IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 11, 2019, accepted August 22, 2019, date of publication August 27, 2019, date of current version September 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937741

Cuckoo Search Algorithm With Neighborhood Attraction for Numerical Optimization

JIATANG CHENG[1,2] **AND LEI WANG**[1]

[1]Faculty of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China
[2]Engineering College, Honghe University, Mengzi 661199, China

Corresponding author: Lei Wang (leeiwang@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 51669006 and Grant 61773314.

ABSTRACT Cuckoo search (CS) algorithm has been proved to be an effective method in solving numerical optimization problems. Nevertheless, with regard to Levy flight, each individual is attracted by the best solution found so far in the entire population, which may lead to premature convergence. Motivated by this observation, a new cuckoo search with neighborhood attraction (NACS) is proposed. In NACS, the neighborhood attraction scheme based on ring topology is firstly designed, where the best solution in a predefined neighborhood is employed to guide individual evolution. Then, to further enhance the exploration ability, the neighborhood attraction scheme and Levy flight are combined to generate potential candidate solutions. Moreover, the step size is adaptively regulated according to the degree of individual evolution. To validate the effectiveness of the presented algorithm, 25 extensively used benchmark test problems with different dimensions are employed. Experimental results reveal that the presented method is a competitive optimizer compared with other algorithms.

INDEX TERMS Cuckoo search, Levy flight, neighborhood attraction, optimization, step size.

I. INTRODUCTION

Lots of real-world problems can be converted to optimization problems. In recent decades, many meta-heuristic algorithms have been proposed to handle these optimization problems, such as particle swarm optimization (PSO) [1], differential evolution (DE) [2], genetic algorithm (GA) [3], teaching-learning-based optimization (TLBO) [4], human mental search (HMS) [5] and squirrel search algorithm (SSA) [6]. Since these optimization algorithms are superior to the traditional heuristic methods, they are appreciated by the evolutionary computing community.

Cuckoo search (CS) [7] is a promising meta-heuristic optimization algorithm for imitating the natural reproduction of some cuckoos. Distinct from other algorithms, the search process of CS is divided into two phases: global and local, corresponding to exploration and exploitation, respectively. The global phase is carried out by using Levy flight. The reason is that Levy distribution has infinite mean and variance, which helps to explore the solution space effectively. The local phase is executed by using the biased random walk with a certain probability to adjust the exploitation degree [8]. Due to the combined use of global and local phases, CS algorithm shows good convergence performance and has received wide attention from different fields. In recent years, a lot of work has been done on the research of CS algorithm and its application.

In Levy flight, the search behavior of CS is guided by the same best solution found so far in the entire population. Although this scheme can accelerate convergence, it is easy to be trapped in local optimum when solving complex multimode problems. Moreover, in evolutionary algorithms, the neighborhood topology is frequently used to enhance the convergence performance [9], and different topological structures can be employed to define neighborhood information, such as ring, star, pyramid and von Neumann. Thus, to strengthen the exploration ability, we propose a novel CS algorithm with neighborhood attraction, referred to as NACS. However, different neighborhood topologies may have different effects on different types of optimization problems. For multimode problems, neighborhood topologies with lower connectivity are better; for unimodal problems, neighborhood topologies with higher connectivity are recommended. That is to say, no neighborhood topology can always be better than others [10]. Also, the ring topology is considered to be the most indirect communication pattern, which is conducive

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

to exploring complex function landscapes. Therefore, the index-based ring topology is employed to define the neighborhood relationship flexibly. In the neighborhood attraction scheme, each individual is only attracted by the best solution obtained so far in a small neighborhood rather than the entire population. After that, considering that Levy flight may escape from the local optimum region with a high probability, the neighborhood attraction scheme and Levy flight are combined using a switching parameter, which appears in NACS algorithm as a new parameter. Further, in view of the correlation between the step size and optimization problem being solved, the average fitness of population is used to assess the evolution degree of individuals. On the basis of this, the step size is regulated adaptively to enhance the versatility of handling different optimization problems.

In brief, the main contributions of this work are as follow. First, the neighborhood attraction scheme based on neighborhood topology is introduced into CS to strengthen the global search capability. Next, a predefined switching parameter is set to control the combination of the neighborhood attraction scheme and Levy flight. Then, the appropriate neighborhood size and switching parameter are investigated experimentally. Finally, the average fitness of population is used to evaluate the degree of individual evolution, and the step size is then adjusted adaptively.

The paper is organized as follows. Section 2 introduces the basic CS, and Section 3 reviews the related work. In Section 4, the presented algorithm is described in detail, and the numerical experiments are reported in Section 5. Finally, the conclusion is given in Section 6.

II. BASIC CUCKOO SEARCH ALGORITHM

Cuckoo search (CS) algorithm is a nature-inspired global optimizer, belonging to the community of swarm intelligence. The nest chosen by cuckoo is considered to be a potential solution of the problem being solved, and the optimal solution may be found by using Levy flight and biased random walk.

It is assumed that N is the number of cuckoos and D represents the dimension of problem, the position of ith cuckoo is denoted as $X_i = \{x_{i1}, x_{i2}, \cdots, x_{iD}\}$. Then, the new position can be generated using the following equations:

$$X_i(k+1) = X_i(k) + step \cdot Levy(\lambda) \tag{1}$$

$$step = \alpha \cdot (X_i(k) - X_{best}(k)) \tag{2}$$

where $X_i(k)$ represents the position of ith cuckoo at kth generation, α is the step size which depends on the problem being solved, $X_{best}(k)$ is the best location obtained so far in the entire population. $Levy(\lambda)$ denotes the random search path, and it can be formulated as:

$$Levy(\lambda) = \frac{u}{|z|^{1/\beta}} \tag{3}$$

where u and z are two random numbers following the normal distribution, β often takes a fixed value of 1.5.

Another search phase of CS algorithm is implemented by means of discovery probability p_a. First of all, some of these nests are abandoned. After that, the same number of new nests can be generated by using biased random walk. This scheme is described as:

$$X_i(k+1) = X_i(k) + \gamma \cdot (X_j(k) - X_l(k)) \tag{4}$$

where γ is the scaling factor uniformly distributed in range [0, 1], X_j and X_l represents two distinct solutions randomly chosen in the population.

III. RELATED WORKS

So far, much effort has been done to further enhance the search ability of CS algorithm, and many CS variants have been developed. These improved versions can be divided into two major categories: parameter control and hybridization [11]. Besides, CS has been applied to various fields successfully because of its simplicity and efficiency.

In view of the important influence of control parameters on the performance, much meaningful work has been done on the control parameter settings of CS algorithm. Walton *et al.* [12] modified the step size in Levy flight by changing the number of iterations. Moreover, the information exchange between solutions with the best fitness was employed to accelerate convergence to the optimal value. After that, the method was tested on 7 benchmark functions. It is found that the proposed algorithm is superior to the classical CS. Li and Yin [13] introduced two mutation rules to balance the exploration and exploitation, and combined these two rules using a linear decreasing probability. Then, according to the relative success number of two newly added parameters in the previous iteration, an adaptive parameter adjustment strategy was developed. Finally, the performance of the proposed algorithm was evaluated on 16 benchmark functions. Comparison results show that this scheme is better than thirteen state-of-the-art algorithms. Huang *et al.* [14] utilized four different chaotic sequences to initialize the population. Also, these sequences were used to adjust the step size and reset the solution beyond the boundary. After that, two groups of test functions were employed to verify the effectiveness of the proposed method. Yang *et al.* [15] first defined the speed factor and aggregation factor using different fitness values. Then, the step size and discovery probability were regulated according to these two factors. Experiments on 10 benchmark functions show that the improved CS version is effective in tackling numerical optimization problems.

In terms of hybridization methods, the structures or operators of other algorithms have been introduced into CS to enhance the search ability. Wang *et al.* [16] combined the pitch adjustment operation in harmony search algorithm with CS to accelerate the convergence speed. Then, 14 classical benchmark functions were employed to test the performance. It is found that the improved version is better than CS and eight other algorithms. Besides, the authors also investigated the parameter sensitivity. Kanagaraj *et al.* [17] proposed a hybrid GA and CS method for engineering design optimization. The effectiveness of this hybrid algorithm were tested on 13 benchmark constrained functions

and 3 well-known design problems. Experimental results show that the proposed algorithm is a competitive scheme. Long *et al.* [18] introduced Solis and Wets local search technique into CS algorithm to balance the tradeoff between exploration and exploitation. Also, an augmented Lagrangian function was used for constraint-handling. Subsequently, the authors investigated the performance of this hybrid algorithm on 13 constrained benchmark problems. Daniel *et al.* [19] developed a hybrid cuckoo search-grey wolf optimization algorithm to fuse medical image. In this hybrid method, CS was used to search for the optimal control parameters of grey wolf optimization. Cheng *et al.* [20] proposed an ensemble CS variant for numerical optimization. With regard to this scheme, three different cuckoo search algorithms coexisted in the entire search process and competed to produce better offspring. Then, an external archive was introduced to further maintain population diversity.

With respect to applications, CS has been extensively applied to many domains, such as facility layout design [21], continuous dynamic optimization [22], reliability redundancy allocation [23], structural optimization [24], mining industry [25], scheduling [26], [27] and fault diagnosis [28] and so on. These applications indicate that CS algorithm is an effective and efficient optimizer for solving some real-world problems.

IV. CS ALGORITHM WITH NEIGHBORHOOD ATTRACTION

A. MOTIVATION

In Levy flight, the new candidate solutions can be generated through the guidance of the best solution found so far. That is to say, all individuals can interact with the best one at every generation, which allows CS algorithm to converge to the same point at a faster speed. Due to the guidance of the best solution in the population, however, CS is prone to premature convergence when solving complex optimization problems. Also, in evolutionary algorithms, the use of neighborhood information is a good idea. The individual interacts with a certain number of neighbors rather than the entire population, which helps to strengthen the exploration capability [29]. Motivated by these, CS with neighborhood attraction (NACS) is proposed in this paper.

Besides, the fixed step size is employed in the original CS. Since the step size is associated with the problem being solved, the fixed step size may lead to the loss of adaptability and cannot coordinate the dynamic search characteristics of CS algorithm. Therefore, according to the degree of individual evolution, an adaptive adjustment scheme of step size is presented to effectively coordinate the performance of large-scale global exploration and local fine search.

B. NEIGHBORHOOD ATTRACTION

In evolutionary algorithms, neighborhood topology is widely used to alleviate premature convergence [30]. Thus, a neighborhood attraction scheme by borrowing the neighborhood topology is proposed for guiding the evolution of individuals.

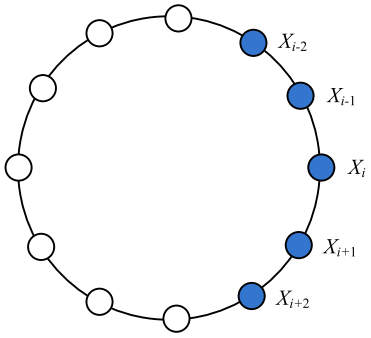

FIGURE 1. 2-neighborhood of X_i.

For the sake of simplicity, the ring topology is employed in this study.

Suppose that there are N individuals in the population, and each individual can be organized on the ring topology in terms of its indices. For example, the first individual X_1 connects with the last individual X_N, and it also connects with X_2. That is, X_N and X_2 are the immediate neighbors of X_1. Clearly, with respect to X_i, the k-neighborhood consists of $2k+1$ individuals, which are $\{X_{i-k}, \cdots, X_i, \cdots, X_{i+k}\}$. Thus, the neighborhood size m is equal to $2k+1$, where k is a predefined integer within the interval $[1, (N-1)/2]$. Note that the population size must be larger than the neighborhood size. To schematically illustrate the concept of ring topology, **Figure 1** reports the 2-neighborhood of X_i.

In this neighborhood attraction scheme, each individual is only attracted by the best solution found so far in the k-neighborhood rather than that in the entire population. That is to say, the individual located in a good region of search space only affects its immediate neighbors, thus this strategy can be regarded as a local search scheme. Also, the k-neighborhood is smaller in comparison with the entire population, so the attraction of the best individual in each neighborhood is weaker, which can enhance the population diversity and alleviate premature convergence. As discussed above, the neighborhood attraction scheme can be expressed as:

$$X_i(k+1) = X_i(k) + r \cdot \left(X_{i,neig}(k) - X_i(k)\right) \tag{5}$$

where $X_{i,neig}$ denotes the best individual in the neighborhood of X_i, r is a random number in $[0, 1]$.

Besides, Levy flight has the characteristics of occasional long jumps, resulting in better global search ability. In this paper, the neighborhood attraction scheme and Levy flight are combined together to yield promising solutions, which can further enhance the probability of jumping out of the local optimum. In this case, the updating equation is then modified as follows:

$$X_i(k+1) = \begin{cases} X_i(k) + step \cdot Levy(\lambda) & rand < p_s \\ X_i(k) + r \cdot \left(X_{i,neig}(k) - X_i(k)\right) & otherwise \end{cases} \tag{6}$$

TABLE 1. Mean errors obtained by NACS with different m values for the classical benchmark functions at 30D ($p_s = 0.8$).

No.	$m = 3$	$m = 5$	$m = 7$	$m = 9$	$m = 11$
f1	2.33E–64	1.51E–72	1.82E–76	1.72E–79	**1.32E–81**
f2	1.86E–39	6.57E–45	5.30E–48	4.25E–50	**1.09E–51**
f3	5.16E–12	1.95E–12	6.70E–13	7.88E–13	**2.57E–13**
f4	5.94E–06	2.01E–06	1.59E–06	1.65E–06	**1.45E–06**
f5	**1.42E–01**	2.66E–01	7.97E–01	5.32E–01	6.64E–01
f6	**6.24E+02**	6.34E+02	6.97E+02	6.83E+02	7.43E+02
f7	1.19E+01	8.59E+00	**8.46E+00**	9.05E+00	9.55E+00
f8	**7.99E–15**	9.30E–15	8.23E–15	8.47E–15	8.35E–15
f9	**0.00E+00**	6.57E–04	4.93E–04	4.93E–04	5.75E–04
f10	**1.57E–32**	3.50E–03	**1.57E–32**	3.50E–03	**1.57E–32**
f11	**1.35E–32**	3.66E–04	1.37E–32	1.37E–32	1.37E–32

where p_s is called the switching parameter, *rand* represents a random number between 0 and 1.

C. STEP SIZE ADJUSTMENT

As mentioned above, the step size α is related to the optimization problem being solved, and the proper setting of step size is helpful to enhance the convergence performance of CS algorithm. Therefore, some work has been done on the adaptive parameter adjustment mechanism. Among these adjustment strategies, it is a common scheme to regulate step size by using the number of iterations. However, the characteristics of different optimization problems are different. Also, it is difficult to accurately grasp the relationship between the number of iterations and the degree of individual evolution when solving complex optimization problems. Thus, the adjustment method using only the iteration number usually has poor universality. In this scheme, the average fitness of the entire population is employed to estimate the evolution degree of individuals, and the step size is adjusted accordingly, which has good universality.

In Levy flight, the adjustment strategy of step size α is as follows:

$$\alpha = \frac{1}{5 + \exp(\alpha_0)} \tag{7}$$

$$\alpha_0 = \frac{f_i - f_{avg}}{f_{best} - f_{avg}} \tag{8}$$

where f_i is the fitness of ith solution, f_{avg} denotes the average fitness of population, and f_{best} represents the fitness of the best solution found so far.

Without losing generality, the problem of minimization is considered in this paper. Clearly, if $f_i$is larger than f_{avg}, it means that the individual is far away from the optimal position. At this time, the step size α can take a larger value to enhance global search capability. Similarly, in case f_i is smaller than f_{avg}, the individual is close to the optimal solution, and the step size is set to a smaller value for local search. Therefore, the average fitness of population can reflect the degree of individual evolution.

In terms of the above descriptions, the implementation of NACS algorithm is presented in **Algorithm 1**.

Algorithm 1 Proposed NACS

```
Set the population size N, problem dimension D,
maximum iteration number k_max, discovery probability
p_a, switching parameter p_s and distribution parameter
β;
Initialize the individual X_i and calculate its fitness value
f_i;
Find out the best solution X_best and its fitness value
f_best;
Define the neighborhood of X_i;
k = 1;
while k < k_max do
   Calculate the step size α using "(7)";
   Determine the best solution X_{i,neig} in the
   neighborhood of X_i;
   for i = 1 to N do
      Generate the new solution X_{i,new} using "(6)" and
      calculate its fitness value f_{i,new};
      if f_{i,new} < f_i then
        X_i = X_{i,new} and f_i = f_{i,new};
      end if
      Generate the new solution X_{i,new} using "(4)"
      and calculate its fitness value f_{i,new};
      if f_{i,new} < f_i then
        X_i = X_{i,new} and f_i = f_{i,new};
      end if
      if f_i < f_best then
        X_best = X_i and f_best = f_i;
      end if
   end for
   k = k + 1;
end while
```

TABLE 2. Mean errors obtained by NACS with different m values for the classical benchmark functions at 50D ($p_s = 0.8$).

No.	$m = 3$	$m = 5$	$m = 7$	$m = 9$	$m = 11$
f1	7.74E–66	5.06E–74	4.27E–78	4.90E–81	**4.52E–83**
f2	4.38E–42	4.08E–48	1.65E–51	8.01E–54	**1.68E–55**
f3	3.92E–06	3.30E–06	1.87E–06	**1.15E–06**	1.52E–06
f4	6.69E–04	**4.23E–04**	4.42E–04	6.71E–04	7.17E–04
f5	7.17E+00	4.58E+00	2.49E+00	**1.36E+00**	1.87E+00
f6	**2.33E+03**	2.69E+03	2.77E+03	2.54E+03	2.73E+03
f7	3.33E+01	3.27E+01	2.90E+01	**2.80E+01**	3.26E+01
f8	1.73E–14	**1.63E–14**	1.69E–14	1.68E–14	1.79E–14
f9	**0.00E+00**	1.60E–03	5.75E–04	3.29E–04	1.70E–03
f10	**9.42E–33**	6.20E–03	2.10E–03	1.04E–02	1.02E–02
f11	**1.47E–32**	7.32E–04	1.10E–03	7.32E–04	1.50E–03

V. EXPERIMENTAL STUDY

A. BENCHMARK FUNCTIONS AND INVOLVED CS VARIANTS

In these experiments, the performance of NACS algorithm is investigated on 25 well-known benchmark functions from two different test suites. These test functions are employed in previous studies [31]–[33] and are given in the Appendix. The first test suite includes 11 classical functions, and the second test suite consists of 14 shifted

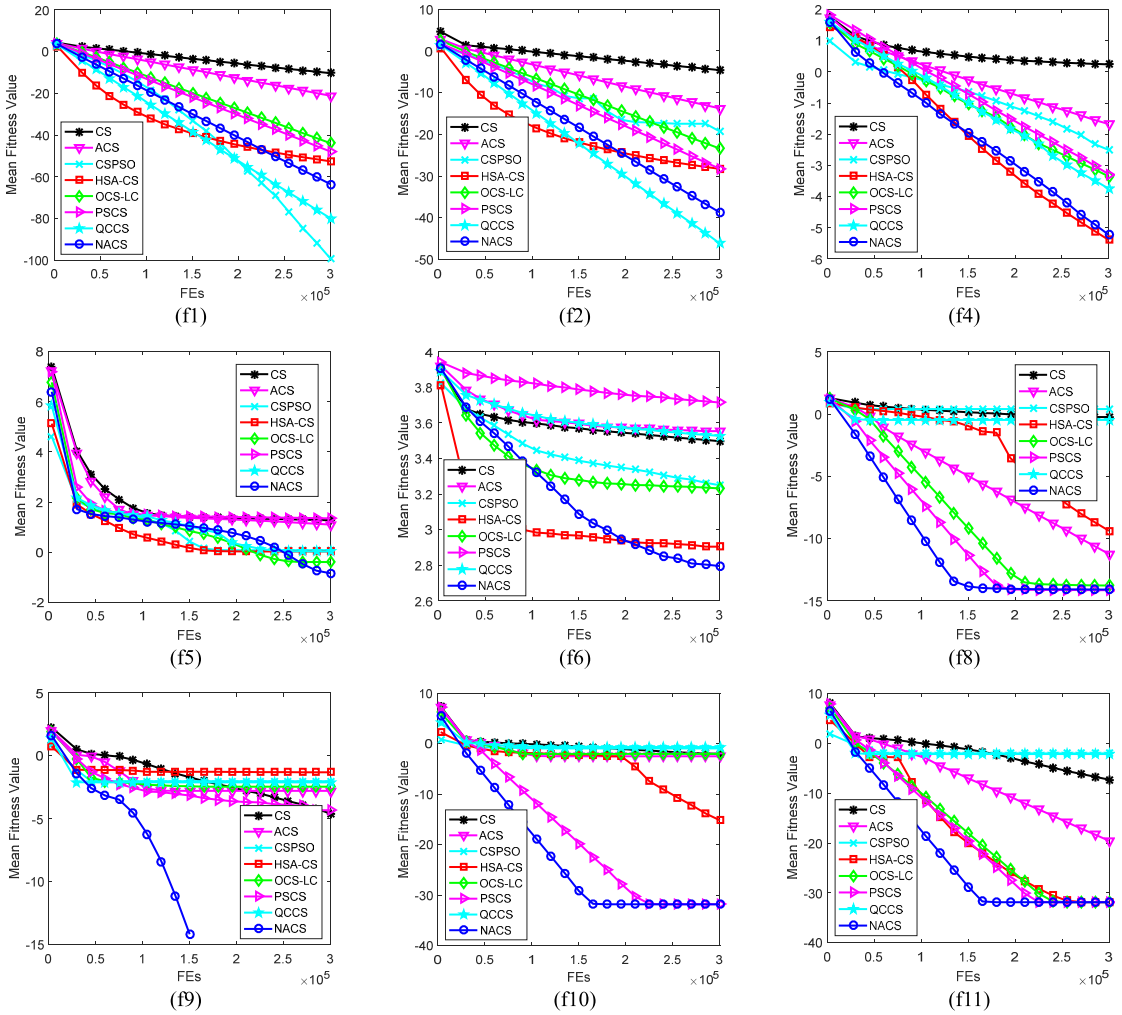

FIGURE 2. Evolutionary curves of CS, ACS, CSPSO, HSA-CS, OCS-LC, PSCS, QCCS and NACS on the first test suite at 30*D*.

TABLE 3. Mean errors obtained by NACS with different p_s values for CEC 2005 benchmark functions F1 – F10 at 30*D* ($m = 3$).

No.	$p_s = 0.1$	$p_s = 0.2$	$p_s = 0.3$	$p_s = 0.4$	$p_s = 0.5$	$p_s = 0.6$	$p_s = 0.7$	$p_s = 0.8$	$p_s = 0.9$	$p_s = 1.0$
F1	3.7E–28	4.3E–28	1.5E–28	1.4E–28	9.3E–29	1.3E–29	1.7E–30	**0.0E+00**	**0.0E+00**	**0.0E+00**
F2	9.8E–02	2.8E–07	1.1E–12	**2.9E–15**	9.3E–15	7.7E–14	2.1E–12	3.3E–11	2.4E–10	8.4E–10
F3	1.9E+05	2.3E+05	1.6E+05	1.0E+05	1.6E+05	**8.1E+04**	8.8E+04	1.0E+05	1.5E+05	1.5E+05
F4	3.0E+03	1.1E+02	9.1E+00	6.8E–02	1.2E–01	1.0E–01	**5.0E–02**	2.6E–01	4.2E–01	2.6E+00
F5	3.6E+03	2.5E+03	1.9E+03	1.6E+03	1.7E+03	1.4E+03	1.3E+03	**1.1E+03**	1.3E+03	1.3E+03
F6	2.1E+03	4.5E+02	7.8E+01	2.1E+01	4.0E+00	**1.3E+00**	1.6E+00	1.4E+00	3.5E+00	7.8E+00
F7	6.6E–02	2.5E–02	2.4E–02	2.7E–02	2.5E–02	2.3E–02	1.9E–02	1.5E–02	9.3E–03	**4.2E–03**
F8	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01
F9	4.6E+01	4.8E+01	4.3E+01	3.5E+01	2.7E+01	2.0E+01	2.0E+01	**1.9E+01**	2.7E+01	4.3E+01
F10	2.5E+02	1.6E+02	1.2E+02	1.0E+02	8.0E+01	8.0E+01	**6.8E+01**	7.8E+01	7.5E+01	7.5E+01

rotated problems proposed in CEC 2005 [34]. These benchmark functions involve a series of problem features, such as unimodal, multimodal, non-separable, rotated and so on. Generally speaking, it is difficult to find out the optimal solutions of these shifted rotated or non-separable problems.

In this section, the proposed NACS is compared with the classical CS and six recently developed CS variants. These modified versions include adaptive CS (ACS) [35], a hybridization of CS and PSO (CSPSO) [36], hybrid self-adaptive CS (HSA-CS) [37], particle swarm inspired CS (PSCS) [38], oriented CS with Levy distribution and

TABLE 4. Mean errors obtained by NACS with different p_s values for CEC 2005 benchmark functions F1 – F10 at 50D ($m = 3$).

No.	$p_s = 0.1$	$p_s = 0.2$	$p_s = 0.3$	$p_s = 0.4$	$p_s = 0.5$	$p_s = 0.6$	$p_s = 0.7$	$p_s = 0.8$	$p_s = 0.9$	$p_s = 1.0$
F1	5.4E–14	2.6E–27	6.9E–28	4.9E–28	3.9E–28	2.0E–28	7.5E–29	1.7E–30	6.7E–30	**0.0E+00**
F2	4.4E+00	2.8E–04	5.7E–08	**1.0E–08**	1.3E–07	1.3E–06	7.4E–06	4.0E–05	2.0E–04	4.9E–04
F3	**3.7E+06**	6.2E+06	6.9E+06	8.0E+06	7.3E+06	8.4E+06	9.9E+06	9.0E+06	1.2E+07	1.1E+07
F4	2.1E+04	7.0E+03	1.5E+03	8.1E+02	**6.3E+02**	7.8E+02	1.1E+03	1.6E+03	2.7E+03	2.9E+03
F5	9.6E+03	7.8E+03	6.3E+03	5.5E+03	5.3E+03	4.6E+03	4.7E+03	**4.4E+03**	5.0E+03	5.0E+03
F6	1.1E+03	1.1E+03	4.3E+02	2.4E+02	1.1E+02	5.2E+01	1.1E+01	**7.6E+00**	3.0E+01	3.6E+01
F7	2.0E–01	7.5E–03	8.0E–03	**4.2E–03**	1.5E–02	4.8E–03	9.0E–03	6.2E–03	4.3E–03	4.4E–03
F8	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01	2.1E+01
F9	1.5E+02	1.5E+02	1.4E+02	1.2E+02	9.5E+01	8.1E+01	6.4E+01	**5.4E+01**	6.1E+01	7.6E+01
F10	5.8E+02	4.5E+02	3.3E+02	2.7E+02	2.4E+02	2.1E+02	1.8E+02	**1.7E+02**	**1.7E+02**	1.9E+02

TABLE 5. Numerical results of CS, ACS, CSPSO, HSA-CS, OCS-LC, PSCS, QCCS and NACS on the first test suite at 30D.

No.	Term	CS	ACS	CSPSO	HSA-CS	OCS-LC	PSCS	QCCS	NACS
f1	Mean	4.94E–011	6.90E–022	**3.89E–100**	2.52E–053	1.10E–044	1.25E–048	1.02E–080	2.33E–064
	SD	6.88E–011	1.00E–021	1.14E–099	3.35E–053	1.55E–044	3.76E–048	5.19E–080	1.00E–063
	Best	1.76E–012	5.89E–023	2.69E–109	2.43E–055	5.00E–047	2.17E–050	2.23E–085	3.51E–066
f2	Mean	2.46E–005	1.63E–014	5.23E–020	5.24E–029	3.80E–024	5.70E–029	**6.55E–047**	1.86E–039
	SD	1.20E–005	1.27E–014	2.75E–019	6.00E–029	4.85E–024	8.28E–029	1.50E–046	1.13E–039
	Best	6.07E–006	2.38E–015	1.82E–032	6.83E–030	1.04E–025	4.22E–030	1.14E–048	2.49E–040
f3	Mean	1.53E–008	1.80E–005	**5.93E–022**	1.95E–013	1.02E–013	2.92E+003	2.17E–012	5.16E–012
	SD	1.99E–008	2.74E–005	3.24E–021	2.85E–013	1.43E–013	1.41E+003	2.66E–012	8.45E–012
	Best	1.76E–010	4.97E–007	3.96E–030	3.92E–015	9.00E–017	5.09E+002	4.43E–014	7.12E–014
f4	Mean	1.75E+000	2.14E–002	3.10E–003	**4.07E–006**	4.43E–004	5.24E–004	1.80E–004	5.94E–006
	SD	9.30E–001	1.50E–002	5.10E–003	5.23E–006	6.94E–004	3.52E–004	3.92E–004	6.73E–006
	Best	2.71E–001	3.90E–003	6.69E–005	5.14E–017	3.45E–005	9.21E–005	2.44E–006	7.51E–007
f5	Mean	1.86E+001	1.25E+001	1.20E+000	1.06E+000	3.99E–001	2.23E+001	1.06E+000	**1.42E–001**
	SD	3.48E+000	2.57E+000	1.86E+000	1.79E+000	1.22E+000	4.42E+000	1.79E+000	7.27E–001
	Best	1.02E+001	4.20E–003	4.90E–023	1.27E–020	2.13E–023	1.98E+001	2.55E–014	1.55E–016
f6	Mean	3.14E+003	3.58E+003	1.78E+003	8.01E+002	1.70E+003	5.18E+003	3.36E+003	**6.24E+002**
	SD	3.01E+002	8.08E+002	6.48E+002	2.77E+002	5.12E+002	6.49E+002	1.08E+003	2.84E+002
	Best	2.50E+003	1.50E+003	4.85E+002	2.37E+002	6.91E+002	2.45E+003	1.49E+003	1.18E+002
f7	Mean	4.43E+001	3.35E+001	1.80E+001	**1.03E+001**	2.07E+001	8.62E+001	2.74E+001	1.19E+001
	SD	9.59E+000	1.04E+001	6.49E+000	5.24E+000	7.30E+000	2.67E+001	7.46E+000	4.14E+000
	Best	2.02E+001	1.69E+001	4.97E+000	2.98E+000	8.95E+000	2.58E+001	1.59E+001	4.97E+000
f8	Mean	5.72E–001	5.62E–012	2.50E+000	4.19E–010	1.68E–014	**7.05E–015**	3.37E–001	7.99E–015
	SD	6.06E–001	3.12E–012	6.71E–001	2.23E–009	3.70E–015	1.60E–015	5.74E–001	0.00E+000
	Best	7.54E–006	2.30E–012	5.77E–014	7.99E–015	7.99E–015	4.44E–015	7.99E–015	7.99E–015
f9	Mean	2.55E–005	1.60E–003	9.80E–003	5.02E–002	3.60E–003	4.84E–005	8.00E–003	**0.00E+000**
	SD	8.92E–005	3.80E–003	1.20E–002	5.49E–002	4.70E–003	2.38E–004	9.40E–003	0.00E+000
	Best	5.58E–011	0.00E+000	2.22E–016	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
f10	Mean	1.06E–002	3.50E–003	1.24E–001	7.19E–016	1.04E–002	**1.57E–032**	2.36E–001	**1.57E–032**
	SD	4.17E–002	1.89E–002	2.30E–001	3.94E–015	3.16E–002	5.57E–048	4.86E–001	5.57E–048
	Best	8.35E–011	9.15E–023	1.57E–032	1.57E–032	1.57E–032	1.57E–032	1.57E–032	1.57E–032
f11	Mean	5.34E–008	2.76E–020	9.40E–003	**1.35E–032**	**1.35E–032**	**1.35E–032**	6.90E–003	**1.35E–032**
	SD	1.90E–007	5.74E–020	2.18E–002	5.57E–048	5.57E–048	5.57E–048	1.20E–002	5.57E–048
	Best	1.41E–010	4.42E–022	1.35E–032	1.35E–032	1.35E–032	1.35E–032	1.35E–032	1.35E–032

Cauchy distribution (OCS-LC) [39] and quantum chaotic CS (QCCS) [40]. For the purpose of comparison, the population size N of all algorithms is set to 50, except for HSA-CS using a linear population reduction strategy. These benchmark problems are once tested with $D = 30$ and once with $D = 50$, and the maximum function evaluation is set to $10000 \times D$. Additionally, each algorithm is executed in 30 independent tests. The mean value, standard deviation (SD) and the best result are reported for comparison. In this work, all the experiments are carried out on a computer (Intel i7-4790 CPU, 8.00GB RAM and Windows 7 system) using MATLAB R2016a.

The parameter configurations of these algorithms are given as follows:

1) CS: $\alpha = 0.01, p_a = 0.25$.

2) ACS: $p_a = 0.25$.

3) CSPSO: $\alpha_{max} = 0.5, \alpha_{min} = 0.01, p_a = 0.25$.

4) HSA-CS: $\alpha = 0.9, F = 0.5, C_r = 0.9, p_a = 0.1, p_b = 0.8, p_e = 0.1, N_{max} = 200, N_{min} = 10$.

5) PSCS: $\vartheta \sim N(0, 0.5^2)$, $\emptyset \sim N(0.5, 0.5^2)$, $\alpha = 0.01, p_a = 0.25, q = 0.05$.

6) OCS-LC: $\alpha = 0.01, p_a = 0.25$.

7) QCCS: $\alpha = 1, \delta = 1.6, p_a = 0.25$.

8) NACS: $p_a = 0.25, m = 3, p_s = 0.8$.

TABLE 6. Numerical results of CS, ACS, CSPSO, HSA-CS, OCS-LC, PSCS, QCCS and NACS on the first test suite at 50*D*.

No.	Term	CS	ACS	CSPSO	HSA-CS	OCS-LC	PSCS	QCCS	NACS
f1	Mean	3.68E–009	6.84E–019	**2.56E–068**	4.39E–060	3.05E–043	2.26E–043	9.44E–068	7.74E–066
	SD	5.21E–009	7.29E–019	9.45E–068	6.50E–060	4.96E–043	3.27E–043	2.28E–067	1.69E–065
	Best	3.12E–010	9.89E–020	4.48E–081	1.06E–061	1.69E–045	4.66E–045	3.70E–070	1.56E–067
f2	Mean	5.23E–006	6.52E–012	2.32E–014	8.32E–034	6.39E–025	1.03E–026	3.42E–041	**4.38E–042**
	SD	2.40E–006	3.25E–012	1.10E–013	9.08E–034	5.17E–025	8.49E–027	6.63E–041	2.82E–042
	Best	2.25E–006	2.77E–012	4.95E–022	1.18E–034	3.39E–026	1.36E–027	1.21E–043	7.33E–043
f3	Mean	4.42E–002	2.01E–001	**1.97E–008**	2.32E–008	7.68E–006	3.18E+004	4.47E–005	3.92E–006
	SD	3.89E–002	9.94E–002	9.85E–008	2.99E–008	1.24E–005	7.41E+003	8.00E–005	3.27E–006
	Best	4.70E–003	2.92E–002	3.07E–014	1.78E–009	3.67E–007	1.07E+004	2.13E–006	5.05E–007
f4	Mean	7.06E+000	1.25E+000	3.25E+000	5.39E–002	1.16E+000	5.57E–001	3.51E+000	**6.69E–004**
	SD	1.92E+000	4.82E–001	1.07E+000	3.57E–002	7.90E–001	1.97E–001	1.95E+000	5.87E–004
	Best	3.04E+000	5.28E–001	1.33E+000	2.07E–004	3.19E–001	2.75E–001	1.02E+000	1.21E–004
f5	Mean	4.55E+001	3.54E+001	**1.06E+000**	1.34E+000	3.62E+000	4.60E+001	2.23E+000	7.17E+000
	SD	2.36E+001	8.54E+000	1.79E+000	2.18E+000	5.20E+000	1.86E+001	2.96E+000	6.48E+000
	Best	7.99E–002	2.95E+001	4.95E–018	1.80E–017	3.42E–018	3.09E+001	2.32E–005	9.22E–011
f6	Mean	6.05E+003	6.03E+003	5.17E+003	**1.92E+003**	3.39E+003	1.02E+004	7.54E+003	2.33E+003
	SD	5.06E+002	6.02E+002	1.06E+003	4.91E+002	6.67E+002	5.26E+002	2.99E+003	5.56E+002
	Best	4.89E+003	5.09E+003	2.93E+003	1.03E+003	1.78E+003	9.15E+003	2.86E+003	1.28E+003
f7	Mean	7.03E+001	6.36E+001	3.89E+001	**2.84E+001**	4.09E+001	1.61E+002	5.83E+001	3.33E+001
	SD	1.31E+001	1.64E+001	8.94E+000	8.90E+000	1.09E+001	5.67E+001	1.40E+001	1.05E+001
	Best	4.77E+001	3.88E+001	1.79E+001	1.29E+001	1.29E+001	5.63E+001	3.18E+001	1.99E+001
f8	Mean	1.20E+000	8.58E–011	3.15E+000	2.57E–007	3.35E–014	**1.00E–014**	1.15E+000	1.73E–014
	SD	6.34E–001	4.18E–011	9.24E–001	1.25E–006	6.33E–015	3.05E–015	9.34E–001	3.55E–015
	Best	1.46E–005	9.93E–012	1.47E+000	1.15E–014	2.22E–014	7.99E–015	2.22E–014	1.15E–014
f9	Mean	1.90E–003	2.47E–004	1.16E–002	6.79E–002	5.60E–003	2.47E–004	6.30E–003	**0.00E+000**
	SD	3.60E–003	1.40E–003	1.60E–002	7.01E–002	6.10E–003	1.40E–003	1.03E–002	0.00E+000
	Best	5.51E–010	0.00E+000	8.88E–016	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
f10	Mean	1.05E–002	6.20E–003	1.06E–001	2.10E–003	1.87E–002	**9.42E–033**	2.66E–001	**9.42E–033**
	SD	2.35E–002	1.90E–002	1.45E–001	1.14E–002	5.93E–002	1.39E–048	4.05E–001	1.39E–048
	Best	1.65E–009	4.55E–020	1.00E–031	9.42E–033	9.42E–033	9.42E–033	9.42E–033	9.42E–033
f11	Mean	1.20E–003	2.01E–017	8.82E–001	9.20E–018	7.32E–004	7.32E–004	1.42E–001	**1.47E–032**
	SD	6.10E–003	3.27E–017	2.28E+000	5.04E–017	2.80E–003	2.80E–003	5.73E–001	3.64E–033
	Best	2.24E–009	2.82E–019	9.24E–032	1.35E–032	1.35E–032	1.35E–032	1.35E–032	1.35E–032

B. INFLUENCE OF PARAMETERS m AND p_s

In this subsection, to demonstrate the influence of different settings of neighborhood size m and switching parameter p_s on the performance of NACS algorithm, the two sets of test problems with 30 and 50 dimensions are used as benchmarks for experimental research. For these 30D and 50D problems in the 30 independent runs, the comparative tests are conducted using different m values (i.e., 3, 5, 7, 9 and 11). After that, the p_s values from 0.1 to 1 are also selected for analysis. The numerical results with regard to the mean errors are reported in Tables 1 – 4 respectively, and the best results are shown in bold. In these experiments, one parameter remains unchanged, and the other investigated is variable to find its approximate optimal setting. The process is repeated until the proper values of all parameters can be found.

From **Table 1**, we can see that NACS performs well on f5, f6, f8, f9, f10 and f11 when the neighborhood size m is set to 3, and $m = 7$ provides the solutions with higher accuracy on f7 and f10. Similarly, $m = 11$ yields better results on f1, f2, f3, f4 and f10, m being set to 5 does not work for multimode functions f9 – f11, the reason is that the individual may oscillate when solving these problems. Apparently, with respect to the unimodal functions f1 – f4, m being set to 11 is the best choice. When m is set to 3, NACS is good at handling these multimodal functions. As indicated in **Table 2**, $m = 3$ is also competitive in solving f6, f9, f10 and f11. Therefore, to enhance the ability to tackle multimode problems, the neighborhood size m is set to 3 in this paper.

With respect to CEC 2005 test functions F1 – F10 at 30D, **Table 3** provides the mean errors obtained by NACS algorithm with different p_s values. Obviously, in terms of the experimental results, $p_s = 0.4$ yields better solution on F2, $p_s = 0.6$ obtains reasonable results on F3 and F6, and $p_s = 0.7$ outperforms others on F4 and F10. Further, $p_s = 0.8$ performs well on F1, F5 and F9, and $p_s = 1.0$ provides reasonable solutions on F1 and F7. Moreover, from **Table 4**, $p_s = 0.8$ performs well on F5, F6, F9 and F10, $p_s = 0.4$ outperforms others on F2 and F7, while it does not work for F6, F9 and F10. Thus, for this test suite with different dimensions, the switching parameter p_s can be set to 0.5 – 1.0, which is a good choice. In this work, p_s is set to 0.8 in the following experiments.

C. COMPARISON RESULTS ON THE FIRST TEST SUITE

In the first test suite, f1 – f5 are unimodal functions, and f6 – f11 are multimodal functions. The unimodal problems are used for evaluating the convergence rate of the algorithm, and the multimodal functions are used to test the exploration capability. For these benchmark functions with

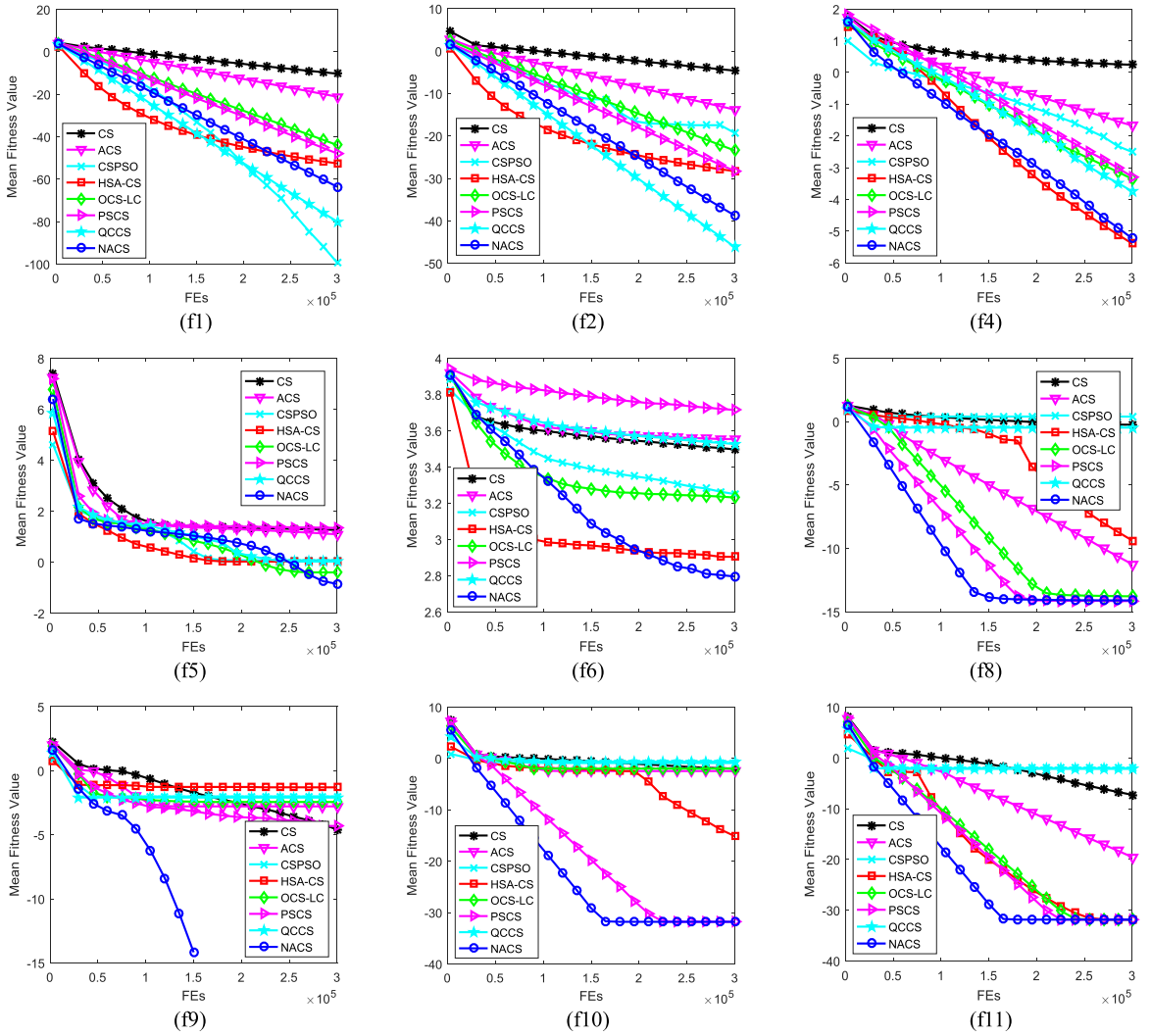

FIGURE 3. Evolutionary curves of CS, ACS, CSPSO, HSA-CS, OCS-LC, PSCS, QCCS and NACS on the second test suite at 30*D*.

30 dimensions, the mean value, standard deviation (SD) and best value obtained by these involved algorithms are presented in **Table 5**. Also, for some functions at 30*D*, the evolutionary curves of the mean fitness values are plotted in **Figure 2**, where *FEs* represents the number of function evaluations. Additionally, to evaluate the influence of dimension on search ability, the convergence performance of NACS is also investigated on these functions with 50 dimensions, and the numerical results are listed in **Table 6**. The lowest mean values are highlighted in bold.

As shown in **Table 5,** in terms of solution accuracy, NACS performs well on f5, f6, f9, f10 and f11. Especially, only NACS converges to the optimal solution on f9. Also, CSPSO performs best on the unimodal functions f1 and f3, HSA-CS has the best performance on f4 and f7, PSCS does well in handling f8, f10 and f11, and QCCS yields high-quality solution on f2. Need to add that, with regard to f1, f2, f4, f7 and f8, NACS also provides the solutions with high accuracy. Besides, in solving these multimodal functions, the searching behavior of these compared algorithms may make the individual oscillates on some problems. Especially for CS, ACS, CSPSO and QCCS, these individuals are easily attracted by the best solution found so far in the entire population. Thus, it is difficult for them to escape from the local optimal region, leading to convergence instability. As discussed above, NACS is very effective on these benchmark functions.

As can be seen from **Figure 2**, NACS also has better convergence performance for the unimodal functions, even though its convergence speed is not the fastest among these selected algorithms. With regard to f5, it has the property of multimode function with the increase of dimension. Therefore, finding its optimal solution is very challenging. Obviously, NACS converges slower than HSA-CS for f5 and f6, while the solutions found are closer to the optimal value. Additionally, compared to other algorithms, NACS obtains high-quality solutions and makes

TABLE 7. Numerical results of CS, ACS, CSPSO, HSA-CS, OCS-LC, PSCS, QCCS and NACS on the second test suite at 30*D*.

No.	Term	CS	ACS	CSPSO	HSA-CS	OCS-LC	PSCS	QCCS	NACS
F1	Mean	1.09E–009	5.75E–022	6.76E–028	**0.00E+000**	8.84E–030	**0.00E+000**	**0.00E+000**	**0.00E+000**
	SD	2.01E–009	7.67E–022	3.29E–028	0.00E+000	3.77E–029	0.00E+000	0.00E+000	0.00E+000
	Best	2.70E–011	2.78E–023	5.05E–029	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
F2	Mean	8.32E–006	1.14E–004	3.90E–006	7.26E–013	**2.62E–013**	2.59E+003	2.55E–011	5.71E–011
	SD	3.06E–005	1.15E–004	9.44E–006	6.89E–013	6.02E–013	1.33E+003	6.66E–011	1.24E–010
	Best	8.74E–009	2.78E–006	6.03E–010	2.10E–014	4.76E–015	9.47E+002	8.57E–014	2.09E–012
F3	Mean	1.20E+005	1.37E+005	2.37E+005	**3.73E+004**	2.18E+005	4.34E+007	1.09E+006	8.00E+004
	SD	1.66E+005	1.38E+005	2.09E+005	6.32E+004	4.09E+005	1.26E+007	6.84E+005	1.04E+005
	Best	3.71E–004	6.15E+003	5.09E+003	4.37E+002	3.72E–004	2.58E+007	6.04E+004	9.16E+002
F4	Mean	3.55E+003	1.97E+002	3.43E+000	**5.43E–002**	2.58E+001	1.25E+004	6.04E+002	1.03E+000
	SD	3.38E+003	2.19E+002	4.51E+000	1.72E–001	5.97E+001	7.36E+003	9.24E+002	2.62E+000
	Best	2.51E+002	4.14E+000	4.90E–003	5.12E–005	5.43E–002	4.09E+003	4.98E+001	2.90E–003
F5	Mean	4.45E+003	2.18E+003	**1.12E+003**	1.72E+003	3.91E+003	2.58E+003	3.95E+003	**1.12E+003**
	SD	1.47E+003	8.20E+002	4.04E+002	6.69E+002	1.51E+003	5.15E+002	1.33E+003	4.78E+002
	Best	1.20E+003	1.05E+003	4.62E+002	7.69E+002	1.59E+003	1.62E+003	1.83E+003	2.37E+002
F6	Mean	3.77E+001	1.80E+001	6.48E+000	1.33E+000	1.24E+000	3.95E+001	9.41E–001	**4.64E–001**
	SD	2.99E+001	1.41E+001	1.98E+001	1.91E+000	1.84E+000	3.98E+001	1.73E+000	1.21E+000
	Best	4.59E+000	8.02E+000	3.80E–014	2.53E–020	8.85E–015	1.62E+001	1.35E–009	6.95E–014
F7	Mean	**5.35E–004**	1.90E–003	1.55E–002	3.26E–002	1.35E–002	1.27E–002	1.53E–002	1.93E–002
	SD	1.40E–003	3.80E–003	1.55E–002	2.26E–002	1.40E–002	7.70E–003	1.55E–002	1.92E–002
	Best	9.55E–010	2.51E–009	0.00E+000	0.00E+000	0.00E+000	3.97E–006	5.55E–016	0.00E+000
F8	Mean	2.09E+001	2.09E+001	2.09E+001	2.10E+001	**2.08E+001**	2.09E+001	2.10E+001	2.09E+001
	SD	5.06E–002	4.57E–002	3.98E–002	5.53E–002	5.48E–002	5.28E–002	4.03E–002	4.74E–002
	Best	2.08E+001	2.08E+001	2.09E+001	2.07E+001	2.07E+001	2.08E+001	2.09E+001	2.08E+001
F9	Mean	7.38E+001	3.97E+001	4.12E+001	2.01E+001	1.99E+001	8.08E+001	3.94E+001	**1.77E+001**
	SD	8.47E+000	1.27E+001	8.45E+000	1.06E+001	6.17E+000	2.54E+001	1.39E+001	5.07E+000
	Best	5.64E+001	1.99E+001	2.69E+001	3.98E+000	9.95E+000	2.51E+001	1.49E+001	6.96E+000
F10	Mean	1.61E+002	**6.00E+001**	7.51E+001	1.84E+002	1.10E+002	1.37E+002	7.40E+001	6.97E+001
	SD	4.11E+001	1.67E+001	1.87E+001	5.31E+001	3.41E+001	1.05E+001	2.52E+001	1.24E+001
	Best	7.17E+001	2.59E+001	4.28E+001	9.05E+001	5.95E+001	1.10E+002	2.79E+001	4.88E+001
F11	Mean	2.89E+001	2.31E+001	2.66E+001	2.38E+001	2.56E+001	3.51E+001	2.86E+001	**2.28E+001**
	SD	1.98E+000	4.44E+000	3.28E+000	4.30E+000	2.55E+000	1.68E+000	3.52E+000	3.83E+000
	Best	2.27E+001	1.22E+001	1.73E+001	1.38E+001	2.12E+001	3.05E+001	1.85E+001	1.48E+001
F12	Mean	1.13E+004	3.78E+003	4.64E+003	2.03E+003	7.51E+003	7.61E+004	6.48E+003	**1.40E+003**
	SD	6.92E+003	5.76E+003	4.31E+003	2.56E+003	8.49E+003	5.44E+004	7.70E+003	1.56E+003
	Best	1.80E+003	1.58E+000	2.98E+001	1.55E–002	1.07E+000	4.54E+003	9.09E+000	6.10E–003
F13	Mean	5.81E+000	4.46E+000	3.67E+000	4.87E+000	3.30E+000	1.12E+001	**3.29E+000**	4.18E+000
	SD	1.25E+000	1.98E+000	1.11E+000	2.06E+000	1.02E+000	1.27E+000	1.40E+000	1.61E+000
	Best	3.61E+000	2.16E+000	1.67E+000	1.97E+000	1.70E+000	7.71E+000	1.71E+000	1.97E+000
F14	Mean	1.31E+001	1.29E+001	1.26E+001	1.29E+001	1.24E+001	1.34E+001	**1.22E+001**	1.27E+001
	SD	2.33E–001	2.67E–001	3.64E–001	2.45E–001	3.45E–001	1.14E–001	6.83E–001	3.35E–001
	Best	1.25E+001	1.24E+001	1.18E+001	1.20E+001	1.17E+001	1.31E+001	1.05E+001	1.18E+001

use of the less number of function evaluations in handling f8 – f11. In short, for the first test suite with 30 dimensions, NACS is the best method among these CS variants.

Seen from **Table 6**, with the aid of the solution quality, NACS obtains the lowest mean values for 5 out of 11 test problems. Especially for the multimode functions, the overall performance of NACS is better than others. Similarly, CSPSO performs best on f1, f3 and f5, HSA-CS is superior to others on f6 and f7, and PSCS performs well on f8 and f10. Also, compared with other algorithms, CS does not work for these functions, and it gets trapped in local minima for them. Due to the adoption of adaptive strategy, ACS algorithm weakens the sensitivity of parameter setting to different problems and achieves good overall performance. Besides, OCS-LC and QCCS perform better than CS algorithm, while they have the limitation of convergence instability. In summary, NACS can acquire promising solutions for these benchmark test problems at 50*D*.

D. COMPARISON RESULTS ON THE SECOND TEST SUITE

In this subsection, the proposed NACS algorithm is further tested on the second test suite, which consists of 14 benchmark functions. Among them, F1 – F5 are unimodal problems, F6 – F12 are basic multimodal problems, and F13 – F14 are expanded multimodal problems. Besides, F1 and F9 are separable functions, and the others are non-separable problems. Because of the interrelationship among variables, it is much more difficult to solve non-separable problems than separable ones. Obviously, it is very difficult to find the optimal solutions of these functions, so they can further test the effectiveness of NACS algorithm. For this test suite with 30 dimensions, the comparison results of these involved algorithms are reported in **Table 7**, and the evolutionary curves for some test problems are shown in **Figure 3**. In addition, the scalability study is also carried out on them with 50 dimensions, and the experimental results are provided in **Table 8** .

TABLE 8. Numerical results of CS, ACS, CSPSO, HSA-CS, OCS-LC, PSCS, QCCS and NACS on the second test suite at 50*D*.

No.	Term	CS	ACS	CSPSO	HSA-CS	OCS-LC	PSCS	QCCS	NACS
F1	Mean	5.40E–008	7.37E–019	2.08E–027	**0.00E+000**	1.82E–028	1.35E–029	2.86E–029	1.68E–029
	SD	6.17E–008	7.17E–019	1.17E–027	0.00E+000	1.95E–028	5.12E–029	6.98E–029	5.19E–029
	Best	6.96E–009	6.31E–020	5.71E–028	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
F2	Mean	2.62E+000	7.24E+002	3.42E+002	4.60E–005	**2.17E–005**	4.02E+004	1.60E–004	4.53E–005
	SD	2.00E+000	2.84E+003	1.87E+003	4.41E–005	1.43E–005	9.03E+003	1.74E–004	5.68E–005
	Best	1.04E–001	6.90E–001	3.63E–005	6.07E–006	3.57E–006	2.53E+004	1.02E–005	8.19E–006
F3	Mean	3.24E+007	1.53E+007	1.24E+007	**3.59E+005**	1.90E+007	1.30E+008	2.31E+007	9.46E+006
	SD	2.11E+007	7.46E+006	6.92E+006	2.32E+005	1.54E+007	4.62E+007	2.36E+007	4.72E+006
	Best	9.20E+006	5.10E+006	2.36E+006	9.74E+004	2.28E+006	4.82E+007	3.71E+005	2.54E+006
F4	Mean	2.83E+004	1.83E+004	9.85E+003	2.36E+003	1.72E+004	9.49E+004	1.98E+004	**1.79E+003**
	SD	7.62E+003	7.17E+003	4.66E+003	1.92E+003	7.89E+003	2.90E+004	7.65E+003	1.57E+003
	Best	1.66E+004	5.37E+003	1.77E+003	3.00E+002	4.85E+003	5.32E+004	9.83E+003	1.69E+002
F5	Mean	1.03E+004	7.47E+003	7.39E+003	5.31E+003	9.34E+003	6.80E+003	1.17E+004	**4.44E+003**
	SD	1.99E+003	9.99E+002	1.41E+003	1.40E+003	2.17E+003	1.38E+003	1.93E+003	1.01E+003
	Best	6.15E+003	5.87E+003	4.69E+003	3.44E+003	5.54E+003	4.40E+003	8.31E+003	2.57E+003
F6	Mean	2.61E+002	4.90E+001	**3.33E+000**	6.98E+000	6.32E+000	7.03E+001	3.61E+000	1.19E+001
	SD	9.50E+002	2.51E+001	5.45E+000	1.73E+001	6.79E+000	4.22E+001	4.73E+000	2.19E+001
	Best	9.86E+000	3.09E+001	3.52E–015	1.90E–009	9.95E–011	3.38E+001	2.50E–006	1.13E–008
F7	Mean	8.40E–003	5.80E–003	8.80E–003	1.03E–002	7.20E–003	**4.40E–003**	6.80E–003	8.30E–003
	SD	7.60E–003	8.80E–003	1.26E–002	1.50E–002	1.06E–002	8.00E–003	1.05E–002	1.06E–002
	Best	6.56E–006	1.23E–007	1.11E–015	1.11E–016	1.55E–015	2.58E–005	2.78E–015	3.33E–016
F8	Mean	2.11E+001	2.11E+001	2.11E+001	2.11E+001	**2.10E+001**	2.11E+001	2.11E+001	2.11E+001
	SD	4.91E–002	5.01E–002	2.47E–002	7.89E–002	1.04E–001	3.68E–002	6.60E–002	3.14E–002
	Best	2.09E+001	2.10E+001	2.11E+001	2.07E+001	2.06E+001	2.11E+001	2.09E+001	2.11E+001
F9	Mean	1.65E+002	7.71E+001	1.13E+002	**2.58E+001**	4.48E+001	1.83E+002	1.01E+002	5.56E+001
	SD	1.96E+001	4.17E+001	2.64E+001	1.12E+001	1.69E+001	5.08E+001	2.76E+001	1.25E+001
	Best	1.14E+002	4.48E+001	6.67E+001	8.95E+000	1.69E+001	6.06E+001	5.31E+001	3.58E+001
F10	Mean	4.69E+002	**1.22E+002**	1.82E+002	4.26E+002	2.97E+002	2.81E+002	2.11E+002	1.74E+002
	SD	9.96E+001	2.12E+001	4.02E+001	7.71E+001	5.60E+001	1.86E+001	4.89E+001	3.34E+001
	Best	3.02E+002	8.06E+001	9.85E+001	2.70E+002	1.68E+002	2.53E+002	1.25E+002	1.04E+002
F11	Mean	5.29E+001	4.44E+001	4.67E+001	4.95E+001	5.11E+001	6.51E+001	5.16E+001	**4.16E+001**
	SD	2.97E+000	7.12E+000	6.24E+000	6.66E+000	5.67E+000	1.56E+000	6.62E+000	6.93E+000
	Best	4.71E+001	2.59E+001	3.00E+001	3.42E+001	3.49E+001	6.18E+001	3.92E+001	2.89E+001
F12	Mean	5.21E+004	7.77E+004	6.12E+004	**1.37E+004**	1.14E+005	8.26E+004	5.69E+004	3.21E+004
	SD	4.63E+004	7.08E+004	4.16E+004	1.36E+004	9.10E+004	9.32E+004	1.05E+005	2.42E+004
	Best	5.12E+003	6.24E+003	3.90E+003	7.56E+002	6.01E+003	1.50E+004	4.67E+002	4.30E+003
F13	Mean	1.25E+001	7.30E+000	7.62E+000	9.45E+000	7.33E+000	2.40E+001	6.65E+000	**5.99E+000**
	SD	2.55E+000	2.15E+000	2.04E+000	4.08E+000	2.03E+000	2.37E+000	1.46E+000	1.52E+000
	Best	8.74E+000	4.79E+000	3.81E+000	3.22E+000	4.32E+000	1.60E+001	4.42E+000	3.87E+000
F14	Mean	2.26E+001	2.24E+001	2.18E+001	2.24E+001	2.19E+001	2.31E+001	**2.17E+001**	2.21E+001
	SD	2.97E–001	3.00E–001	3.17E–001	2.86E–001	4.57E–001	1.55E–001	6.07E–001	4.64E–001
	Best	2.17E+001	2.17E+001	2.12E+001	2.15E+001	2.03E+001	2.27E+001	2.03E+001	2.04E+001

TABLE 9. Average rankings obtained by the friedman test on the first test suite.

Algorithm	Average ranking (30*D*)	Average ranking (50*D*)
CS	6.3636	6.5455
ACS	5.0909	4.7727
CSPSO	5.2727	4.7727
HSA-CS	3.2273	3.0455
OCS-LC	3.4545	4.0000
PSCS	5.0000	4.8182
QCCS	5.1364	5.5909
NACS	**2.4545**	**2.4545**

TABLE 10. Average rankings obtained by the friedman test on the second test suite.

Algorithm	Average ranking (30*D*)	Average ranking (50*D*)
CS	5.8929	6.5714
ACS	4.1429	4.2500
CSPSO	4.2857	4.0714
HSA-CS	4.2143	3.8214
OCS-LC	3.6429	4.0000
PSCS	6.7500	6.2857
QCCS	4.3929	4.2857
NACS	**2.6786**	**2.7143**

Apparently, the results appearing in **Table 7** indicate that the convergence performance of all algorithms is seriously affected by the coordinate shifting and/or rotation. In terms of solution quality, HSA-CS, PSCS, QCCS and NACS can converge to the optimal value on F1. Further, NACS produces lower mean values in 6 out of 14 test functions, HSA-CS and QCCS perform well in 3 out of 14 problems. Followed by OCS-LC, it surpasses others on F2 and F8. Similarly, CS, ACS, CSPSO and PSCS provide the solution with higher quality in 1 out of 14 functions, respectively. Besides, NACS is the second best on F3, F4 and F10. Therefore, we can conclude that NACS is a competitive method in comparison with other improved CS versions.

From **Figure 3**, one can observed that no algorithm has the fastest convergence speed for all test functions. With regard to HSA-CS, the mutation operator in differential evolution

TABLE 11. Numerical results of ABC, DE, GSA, PSO, BBPSO, jDE, SaDE and NACS on the first test suite at 30*D*.

No.	Term	ABC	DE	GSA	PSO	BBPSO	jDE	SaDE	NACS
f1	Mean	4.31E–097	3.48E–105	8.35E–018	**7.22E–179**	2.72E–099	1.31E–138	9.48E–105	2.33E–064
	SD	1.06E–096	1.78E–104	1.85E–018	0.00E+000	1.10E–098	3.75E–138	5.04E–104	1.00E–063
f2	Mean	1.63E–049	4.62E–057	1.53E–008	4.15E–023	1.13E–067	**9.66E–081**	1.62E–063	1.86E–039
	SD	2.52E–049	4.94E–057	1.73E–009	2.27E–022	2.30E–067	3.12E–080	1.86E–063	1.13E–039
f3	Mean	2.65E+003	2.10E–014	1.01E–002	**1.35E–014**	3.96E–001	1.05E–011	6.70E–008	5.16E–012
	SD	6.25E+002	7.12E–014	1.81E–002	6.62E–014	3.04E–001	1.85E–011	1.10E–007	8.45E–012
f4	Mean	3.23E–001	8.33E+000	**1.82E–009**	2.20E–008	3.18E–007	7.21E+000	3.51E–001	5.94E–006
	SD	1.08E–001	2.78E+000	2.75E–010	6.03E–008	4.37E–007	4.82E+000	8.88E–001	6.73E–006
f5	Mean	**1.41E–001**	1.28E+001	1.91E+001	7.39E+000	4.37E+001	3.99E–001	2.59E+001	1.42E–001
	SD	3.15E–001	8.54E+000	2.30E–001	6.04E+000	3.37E+001	1.22E+000	1.89E+001	7.27E–001
f6	Mean	**3.82E–004**	5.25E+002	9.81E+003	4.71E+003	5.09E+002	2.37E+001	2.76E+001	6.24E+002
	SD	0.00E+000	2.78E+002	5.39E+002	6.80E+002	2.31E+002	4.82E+001	5.10E+001	2.84E+002
f7	Mean	**0.00E+000**	1.40E+001	1.31E+001	4.11E+001	6.27E+000	**0.00E+000**	2.79E+000	1.19E+001
	SD	0.00E+000	5.05E+000	3.32E+000	1.26E+001	3.24E+000	0.00E+000	1.64E+000	4.14E+000
f8	Mean	3.48E–014	6.81E–015	2.36E–009	1.87E–014	7.88E–015	**5.74E–015**	3.85E–002	7.99E–015
	SD	3.93E–015	1.70E–015	3.12E–010	6.12E–015	6.49E–016	1.74E–015	2.11E–001	0.00E+000
f9	Mean	6.56E–004	2.10E–003	**0.00E+000**	1.57E–002	3.29E–004	7.39E–004	1.30E–003	**0.00E+000**
	SD	3.60E–003	4.10E–003	0.00E+000	1.29E–002	1.80E–003	3.00E–003	3.80E–003	0.00E+000
f10	Mean	**1.57E–032**	1.04E–002	5.62E–020	3.50E–003	1.04E–002	**1.57E–032**	6.90E–003	**1.57E–032**
	SD	5.57E–048	4.17E–002	1.86E–020	1.89E–002	3.16E–002	5.57E–048	2.63E–002	5.57E–048
f11	Mean	**1.35E–032**	4.50E–032	3.01E–004	1.50E–003	3.66E–004	1.40E–032	1.50E–003	**1.35E–032**
	SD	5.57E–048	1.17E–031	1.60E–003	3.80E–003	2.00E–003	1.27E–033	3.80E–003	5.57E–048
	ARV	**3.4091**	5.0455	5.0455	5.0000	4.9545	3.5909	5.4545	3.5000

and linear population reduction strategy are introduced into CS, thus it converges faster for some problems. Since the neighborhood attraction scheme is employed to guide the generation of solutions, it will undoubtedly reduce the convergence rate of NACS in dealing with complex optimization problems. Specifically, for F5, F6, F9, F11 and F12, NACS converges slower than some other CS variants in the initial stage of evolution process. Nevertheless, in the middle and later periods, NACS has a strong capability of refined search. Considering the convergence speed and solution accuracy, we can say that NACS is more promising in handling these test problems.

As described in **Table 8**, for these test problems with 50 dimensions, NACS also obtains better overall performance with the help of solution accuracy. To be specific, on the aspect of reliability, the convergence of NACS, OCS-LC, PSCS and QCCS on F1 is unstable, while HSA-CS can find the global optimal solution in all 30 runs. Besides, both NACS and HSA-CS produce the lowest average values in 4 out of 14 problems, followed by OCS-LC for 2 out of 14 ones, whereas CS cannot provide a lower mean value for any problem. Note that HSA-CS does not work for F10 and F14, while NACS also yields better results on F2, F3, F10 and F12. All in all, NACS performs well on this test suite with 50 dimensions.

E. NONPARAMETRIC STATISTICAL TEST

To further compare the performance differences of all algorithms, the non-parameter statistic Friedman test is conducted, and the average rankings of these selected algorithms on each test suite are reported in **Tables 9** and **10**, respectively. For the sake of clarity, the best ranking, that is, the lowest ranking value is marked in bold.

Obviously, with regard to the first test suite with 30 and 50 dimensions, the average rankings appearing in **Table 9** demonstrate that NACS yields the lowest ranking values. For these benchmark functions with 30 dimensions, these involved algorithms are sorted in the following order: NACS, HSA-CS, OCS-LC, PSCS, ACS, QCCS, CSPSO and CS. For these problems with 50 dimensions, NACS also provides the best ranking. Moreover, the ranking value of PSCS is slightly larger than those of ACS and CSPSO, which they obtain the same ranking.

According to the average ranking results in **Table 10**, for the second test suite with 30 dimensions, the rankings of these algorithms involved are as follows: NACS, OCS-LC, ACS, HSA-CS, CSPSO, QCCS, CS and PSCS. Apparently, PSCS is less capable of handling these complex optimization problems. Additionally, as mentioned earlier, HSA-CS and NACS obtain the lowest mean values in 4 problems when tackling these test functions with 50 dimensions. However, NACS is significantly superior to its competitors in terms of the average ranking results, followed by HSA-CS. Thus, NACS is the best scheme among these algorithms on the second test suite.

F. COMPARISON WITH OTHER EVOLUTIONARY ALGORITHMS

To further verify the superiority of NACS, a few other evolutionary algorithms are used for comparison. These optimization techniques include ABC [41], DE, GSA [42], PSO, BBPSO [43], jDE [44] and SaDE [45]. In this subsection, the population size N and problem dimension D are set to 50 and 30, respectively. For each test problem, all algorithms are run 30 times, and the maximum function evaluation is set to 300000 for each run. The numerical results produced by

TABLE 12. Numerical results of ABC, DE, GSA, PSO, BBPSO, jDE, SaDE and NACS on the second test suite at 30*D*.

No.	Term	ABC	DE	GSA	PSO	BBPSO	jDE	SaDE	NACS
F1	Mean	**0.00E+000**	2.79E–028	9.91E–018	2.44E–029	1.38E–028	8.41E–030	**0.00E+000**	**0.00E+000**
	SD	0.00E+000	2.00E–028	2.55E–018	5.23E–009	1.69E–028	3.77E–029	0.00E+000	0.00E+000
F2	Mean	4.63E+003	**7.84E–014**	2.58E+002	1.83E–013	3.91E–001	2.89E–010	3.72E–006	5.71E–011
	SD	9.94E+002	1.91E–013	6.19E+001	9.77E–013	3.34E–001	1.15E–009	4.83E–006	1.24E–010
F3	Mean	6.39E+006	1.72E+005	1.47E+006	1.03E+006	7.24E+006	1.27E+005	4.55E+005	**8.00E+004**
	SD	1.49E+006	8.50E+004	9.41E+005	6.31E+005	6.72E+006	9.64E+004	2.69E+005	1.04E+005
F4	Mean	4.34E+004	**7.64E–005**	2.25E+004	1.03E+001	1.28E+003	3.09E–001	1.51E+002	1.03E+000
	SD	1.54E+004	1.61E–004	2.80E+003	1.76E+001	9.29E+002	6.42E–001	2.59E+002	2.62E+000
F5	Mean	1.09E+004	**2.46E+002**	5.81E+003	3.23E+003	2.94E+003	1.08E+003	3.21E+003	1.12E+003
	SD	1.64E+003	2.17E+002	1.21E+003	7.80E+002	6.61E+002	6.03E+002	4.88E+002	4.78E+002
F6	Mean	4.81E+000	2.14E+001	2.83E+002	1.58E+001	6.54E+001	**1.44E–001**	4.88E+001	4.64E–001
	SD	7.66E+000	2.05E+001	2.38E+002	2.28E+001	4.87E+001	7.28E–001	3.22E+001	1.21E+000
F7	Mean	**1.54E–002**	2.20E–002	2.94E+003	2.36E–002	2.18E–002	1.82E–002	2.17E–002	1.93E–002
	SD	3.90E–003	1.68E–002	1.40E+002	1.95E–002	1.61E–002	1.50E–002	2.04E–002	1.92E–002
F8	Mean	2.07E+001	2.11E+001	**2.00E+001**	2.09E+001	2.09E+001	2.11E+001	2.09E+001	2.09E+001
	SD	6.61E–002	5.43E–002	2.72E–002	7.32E–002	6.00E–002	4.73E–002	5.03E–002	4.74E–002
F9	Mean	**0.00E+000**	1.41E+001	2.94E+001	7.00E+001	6.73E+000	5.92E–017	9.95E–002	1.77E+001
	SD	0.00E+000	4.14E+000	5.43E+000	1.72E+001	3.55E+000	3.24E–016	3.04E–001	5.07E+000
F10	Mean	3.58E+002	1.47E+002	**2.23E+001**	8.86E+001	1.47E+002	9.46E+001	4.61E+001	6.97E+001
	SD	5.01E+001	7.47E+001	5.90E+000	2.76E+001	1.95E+001	4.75E+001	8.99E+000	1.24E+001
F11	Mean	2.96E+001	2.34E+001	**5.33E–002**	1.78E+001	3.27E+001	3.63E+001	1.63E+001	2.28E+001
	SD	2.73E+000	1.59E+001	2.88E–001	2.94E+000	5.65E+000	7.29E+000	2.58E+000	3.83E+000
F12	Mean	6.37E+003	5.65E+003	**1.17E–001**	3.13E+004	9.45E+003	1.66E+004	5.10E+003	1.40E+003
	SD	2.45E+003	7.97E+003	3.85E–001	3.57E+004	6.64E+003	2.57E+004	7.18E+003	1.56E+003
F13	Mean	**9.90E–001**	4.45E+000	4.64E+000	3.17E+000	1.98E+000	3.51E+000	3.84E+000	4.18E+000
	SD	1.19E–001	4.22E+000	1.10E+000	7.04E–001	3.98E–001	1.75E+000	4.66E–001	1.61E+000
F14	Mean	1.31E+001	1.36E+001	1.40E+001	**1.21E+001**	1.30E+001	1.38E+001	1.27E+001	1.27E+001
	SD	2.45E–001	1.56E–001	2.50E–001	5.38E–001	3.18E–001	2.05E–001	2.13E–001	3.35E–001
	ARV	4.8571	4.5714	5.3214	4.6071	5.3214	4.0000	3.9643	**3.3571**

these algorithms in terms of the mean and standard deviation of the function error are reported in **Tables 11** and **12**. To compare the performance of all algorithms, the Friedman test is conducted, and the average ranking value (ARV) is also presented in these comparative results.

According to **Table 11**, the proposed algorithm finds the optimum values on f9, f10 and f11. Similarly, ABC performs well on f5, f6, f7, f10 and f11, GSA provides better solutions on f4 and f9, PSO surpasses others on f1 and f3, and jDE does well in handling f2, f7, f8 and f10. DE, BBPSO and SaDE are failed to produce best result for any problem. Further, in terms of the average ranking results, NACS yields the average ranking value of 3.5, which is closely successful with ABC in tackling these classical functions.

From **Table 12**, one can observed that these algorithms seem to be good at tackling different benchmark functions in terms of solution quality. Specifically, both ABC and GSA perform well on 4 problems, DE is superior to others in 3 cases, PSO performs best on F14, jDE beats other algorithms on F6, SaDE produces the optimum solution on F1, and NACS provides better results on F1 and F3. However, the average rankings obtained by Friedman test indicate that NACS exhibits the best optimization performance. To sum up, compared with other optimization techniques, NACS is efficient and effective optimizer for these two test suites.

As discussed above, for these benchmark functions from two different test sets, the proposed NACS algorithm can achieve fruitful results in comparison with other CS variants and several popular evolutionary algorithms. The main reason is the combination of neighborhood attraction scheme and Levy flight to guide the search behavior of individuals. However, there are some limitations in NACS algorithm. Since each individual is only attracted by the best solution found so far in the neighborhood rather than the entire population, it may reduce the convergence speed. Moreover, to further enhance the convergence performance of the proposed algorithm, it is also a promising research direction to regulate the switching parameter using an adaptive control scheme.

VI. CONCLUSION

In this paper, a new CS with neighborhood attraction namely NACS is proposed. NACS algorithm adopts two different strategies to yield potential candidate solutions. One is the neighborhood attraction scheme, and the other is Levy flight used in the classical CS. With regard to the first scheme, each individual is attracted by the best solution in k-neighborhood to maintain the population diversity. After that, a switching parameter is defined to achieve the combination of the neighborhood attraction scheme and Levy flight. Also, according to the degree of individual evolution, the step size is adaptively adjusted in the evolutionary process, which weakens the sensitivity of step size to optimization problem being solved. To evaluate the performance of NACS algorithm, two sets of test problems with 30 and 50 dimensions are employed. Moreover, the settings of neighborhood size and switching parameter are also investigated experimentally. Numerical

results reveal that NACS is a competitive CS variant for some test problems, and the average rankings obtained by the Friedman test also show that NACS performs well compared with the other six newly-developed CS versions and seven popular evolutionary algorithms.

In the future, we intend to expand the current work in the following directions. First, since the neighborhood attraction scheme is developed based on the ring topology, other topological structures can also be studied. Next, with regard to the switching parameter, an adaptive adjustment mechanism can be tried to further enhance the search ability. Finally, we will employ the presented NACS algorithm to address some real-world problems.

APPENDIX

These 11 classical functions are given below.

Sphere function

$$f_1(x) = \sum_{i=1}^{D} x_i^2, \quad x \in [-100, 100]^D$$

Schwefel function 2.22

$$f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|, \quad x \in [-10, 10]^D$$

Schwefel function 1.2

$$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2, \quad x \in [-100, 100]^D$$

Schwefel function 2.21

$$f_4(x) = max_i \{|x_i|, 1 \le i \le D\}, \quad x \in [-100, 100]^D$$

Rosenbrock function

$$f_5(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right], \quad x \in [-30, 30]^D$$

Schwefel function 2.26

$$f_6(x) = 418.9829 \cdot D - \sum_{i=1}^{D} x_i \, sin\left(\sqrt{|x_i|}\right), \quad x \in [-500, 500]^D$$

Rastrigin function

$$f_7(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right], \quad x \in [-5.12, 5.12]^D$$

Ackley function

$$f_8(x) = -20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right), \quad x \in [-32, 32]^D$$

Griewank Function

$$f_9(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad x \in [-600, 600]^D$$

Penalized function 1

$$f_{10}(x) = \frac{\pi}{D}\left\{10\, sin(\pi y_i) + \sum_{i=1}^{D} (y_i - 1)^2 \left[1 + 10\, sin^2(\pi y_{i+1})\right] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4), \; y_i = 1 + (x_i + 1)/4, \; x \in [-50, 50]^D$$

Penalized function 2

$$f_{11}(x) = 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{D} (x_i - 1)^2 \times \left[1 + sin^2(3\pi x_{i+1})\right] + (x_D - 1)^2 \cdot [1 + sin^2(2\pi x_D)]\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4), \; x \in [-50, 50]^D$$

Abbreviation to CEC 2005 benchmark functions F1 – F14.

F1: Shifted Sphere function.

F2: Shifted Schwefel function 1.2.

F3: Shifted rotated high conditioned Elliptic function.

F4: Shifted Schwefel function 1.2 with noise.

F5: Schwefel function 2.6 with global optimum on bounds.

F6: Shifted Rosenbrock function.

F7: Shifted rotated Griewank function without bounds.

F8: Shifted rotated Ackley function with global optimum on bounds.

F9: Shifted Rastrigin function.

F10: Shifted rotated Rastrigin function.

F11: Shifted rotated Weierstrass function.

F12: Schwefel's problem 2.13.

F13: Shifted expanded Griewank's plus Rosenbrock's function.

F14: Shifted rotated expanded Scaffer's F6 function.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, Jul. 1995, pp. 1942–1948.

[2] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[3] W. K. Zang, L. Y. Ren, W. Q. Zhang, and X. Y. Liu, "A cloud model based DNA genetic algorithm for numerical optimization problems," *Future Gener. Comput. Syst.*, vol. 81, pp. 465–477, Apr. 2018.

[4] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems," *Inf. Sci.*, vol. 183, pp. 1–15, Jan. 2012.

[5] S. J. Mousavirad and H. Ebrahimpour-Komleh, "Human mental search: A new population-based metaheuristic optimization algorithm," *Appl. Intell.*, vol. 47, no. 3, pp. 850–887, 2017.

[6] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," *Swarm Evol. Comput.*, vol. 44, pp. 148–175, Feb. 2019

[7] X.-S. Yang and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 169–174, Jan. 2014.

[8] R. Salgotra, U. Singh, and S. Saha, "New cuckoo search algorithms with enhanced exploration and exploitation properties," *Expert Syst. Appl.*, vol. 95, pp. 384–420, Apr. 2018.

[9] Y. Cai, G. Sun, T. Wang, H. Tian, Y. Chen, and J. Wang, "Neighborhood-adaptive differential evolution for global numerical optimization," *Appl. Soft Comput.*, vol. 59, pp. 659–706, Sep. 2017.
[10] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proc. Conf. Evol. Comput.*, Washington, DC, USA, Jul. 1999, pp. 1931–1938.
[11] H. Chiroma, T. Herawan, I. Fister, Jr., I. Fister, S. Abdulkareem, L. Shuib, M. F. Hamza, Y. Saadi, and A. Abubakar, "Bio-inspired computation: Recent development on the modifications of the cuckoo search algorithm," *Appl. Soft Comput.*, vol. 61, pp. 149–173, Dec. 2017.
[12] S. Walton, O. Hassan, K. Morgan, and M. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons Fractals*, vol. 44, no. 9, pp. 710–718, 2011.
[13] X. Li and M. Yin, "Modified cuckoo search algorithm with self adaptive parameter method," *Inf. Sci.*, vol. 298, pp. 80–97, Mar. 2015.
[14] L. Huang, S. Ding, S. Yu, J. Wang, and K. Lu, "Chaos-enhanced Cuckoo search optimization algorithms for global optimization," *Appl. Math. Model.*, vol. 40, pp. 3860–3875, Mar. 2016.
[15] B. Yang, J. Miao, Z. C. Fan, J. Long, and X. H. Liu, "Modified cuckoo search algorithm for the optimal placement of actuators problem," *Appl. Soft Comput.*, vol. 67, pp. 48–60, Jun. 2018.
[16] G. G. Wang, A. H. Gandomi, X. J. 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.
[17] G. Kanagaraj, S. G. Ponnambalam, N. Jawahar, and M. J. Nilakantan, "An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization," *Eng. Optim.*, vol. 46, no. 10, pp. 1331–1351, Oct. 2014.
[18] W. Long, X. Liang, Y. Huang, and Y. Chen, "An effective hybrid cuckoo search algorithm for constrained global optimization," *Neural Comput. Appl.*, vol. 25, no. 3, pp. 911–926, Sep. 2014.
[19] E. Daniel, J. Anitha, and J. Gnanaraj, "Optimum laplacian wavelet mask based medical image using hybrid cuckoo search–grey wolf optimization algorithm," *Knowl.-Based Syst.*, vol. 131, pp. 58–69, Sep. 2017.
[20] J. T. Cheng, L. Wang, and Y. Xiong, "Ensemble of cuckoo search variants," *Comput. Ind. Eng.*, vol. 135, pp. 299–313, Sep. 2019.
[21] S. Kang, M. Kim, and J. Chae, "A closed loop based facility layout design using a cuckoo search algorithm," *Expert Syst. Appl.*, vol. 93, pp. 322–335, Mar. 2018.
[22] J. K. Kordestani, H. A. Firouzjaee, and M. R. Meybodi, "An adaptive bi-flight cuckoo search with variable nests for continuous dynamic optimization problems," *Appl. Intell.*, vol. 48, no. 1, pp. 97–117, 2018.
[23] E. Valian and E. Valian, "A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems," *Eng. Optim.*, vol. 45, no. 11, pp. 1273–1286, Nov. 2013.
[24] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, 2011.
[25] N. Fouladgar, M. Hasanipanah, and H. B. Amnieh, "Application of cuckoo search algorithm to estimate peak particle velocity in mine blasting," *Eng. with Comput.*, vol. 33, no. 2, pp. 181–189, 2017.
[26] D. Laha and N. D. Jatinder Gupta, "An improved cuckoo search algorithm for scheduling jobs on identical parallel machines," *Comput. Ind. Eng.*, vol. 126, pp. 348–360, Dec. 2018.
[27] A. Majumder, D. Laha, and P. N. Suganthan, "A hybrid cuckoo search algorithm in parallel batch processing machines with unequal job ready times," *Comput. Ind. Eng.*, vol. 124, pp. 65–76, Oct. 2018.
[28] J. T. Cheng, L. Wang, and Y. Xiong, "Cuckoo search algorithm with memory and the vibrant fault diagnosis for hydroelectric generating unit," *Eng. Comput.*, vol. 35, no. 2, pp. 687–702, Apr. 2019.
[29] Z. Hu, X. Cai, and Z. Fan, "An improved memetic algorithm using ring neighborhood topology for constrained optimization," *Soft Comput.*, vol. 18, no. 10, pp. 2023–2041, Oct. 2014.
[30] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.
[31] H. Wang, W. J. Wang, X. Y. Zhou, H. Sun, J. Zhao, X. Yu, and Z. H. Cui, "Firefly algorithm with neighborhood attraction," *Inf. Sci.*, vols. 382–383, pp. 374–387, Mar. 2017.
[32] J. T. Cheng, L. Wang, Q. Y. Jiang, Z. J. Cao, and Y. Xiong, "Cuckoo search algorithm with dynamic feedback information," *Future Gener. Comput. Syst.*, vol. 89, pp. 317–334, Dec. 2018.
[33] M. A. Al-Betar, M. A. Awadallah, H. Faris, X.-S. Yang, A. T. Khader, and O. A. Alomari, "Bat-inspired algorithms with natural selection mechanisms for global optimization," *Neurocomputing*, vol. 273, pp. 448–465, Jan. 2018.
[34] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Kanpur Genetic Algorithms Lab., Nanyang Technol. Univ., Singapore, KanGAL Rep. 2005005, May 2005.
[35] S. K. Sarangi, R. Panda, P. K. Das, and A. Abraham, "Design of optimal high pass and band stop FIR filters using adaptive cuckoo search algorithm," *Eng. Appl. Artif. Intell.*, vol. 70, pp. 67–80, Apr. 2018.
[36] R. Chi, Y. X. Su, D. H. Zhang, X. X. Chi, and H. J. Zhang, "A hybridization of cuckoo search and particle swarm optimization for solving optimization problems," *Neural Comput. Appl.*, vol. 31, pp. 653–670, Jan. 2019.
[37] U. Mlakar, I. Fister, Jr., and I. Fister, "Hybrid self-adaptive cuckoo search for global optimization," *Swarm Evol. Comput.*, vol. 29, pp. 47–72, Aug. 2016.
[38] X. Li and M. Yin, "A particle swarm inspired cuckoo search algorithm for real parameter optimization," *Soft Comput.*, vol. 20, no. 4, pp. 1389–1413, 2016.
[39] Z. Cui, B. Sun, G. Wang, Y. Xue, and J. Chen, "A novel oriented cuckoo search algorithm to improve DV–Hop performance for cyber–physical systems," *J. Parallel Distrib. Comput.*, vol. 103, pp. 42–52, May 2017.
[40] S. I. Boushaki, N. Kamel, and O. Bendjeghaba, "A new quantum chaotic cuckoo search algorithm for data clustering," *Expert Syst. Appl.*, vol. 96, pp. 358–372, Apr. 2018.
[41] 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, Apr. 2007.
[42] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009.
[43] H. Liu, G. Y. Ding, and B. Wang, "Bare-bones particle swarm optimization with disruption operator," *Appl. Math. Comput.*, vol. 238, pp. 106–122, Jul. 2014.
[44] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.
[45] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

JIATANG CHENG received the B.S. and M.S. degrees in automation from the Kunming University of Science and Technology, Kunming, China, in 2000 and 2007, respectively. He is currently pursuing the Ph.D. degree in pattern recognition and intelligent systems with the Faculty of Computer Science and Engineering, Xi'an University of Technology, Xi'an, China. His current research interests include evolutionary algorithms, neural networks, and multi-objective optimization.

LEI WANG received the B.S. and M.S. degrees in computer science and technology from the Xi'an University of Technology, Xi'an, China, in 1994 and 1997, respectively, and the Ph.D. degree in electronic science and technology from Xidian University, Xi'an, in 2001. He is currently a Professor with the Faculty of Computer Science and Engineering, Xi'an University of Technology. His current research interests include evolutionary algorithms, neural networks, and data mining.