2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Improved Differential Evolution with Adaptive Opposition Strategy

Huichao Liu *†‡, Zhijian Wu *†, Hui Wang †, Shahryar Rahnamayan §, Changshou Deng ¶
*Computer School of Wuhan University,Wuhan 430072, China
†State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China
‡College of Information Engineering, Huanghuai University, Zhumadian 463000, China
§Department of Electrical, Computer and Software Engineering,
University of Ontario Institute of Technology, Oshawa, ONL1H7K4, Canada
¶School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China
E-mail: zhijianwu,liuhuichao,huiwang@whu.edu.cn,shahryar.rahnamayan@uoit.ca

***Abstract*—Generalized opposition-based differential evolution (GODE) is an effective algorithm for global optimization over continuous search space. However, the performance of GODE highly depends on its control parameters. To improve the performance of GODE, this paper proposes an enhanced GODE algorithm called AGODE, which employs an adaptive generalized opposition-based learning (GOBL) mechanism to automatically adjust the probability of opposition during the evolution. Experimental study is conducted on a set of 19 well-known benchmark functions. Computational results show that the proposed approach AGODE outperforms some state-of-the-art DE variants on the majority of test problems.**

I. Introduction

Differential evolution (DE) was proposed by Price and Storn [1] [2]. It has attracted many researcher's attention for its simplicity, effectiveness and robustness in solving a broad range of benchmark and real-world optimization problems. In order to further improve the performance of DE and expand its application scope, some variants of DE have been proposed in the past decades. The opposition-based differential evolution (ODE) and generalized opposition-based differential evolution (GODE) algorithm are the representatives of them. The concept of opposite-based learning (OBL) was introduced by Tizhoosh [3]. Rahnamayan et al. [4] firstly utilized OBL to accelerate the convergence rate of DE, and presented the ODE algorithm. Wang et al. [5] generalized the OBL, called GOBL, and proposed the GODE algorithm, which has achieved better performance for solving high-dimensional optimization problems [6].

However, some previous works also found that the performance of DE greatly depends on the choice of mutant strategies and its control parameters. Different mutation strategies or parameter settings may result significant differences of performance. The classical DE algorithm has five mutation schemes, two crossover strategies and three basic control parameters (F, Cr and NP). It is very difficult to manually select appropriate strategies and control parameters for DE to achieve the optimal performance in solving different problems. In order to automatically choose the suitable mutant strategy and control parameters, the adaptive and self-adaptive mechanisms are the most commonly used approaches.

The above mentioned problem also exists in GODE. The performance of GODE is sensitive to its control parameter, probability of opposition (p_o), which is usually a fixed value during the search process. In this paper, we propose an adaptive GODE algorithm, called AGODE, which employs adaptive opposition mechanism to automatically adjust the probability of opposition p_o during the search process. The proposed AGODE algorithm is tested on a suite of 19 global optimization problems with D=60. Experimental results confirm the effectiveness of the adaptive mechanism, and show faster convergence speed and better performance of AGODE than other five DE algorithms on the majority of test problems.

The rest paper is organized as follows. The DE algorithm is briefly reviewed in Section II. In Section III, the ODE algorithm, GODE algorithm, the proposed adaptive opposition mechanism and AGODE algorithm are explained in detail. Then, a comprehensive set of experiment results is given in Section IV. Finally, the work is concluded in Section V.

II. A Brief Review Of Differential Evolution

A. Differential Evolution (DE)

As a population-based search method, DE starts with an initial vector population, which is often randomly generated within the search space, and the values of each dimension for all individuals conform to the uniform distribution. Let's assume that X_i^G (i = 1, 2, ..., NP) is the ith individual in population $P(G)$ (NP is the population size, G is the generation index). For DE algorithm, its mutation, crossover, and selection operators are defined as follows.

Mutation–For each vector X_i^G in generation G, a mutant (or donor) vector V_i^G is defined by:

$$V_i^G = X_{r_1}^G + F \cdot (X_{r_2}^G - X_{r_3}^G) \tag{1}$$

where $i \in [1, NP]$, r_1, r_2 and r_3 are three random integer indices selected from {1, 2, ..., NP}. Furthermore, i, r_1, r_2 and r_3 are different from each other so that $NP \geq 4$. $F \in (0, 2]$ is a real constant which determines the amplification of the added differential variation of $(X_{r_1}^G - X_{r_2}^G)$.

Crossover–DE utilizes the crossover operation to generate new trial vector $U_i^G = (U_{1i}^G, U_{2i}^G, \ldots, U_{Di}^G)$ (D indicates problem dimension) by exchanging the components of donor vector V_i^G and target vector X_i^G. In exponential crossover, we should choose two integers n and L randomly among

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

[1, D]. The integer n acts as a starting point in the target vector, from where the exchange of components with the donor vector starts. And L, which is determined by crossover rate (Cr), denotes the number of components that the donor vector actually contributes to the target vector. Then the trial vector is obtained as:

$$U_{ji}^{G}=\begin{cases} V_{ji}^{G} & \text{if } j=\langle n\rangle_D, \langle n+1\rangle_D, \ldots, \langle n+L-1\rangle_D \\ X_{ji}^{G} & \text{otherwise} \end{cases} \quad (2)$$

where the angular brackets $\langle\rangle_D$ denote a modulo function with modulus D.

Selection–It is an approach which decides which vector (U_i^G or X_i^G) should be a member of next generation G+1. The survival mechanism is defined by:

$$X_i^{G+1} = \begin{cases} U_i^G & \text{if } f(U_i^G) \le f(X_i^G) \\ X_i^G & \text{otherwise} \end{cases} \quad (3)$$

Without loss of generality, this paper only considers minimization problems. If the function value of trial vector is smaller than target vector's, then the target vector will be replaced by the trial vector.

B. A Brief Review of the Related Works

1) Opposition-based Learning Mechanism: The opposite-based learning (OBL) mechanism has been widely studied and extensively used in the past decade. OBL mechanism was introduced by Tizhoosh [3], and then used to enhance the performance of reinforcement learning [7] in machine intelligence. Rahnamayan et al. [4] firstly utilized OBL to accelerate the convergence rate of DE. In [8], Rahnamayan et al. take a comprehensive experiments on 58 complex benchmark functions to investigate the performance of OBL. Experimental results confirm that ODE outperforms the original DE and FADE in terms of convergence speed and solution accuracy. Meanwhile, the mathematical proofs [9] shown that, opposite numbers are more likely to be closer to the optimal solution than purely randomly generated ones. In [10], Rahnamayan et al. proposed an Euclidean distance-to-optimal solution proof that shows intuitively why considering the opposite of a candidate solution is more beneficial than another random solution. Wang et al. [5] generalized the OBL, called GOBL, which transforms candidate solution in current search space to a new search space. The proposed algorithm, named GODE, has achieved better performance for solving high-dimensional optimization problems [6]. Moreover, The GODE algorithm has also been implemented on GPU platform to obtain the more computational power in parallel [11].

In addition, the OBL mechanism has been used in other optimization algorithms. Malisia et al. [12] proposed the opposition-based ant colony optimization (OACO) algorithm. Ergezer et al. [13] presented the oppositional biogeography-based optimization algorithm. El-Abd [14] introduced the opposition-based artificial bee colony algorithm. Zhou et al. [15] presented the elite opposition-based particle swarm optimization (EOPSO). At the same time, the opposition-based algorithms are also used to solve some real-world problems. Dhahri et al. [16] uses the opposition-based differential evolution for beta basis function neural network. Shaw et al. [17] applies the opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems.

2) Adaptive and Self-adaptive Mechanism: Adaptive mechanism is an important method for selecting control parameters and mutant strategy automatically in DE. Some excellent works have been proposed in the past decade. Liu and Lampinen [18] proposed an FADE algorithm which uses fuzzy logic controllers to adapt the parameters in their mutation and crossover operations. Zhang and Sanderson [19] [20] proposed a JADE algorithm, which applies normal distribution $N(\mu,0.1)$ to determine the parameters F and Cr, and μ is the weighted average of the recorded F and Cr values which have produced better individuals during the last generation. Qin et al. [21] [22] introduced a SaDE algorithm which selects the mutant strategy from the strategy pool according to its success rate for producing promising solutions in the past evolution phase.

Random selection is a simple and effective self-adaptive mechanism to determine the control parameters and mutant strategy of DE. In jDE [23] [24], the control parameters F and Cr are encoded into each individual and adjusted by two new parameters τ_1 and τ_2. In SaDE [21] [22], the parameter F is selected randomly which obeys the normal distribution with mean value 0.5 and standard deviation 0.3. Yang et al. [25] introduced a neighborhood search (NS) mechanism to replace the fix parameter F, and the parameter Cr is also generated by uniform distribution within (0, 1). In [26], Mallipeddi and Suganthan introduced an ensemble of mutation strategies and control parameters with DE (EPSDE). In CoDE [27], parameters F and Cr for each mutant strategy are randomly selected from the parameter candidates pool that obey the uniform distribution. Wang et al. [28] proposed a gaussian bare-bones differential evolution, namely GBDE, in which the mutant operator is simplified into a Gaussian distribution $N(\mu,\sigma)$, where $\mu = (X_{best,G} + X_{i,G}/2)$ and $\sigma = \|X_{best,G} - X_{i,G}\|$.

The population size NP also play an important role among the control parameters [29]. Teo in [30] made a first attempt at self-adapting the parameter of population size. Brest et al. [31] presented a population size reduction mechanism, which gradually reduce the population size during the evolution process. Wang et al. [32] proposed a variable population size mechanism which adjusts the population size dynamically by counting the improvement of best fitness value in a specified learning period. Sarker et al. [33] proposed a new mechanism to dynamically select the best performing combinations of parameters F, Cr and NP for a problem during the course of a single run. The proposed algorithm, called DE-DPS, not only saves the computational time, but also shows better performance over some state-of-the-art algorithms.

In this section, we have only presented a brief overview of some recently proposed DE variants which using opposition-based learning mechanism and adaptive (and/or) self-adaptive mechanism, respectively, to select its mutant/crossover strategies and control parameters; some comprehensive surveys can be found in [34], [35] and [36].

III. The Adaptive GODE Algorithm (AGODE)

A. Opposition-based Differential Evolution (ODE)

The ODE algorithm [4][8], proposed by Rahnamayan et al., first utilizes the OBL to enhance the performance of an

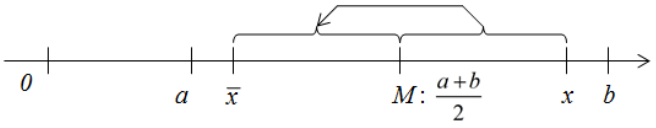

Fig. 1. Opposite number, $\overline{x}$, for OBL can be seen as a mirror to x

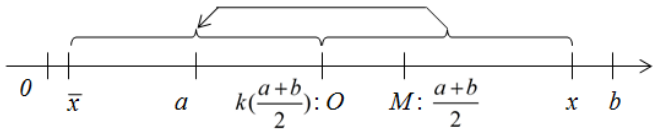

Fig. 2. Opposite number, $\overline{x}$, for GOBL can be seen as a mirror to x

optimization process, DE as a case study. The main idea behind OBL is the simultaneous consideration of an estimate and its opposite estimation in order to achieve a better approximation for the current candidate solution. In fact, mathematical proofs [9] have shown that, opposite numbers are more likely to be closer to the optimal solution than purely randomly generated ones. In order to understand OBL well, we need to explain some concepts first.

Opposite number: Let $x \in [a, b]$ be a real number. The opposite number $\overline{x}$ is defined as follow:

$$\overline{x} = a + b - x \tag{4}$$

From Figure 1, the opposite number $\overline{x}$ can be seen as a mirror to x centered by the middle point of the defined boundary $[a, b]$. Similarly, the definition can be generalized to higher dimensions as follows.

Opposite point: Let $X = (x_1, x_2, \ldots, x_D)$ be a point in a D-dimensional search space, where $(x_1, x_2, \ldots, x_D) \in R^D$ and $x_j \in [a_j, b_j]$, $j \in [1, 2, \ldots, D]$. The opposite point $\overline{X} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_D)$ is defined by:

$$\overline{x}_j = a_j + b_j - x_j \tag{5}$$

Obviously, an opposite point shouldn't jump out of the original space R^D during the search process. When evaluating a solution X to a given problem, its opposite solution, $\overline{X}$, will be simultaneously computed to provide another chance to find a candidate solution closer to the global optimum. In ODE, OBL operation is conducted by the specified probability, known as *jumping rate* (Jr), which is a constant number in (0,1) and often is determined by empirical experiences.

B. Generalized opposition-based DE(GODE)

The GODE algorithm [6] [37] employs the GOBL mechanism to enhance the performance of DE for solving some complex problems. GOBL is the generalized version of OBL. Compared to the OBL, GOBL redefines the concept of opposite number by introducing the new space transfer factor k.

Let $X = (x_1, x_2, \ldots, x_D)$ be a solution in the current generation. Where D indicates the dimension number, and x_j represents each dimension in an individual, and $x_j \in [a_j, b_j]$, $j \in [1, 2, \ldots, D]$. a_j and b_j are the boundaries of the jth dimensional element in current population. Being similar to the OBL, each dimension of the opposite solution $\overline{X}$ can be defined as:

$$\overline{x}_j = k(a_j + b_j) - x_j \tag{6}$$

where k is a random number within [0,1), and used for an entire generation. Obviously, each dimension $\overline{x}_j$ of the opposite solution belongs to $[k(a_j + b_j) - b_j, k(a_j + b_j) - a_j]$.

As shown in Figure 2, we can find that the difference between the current search space S and the opposite search space $\overline{S}$ is the center position of search space. The center of current search space is a fixed point $\frac{a+b}{2}$, while the center of opposite space is a random value between 0 and $\frac{a+b}{2}$. The opposite search space is not always the same as the search space S, Therefore, an opposite solution may jump out of the predefined domains in some situations. If the opposite solution jumps beyond the bounds of the variable, it should be set to a random number within (a, b). The opposite solution $\overline{x}$ can also be seen as a mirror to x based on the new center point $k(\frac{a+b}{2})$. Similar to ODE, the GOBL is conducted by the *jumping rate* parameter, but its name in GODE is called *probability of opposition* (p_o), which is also a predefined number before evolution.

C. Adaptive opposition mechanism

The effectiveness of OBL and GOBL mechanisms has been validated in [8] [5]. But unlike the mutation strategy, opposite operation is not used in every generation during the search process. In fact, some individuals in current population may have not changed over the past few generations. So, too frequent opposite operation cannot achieve any benefit but wastes the evolutionary opportunities. Therefore, in ODE or GODE, the opposite operation is conducted by the probability of opposition (p_o), which is often a fixed small number (ODE use 0.3, and GODE use 0.05).

However, this probability-driven mechanism could result in a problem, that is, opposite operation appears randomly in the whole evolution process, which does not follow the needs of different evolution stages. When DE can find better solution quickly, the appeared opposite operation may be useless. Otherwise, when DE cannot find better solution any more, the opposite operation may be needed, but it will be absent due to the probability-driven mechanism. Therefore, it is the best situation that opposite operation could appear when needed, and be absent when undesired.

This section introduces an adaptive opposition (AO) mechanism which can adjust the parameter p_o dynamically. The main idea behind this mechanism is that if the appeared opposite operation can produce more promising solutions, this may imply that the current population needs more opposite operation, which should be increased in the following evolution process by enlarging the value of p_o. On the contrary, if the current opposite operation cannot provide enough better solution, the value of p_o should be decreased in order to reduce the opposite operation.

Therefore, we use N_{better} to count the better offspring produced by opposite operation in current generation when GOBL is carried out. Then the *success rate* of opposite operation can be indicated by a variable sr, which is defined

by:

$$sr = \frac{N_{better}}{NP} \tag{7}$$

where NP is the population size of current generation. A memory buffer, called srm, also be defined to record the value of sr. After each opposition generation, the sr value will be calculated and recorded into the buffer. After LP (LP is a predefined learning period) such generations, the average value sr_{ave} of the recorded sr value in memory srm can be calculated as:

$$sr_{ave} = \frac{\sum_{i=1}^{LP} sr_i}{LP} \tag{8}$$

where sr_i is the ith sr value in srm. If sr is not equal to sr_{avr}, the value of p_o should be adjusted by:

$$p_o = p_o \times (1 + sr - sr_{ave}) \tag{9}$$

According to the Eq. (9), if sr is larger than sr_{ave}, then the value of $(1 + sr - sr_{ave})$ will be greater than 1, and the value of p_o will be enlarged, thus, the opposite operation will be increased in the following generations. Otherwise, when sr is less than sr_{ave}, a smaller value of p_o will be given.

In addition to parameter p_o, the proposed AO mechanism also introduces another parameter LP. However, It can be seen that the influence of these two parameters on the performance of the algorithm has been weakened. That is because the parameter p_o only needs an initial value, and its appropriate value can be found during the search process. Moreover, as a denominator, the value of LP belongs to [5, 10] has no significant affect on the algorithm performance due to the recorded values of p_o themselves are some very small number. Therefore, the two parameters can be specified by some empirical values. As the recommendation in [8], the *jumping rate* should belong to (0, 0.4) for ODE. But the smaller value of p_o = 0.05 is used in literature [37] for GODE. Therefore, we use the intersection [0.05, 0.4] as the bounds of the parameter. Moreover, we choose the median value 0.2 as the initial value of p_o, and LP is set to 7 for the following experiments.

D. The AGODE algorithm

The AGODE algorithm chooses GODE as its parent algorithm, as well as uses GOBL mechanism to initialize population and produce new candidates in evolutionary generations. The AGODE and GODE have the similar algorithmic framework, and the main difference between the two is that AGODE has AO mechanism to automatically change the control parameters p_o, while which is predefined in the GODE algorithm. The pseudocode of AGODE is given in Algorithm 1, where P is the current population, GOP is the transformed population after using GOBL, FEs is the current fitness evaluations number, and MAX_FEs is the maximum fitness evaluations number.

E. Computational time complexity

Assume that $O(F(D))$ is the computational time complexity of a fitness evaluation function $F(D)$ on dimension D. The computational time complexity of GODE algorithm [37] is $O(D^2) + O(D) \cdot O(F(D))$. Compared to GODE algorithm, the additional steps of AGODE algorithm shown in Algorithm 1 are step 4, 10 and 12. It can be seen easily that the computational time complexities of these steps are $O(1)$, $O(LP)$ and $O(1)$, respectively. Therefore, the new AO mechanism does not increase the computational time complexity of GODE asymptotically. It means that the computational time complexity of AGODE algorithm is also $O(D^2)+O(D)\cdot O(F(D))$, which is $O(D^2)$ or $O(D^3)$ when $O(F(D))$ is equal to $O(D)$ or $O(D^2)$, respectively.

Algorithm 1 The AGODE Algorithm

1: Randomly initialize each individual in population $P(0)$;
2: Conduct the GOBL operation and produce the population $GOP(0)$;
3: Select Np fittest individuals from $\{P, GOP\}$ as the initial population P ;
4: Record the sr into srm for the first generation;
5: **while** $FEs \leq MAX\_FEs$ **do**
6: **if** $rand(0,1) \leq p_o$ **then**
7: Conduct GOBL operation and produce the population GOP;
8: Select Np fittest individuals from $\{P, GOP\}$ as the new population P;
9: **if** GOBL is conducted more than LP generations **then**
10: Calculate sr, sr_{ave} and adjust p_o by Eq. (7) - (9),
11: **end if**
12: Record sr into srm;
13: **else**
14: Execute the classical DE, using $rand/1/exp$ mutant strategy;
15: **end if**
16: **end while**

IV. Experimental Verifications

A. Benchmark function

In the following experiments, we have chosen 19 global optimization functions which were proposed by the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Meta-heuristics for Large Scale Continuous Optimization Problems [38]. Functions $F_1 - F_6$ are provided by [39] and widely tested in CEC–2008. Functions $F_7 - F_{11}$ are proposed in [40] and used for the ISDA-2009. Functions $F_{12} - F_{19}$ are built by combining two functions belongs to the set of functions $F_1 - F_{11}$. The detailed descriptions of $F_1 - F_{19}$ can be found in [38].

B. Algorithms and settings

Experiments have been conducted to compare six algorithms including ODE, GODE, jDE, SaDE, JADE and the proposed AGODE algorithm on the mentioned test suite. Similar to AGODE, ODE and GODE also use the OBL and GOBL, respectively, as the main optimizing operator to DE. Therefore, taking a comprehensive comparison between the three algorithms is very meaningful. The other three algorithms: jDE, SaDE and JADE are the representatives of adaptive DE variants. They use different adaptive mechanisms to produce the parameters F and Cr. Moreover, SaDE also adaptively select different mutant strategy in the evolution process. By

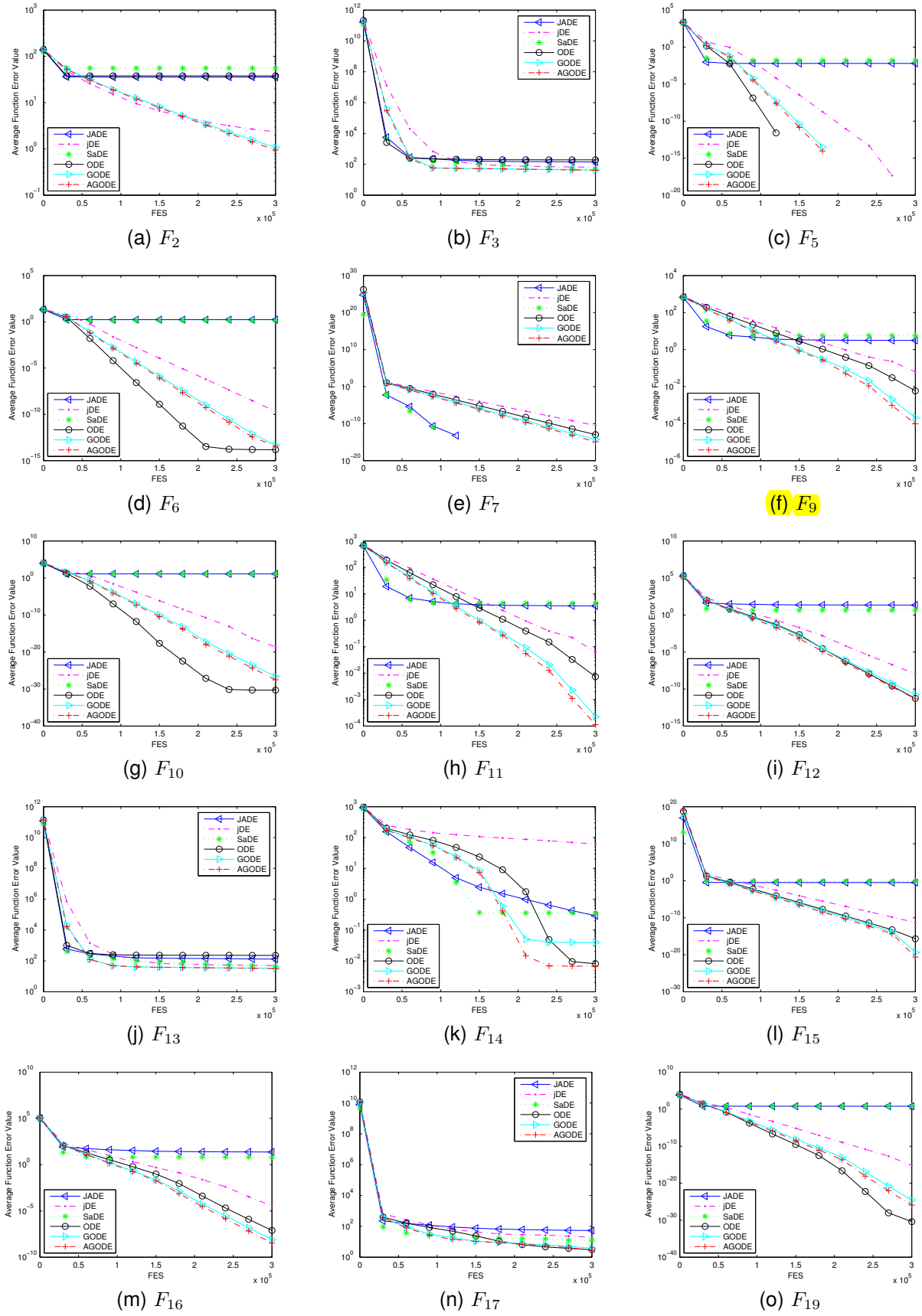

Fig. 3. The average convergence curves of ODE, GODE, jDE, SaDE, JADE and AGODE on F_2, F_3, F_5–F_7, F_9–F_{17} and F_{19} with $D = 60$.

TABLE I. COMPUTATIONAL RESULTS ARCHIVED BY ODE, GODE, JDE, SADE, JADE AND AGODE ON D=60

	ODE	GODE	jDE	SaDE	JADE	AGODE
Functions	Mean Error±Std Dev[1]	Mean Error±Std Dev	Mean Error±Std Dev	Mean Error±Std Dev	Mean Error±Std Dev	Mean Error±Std Dev
F_1	4.04E-29±1.98E-29	9.24E-27±9.45E-28	8.36E-19±1.15E-19	2.42E-29±1.15E-29	**4.04E-30±2.74E-30**	7.75E-28±2.08E-28
F_2	3.78E+01±1.74E+00	1.07E+00±3.43E-02	2.33E+00±2.19E-01	5.57E+01±9.10E-01	3.56E+01±7.45E-01	**9.43E-01±1.75E-02**
F_3	1.94E+02±3.80E+01	4.19E+01±1.86E+00	6.52E+01±5.84E+00	4.93E+01±6.22E+00	1.41E+02±1.99E+01	**4.15E+01±1.51E+00**
F_4	1.06E-01±7.43E-02	**1.35E-11±2.19E-12**	8.57E+01±1.60E+00	7.56E-01±3.66E-01	3.43E-09±6.76E-10	6.63E-02±4.53E-02
F_5	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	1.61E-02±4.53E-03	6.20E-03±1.99E-03	**0.00E+00±0.00E+00**
F_6	**1.47E-14±4.49E-16**	4.68E-14±1.26E-15	2.03E-10±1.28E-11	1.51E+00±7.00E-02	1.70E+00±1.55E-01	3.57E-14±6.47E-16
F_7	1.00E-13±8.76E-15	5.58E-15±3.77E-16	2.85E-11±2.43E-12	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	1.16E-15±1.40E-16
F_8	7.98E+01±4.14E+00	2.60E+01±1.22E+00	2.62E+03±1.19E+02	**9.99E-02±9.52E-02**	5.23E+02±3.46E+01	1.71E+01±8.86E-01
F_9	5.85E-03±4.09E-04	2.07E-04±9.69E-06	6.23E-02±3.22E-03	5.67E+00±1.52E+00	3.11E+00±5.38E-01	**9.64E-05±5.96E-06**
F_{10}	**5.27E-31±2.32E-31**	2.36E-27±2.19E-28	2.21E-19±2.89E-20	1.27E+01±5.22E-01	1.30E+01±9.98E-01	3.18E-28±5.14E-29
F_{11}	7.35E-03±5.44E-04	2.26E-04±1.10E-05	6.24E-02±3.21E-03	4.37E+00±8.36E-01	3.50E+00±7.48E-01	**1.13E-04±5.40E-06**
F_{12}	5.31E-12±5.71E-13	1.74E-11±9.34E-13	1.13E-08±9.07E-10	4.47E+00±1.43E+00	2.19E+01±3.98E+00	**4.75E-12±3.98E-13**
F_{13}	2.29E+02±1.77E+02	3.28E+01±2.21E-01	5.06E+01±5.14E+00	4.40E+01±7.71E+00	1.31E+02±1.21E+01	**3.21E+01±1.52E-01**
F_{14}	8.10E-03±7.92E-03	3.98E-02±3.90E-02	6.19E+01±1.09E+00	3.58E-01±1.11E-01	2.92E-01±8.09E-03	**6.73E-03±6.62E-03**
F_{15}	2.05E-16±4.61E-17	4.66E-20±1.08E-20	5.17E-12±5.19E-13	6.94E-01±1.26E-01	3.23E-01±1.03E-01	**2.54E-21±1.87E-21**
F_{16}	7.89E-08±4.43E-09	7.91E-09±3.10E-10	3.52E-05±2.47E-06	6.60E+00±1.60E+00	2.34E+01±4.46E+00	**3.33E-09±2.04E-10**
F_{17}	3.05E+00±5.28E-01	3.83E+00±2.52E-01	1.92E+01±4.60E+00	1.25E+01±2.33E+00	5.37E+01±9.34E+00	**2.76E+00±1.78E-01**
F_{18}	1.16E-01±6.29E-02	3.98E-02±3.90E-02	3.75E-01±7.99E-02	**4.29E-04±4.20E-04**	1.00E+00±3.01E-02	3.32E-02±3.26E-02
F_{19}	**4.02E-31±1.73E-31**	2.64E-25±3.51E-26	6.45E-16±2.23E-16	7.32E+00±4.88E-01	6.04E+00±5.32E-01	1.16E-26±2.28E-27
$w/t/l$[2]	14/1/4	17/1/1	18/1/0	15/0/4	17/0/2	—

[1] "Mean Error" and "Std Dev" indicate the average and standard deviation of the error values obtained in 25 runs.
[2] "$w/t/l$" means that the AGODE algorithm wins in w functions, ties in t functions, and loses in l functions.

contrast, the AGODE utilizes AO mechanism to optimize the parameter p_o during the evolution. It is interesting to compare the performance of the four algorithms. For convenience, we put all comparisons together and analyze the experiment results simultaneously, which does not affect the final conclusion.

In AGODE, the parameter LP and initial value of p_o are set to 7 and 0.2 (see explanation in Section III-C). The rest parameters F and Cr are fixed to 0.5 and 0.9, respectively, and the $rand/1/exp$ strategy is also employed, which are same as the settings in the parent algorithm GODE [37]. For a fair comparison, the control parameters and mutant strategy of rest five algorithms are according to the settings in the original papers of ODE [8], GODE [37], jDE [24], SaDE [22] and JADE [19], respectively.

In this paper, we focus on investigating the optimization performance of each algorithm on the test functions with D=60. The maximum fitness evaluation number, MAX_FEs, is set to $5000 \cdot D$. Each run stops when the MAX_FEs is meet. In the following experiments, each algorithm is run 25 times for each test function. The average errors and standard deviation of the best individual found in 25 runs has been recorded for measuring the performance of each algorithm.

C. Results analysis and discussion

The comparison results among ODE, GODE, jDE, SaDE, JADE and AGODE on problems with D = 60 are presented in Table I. The best results among the six algorithms are shown in **boldface**. It can be seen that AGODE algorithm obtains the best solutions on 11 functions F_2, F_3, F_5, F_9 and F_{11}–F_{17}, while ODE, SaDE, GODE, JADE and jDE only get 4, 3, 2, 2 and 1 best values, respectively. All algorithms, including AGODE, converge to the global optimum on only one function. The bottom of Table I summarizes the comparison results between AGODE and other algorithms. As seen, AGODE outperforms ODE, GODE, jDE, SaDE and JADE on 14, 17, 18, 15 and 17 functions, respectively. AGODE only loses to other algorithms on 4, 1, 0, 4, and 2 functions out of the 19 test functions.

Figure 3 shows the average convergence graphs for F_2, F_3, F_5–F_7, F_9–F_{17} and F_{19} in detail. We can see that AGODE gets the fastest convergence speed and best values on F_2, F_9, F_{11}, F_{14}, F_{15} and F_{16}, and achieves faster convergence speed on other functions. The ODE algorithm obtains the fastest convergence speed on 4 functions of F_5, F_6, F_{10} and F_{19}, but its performance is always worse than AGODE on the other functions. Although GODE algorithm does not achieve any fastest convergence speed, but its performance is only slightly worse than AGODE. In addition, the convergence performance of jDE algorithm is often in the middle of the six algorithms. But, JADE and SaDE often get the slowest convergence speed except on F_7 and F_{14}, and show the worst performance. In a whole, the AGODE algorithm achieves the best comprehensive performance among the six algorithms. Moreover, it also gets better stability, as well as its parent algorithm GODE.

By the suggestion of [41], we use non-parametric tests to compare the performance of the algorithms SaDE, jDE, JADE, ODE, GODE and AGODE on the test suite. The results of Friedman average ranking test are shown in Table II. These results are calculated by the SPSS statistical software. We can see that the AGODE algorithm gets the lowest score in the comparisons for all algorithms. It means that AGODE algorithm has the best performance among the six algorithms. Table III shows the p-values of applying Wilcoxon's test between AGODE and the other five algorithms. The p-values below 0.05 (the significant level) are shown in **boldface**. As seen, AGODE is significantly better than ODE, GODE, SaDE, jDE and JADE on the test bed when D=60.

TABLE II. AVERAGE RANKINGS ACHIEVED BY AGODE AND OTHER ALGORITHMS WHEN $D = 60$

Algorithms	Ranking
AGODE	**1.71**
GODE	2.66
ODE	3.18
SaDE	4.29
jDE	4.50
JADE	4.66

TABLE III. WILCOXON'S TEST BETWEEN AGODE AND OTHER ALGORITHMS WHEN $D = 60$

Algorithms	p-values of AGODE vs. other algorithms
GODE	**1.59E-03**
ODE	**1.18E-03**
SaDE	**4.85E-03**
jDE	**1.96E-04**
JADE	**3.98E-04**

V. CONCLUSION

In this paper, we propose a new adaptive DE algorithm, called AGODE, which employs an adaptive GOBL mechanism to dynamically adjust the probability of opposition (p_o) according to the success rate of the current opposition operations. When the opposition achieves better candidate solutions, p_o will be increased to provide more opposition; otherwise, p_o will be decreased in order to reduce the opposition. Experimental verifications on 19 test functions with D=60 show that the proposed AGODE outperforms ODE, GODE, jDE, JADE and SaDE on the majority of the test functions.

However, the AO mechanisms is not always effective. For example, AGODE achieve worse results than GODE on F_4. Moreover, the effectiveness of AO mechanism for other problems will be investigated in the future work.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China (No.61070008, No. 61364025 and No. 61305150), the Humanity and Social Science Foundation of Ministry of Education of China (No. 13YJCZH174), the Science and Technology Plan Projects of Jiangxi Provincial Education Department (No.GJJ13744), the Foundation of State Key Laboratory of Software Engineering (No.SKLSE2012-09-19), the Henan Province Science and Technology R&D Program (No. 122102310474), and the Fundamental Research Funds for the Central Universities (No. 2012211020205).

REFERENCES

[1] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute, Berkeley, CA, Tech. Rep. TR-95-012, March 1995.

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

[3] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Inter. Conf. Comput. Intell. for Model., Control and Auto., and Inter. Conf. Intell. Agents, Web Tech. and Inter. Commerce, 2005*, vol. 1. IEEE, 2005, pp. 695–701.

[4] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution algorithms," in *Evolutionary Computation (CEC), 2006 IEEE Congress on*. IEEE, July 2006, pp. 2010–2017.

[5] H. Wang, Z. Wu, Y. Liu, J. Wang, D. Jiang, and L. Chen, "Space transformation search: a new evolutionary technique," in *Proc. of the first ACM/SIGEVO Summit on Genetic and Evol. Comput., GEC'09*. ACM, 2009, pp. 537–544.

[6] 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, Nov. 2011.

[7] H. R. Tizhoosh, "Opposition-based reinforcement learning," *J. of Advanced Comput. Intelligences and Intelligent Inform.*, vol. 10, no. 4, pp. 578–585, 2006.

[8] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[9] S. Rahnamayan, H. R. Tizhoosh, and M. Salama, "Opposition versus randomness in soft computing techniques," *Applied Soft Computing*, vol. 8, no. 2, pp. 906–918, March 2008.

[10] S. Rahnamayan, G. G. Wang, and M. Ventresca, "An intuitive distance-based explanation of opposition-based sampling," *Applied Soft Computing*, vol. 12, no. 9, pp. 2828–2839, Sept. 2012.

[11] H. Wang, S. Rahnamayan, and Z. Wu, "Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems," *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 62–73, Jan. 2013.

[12] A. R. Malisia and H. R. Tizhoosh, "Applying opposition-based ideas to the ant colony system," in *Swarm Intell. Symp., SIS'07*. IEEE, April 2007, pp. 182–189.

[13] M. Ergezer, D. Simon, and D. Du, "Oppositional biogeography-based optimization," in *IEEE Inter. Conf. Systems, Man and Cybernetics, SMC'09*. IEEE, Oct. 2009, pp. 1009–1014.

[14] M. El-Abd, "Opposition-based artificial bee colony algorithm," in *Proc. of the 13th Annual Conf. on Genetic and Evol. Comput., GECCO'11*. ACM, July 2011, pp. 109–116.

[15] X. Zhou, Z. Wu, H. Wang, and K. Li, "Elite opposition-based particle swarm optimization," *Acta Electronica Sinica*, vol. 41, no. 8, pp. 1647–1652, March 2013.

[16] H. Dhahri and A. M. Alimi, "Opposition-based differential evolution for beta basis function neural network," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, July 2010, pp. 1–8.

[17] B. Shaw, V. Mukherjee, and S. Ghoshal, "A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems," *Inter. Journal of Electrical Power & Energy Systems*, vol. 35, no. 1, pp. 21–33, Feb. 2012.

[18] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, June 2005.

[19] J. Zhang and A. C. Sanderson, "JADE: Self-adaptive differential evolution with fast and reliable convergence performance," in *Evolutionary Computation (CEC), 2007 IEEE Congress on*. IEEE, Sept. 2007, pp. 2251–2258.

[20] ——, "JADE: adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[21] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Evolutionary Computation (CEC), 2005 IEEE Congress on*, vol. 2. IEEE, Sept. 2005, pp. 1785–1791.

[22] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, April 2009.

[23] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *Evolutionary Computation (CEC), 2006 IEEE Congress on*. IEEE, July 2006, pp. 215–222.

[24] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[25] Z. Yang, X. Yao, and J. He, "Making a difference to differential evolution," in *Advances in Metaheuristics for Hard Optimization*. Springer, 2008, pp. 397–414.

[26] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," in *Swarm, Evolutionary, and Memetic Computing*. Springer, Dec. 2010, vol. LNCS 6466, pp. 71–78.

[27] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.

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

[29] J. Brest, B. Boskovic, A. Zamuda, I. Fister, and M. S. Maucec, "Self-adaptive differential evolution algorithm with a small and varying population size," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, June 2012, pp. 1–8.

[30] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing*, vol. 10, no. 8, pp. 673–686, June 2006.

[31] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, Dec. 2008.

[32] H. Wang, S. Rahnamayan, and Z. Wu, "Adaptive differential evolution with variable population size for solving high-dimensional problems," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, June 2011, pp. 2626–2632.

[33] R. Sarker, S. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Trans. Evol. Comput.*, 2013, Available on-line (Open access), DOI: 10.1109/TEVC.2013.2281528.

[34] F. S. Al-Qunaieer, H. R. Tizhoosh, and S. Rahnamayan, "Opposition based computing–a survey," in *Inter. Joint Conf. Neural Networks, IJCNN'10*. IEEE, July 2010, pp. 1–7.

[35] 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.

[36] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[37] H. Wang, Z. Wu, S. Rahnamayan, and L. Kang, "A scalability test for accelerated de using generalized opposition-based learning," in *Inter. Conf. Intell. Systems Design and Applic., ISDA'09*. IEEE, Nov. 2009, pp. 1090–1095.

[38] F. Herrera, M. Lozano, and D. Molina, "Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems," University of Granada, Spain, Tech. Rep., 2010.

[39] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang, "Benchmark functions for the cec2008 special session and competition on large scale global optimization," *Nature Inspired Computation and Applications Laboratory, USTC, China*, 2007.

[40] F. Herrera and M. Lozano, "Workshop for evolutionary algorithms and other metaheuristics for continuous optimization problemsca scalability test," in *Inter. Conf. on Intelligent System Design and Applications*, Pisa, Italy, 2009.

[41] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, March 2011.