E-09	9.70E-10
OCSSA10	1.49E+01	2.95E+01	2.68E+01	OCSSA10	3.82E-09	7.92E-09	2.39E-09
F7	Best	Mean	Std				
SSA	2.53E-02	6.45E-02	2.56E-02				
OCSSA1	2.04E-06	6.28E-04	7.36E-04				
OCSSA2	7.00E-06	4.34E-04	6.05E-04				
OCSSA3	8.96E-06	3.96E-04	5.62E-04				
OCSSA4	7.20E-06	5.18E-04	6.90E-04				
OCSSA5	**1.46E-07**	**3.46E-05**	**2.23E-05**				
OCSSA6	1.61E-06	6.82E-04	6.87E-04				
OCSSA7	9.78E-06	5.75E-04	6.13E-04				
OCSSA8	1.13E-05	4.95E-04	7.89E-04				
OCSSA9	5.48E-06	2.43E-04	2.93E-04				
OCSSA10	1.91E-05	1.85E-02	1.94E-02				

TABLE 5. Results of multimodal benchmark functions.

F8	Best	Mean	Std	**F9**	Best	Mean	Std
SSA	-6360.384	-5100.721	537.607	SSA	5.97E+00	2.92E+01	8.73E+02
OCSSA1	-21777.578	-12304.429	3402.744	OCSSA1	0.00E+00	1.34E+00	2.85E+00
OCSSA2	-17624.573	-10567.805	3418.340	OCSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA3	-14301.474	-12155.986	2163.005	OCSSA3	0.00E+00	2.49E-01	6.34E-01
OCSSA4	-26072.400	-11567.622	5019.843	OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA5	-20970.766	-11363.730	4055.461	OCSSA5	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA6	-14301.449	-11523.506	2611.771	OCSSA6	0.00E+00	3.42E-01	8.78E-01
OCSSA7	**-8379.658**	**-8379.658**	**0.000**	OCSSA7	0.00E+00	4.97E-01	1.43E+00
OCSSA8	-20473.338	-11424.895	4154.301	OCSSA8	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA9	-14300.685	-10097.616	2540.583	OCSSA9	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA10	-16857.835	-13571.517	2249.651	OCSSA10	6.40E-11	8.05E+00	1.46E+01
F10	Best	Mean	Std	**F11**	Best	Mean	Std
SSA	1.28E-05	9.69E-01	2.91E+01	SSA	7.14E-08	2.29E-02	2.11E-02
OCSSA1	8.88E-16	6.44E-15	5.63E-15	OCSSA1	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA2	8.88E-16	7.44E-15	5.60E-15	OCSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA3	8.88E-16	4.55E-15	2.59E-15	OCSSA3	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA4	**8.88E-16**	**1.11E-15**	**6.38E-16**	OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA5	8.88E-16	2.81E-08	7.98E-08	OCSSA5	**0.00E+00**	**1.08E-09**	**2.84E-09**
OCSSA6	8.88E-16	4.33E-15	2.15E-15	OCSSA6	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA7	8.88E-16	2.89E-15	1.76E-15	OCSSA7	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA8	8.88E-16	1.55E-15	1.32E-15	OCSSA8	0.00E+00	6.73E-18	1.99E-17
OCSSA9	8.88E-16	1.44E-15	1.21E-15	OCSSA9	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA10	4.71E-06	1.01E-05	4.55E-06	OCSSA10	3.63E-10	1.87E-03	4.39E-03
F12	Best	Mean	Std	**F13**	Best	Mean	Std
SSA	4.64E-01	3.56E+00	1.87E+00	SSA	1.04E-07	1.05E-01	4.11E-01
OCSSA1	3.06E-11	1.14E-01	1.66E-01	OCSSA1	1.23E-10	1.28E-09	1.50E-09
OCSSA2	4.50E-11	1.26E-01	1.90E-01	OCSSA2	1.36E-10	1.71E-08	4.73E-08
OCSSA3	1.31E-11	5.90E-02	8.47E-02	OCSSA3	1.15E-11	3.18E-10	2.18E-10
OCSSA4	1.34E-14	**1.55E-12**	**1.56E-12**	OCSSA4	**6.03E-17**	3.36E-02	1.60E-01
OCSSA5	8.50E-11	7.26E-02	1.30E-01	OCSSA5	1.67E-10	1.32E-03	7.10E-03
OCSSA6	4.65E-12	1.71E-02	4.15E-02	OCSSA6	1.43E-13	2.48E-10	2.58E-10
OCSSA7	2.04E-13	3.11E-11	3.76E-11	OCSSA7	5.89E-11	3.20E-01	6.54E-01
OCSSA8	4.60E-11	3.98E-04	9.85E-04	OCSSA8	1.51E-11	1.51E-08	3.81E-08
OCSSA9	**3.48E-15**	3.71E-12	3.88E-12	OCSSA9	3.89E-14	**3.00E-11**	**2.53E-11**
OCSSA10	1.00E-09	6.93E-01	1.14E+00	OCSSA10	6.51E-09	4.78E-03	7.65E-03

functions (F1-F6), OCSSA5 has one best mean value on F7. From the perspective of standard deviation, OCSSA4 achieved the 5 best results, and OCSSA3 and OCSSA5 each has an optimal standard deviation on 7 test functions. It is worth mentioning that OCSSA4 has achieved theoretical optimal values on the F1 and F3 test function.

Table 5 shows the performance of the original SSA and the proposed algorithm on multimodal functions. In terms of mean value, OCSSA4 performed the best, achieving 4 statistical bests on 6 test functions (F9-F12), OCSSA9 ranked second, and achieved 3 optimal values (F9, F11, F13), OCSSA2 (F9, F11) and OCSSA7 (F8, F11) both have two optimal mean value, while OCSSA5 and OCSSA8 both obtain the best average value on F8. From the perspective of standard deviation and statistical optimality, OCSSA4 is still the best performing improved algorithm.

It can be seen from the results of Table 4 and Table 5 that OCSSA4 which combined with the sinusoidal chaotic map performs best among many different improved versions of OCSSA algorithm. The improved algorithm has better optimization effect than the original SSA algorithm in both unimodal and multimodal functions.

2) DIVERSITY OF THE ALGORITHMS

In order to evaluate the effect of chaotic local search and opposition-based learning on the exploration and exploitation

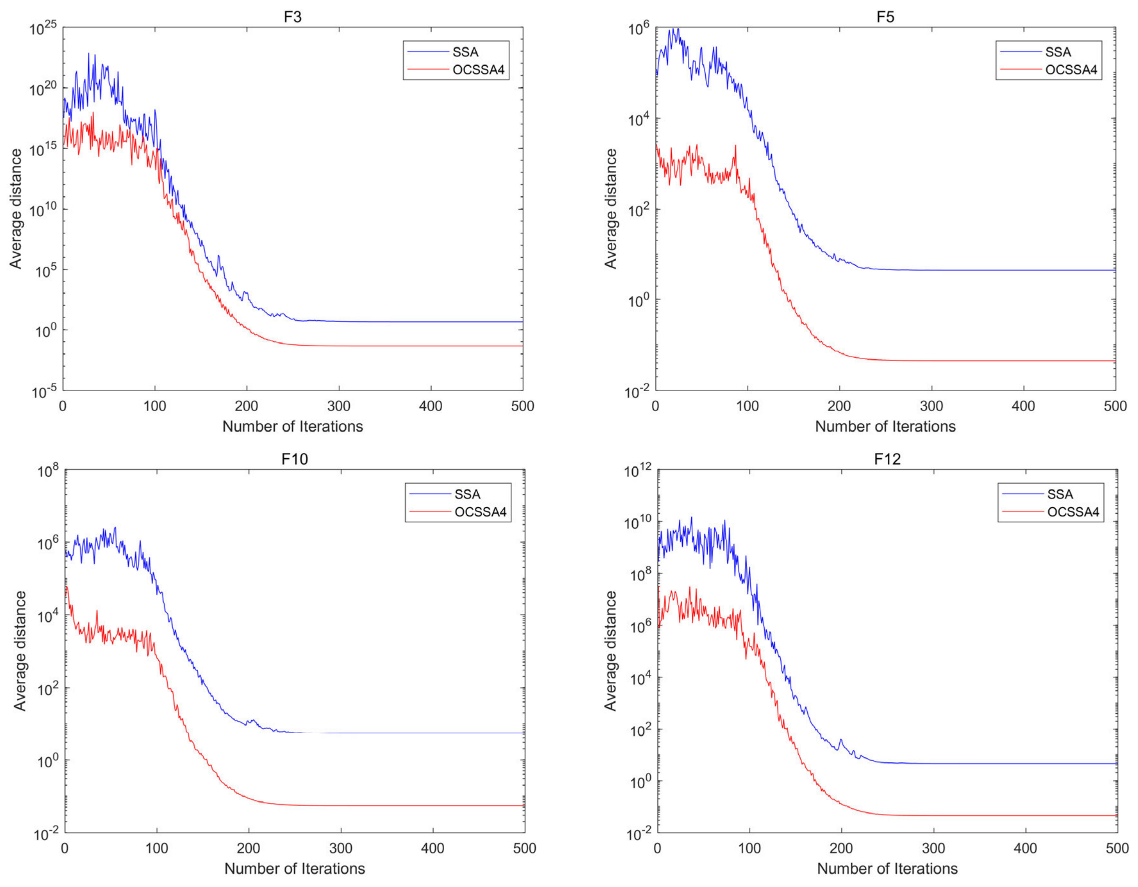

FIGURE 1. The diversity of the proposed algorithm and SSA on F3, F5, F10, F12 benchmark functions (CEC2005).

TABLE 6. Results of algorithm runtime.

		SSA	OCSSA4			SSA	OCSSA4
F1	Time	0.404	0.848	F8	Time	0.432	0.810
	SR	0	100		SR	6.67	100
F2	Time	0.446	0.794	F9	Time	0.438	0.934
	SR	0	100		SR	0	100
F3	Time	0.844	1.429	F10	Time	0.505	1.328
	SR	0	100		SR	0	96.67
F4	Time	0.400	0.983	F11	Time	0.489	1.056
	SR	0	100		SR	0	100
F5	Time	0.447	0.960	F12	Time	0.795	1.358
	SR	0	18		SR	0	60
F6	Time	0.337	0.861	F13	Time	0.782	1.312
	SR	20	100		SR	26.67	96.67
F7	Time	0.514	1.132				
	SR	0	100				

of OCSSA, diversity plots are presented in Figs.1. The diversity plots represent the average distance between each search agent in the optimization process. large average distance between search agents indicates high population diversity and vice-versa [28]. As can be analyzed in Figs.1, OCSSA4 can keep high population diversity in the initial phases of optimization process. this allows OCSSA4 to avoid the local optimal solution phenomenon, and make the algorithm converge to the global optimal solution direction to obtain a more accurate solution. At the same time, OCSSA4 has a

TABLE 7. Simulation results of best OCSSA versus other metaheuristics on CEC 2005 benchmark functions.

Functions		DE	WOA	PSO	ABC	GWO	OCSSA4
F1	Best	5.01E-04	2.82E-86	2.13E-06	4.37E-08	3.66E-37	**0.00E+00**
	Mean	8.58E-03	5.37E-72	1.56E-05	1.41E-07	6.22E-36	**0.00E+00**
	Std	1.59E-02	1.77E-71	1.14E-05	9.16E-08	1.01E-35	**0.00E+00**
F2	Best	1.12E-02	2.38E-57	3.76E-04	2.07E-05	9.47E-23	**1.85E-237**
	Mean	3.53E-02	3.08E-51	2.05E-03	6.73E-05	1.10E-21	**3.27E-199**
	Std	4.66E-02	8.66E-51	1.92E-03	3.99E-05	8.46E-22	**0.00E+00**
F3	Best	2.89E+01	1.22E+03	1.61E+00	4.65E+03	3.91E-14	**0.00E+00**
	Mean	1.03E+02	7.70E+03	4.54E+00	7.91E+03	9.33E-11	**0.00E+00**
	Std	6.32E+01	3.73E+03	2.76E+00	1.72E+03	1.49E-10	**0.00E+00**
F4	Best	2.68E+00	1.43E-02	2.08E-01	1.63E+01	8.61E-11	**3.45E-235**
	Mean	5.44E+00	1.35E+01	3.37E-01	2.37E+01	7.26E-10	**4.39E-210**
	Std	1.94E+00	1.88E+01	1.21E-01	6.16E+00	8.66E-10	**0.00E+00**
F5	Best	1.50E+01	1.69E+01	1.12E+01	9.97E+00	1.60E+01	**1.16E-07**
	Mean	4.75E+01	1.75E+01	3.72E+01	2.13E+01	1.68E+01	**1.19E+01**
	Std	4.07E+01	**4.79E-01**	2.94E+01	9.20E+00	7.47E-01	8.55E+00
F6	Best	1.17E-03	1.20E-02	3.61E-06	4.67E-08	1.74E-05	**3.46E-13**
	Mean	1.32E-02	1.73E-01	2.71E-05	9.55E-08	2.05E-01	**8.08E-11**
	Std	2.97E-02	1.47E-01	2.75E-05	8.44E-08	1.75E-01	**1.76E-10**
F7	Best	2.40E-02	1.39E-04	2.31E-02	1.03E-01	1.22E-04	**5.69E-05**
	Mean	3.41E-02	3.56E-03	4.37E-02	1.43E-01	1.03E-03	**1.62E-04**
	Std	9.39E-03	3.66E-03	1.60E-02	3.71E-02	5.80E-04	**1.60E-04**
F8	Best	-4331.929	**-8379.270**	-4350.107	-6.98E+07	-4784.750	-22853.089
	Mean	-3771.686	**-6899.816**	-3135.995	-1.90E+07	-4034.524	-11365.189
	Std	**327.686**	1201.246	689.384	1.84E+07	557.128	5801.900
F9	Best	1.08E+02	**0.00E+00**	1.20E+01	1.01E+00	**0.00E+00**	**0.00E+00**
	Mean	1.21E+02	5.68E-15	2.43E+01	2.17E+00	4.49E-01	**0.00E+00**
	Std	6.74E+00	1.71E-14	7.43E+00	1.03E+00	1.35E+00	**0.00E+00**
F10	Best	4.49E-01	**8.88E-16**	2.23E-02	5.99E-01	7.19E-14	**8.88E-16**
	Mean	1.29E+00	2.50E-15	8.18E-02	1.02E+00	8.42E-14	**1.53E-15**
	Std	8.94E-01	1.78E-15	5.13E-02	2.67E-01	1.34E-14	**1.42E-15**
F11	Best	1.98E-03	**0.00E+00**	4.19E-02	5.42E-04	**0.00E+00**	**0.00E+00**
	Mean	6.78E-02	2.00E-02	2.47E-01	6.54E-03	6.80E-03	**0.00E+00**
	Std	7.49E-02	4.39E-02	4.33E-01	7.77E-03	8.62E-03	**0.00E+00**
F12	Best	6.20E-05	1.90E-03	4.96E-08	3.55E-09	5.43E-06	**5.67E-15**
	Mean	9.14E-04	1.42E-02	4.24E-02	5.36E-08	2.21E-02	**8.68E-13**
	Std	1.16E-03	7.34E-03	7.13E-02	8.35E-08	1.21E-02	**1.04E-12**
F13	Best	5.44E-04	8.67E-02	1.70E-06	3.14E-08	2.82E-05	**1.66E-15**
	Mean	5.30E-03	2.43E-01	2.46E-03	**1.20E-05**	1.00E-01	1.98E-02
	Std	5.58E-03	1.69E-01	4.32E-03	**2.36E-05**	8.41E-02	5.93E-02

smaller population diversity during the exploration stage, which means that OCSSA4 can achieve higher accuracy than original SSA. this conclusion is confirmed in the previous section. it should be noted that during the exploration phase of the algorithm, the diversity of OCSSA is lower than the original SSA, which can be explained. because OCSSA introduces chaotic local search and opposition-based learning during the population initialization stage, which results in a population that tends to be more optimal, which will reduce the diversity of the population. although the convergence speed of the two is not much different from the Figs.1, from the perspective of convergence accuracy, the performance of the proposed algorithm is better, which means that the proposed algorithm has a stronger ability to find the global optimal solution.

3) COMPUTATIONAL COMPLEXITY AND RUNTIME

The computational complexity of an optimization algorithm is a key metric for evaluating the runtime of an algorithm. The computational complexity can be defined based on the structure of the algorithm. The computational

TABLE 8. Simulation results of best OCSSA versus other metaheuristics on CEC 2015 benchmark functions.

Functions		ABC	GWO	WOA	PSO	DE	OCSSA4
	Best	9.52E+08	5.78E+08	4.58E+09	7.34E+09	9.23E+09	**3.62E+08**
F1	Mean	1.92E+09	3.97E+09	1.09E+10	1.23E+10	1.48E+10	**6.03E+08**
	Std	6.23E+08	3.68E+09	4.31E+09	4.88E+09	3.87E+09	**1.85E+08**
	Best	8.13E+10	1.04E+11	2.06E+11	1.78E+11	2.67E+11	**3.69E+10**
F2	Mean	1.06E+11	1.66E+11	2.79E+11	2.53E+11	3.19E+11	**4.79E+10**
	Std	1.63E+10	2.07E+10	3.77E+10	3.79E+10	3.25E+10	**7.89E+09**
	Best	320.7851	321.2041	321.1957	321.0532	321.4291	**320.6450**
F3	Mean	320.8792	321.3010	321.3085	321.2574	321.5082	**320.8289**
	Std	0.0476	0.0423	0.0444	0.0965	**0.0400**	0.1036
	Best	1.10E+03	9.33E+02	1.26E+03	1.34E+03	1.61E+03	**8.77E+02**
F4	Mean	1.20E+03	1.18E+03	1.49E+03	1.45E+03	1.75E+03	**9.71E+02**
	Std	5.79E+01	9.15E+01	1.51E+02	8.27E+01	8.18E+01	**4.76E+01**
	Best	1.05E+04	1.15E+04	1.41E+04	1.44E+04	1.67E+04	**8.70E+03**
F5	Mean	1.09E+04	1.42E+04	1.56E+04	1.53E+04	1.76E+04	**1.01E+04**
	Std	**2.74E+02**	1.29E+03	8.20E+02	4.85E+02	5.22E+02	5.57E+02
	Best	7.69E+07	1.07E+07	8.74E+07	3.43E+08	3.35E+08	**2.89E+06**
F6	Mean	2.23E+08	1.04E+08	6.84E+08	8.66E+08	1.60E+09	**1.77E+07**
	Std	1.10E+08	7.01E+07	4.66E+08	4.54E+08	6.84E+08	**1.21E+07**
	Best	1.47E+03	9.11E+02	2.28E+03	3.18E+03	5.06E+03	**8.60E+02**
F7	Mean	2.10E+03	2.51E+03	6.25E+03	5.62E+03	7.88E+03	**9.42E+02**
	Std	5.61E+02	1.91E+03	2.25E+03	1.71E+03	2.98E+03	**4.93E+01**
	Best	3.96E+07	4.40E+06	1.21E+08	8.91E+07	2.84E+08	**1.49E+06**
F8	Mean	1.16E+08	3.96E+07	3.71E+08	3.04E+08	7.39E+08	**6.03E+06**
	Std	6.17E+07	4.42E+07	2.85E+08	1.98E+08	4.19E+08	**2.22E+06**
	Best	1.32E+03	1.19E+03	1.67E+03	1.98E+03	1.96E+03	**1.07E+03**
F9	Mean	1.49E+03	1.71E+03	2.51E+03	2.70E+03	2.72E+03	**1.10E+03**
	Std	7.43E+01	5.06E+02	4.50E+02	3.48E+02	2.91E+02	**1.86E+01**
	Best	9.48E+07	1.93E+07	6.43E+07	2.05E+08	3.03E+08	**2.23E+06**
F10	Mean	2.63E+08	1.73E+08	6.06E+08	5.75E+08	1.39E+09	**6.65E+06**
	Std	1.08E+08	1.26E+08	3.66E+08	3.59E+08	4.86E+08	**3.34E+06**
	Best	2.85E+03	**2.20E+03**	3.64E+03	3.91E+03	4.23E+03	2.82E+03
F11	Mean	3.20E+03	3.75E+03	3.82E+03	4.90E+03	5.55E+03	**3.18E+03**
	Std	1.34E+02	7.63E+02	**1.11E+02**	7.73E+02	1.13E+03	2.39E+02
	Best	1.39E+03	1.44E+03	1.43E+03	1.52E+03	1.55E+03	**1.34E+03**
F12	Mean	1.42E+03	1.57E+03	1.52E+03	1.67E+03	1.69E+03	**1.35E+03**
	Std	1.35E+01	7.15E+01	5.03E+01	5.51E+01	6.20E+01	**6.92E+00**
	Best	1.54E+03	1.62E+03	1.55E+03	2.08E+03	1.99E+03	**1.53E+03**
F13	Mean	**1.54E+03**	4.80E+03	1.57E+03	5.81E+03	5.72E+03	**1.54E+03**
	Std	**2.32E+00**	2.87E+03	3.06E+01	3.73E+03	2.66E+03	7.49E+00
	Best	1.29E+05	1.49E+05	1.75E+05	2.80E+05	4.28E+05	**6.92E+00**
F14	Mean	**1.47E+05**	4.24E+05	2.72E+05	7.04E+05	8.58E+05	8.16E+05
	Std	**1.03E+04**	2.97E+05	4.61E+04	2.97E+05	2.73E+05	1.83E+06
	Best	1.70E+03	7.90E+05	4.05E+04	2.91E+05	4.13E+04	**1.63E+03**
F15	Mean	1.93E+03	2.55E+06	6.04E+04	7.45E+05	6.59E+05	**1.65E+03**
	Std	1.51E+02	1.59E+06	1.27E+04	3.55E+05	6.41E+05	**1.17E+01**

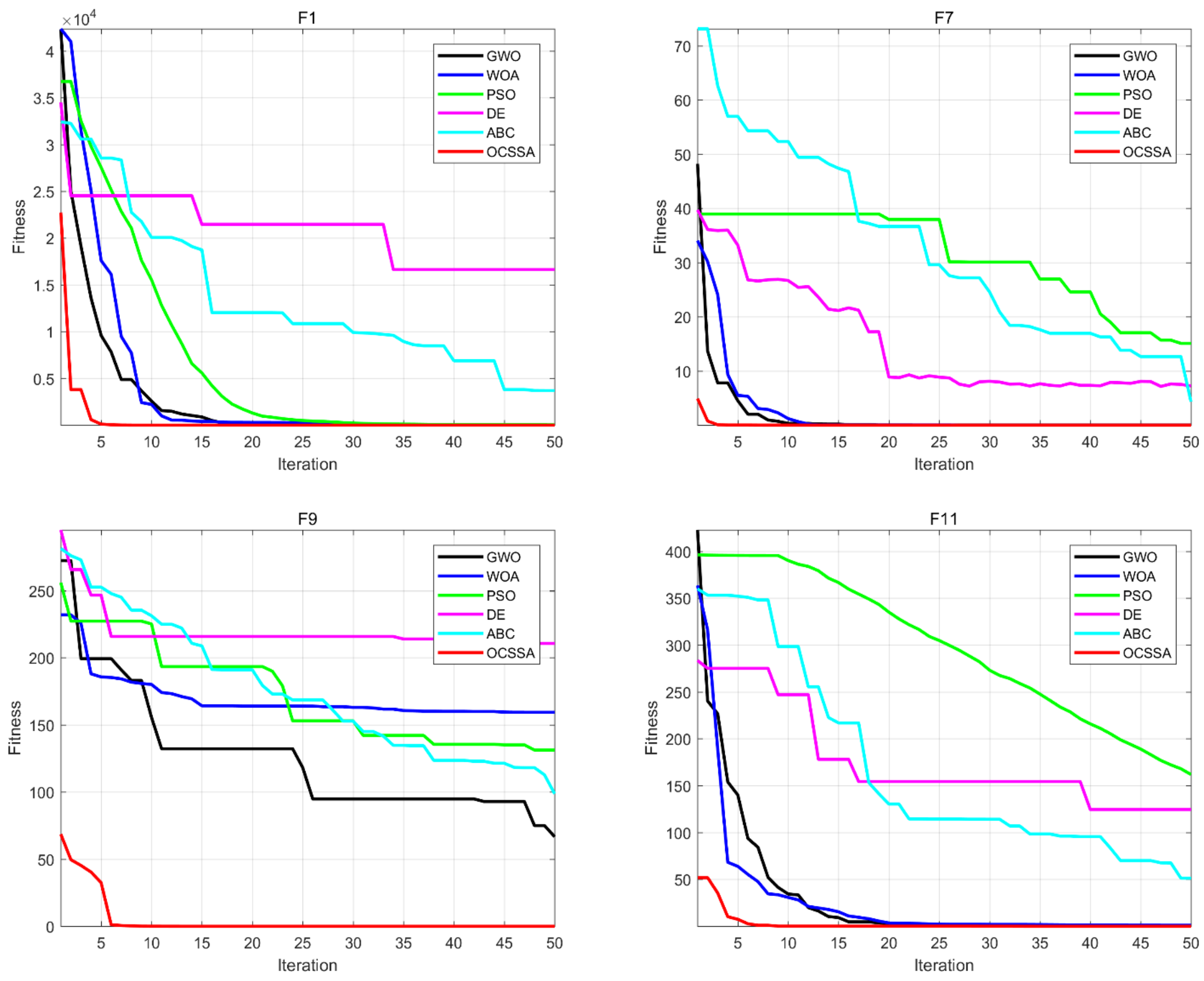

FIGURE 2. **The convergence curves of proposed algorithm on CEC2005 benchmark functions.**

complexity of SSA depends on the number of salps, dimension of the problem and maximum number of iterations. Overall, by analyzing the steps of algorithms, the computational complexity of the original SSA algorithm is $O\left(t\left(d*n+Cof*n\right)\right)$. OCSSA adds CLS and OBL during the population initialization phase and iterative optimization process, respectively. In the population initialization phase, the time complexity of OCSSA is $O\left(d*n+Cof*n\right)$, and the time complexity of the optimization iteration process is $O\left(t\left(d*\left(n+\max C\right)+Cof*n\right)\right)$. Combine the above two parts, the time complexity of OCSSA is $O\left(t\left(d*\left(n+\max C\right)+Cof*n\right)\right)$, where t is the number of iterations, d shows the number of dimension, n indicates the number of search agents, Cof is the cost of objective function, and $\max C$ is the maximum iteration number of CLS.

It can be concluded that the limitation of the proposed algorithm is still the computational complexity which needs to be reduced. The reason for this complexity results from two components 1) The OBL strategy 2) The CLS method, since both are applied to the whole population.

In addition, we analyzed the results of the two algorithms from the aspect of experimental execution time. The detailed results are shown in Table 6. In this table, the runtime of algorithms is the average of 50 experimental results, and SR represents the probability that algorithms can reach the ideal global optimal value. From algorithms runtime of the 13 test functions in the table, it can be seen that the execution time of OCSSA4 is longer than the original SSA algorithm. The main reason is that OCSSA algorithm adds two methods based on the original algorithm, so the long runtime is unquestionable. However, from the perspective of the optimal value obtained by the algorithm, the effect of OCSSA is much better than the original SSA algorithm. We sacrificed the execution time of the algorithm to improve the accuracy of its optimization.

E. COMPARISON OF OCSSA AND ORIGINAL SSA

In this part, we use the two types of benchmark functions (CEC2005 and CEC 2015 functions) to evaluate the performance of the proposed algorithm and compare it with several other algorithms which including ABC, DE, GWO, PSO, WOA. Here we choose the fourth of 10 different versions of OCSSA, that is, OCSSA4 as the algorithm of this paper compared with other algorithms. Meanwhile, in order to add further analysis to our results, a nonparametric Wilcoxon's

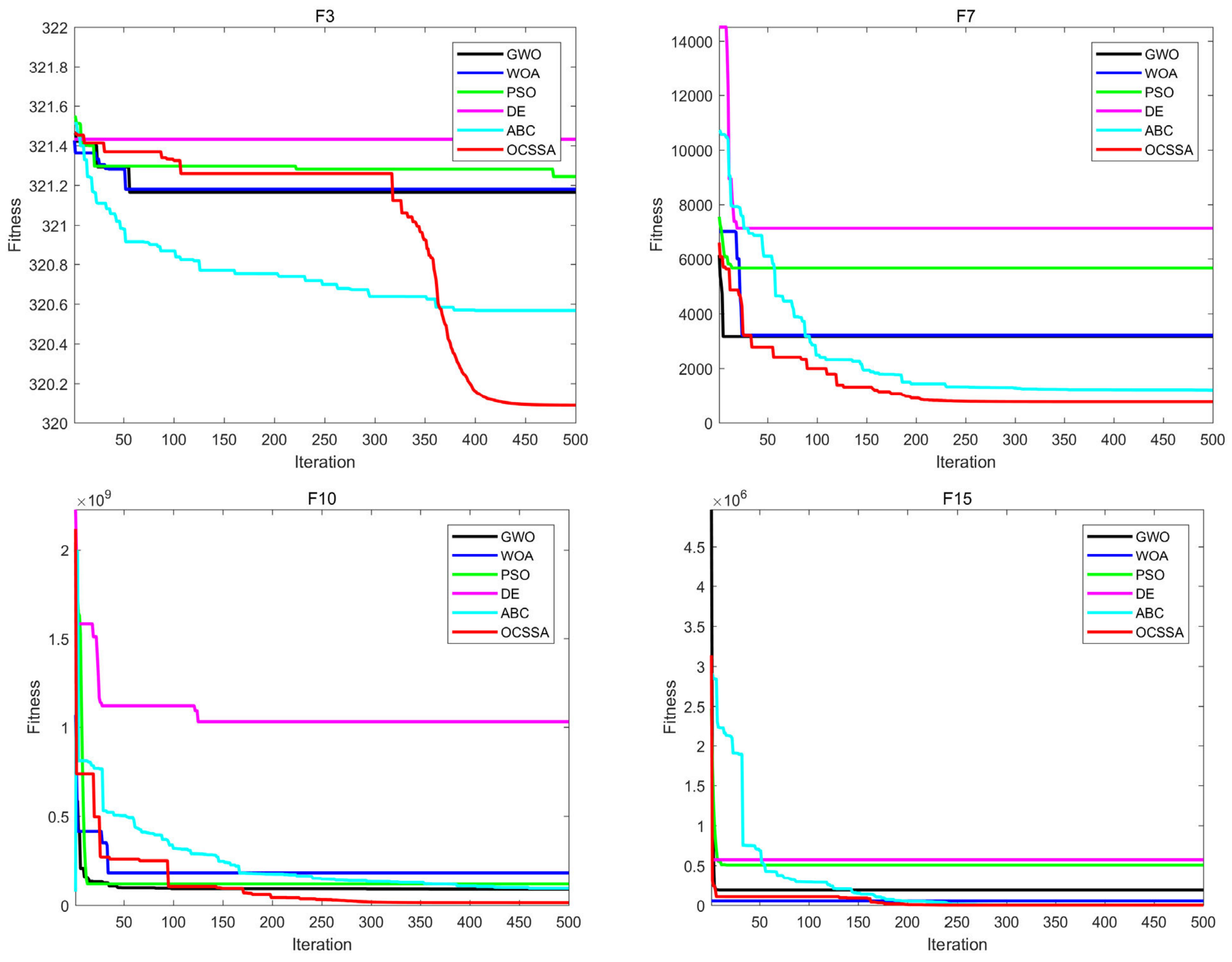

FIGURE 3. The convergence curves of proposed algorithm on CEC2015 benchmark functions.

rank sum (WRS) test is used to give a statistical value to determine if the two being compared are significantly different. Experimental data and discussion are explained in detail in the following sections.

1) NUMERICAL CHARACTERISTICS

The comparison results of OCSSA4 and other meta-heuristic algorithms on the two types of benchmark functions are shown in Tables 7-8, also, Figures 2-3 show the convergence curves for the algorithms.

Table 7 shows the results of the OCSSA4 and other optimization algorithms on the CEC2005 classical functions. The best performing values in each evaluation criterion are bold. The simulation results in Table 6 show that OCSSA4 has the best mean value among the 11 benchmark functions (F1, F2, . . ., F7, F9, . . ., F12), and the WOA and ABC algorithms rank second, respectively achieving the ideal mean value in F8 and F12. From the perspective of optimal value, OCSSA4 obtained the optimal value in 12 test functions (F1-F7, F9-F12), and WOA and GWO obtained the optimal value in four (F8-F11) and two (F9, F11) benchmark test functions, respectively. Comparing the results of the standard deviation, it can be found that OCCSSA4 still has a large advantage, while DE, ABC, and WOA each obtain the optimal standard deviation in one test function.

Table 8 displays the performance of the OCSSA4 and other optimization algorithms on the CEC2015 benchmark functions. From an average point of view, OCSSA4 has the best performance, achieving the best performance among 14 (F1-F13, F15) functions, and the second is the ABC algorithm, which obtains the best average value on the F14 function. From this table, we can see that OCSSA4 obtained the optimal standard deviation on 10 test functions (F1, F2, F4, F6-F10, F12, F15). The second one was ABC, which performed best on three test functions (F5, F13, F14). The two algorithms WOA and DE were on F11 and F3, respectively get the best standard deviation. In terms of the statistically optimal values that the algorithm can obtain, OCSSA4 performs best on the majority of test functions compared to several other algorithms. The experimental results also confirm the NFL (No Free Lunch) theorem [29] from the side, i.e., an algorithm cannot perform optimally on all optimization problems.

To clearly observe and analyze the convergence curves of OCSSA and other algorithms, these algorithms were run 50 times independently, and the convergence performance of

TABLE 9. The WRS test for CEC2005 benchmark functions.

	Statistics	DE	WOA	PSO	ABC	GWO
F1	p-value	1.57E-04	1.57E-04	1.57E-04	1.57E-04	1.57E-04
	H	1	1	1	1	1
F2	p-value	1.57E-04	1.57E-04	1.57E-04	1.57E-04	1.57E-04
	H	1	1	1	1	1
F3	p-value	1.57E-04	1.57E-04	1.57E-04	1.57E-04	1.57E-04
	H	1	1	1	1	1
F4	p-value	1.57E-04	1.57E-04	1.57E-04	1.57E-04	1.57E-04
	H	1	1	1	1	1
F5	p-value	9.63E-02	1.62E-01	9.63E-02	9.63E-02	1.62E-01
	H	0	0	0	0	0
F6	p-value	1.57E-04	1.57E-04	1.57E-04	1.57E-04	1.57E-04
	H	1	1	1	1	1
F7	p-value	1.57E-04	2.85E-04	1.57E-04	1.57E-04	2.85E-04
	H	1	1	1	1	1
F8	p-value	3.81E-04	2.50E-03	2.12E-04	1.57E-04	6.70E-04
	H	1	1	1	1	1
F9	p-value	1.57E-04	7.05E-01	1.57E-04	1.57E-04	8.15E-03
	H	1	0	1	1	1
F10	p-value	1.57E-04	2.57E-01	1.57E-04	1.57E-04	1.57E-04
	H	1	0	1	1	1
F11	p-value	1.57E-04	4.50E-01	1.57E-04	1.57E-04	5.88E-02
	H	1	0	1	1	0
F12	p-value	1.57E-04	1.57E-04	1.57E-04	1.57E-04	1.57E-04
	H	1	1	1	1	1
F13	p-value	2.50E-03	6.70E-04	2.50E-03	2.50E-03	1.15E-03
	H	1	1	1	1	1

the algorithms was tested using two set of benchmark functions. Fig.2 shows the convergence performance of the algorithm on the CEC2005 test function. On unimodal functions (F1, F7), OCSSA's convergence speed is better than several other algorithms, and on multimodal functions (F9, F11), Although WOA and GWO can finally reach the I deal optimal value, the convergence speed of OCSSA is still the fastest. Fig.3 displays the convergence speed of the algorithm on the CEC2015 test function. It can be seen from the figure that the convergence speed of OCSSA is still relatively good.

By synthesizing the performance of OCSSA and several other algorithms, it can be concluded that OCSSA performs very well both in terms of convergence accuracy and convergence speed. The main reason lies in the two methods introduced, which perform chaotic transformation of the population position during the initialization stage. And OBL, which makes the initial position of the population better than the randomly generated location, which accelerates the convergence rate to a certain extent, and during the optimization process, CLS can avoid the algorithm from falling into the local optimal solution, and OBL speeds up the convergence speed of the algorithm.

2) STATISTICAL TEST

In order to prove that the performance of the OCSSA4 algorithm is significantly different from several other algorithms on the test function, this section introduces Wilcoxon's rank

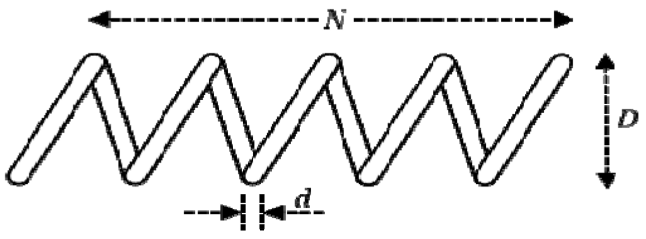

FIGURE 4. The schematic of the compression spring.

sum test. The detailed results are shown in Tables 9-10 for the CEC2005 and the CEC2015 benchmark functions, respectively. These two tables show the p-values and H-values of several comparison algorithms. When the P value is less than 0.05, H = 1, which means the null hypothesis, that is, the two have significant differences. Conversely, when the p-value is greater than 0.05, the null hypothesis does not hold, and H = 0, the difference between the two is not obvious. From Table 9, it can be concluded that OCSSA4 has significant differences in most test functions compared to other algorithms except F5. On the CEC2015 from table 10 test functions, OCSSA4 also shows significantly different performance from other algorithms.

F. REAL WORLD APPLICATIONS

This section applies the OCSSA proposed in this paper to a real-world problem, the tension/compression spring

TABLE 10. **The WRS test for CEC2015 benchmark functions.**

	Statistics	DE	WOA	PSO	ABC	GWO
F1	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	3.67E-05
	H	1	1	1	1	1
F2	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	3.07E-06
	H	1	1	1	1	1
F3	p-value	3.07E-06	3.07E-06	4.58E-06	5.38E-02	3.07E-06
	H	1	1	1	0	1
F4	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	5.25E-05
	H	1	1	1	1	1
F5	p-value	3.07E-06	3.07E-06	3.07E-06	1.60E-04	3.07E-06
	H	1	1	1	1	1
F6	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	1.46E-05
	H	1	1	1	1	1
F7	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	2.12E-05
	H	1	1	1	1	1
F8	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	2.00E-03
	H	1	1	1	1	1
F9	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	3.07E-06
	H	1	1	1	1	1
F10	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	3.07E-06
	H	1	1	1	1	1
F11	p-value	3.07E-06	1.76E-05	3.07E-06	2.13E-01	1.07E-02
	H	1	1	1	0	1
F12	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	3.07E-06
	H	1	1	1	1	1
F13	p-value	3.07E-06	1.25E-04	3.07E-06	5.34E-01	3.07E-06
	H	1	1	1	0	1
F14	p-value	3.07E-06	3.07E-06	3.07E-06	3.07E-06	3.07E-06
	H	1	1	1	1	1
F15	p-value	1.57E-04	1.57E-04	1.57E-04	1.57E-04	1.57E-04
	H	1	1	1	1	1

TABLE 11. **Comparative results for the tension/compression spring design problem.**

Algorithm	Optimum variables			Optimum weight
	d	D	N	
OCSSA	0.051550	0.353381	11.490433	0.0126684
PSO(Ha and Wang)	0.051728	0.357644	11.244543	0.0126747
DE(Huang et al.)	0.051609	0.354714	11.410831	0.0126702
WOA(Mirjalili and Lewis)	0.051207	0.345215	12.004032	0.0126763
ABC(Akay and Karaboga)	0.051690	0.356737	11.288000	0.012665
GWO(Mirjalili et al.)	0.051690	0.356737	11.288850	0.0126663
Mathematical optimization	0.053396	0.399180	9.185400	0.0127303
Constraint correction	0.050000	0.315900	14.250000	0.0128334

design problem. The optimal solution obtained by the algorithm should not violate many constraints.

The tension/compression problem consists of minimizing the weight $\left(f\left(\vec{x}\right)\right)$ of a tension/compression spring (Fig.4) subject to constraints on minimum deflection, shear stress, surge frequency, limit on outside diameter and on design variables. This problem has three decision variables, namely average coil diameter D, the wire diameter d, and number of effective coils N.

Formally, the problem can be expressed as:

Consider $\vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N]$,

$$\text{Minimize}\, f\left(\vec{x}\right) = (x_3+2)\,x_2x_1^2 \tag{8}$$

$$\text{Subject to}\ g_1\left(\vec{x}\right) = 1 - \frac{x_2^3x_3}{71758x_1^4} \le 0 \tag{9}$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1 \leq 0 \tag{10}$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \tag{11}$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{12}$$

Variable range:

$$\begin{cases} 0.05 \leq x_1 \leq 2 \\ 0.25 \leq x_2 \leq 1.3 \\ 2 \leq x_3 \leq 15 \end{cases} \tag{13}$$

This test case was solved using either mathematical techniques (for example, constraints correction at constant cost [30] and penalty functions [31]) or meta-heuristic techniques such as PSO [32], WOA, DE [33], GWO, ABC [34]. The comparison of results of these techniques and GWO are provided in Table 11. A different penalty function constraint handling strategy was applied in order to perform a fair comparison with literature [35]. It can be seen from Table 11 that OCSSA finds a design with the minimum weight for this problem, and it performs better than most algorithms except ABC and GWO.

V. CONCLUSION

In the present article, a novel OCSSA algorithm which combines OBL and CLS strategies is proposed to solve the global optimization problem. The OBL is introduced in the proposed algorithm to approximate the closer candidate solution to the global optima and CLS is employed for the exploitation of promising search regions of search space. The simulation results show that OCSSA performs better than SSA in optimizing 28 benchmark functions (including CEC2005 and CEC2015 functions) and maintains a fair balance between exploration and exploitation, which makes it robust. It also can be observed that the OCSSA algorithm combined with Sinusoidal chaotic mapping has the strongest competition. Moreover, the best-performing version of OCSSA is used to compare with other meta-heuristics. The OCSSA algorithm show the superiority over other algorithms in terms of optimization accuracy and convergence speed. In addition, we used Wilcoxon's rank sum to statistically test the algorithm, and the results indicate that the algorithm proposed in this paper is significantly different from several other algorithms in most benchmark functions. Finally, we apply the OCSSA algorithm to classic engineering problems, and the results show that the algorithm can solve real-world problem well. The future work may include adjusting SSA algorithm control parameters to optimize algorithm performance. In addition, more chaotic maps are also worth employed to OCSSA.

REFERENCES

[1] G. Kaur and S. Arora, "Chaotic whale optimization algorithm," *J. Comput. Des. Eng.*, vol. 5, no. 3, pp. 275–284, Jul. 2018.
[2] S. Saha and V. Mukherjee, "A novel quasi-oppositional chaotic antlion optimizer for global optimization," *Int. J. Speech Technol.*, vol. 48, no. 9, pp. 2628–2660, Dec. 2017.
[3] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evol. Comput.*, vol. 25, no. 1, pp. 1–54, Mar. 2017.
[4] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.
[5] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Inf. Sci.*, vol. 300, pp. 140–157, Apr. 2015.
[6] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[7] X. Zhao, H. Zhu, S. Aleksic, and Q. Gao, "Energy-efficient routing protocol for wireless sensor networks based on improved grey wolf optimizer," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 6, pp. 2644–2657, Jun. 2018.
[8] S. Arora and S. Singh, "Butterfly optimization algorithm: A novel approach for global optimization," *Soft Comput.*, vol. 23, no. 3, pp. 715–734, Mar. 2018.
[9] X. Zhao, Y. Cui, C. Gao, Z. Guo, and Q. Gao, "Energy-efficient coverage enhancement strategy for 3-D wireless sensor networks based on a vampire bat optimizer," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 325–338, Jan. 2020, doi: 10.1109/JIOT.2019.2952718.
[10] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.
[11] Y. Song, F. Wang, and X. Chen, "An improved genetic algorithm for numerical function optimization," *Int. J. Speech Technol.*, vol. 49, no. 5, pp. 1880–1902, Dec. 2018.
[12] S. L. Tilahun and J. M. T. Ngnotchouye, "Firefly algorithm for discrete optimization problems: A survey," *KSCE J. Civil Eng.*, vol. 21, no. 2, pp. 535–545, Jan. 2017.
[13] X. Cai, H. Wang, Z. Cui, J. Cai, Y. Xue, and L. Wang, "Bat algorithm with triangle-flipping strategy for numerical optimization," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 2, pp. 199–215, Nov. 2017.
[14] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel chaotic salp swarm algorithm for global optimization and feature selection," *Int. J. Speech Technol.*, vol. 48, no. 10, pp. 3462–3481, Mar. 2018.
[15] S. Asaithambi and M. Rajappa, "Swarm intelligence-based approach for optimal design of CMOS differential amplifier and comparator circuit using a hybrid salp swarm algorithm," *Rev. Sci. Instrum.*, vol. 89, no. 5, May 2018, Art. no. 054702.
[16] N. Singh, L. H. Son, F. Chiclana, and J.-P. Magnot, "A new fusion of salp swarm with sine cosine for optimization of non-linear functions," *Eng. with Comput.*, vol. 36, no. 1, pp. 185–212, Jan. 2019.
[17] A. Peng, "Particle swarm optimization algorithm based on chaotic theory and adaptive inertia weight," *J. Nanoelectron. Optoelectron.*, vol. 12, no. 4, pp. 404–408, Apr. 2017.
[18] S. Arora and P. Anand, "Chaotic grasshopper optimization algorithm for global optimization," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 4385–4405, Jan. 2018.
[19] M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *J. Comput. Des. Eng.*, vol. 5, no. 4, pp. 458–472, Oct. 2018.
[20] R. Jordehi, "Chaotic bat swarm optimisation (CBSO)," *Appl. Soft Comput.*, vol. 26, no. 1, pp. 523–530, Jan. 2015.
[21] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Modelling, Control Automat.*, Nov. 2005, pp. 695–701.
[22] Q. Kang, C. Xiong, M. Zhou, and L. Meng, "Opposition-based hybrid strategy for particle swarm optimization in noisy environments," *IEEE Access*, vol. 6, pp. 21888–21900, 2018.
[23] S. Zhang, Q. Luo, and Y. Zhou, "Hybrid grey wolf optimizer using elite opposition-based learning strategy and simplex method," *Int. J. Comput. Intell. Appl.*, vol. 16, no. 02, Jun. 2017, Art. no. 1750012.
[24] S.-Y. Park and J.-J. Lee, "Stochastic opposition-based learning using a beta distribution in differential evolution," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2184–2194, Oct. 2016.
[25] A. A. Ewees, M. Abd Elaziz, and E. H. Houssein, "Improved grasshopper optimization algorithm using opposition-based learning," *Expert Syst. Appl.*, vol. 112, pp. 156–172, Dec. 2018.

[26] S. Gupta and K. Deep, "An opposition-based chaotic grey wolf optimizer for global optimisation tasks," *J. Exp. Theor. Artif. Intell.*, vol. 31, no. 5, pp. 751–779, Dec. 2018.

[27] A. E. Hegazy, M. A. Makhlouf, and G. S. El-Tawel, "Feature selection using chaotic salp swarm algorithm for data classification," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 3801–3816, Dec. 2018.

[28] R. A. Ibrahim, M. A. Elaziz, and S. Lu, "Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization," *Expert Syst. Appl.*, vol. 108, pp. 1–27, Oct. 2018.

[29] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[30] S. Arora, *Introduction to Optimum Design*. New York, NY, USA: Academic, 2004.

[31] A. D. Belegundu and S. Arora, "A Study of Mathematical Programming Methods for Structural Optimization," *Int. J. Numer. Meth. Eng.*, vol. 21, no. 9, pp. 1601–1623, Oct. 1984.

[32] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Eng. Appl. Artif. Intell.*, vol. 20, no. 1, pp. 89–99, Feb. 2007.

[33] L. J. Li, Z. B. Huang, F. Liu, and Q. H. Wu, "A heuristic particle swarm optimizer for optimization of pin connected structures," *Comput. Struct.*, vol. 85, nos. 7–8, pp. 340–349, Apr. 2007.

[34] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1001–1014, Mar. 2010.

[35] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*. Bristol, U.K.: Luniver Press, 2010, pp. 105–114.

XIAOQIANG ZHAO received the Ph.D. degree from the Xi'an University of Technology, Xi'an, China, in 2015. He is currently a Professor with the Xi'an University of Posts and Telecommunications, Xi'an, China. His current research interests include the Internet of Things technology and environmental engineering.

FAN YANG received the B.S. degree in communication engineering from the Xi'an University of Posts and Telecommunications, Xi'an, Shaanxi, China, in 2016, where he is currently pursuing the master's degree in electronic and communication engineering. His research interests include group intelligence optimization and machine learning.

YAZHOU HAN received the B.S. degree in electronic information engineering from the Huanghe Science and Technology University, Henan, China, in 2017, where he is currently pursuing the master's degree in communication and information systems with the Xi'an University of Posts and Telecommunications, Xi'an, China. His research interests include signal and information processing, and wireless sensor networks.

YANPENG CUI is currently pursuing the master's degree with the Xi'an University of Posts and Telecommunications, Xi'an, China. His research interests include the technology and application of the Internet of Things.