Research Article

Cuckoo Search Algorithm with Chaotic Maps

Lijin Wang and Yiwen Zhong

College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou 350002, China

Correspondence should be addressed to Lijin Wang; lijinwang@fafu.edu.cn

Received 5 March 2015; Revised 25 June 2015; Accepted 28 June 2015

Academic Editor: Evangelos J. Sapountzakis

Copyright © 2015 L. Wang and Y. Zhong. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cuckoo search algorithm is a novel nature-inspired optimization technique based on the obligate brood parasitic behavior of some cuckoo species. It iteratively employs Lévy flights random walk with a scaling factor and biased/selective random walk with a fraction probability. Unfortunately, these two parameters are used in constant value schema, resulting in a problem sensitive to solution quality and convergence speed. In this paper, we proposed a variable value schema cuckoo search algorithm with chaotic maps, called CCS. In CCS, chaotic maps are utilized to, respectively, define the scaling factor and the fraction probability to enhance the solution quality and convergence speed. Extensive experiments with different chaotic maps demonstrate the improvement in efficiency and effectiveness.

1. Introduction

Cuckoo search algorithm (CS) is a novel nature-inspired approach based on the obligate brood parasitic behavior of some cuckoo species in combination with the Lévy flights behavior of some birds and fruit flies [1, 2]. Subsequent investigations [2, 3] have demonstrated that CS is a simple yet very promising population-based stochastic search technique by using Lévy flights random walk (LFRW) and biased/selective random walk (BSRW). LFRW with a scaling factor parameter uses a mutation operator to generate new solutions based on a best solution obtained so far, while BSRW with a fraction probability parameter employs a complex crossover operator to search new solutions. After each random walk, a greedy strategy is utilized to select a better solution from the current and new generated solutions according to their fitness.

Due to its promising performance, CS has received much attention. Some studies have focused on improving LFRW [4–10] and BSRW [11–15]. Some attempts have been made to combine CS with other optimization techniques like particle swarm optimization [16, 17], Tabu search [18], differential evolution [19], ant colony optimization [20], and cooperative coevolutionary framework [21, 22]. The above studies have shown their contribution to the research on CS. Except for the literatures [9, 10], however, these studies used the

definition of the scaling factor and the fraction probability in the constant value way, resulting in making CS sensitive to the optimization problems. This motivates us to study the scaling factor and the fraction probability using the variable value schema.

One of the mathematical approaches for the variable value schema is chaos. Chaos theory is related to the study of chaotic dynamical systems that are highly sensitive to the initial conditions [23]. Recently, chaos theory has been integrated into genetic algorithm [24], differential evolution [25], firefly algorithm [26], krill herd [27, 28], and biogeography-based optimization [23, 29], and these have shown the effectiveness and efficiency of chaos theory. In light of the above investigations, we propose chaotic cuckoo search algorithm, called CCS, which utilizes chaotic maps to define the scaling factor and the fraction probability. The comprehensive experiments are carried out on 20 benchmark functions, and the results show that chaotic maps can improve the solution quality and convergence speed of CS effectively and efficiently.

The main contribution of this paper is to define the variable value for the scaling factor and the fraction probability using chaotic maps. This leads to the major advantages of our approach as follows: (i) since the scaling factor and the fraction probability are used in constant value way, the

variable value schema of two parameters is generally more suitable for the optimization problems, resulting in better performance; (ii) due to the simpleness of chaotic maps, our approach does not increase the overall complexity of CS; (iii) our approach does not destroy the structure of CS; thus, it is still very simple.

The remainder of this paper is organized as follows. Section 2 describes the standard cuckoo search algorithm. Section 3 presents the cuckoo search algorithm with chaos. Section 4 reports the experimental results. Section 5 draws conclusion on this paper.

2. Cuckoo Search Algorithm

CS, developed recently by Yang and Deb [1, 2], is a simple yet very promising population-based stochastic search technique. In general, when CS is used to solve an objective function $f(x)$ with the solution space $[x_{j,\min}, x_{j,\max}]$, $j = 1, 2, \dots, D$, a nest represents a candidate solution $X = (x_1, \dots, x_D)$.

In the initialization phase, CS initializes solutions that are randomly sampled from solution space by

$$x_{i,j,0} = x_{i,j,\min} + r(x_{i,j,\max} - x_{i,j,\min}), \quad (1)$$

$$i = 1, 2, \dots, NP,$$

where r represents a uniformly distributed random variable on the range $[0, 1]$ and NP is the population size.

After initialization, CS goes into an iterative phase where two random walks: Lévy flights random walk and biased/selective random walk, are employed to search for new solutions. After each random walk, CS selects a better solution according to the new generated and current solutions fitness using the greedy strategy. At the end of each iteration process, the best solution is updated.

2.1. Lévy Flights Random Walk. Broadly speaking, LFRW is a random walk whose step-size is drawn from Lévy distribution. At generation G ($G > 0$), LFRW can be formulated as follows:

$$X_{i,G+1} = X_{i,G} + \alpha \oplus \text{Lévy}(\beta), \quad (2)$$

where α is a step-size which is related to the scales of the problem. In CS, LFRW is employed to search for new solutions around the best solution obtained so far. Therefore, the step-size can be obtained by the following equation [2]:

$$\alpha = \alpha_0 \times (X_{i,G} - X_{\text{best}}), \quad (3)$$

where α_0 is a scaling factor (generally, $\alpha_0 = 0.01$) and X_{best} represents the best solution obtained so far.

The product $\oplus$ means entry-wise multiplications. $\text{Lévy}(\beta)$ is a random number, which is drawn from a Lévy distribution for large steps:

$$\text{Lévy}(\beta) \sim u = t^{-1-\beta}, \quad 0 < \beta \leq 2. \quad (4)$$

In implementation, $\text{Lévy}(\beta)$ can be calculated as follows [2]:

$$\text{Lévy}(\beta) \sim \frac{\phi \times u}{|v|^{1/\beta}}, \quad (5)$$

$$\phi = \left(\frac{\Gamma(1+\beta) \times \sin(\pi \times \beta/2)}{\Gamma((1+\beta)/2 \times \beta \times 2^{(\beta-1)/2})} \right)^{1/\beta},$$

where β is a constant and set to 1.5 in the standard software implementation of CS [2], u and v are random numbers drawn from a normal distribution with mean of 0 and standard deviation of 1, and Γ is a *gamma function*.

Obviously, (2) can be reformulated as

$$X_{i,G+1} = X_{i,G} + \alpha_0 \frac{\phi \times u}{|v|^{1/\beta}} (X_{i,G} - X_{\text{best}}). \quad (6)$$

2.2. Biased/Selective Random Walk. BSRW is used to discover new solutions far enough away from the current best solution by far field randomization [1]. First, a trial solution is built with a mutation of the current solution as base vector and two randomly selected solutions as perturbed vectors. Second, a new solution is generated by a crossover operator from the current and the trial solutions. BSRW can be formulated as follows:

$$x_{i,j,G+1} = \begin{cases} x_{i,j,G} + r(x_{m,j,G} - x_{n,j,G}), & \text{if } ra > p_a \\ x_{i,j,G}, & \text{otherwise,} \end{cases} \quad (7)$$

where the random indexes m and n are the m th and n th solutions in the population, respectively, j is the j th dimension of the solution, r and ra are random numbers on the range $[0, 1]$, and p_a is a fraction probability.

3. Chaotic Cuckoo Search Algorithm

In this section, we first present different chaotic maps. Then, we apply them to define the scaling factor and the fraction probability. We last propose the framework of cuckoo search algorithm with chaotic maps, called CCS.

3.1. Chaotic Maps. Chaos theory is a field of study in mathematics, with applications in several disciplines including physics, engineering, economics, biology, and philosophy. Chaos theory studies the behavior of dynamical systems that are highly sensitive to initial conditions, an effect which is popularly referred to as the butterfly effect. One of ways to make quantitative statements about the behavior of chaotic systems is chaotic map like Circle map [30], Gauss map [30], Logistic map [31], Piecewise map [32], Sine map [33], Singer map [34], Sinusoidal map [31], and Tent map [35], shown in Table 1. Additionally, the visualization of these chaotic maps with the initial point at 0.7 is plotted in Figure 1. The other chaotic maps can be found in [26, 28].

3.2. Chaotic Maps for the Scaling Factor. As seen from (6), the large scaling factor does not fit the problems with the narrow

TABLE 1: Chaotic maps.

Number	Name	Chaotic map	Range
1	Circle	$x_{i+1} = \text{mod} \left(x_i + b - \left(\frac{a}{2\pi} \right) \sin(2\pi x_k), 1 \right), a = 0.5 \text{ and } b = 0.2$	(0, 1)
2	Gauss	$x_{i+1} = \begin{cases} 1 & x_i = 0 \\ \frac{1}{\text{mod}(x_i, 1)} & \text{otherwise} \end{cases}$	(0, 1)
3	Logistic	$x_{i+1} = ax_i(1 - x_i), a = 4$	(0, 1)
4	Piecewise	$x_{i+1} = \begin{cases} \frac{x_i}{p} & 0 \leq x_i < p \\ \frac{\frac{x_i}{p} - p}{0.5} - p & p \leq x_i < 0.5 \\ \frac{1 - x_i - p}{0.5} - p & 0.5 \leq x_i < 1 - p \\ \frac{1 - x_i}{p} & 1 - p \leq x_i < 1 \end{cases}, p = 0.4$	(0, 1)
5	Sine	$x_{i+1} = \frac{a}{4} \sin(\pi x_i), a = 4$	(0, 1)
6	Singer	$x_{i+1} = \mu(7.86x_i - 23.31x_i^2 + 28.75x_i^3 - 13.302875x_i^4), \mu = 1.07$	(0, 1)
7	Sinusoidal	$x_{i+1} = ax_i^2 \sin(\pi x_i), a = 2.3$	(0, 1)
8	Tent	$x_{i+1} = \begin{cases} \frac{x_i}{0.7} & x_i < 0.7 \\ \frac{10(1 - x_i)}{3} & x_i \geq 0.7 \end{cases}$	(0, 1)

search space because it may make Lévy flights random walk become too aggressive and then jump outside of the search domain, resulting in wasting of function evaluations. In addition, for the wide search space, the small scaling factor cannot make contribution to the efficiency of search. Obviously, utilizing the constant value scaling factor is not more optimum for the problems. Therefore, we employ the chaotic maps to provide the chaotic behaviors for cuckoo search to define the scaling factor and rewrite (6) as follows:

$$X_{i,G+1} = X_{i,G} + c_1 \frac{\phi \times u}{|v|^{1/\beta}} (X_{i,G} - X_{\text{best}}), \quad (8)$$

where c_1 is a chaotic sequence.

3.3. Chaotic Maps for the Fraction Probability. In (7), the fraction probability p_a is used to control how many dimensions in expectation are changed in a solution. For low values of p_a , a large number of dimensions of a solution are changed in each generation. In this case, it is in favor of the exploration of CS. On the other hand, high values of p_a cause most of the directions of the new solution to be inherited from itself. This is beneficial to the exploitation of CS. Apparently, a variable value p_a can dynamically balance the exploration and exploitation. Thus, we utilize chaotic maps to define the fraction probability p_a to balance the exploration and exploitation and rewrite (7) as follows:

$$x_{i,j,G+1} = \begin{cases} x_{i,j,G} + r(x_{m,j,G} - x_{n,j,G}), & \text{if } ra > c_2 \\ x_{i,j,G}, & \text{otherwise,} \end{cases} \quad (9)$$

where c_2 is a chaotic sequence.

```

G ← 0;
Nest0 = (Xi,0, ..., XNP,0) ← InitializeSolution();
Fitness ← Evaluation(Nest0);
FES ← NP;
BestX ← FindBestSolutionByFitness();
c1 ← InitializeChaoticSequence();
c2 ← InitializeChaoticSequence();
WHILE (FES < MaxFES)
    G ← G + 1;
    c1 ← InitializeChaoticSequence();
    c2 ← InitializeChaoticSequence();
    FOR (i from 1 to NP)
        newXi,G ← Generating new solution with (8)
        Xi,G ← EvaluatingAndSelecting(newXi,G, Xi,G);
        FES ← FES + 1;
    ENDFOR
    FOR (i from 1 to NP)
        newXi,G ← Generating new solution with (9)
        Xi,G ← EvaluatingAndSelecting(newXi,G, Xi,G);
        FES ← FES + 1;
    ENDFOR
    BestX ← FindAndUpdateBestSolution();
ENDWHILE

```

ALGORITHM 1: CCS.

3.4. Framework of CCS. According to the above descriptions, we give the framework of CCS in Algorithm 1.

4. Simulation and Results

In this section, a suit of 20 benchmark functions used in [36] is utilized to verify the performance of the proposed

FIGURE 1: Visualization of different chaotic maps.

TABLE 2: Average Error values obtained by CCS with different chaotic maps for 20 benchmark functions at $D = 30$.

	cCCS	gCCS	lgCCS	pCCS	seCCS	srCCS	slCCS	tCCS
F_{sph}	5.08e - 55	3.88E - 46	7.36E - 48	6.33E - 52	3.06E - 48	9.10E - 47	4.38E - 44	3.16E - 52
F_{ros}	2.12e - 01	6.24E - 01	6.15E - 01	6.89E - 01	7.74E - 01	5.46E - 01	8.38E - 01	3.16E - 01
F_{ack}	8.24e - 15	1.11E - 14	8.95E - 15	7.82E - 15	7.25E - 15	7.53E - 15	7.11E - 15	8.67E - 15
F_{grw}	0.00e + 00	0.00E + 00	0.00E + 00	2.96E - 04	0.00E + 00	0.00E + 00	0.00E + 00	0.00E + 00
F_{ras}	1.48e + 01	1.70E + 01	1.59E + 01	1.21E + 01	1.53E + 01	1.43E + 01	1.43E + 01	1.72E + 01
F_{sch}	7.65e + 02	1.09E + 03	8.56E + 02	7.62E + 02	8.17E + 02	6.75E + 02	7.50E + 02	6.99E + 02
F_{sal}	2.64e - 01	3.54E - 01	2.84E - 01	2.64E - 01	3.04E - 01	2.44E - 01	2.32E - 01	2.68E - 01
F_{wht}	1.76e + 02	2.10E + 02	2.14E + 02	2.06E + 02	1.88E + 02	2.06E + 02	2.19E + 02	1.82E + 02
F_{pn1}	1.57e - 32	4.15E - 03	1.57E - 32	1.57E - 32	4.15E - 03	1.57E - 32	1.57E - 32	4.15E - 03
F_{pn2}	1.35E - 32	1.35E - 32	1.35E - 32	1.35E - 32	1.35E - 32	1.35E - 32	4.39E - 04	1.35E - 32
F_1	0.00E + 00	0.00E + 00	0.00E + 00	8.08E - 30	2.02E - 30	0.00E + 00	0.00E + 00	0.00E + 00
F_2	5.34e - 09	1.97E - 08	3.07E - 08	6.49E - 09	9.82E - 09	7.30E - 08	1.44E - 07	9.81E - 09
F_3	8.89e + 04	7.92E + 04	8.36E + 04	8.66E + 04	7.99E + 04	1.08E + 05	1.35E + 05	8.34E + 04
F_4	3.20e + 01	6.52E + 01	4.53E + 01	3.59E + 01	3.01E + 01	3.00E + 01	2.78E + 01	3.58E + 01
F_5	6.84e + 02	5.34E + 02	5.40E + 02	6.39E + 02	6.34E + 02	6.24E + 02	7.74E + 02	5.71E + 02
F_6	9.31e - 01	2.16E + 00	6.32E - 01	7.99E - 01	7.78E - 01	1.57E + 00	2.05E + 00	8.64E - 01
F_7	5.71e - 03	6.60E - 03	6.89E - 03	3.45E - 03	4.72E - 03	3.75E - 03	2.95E - 03	6.01E - 03
F_8	2.09e + 01	2.09E + 01	2.09E + 01	2.09E + 01	2.08E + 01	2.09E + 01	2.09E + 01	2.10E + 01
F_9	1.62e + 01	1.92E + 01	1.70E + 01	1.58E + 01	1.59E + 01	1.54E + 01	1.49E + 01	1.56E + 01
F_{10}	6.14e + 01	8.34E + 01	6.58E + 01	6.19E + 01	7.06E + 01	6.35E + 01	5.87E + 01	6.35E + 01

approach. These 20 benchmark functions can be divided into three groups: (i) unimodal functions including F_{sph} and F_{ros} ; (ii) multimodal functions containing F_{ack} , F_{grw} , F_{ras} , F_{sch} , F_{sal} , F_{wht} , F_{pn1} , and F_{pn2} ; and (iii) rotated and/or shifted functions F_1 – F_{10} . A more detailed description of them can be found in [36, 37]. Additionally, we use *Error*, *Evaluation*, and *Convergence graphs* as performance evaluation criteria.

Error is the function error which is defined as $(f(x) - f(x^*))$, where x^* is the global optimum of the function and x is the best solution obtained by the algorithm in a given run. In addition, *Error* is also recorded in different runs, and the average and the standard deviation of *Error* are calculated and noted as “AVG_{Er} ± STD_{Er}” used in different tables. Moreover, the Wilcoxon signed-rank test at the 0.05 significance level is used to show significance between two algorithms. The “–” symbol shows that the null hypothesis is rejected, and the first algorithm outperforms the second one. The “+” symbol means the null hypothesis is rejected and the first algorithm is inferior to the second one. The “=” symbol reveals that the null hypothesis is accepted and the first algorithm ties the second one. Additionally, the total number of each symbol “–/+/=” is summarized at the bottom of different tables.

Evaluation is the number of function evaluations needed for reaching the accuracy level $\varepsilon = 10^{-6}$ or 10^{-2} suggested in [36] within the maximum number of fitness evaluations set to $10000 \times D$, where D is the dimension of function. Furthermore, we also recorded *Evaluation* in different runs and calculate the average and standard deviation of it which are signed as “AVG_{Ev} ± STD_{Ev}(CNT),” where CNT denotes the number of successful runs in which an algorithm within the maximum number of fitness evaluations could reach the accuracy level ε .

Convergence graphs are the convergence curve graphs of each algorithm for the problems within the maximum number of fitness evaluations. These graphs show the average

TABLE 3: Average ranking of eight algorithms by the Friedman test for 20 functions at $D = 30$.

Algorithm	cCCS	gCCS	lgCCS	pCCS	seCCS	srCCS	slCCS	tCCS
Ranking	3.70	6.10	4.92	4.22	4.33	3.95	4.47	4.30

Error performance of the total runs, in respective experiments.

4.1. Sensitivities to Chaotic Maps. It can be observed from Figure 1 that different chaotic maps show different chaotic behaviors. In this section, therefore, we analyze the performance of CCS affected by different chaotic maps. To verify the sensitivity of different chaotic maps to the performance, we utilize a simple combination where different chaotic maps are employed to define the scaling factor, and the fraction probability is defined by using Gauss map according to low constant value which is used in CS. In this case, we have cCCS with Circle map, gCCS with Gauss map, lgCCS with Logistic map, pCCS with Piecewise map, seCCS with Sine map, srCCS with Singer map, slCCS with Sinusoidal map, and tCCS with Tent map. Table 2 lists the average *Error* of CCS with different chaotic maps, and Table 3 gives the results of the Friedman test similarly done in [38].

As observed from Table 2, for most functions, CCS with different chaotic maps shows similar average *Error*. However, Table 3 shows that cCCS is best, followed by srCCS, pCCS, tCCS, seCCS, slCCS, lgCCS, and gCCS. This suggests that the performance of CCS for part of functions is slightly sensitive to chaotic maps, and the combination of Cycle map and Gauss map is the better selection for cuckoo search algorithm. It is worthy saying that there are many combinations of chaotic maps to be used. Thus, in the future work, we will comprehensively test different combinations in CCS.

TABLE 4: Error obtained by rCS and CCS at $D = 30$.

	rCS			p value	CCS	
	AVG _{Er}	STD _{Er}			AVG _{Er}	STD _{Er}
F_{sph}	$5.99E - 46$	$\pm 9.45E - 46$	+	0.000012	5.08E - 55	$\pm 9.57E - 55$
F_{ros}	$1.11E + 01$	$\pm 2.49E + 00$	+	0.000012	2.12E - 01	$\pm 8.10E - 01$
F_{ack}	7.11E - 15	$\pm 0.00E + 00$	=	0.125000	$8.24E - 15$	$\pm 2.66E - 15$
F_{grw}	$0.00E + 00$	$\pm 0.00E + 00$	=	1.000000	$0.00E + 00$	$\pm 0.00E + 00$
F_{ras}	$1.48E + 01$	$\pm 3.97E + 00$	=	0.492633	$1.48E + 01$	$\pm 6.90E + 00$
F_{sch}	5.10E + 02	$\pm 2.55E + 02$	-	0.001721	$7.65E + 02$	$\pm 2.67E + 02$
F_{sal}	2.32E - 01	$\pm 4.76E - 02$	=	0.103553	$2.64E - 01$	$\pm 4.90E - 02$
F_{wht}	$2.14E + 02$	$\pm 4.38E + 01$	=	0.103553	1.76E + 02	$\pm 7.48E + 01$
F_{pn1}	$1.57E - 32$	$\pm 5.59E - 48$	=	1.000000	$1.57E - 32$	$\pm 5.59E - 48$
F_{pn2}	$1.35E - 32$	$\pm 5.59E - 48$	=	1.000000	$1.35E - 32$	$\pm 5.59E - 48$
F_1	$0.00E + 00$	$\pm 0.00E + 00$	=	1.000000	$0.00E + 00$	$\pm 0.00E + 00$
F_2	$5.73E - 02$	$\pm 4.41E - 02$	+	0.000012	5.34E - 09	$\pm 7.74E - 09$
F_3	$4.64E + 05$	$\pm 2.03E + 05$	+	0.000012	8.89E + 04	$\pm 5.33E + 04$
F_4	$6.13E + 02$	$\pm 5.31E + 02$	+	0.000012	3.20E + 01	$\pm 9.74E + 01$
F_5	$1.85E + 03$	$\pm 6.70E + 02$	+	0.000090	6.84E + 02	$\pm 6.65E + 02$
F_6	$1.42E + 01$	$\pm 4.64E + 00$	+	0.000012	9.31E - 01	$\pm 2.58E + 00$
F_7	3.67E - 03	$\pm 3.93E - 03$	=	0.427339	$5.71E - 03$	$\pm 6.58E - 03$
F_8	$2.09E + 01$	$\pm 4.38E - 02$	=	0.475825	$2.09E + 01$	$\pm 7.18E - 02$
F_9	1.28E + 01	$\pm 2.81E + 00$	-	0.011000	$1.62E + 01$	$\pm 5.19E + 00$
F_{10}	$8.84E + 01$	$\pm 1.77E + 01$	+	0.000090	6.14E + 01	$\pm 1.48E + 01$
-/+					2/10/8	

TABLE 5: Results of the multiple-problem Wilcoxon's test for CCS and rCS for 20 functions at $D = 30$.

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
CCS versus rCS	150.125	59.875	0.092059	=	+

4.2. Comparison with CS via Random Value. Note that the random value can also be regarded as the variable value schema. To show the advantage of CS with chaotic maps, CS with random value, called rCS, is tested on 20 benchmark functions at $D = 30$. In rCS, the random strategy is used to define the scaling factor and the fraction probability whose values are sampled from a uniform distribution on range between 0 and 1. Table 4 lists the statistical Error, and Table 5 reports the multiple problems statistical analysis between CCS and rCS for all functions based on the Wilcoxon test similarly done in [38, 39].

We can find from Table 4 that rCS and CCS, respectively, show their advantage on different functions. These two algorithms have the same performance on a handful of functions like F_{grw} , F_{ras} , F_{pn1} , F_{pn2} , F_1 , and F_8 . Moreover, rCS performs better on F_{ack} , F_{sch} , F_{sal} , F_7 , and F_9 , while CCS gains better performance on F_{sph} , F_{ros} , and F_{wht} , especially on rotated and/or shifted functions like F_2 , F_3 , F_4 , F_5 , F_6 , and F_{10} . According to the results of “-/+”, CCS is superior to rCS on 8 out of 20 functions, is equal to rCS on 10 out of 20 ones, and is inferior to rCS on 2 out of 2 ones.

Additionally, it can be seen from Table 5 that CCS gets higher R^+ value than R^- value. The above suggests that chaotic sequences make greater and more stable contribution to the performance of CS than random sampling sequences. This is because chaotic sequences are in fact generated

deterministically from the dynamical system, while random sampling sequences are nondeterministic and different, even if the initial state is the same.

4.3. Effect of Chaotic Maps on CS. To show how chaotic maps can improve the performance of CS, we carry out experiments on the 20 benchmark functions at $D = 10$ with population size $NP = 30$, at $D = 30$ with population size $NP = 30$, and at $D = 50$ with population size $NP = 50$, respectively, since a part of benchmark functions are defined for up to $D = 50$ [37]. CS and CCS are tested 25 times for each function, respectively. The fraction probability p_a of CS is 0.25, while the Cycle map and the Gauss map, whose initial values are 0.7 similarly done in [23, 26], are used to define the scaling factor and the fraction probability p_a , respectively. Table 6 shows Error of two algorithms at different dimensions.

Table 6 clearly shows that chaotic maps can overall significantly improve the performance of CS according to the average Error at $D = 10$, $D = 30$, and $D = 50$.

In the case of $D = 10$, observed from Table 6, CCS can gain solutions with higher accuracy for all functions except for F_8 . In terms of the Wilcoxon signed-rank test, CCS performs better on 19 out of 20 functions and shows equivalent performance to CS on 1 out of 20 ones.

In the case of $D = 30$, for unimodal functions, CCS outperforms CS significantly. For multimodal functions, CCS apparently achieves higher accurate solutions than CS does. In addition, CCS obtains the global optimal solution to F_{grw} . For rotated and/or shifted functions, CCS is not significantly inferior to CS on F_7 and equivalent to CS on F_8 . However, CCS performs better than CS for the other 8 out of 10 functions. Especially on F_1 , CCS achieves the global optima. In all, in terms of “-/+”, compared with CS, CCS,

TABLE 6: Error obtained by CS and CCS for 20 functions at $D = 10$, $D = 30$, and $D = 50$.

	$D = 10$			$D = 30$			$D = 50$		
	CS	CCS	CS	CCS	CS	CCS	CS	CCS	CCS
F_{sph}	$4.81E-26 \pm 5.99E-26$	$2.25E-59 \pm 4.30E-59$	$1.09E-30 \pm 1.35E-30$	$5.08E-55 \pm 9.57E-55$	$3.92E-17 \pm 3.31E-17$	$1.02E-26 \pm 5.70E-27$	$3.92E-17 \pm 3.31E-17$	$1.02E-26 \pm 5.70E-27$	$1.02E-26 \pm 5.70E-27$
F_{ros}	$5.22E-01 \pm 6.53E-01$	$1.34E-05 \pm 6.68E-05$	$1.68E+01 \pm 2.09E+01$	$2.12E-01 \pm 8.10E-01$	$4.46E+01 \pm 2.06E+01$	$3.82E+01 \pm 2.03E+01$	$4.46E+01 \pm 2.06E+01$	$3.82E+01 \pm 2.03E+01$	$3.82E+01 \pm 2.03E+01$
F_{ack}	$3.08E-11 \pm 4.71E-11$	$3.55E-15 \pm 0.00E+00$	$3.73E-02 \pm 1.86E-01$	$8.24E-15 \pm 2.66E-15$	$3.55E-02 \pm 1.76E-01$	$5.53E-14 \pm 1.92E-14$	$3.55E-02 \pm 1.76E-01$	$5.53E-14 \pm 1.92E-14$	$5.53E-14 \pm 1.92E-14$
F_{grw}	$3.18E-02 \pm 1.10E-02$	$1.73E-02 \pm 1.34E-02$	$9.23E-12 \pm 4.62E-11$	$0.00E+00 \pm 0.00E+00$	$2.16E-11 \pm 7.22E-11$	$0.00E+00 \pm 0.00E+00$	$2.16E-11 \pm 7.22E-11$	$0.00E+00 \pm 0.00E+00$	$0.00E+00 \pm 0.00E+00$
F_{ras}	$3.34E+00 \pm 1.23E+00$	$1.50E+00 \pm 1.34E+00$	$2.47E+01 \pm 4.87E+00$	$1.48E+01 \pm 6.90E+00$	$8.52E+01 \pm 1.25E+01$	$5.77E+01 \pm 1.77E+01$	$8.52E+01 \pm 1.25E+01$	$5.77E+01 \pm 1.77E+01$	$5.77E+01 \pm 1.77E+01$
F_{sch}	$7.46E+01 \pm 6.44E+01$	$0.00E+00 \pm 0.00E+00$	$1.44E+03 \pm 2.65E+02$	$7.65E+02 \pm 2.67E+02$	$4.93E+03 \pm 2.94E+02$	$3.00E+03 \pm 9.94E+02$	$4.93E+03 \pm 2.94E+02$	$3.00E+03 \pm 9.94E+02$	$3.00E+03 \pm 9.94E+02$
F_{sal}	$9.99E-02 \pm 2.41E-06$	$9.99E-02 \pm 1.11E-10$	$3.88E-01 \pm 8.33E-02$	$2.64E-01 \pm 4.90E-02$	$6.60E-01 \pm 7.63E-02$	$3.88E-01 \pm 4.40E-02$	$6.60E-01 \pm 7.63E-02$	$3.88E-01 \pm 4.40E-02$	$3.88E-01 \pm 4.40E-02$
F_{whit}	$2.11E+01 \pm 6.93E+00$	$1.00E+01 \pm 5.84E+00$	$3.75E+02 \pm 4.50E+01$	$1.76E+02 \pm 7.48E+01$	$1.32E+03 \pm 9.83E+01$	$9.85E+02 \pm 1.53E+02$	$1.32E+03 \pm 9.83E+01$	$9.85E+02 \pm 1.53E+02$	$9.85E+02 \pm 1.53E+02$
F_{pn1}	$5.80E-19 \pm 1.24E-18$	$4.71E-32 \pm 1.12E-47$	$5.36E-20 \pm 1.57E-19$	$1.57E-32 \pm 5.59E-48$	$1.00E-03 \pm 2.94E-03$	$2.52E-24 \pm 4.49E-24$	$1.00E-03 \pm 2.94E-03$	$2.52E-24 \pm 4.49E-24$	$2.52E-24 \pm 4.49E-24$
F_{pn2}	$3.26E-23 \pm 5.50E-23$	$1.35E-32 \pm 5.59E-48$	$1.82E-25 \pm 6.31E-25$	$1.35E-32 \pm 5.59E-48$	$5.74E-14 \pm 1.80E-13$	$1.29E-25 \pm 1.17E-25$	$5.74E-14 \pm 1.80E-13$	$1.29E-25 \pm 1.17E-25$	$1.29E-25 \pm 1.17E-25$
F_1	$4.38E-26 \pm 7.14E-26$	$0.00E+00 \pm 0.00E+00$	$5.30E-30 \pm 1.25E-29$	$0.00E+00 \pm 0.00E+00$	$1.95E-16 \pm 1.09E-16$	$2.11E-26 \pm 1.52E-26$	$1.95E-16 \pm 1.09E-16$	$2.11E-26 \pm 1.52E-26$	$2.11E-26 \pm 1.52E-26$
F_2	$1.20E-13 \pm 1.60E-13$	$1.60E-27 \pm 1.13E-27$	$6.08E-03 \pm 3.12E-03$	$5.34E-09 \pm 7.74E-09$	$2.33E+02 \pm 5.51E+01$	$3.28E-01 \pm 1.90E-01$	$2.33E+02 \pm 5.51E+01$	$3.28E-01 \pm 1.90E-01$	$3.28E-01 \pm 1.90E-01$
F_3	$2.31E+02 \pm 1.33E+02$	$1.97E-09 \pm 2.11E-09$	$2.08E+06 \pm 5.37E+05$	$8.89E+04 \pm 5.33E+04$	$8.22E+06 \pm 1.12E+06$	$3.76E+05 \pm 2.24E+05$	$8.22E+06 \pm 1.12E+06$	$3.76E+05 \pm 2.24E+05$	$3.76E+05 \pm 2.24E+05$
F_4	$1.13E-05 \pm 9.76E-06$	$9.85E-18 \pm 2.19E-17$	$1.28E+03 \pm 7.90E+02$	$3.20E+01 \pm 9.74E+01$	$2.63E+04 \pm 4.53E+03$	$5.47E+03 \pm 2.19E+03$	$2.63E+04 \pm 4.53E+03$	$5.47E+03 \pm 2.19E+03$	$5.47E+03 \pm 2.19E+03$
F_5	$1.07E-04 \pm 8.29E-05$	$2.40E-12 \pm 1.87E-12$	$2.92E+03 \pm 6.56E+02$	$6.84E+02 \pm 6.65E+02$	$1.02E+04 \pm 1.10E+03$	$3.45E+03 \pm 5.55E+02$	$1.02E+04 \pm 1.10E+03$	$3.45E+03 \pm 5.55E+02$	$3.45E+03 \pm 5.55E+02$
F_6	$1.31E+00 \pm 1.21E+00$	$5.24E-05 \pm 2.61E-04$	$1.99E+01 \pm 2.48E+01$	$9.31E-01 \pm 2.58E+00$	$6.67E+01 \pm 3.59E+01$	$3.83E+01 \pm 1.95E+01$	$6.67E+01 \pm 3.59E+01$	$3.83E+01 \pm 1.95E+01$	$3.83E+01 \pm 1.95E+01$
F_7	$4.63E-02 \pm 1.78E-02$	$2.93E-02 \pm 1.66E-02$	$1.09E-03 \pm 2.04E-03$	$5.71E-03 \pm 6.58E-03$	$2.26E-03 \pm 2.99E-03$	$8.09E-04 \pm 2.47E-03$	$2.26E-03 \pm 2.99E-03$	$8.09E-04 \pm 2.47E-03$	$8.09E-04 \pm 2.47E-03$
F_8	$2.03E+01 \pm 8.09E-02$	$2.03E+01 \pm 7.34E-02$	$2.09E+01 \pm 5.26E-02$	$2.09E+01 \pm 7.18E-02$	$2.11E+01 \pm 4.56E-02$	$2.11E+01 \pm 3.84E-02$	$2.11E+01 \pm 4.56E-02$	$2.11E+01 \pm 3.84E-02$	$2.11E+01 \pm 3.84E-02$
F_9	$2.76E+00 \pm 8.81E-01$	$8.13E-01 \pm 1.09E+00$	$2.73E+01 \pm 5.84E+00$	$1.62E+01 \pm 5.19E+00$	$1.20E+02 \pm 1.44E+01$	$7.41E+01 \pm 1.89E+01$	$1.20E+02 \pm 1.44E+01$	$7.41E+01 \pm 1.89E+01$	$7.41E+01 \pm 1.89E+01$
F_{10}	$1.90E+01 \pm 5.13E+00$	$8.42E+00 \pm 3.29E+00$	$1.58E+02 \pm 2.39E+01$	$6.14E+01 \pm 1.48E+01$	$3.94E+02 \pm 6.64E+01$	$1.38E+02 \pm 2.56E+01$	$3.94E+02 \pm 6.64E+01$	$1.38E+02 \pm 2.56E+01$	$1.38E+02 \pm 2.56E+01$
$-/=/+$		01/19		03/17		02/18

FIGURE 2: Continued.

FIGURE 2: Convergence graphs of CS and CCS.

respectively, shows better and equivalent performance on 17 and 3 out of 20 benchmark functions.

When $D = 50$, the accuracy of both solutions of two algorithms is reduced on most functions. However, compared with CS, CCS still achieves higher accurate solutions to all functions except for F_8 . In addition, CCS reaches the global optimal solution to F_{grw} . According to the statistical results, CCS outperforms CS on 18 out of 20 benchmark functions.

Furthermore, to show the convergence speed of CCS reaching the accuracy level ε , Table 7 lists the *Evaluation* performance of two algorithms at $D = 30$. Table 7 clearly shows that CCS performs the overall more stable convergence to the accuracy level ε . For example, CS and CCS both reach the accuracy level ε steadily on F_{sph} , F_{grw} , F_{pn1} , F_{pn2} , and F_1 , but CCS converges faster than CS does. Moreover, CCS has more stable convergence on F_{ros} , F_2 , and F_6 . In addition, for F_7 , although CS converges steadily to the accuracy level, CCS has faster convergence speed.

TABLE 7: Average Evaluation obtained by CS and CCS at $D = 30$.

	CS AVG _{Ev} $\pm$ STD _{Ev}	CCS AVG _{Ev} $\pm$ STD _{Ev}
F_{sph}	88868 $\pm$ 1709 (25)	52414 $\pm$ 1420 (25)
F_{ros}	292080 $\pm$ 0 (1)	268680 $\pm$ 10353 (2)
F_{ack}	166055 $\pm$ 17655 (24)	79652 $\pm$ 2091 (25)
F_{grw}	141264 $\pm$ 31378 (25)	67294 $\pm$ 26143 (25)
F_{pn1}	153843 $\pm$ 24442 (25)	59170 $\pm$ 5439 (25)
F_{pn2}	109678 $\pm$ 12077 (25)	57044 $\pm$ 2408 (25)
F_1	93476 $\pm$ 1709 (25)	53530 $\pm$ 1538 (25)
F_2	—	240891 $\pm$ 11343 (25)
F_6	—	259483 $\pm$ 49358 (14)
F_7	165087 $\pm$ 27396 (25)	101414 $\pm$ 45112 (18)

Additionally, convergence graphs of CS and CCS for some functions at $D = 30$ are plotted in Figure 2. It can be observed

TABLE 8: Error obtained by CCS₂₅, CCS₅, and CCS for 20 functions at $D = 30$.

	CCS ₂₅			CCS			CCS ₅	
	AVG _{Er} ± STD _{Er}		<i>p</i> value	AVG _{Er} ± STD _{Er}		<i>p</i> value	AVG _{Er} ± STD _{Er}	
<i>F</i> _{sph}	8.17 <i>E</i> − 56 ± 3.11 <i>E</i> − 55	−	0.000018	5.08 <i>E</i> − 55 ± 9.57 <i>E</i> − 55	+	0.000058	2.49<i>E</i> − 56 ± 4.54<i>E</i> − 56	
<i>F</i> _{ros}	3.19 <i>E</i> − 01 ± 1.10 <i>E</i> + 00	−	0.004162	2.12<i>E</i> − 01 ± 8.10<i>E</i> − 01	=	0.103553	6.40 <i>E</i> − 01 ± 1.49 <i>E</i> + 00	
<i>F</i> _{ack}	1.02 <i>E</i> − 14 ± 3.45 <i>E</i> − 15	=	0.065430	8.24<i>E</i> − 15 ± 2.66<i>E</i> − 15	=	0.138672	9.81 <i>E</i> − 15 ± 3.29 <i>E</i> − 15	
<i>F</i> _{grw}	0.00<i>E</i> + 00 ± 0.00<i>E</i> + 00	=	1.000000	0.00<i>E</i> + 00 ± 0.00<i>E</i> + 00	=	1.000000	1.33 <i>E</i> − 17 ± 6.66 <i>E</i> − 17	
<i>F</i> _{ras}	2.79 <i>E</i> + 01 ± 9.18 <i>E</i> + 00	+	0.000081	1.48<i>E</i> + 01 ± 6.90<i>E</i> + 00	−	0.000119	2.47 <i>E</i> + 01 ± 8.27 <i>E</i> + 00	
<i>F</i> _{sch}	1.25 <i>E</i> + 03 ± 3.43 <i>E</i> + 02	+	0.000266	7.65<i>E</i> + 02 ± 2.67<i>E</i> + 02	−	0.000735	1.19 <i>E</i> + 03 ± 4.07 <i>E</i> + 02	
<i>F</i> _{sal}	3.08 <i>E</i> − 01 ± 4.93 <i>E</i> − 02	+	0.004927	2.64<i>E</i> − 01 ± 4.90<i>E</i> − 02	−	0.017253	3.00 <i>E</i> − 01 ± 5.00 <i>E</i> − 02	
<i>F</i> _{wht}	2.31 <i>E</i> + 02 ± 9.18 <i>E</i> + 01	+	0.022988	1.76<i>E</i> + 02 ± 7.48<i>E</i> + 01	−	0.045010	2.23 <i>E</i> + 02 ± 6.08 <i>E</i> + 01	
<i>F</i> _{pn1}	4.15 <i>E</i> − 03 ± 2.07 <i>E</i> − 02	=	1.000000	1.57<i>E</i> − 32 ± 5.59<i>E</i> − 48	=	1.000000	1.57<i>E</i> − 32 ± 5.59<i>E</i> − 48	
<i>F</i> _{pn2}	1.37 <i>E</i> − 32 ± 4.61 <i>E</i> − 34	=	0.125000	1.35<i>E</i> − 32 ± 5.59<i>E</i> − 48	=	1.000000	1.35<i>E</i> − 32 ± 2.47<i>E</i> − 34	
<i>F</i> ₁	2.02 <i>E</i> − 30 ± 1.01 <i>E</i> − 29	=	1.000000	0.00<i>E</i> + 00 ± 0.00<i>E</i> + 00	=	1.000000	0.00<i>E</i> + 00 ± 0.00<i>E</i> + 00	
<i>F</i> ₂	1.48<i>E</i> − 11 ± 2.34<i>E</i> − 11	−	0.000012	5.34 <i>E</i> − 09 ± 7.74 <i>E</i> − 09	+	0.000012	2.24 <i>E</i> − 11 ± 4.81 <i>E</i> − 11	
<i>F</i> ₃	9.67 <i>E</i> + 04 ± 5.30 <i>E</i> + 04	=	0.696425	8.89<i>E</i> + 04 ± 5.33<i>E</i> + 04	=	0.492633	9.97 <i>E</i> + 04 ± 5.25 <i>E</i> + 04	
<i>F</i> ₄	3.75 <i>E</i> + 01 ± 6.47 <i>E</i> + 01	=	0.637733	3.20<i>E</i> + 01 ± 9.74<i>E</i> + 01	=	0.069337	3.92 <i>E</i> + 01 ± 5.56 <i>E</i> + 01	
<i>F</i> ₅	6.44<i>E</i> + 02 ± 6.02<i>E</i> + 02	=	0.967806	6.84 <i>E</i> + 02 ± 6.65 <i>E</i> + 02	=	0.777543	7.05 <i>E</i> + 02 ± 6.53 <i>E</i> + 02	
<i>F</i> ₆	6.61 <i>E</i> − 01 ± 1.48 <i>E</i> + 00	=	0.128451	9.31 <i>E</i> − 01 ± 2.58 <i>E</i> + 00	=	0.051087	4.81<i>E</i> − 01 ± 1.32<i>E</i> + 00	
<i>F</i> ₇	1.10 <i>E</i> − 02 ± 1.13 <i>E</i> − 02	+	0.024651	5.71<i>E</i> − 03 ± 6.58<i>E</i> − 03	=	0.389602	7.98 <i>E</i> − 03 ± 1.03 <i>E</i> − 02	
<i>F</i> ₈	2.09 <i>E</i> + 01 ± 1.01 <i>E</i> − 01	=	0.121828	2.09 <i>E</i> + 01 ± 7.18 <i>E</i> − 02	=	0.777543	2.09 <i>E</i> + 01 ± 4.18 <i>E</i> − 02	
<i>F</i> ₉	2.69 <i>E</i> + 01 ± 6.76 <i>E</i> + 00	+	0.000058	1.62<i>E</i> + 01 ± 5.19<i>E</i> + 00	−	0.000025	3.20 <i>E</i> + 01 ± 8.96 <i>E</i> + 00	
<i>F</i> ₁₀	6.93 <i>E</i> + 01 ± 1.89 <i>E</i> + 01	=	0.150003	6.14<i>E</i> + 01 ± 1.48<i>E</i> + 01	−	0.002259	7.53 <i>E</i> + 01 ± 1.74 <i>E</i> + 01	
−/=/+			3/11/6			6/12/2		

that CCS apparently converges faster than CS in terms of convergence curves.

According to *Error, Evaluation, and Convergence graphs*, CCS overall significantly improves the solution quality and convergence speed of CS. This is because chaotic maps can provide various search step information, and more probabilistic learning from others, which are beneficial to improve the search ability of CS. Additionally, the analysis of scalability suggests that the advantage of CCS over CS is overall stable when the dimensionality of the problems increases.

4.4. Sensitivities to Initial Value of Chaotic Maps. It is worthy pointing out that the chaotic sequences are highly sensitive to initial condition. To show the performance of CCS affected by the initial value, we perform the experiments on chaotic maps with different initial values. The results are listed in Table 8, where the initial values are 0.25 and 0.5, resulting in CCS₂₅ and CCS₅, respectively. The other parameters are kept unchanged.

Seen from Table 8, we can find that the performance of CCS will be influenced weakly by the initial value of chaotic maps in terms of *Error*. CCS₂₅ obtains the highest accuracy on F_{grw} , F_2 , and F_5 , while CCS₅ brings the highest accurate solutions to F_{sph} , F_{pn1} , F_{pn2} , F_1 , and F_6 . However, CCS achieves the solutions with highest accuracy for most functions. Nevertheless, according to their statistical results “–/=/+,” CCS shows better performance than CCS₂₅ and CCS₅ on 6 out of 20 functions and draws a tie of CCS₂₅ and CCS₅ on 11 and 12 out of 20 functions, respectively.

This suggests that the initial value 0.7 in default is the better selection.

4.5. Comparison with Other Improved CS Algorithms. To show the competitiveness of CCS with the other improved CS algorithms, we compare it at $D = 30$ with three improved versions, called ICS [9], CSPSO [16], and OLCS [40]. Note that ICS defines the scaling factor and the fraction probability in variable value schema based on two maximum and minimum parameters. The results are reported in Tables 9, 10, and 11, respectively.

As observed from Table 9, each algorithm shows its advantage on parts of functions. For example, ICS performs better on F_{grw} , F_{pn1} , F_{pn2} , F_1 , and F_9 . CSPSO gains the highest accurate solution to F_2 . OLCS obtains the solutions with higher accuracy on F_{sph} , F_{ack} , F_{sal} , and F_7 and reaches the global optima on F_{grw} and F_{ras} . CCS achieves the global solutions to F_{grw} and F_1 and presents its advantage on rotated or shifted functions like F_3 , F_4 , F_5 , F_6 , and F_{10} . Nevertheless, with the help of “–/=/+,” CCS outperforms ICS, CSPSO, and OLCS on 9, 16, and 13 out of 20 functions. Moreover, Table 10 shows that CCS yields the higher R^+ values than R^- values in all cases. In addition, it can be seen from Table 11 clearly that CCS gains the first average ranking, followed by ICS, OLCS, and CSPSO.

4.6. Discussion. CCS shows its promising performance by using two chaotic maps simultaneously to define the scaling factor and the fraction probability. In this case, two chaotic maps make cooperative contribution to the performance of

TABLE 9: Error obtained by ICS, CSPSO, OLCS, and CCS for 20 functions at $D = 30$.

	ICS	CSPSO	OLCS	CCS
	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$	$AVG_{Er} \pm STD_{Er}$
F_{sph}	$1.31E - 48 \pm 2.82E - 48$	$4.42E - 44 \pm 1.46E - 43$	$6.27E - 128 \pm 3.01E - 127$	$5.08E - 55 \pm 9.57E - 55$
F_{ros}	$9.57E + 00 \pm 3.30E + 00$	$4.79E - 01 \pm 1.32E + 00$	$2.33E + 00 \pm 8.61E - 01$	$2.12E - 01 \pm 8.10E - 01$
F_{ack}	$9.66E - 15 \pm 3.33E - 15$	$6.41E - 14 \pm 2.59E - 14$	$2.66E - 15 \pm 0.00E + 00$	$8.24E - 15 \pm 2.66E - 15$
F_{grw}	$0.00E + 00 \pm 0.00E + 00$	$6.20E - 03 \pm 8.53E - 03$	$0.00E + 00 \pm 0.00E + 00$	$0.00E + 00 \pm 0.00E + 00$
F_{ras}	$1.80E + 01 \pm 3.95E + 00$	$2.62E + 01 \pm 6.52E + 00$	$0.00E + 00 \pm 0.00E + 00$	$1.48E + 01 \pm 6.90E + 00$
F_{sch}	$7.06E + 02 \pm 4.12E + 02$	$4.66E + 03 \pm 7.35E + 02$	$2.16E + 03 \pm 3.18E + 02$	$7.65E + 02 \pm 2.67E + 02$
F_{sal}	$2.16E - 01 \pm 3.74E - 02$	$4.24E - 01 \pm 5.97E - 02$	$6.31E - 07 \pm 2.78E - 06$	$2.64E - 01 \pm 4.90E - 02$
F_{wht}	$2.54E + 02 \pm 5.46E + 01$	$4.90E + 02 \pm 8.07E + 01$	$3.15E + 02 \pm 1.74E + 01$	$1.76E + 02 \pm 7.48E + 01$
F_{pn1}	$1.57E - 32 \pm 5.59E - 48$	$5.39E - 02 \pm 1.12E - 01$	$3.39E - 30 \pm 3.63E - 30$	$1.57E - 32 \pm 5.59E - 48$
F_{pn2}	$1.35E - 32 \pm 5.59E - 48$	$1.32E - 03 \pm 3.64E - 03$	$4.37E - 29 \pm 5.11E - 29$	$1.35E - 32 \pm 5.59E - 48$
F_1	$0.00E + 00 \pm 0.00E + 00$	$2.52E - 28 \pm 2.31E - 28$	$1.31E - 26 \pm 1.05E - 26$	$0.00E + 00 \pm 0.00E + 00$
F_2	$1.77E - 03 \pm 2.24E - 03$	$3.28E - 11 \pm 9.03E - 11$	$5.80E - 02 \pm 5.16E - 02$	$5.34E - 09 \pm 7.74E - 09$
F_3	$3.27E + 05 \pm 2.03E + 05$	$7.44E + 05 \pm 6.43E + 05$	$2.83E + 06 \pm 7.55E + 05$	$8.89E + 04 \pm 5.33E + 04$
F_4	$2.95E + 02 \pm 1.75E + 02$	$8.41E + 01 \pm 1.05E + 02$	$2.01E + 03 \pm 8.15E + 02$	$3.20E + 01 \pm 9.74E + 01$
F_5	$1.64E + 03 \pm 5.85E + 02$	$2.89E + 03 \pm 7.82E + 02$	$2.54E + 03 \pm 5.86E + 02$	$6.84E + 02 \pm 6.65E + 02$
F_6	$1.19E + 01 \pm 4.08E + 00$	$2.31E + 00 \pm 4.79E + 00$	$2.52E + 01 \pm 2.04E + 01$	$9.31E - 01 \pm 2.58E + 00$
F_7	$2.59E - 03 \pm 3.44E - 03$	$7.40E - 03 \pm 2.20E - 15$	$3.43E - 04 \pm 4.42E - 04$	$5.71E - 03 \pm 6.58E - 03$
F_8	$2.09E + 01 \pm 7.85E - 02$	$2.09E + 01 \pm 5.49E - 02$	$2.09E + 01 \pm 5.75E - 02$	$2.09E + 01 \pm 7.18E - 02$
F_9	$1.47E + 01 \pm 3.14E + 00$	$1.55E + 02 \pm 2.47E + 01$	$3.56E + 01 \pm 6.56E + 00$	$1.62E + 01 \pm 5.19E + 00$
F_{10}	$8.15E + 01 \pm 1.30E + 01$	$2.57E + 02 \pm 6.81E + 01$	$1.52E + 02 \pm 3.48E + 01$	$6.14E + 01 \pm 1.48E + 01$
-/=/+	1/10/9	1/3/16	4/3/13	

TABLE 10: Results of the multiple-problem Wilcoxon's test for ICS, CSPSO, OLCS, and CCS for 20 functions at $D = 30$.

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
CCS versus ICS	156.375	53.625	0.055115	=	+
CCS versus CSPSO	206	4	0.000163	+	+
CCS versus OLCS	171.5	38.5	0.013042	+	+

TABLE 11: Average ranking of eight algorithms by the Friedman test for 20 functions at $D = 30$.

Algorithm	ICS	CSPSO	OLCS	CCS
Ranking	2.23	3.45	2.70	1.63

CCS. In this section, therefore, we discuss the contribution of each chaotic map to the performance of CCS. To analyze the contribution of each chaotic map, we consider two derived algorithms: CCS1 and CCS2. The former uses chaotic map to define the scaling factor and keeps the original BSRW, while the later utilizes chaotic map to define the fraction probability and keeps the original LFRW. CCS1 and CCS2 are performed on 20 benchmark functions at $D = 30$, and the results are listed in Table 12.

It can be observed from Table 12 that the single chaotic map makes different contribution to the performance of CCS for different functions. Compared with CS, CCS1 singly brings solutions with higher accuracy to F_{sph} , F_{ack} , F_{grw} , F_{sch} , F_{sal} , F_{pn1} , F_{pn2} , and F_1 , while CCS2 alone achieves higher accurate solutions to F_3 . Due to the contribution of these

higher accurate solutions, CCS yields better performance. Moreover, it can be suggested from Table 12 that CCS1 and CCS2 both achieve better performance and cooperatively make contribution to the performance of CCS. For example, for most of rotated and/or shifted functions, for example, F_4 , F_5 , F_6 , F_9 , and F_{10} , CCS1 and CCS2 obtain the slightly higher accurate solutions, but CCS further performs better due to their cooperative contribution.

5. Conclusion and Future Work

In CS, the scaling factor and the fraction probability parameters are used in constant value way, resulting in a problem sensitive to solution quality and convergence speed. In this paper, we employed chaotic maps to define the scaling factor and the fraction probability in variable value schema and proposed chaotic cuckoo search algorithm, called CCS. Comprehensive experiments were carried out on 20 benchmark functions to test the performances of CCS. The results show that chaotic maps can improve the performance of CS effectively and efficiently. The scalability study reveals that the advantage of CCS over CS is overall stable when increasing the dimensionality of problems. The results in comparison with another study on the scaling factor and the fraction probability verify that chaotic maps are a better selection to define the variable value schema.

There are several interesting directions for future work. First, it is interesting to test the different combinations of chaotic maps to find the optimal one. Second, we plan to integrate chaotic maps into improved CS algorithms to

TABLE 12: Error obtained by CS, CCS1, CCS2, and CCS for 20 functions at $D = 30$.

	CS			CCS1			CCS			CCS2		
	$AVG_{Er} \pm STD_{Er}$	p value		$AVG_{Er} \pm STD_{Er}$	p value		$AVG_{Er} \pm STD_{Er}$	p value		$AVG_{Er} \pm STD_{Er}$	p value	
F_{sph}	$1.09E - 30 \pm 1.35E - 30$		+	$1.84E - 52 \pm 2.30E - 52$	0.000012	+	$5.08E - 55 \pm 9.57E - 55$	—	—	$3.66E - 27 \pm 4.81E - 27$	0.000012	—
F_{ros}	$1.68E + 01 \pm 2.09E + 01$		+	$3.26E + 00 \pm 2.45E + 00$	0.000101	+	$2.12E - 01 \pm 8.10E - 01$	—	—	$7.53E + 00 \pm 4.23E + 00$	0.000041	—
F_{ack}	$3.73E - 02 \pm 1.86E - 01$		=	$7.39E - 15 \pm 1.42E - 15$	0.375000	=	$8.24E - 15 \pm 2.66E - 15$	—	—	$4.94E - 01 \pm 6.45E - 01$	0.000012	—
F_{grw}	$9.23E - 12 \pm 4.62E - 11$		=	$0.00E + 00 \pm 0.00E + 00$	1.000000	=	$0.00E + 00 \pm 0.00E + 00$	—	—	$1.38E - 03 \pm 3.27E - 03$	0.031250	—
F_{ras}	$2.47E + 01 \pm 4.87E + 00$		=	$1.93E + 01 \pm 9.04E + 00$	0.115475	=	$1.48E + 01 \pm 6.90E + 00$	—	—	$2.11E + 01 \pm 5.74E + 00$	0.000665	—
F_{sch}	$1.44E + 03 \pm 2.65E + 02$		=	$6.33E + 02 \pm 4.41E + 02$	0.142532	=	$7.65E + 02 \pm 2.67E + 02$	—	—	$1.42E + 03 \pm 2.88E + 02$	0.000012	—
F_{sal}	$3.88E - 01 \pm 8.33E - 02$		=	$2.32E - 01 \pm 4.76E - 02$	0.057836	=	$2.64E - 01 \pm 4.90E - 02$	—	—	$5.88E - 01 \pm 1.45E - 01$	0.000014	—
F_{whit}	$3.75E + 02 \pm 4.50E + 01$		+	$2.57E + 02 \pm 6.93E + 01$	0.000891	+	$1.76E + 02 \pm 7.48E + 01$	—	—	$3.58E + 02 \pm 6.16E + 01$	0.000012	—
F_{pn1}	$5.36E - 20 \pm 1.57E - 19$		=	$1.57E - 32 \pm 5.59E - 48$	1.000000	=	$1.57E - 32 \pm 5.59E - 48$	—	—	$3.27E - 17 \pm 1.47E - 16$	0.000012	—
F_{pn2}	$1.82E - 25 \pm 6.31E - 25$		=	$1.35E - 32 \pm 5.59E - 48$	1.000000	=	$1.35E - 32 \pm 5.59E - 48$	—	—	$5.56E - 22 \pm 2.58E - 21$	0.000012	—
F_1	$5.30E - 30 \pm 1.25E - 29$		=	$0.00E + 00 \pm 0.00E + 00$	1.000000	=	$0.00E + 00 \pm 0.00E + 00$	—	—	$1.66E - 26 \pm 3.10E - 26$	0.000012	—
F_2	$6.08E - 03 \pm 3.12E - 03$		+	$1.15E - 02 \pm 7.88E - 03$	0.000012	+	$5.34E - 09 \pm 7.74E - 09$	—	—	$4.32E - 06 \pm 1.17E - 05$	0.000012	—
F_3	$2.08E + 06 \pm 5.37E + 05$		+	$3.21E + 06 \pm 1.32E + 06$	0.000012	+	$8.89E + 04 \pm 5.33E + 04$	+	+	$2.85E + 04 \pm 2.73E + 04$	0.000072	—
F_4	$1.28E + 03 \pm 7.90E + 02$		+	$3.64E + 02 \pm 2.42E + 02$	0.000072	+	$3.20E + 01 \pm 9.74E + 01$	—	—	$6.00E + 02 \pm 6.00E + 02$	0.000032	—
F_5	$2.92E + 03 \pm 6.56E + 02$		+	$1.79E + 03 \pm 7.10E + 02$	0.000012	+	$6.84E + 02 \pm 6.65E + 02$	—	—	$1.10E + 03 \pm 5.45E + 02$	0.018555	—
F_6	$1.99E + 01 \pm 2.48E + 01$		+	$1.50E + 01 \pm 2.17E + 01$	0.000090	+	$9.31E - 01 \pm 2.58E + 00$	—	—	$1.03E + 01 \pm 6.02E + 00$	0.000023	—
F_7	$1.09E - 03 \pm 2.04E - 03$		=	$1.46E - 03 \pm 3.44E - 03$	0.174210	=	$5.71E - 03 \pm 6.58E - 03$	—	—	$4.53E - 03 \pm 7.16E - 03$	0.967806	—
F_8	$2.09E + 01 \pm 5.26E - 02$		=	$2.09E + 01 \pm 5.48E - 02$	0.657069	=	$2.09E + 01 \pm 7.18E - 02$	—	—	$2.09E + 01 \pm 5.48E - 02$	0.339479	—
F_9	$2.73E + 01 \pm 5.84E + 00$		=	$1.57E + 01 \pm 6.49E + 00$	0.798248	=	$1.62E + 01 \pm 5.19E + 00$	—	—	$2.66E + 01 \pm 7.01E + 00$	0.000020	—
F_{10}	$1.58E + 02 \pm 2.39E + 01$		+	$1.00E + 02 \pm 1.30E + 01$	0.000012	+	$6.14E + 01 \pm 1.48E + 01$	—	—	$1.57E + 02 \pm 2.57E + 01$	0.000012	—
$-/+/-/+$					0/11/9						17/2/1	

further verify their efficiency and effectiveness. Last but not least, we also plan to apply CCS to some real-world optimization problems for further examinations.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are very grateful to the editor and the anonymous reviewers for their constructive comments and suggestions to this paper. This work was supported by the Natural Science Foundation of Fujian Province of China under Grant no. 2013J01216.

References

- [1] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC '09)*, pp. 210–214, IEEE, Coimbatore, India, 2009.
- [2] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.
- [3] P. Civicioglu and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 315–346, 2013.
- [4] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, "Modified cuckoo search: a new gradient free optimisation algorithm," *Chaos, Solitons & Fractals*, vol. 44, no. 9, pp. 710–718, 2011.
- [5] P. Nasa-Ngium, K. Sunat, and S. Chiewchanwattana, "Enhancing modified cuckoo search by using Mantegna Lévy flights and chaotic sequences," in *Proceedings of the 10th International Joint Conference on Computer Science and Software Engineering (JCSSE '13)*, pp. 53–57, IEEE, Maha Sarakham, Thailand, May 2013.
- [6] S. Das, P. Dasgupta, and B. K. Panigrahi, "Inter-species Cuckoo search via different Lévy flights," in *Proceedings of the 4th International Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO '13)*, B. K. Panigrahi, P. N. Suganthan, S. Das, and S. S. Dash, Eds., vol. 8297 of *Lecture Notes in Computer Science*, part I, pp. 515–526, Springer International Publishing, Cham, Switzerland, 2013.
- [7] X. M. Ding, Z. K. Xu, N. J. Cheung, and X. H. Liu, "Parameter estimation of Takagi-Sugeno fuzzy system using heterogeneous cuckoo search algorithm," *Neurocomputing*, vol. 151, no. 3, pp. 1332–1342, 2015.
- [8] Z. X. Zhang and Y. J. Chen, "An improved cuckoo search algorithm with adaptive method," in *Proceedings of the 7th International Joint Conference on Computational Sciences and Optimization (CSO '14)*, pp. 204–207, Beijing, China, July 2014.
- [9] E. Valian, S. Mohanna, and S. Tavakoli, "Improved cuckoo search algorithm for global optimization," *International Journal of Communications and Information Technology*, vol. 1, no. 1, pp. 31–44, 2011.
- [10] L. J. Wang, Y. L. Yin, and Y. W. Zhong, "Cuckoo search with varied scaling factor," *Froniers of Computer Science*, 2015.
- [11] M. Tuba, M. Subotic, and N. Stanarevic, "Performance of a modified cuckoo search algorithm for unconstrained optimization problems," *WSEAS Transactions on Systems*, vol. 2, no. 1, pp. 263–268, 2012.
- [12] L. J. Wang, Y. L. Yin, and Y. W. Zhong, "Cuckoo search algorithm with dimension by dimension improvement," *Journal of Software*, vol. 24, no. 11, pp. 2687–2698, 2013.
- [13] S. K. Fateen and A. Bonilla-Petriciolet, "Gradient-based cuckoo search for global optimization," *Mathematical Problems in Engineering*, vol. 2014, Article ID 493740, 12 pages, 2014.
- [14] S.-E. K. Fateen and A. Bonilla-Petriciolet, "A note on effective phase stability calculations using gradient-based cuckoo search algorithm," *Fluid Phase Equilibria*, vol. 375, pp. 360–366, 2014.
- [15] Y. Q. Zhou and H. Q. Zheng, "A novel complex valued cuckoo search algorithm," *The Scientific World Journal*, vol. 2013, Article ID 597803, 6 pages, 2013.
- [16] F. Wang, X. S. He, L. G. Luo, and Y. Wang, "Hybrid optimization algorithm of PSO and cuckoo search," in *Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electric Commerce (AIMSEC '11)*, pp. 1172–1175, Zhengzhou, China, 2011.
- [17] A. Ghodrati and S. Lotfi, "A hybrid CS/PSO algorithm for global optimization," in *Intelligent Information and Database Systems: 4th Asian Conference, ACIIDS 2012, Kaohsiung, Taiwan, March 19-21, 2012, Proceedings, Part III*, vol. 7198 of *Lecture Notes in Computer Science*, pp. 89–98, Springer, Berlin, Germany, 2012.
- [18] P. R. Srivastava, R. Khandelwal, S. Khandelwal, S. Kumar, and S. S. Ranganatha, "Automated test data generation using cuckoo search and tabu search (CSTS) algorithm," *Journal of Intelligent Systems*, vol. 21, no. 2, pp. 195–224, 2012.
- [19] G. G. Wang, L. H. Guo, H. Duan, and L. Liu, "A hybrid meta-heuristic DE/CS Algorithm for UCAV path planning," *Journal of Information & Computational Science*, vol. 9, no. 16, pp. 4811–4818, 2012.
- [20] R. G. Babukartik and P. Dhavachelvan, "Hybrid algorithm using the advantage of ACO and Cuckoo Search for Job Scheduling," *International Journal of Information Technology Convergence and Services*, vol. 2, no. 4, pp. 25–34, 2012.
- [21] X.-X. Hu and Y.-L. Yin, "Cooperative co-evolutionary cuckoo search algorithm for continuous function optimization problems," *Pattern Recognition and Artificial Intelligence*, vol. 26, no. 11, pp. 1041–1049, 2013.
- [22] H. Zheng and Y. Zhou, "A cooperative coevolutionary cuckoo search algorithm for optimization problem," *Journal of Applied Mathematics*, vol. 2013, Article ID 912056, 9 pages, 2013.
- [23] S. Saremi, S. Mirjalili, and A. Lewis, "Biogeography-based optimisation with chaos," *Neural Computing & Applications*, vol. 25, no. 5, pp. 1077–1097, 2014.
- [24] R. Ebrahimzadeh and M. Jampour, "Chaotic genetic algorithm based on lorenz chaotic system for optimization problems," *International Journal of Intelligent Systems and Applications*, vol. 5, no. 5, pp. 19–24, 2013.
- [25] Z. Y. Guo, B. Chen, M. Ye, and B. Cao, "Self-adaptive chaos differential evolution," in *Advances in Natural Computation: Second International Conference, ICNC 2006, Xi'an, China, September 24-28, 2006. Proceedings, Part I*, L. Jiao, L. Wang, X.-B. Gao, J. Liu, and F. Wu, Eds., vol. 4221 of *Lecture Notes in Computer Science*, pp. 972–975, Springer, Berlin, Germany, 2006.
- [26] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 1, pp. 89–98, 2013.

- [27] S. Saremi, S. M. Mirjalili, and S. Mirjalili, "Chaotic krill herd optimization algorithm," *Procedia Technology*, vol. 12, pp. 180–185, 2014.
- [28] G.-G. Wang, L. H. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences*, vol. 274, pp. 17–34, 2014.
- [29] S. Saremi and S. Mirjalili, "Integrating chaos to biogeography-based optimization algorithm," *International Journal of Computer and Communication Engineering*, vol. 2, no. 6, pp. 655–658, 2013.
- [30] R. C. Hilborn, *Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers*, Oxford University Press, New York, NY, USA, 2nd edition, 2004.
- [31] Y. Li, S. Deng, and D. Xiao, "A novel hash algorithm construction based on chaotic neural network," *Neural Computing and Applications*, vol. 20, no. 1, pp. 133–141, 2011.
- [32] A. G. Tomida, "Matlab toolbox and GUI for analyzing one-dimensional chaotic maps," in *Proceedings of the International Conference on Computational Sciences and Its Applications (ICCSA '08)*, pp. 321–330, IEEE Press, Perugia, Italy, July 2008.
- [33] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, Addison-Wesley, Redwood City, Calif, USA, 1986.
- [34] H.-O. Peitgen, H. Jurgens, and D. Saupe, *Chaos and Fractals*, Springer, Berlin, Germany, 1992.
- [35] E. Ott, *Chaos in Dynamical Systems*, Cambridge University Press, Cambridge, UK, 2002.
- [36] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.
- [37] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, Singapore, 2005.
- [38] W. Y. Gong and Z. H. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2066–2081, 2013.
- [39] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.
- [40] X. T. Li, J. N. Wang, and M. H. Yin, "Enhancing the performance of cuckoo search algorithm using orthogonal learning method," *Neural Computing & Applications*, vol. 24, no. 6, pp. 1233–1247, 2014.