Research Article

Cuckoo Search Algorithm with Hybrid Factor Using Dimensional Distance

Yaohua Lin,¹ Cuiping Zhang,² and Zhong Liang¹

¹College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou 350002, China

²College of Management, Fujian University of Traditional Chinese Medicine, Fuzhou 350002, China

Correspondence should be addressed to Zhong Liang; 553672109@qq.com

Received 15 May 2016; Accepted 6 November 2016

Academic Editor: Salvatore Alfonzetti

Copyright © 2016 Yaohua Lin et al. 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.

This paper proposes a hybrid factor strategy for cuckoo search algorithm by combining constant factor and varied factor. The constant factor is used to the dimensions of each solution which are closer to the corresponding dimensions of the best solution, while the varied factor using a random or a chaotic sequence is utilized to farer dimensions. For each solution, the dimension whose distance to the corresponding one of the best solution is shorter than mean distance of all dimensional distances will be regarded as the closer one, otherwise as the farer one. A suit of 20 benchmark functions are employed to verify the performance of the proposed strategy, and the results show the improvement in effectiveness and efficiency of the hybridization.

1. Introduction

Cuckoo search algorithm (CS), proposed by Yang and Deb in 2009, is a new nature-inspired method for solving real-valued numerical optimization problems [1, 2]. The method utilizes Lévy flights random walk (LFRW) and biased random walk to search for new solutions and achieves the promising performance for many tough problems. This has attracted a lot of researchers, and many studies have been proposed. Some studies have focused on the combination CS with other optimization methods [3–13]. Some attempts have been made to improve search ability of LFRW and BSW [14–32]. Other attentions have been played on CS for the combinational and multiobject problems [33–41].

The above studies have made great contributions to CS. Nevertheless, according to the implementation in the literature [2], Lévy flights random walk (LFRW), one of search components, is used iteratively to search for new solutions. LFRW uses a mutation operator to generate new solutions based on the best solution obtained so far. A factor in LFRW is utilized to control Lévy flights not to be too aggressive; thus, it is suggested to a constant value, typically 0.01 [2]. In this case, it is beneficial to the solutions which are close to the best one, but it is a disadvantage to those

far away from the best one. To avoid the above, Wang et al. [22] proposed a varied factor strategy for CS, named as VCS, where the random sequence factor obeying uniformly distribution is used to replace the constant one. Wang and Zhong [23] used a chaotic sequence factor instead of the constant one, called CCS. However, the above researches make the scale of step size of all dimensions of one solution not different due to the same factor. This may cause a part of dimensions to be too aggressive when the large factor is sampled or too inefficient in the case of the small factor.

In this paper, we aim at avoiding the above problem by using the different factor for the dimensions of each solution and then propose a hybrid factor based cuckoo search algorithm, termed as HFCS. The hybrid factor strategy (HF) combines the constant factor and the varied factor. The constant factor, typically 0.01, is used to benefit the dimensions which are closer to the corresponding ones of the best solution. The varied factor using a random sequence or a chaotic sequence is employed to drive the farer dimensions to be near the corresponding ones of the best one. HFCS selects the dimension of each solution by using the dimensional distance that can be defined as the distance between one dimension and the corresponding one of the best solution. If the dimensional distance of one

dimension is shorter than the average of all dimensional distances, then this dimension is selected as the closer one, else as the farer one. The experiments are carried out on 20 benchmark functions to test HFCS, and the results show the improvement in effectiveness and efficiency of hybrid factor strategy.

The remainder of this paper is organized as follows. Section 2 describes the cuckoo search algorithm and the variants. Section 3 presents the proposed algorithm. Section 4 reports the experimental results. Section 5 concludes this paper.

2. Cuckoo Search Algorithm

2.1. CS. CS, a new nature-inspired algorithm 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] is a simple yet very promising population-based stochastic search technique. Generally, a nest represents a candidate solution $X = (x_1, \dots, x_D)$, when solving an objective function $f(x)$ with the solution space $[x_{j,\min}, x_{j,\max}]$, $j = 1, 2, \dots, D$. Like evolutionary algorithms, the iteration process of CS includes the initial phase and evolutionary phase.

In the initial phase, the whole population called solution is randomly sampled from solution space by

$$x_{i,j,0} = x_{i,j,\min} + r(x_{i,j,\max} - x_{i,j,\min}), \quad i = 1, 2, \dots, N, \quad (1)$$

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

According to the implementation of CS shown in the literature [2], CS iteratively uses two random walks: Lévy flights random walk (LFRW) and biased random walk (BRW) to search for new solutions.

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 related to the scales of the problem. The $\oplus$ means entry-wise multiplications. $\text{Lévy}(\beta)$ is drawn from a Lévy distribution for large steps:

$$\text{Lévy}(\beta) \sim u = t^{-1-\beta}, \quad 0 < \beta \leq 2. \quad (3)$$

In CS, LFRW is employed to search for new solutions around the best solution obtained so far and implemented according to the following equation [2]:

$$X_{i,G+1} = X_{i,G} + \alpha_0 \times \frac{\phi \times u}{|v|^{1/\beta}} \times (X_{i,G} - X_{\text{best}}), \quad (4)$$

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

$$\phi = \left(\frac{\Gamma(1 + \beta) \times \sin((\pi \times \beta)/2)}{\Gamma((1 + \beta)/2) \times \beta \times 2^{(\beta-1)/2}} \right)^{1/\beta}, \quad (5)$$

where β is a constant and suggested to be 1.5, 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*.

BRW 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:

$$x_{i,j,G+1} = \begin{cases} x_{i,j,G} + r(x_{m,j,G} - x_{n,j,G}), & \text{if } r_a > p_a, \\ x_{i,j,G}, & \text{otherwise,} \end{cases} \quad (6)$$

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 r_a are random numbers on the range $[0, 1]$, and p_a is a fraction probability.

After each random walk, CS selects a better solution according to the new generated and the current solutions fitness using the greedy strategy. At the end of each iteration process, the best solution is updated.

2.2. Variants of CS. CS is developed recently, but this algorithm has been researched a lot.

Some studies are an attempt to combine CS with other optimization techniques. Wang et al. [3] and Ghodrati and Lotfi [4], respectively, proposed a hybrid CS with particle swarm optimization. Wang et al. [5] applied differential evolution to optimize the process of selecting cuckoo in nest updating. Babukartik and Dhavachelvan [6] proposed the hybrid algorithm combining ant colony optimization and CS. Srivastava et al. [7] combined the CS algorithm's strength of converging to the solution in minimal time along with the tabu mechanism of backtracking from local optimal by Lévy flight. Liu and Fu [8] applied the local search mechanism of the frog leaping algorithm to enhance the local search ability of cuckoo search. Other techniques, such as the orthogonal learning strategy [9], cooperative coevolutionary (CC) framework [10–12], and the teaching-learning-based optimization [13], are also hybridized to enhance the search ability of cuckoo search.

Some variants have paid attention to improve search ability of LFRW and BSW. Walton et al. [14] made a modification to the step size of Lévy flights decreasing as the number of generations increases in order to increase the convergence rate. Ljouad et al. [15] modified Lévy flights model with an adaptive step size based on the number of generations. Valian et al. [16], Wang and Zhou [17], and Mohapatra et al. [18] proposed adaptive step sizes of Lévy flights according to different equations with maximal and minimal step sizes, respectively. Wang et al. [19] and Huang et al. [20] used the chaotic sequence to change the step size of Lévy flights, respectively. Jia et al. [21] proposed the variable step length of Lévy flights and a method of discovering probability. Wang et al. [22, 23], respectively, used random sequence and chaotic

```

 $G \leftarrow 0$ ;
 $Nest_0 = (X_{i,0}, \dots, X_{N,0}) \leftarrow$  Initialize solutions using Eq. (1);
Fitness  $\leftarrow$  Evaluate the solution  $Nest_0$ ;
FES  $\leftarrow N$ ;
BestX  $\leftarrow$  Find the best solution according to the Fitness;
WHILE (FES < MaxFES)
   $G \leftarrow G + 1$ ;
  variedFactor  $\leftarrow$  Get a rand or a chaotic factor from sequence
  FOR ( $i$  from 1 to  $N$ )
    FOR ( $j$  from 1 to  $D$ )
      If the  $j$ th of the  $i$ th solution is closer to the  $j$ th of BestX
         $\alpha_{0,j} \leftarrow 0.01$ 
      Else
         $\alpha_{0,j} \leftarrow$  variedFactor
      Endif
      new $X_{i,j,G} \leftarrow$  Generate  $j$ th of new solution with Eq. (4)
    ENDFOR
     $X_{i,G} \leftarrow$  Evaluate and select from new $X_{i,G}$  and  $X_{i,G}$ ;
    FES  $\leftarrow$  FES + 1;
  ENDFOR
  FOR ( $i$  from 1 to  $NP$ )
    new $X_{i,G} \leftarrow$  Generate a new solution with Eq. (6)
     $X_{i,G} \leftarrow$  Evaluate and select from new $X_{i,G}$  and  $X_{i,G}$ ;
    FES  $\leftarrow$  FES + 1;
  ENDFOR
  BestX  $\leftarrow$  Find and update the best solution;
ENDWHILE

```

ALGORITHM 1: HFCS.

sequence as the factor instead of the constant 0.01 in Lévy flights. Coelho et al. [24] integrated the differential operator into Lévy flights to search for new solutions. Mlakar et al. [25] proposed the hybrid algorithm using explicit control of exploration search strategies with the CS algorithm. Ding et al. [26] proposed heterogeneous search strategies based on the quantum mechanism. Wang et al. [27] employed a probabilistic mutation to enhance the Lévy flights. Wang and Zhong [28] added a crossover-like operator in search schema of Lévy flights using one-position inheritance mechanism. Inspired by the social learning and cognitive learning, Li and Yin [29] added these two learning parts into Lévy flights and into BSW. Wang et al. [30, 31] utilized orthogonal crossover and dimension by dimension improvement to enhance the search ability of BSW, respectively. Li and Yin [32] used two new mutation operators based on the rand and best individuals among the entire population to enhance the search ability of BSW.

Other versions have focused on the combinational and multiobject problems. Yang and Deb [33], Hanoun et al. [34], and Chandrasekaran and Simon [35] modified the cuckoo search to solve multiobjective optimization problems. Ouyang et al. [36] and Quaarab et al. [37] proposed the improved CS to solve the travelling salesman problem. Zhou et al. [38] applied an improved CS for solving planar graph coloring problem. Marichelvam et al. [39] and Dasgupta and Das [40] presented the discrete versions for the flow shop

scheduling problem. Teymourian et al. [41] applied CS for solving the capacitated vehicle routing problem.

3. HFCS

According to the implementation of CS [2], the factor, for example, 0.01, is used to control Lévy flights not to be too aggressive and to try to make the solutions not jump outside of the search space. In this case, a small step size will be got due to this small factor. Obviously, this makes a contribution to the solutions nearby the best one, but it is not more helpful to the solutions far away from the best one, resulting in the slow convergence. Wang et al. [22, 23] utilized the random sequence and the chaotic sequence instead of the constant one and proved the improvement on the convergence and solution quality. However, the factor with constant or with random sequence or chaotic sequence makes the scale of each dimension of each solution be the same. This perhaps results in a problem that some dimensions near the corresponding dimensions of the best one have larger scale when using random or chaotic sequence, while some dimensions far away from the corresponding dimensions of the best one get the smaller scale in the case of the constant factor. To remedy the above problem, a hybrid factor based cuckoo search is proposed, called HFCS, and presented in Algorithm 1.

It is worth pointing out that the key part of HFCS is to select the dimension where the constant factor or the varied

factor is used. Herein, the dimensional distance and the mean dimension distance, similarly done in [42], are used and shown in (7) and (8)

$$\text{DimDist}_{i,j} = \text{abs}(\text{Best}X_j - X_{i,j}), \quad j = 1, \dots, D, \quad (7)$$

where $\text{DimDist}_{i,j}$ presents the j th dimensional distance from the i th solution to the best one

$$\text{MeanDist}_i = \frac{1}{D} \sum_{j=1}^D \text{DimDist}_{i,j}, \quad (8)$$

where MeanDist_i is the average dimensional distance of the i th solution.

For each dimension of each solution, the j th dimension is closer to the corresponding dimension of the best solution, called the closer dimension, when the j th dimensional distance is shorter than the mean dimensional distance; else the j th dimension is called the farer one. For the closer dimension, the constant factor 0.01 is used in (4), while the random sequence or the chaotic sequence is employed in (4) for the farer one.

4. Experiments and Results

In this section, HFCS is tested on 20 benchmark functions [43]. These 20 benchmark functions include 2 unimodal functions F_{sph} and F_{ros} ; 8 multimodal functions F_{ack} , F_{grw} , F_{ras} , F_{sch} , F_{sal} , F_{wht} , F_{pn1} , and F_{pn2} ; and 10 rotated and/or shifted functions F_1 – F_{10} . More detail about 20 functions can be found in [43, 44].

HFCS has the same parameters as CS, and we use the same setting for them, unless a change is mentioned. The parameter p_a is 0.25. Each algorithm is performed 25 times for each function with the dimension $D = 10, 30$, and 50 , respectively. The population size of each algorithm N is D when $D = 30$ and $D = 50$, while it is 30 in the case of $D = 10$. The maximum function evaluations are $10000 \times D$. Note that HFCS can hybridize the constant factor and the random sequence or the chaotic sequence, resulting in two algorithms: HFCS with the combination of the constant factor and the random sequence similarly done in [22], termed as rHFCS, and HFCS with the combination of the constant factor and the chaotic sequence same generated in [23], called cHFCS.

Error and the convergence speed are employed to analyze HFCS. Error, which is the function fitness Error for the solution X obtained by the algorithms, is defined as $f(X) - f(X^*)$, where X^* is the known global optimum of function. Moreover, the average and standard deviation of the best error values, presented as “ $\text{AVG}_{\text{Er}} \pm \text{STD}_{\text{Er}}$,” are used in the different tables. Additionally, the Wilcoxon signed-rank at the 5% significance level is used to show the differences of Error between two algorithms. The “+” symbol shows that HFCS outperforms the compared algorithm at the 5% significant level, the “–” symbol shows that the compared algorithm does better than HFCS, and the “=” symbol means that HFCS is equal to the compared algorithm. We summarily give the total number of statistical significant cases at the bottom of each table.

The convergence speed is measured by using the function evaluations (FES) which are spent till the algorithm reaches the given Error which is 10^{-6} or 10^{-2} as suggested in [43] within the maximum function evaluations. The average and standard deviation of FES and successful runs (SR) to the given Error during 25 runs, termed as “ $\text{AVG}_{\text{FES}} \pm \text{STD}_{\text{FES}}$ (SR),” are utilized in Table 3. Also, the convergence speed is shown graphically by using the convergence graphs where the mean Error of the best solution at iteration process over the total run is presented.

4.1. Performance of HFCS. To show the effect of the proposed algorithm, Table 1 lists Error obtained by CS, rHFCS, and cHFCS, and Table 2 lists the results of the multiple-problem Wilcoxon's test which was done similarly in [45, 46].

It can be seen from Table 1 that the hybrid factor strategy can overall improve the solution quality in terms of Error. As for 2 unimodal functions, rHFCS and cHFCS bring more accurate solutions than CS. As for 8 multimodal functions, rHFCS obtains the solutions with higher accuracy, while cHFCS does the same. As for 10 rotated and/or shifted functions, rHFCS and cHFCS both make the great improvement on the accuracy of solutions except for F_3 and F_7 . According to the summary of “+,” “=,” and “–,” rHFCS wins CS on 16 out of 20 functions, ties CS on 2 out of 20 functions, and loses CS on 2 out of functions. cHFCS is superior to CS on 16 out of 20 functions, is equal to CS on 2 out of 20 functions, and is inferior to CS on 2 out of 20 functions. In addition, in terms of the results in Table 2, rHFCS and cHFCS both get apparently higher R^+ value than R^- value. This suggests that rHFCS and cHFCS outperform CS obviously.

Moreover, Table 3 lists the required function evaluations (FES) to the given Error within the maximum function evaluations. It can be observed from Table 2 that rHFCS and cHFCS overall have a quicker convergence to the given Error within the maximum function evaluations. For instance, CS, rHFCS, and cHFCS share the same stable convergence to the given Error in terms of SR on F_{sph} , F_{pn2} , and F_1 ; however, rHFCS and cHFCS converge quicker according to FES. As for F_{ack} , F_{grw} , and F_{pn1} , rHFCS and cHFCS show the better stability and the quicker convergence. As for F_{ras} , F_{wht} , and F_6 , rHFCS and cHFCS still converge quicker than CS, although these two algorithms do not gain the stability convergence with the help of SR. As for F_7 , CS stably converge to the given Error, but rHFCS and cHFCS obtain the quicker convergence.

To further show the convergence performance, the convergence curves obtained by CS, rHFCS, and cHFCS for parts of functions are plotted in Figure 1.

It can be observed from Figure 1 that rHFCS and cHFCS achieve overall the better convergence speed in terms of curves. As for the functions solved better by rHFCS and cHFCS, for example, F_{sph} , F_{pn1} , F_{pn2} , F_2 , and F_6 , rHFCS and cHFCS still converge quicker than CS; see Figures 1(a), 1(c), 1(d), 1(e), and 1(f). As for F_{grw} , cHFCS gains the same accurate solution with CS, but cHFCS obtains a quicker convergence than CS on the beginning of iteration. It is interesting that CS does better than rHFCS and cHFCS when converging to

TABLE 1: Error obtained by CS, rHFCS, and cHFCS for 30-dimensional functions.

	CS			rHFCS			cHFCS		
	$AVG_{Er} \pm STD_{Er}$	p value		$AVG_{Er} \pm STD_{Er}$	p value		$AVG_{Er} \pm STD_{Er}$		
F_{sph}	$7.69E - 31 \pm 1.33E - 30$	+	0.000012	$3.10E - 67 \pm 3.87E - 67$	+	0.000012	$2.38E - 60 \pm 2.95E - 60$		
F_{ros}	$1.29E + 01 \pm 1.23E + 01$	+	0.000012	$6.47E - 01 \pm 1.27E + 00$	+	0.000194	$3.29E + 00 \pm 1.41E + 01$		
F_{ack}	$3.73E - 02 \pm 1.86E - 01$	+	0.000012	$7.67E - 15 \pm 1.97E - 15$	+	0.000012	$7.39E - 15 \pm 1.42E - 15$		
F_{grw}	$2.96E - 04 \pm 1.48E - 03$	=	0.500000	$0.00E + 00 \pm 0.00E + 00$	=	0.500000	$0.00E + 00 \pm 0.00E + 00$		
F_{ras}	$2.47E + 01 \pm 5.28E + 00$	+	0.000016	$1.16E + 01 \pm 5.98E + 00$	+	0.000081	$1.54E + 01 \pm 5.67E + 00$		
F_{sch}	$1.44E + 03 \pm 2.95E + 02$	+	0.000012	$3.57E + 02 \pm 3.08E + 02$	+	0.000018	$5.66E + 02 \pm 3.97E + 02$		
F_{sal}	$3.68E - 01 \pm 6.90E - 02$	+	0.000023	$2.60E - 01 \pm 5.00E - 02$	+	0.000016	$2.48E - 01 \pm 5.10E - 02$		
F_{wht}	$3.64E + 02 \pm 4.92E + 01$	+	0.000016	$1.81E + 02 \pm 9.33E + 01$	+	0.000020	$2.02E + 02 \pm 7.95E + 01$		
F_{pn1}	$1.66E - 02 \pm 8.29E - 02$	+	0.000012	$1.57E - 32 \pm 5.59E - 48$	+	0.000012	$1.57E - 32 \pm 5.59E - 48$		
F_{pn2}	$1.50E - 28 \pm 2.28E - 28$	+	0.000012	$1.35E - 32 \pm 5.59E - 48$	+	0.000012	$1.35E - 32 \pm 5.59E - 48$		
F_1	$9.21E - 30 \pm 2.29E - 29$	+	0.015625	$0.00E + 00 \pm 0.00E + 00$	+	0.015625	$0.00E + 00 \pm 0.00E + 00$		
F_2	$8.74E - 03 \pm 6.88E - 03$	+	0.000126	$1.84E - 03 \pm 1.84E - 03$	+	0.004162	$3.30E - 03 \pm 3.47E - 03$		
F_3	$2.12E + 06 \pm 5.42E + 05$	-	0.005355	$3.02E + 06 \pm 1.10E + 06$	-	0.001721	$2.83E + 06 \pm 8.65E + 05$		
F_4	$1.65E + 03 \pm 1.00E + 03$	+	0.000018	$3.27E + 02 \pm 1.82E + 02$	+	0.000014	$3.70E + 02 \pm 2.21E + 02$		
F_5	$3.01E + 03 \pm 7.74E + 02$	+	0.000029	$1.46E + 03 \pm 6.61E + 02$	+	0.000240	$2.16E + 03 \pm 6.52E + 02$		
F_6	$2.01E + 01 \pm 2.05E + 01$	+	0.002064	$7.65E + 00 \pm 1.87E + 01$	+	0.000014	$6.39E - 01 \pm 1.23E + 00$		
F_7	$6.56E - 04 \pm 1.77E - 03$	-	0.028314	$6.37E - 03 \pm 8.33E - 03$	-	0.002699	$1.45E - 02 \pm 1.76E - 02$		
F_8	$2.10E + 01 \pm 4.29E - 02$	=	0.287862	$2.09E + 01 \pm 5.36E - 02$	=	0.157770	$2.09E + 01 \pm 6.50E - 02$		
F_9	$2.73E + 01 \pm 4.60E + 00$	+	0.000065	$1.54E + 01 \pm 7.09E + 00$	+	0.000023	$1.53E + 01 \pm 5.74E + 00$		
F_{10}	$1.64E + 02 \pm 3.19E + 01$	+	0.000051	$1.11E + 02 \pm 1.98E + 01$	+	0.000014	$9.53E + 01 \pm 1.38E + 01$		
+/-/-			16/2/2			16/2/2			

TABLE 2: Results of the multiple-problem Wilcoxon's test for HFCS and CS for 20 functions at $D = 30$.

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
rHFCSversusCS	185.000	25.000	0.002821	+	+
cHFCSversusCS	184.000	26.000	0.003185	+	+

the given *Error*, shown in Table 3; however, because of the lack of convergence stability, rHFCS and cHFCS show better convergence curves; see Figure 1(f).

According to the *Error*, FES, and convergence curve, HFCS overall makes an improvement on solution quality and convergence speed. This is because different factors provide different step size information, resulting in the improvement of search ability.

4.2. Scalability of HFCS. The scalability study is investigated to show the performance of HFCS when the dimensionality of problem changes. The experiments are carried out on the 20 functions at 10- D and 50- D due to their definition up to 50- D [44]. The results are tabulated in Tables 4 and 5.

In the case of $D = 10$, according to *Error*, HFCS exhibits a great improvement on solutions to most of functions. For example, except for F_3 , F_7 , and F_8 , rHFCS brings the higher accurate solutions, while cHFCS gains the solutions with higher accuracy except for F_3 and F_8 . Furthermore, in terms of the total of “+,” “=,” and “-,” rHFCS performs better than

TABLE 3: FES obtained by CS, rHFCS, and cHFCS at $D = 30$.

Fun	CS	rHFCS	cHFCS
	$AVG_{FES} \pm STD_{FES} (SR)$	$AVG_{FES} \pm STD_{FES} (SR)$	$AVG_{FES} \pm STD_{FES} (SR)$
F_{sph}	88383 ± 2231 (25)	43527 ± 1131 (25)	45389 ± 1211 (25)
F_{ros}	—	232260 ± 22231 (2)	235200 ± 0 (1)
F_{ack}	168073 ± 18358 (24)	65055 ± 1079 (25)	68547 ± 1628 (25)
F_{grw}	133903 ± 24099 (24)	58582 ± 13464 (25)	61533 ± 14039 (24)
F_{ras}	—	236640 ± 36317 (2)	—
F_{wht}	—	102330 ± 5558 (2)	145140 ± 2715 (2)
F_{pn1}	162088 ± 28646 (24)	48329 ± 3944 (25)	51641 ± 4573 (25)
F_{pn2}	104043 ± 4511 (25)	49906 ± 3471 (25)	51164 ± 2413 (25)
F_1	93267 ± 2693 (25)	45024 ± 1427 (25)	46515 ± 1190 (25)
F_6	191580 ± 0 (1)	224858 ± 43665 (8)	259395 ± 29000 (4)
F_7	157916 ± 29354 (25)	128916 ± 64695 (20)	156660 ± 64681 (19)

TABLE 4: Error obtained by CS and HFCS for 10- and 50-dimensional benchmark functions.

	CS		rHFCS		cHFCS		CS		rHFCS		cHFCS	
	$D = 10$		$D = 50$		$D = 10$		$D = 50$		$D = 10$		$D = 50$	
	$AVG_{Er} \pm STD_{Er}$		$AVG_{Er} \pm STD_{Er}$		$AVG_{Er} \pm STD_{Er}$		$AVG_{Er} \pm STD_{Er}$		$AVG_{Er} \pm STD_{Er}$		$AVG_{Er} \pm STD_{Er}$	
F_{sph}	$4.32E - 26 \pm 5.24E - 26$	+	$3.09E - 65 \pm 3.63E - 65$	+	$2.65E - 65 \pm 6.00E - 65$	+	$3.82E - 17 \pm 2.54E - 17$	+	$1.30E - 35 \pm 1.15E - 35$	+	$7.74E - 30 \pm 5.16E - 30$	+
F_{ros}	$1.03E + 00 \pm 9.57E - 01$	+	$2.53E - 01 \pm 6.96E - 01$	+	$9.20E - 02 \pm 4.02E - 01$	+	$4.20E + 01 \pm 1.79E + 01$	=	$3.75E + 01 \pm 2.35E + 01$	=	$3.64E + 01 \pm 2.15E + 01$	=
F_{ack}	$1.02E - 10 \pm 3.96E - 10$	+	$3.55E - 15 \pm 0.00E + 00$	+	$3.41E - 15 \pm 7.11E - 16$	+	$2.42E - 04 \pm 4.99E - 04$	+	$1.52E - 14 \pm 2.62E - 15$	+	$2.00E - 14 \pm 2.69E - 15$	+
F_{grw}	$3.50E - 02 \pm 1.31E - 02$	+	$2.57E - 02 \pm 1.55E - 02$	+	$2.61E - 02 \pm 1.14E - 02$	+	$3.06E - 11 \pm 1.50E - 10$	+	$0.00E + 00 \pm 0.00E + 00$	+	$0.00E + 00 \pm 0.00E + 00$	+
F_{ras}	$3.01E + 00 \pm 7.57E - 01$	+	$2.76E - 01 \pm 6.51E - 01$	+	$1.63E - 01 \pm 4.14E - 01$	+	$8.52E + 01 \pm 1.15E + 01$	+	$6.98E + 01 \pm 2.72E + 01$	=	$8.15E + 01 \pm 1.73E + 01$	=
F_{sch}	$7.21E + 01 \pm 5.97E + 01$	+	$1.46E - 13 \pm 3.40E - 13$	+	$0.00E + 00 \pm 0.00E + 00$	+	$4.83E + 03 \pm 3.91E + 02$	=	$4.09E + 03 \pm 1.83E + 03$	+	$3.92E + 03 \pm 1.90E + 03$	+
F_{sal}	$1.04E - 01 \pm 2.00E - 02$	=	$9.99E - 02 \pm 1.30E - 09$	+	$9.99E - 02 \pm 4.19E - 10$	+	$6.80E - 01 \pm 7.64E - 02$	+	$3.72E - 01 \pm 4.58E - 02$	+	$3.68E - 01 \pm 4.73E - 02$	+
F_{vht}	$2.33E + 01 \pm 7.25E + 00$	+	$7.27E + 00 \pm 6.45E + 00$	+	$9.08E + 00 \pm 8.09E + 00$	+	$1.32E + 03 \pm 2.61E + 02$	+	$1.11E + 03 \pm 3.25E + 02$	+	$1.23E + 03 \pm 2.52E + 02$	+
F_{pn1}	$1.14E - 18 \pm 2.69E - 18$	+	$4.71E - 32 \pm 1.12E - 47$	+	$4.71E - 32 \pm 1.12E - 47$	+	$3.67E - 04 \pm 1.08E - 03$	+	$1.27E - 32 \pm 7.83E - 33$	+	$5.11E - 26 \pm 1.78E - 25$	+
F_{pn2}	$1.05E - 23 \pm 1.93E - 23$	+	$1.35E - 32 \pm 5.59E - 48$	+	$1.35E - 32 \pm 5.59E - 48$	+	$1.70E - 14 \pm 3.17E - 14$	+	$1.35E - 32 \pm 2.47E - 34$	+	$8.04E - 28 \pm 1.61E - 27$	+
F_1	$7.01E - 26 \pm 1.19E - 25$	+	$0.00E + 00 \pm 0.00E + 00$	+	$0.00E + 00 \pm 0.00E + 00$	+	$1.56E - 16 \pm 1.16E - 16$	+	$0.00E + 00 \pm 0.00E + 00$	+	$0.00E + 00 \pm 0.00E + 00$	+
F_2	$7.59E - 14 \pm 5.16E - 14$	+	$3.68E - 22 \pm 4.68E - 22$	+	$2.09E - 21 \pm 2.21E - 21$	+	$2.49E + 02 \pm 1.06E + 02$	-	$4.11E + 02 \pm 1.23E + 02$	-	$6.10E + 02 \pm 1.90E + 02$	-
F_3	$2.08E + 02 \pm 1.18E + 02$	-	$9.68E + 02 \pm 6.11E + 02$	-	$8.05E + 02 \pm 3.55E + 02$	-	$8.59E + 06 \pm 1.47E + 06$	-	$1.58E + 07 \pm 3.62E + 06$	-	$1.58E + 07 \pm 4.47E + 06$	-
F_4	$8.09E - 06 \pm 7.69E - 06$	+	$7.83E - 12 \pm 9.70E - 12$	+	$9.72E - 12 \pm 1.15E - 11$	+	$2.78E + 04 \pm 4.01E + 03$	+	$1.60E + 04 \pm 2.66E + 03$	+	$1.78E + 04 \pm 3.38E + 03$	+
F_5	$1.37E - 04 \pm 1.02E - 04$	+	$4.07E - 12 \pm 1.91E - 12$	+	$3.57E - 12 \pm 8.27E - 13$	+	$1.09E + 04 \pm 1.53E + 03$	+	$6.11E + 03 \pm 7.65E + 02$	+	$6.25E + 03 \pm 8.47E + 02$	+
F_6	$1.10E + 00 \pm 1.40E + 00$	+	$2.21E - 01 \pm 9.43E - 01$	+	$2.86E - 01 \pm 9.04E - 01$	+	$6.01E + 01 \pm 2.93E + 01$	+	$4.25E + 01 \pm 3.14E + 01$	+	$4.40E + 01 \pm 2.58E + 01$	+
F_7	$5.50E - 02 \pm 2.20E - 02$	=	$6.35E - 02 \pm 2.59E - 02$	=	$4.91E - 02 \pm 2.15E - 02$	=	$8.91E - 04 \pm 1.61E - 03$	=	$5.90E - 03 \pm 1.02E - 02$	=	$2.33E - 03 \pm 4.97E - 03$	=
F_8	$2.04E + 01 \pm 7.27E - 02$	=	$2.04E + 01 \pm 7.36E - 02$	=	$2.04E + 01 \pm 7.90E - 02$	=	$2.11E + 01 \pm 2.16E - 02$	=	$2.11E + 01 \pm 3.13E - 02$	+	$2.11E + 01 \pm 2.99E - 02$	+
F_9	$2.88E + 00 \pm 1.10E + 00$	+	$2.30E - 01 \pm 4.71E - 01$	+	$1.50E - 01 \pm 5.68E - 01$	+	$1.18E + 02 \pm 1.18E + 01$	+	$8.25E + 01 \pm 1.61E + 01$	+	$8.38E + 01 \pm 1.57E + 01$	+
F_{10}	$2.08E + 01 \pm 5.23E + 00$	+	$1.47E + 01 \pm 4.90E + 00$	+	$1.14E + 01 \pm 4.09E + 00$	+	$3.84E + 02 \pm 4.37E + 01$	+	$2.75E + 02 \pm 2.53E + 01$	+	$2.50E + 02 \pm 2.42E + 01$	+
+/-	16/3/1		17/2/1		14/4/2		15/3/2

FIGURE 1: Convergence curves of CS, rHFCS, and cHFCS.

TABLE 5: Results of the multiple-problem Wilcoxon's test for HFCS and CS for 20 functions at $D = 10$ and $D = 50$.

	Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
$D = 10$	rHFCS versus CS	180.000	30.000	0.005111	+	+
	cHFCS versus CS	190.000	20.000	0.001507	+	+
$D = 50$	rHFCS versus CS	168.000	42.000	0.018675	+	+
	cHFCS versus CS	167.000	43.000	0.020633	+	+

CS on 16 out of 20 functions, shows equivalence to CS on 3 out of 20 functions, and performs worse than CS on 1 out of 20 ones. cHFCS wins CS on 17 out of 20 functions, ties CS on 2 out of 20 ones, and loses CS on 1 out of 20 ones. Additionally, rHFCS and cHFCS gain higher R^+ value than R^- value markedly with the help of the results listed in Table 5.

When $D = 50$, HFCS still can bring solutions with higher quality to most of functions. For instance, rHFCS achieves the better solutions except for F_2 , F_3 , F_7 , and F_8 , while cHFCS does well except for F_2 , F_3 , F_7 , and F_8 . rHFCS outperforms, draws a tie, and loses CS on 14, 4, and 2 out of 20 functions, respectively. cHFCS is superior to CS on 15 out of 20 functions, is equal to CS on 3 out of 20 functions, and

is inferior to CS on 2 out of 20 ones. Moreover, by the aid of the results reported in Table 5, rHFCS and cHFCS also obtain remarkable higher R^+ value than R^- value.

Summarily, it suggests that the improvement of HFCS is stable when the dimensionality of problems increases.

4.3. Comparison with VCS and CCS. The comparison is made between HFCS with VCS and CCS due to the random sequence and the chaotic sequence used in these two algorithms, respectively. Table 6 lists the Error obtained by VCS and rHFCS, while Table 7 shows the Error achieved by cHFCS and CCS where CCS employs the chaotic sequence as factor in (4) and the parameter p_a is 0.25, termed as fCCS. Table 8 reports the results of the multiple-problem Wilcoxon's test between HFCS and VCS and fCCS for all functions.

It can be observed from Table 6 that the hybrid factor of the constant and the random sequence overall perform better than the random sequence. As for unimodal functions, compared with VCS, rHFCS brings solutions with higher accuracy. Moreover, as for multimodal functions, rHFCS not only keeps the same Error as VCS for F_{grw} , F_{pn1} , and F_{pn2} , but also enhances the accuracy except for F_{ack} and F_{sal} . Additionally, as for rotated and/or shifted functions, the same conclusion on multimodal functions can be drawn on rHFCS. In all, rHFCS outperforms VCS on 7 out of 20 functions, ties VCS on 11 out of 20 functions, and loses VCS on 2 out of

TABLE 6: Error obtained by VCS and rHFCS at $D = 30$.

	VCS		rHFCS	
	AVG _{Er} ± STD _{Er}	p value	AVG _{Er} ± STD _{Er}	
F_{sph}	4.53E - 49 ± 7.48E - 49	+	0.000012	3.10E - 67 ± 3.87E - 67
F_{ros}	4.67E + 00 ± 2.33E + 00	+	0.000018	6.47E - 01 ± 1.27E + 00
F_{ack}	7.25E - 15 ± 7.11E - 16	=	0.500000	7.67E - 15 ± 1.97E - 15
F_{grw}	0.00E + 00 ± 0.00E + 00	=	1.000000	0.00E + 00 ± 0.00E + 00
F_{ras}	1.78E + 01 ± 6.96E + 00	+	0.007423	1.16E + 01 ± 5.98E + 00
F_{sch}	5.61E + 02 ± 3.38E + 02	+	0.037043	3.57E + 02 ± 3.08E + 02
F_{sal}	2.24E - 01 ± 4.36E - 02	-	0.009417	2.60E - 01 ± 5.00E - 02
F_{wht}	2.42E + 02 ± 6.47E + 01	+	0.003507	1.81E + 02 ± 9.33E + 01
F_{pn1}	1.57E - 32 ± 5.59E - 48	=	1.000000	1.57E - 32 ± 5.59E - 48
F_{pn2}	1.35E - 32 ± 5.59E - 48	=	1.000000	1.35E - 32 ± 5.59E - 48
F_1	0.00E + 00 ± 0.00E + 00	=	1.000000	0.00E + 00 ± 0.00E + 00
F_2	1.44E - 02 ± 1.27E - 02	+	0.000020	1.84E - 03 ± 1.84E - 03
F_3	3.34E + 06 ± 9.76E + 05	=	0.353258	3.02E + 06 ± 1.10E + 06
F_4	4.45E + 02 ± 2.74E + 02	=	0.097970	3.27E + 02 ± 1.82E + 02
F_5	1.84E + 03 ± 7.73E + 02	=	0.121828	1.46E + 03 ± 6.61E + 02
F_6	1.91E + 01 ± 2.65E + 01	+	0.004162	7.65E + 00 ± 1.87E + 01
F_7	3.46E - 03 ± 6.55E - 03	=	0.275832	6.37E - 03 ± 8.33E - 03
F_8	2.09E + 01 ± 4.42E - 02	=	0.696425	2.09E + 01 ± 5.36E - 02
F_9	1.74E + 01 ± 6.33E + 00	=	0.210872	1.54E + 01 ± 7.09E + 00
F_{10}	9.87E + 01 ± 1.50E + 01	-	0.034670	1.11E + 02 ± 1.98E + 01
+/-/-		7/11/2		

20 ones. Additionally, rHFCS gets higher R^+ value than R^- value, and rHFCS and VCS are of significant difference at two significant levels. It suggests that rHFCS is overall better than VCS.

Table 7 shows that the hybrid factor of the constant and the chaotic sequence overall perform better than the chaotic sequence. Compared with fCCS, cHFCS works better on unimodal functions. As for multimodal functions, cHFCS gains the same performance as fCCS for F_{grw} , F_{pn1} , and F_{pn2} and increases the accuracy of solutions to F_{ras} , F_{sch} , and F_{wht} . As for rotated and/or shifted functions, cHFCS performs

TABLE 7: Error obtained by fCCS and cHFCS at $D = 30$.

	fCCS		cHFCS	
	AVG _{Er} ± STD _{Er}	p value	AVG _{Er} ± STD _{Er}	
F_{sph}	1.63E - 52 ± 1.87E - 52	+	0.000012	2.38E - 60 ± 2.95E - 60
F_{ros}	3.64E + 00 ± 2.53E + 00	+	0.001079	3.29E + 00 ± 1.41E + 01
F_{ack}	7.11E - 15 ± 0.00E + 00	=	1.000000	7.39E - 15 ± 1.42E - 15
F_{grw}	0.00E + 00 ± 0.00E + 00	=	1.000000	0.00E + 00 ± 0.00E + 00
F_{ras}	1.60E + 01 ± 6.52E + 00	=	0.509755	1.54E + 01 ± 5.67E + 00
F_{sch}	7.60E + 02 ± 4.90E + 02	=	0.300241	5.66E + 02 ± 3.97E + 02
F_{sal}	2.12E - 01 ± 3.32E - 02	-	0.005355	2.48E - 01 ± 5.10E - 02
F_{wht}	2.48E + 02 ± 5.93E + 01	+	0.047967	2.02E + 02 ± 7.95E + 01
F_{pn1}	1.57E - 32 ± 5.59E - 48	=	1.000000	1.57E - 32 ± 5.59E - 48
F_{pn2}	1.35E - 32 ± 5.59E - 48	=	1.000000	1.35E - 32 ± 5.59E - 48
F_1	0.00E + 00 ± 0.00E + 00	=	1.000000	0.00E + 00 ± 0.00E + 00
F_2	9.96E - 03 ± 6.91E - 03	+	0.000194	3.30E - 03 ± 3.47E - 03
F_3	2.59E + 06 ± 7.35E + 05	=	0.353258	2.83E + 06 ± 8.65E + 05
F_4	3.68E + 02 ± 2.46E + 02	=	0.946369	3.70E + 02 ± 2.21E + 02
F_5	1.76E + 03 ± 6.68E + 02	=	0.078001	2.16E + 03 ± 6.52E + 02
F_6	1.88E + 01 ± 2.40E + 01	+	0.000016	6.39E - 01 ± 1.23E + 00
F_7	3.96E - 03 ± 7.53E - 03	-	0.028314	1.45E - 02 ± 1.76E - 02
F_8	2.09E + 01 ± 6.14E - 02	=	0.819095	2.09E + 01 ± 6.50E - 02
F_9	1.91E + 01 ± 6.96E + 00	+	0.047967	1.53E + 01 ± 5.74E + 00
F_{10}	1.01E + 02 ± 1.68E + 01	=	0.051087	9.53E + 01 ± 1.38E + 01
+/-/-		6/12/2		

TABLE 8: Results of the multiple-problem Wilcoxon's test for HFCS, VCS, and fCCS for 20 functions at $D = 30$.

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
rHFCS versus VCS	164.500	45.500	0.026331	+	+
cHFCS versus fCCS	118.875	91.125	0.604645	=	=

better on F_1 , F_2 , F_6 , F_9 , and F_{10} . In summary, cHFCS is superior to fCCS on 6 out of 20 functions, is equal to fCCS on 12 out of 20 functions, and is inferior to fCCS on 2 out of 20 ones. In addition, cHFCS gains higher R^+ value than R^- value,

TABLE 9: Error obtained by OPICS and HFOPICS at $D = 30$.

Fun	OPICS	rHFOPICS	cHFOPICS
F_{sph}	1.99E - 88 ± 1.67E - 88*	2.17E - 85 ± 3.79E - 85	6.43E - 77 ± 9.35E - 77
F_{ros}	4.94E + 00 ± 1.46E + 01	1.13E + 00 ± 4.10E + 00	3.23E - 01 ± 9.88E - 01*
F_{ack}	7.67E - 15 ± 1.97E - 15	7.25E - 15 ± 1.62E - 15	7.11E - 15 ± 0.00E + 00*
F_{grw}	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00
F_{ras}	5.17E - 01 ± 6.50E - 01	7.96E - 02 ± 2.75E - 01	3.18E - 01 ± 5.54E - 01*
F_{sch}	1.90E + 01 ± 4.43E + 01*	1.42E + 01 ± 3.93E + 01	4.74E + 01 ± 7.65E + 01
F_{sal}	3.28E - 01 ± 5.42E - 02	2.12E - 01 ± 3.32E - 02	2.08E - 01 ± 2.77E - 02*
F_{wht}	6.70E + 00 ± 1.66E + 01*	3.07E + 01 ± 2.65E + 01	3.26E + 01 ± 4.20E + 01
F_{pn1}	1.57E - 32 ± 5.59E - 48	1.57E - 32 ± 5.59E - 48	1.57E - 32 ± 5.59E - 48
F_{pn2}	1.35E - 32 ± 5.59E - 48	1.35E - 32 ± 5.59E - 48	1.35E - 32 ± 5.59E - 48
F_1	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00
F_2	4.54E - 02 ± 1.04E - 01	1.15E - 03 ± 9.03E - 04	3.50E - 03 ± 2.73E - 03*
F_3	3.48E + 06 ± 9.19E + 05	2.68E + 06 ± 7.82E + 05	3.12E + 06 ± 8.87E + 05*
F_4	8.62E + 02 ± 6.36E + 02	1.90E + 02 ± 1.51E + 02	2.85E + 02 ± 1.97E + 02*
F_5	2.47E + 03 ± 5.40E + 02	1.71E + 03 ± 6.61E + 02	1.80E + 03 ± 6.17E + 02*
F_6	1.12E + 01 ± 2.05E + 01	2.49E + 00 ± 7.82E + 00	7.54E + 00 ± 1.93E + 01*
F_7	1.60E - 03 ± 2.88E - 03*	8.30E - 03 ± 7.37E - 03	6.46E - 03 ± 6.39E - 03
F_8	2.09E + 01 ± 5.51E - 02	2.09E + 01 ± 4.97E - 02	2.09E + 01 ± 4.37E - 02
F_9	1.23E + 00 ± 1.55E + 00	6.37E - 01 ± 7.53E - 01	9.95E - 01 ± 9.53E - 01*
F_{10}	1.10E + 02 ± 1.79E + 01	7.92E + 01 ± 1.31E + 01	8.31E + 01 ± 1.82E + 01*

TABLE 10: Results of the multiple-problem Wilcoxon's test for OPICS and HFOPICS for 20 functions at $D = 30$.

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
rHFOPICS					
versus OPICS	173.125	36.875	0.010981	+	+
cHFOPICS					
versus OPICS	155.750	54.250	0.058141	=	+

although cHFCS and fCCS are not significant differences at two significant levels. It reveals that cHFCS is overall better.

4.4. Effect of Integration into Improved Algorithms. In this section, we investigate the performance of the hybrid factor integrated into improved algorithms to analyze the suitability. Considering the implementation of the improved algorithm, we choose one-position inheritance cuckoo search algorithm, called OPICS, to be integrated, resulting in rHFOPICS and cHFOPICS. rHFOPICS presents that OPICS is enhanced with the hybrid factor of the constant and the random sequence, while cHFOPICS presents that OPICS is combined with the hybrid factor of the constant and chaotic sequence. Table 9 lists Error obtained by rHFOPICS, cHFOPICS, and OPICS, where the better Error between rHFOPICS and OPICS is highlighted in boldface, and the better Error between cHFOPICS and OPICS is marked with asterisk. Table 10 lists the results of the multiple-problem Wilcoxon's test between OPICS and HFOPICS for all functions.

It can be seen from Table 7 that the hybrid factor can make the OPICS enhance the accuracy of solutions on most of functions. In terms of Error, the results can be kept when rHFOPICS and cHFOPICS solve for F_{grw} , F_{pn1} , F_{pn2} , F_1 , and F_8 . As for unimodal functions, rHFOPICS and cHFOPICS bring the higher accurate solutions to F_{ros} . As for multimodal functions, except for F_{wht} , rHFOPICS achieves the solutions with higher accuracy, while cHFOPICS does the same things except for F_{sch} and F_{wht} . As for rotated and/or shifted functions, rHFOPICS and cHFOPICS perform better than OPICS except for F_7 . Moreover, Table 8 shows that rHFOPICS gains higher R^+ value than R^- value, and rHFOPICS and OPICS are of significant difference at two significant levels. It suggests that rHFOPICS is significantly better than OPICS. We can see from Table 8 that cHFOPICS also achieves higher R^+ value than R^- value, and OPICS is significantly inferior to cHFOPICS at $\alpha = 0.1$ significant level.

5. Conclusion

In this paper, we presented a hybrid factor strategy for CS, called HFCS. The hybrid factor strategy was combined with the constant and the random sequence or the chaotic sequence. HFCS employed the dimensional distance to select dimensions to use the constant or the random sequence or the chaotic sequence. HFCS was tested on a suit of 20 benchmark functions. The results show that the hybrid factor strategy can effectively improve the performance of CS for most of functions including solution quality and convergence speed. The results also show that the hybrid factor strategy is effective and stable when the dimension of

problems increases. In addition, the hybrid factor can be easy to integrate into other improved algorithms.

There are several interesting directions for future work. Experimentally, the mean dimensional distance is used to select some dimensions; thus, the next work will be performed using different distance, for example, the median dimensional distance. Secondly, we plan to investigate the hybrid factor strategy for other improved algorithms. Last but not least, we plan to apply HFCS to some real-world optimization problems.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the Natural Science Foundation of Fujian Province of China under Grant no. 2016J01280 and the Project of Science and Technology of Fujian Department of Education of China under Grant no. 14156.

References

- [1] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NaBIC '09)*, pp. 210–214, IEEE, Coimbatore, India, December 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] F. Wang, L. G. Luo, X.-S. He, and Y. Wang, "Hybrid optimization algorithm of PSO and Cuckoo Search," in *Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC '11)*, pp. 1172–1175, Dengfeng, China, August 2011.
- [4] A. Ghodrati and S. Lotfi, "A hybrid CS/PSO algorithm for global optimization," in *Intelligent Information and Database Systems*, vol. 7198 of *Lecture Notes in Computer Science*, pp. 89–98, Springer, Berlin Heidelberg, 2012.
- [5] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, and J. Wang, "A hybrid meta-heuristic DE/CS Algorithm for UCAV path planning," *Journal of Information and Computational Science*, vol. 9, no. 16, pp. 4811–4818, 2012.
- [6] 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.
- [7] 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.
- [8] X. Y. Liu and M. L. Fu, "Cuckoo search algorithm based on frog leaping local search and chaos theory," *Applied Mathematics and Computation*, vol. 266, pp. 1083–1092, 2015.
- [9] X. Li, J. N. Wang, and M. H. Yin, "Enhancing the performance of cuckoo search algorithm using orthogonal learning method," *Neural Computing and Applications*, vol. 24, no. 6, pp. 1233–1247, 2014.
- [10] H. Q. Zheng and Y. Q. Zhou, "A cooperative coevolutionary Cuckoo search algorithm for optimization problem," *Journal of Applied Mathematics*, vol. 2013, Article ID 912056, 9 pages, 2013.
- [11] 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.
- [12] L. J. Wang, Y. W. Zhong, and Y. L. Yin, "A hybrid cooperative cuckoo search algorithm with particle swarm optimisation," *International Journal of Computing Science and Mathematics*, vol. 6, no. 1, pp. 18–29, 2015.
- [13] J. D. Huang, L. Gao, and X. Y. Li, "An effective teaching-learning-based cuckoo search algorithm for parameter optimization problems in structure designing and machining processes," *Applied Soft Computing Journal*, vol. 36, pp. 349–356, 2015.
- [14] 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.
- [15] T. Ljouad, A. Amine, and M. Rziza, "A hybrid mobile object tracker based on the modified Cuckoo Search algorithm and the Kalman Filter," *Pattern Recognition*, vol. 47, no. 11, pp. 3597–3613, 2014.
- [16] 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.
- [17] J. Wang and B. H. Zhou, "A hybrid adaptive cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation," *Neural Computing & Applications*, vol. 27, no. 6, pp. 1511–1517, 2016.
- [18] P. Mohapatra, S. Chakravarty, and P. K. Dash, "An improved cuckoo search based extreme learning machine for medical data classification," *Swarm and Evolutionary Computation*, vol. 24, pp. 25–49, 2015.
- [19] G.-G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "Chaotic cuckoo search," *Soft Computing*, vol. 20, no. 9, pp. 3349–3362, 2015.
- [20] L. Huang, S. Ding, S. H. Yu, J. Wang, and K. Lu, "Chaos-enhanced Cuckoo search optimization algorithms for global optimization," *Applied Mathematical Modelling*, vol. 40, no. 5–6, pp. 3860–3875, 2016.
- [21] B. Jia, B. Yu, Q. Wu, C. Wei, and R. Law, "Adaptive affinity propagation method based on improved cuckoo search," *Knowledge-Based Systems*, vol. 111, pp. 27–35, 2016.
- [22] L. Wang, Y. Yin, and Y. Zhong, "Cuckoo search with varied scaling factor," *Frontiers of Computer Science*, vol. 9, no. 4, pp. 623–635, 2015.
- [23] L. J. Wang and Y. W. Zhong, "Cuckoo search algorithm with chaotic maps," *Mathematical Problems in Engineering*, vol. 2015, Article ID 715635, 14 pages, 2015.
- [24] L. D. S. Coelho, C. E. Klein, S. L. Sabat, and V. C. Mariani, "Optimal chiller loading for energy conservation using a new differential cuckoo search approach," *Energy*, vol. 75, pp. 237–243, 2014.
- [25] U. Mlakar, I. Fister Jr., and I. Fister, "Hybrid self-adaptive cuckoo search for global optimization," *Swarm and Evolutionary Computation*, vol. 29, pp. 47–72, 2016.
- [26] X. Ding, Z. Xu, N. J. Cheung, and X. Liu, "Parameter estimation of Takagi-Sugeno fuzzy system using heterogeneous cuckoo search algorithm," *Neurocomputing*, vol. 151, no. 3, pp. 1332–1342, 2015.

- [27] L. Wang, Y. Zhong, and Y. Yin, "Nearest neighbour cuckoo search algorithm with probabilistic mutation," *Applied Soft Computing*, vol. 49, pp. 498–509, 2016.
- [28] L. J. Wang and Y. W. Zhong, "One-position inheritance based cuckoo search algorithm," *International Journal of Computing Science and Mathematics*, vol. 6, no. 6, pp. 546–554, 2015.
- [29] X. T. Li and M. H. Yin, "A particle swarm inspired cuckoo search algorithm for real parameter optimization," *Soft Computing*, vol. 20, no. 4, pp. 1389–1413, 2016.
- [30] L. J. Wang, Y. W. Zhong, and Y. L. Yin, "Orthogonal crossover cuckoo search algorithm with external archive," *Journal of Computer Research and Development*, vol. 52, no. 11, pp. 2496–2507, 2015.
- [31] 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.
- [32] X. Li and M. Yin, "Modified cuckoo search algorithm with self adaptive parameter method," *Information Sciences*, vol. 298, pp. 80–97, 2015.
- [33] X.-S. Yang and S. Deb, "Multiobjective cuckoo search for design optimization," *Computers and Operations Research*, vol. 40, no. 6, pp. 1616–1624, 2013.
- [34] S. Hanoun, D. Creighton, and S. Nahavandi, "A hybrid cuckoo search and variable neighborhood descent for single and multiobjective scheduling problems," *The International Journal of Advanced Manufacturing Technology*, vol. 75, no. 9–12, pp. 1501–1516, 2014.
- [35] K. Chandrasekaran and S. P. Simon, "Multi-objective scheduling problem: hybrid approach using fuzzy assisted cuckoo search algorithm," *Swarm and Evolutionary Computation*, vol. 5, pp. 1–16, 2012.
- [36] X. X. Ouyang, Y. Q. Zhou, Q. F. Luo, and H. Chen, "A novel discrete cuckoo search algorithm for spherical traveling salesman problem," *Applied Mathematics & Information Sciences*, vol. 7, no. 2, pp. 777–784, 2013.
- [37] A. Ouaraab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Computing and Applications*, vol. 24, no. 7-8, pp. 1659–1669, 2014.
- [38] Y. Q. Zhou, H. Q. Zheng, Q. F. Luo, and J. Wu, "An improved cuckoo search algorithm for solving planar graph coloring problem," *Applied Mathematics and Information Sciences*, vol. 7, no. 2, pp. 785–792, 2013.
- [39] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan," *Applied Soft Computing Journal*, vol. 19, pp. 93–101, 2014.
- [40] P. Dasgupta and S. Das, "A discrete inter-species cuckoo search for flowshop scheduling problems," *Computers and Operations Research*, vol. 60, pp. 111–120, 2015.
- [41] E. Teymourian, V. Kayvanfar, G. Komaki, and M. Zandieh, "Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem," *Information Sciences*, vol. 334–335, pp. 354–378, 2016.
- [42] X. Jin, Y. Q. Liang, D. P. Tian, and F. Zhuang, "Particle swarm optimization using dimension selection methods," *Applied Mathematics and Computation*, vol. 219, no. 10, pp. 5185–5197, 2013.
- [43] 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.
- [44] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," KanGAL Report 2005005, Nanyang Technological University, Singapore, 2005.
- [45] 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.
- [46] 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.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>