Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2015, Article ID 583759, 11 pages
http://dx.doi.org/10.1155/2015/583759

Research Article

An Enhanced Differential Evolution with Elite Chaotic Local Search

Zhaolu Guo,[1] Haixia Huang,[2] Changshou Deng,[3] Xuezhi Yue,[1] and Zhijian Wu[4]

[1]*Institute of Medical Informatics and Engineering, School of Science, Jiangxi University of Science and Technology, Ganzhou 341000, China*
[2]*School of Literature and Law, Jiangxi University of Science and Technology, Ganzhou 341000, China*
[3]*School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China*
[4]*State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China*

Correspondence should be addressed to Zhaolu Guo; gzl@whu.edu.cn

Received 8 October 2014; Accepted 27 April 2015

Academic Editor: Rafik Aliyev

Differential evolution (DE) is a simple yet efficient evolutionary algorithm for real-world engineering problems. However, its search ability should be further enhanced to obtain better solutions when DE is applied to solve complex optimization problems. This paper presents an enhanced differential evolution with elite chaotic local search (DEECL). In DEECL, it utilizes a chaotic search strategy based on the heuristic information from the elite individuals to promote the exploitation power. Moreover, DEECL employs a simple and effective parameter adaptation mechanism to enhance the robustness. Experiments are conducted on a set of classical test functions. The experimental results show that DEECL is very competitive on the majority of the test functions.

1. Introduction

Numerous problems in science and engineering can be converted into optimization problems. Therefore, it is of significance both in theory and in engineering applications to develop effective and efficient optimization algorithms for solving complex problems of science and engineering. Differential evolution (DE), proposed by Storn and Price in 1997 [1], is a simple yet effective global optimization algorithm. According to frequently reported theoretical and experimental studies, DE has exhibited competitive performance than many other evolutionary algorithms in terms of both convergence speed and solution precision over several benchmark functions and real-life problems [2–4]. Due to its simplicity, easy implementation, and efficiency, DE has stimulated many researchers' interests since its development. Therefore, it has become a hot research topic in evolutionary computation over the past decades [5–7].

However, its search ability should be further enhanced to obtain better solutions when DE is used to solve various real-life optimization problems [2, 8, 9]. Particularly, DE may suffer from premature convergence and/or slow convergence when solving complex multimodal optimization problems. In order to improve the performance of the conventional DE, a number of DE variants have been proposed in recent decades [2, 6, 10]. Recognizing that the performance of DE depends on the control parameters, Brest et al. [11] presented a self-adaptive DE (jDE), in which both F and CR are created independently for each individual by an adaptive mechanism. Specifically, the new F is created by a random value from 0.1 to 0.9 with a probability 0.1 during the search process. Meanwhile, the new CR obtains a random value from 0.0 to 1.0 with a probability 0.1. Unlike jDE, JADE, proposed by Zhang and Sanderson [12], utilizes a distinct parameter adaptation mechanism, in which the new F and CR are created for each individual by a normal distribution and a Cauchy distribution, respectively. In addition, JADE learns knowledge from the recent successful F and CR and applies the learned knowledge for creating new F and CR. Identifying that both the mutation strategies and their associated control parameters can directly influence the performance of DE, Qin et al. [7] proposed a novel self-adaptive DE, SaDE, which adaptively tunes the trial vector generation strategies and their associated control parameter values by extracting

knowledge from the previous search process in generating promising solutions. Mallipeddi et al. [13] introduced an improved DE with ensemble of parameters and mutation strategies (EPSDE), which employs a pool of diverse trial vector generation strategies and a pool of values for the control parameters F and CR. By incorporating an opposition-based learning strategy into the traditional DE for population initialization and generating new solutions, Rahnamayan et al. [14] proposed an opposition-based DE (ODE). The experimental results confirmed that the opposition-based learning strategy can improve the convergence speed and the solution accuracy of DE. Further, Wang et al. [15] improved the opposition-based learning strategy, proposed a generalized opposition-based learning strategy, and presented an enhanced DE with generalized opposition-based learning strategy (GODE). Jia et al. [16] presented an effective memetic DE algorithm, DECLS, which utilizes a chaotic local search with a shrinking strategy to improve the search ability. Experimental results indicated that the performance of the canonical DE is significantly improved by the chaotic local search. Recently, Wang et al. [17] proposed a composite DE, called CoDE, the main idea of which is to randomly combine several well studied trial vector generation strategies with a number of control parameter settings highly recommended by other researchers at each generation to create new trial vectors. Experimental results on all the CEC2005 contest test instances show that CoDE is very competitive.

Although there already exist many DE variants for solving complex optimization problems, according to the no free lunch (NFL) theory [18], the performance of DE for some benchmark functions and real-life problems should be further enhanced to obtain better solutions. Moreover, many studies have revealed that embedding local search strategy can greatly enhance the search ability of DE [14, 16, 19]. Motivated by these considerations, in order to promote the performance of DE on complex optimization problems, this study proposes an enhanced differential evolution with elite chaotic local search, called DEECL. In DEECL, we utilize a chaotic search strategy based on the heuristic information from the elite individuals to promote the exploitation power. Further, we also design a simple and effective parameter adaptation mechanism to enhance the robustness.

The rest of the paper is organized as follows. The conventional DE is introduced in Section 2. Section 3 presents the enhanced DE. Numerical experiments are presented in Section 4 for the comparison and analysis. Finally, the paper is concluded in Section 5.

2. Differential Evolution

Without loss of generality, only minimization problems are considered in this study. We suppose that the objective function to be minimized is Min $f(X)$, $X = [X_1, X_2, \ldots, X_D]$, and the search space is

$$\Omega = \prod_{j=1}^{D} \left[\mathrm{LB}_j, \mathrm{UB}_j\right], \tag{1}$$

where D is the number of dimensions of the problem, LB_j and UB_j denote the lower and upper boundaries of the search space, respectively.

Similar to other evolutionary algorithms, DE also has a simple structure, only including three simple operators, namely, mutation, crossover, and selection operators [2]. In the initial phase, DE creates an initial population $P(t) = \{X_i^t\}$, which is randomly generated from the search space, where $X_i^t = [X_{i,1}^t, X_{i,2}^t, \ldots, X_{i,D}^t]$, $i = 1, 2, \ldots, NP$; NP is the population size and t is the generation. After initialization, the mutation and crossover operators are performed to create the trial vectors, and then the selection operator is utilized to select the better one between the offspring individual and the parent individual for the next generation. DE performs these steps repeatedly to converge toward the global optima until the terminating criterion is reached [20]. In the following subsections, the evolutionary operators of DE will be introduced in detail.

2.1. Mutation Operator. In the mutation operator, a mutant vector V_i^t is created by using a predetermined mutation strategy for each individual X_i^t, namely, target vector, in the current population [17]. DE has many mutation strategies used in its implementations, such as DE/rand/1, DE/best/1, DE/rand-to-best/1, DE/best/2 and DE/rand/2 [2]. Among these mutation strategies, DE/rand/1 is the most frequently used mutation strategy, which is expressed as follows [1]:

$$V_i^t = X_{r1}^t + F \times \left(X_{r2}^t - X_{r3}^t\right), \tag{2}$$

where $r1$, $r2$, and $r3$ are randomly selected from the set $\{1, 2, \ldots, NP\} \setminus \{i\}$, and they are mutually different from each other. F is called as scaling factor, amplifying the difference vector $X_{r2}^t - X_{r3}^t$.

2.2. Crossover Operator. Following mutation, a trial vector U_i^t is generated by executing the crossover operator for each pair of target vector X_i^t and its corresponding mutant vector V_i^t [2]. Binomial crossover is the most commonly used crossover operators in current popular DE. The binomial crossover is described as follows [1]:

$$U_{i,j}^t = \begin{cases} V_{i,j}^t, & \text{if rand}(0,1) < \mathrm{CR} \text{ or } j == j\text{rand} \\ X_{i,j}^t, & \text{otherwise,} \end{cases} \tag{3}$$

where rand$(0, 1)$ is generated for each j and takes a value from 0.0 to 1.0 in a uniformly random manner, and $\mathrm{CR} \in [0, 1]$ is the crossover probability, which limits the number of parameters inherited from the mutant vector V_i^t. The integer $j$rand is randomly chosen from the range $[1, D]$, which guarantees that at least one parameter of the trial vector U_i^t is inherited from the mutant vector V_i^t [7].

2.3. Selection Operator. Like the genetic algorithm, the selection process of DE is also based on the Darwinian law of survival of the fittest. The selection process is performed in order to choose the more excellent individuals for the

```
t = 0;
FES = 0;
/* Initialize the population */
for i = 1 to NP do
  for j = 1 to D do
    X^t_{i,j} = LB_j + rand(0, 1) × (UB_j − LB_j);
  end for
  Evaluate individual X^t_i;
  FES = FES + 1;
end for
while FES < MAX_FES do
  for i = 1 to NP do
    Choose three mutually different integers r1, r2, r3 from
    the set {1, 2, ..., NP} \ {i} in a random manner;
    jrand = randint(1, D);
    for j = 1 to D do
      if rand(0, 1) < CR or j == jrand then
        U^t_{i,j} = X^t_{r1,j} + F × (X^t_{r2,j} − X^t_{r3,j});
      else
        U^t_{i,j} = X^t_{i,j};
      end if
    end for
    /* Selection step */
    if f(U^t_i) ≤ f(X^t_i) then
      X^{t+1}_i = U^t_i;
      if f(U^t_i) < f(X^t_Best) then
        X^t_Best = U^t_i;
      end if
    else
      X^{t+1}_i = X^t_i;
    end if
    FES = FES + 1;
  end for
  t = t + 1;
end while
```

Algorithm 1: DE algorithm.

next generation. For minimization problems, the selection operator can be defined in the following form [1]:

$$X_i^{t+1} = \begin{cases} U_i^t, & \text{if } f\left(U_i^t\right) \le f\left(X_i^t\right) \\ X_i^t, & \text{otherwise,} \end{cases} \tag{4}$$

where $f(X_i^t)$ and $f(U_i^t)$ indicate the fitness values of the target vector X_i^t and its corresponding trial vector U_i^t, respectively.

2.4. Algorithmic Framework of DE. Based on the above elaborate introduction of the DE's operators, we present the framework of DE with DE/rand/1/bin strategy in Algorithm 1, where FES is the number of fitness evaluations, Max_FES is the maximum number of evaluations, rand(0,1) indicates a random real number in the range $[0, 1]$, randint$(1, D)$ represents a random integer in the range $[1, D]$, and X_{Best}^t is the global best individual found so far.

3. Proposed Approach

3.1. Motivations. DE has been demonstrated to yield superior performance for solving various real-world optimization problems [21–23]. However, it tends to suffer from premature convergence and/or slow convergence when solving complex optimization problems [6, 24]. To enhance the performance of DE, many researchers have proposed various improved DE algorithms during the past decade [25–27]. Among the DE variations, memetic method is a promising approach to improve the performance of the traditional DE, which utilizes various local search strategies, such as chaotic search strategy [16], simplex crossover search strategy [19], and orthogonal search strategy [28], to strengthen the exploitation ability of the traditional DE and consequently accelerate the convergence speed. Among the local search strategies commonly used in memetic DE, chaotic search strategy is inspired by the chaos phenomenon in nature. Chaos is a classic nonlinear dynamical system, which is widely known as a system with the properties of ergodicity, randomicity, and sensitivity to its initial conditions [16, 29, 30]. Due to its ergodicity and randomicity, a chaotic system can randomly generate a long-time sequence which is able to traverse through every state of the system and every state is generated only once if given a long enough time period [16, 31]. Taking advantage of the well-known characteristics of the chaotic systems, researchers have proposed many chaotic search strategies for optimizing various problems [16, 32–34]. However, to the best of our knowledge, among many chaotic search strategies, they pay more attention to the characteristics of the ergodicity and randomicity of the chaotic system. Therefore, the exploration capacity can be indeed improved. However, in order to maintain a balance between exploration and exploitation, the exploitation ability of the chaotic search strategy should be further enhanced. Thus, when designing a relatively comprehensive chaotic search strategy, we should further integrate more heuristic information into the chaotic search strategy to promote its exploitation power. Generally, the elite individuals in the current population known as a promising search direction toward the optimum are the favorable source that can be employed to enhance the exploitation ability. Based on these considerations, we present an elite chaotic search strategy, which not only utilizes the characteristics of the ergodicity and randomicity of the chaotic system, but also merges the superior information of the current population into the chaotic search process.

3.2. Elite Chaotic Search. In many chaotic search strategies, the Logistic chaotic function is utilized to generate a chaotic sequence, which is formulated as follows [16]:

$$\begin{aligned} &K^0 = \text{rand}\,(0, 1), \quad K^0 \neq 0.25, 0.5, 0.75, \\ &K^n = 4.0 \cdot K^{n-1} \cdot \left(1 - K^{n-1}\right), \quad n = 1, 2, \ldots, N, \end{aligned} \tag{5}$$

where K^0 is the initial value of the chaotic system, which is randomly generated from the range $[0, 1]$, but cannot be equal to 0.25, 0.5, or 0.75. K^n is the nth state of the chaotic system. As known, the initial state K^0 of the chaotic system

$n = 0$;
$N = D/5$;
Randomly choose an individual X_I^t from the current population;
$K^0 = \text{rand}(0, 1), K^0 \neq 0.25, 0.5, 0.75$;
$p = \text{rand}(2.0/NP, 0.1)$;
while $n < N$ do
  Randomly choose an individual $X_{p\text{Best}}^t$ from the top $100p\%$ individuals in the current population;
  for $j = 1$ to D do
    $E_{I,j}^n = X_{I,j}^t + K^n \times (X_{p\text{Best},j}^t - X_{I,j}^t)$;
    if $E_{I,j}^n > \text{UB}_j$ or $E_{I,j}^n < \text{LB}_j$ then
      $E_{I,j}^n = \text{rand}(\text{LB}_j, \text{UB}_j)$;
    end if
  end for
  if $f(E_I^n) < f(X_I^t)$ then
    $X_I^t = E_I^n$;
    break;
  end if
  $n = n + 1$;
  $K^n = 4.0 \cdot K^{n-1} \cdot (1 - K^{n-1})$;
  FES = FES + 1;
end while

Algorithm 2: Elite chaotic search operator.

is randomly produced. Due to its ergodicity and sensitivity to the initial state K^0, K^n is a random long-time sequence, which can traverse through every state of the system and every state is generated only once if N is large enough.

In order to enhance the exploitation ability of the traditional chaotic search strategy, we integrate the heuristic information learned from the elite individuals into the chaotic search strategy to promote the exploitation power. The proposed elite chaotic search strategy is defined by

$$E_I^n = X_I^t + K^n \times \left(X_{p\text{Best}}^t - X_I^t\right), \tag{6}$$

where X_I^t is an individual to be performed the elite chaotic search, which is randomly chosen from the current population. K^n is the chaotic sequence, where $n = 1, 2, \ldots, N$, $N = D/5$, and $X_{p\text{Best}}^t$ is an elite individual, which is randomly chosen from the top $100p\%$ individuals in the current population with $p = \text{rand}(2.0/NP, 0.1)$.

In the proposed elite chaotic search operator, an individual X_I^t is randomly selected from the current population to undergo the elite chaotic search strategy. After that, the initial value of the chaotic system takes a value from range $[0, 1.0]$ in a uniformly random manner. Then, an elite chaotic search procedure for individual X_I^t is repeatedly performed until finding a better solution than individual X_I^t or the number of iterations n is equal to N. The framework of the elite chaotic search operator is described in Algorithm 2.

3.3. Parameter Adaptation. Since the setting of control parameters can significantly influence the performance of DE, parameter adaptation mechanism is essential for an efficient DE [7, 11, 12]. To this end, we design a simple and effective parameter adaptation mechanism inspired by [11] into DEECL. In DEECL, each individual is independently associated with its own mutation factor F_i^t and crossover probability CR_i^t. For individual i, its control parameters F_i^t and CR_i^t are initialized to 0.5 and 0.9, respectively. Generally, a normal distribution with mean value 0.5 and standard deviation 0.3 is a promising adaptive approach for the mutation factor of DE [7], whereas Cauchy distribution is more favorable to diversify the mutation factors and thus avoid premature convergence [12]. Based on these considerations, at each generation, the new mutation factor NF_i^t associated with individual i is generated by a Cauchy distribution random real number with location parameter 0.5 and scale parameter 0.3 with probability 0.1. Additionally, following the suggestions in [11], the new crossover probability NCR_i^t associated with individual i acquires a random value from 0.0 to 1.0 with probability 0.1. Mathematically, the new control parameters NF_i^t and NCR_i^t associated with individual i for generating its corresponding trial vector U_i^t are obtained by

$$\begin{aligned} \text{NF}_i^t &= \begin{cases} \text{randc}\,(0.5, 0.3)\,, & \text{if rand}\,(0, 1) < 0.1 \\ F_i^t, & \text{otherwise,} \end{cases} \\ \text{NCR}_i^t &= \begin{cases} \text{rand}\,(0, 1)\,, & \text{if rand}\,(0, 1) < 0.1 \\ \text{CR}_i^t, & \text{otherwise,} \end{cases} \end{aligned} \tag{7}$$

where $\text{randc}(0.5, 0.3)$ is a Cauchy distribution random real number with location parameter 0.5 and scale parameter 0.3 and $\text{rand}(0, 1)$ is a uniformly random number within the range $[0, 1]$. After obtaining the new control parameters NF_i^t and NCR_i^t, the corresponding trial vector U_i^t are created

Table 1: The 13 classical test functions.

Function	Name	Initial range	$f_{\min}$
$f1$	Sphere Problem	$[-100, 100]^D$	0
$f2$	Schwefel's Problem 2.22	$[-10, 10]^D$	0
$f3$	Schwefel's Problem 1.2	$[-100, 100]^D$	0
$f4$	Schwefel's Problem 2.21	$[-100, 100]^D$	0
$f5$	Rosenbrock's Function	$[-30, 30]^D$	0
$f6$	Step Function	$[-100, 100]^D$	0
$f7$	Quartic Function with Noise	$[-1.28, 1.28]^D$	0
$f8$	Schwefel's Problem 2.26	$[-500, 500]^D$	0
$f9$	Rastrigin's Function	$[-5.12, 5.12]^D$	0
$f10$	Ackley's Function	$[-32, 32]^D$	0
$f11$	Griewank Function	$[-600, 600]^D$	0
$f12$	Penalized Function 1	$[-50, 50]^D$	0
$f13$	Penalized Function 2	$[-50, 50]^D$	0

by using the new control parameters NF_i^t and NCR_i^t. It is widely acknowledged that better control parameter values tend to produce better individuals that have a greater chance to survive and thus these values should be propagated to the next generations [12]. Therefore, in the selection step, the control parameters F_i^{t+1} and CR_i^{t+1} associated with individual i for the next generation are updated by

$$F_i^{t+1} = \begin{cases} \mathrm{NF}_i^t, & \text{if } f\left(U_i^t\right) < f\left(X_i^t\right) \\ F_i^t, & \text{otherwise,} \end{cases}$$
$$\mathrm{CR}_i^{t+1} = \begin{cases} \mathrm{NCR}_i^t, & \text{if } f\left(U_i^t\right) < f\left(X_i^t\right) \\ \mathrm{CR}_i^t, & \text{otherwise.} \end{cases} \quad (8)$$

From the above designed parameter adaptation mechanism, we can infer that the better control parameters of DEECL can be propagated to the next generations. Therefore, the control parameters of DEECL can be adaptively tuned according to the feedback from the search process.

4. Numerical Experiments

4.1. Experimental Setup. In order to assess the performance of the proposed DEECL, we use 13 classical test functions ($f1$–$f13$) that are widely used in the evolutionary computation community [8, 12, 35] to verify the effectiveness of the proposed DEECL. We describe these test functions in Table 1. Among these test functions [35], $f1$–$f4$ are continuous unimodal functions. $f5$ is the Rosenbrock function which is unimodal for $D = 2$ and 3; however, it may have multiple minima in high dimension cases [36]. $f6$ is a discontinuous step function, and $f7$ is a noisy function. $f8$–$f13$ are multimodal functions and they exist many local minima [35].

In all experiments, we set the number of dimensions D to 30 for all these test functions. We carry out 30 independent runs for each algorithm and each test function with 150,000 function evaluations (FES) as the termination criterion. Moreover, we record the average and standard deviation of the function error value ($f(x) - f(x^*)$) for estimating the performance of the algorithms, as recommended by [17], where x is the best solution gained by the algorithm in a run and x^* is the global optimum of the test function.

4.2. Benefit of the Two Components. There are two important components in the proposed DEECL: the proposed elite chaotic search strategy and the designed parameter adaptation mechanism. Accordingly, it is interesting to recognize the benefit of the two components of the proposed DEECL. For this purpose, we conduct experiments to compare the proposed DEECL with the traditional DE with DE/rand/1 strategy and two variants of DEECL, namely, DE with the proposed elite chaotic search strategy (DEwEC) and DE with the designed parameter adaptation mechanism (DEwPA). In the experiments, we set the population size of all the algorithms to 100. For the other parameters of DE and DEwEC, we set $F = 0.5$ and $\mathrm{CR} = 0.9$, following the suggestions in [11].

We present the experimental results of the above mentioned algorithms in Table 2. The best results among the four algorithms are highlighted in *boldface*. "Mean Error" and "Std Dev" indicate the mean and standard deviation of the function error values achieved in 30 independent runs, respectively. From the results of comparison between DE and DEwEC, DEwEC performs better than DE on all test functions with the exception of $f6$. On test function $f6$, both DE and DEwEC exhibit similar performance. In total, DEwEC is better than DE on twelve test functions. The results of comparison between DE and DEwEC indicate that our introduced elite chaotic search strategy is effective to enhance the performance of the traditional DE.

From the comparison of DE with DEwPA, DEwPA surpasses DE on all test functions except for $f5$ and $f6$. On test function $f6$, both DE and DEwPA demonstrate similar performance, whereas DE is better than DEwPA on test function $f5$. In summary, DEwPA outperforms DE on eleven test functions. The comparison of DE with DEwPA reveals that our designed parameter adaptation mechanism is capable of improving the efficiency of the traditional DE.

By incorporation of both the proposed elite chaotic search strategy and the designed parameter adaptation mechanism, DEECL achieves promising performance, which is better than other three DE algorithms on the majority of the test functions. To be specific, DEECL is better than DE, DEwEC, and DEwPA on eleven, nine, and ten test functions, respectively. DE, DEwEC, and DEwPA can outperform DEECL only on one test function. Comparison results suggest that both the introduced elite chaotic search strategy and the designed parameter adaptation mechanism demonstrate positive effect on the performance of DEECL. In addition, the comparison results confirm that the introduced elite chaotic search strategy and the designed parameter adaptation mechanism can help DE with both outperform DE with either or neither one on the majority of the test functions. Moreover, the introduced elite chaotic search strategy and the designed parameter adaptation mechanism work together to improve the performance of the traditional DE rather than contradict each other. The evolution of the average function error values derived from DE, DEwEC, DEwPA, and DEECL versus

Table 2: Experimental results of DE, DEwEC, DEwPA, and DEECL over 30 independent runs for the 13 test functions.

Function	DE	DEwEC	DEwPA	DEECL
	Mean ± Std Dev	Mean ± Std Dev	Mean ± Std Dev	Mean ± Std Dev
$f1$	$2.23E-16 \pm 2.50E-16$	$4.38E-24 \pm 3.95E-24$	$4.14E-33 \pm 4.76E-33$	**6.89E − 38 ± 6.06E − 38**
$f2$	$2.86E-08 \pm 1.26E-08$	$1.63E-12 \pm 6.75E-13$	$3.58E-20 \pm 2.80E-20$	**1.74E − 22 ± 1.21E − 22**
$f3$	$1.88E-01 \pm 6.12E-02$	$9.12E-02 \pm 1.03E-02$	$5.25E-02 \pm 5.94E-02$	**2.42E − 02 ± 3.44E − 02**
$f4$	$1.70E-01 \pm 2.13E-01$	$1.51E-02 \pm 2.10E-02$	$3.23E-04 \pm 1.55E-04$	**4.06E − 05 ± 3.05E − 05**
$f5$	$1.39E+01 \pm 8.74E-01$	**1.27E + 01 ± 5.12E + 01**	$2.46E+01 \pm 1.58E+01$	$2.95E+01 \pm 2.21E+01$
$f6$	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**
$f7$	$8.82E-03 \pm 2.61E-03$	$2.15E-03 \pm 8.72E-04$	$6.52E-03 \pm 1.59E-03$	**1.17E − 03 ± 6.52E − 04**
$f8$	$7.31E+03 \pm 3.75E+02$	$5.04E+03 \pm 3.25E+02$	$1.53E-02 \pm 1.82E-12$	**1.34E − 02 ± 1.19E − 12**
$f9$	$1.77E+02 \pm 1.10E+01$	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**
$f10$	$5.93E-09 \pm 3.10E-09$	$5.22E-13 \pm 1.76E-13$	$4.35E-15 \pm 1.07E-15$	**4.00E − 15 ± 0.00E + 00**
$f11$	$6.33E-16 \pm 1.16E-15$	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**
$f12$	$2.20E-17 \pm 1.81E-17$	$2.21E-25 \pm 1.38E-25$	$1.61E-30 \pm 7.69E-48$	**1.57E − 32 ± 2.74E − 48**
$f13$	$8.26E-17 \pm 3.59E-17$	$4.23E-24 \pm 6.14E-24$	$1.46E-32 \pm 2.02E-33$	**1.36E − 32 ± 3.70E − 34**

the number of FES is plotted in Figure 1 for some typical test functions. As can be seen from Figure 1, DEECL converges faster than DE, DEwEC, and DEwPA.

4.3. Comparison with Other DE Variants. In order to verify the effectiveness of the proposed DEECL algorithm, we compare DEECL with the traditional DE and three other DE variants, namely, jDE [11], ODE [14], and DECLS [16]. In addition, jDE is a self-adaptive DE, in which both parameters F and CR are generated independently for each individual by an adaptive mechanism [11]. ODE is proposed by Rahnamayan et al. [14], which incorporates the opposition-based learning strategy into the traditional DE for population initialization and creating new solutions. DECLS is an effective memetic DE algorithm [16], which utilizes the chaotic local search strategy and an adaptive parameter control approaches similar to jDE [11] to improve the search ability. In the experiments, in order to have a fair comparison, we set the population size of all the algorithms to 100. The other parameter settings of these three DE variants are the same as in their original papers.

The mean and standard deviation of the function error values achieved by each algorithm for the 13 classical test functions are presented in Table 3. For convenience of analysis, the best results among the four DE algorithms are highlighted in *boldface*. In order to gain statistically significant conclusions, we conduct two-tailed t-tests at the significance level of 0.05 [28, 35] on the experimental results. The summary comparison results are described in the last three rows of Table 3. "+," "−," and "≈" suggest that DEECL is better than, worse than, and similar to the corresponding algorithm in terms of the two-tailed t-tests at the significance level of 0.05, respectively.

From Table 3, we can infer that DEECL achieves the better results than all the other four algorithms on the majority of the 13 classical test functions. Specifically, DEECL is significantly better than DE, jDE, ODE, and DECLS on eleven, seven, nine, and six test functions according to the two-tailed t-test, respectively. In addition, DEECL is similar to DE, jDE, ODE, and DECLS on one, five, two, and five test functions, respectively. DE and jDE surpasses DEECL only on one test function. Additionally, ODE and DECLS perform better than DEECL only on two test functions.

Overall, DEECL performs better than the traditional DE, jDE, ODE, and DECLS on the majority of the test functions. This can be because the proposed elite chaotic search strategy learning the heuristic information from the elite individuals can promote the exploitation power, and the designed parameter adaptation mechanism can enhance the robustness. The evolution of the average function error values derived from DE, jDE, ODE, DECLS, and DEECL versus the number of FES is plotted in Figure 2 for some typical test functions. It can be known from Figure 2 that DEECL converges faster than DE, jDE, ODE, and DECLS.

In order to compare the total performance of the five DE algorithms on the all 13 classical test functions, we carry out the average ranking of Friedman test on the experimental results following the suggestions in [37–39]. Table 4 presents the average ranking of the five DE algorithms on the all 13 classical test functions. We can sort these five DE algorithms by the average ranking into the following order: DEECL, DECLS, jDE, ODE, and DE. Therefore, DEECL obtains the best average ranking, and its total performance is better than that of the other four algorithms on the all 13 test instances.

5. Conclusions

DE is a popular evolutionary algorithm for the continuous global optimization problems, which has a simple structure yet exhibits efficient performance on various real-world engineering problems. However, according to the no free lunch (NFL) theory, the performance of DE should be further enhanced to obtain better solutions in some cases. In this paper, we propose an enhanced differential evolution with elite chaotic local search, called DEECL, which uses a chaotic search strategy based on the heuristic information

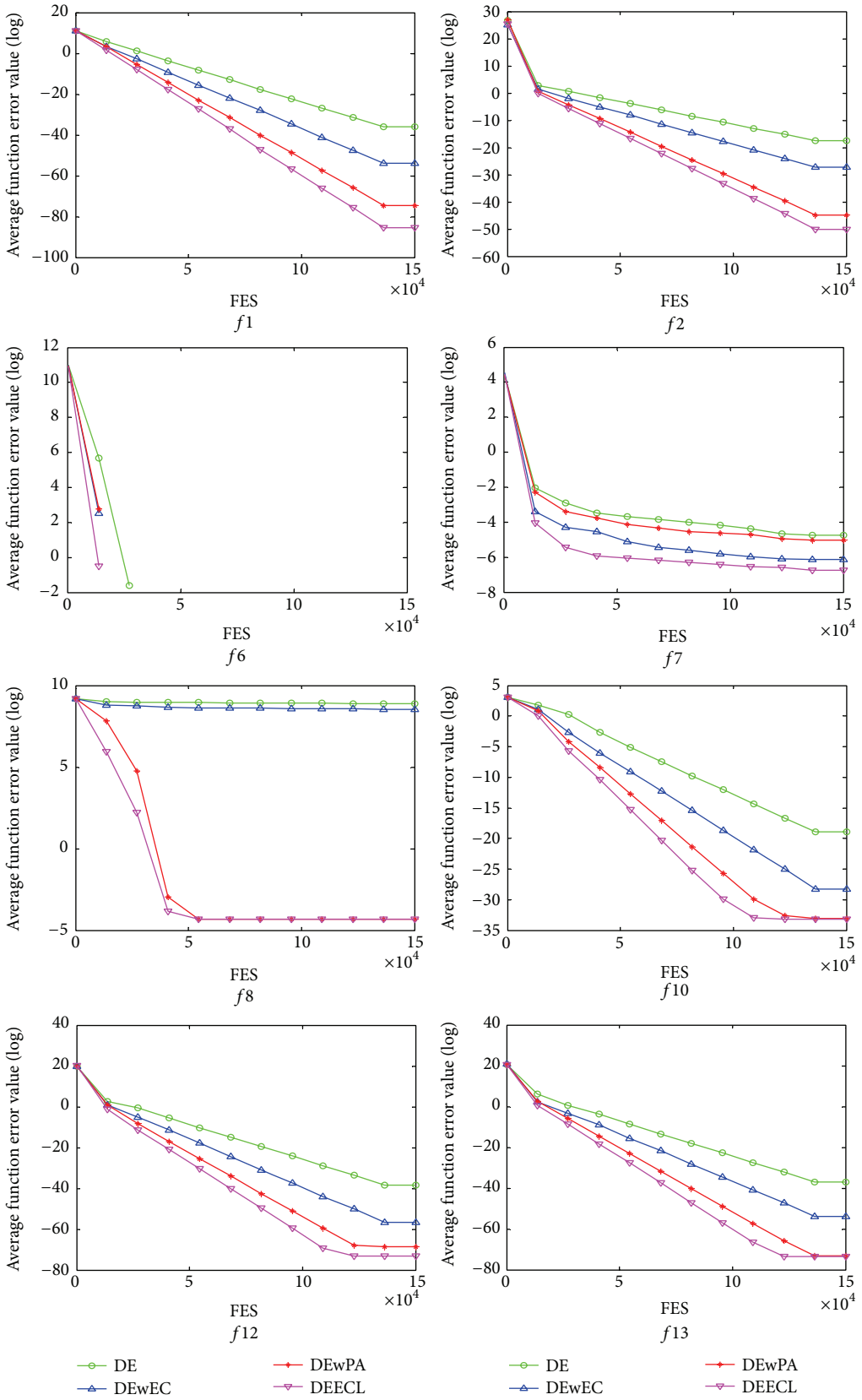

Figure 1: Evolution of the average function error values derived from DE, DEwEC, DEwPA, and DEECL versus the number of FES.

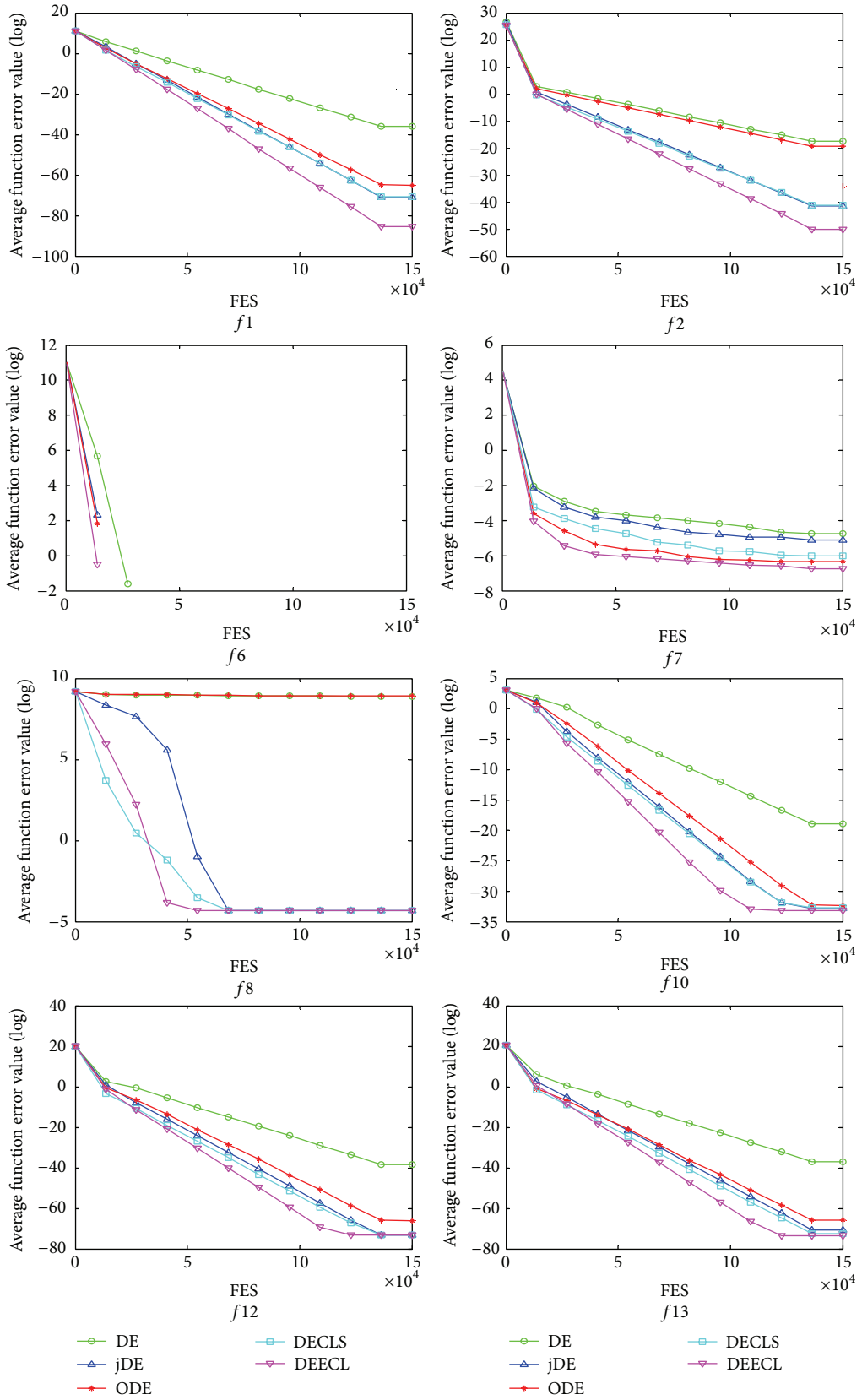

Figure 2: Evolution of the average function error values derived from DE, jDE, ODE, DECLS, and DEECL versus the number of FES.

Table 3: Experimental results of DE, jDE, ODE, DECLS, and DEECL over 30 independent runs for the 13 test functions.

Function	DE Mean ± Std Dev	jDE Mean ± Std Dev	ODE Mean ± Std Dev	DECLS Mean ± Std Dev	DEECL Mean ± Std Dev
$f1$	$2.23E-16 \pm 2.50E-16+$	$1.51E-31 \pm 1.82E-31+$	$6.20E-29 \pm 3.92E-29+$	$2.01E-31 \pm 2.22E-31+$	**$6.89E-38 \pm 6.06E-38$**
$f2$	$2.86E-08 \pm 1.26E-08+$	$9.13E-19 \pm 3.70E-19+$	$4.31E-09 \pm 2.61E-09+$	$1.46E-18 \pm 9.36E-19+$	**$1.74E-22 \pm 1.21E-22$**
$f3$	$1.88E-01 \pm 6.12E-02+$	$1.85E-02 \pm 6.45E-03\approx$	$1.45E-01 \pm 1.17E-01+$	**$5.39E-04 \pm 5.78E-04-$**	$2.42E-02 \pm 3.44E-02$
$f4$	$1.70E-01 \pm 2.13E-01+$	$3.46E-04 \pm 1.23E-04+$	**$1.14E-07 \pm 3.43E-07-$**	$3.31E-05 \pm 2.00E-05\approx$	$4.06E-05 \pm 3.05E-05$
$f5$	$1.39E+01 \pm 8.74E-01-$	$1.87E+01 \pm 5.47E-01-$	$2.29E+01 \pm 1.28E+00-$	**$5.50E-05 \pm 1.45E-04-$**	$2.95E+01 \pm 2.21E+01$
$f6$	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00$**
$f7$	$8.82E-03 \pm 2.61E-03+$	$5.89E-03 \pm 1.45E-03+$	$1.78E-03 \pm 6.21E-04+$	$2.45E-03 \pm 2.36E-03+$	**$1.17E-03 \pm 6.52E-04$**
$f8$	$7.31E+03 \pm 3.75E+02+$	**$1.34E-02 \pm 1.82E-12\approx$**	$7.51E+03 \pm 2.36E+02+$	**$1.34E-02 \pm 1.82E-12\approx$**	**$1.34E-02 \pm 1.19E-12$**
$f9$	$1.77E+02 \pm 1.10E+01+$	**$0.00E+00 \pm 0.00E+00\approx$**	$7.83E+01 \pm 2.21E+01+$	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00$**
$f10$	$5.93E-09 \pm 3.10E-09+$	$5.42E-15 \pm 1.74E-15+$	$8.97E-15 \pm 1.74E-15+$	$6.48E-15 \pm 1.63E-15+$	**$4.00E-15 \pm 0.00E+00$**
$f11$	$6.33E-16 \pm 1.16E-15+$	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00\approx$**	**$0.00E+00 \pm 0.00E+00$**
$f12$	$2.20E-17 \pm 1.81E-17+$	$1.97E-32 \pm 8.15E-33+$	$2.23E-29 \pm 2.32E-29+$	$1.64E-32 \pm 1.55E-33+$	**$1.57E-32 \pm 2.74E-48$**
$f13$	$8.26E-17 \pm 3.59E-17+$	$2.09E-31 \pm 2.93E-31+$	$2.57E-29 \pm 3.07E-29+$	$3.25E-32 \pm 2.37E-32+$	**$1.36E-32 \pm 3.70E-34$**
−	1	1	2	2	
+	11	7	9	6	
≈	1	5	2	5	

Table 4: Average rankings of the five algorithms for the 13 test functions achieved by Friedman test.

Algorithm	Ranking
DEECL	**2.04**
DECLS	2.27
jDE	2.65
ODE	3.50
DE	4.54

from the elite individuals to promote the exploitation power and employs a simple and effective parameter adaptation mechanism to enhance the robustness. In the experiments, we use 13 classical test functions that are widely used in the evolutionary computation community to evaluate the performance of DEECL. The experimental results show that DEECL can outperform the conventional DE, jDE, ODE, and DECLS on the majority of the test functions.

In the future, we will apply DEECL to handle more complex optimization problems, such as high-dimensional optimization problems and multiobjective optimization problems.

Conflict of Interests

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

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (nos. 61364025, 61462036, and 41261093), by the Natural Science Foundation of Jiangxi, China (nos. 20151BAB217010 and 20151BAB201015), by the Education Department Youth Scientific Research Foundation of Jiangxi Province, China (nos. GJJ14456 and GJJ13378).

References

[1] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[2] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[3] J. Vesterstrøm and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 2, pp. 1980–1987, IEEE, June 2004.

[4] L. Wang and L.-P. Li, "Fixed-structure H_∞ controller synthesis based on differential evolution with level comparison," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 120–129, 2011.

[5] F. Martín, L. Moreno, M. L. Muñoz, and D. Blanco, "Initial population size estimation for a differential-evolution-based global localization filter," *International Journal of Robotics and Automation*, vol. 29, no. 3, 2014.

[6] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.

[7] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[8] W. Gong, Z. Cai, C. X. Ling, and C. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 2, pp. 397–413, 2011.

[9] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 634–647, 2013.

[10] A. Deb, J. S. Roy, and B. Gupta, "Performance comparison of differential evolution, particle swarm optimization and genetic algorithm in the design of circularly polarized microstrip antennas," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 8, pp. 3920–3928, 2014.

[11] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[12] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[13] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[14] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[15] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Computing*, vol. 15, no. 11, pp. 2127–2140, 2011.

[16] D. Jia, G. Zheng, and M. Khurram Khan, "An effective memetic differential evolution algorithm based on chaotic local search," *Information Sciences*, vol. 181, no. 15, pp. 3175–3187, 2011.

[17] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[18] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[19] 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.

[20] Q.-K. Pan, P. N. Suganthan, L. Wang, L. Gao, and R. Mallipeddi, "A differential evolution algorithm with self-adapting strategy and control parameters," *Computers and Operations Research*, vol. 38, no. 1, pp. 394–408, 2011.

[21] D. Kranjcic and G. Stumberger, "Differential evolution-based identification of the nonlinear kaplan turbine model," *IEEE Transactions on Energy Conversion*, vol. 29, no. 1, pp. 178–187, 2014.

[22] Q.-K. Pan, L. Wang, L. Gao, and W. D. Li, "An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers," *Information Sciences*, vol. 181, no. 3, pp. 668–685, 2011.

[23] L. X. Tang, Y. Zhao, and J. Y. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 209–225, 2014.

[24] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[25] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 99–119, 2011.

[26] S. Kundu, S. Das, A. V. Vasilakos, and S. Biswas, "A modified differential evolution-based combined routing and sleep scheduling scheme for lifetime maximization of wireless sensor networks," *Soft Computing*, vol. 19, no. 3, pp. 637–659, 2014.

[27] T. Bhadra and S. Bandyopadhyay, "Unsupervised feature selection using an improved version of differential evolution," *Expert Systems with Applications*, vol. 42, no. 8, pp. 4042–4053, 2015.

[28] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Information Sciences*, vol. 185, no. 1, pp. 153–177, 2012.

[29] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[30] B. Li and W. S. Jiang, "Optimizing complex functions by chaos search," *Cybernetics & Systems*, vol. 29, no. 4, pp. 409–419, 1998.

[31] Y. He, Q. Xu, S. Yang, and L. Liao, "Reservoir flood control operation based on chaotic particle swarm optimization algorithm," *Applied Mathematical Modelling*, vol. 38, no. 17, pp. 4480–4492, 2014.

[32] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[33] T. Xiang, X. Liao, and K.-W. Wong, "An improved particle swarm optimization algorithm combined with piecewise linear chaotic map," *Applied Mathematics and Computation*, vol. 190, no. 2, pp. 1637–1645, 2007.

[34] X. F. Yan, D. Z. Chen, and S. X. Hu, "Chaos-genetic algorithms for optimizing the operating conditions based on RBF-PLS model," *Computers & Chemical Engineering*, vol. 27, no. 10, pp. 1393–1404, 2003.

[35] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[36] Y.-W. Shang and Y.-H. Qiu, "A note on the extended Rosenbrock function," *Evolutionary Computation*, vol. 14, no. 1, pp. 119–126, 2006.

[37] S. Garcia and F. Herrera, "An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[38] 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.

[39] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J.-S. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," *Information Sciences*, vol. 223, pp. 119–135, 2013.