Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2017, Article ID 7974218, 18 pages
https://doi.org/10.1155/2017/7974218

Research Article

Differential Evolution with Novel Mutation and Adaptive Crossover Strategies for Solving Large Scale Global Optimization Problems

Ali Wagdy Mohamed[1,2] and Abdulaziz S. Almazyad[1,3]

[1]*College of Computer and Information Systems, Al-Yamamah University, P.O. Box 45180, Riyadh 11512, Saudi Arabia*
[2]*Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza 12613, Egypt*
[3]*College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia*

Correspondence should be addressed to Ali Wagdy Mohamed; aliwagdy@gmail.com

Received 27 September 2016; Revised 3 December 2016; Accepted 6 February 2017; Published 8 March 2017

Academic Editor: Miin-Shen Yang

This paper presents Differential Evolution algorithm for solving high-dimensional optimization problems over continuous space. The proposed algorithm, namely, ANDE, introduces a new triangular mutation rule based on the convex combination vector of the triplet defined by the three randomly chosen vectors and the difference vectors between the best, better, and the worst individuals among the three randomly selected vectors. The mutation rule is combined with the basic mutation strategy DE/rand/1/bin, where the new triangular mutation rule is applied with the probability of 2/3 since it has both exploration ability and exploitation tendency. Furthermore, we propose a novel self-adaptive scheme for gradual change of the values of the crossover rate that can excellently benefit from the past experience of the individuals in the search space during evolution process which in turn can considerably balance the common trade-off between the population diversity and convergence speed. The proposed algorithm has been evaluated on the 20 standard high-dimensional benchmark numerical optimization problems for the IEEE CEC-2010 Special Session and Competition on Large Scale Global Optimization. The comparison results between ANDE and its versions and the other seven state-of-the-art evolutionary algorithms that were all tested on this test suite indicate that the proposed algorithm and its two versions are highly competitive algorithms for solving large scale global optimization problems.

1. Introduction

In general, global numerical optimization problem can be expressed as follows (without loss of generality minimization problem is considered here):

$$\begin{aligned} \min \quad & f(x), \\ & \vec{x} = [x_1, x_2, \ldots, x_D] \in \mathbb{R}^D; \\ & x_j \in \left[x_j^L, x_j^U\right], \quad \forall j = 1, 2, \ldots, D, \end{aligned} \tag{1}$$

where f is the objective function, $\vec{x}$ is the decision vector $\in \mathbb{R}^D$ space consisting of D variables, D is the problem dimension, that is, the number of variables to be optimized, and x_j^L and x_j^U are the lower and upper bounds for each decision variable, respectively.

The optimization of the large scale problems of this kind (i.e., $D = 1000$) is considered a challenging task since the solution space of a problem often increases exponentially with the problem dimension and the characteristics of a problem may change with the scale [1]. Generally speaking, there are different types of real-world large scale global optimization (LSGO) problems in engineering, manufacturing, and economy applications (biocomputing, data or web mining, scheduling, vehicle routing, etc.). In order to draw more attention to this challenge of optimization, the first competition on LSGO was held in CEC 2008 [2]. Consequently, In the recent few years, LSGO has gained considerable attention and has attracted much interest from Operations Research and

Computer Science professionals, researchers, and practitioners as well as mathematicians and engineers. Therefore, the challenges mentioned above have motivated the researchers to design and improve many kinds of efficient, effective, and robust various kinds of metaheuristics algorithms that can solve (LSGO) problems with high quality solution and high convergence performance with low computational cost. Evolutionary algorithms (EAs) have been proposed to meet the global optimization challenges. The structure of EAs has been inspired from the mechanisms of natural evolution. Due to their adaptability and robustness, EAs are especially capable of solving difficult optimization problems, such as highly nonlinear, nonconvex, nondifferentiable, and multimodal optimization problems. Generally, the process of EAs is based on the exploration and the exploitation of the search space through selection and reproduction operators [3]. Similar to other evolutionary algorithms (EAs), Differential Evolution (DE) is a stochastic population-based search method, proposed by Storn and Price [4]. The advantages are its simplicity of implementation, ease of use, speed, and robustness. Due to these advantages, it has been successfully applied for solving many real-world applications, like admission capacity planning in higher education [5, 6], financial markets dynamic modeling [7], solar energy [8], and many others. In addition, many recent studies prove that the performance of DE is highly competitive with and in many cases superior to other EAs in solving unconstrained optimization problems, constrained optimization problems, multiobjective optimization problems, and other complex optimization problems [9]. However, DE has many weaknesses as all other evolutionary search techniques. Generally, DE has a good global exploration ability that can reach the region of global optimum, but it is slow at exploitation of the solution [10]. Additionally, the parameters of DE are problem dependent and it is difficult to adjust them for different problems. Moreover, DE performance decreases as search space dimensionality increases [11]. Finally, the performance of DE deteriorates significantly when the problems of premature convergence and/or stagnation occur [11, 12]. The performance of DE basically depends on the mutation strategy and the crossover operator. Besides, the intrinsic control parameters (population size NP, scaling factor F, and the crossover rate CR) play a vital role in balancing the diversity of population and convergence speed of the algorithm. For the original DE, these parameters are user-defined and kept fixed during the run. However, many recent studies indicate that the performance of DE is highly affected by the parameter setting and the choice of the optimal values of parameters is always problem dependent. Moreover, prior to an actual optimization process, the traditional time-consuming trial-and-error method is used for fine-tuning the control parameters for each problem. Alternatively, in order to achieve acceptable results even for the same problem, different parameter settings along with different mutation schemes at different stages of evolution are needed. Therefore, some techniques have been designed to adjust control parameters in adaptive or self-adaptive manner instead of trial-and-error procedure plus new mutation rules have been developed to improve the search capability of DE [13–22]. Based on the above considerations, in this paper, we present a novel DE, referred to as ANDE, including two novel modifications: triangular mutation rule and self-adaptive scheme for gradual change of CR values. In ANDE, a novel triangular mutation rule can balance the global exploration ability and the local exploitation tendency and enhance the convergence rate of the algorithm. Furthermore, a novel adaptation scheme for CR is developed that can benefit from the past experience of the individuals in the search space during evolution process. Scaling factors are produced according to a uniform distribution to balance the global exploration and local exploitation during the evolution process. ANDE has been tested on 20 benchmark test functions developed for the 2010 IEEE Congress on Evolutionary Computation (IEEE CEC 2010) [1]. The experimental results indicate that the proposed algorithm and its two versions are highly competitive algorithms for solving large scale global optimization problems. The remainder of this paper is organized as follows. Section 2 briefly introduces DE and its operators. Section 3 reviews the related work. Then, ANDE is presented in Section 4. The experimental results are given in Section 5. Section 6 discusses the effectiveness of the proposed modifications. Finally, the conclusions and future works are drawn in Section 7.

2. Differential Evolution (DE)

This section provides a brief summary of the basic Differential Evolution (DE) algorithm. In simple DE, generally known as DE/rand/1/bin [4, 23], an initial random population consists of NP vectors $\vec{X}_i$, $\forall i = 1, 2, \ldots, \text{NP}$, and is randomly generated according to a uniform distribution within the lower and upper boundaries (x_j^L, x_j^U). After initialization, these individuals are evolved by DE operators (mutation and crossover) to generate a trial vector. A comparison between the parent and its trial vector is then done to select the vector which should survive to the next generation [9]. DE steps are discussed below.

2.1. Initialization. In order to establish a starting point for the optimization process, an initial population must be created. Typically, each decision variable in every vector of the initial population is assigned a randomly chosen value from the boundary constraints:

$$x_{ij}^0 = x_j^L + \text{rand}_j \cdot \left(x_j^U - x_j^L\right), \tag{2}$$

where rand_j denotes a uniformly distributed number between $[0, 1]$, generating a new value for each decision variable.

2.2. Mutation. At generation G, for each target vector x_i^G, a mutant vector v_i^{G+1} is generated according to the following:

$$v_i^{G+1} = x_{r_1}^G + F * \left(x_{r_2}^G - x_{r_3}^G\right), \quad r_1 \neq r_2 \neq r_3 \neq i \tag{3}$$

with randomly chosen indices $r_1, r_2, r_3 \in \{1, 2, \ldots, \text{NP}\}$. F is a real number to control the amplification of the difference vector $(x_{r_2}^G - x_{r_3}^G)$. According to Price et al. [24], the range of

F is in $[0, 2]$. In this work, if a component of a mutant vector violates search space, the value of this component is generated newly using (2).

2.3. Crossover. There are two main crossover types, binomial and exponential. In the binomial crossover, the target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector u_i^{G+1}.

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, & \text{rand}(j) \le \text{CR or } j = \text{randn}(i), \\ x_{ij}^{G}, & \text{rand}(j) > \text{CR and } j \ne \text{randn}(i), \end{cases} \quad (4)$$

where $j = 1, 2, \ldots, D$; rand$(j) \in [0, 1]$ is the jth evaluation of a uniform random generator number. CR $\in [0, 1]$ is the crossover rate; randn$(i) \in \{1, 2, \ldots, D\}$ is a randomly chosen index which ensures that u_i^{G+1} gets at least one element from v_i^{G+1}; otherwise no new parent vector would be produced and the population would not alter.

In an exponential crossover, an integer value l is randomly chosen within the range $\{1, D\}$. This integer value acts as a starting point in $\vec{x}_{j,G}$, from where the crossover or exchange of components with $\vec{V}_{i,G+1}$ starts. Another integer value L (denotes the number of components) is also chosen from the interval $\{1, D - l\}$. The trial vector $(\vec{u}_{i,G+1})$ is created by inheriting the values of variables in locations l to $l + L$ from $\vec{V}_{i,G+1}$ and the remaining ones from $\vec{x}_{j,G}$.

2.4. Selection. DE adapts a greedy selection strategy. If and only if the trial vector u_i^{G+1} yields fitness function value as good as or a better than x_i^G, then u_i^{G+1} is set to x_i^{G+1}. Otherwise, the old vector x_i^G is retained. The selection scheme is as follows (for a minimization problem):

$$x_i^{G+1} = \begin{cases} u_i^{G+1}, & f\left(u_i^{G+1}\right) \le f\left(x_i^G\right) \\ x_i^G, & \text{otherwise.} \end{cases} \quad (5)$$

A detailed description of standard DE algorithm is given in Algorithm 1.

3. Related Work

As previously mentioned, during the past few years, LSGO has attracted much attention by the researches due to its significance as many real-world problems and applications are high-dimensional problems in nature. Basically, the current EA-based LSGO research can be classified into two categories:

(i) Cooperative Coevolution (CC) framework algorithms or divide-and-conquer methods

(ii) Noncooperative Coevolution (CC) framework algorithms or no divide-and-conquer methods

Cooperative Coevolution (CC) has become a popular and effective technique in evolutionary algorithms (EAs) for large scale global optimization since its initiation in the publication of Potter and Jong [25]. The main idea of CC is to partition the LSGO problem into a number of subproblems; that is, the decision variables of the problem are divided into smaller subcomponents, each of which is optimized using a separate EA. By using this divide-and-conquer method, the classical EAs are able to effectively solve many separable problems [25]. CC show better performance on separable problems but deteriorated on nonseparable problems because the interacting variables could not be grouped in one subcomponent. Recently, different versions of CC-based EAs have been developed and shown excellent performance. Yang et al. [26] proposed a new decomposition strategy called random grouping as a simple way of increasing the probability of grouping interacting variables in one subcomponent. According to this strategy, without any prior knowledge of the nonseparability of a problem, subdivide n-dimensional decision vector into m s-dimensional subcomponents. Later, Omidvar et al. [27] proposed DECC-DML algorithm which is a Differential Evolution algorithm adopting CC frame. They suggested a new decomposition strategy called delta grouping. The central idea of this technique was that the improvement interval of interacting variables would be limited if they were in different subcomponents. Delta method measures the averaged difference in a certain variable across the entire population and uses it for identifying interacting variables. The experimental results show that this new method is more effective than the existing random grouping method. However, DECC-DML is less efficient on nonseparable functions with more than one group of rotated variables. Likewise, Wang and Li [28] proposed another CC-based technique, named EOEA, to handle LSGO problems, in which the search procedure is divided into two stages: (1) the global shrinking stage and (2) the local exploitation stage. The objective of the first stage is to shrink the searching scope to the promising area as quickly as possible by using an EDA based on mixed Gaussian and Cauchy models (MUEDA) [29], while, to achieve the second objective, CC-based algorithm, different from the previous CC- based methods, is adopted to explore the limited area extensively to find solution as better as possible. Compared with some previous LSGO algorithms, EOEA demonstrates better performance. Many CC-based algorithms have been developed during the past decade such as FEPCC [30], DECC-I, DECC-II [31], MLCC [32], DEwSaCC [33], and CPSO [34]. On the other hand, there are many other approaches that optimize LSGO problems as a whole; that is, no divide-and-conquer methods was used. Actually, it is considered a challenging task as it needs to develop novel evolutionary operators that can promote and strengthen the capability of the algorithms to improve the overall optimization process in high-dimensional search space. In [35], Korošec et al. proposed an Ant-Colony Optimization- (ACO-) based algorithm for solving LSGO problems with continuous variables, labeled Differential Ant-Stigmergy Algorithm (DASA). The DASA transforms a real-parameter optimization problem into a graph-search problem, where the parameters' differences assigned to the graph vertices are used to navigate through the search space. Brest et al. [36] presented self-adaptive Differential Evolution algorithm (jDElsgo). In this approach, self-adaptive F and Cr control parameters and

(01) Begin
(02) $G = 0$
(03) Create a random initial population $\vec{x}_i^G \ \forall i, i = 1, \ldots, \text{NP}$
(04) Evaluate $f(\vec{x}_i^G) \ \forall i, i = 1, \ldots, \text{NP}$
(05) **For** $G = 1$ to GEN **Do**
(06) **For** $i = 1$ to NP **Do**
(07) Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, \text{NP}]$
(08) $j_{\text{rand}} = \text{randint}(1, D)$
(09) **For** $j = 1$ to D **Do**
(10) **If** $(\text{rand}_j[0, 1] < \text{CR}$ **or** $j = j_{\text{rand}})$ **Then**
(11) $u_{i,j}^{G+1} = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$
(12) **Else**
(13) $u_{i,j}^{G+1} = x_{i,j}^G$
(14) **End If**
(15) **End For**
(16) **If** $(f(\vec{u}_i^{G+1}) \leq f(\vec{x}_i^G))$ **Then**
(17) $\vec{x}_i^{G+1} = \vec{u}_i^{G+1}$
(18) **Else**
(19) $\vec{x}_i^{G+1} = \vec{x}_i^G$
(20) **End If**
(21) **End For**
(22) $G = G + 1$
(23) **End For**
(24) **End**

Algorithm 1: Description of standard DE algorithm. rand[0, 1) is a function that returns a real number between 0 and 1. randint(min, max) is a function that returns an integer number between min and max. NP, GEN, CR, and F are user-defined parameters. D is the dimensionality of the problem.

"rand/1/bin" strategy along with population size reduction mechanism are used. Similarly, Wang et al. [37] introduced a sequential Differential Evolution (DE) enhanced by neighborhood search (SDENS), where hybrid crossover strategy "rand/1/bin" and "rand/1/exp" are used. In order to search the neighbors of each individual, two trial individuals by local and global neighborhood search strategies are created. Then, the fittest one among the current individual and the two created trial individuals is selected as a new current individual. Molina et al. [38] put forward a memetic algorithm based on local search chains, named MA-SW-CHAINS, which assigned local search intensity to each individual depending on its features by changing different local search applications. Zhao et al. [39] proposed a hybrid approach called (DMS-PSO-SHS) by combining the dynamic multiswarm particle swarm optimizer (DMS-PSO) with a subregional harmony search (SHS). A modified multitrajectory search (MTS) algorithm is also applied frequently on several selected solutions. In addition, an external memory of selected past solutions is used to enhance the diversity of the swarm. Generally, the proposed ANDE algorithm belongs to this category.

4. ANDE Algorithm

In this section, we outline a novel DE algorithm, ANDE, and explain the steps of the algorithm in detail.

4.1. Triangular Mutation Scheme. Storn and Price [4, 24] proposed the basic mutation scheme DE/rand/1. In fact, from the literature [9], it is considered as the most successful and widely used operator. Virtually, the main idea behind this strategy is that the three vectors are randomly selected from the population to form the mutation and the base vector is then chosen at random among the three. The difference vector is formed using the other two vectors added to the base vector. Obviously, it can be seen that the basic mutation scheme has excellent ability to maintain population diversity and global search capability as it is not directed to any specific search direction. However, the convergence speed of DE algorithms significantly slows down [15]. In the same context, DE/rand/2 strategy, which is similar to the basic scheme with another difference vector that is formed by extra two vectors, has better perturbation than original mutation with one difference vector [15]. Consequently, it is better than the

DE/rand/1/bin strategy as it can provide much more various differential trial vectors. On the other hand, there is another type of mutations which is called greedy strategies such as DE/best/1, DE/best/2, and DE/current-to-best/1. Actually, in order to increase the local search tendency that improves the convergence behavior of the algorithm, this type is based on benefits from the best solution found so far in the evolutionary process by incorporating it into the mutation operator. Nonetheless, the population diversity and exploration capability of the algorithm may be deteriorated or may be completely lost at the early stage of the optimization process that causes problems such stagnation and/or premature convergence. Accordingly, the recent studies proposed a strategy candidate pool that includes many mutation schemes that are different in structure and have distinct optimization capabilities to overcome the shortcomings of both types of mutation strategies. Then, it is combined with different control parameter adaptation rules to deal with various types of problems with different features at different stages of evolution [15, 17, 40]. Contrarily, taking into consideration the weakness of existing greedy strategies, Mohamed [16] introduced a new Differential Evolution (DE) algorithm, named JADE, to improve optimization performance by implementing a new mutation strategy "DE/current-to-pbest" with optional external archive and by updating control parameters in an adaptive manner. Simulation results show that JADE was better than, or at least competitive to, other classic or adaptive DE algorithms such as particle swarm and other evolutionary algorithms from the literature in terms of convergence performance. Consequently, from the literature, there are a few attempts in developing new mutations rule. Therefore, in this paper, a new triangular mutation rule is introduced. The proposed mutation can significantly balance both the global exploration ability and the local exploitation tendency that in turn will improve the convergence rate of the algorithm. The proposed mutation strategy is based on the convex combination vector of the triplet defined by the three randomly chosen vectors and three difference vectors between the tournament best, better, and worst selected vectors. The proposed mutation vector is generated in the following manner:

$$v_i^{G+1} = \overline{x}_c^G + F1 \cdot \left(x_{\text{best}}^G - x_{\text{better}}^G\right) + F2 \cdot \left(x_{\text{best}}^G - x_{\text{worst}}^G\right) + F3 \cdot \left(x_{\text{better}}^G - x_{\text{worst}}^G\right), \tag{6}$$

where $\overline{x}_c^G$ is a convex combination vector of the triangle and $F1$, $F2$, and $F3$ are the mutation factors that are associated with x_i and are independently generated according to uniform distribution in $(0, 1)$ and x_{best}^G, x_{better}^G, and x_{worst}^G are the three tournament best, better, and worst randomly selected vectors, respectively. The convex combination vector $\overline{x}_c^G$ of the triangle is a linear combination of the three randomly selected vectors and is defined as follows:

$$\overline{x}_c^G = w_1 \cdot x_{\text{best}} + w_2 \cdot x_{\text{better}} + w_3 \cdot x_{\text{worst}}, \tag{7}$$

where the real weights w_i satisfy $w_i \geq 0$ and $\sum_{i=1}^{3} w_i = 1$, where the real weights w_i are given by $w_i = p_i / \sum_{i=1}^{3} p_i$, $i = 1, 2, 3$, where $p_1 = 1$, $p_2 = \text{rand}(0.75, 1)$, and $p_3 = \text{rand}(0.5, p(2))$, $\text{rand}(a, b)$ is a function that returns a real number between a and b, where a and b are not included. For unconstrained optimization problems at any generation $g > 1$, for each target vector, three vectors are randomly selected; then sort them in ascending according to their objective function values and assign w_1, w_2, and w_3 to x_{best}^G, x_{better}^G, and x_{worst}^G, respectively. Without loss of generality, we only consider minimization problem.

Obviously, from mutation equation (6), it can be observed that the incorporation of the objective function value in the mutation scheme has two benefits. Firstly, the perturbation part of the mutation is formed by the three sides of the triangle in the direction of the best vector among the three randomly selected vectors. Therefore, the directed perturbations in the proposed mutation resembles the concept of gradient as the difference vectors are directed from the worst to the better to the best vectors [41]. Thus, it is considerably used to explore the landscape of the objective function being optimized in different subregion around the best vectors within search space through optimization process. Secondly, the convex combination vector $\overline{x}_c^G$ is a weighted sum of the three randomly selected vectors where the best vector has the significant contribution. Therefore, $\overline{x}_c^G$ is extremely affected and biased by the best vector more than the remaining two vectors. Consequently, the global solution can be easily reached if all vectors follow the direction of the best vectors; besides they also follow the opposite direction of the worst vectors among the randomly selected vectors. Indeed, the new mutation process exploits the nearby region of each $\overline{x}_c^G$ in the direction of $(x_{\text{best}}^G - x_{\text{worst}}^G)$ for each mutated vector. In a nutshell, it concentrates the exploitation of some subregions of the search space. Thus, it has better local search tendency so it accelerates the convergence speed of the proposed algorithm. Besides, the global exploration ability of the algorithm is significantly enhanced by forming many different sizes and shapes of triangles in the feasible region through the optimization process. Thus, the proposed directed mutation balances both global exploration capability and local exploitation tendency.

Thus, since the proposed directed mutation balances both global exploration capability and local exploitation tendency while the basic mutation favors exploration only, the probability of using the proposed mutation is twice as much the probability of applying the basic rule. The new mutation strategy is embedded into the DE algorithm and combined with the basic mutation strategy DE/rand/1/bin as follows.

If $(u(0, 1) \leq (2/3))$ then

$$v_i^{G+1} = \overline{x}_c^G + F1 \cdot \left(x_{\text{best}}^G - x_{\text{better}}^G\right) + F2 \cdot \left(x_{\text{best}}^G - x_{\text{worst}}^G\right) + F3 \cdot \left(x_{\text{better}}^G - x_{\text{worst}}^G\right). \tag{8}$$

Else

$$v_{i,j}^{G+1} = x_{r_1,j}^G + F \cdot \left(x_{r_1,j}^G - x_{r_3,j}^G\right), \tag{9}$$

where F is a uniform random variable in $[(-1, 0) \cup (0, 1)]$, $u(0, 1)$ returns a real number between 0 and 1 with uniform

random probability distribution. From the abovementioned scheme, it can be realized that for each vector only one of the two strategies is used for generating the current trial vector, depending on a uniformly distributed random value within the range $(0, 1)$. For each vector, if the random value is greater than 2/3, then the basic mutation is applied. Otherwise, the proposed one is performed. It is noteworthy mentioning that the proposed triangular mutation and the trigonometric mutation proposed by Fan and Lampinen [23] use three randomly selected vectors but they are completely different in the following two main points.

(1) The proposed mutation strategy is based on the convex combination vector (weighted mean) of the triplet defined by the three randomly chosen vectors (as a donor) and three difference vectors between the tournament best, better, and worst selected vectors (they are directed difference, i.e., resembling the concept of gradient as the difference vectors are directed from the worst to the better to the best vectors). However, the trigonometric mutation is based on the center point (the mean) of the hypergeometric triangle defined by the three randomly chosen vectors. The perturbation to be imposed to the donor is then made up with a sum of three weighted vector differentials that are randomly constructed (not directed).

(2) With respect to the scaling factors in the proposed algorithm, at each generation G, the scale factors $F1$, $F2$, and $F3$ of each individual target vector are independently generated according to uniform distribution in $(0, 1)$ to enrich the search behavior. However, the scaling factors in the trigonometric mutation are constants; at each generation G, the scale factors of each individual target vector are computed as the ratio of the objective function value of each vector divided by the sum of the objective function values of the three vectors (sum equals 1). Therefore, it is obviously deduced that the trigonometric mutation operation is a rather greedy operator since it biases the new trial solution strongly in the direction where the best one of three individuals chosen for the mutation is. Therefore, the trigonometric mutation can be viewed as a local search operator and the perturbed individuals are produced only within a trigonometric region that is defined by the triangle used for a mutation operation [23]. Consequently, it is easily trapped in local points with multimodal problems and it may be also stagnated as it has not an exploration capability to seek the whole search space. However, the proposed triangular mutation has both the exploration capability and the exploitation tendency because directed perturbation in the proposed mutation resembles the concept of gradient as the difference vectors are directed from the worst to the better to the best vectors. Thus, it is considerably used to explore the landscape of the objective function being optimized in different subregion around the best vectors outside the trigonometric region that is defined by the triangle used for a mutation operation within search space through optimization process. Secondly, the convex combination vector $\overline{x}_c^G$ is a weighted sum of the three randomly selected vectors where the best vector has the significant contribution. Therefore, $\overline{x}_c^G$ is extremely affected and biased by the best vector more than the remaining two vectors. Consequently, the global solution can be easily reached if all vectors follow the direction of the best vectors; besides they also follow the opposite direction of the worst vectors among the randomly selected vectors. Indeed, the new mutation process exploits the nearby region of each $\overline{x}_c^G$ in the direction of $(x_{\text{best}}^G - x_{\text{worst}}^G)$ for each mutated vector. In a nutshell, it concentrates the exploitation of some subregions of the search space. Thus, it has better local search tendency so it accelerates the convergence speed of the proposed algorithm. Besides, the global exploration ability of the algorithm is significantly enhanced by forming many different sizes and shapes of triangles in the feasible region through the optimization process.

4.2. Parameter Adaptation Schemes in ANDE. The successful performance of DE algorithm is significantly dependent upon the choice of its three control parameters: the scaling factor F and crossover rate CR and population size NP [4, 24]. In fact, they have a vital role because they greatly influence the effectiveness, efficiency, and robustness of the algorithm. Furthermore, it is difficult to determine the optimal values of the control parameters for a variety of problems with different characteristics at different stages of evolution. In the proposed ANDE algorithm, NP is kept as a user-specified parameter since it highly depends on the problem complexity. Generally speaking, F is an important parameter that controls the evolving rate of the population; that is, it is closely related to the convergence speed [15]. A small F value encourages the exploitation tendency of the algorithm that makes the search focus on neighborhood of the current solutions; hence it can enhance the convergence speed. However, it may also lead to premature convergence [41]. On the other hand, a large F value improves the exploration capability of the algorithm that can make the mutant vectors distributed widely in the search space and can increase the diversity of the population [40]. However, it may slow down the search [41]. With respect to the scaling factors in the proposed algorithm, at each generation G, the scale factors $F1$, $F2$, and $F3$ of each individual target vector are independently generated according to uniform distribution in $(0, 1)$ to enrich the search behavior. The constant crossover (CR) reflects the probability with which the trial individual inherits the actual individual's genes, that is, which and how many components are mutated in each element of the current population [17, 41]. The constant crossover CR practically controls the diversity of the population [40]. As a matter of fact, if CR is high, this will increase the population diversity. Nevertheless, the stability of the algorithm may reduce. On the other hand, small values of CR increase the possibility of stagnation that may weaken the exploration ability of the algorithm to open up new search space. Additionally, CR is usually more sensitive to problems with different characteristics such as unimodality and multimodality, separable and nonseparable problems. For separable problems, CR from the range $(0, 0.2)$ is the best while for multimodal parameter dependent problems CR in the range $(0.9, 1)$ is suitable [42]. On the other hand, there are wide varieties of approaches for adapting or self-adapting control parameters values through optimization process. Most of these methods are based on generating random values from

uniform, normal, or Cauchy distributions or by generating different values from predefined parameter candidate pool. Besides, they use the previous experience (of generating better solutions) to guide the adaptation of these parameters [11, 15–17, 19, 40, 43–46]. The presented work proposed a novel self-adaptation scheme for CR.

The core idea of the proposed self-adaptation scheme for the crossover rate CR is based on the following fundamental principle. In the initial stage of the search process, the difference among individual vectors is large because the vectors in the population are completely dispersed or the population diversity is large due to the random distribution of the individuals in the search space that requires a relatively smaller crossover value. Then, as the population evolves through generations, the diversity of the population decreases as the vectors in the population are clustered because each individual gets closer to the best vector found so far. Consequently, in order to maintain the population diversity and improve the convergence speed, crossover should be gradually utilized with larger values along with the generations of evolution increased to preserve well genes in so far as possible and promote the convergence performance. Therefore, the population diversity can be greatly enhanced through generations. However, there is no appropriate CR value that balance both the diversity and convergence speed when solving a given problem during overall optimization process. Consequently, to address this problem and following the SaDE algorithm [15], in this paper, a novel adaptation scheme for CR is developed that can benefit from the past experience through generations of evolutionary.

Crossover Rate Adaptation. At each generation G, the crossover probability CR_i of each individual target vector is independently generated randomly from pool A according to uniform distribution and Procedure 1 exists through generations.

In Procedure 1, A is the pool of values of crossover rate CR that changes during and after the learning period LP; we set LP = 10% of GEN, G is the current generation number; GEN is the maximum number of generations. The lower and upper limits of the ranges for (G) are experimentally determined; CR_Flag_List[i] is the list that contains one of two binary values (0, 1) for each individual i through generation G, where 0 represents failure, no improvement, when the target vector is better than the trial vector during and after the learning period and 1 represents success, improvement, when the trial vector is better than the target vector during and after the learning period, the failure_counter_list[i] is the list that monitors the working of individuals in terms of fitness function value during generations after completion of the learning period, and if there is no improvement in fitness, then the failure counter of this target vector is increased by unity. This process is repeated until it achieves prespecified value of Max_failure_counter which assigned a value 20 that is experimentally determined; CR_Ratio_List[k] is the list that records the relative change improvement ratios between the trial and target objective function values with respect to each value k of the pool of values A of CR through generation G. It can be clearly seen from procedure 1 that, at $G = 1$, CR = 0.05 for each target vector and then, at each generation G, if the generated trial vector produced is better than the target vector, the relative change improvement ratio (RCIR) associated with this CR value is computed and the correspondence ratio is updated. On the other hand, during the learning period, if the target vector is better than the trial vector, then the CR value is chosen randomly from the associated pool A of CR values, that is, gradually added more values, according to generation number and hence, for this CR value, there is no improvement and its ratio remains unchanged. However, after termination of the learning period, if the target vector is better than the trial vector, that is, if there is no improvement in fitness, then the failure_counter is increased by one in each generation till this value achieves a prespecified value of Max_failure_counter which assigned a value 20; then this CR value should change to a new value that is randomly selected from the pool A of CR values that is taken in range 0.1–0.9 in steps of 0.1 and 0.05 and 0.95 are also included as lower and upper values, respectively. Note that the RCIR is only updated if there is an improvement. Otherwise, it remains constant. Thus, the CR value with maximum ratio is continuously changing according to the evolution process at each subsequent generation. In fact, although all test problems included in this study have optimum of zero, the absolute value is used in calculating RCIR as a general rule in order to deal with positive, negative, or mixed values of objective function.

Concretely, Procedure 1 shows that, during the first half of the learning period, the construction of pool A of CR values ensures the diversity of the population such that the crossover probability for ith individual target increases gradually in staircase along with the generations of evolution process increased, taking into consideration that the probability of chosen small CR values is greater than the probability of chosen larger CR values as the diversity of the population is still large. Additionally, in the second half of the learning period, larger values of 0.9 and 0.95 are added to the pool as it favors nonseparable functions. However, all the values have an equally likely chance of occurrence to keep on the diversity with different values of CR. Consequently, the successful CR values with high relative change improvement ratio in this period will survive to be used in the next generations of the optimization process until it fails to achieve improvement for a specific value of 20; then it must be changed randomly by a new value. Thus, the value of CR is adaptively changed as the diversity of the population changes through generations. Distinctly, it varies from one individual to another during generations, and also it is different from one function to another one being optimized. Generally, adaptive control parameters with different values during the optimization process in successive generations enrich the algorithm with controlled-randomness which enhances the global optimization performance of the algorithm in terms of exploration and exploitation capabilities. Therefore, it can be concluded that the proposed novel adaptation scheme for gradual change of the values of the crossover rate can excellently benefit from the past experience of the individuals in the search space during evolution process which in turn can considerably balance the common trade-off between the population diversity and

If ((CR_Flag_List[i] = 0) and (G <= LP)), If the target vector is better than the trial vector during the learning period, then:

$$\mathrm{Cr}_i = \begin{cases} \text{Randomly select one value from } A,\ A = [0.05], & 0 \le G < \left(\frac{1}{6}\right) * (\mathrm{LP}) \\ \text{Randomly select one value from } A,\ A = [0.05, 0.1, 0.2], & \left(\frac{1}{6}\right) * (\mathrm{LP}) \le G < \left(\frac{1}{4}\right) * (\mathrm{LP}) \\ \text{Randomly select one value from } A,\ A = [0.05, 0.1, 0.2, 0.3, 0.4], & \left(\frac{1}{4}\right) * (\mathrm{LP}) \le G < \left(\frac{1}{3}\right) * (\mathrm{LP}) \\ \text{Randomly select one value from } A,\ A = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6], & \left(\frac{1}{3}\right) * (\mathrm{LP}) \le G < \left(\frac{5}{12}\right) * (\mathrm{LP}) \\ \text{Randomly select one value from } A,\ A = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8], & \left(\frac{5}{12}\right) * (\mathrm{LP}) \le G < \left(\frac{1}{2}\right) * (\mathrm{LP}) \\ \text{Randomly select one value from } A,\ A = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95], & \left(\frac{1}{2}\right) * (\mathrm{LP}) \le G < (1) * (\mathrm{LP}) \end{cases}$$

Else If ((CR_Flag_List[i] = 0) and (G > LP)), If the target vector is better than the trial vector after the learning period, then:
If the failure_counter_list[i] = Max_failure_counter,
Randomly select one value from list A, A = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95].
else
failure_counter_list[i]= *failure_counter_list*[i] + 1;
End If
Else ((CR_Flag_List[i] = 1)), If the trial vector is better than the target vector through generations, then:
Select the CR value from A with maximum relative change improvement ratio (RCIR) from CR_Ratio_List[k].
End IF

Procedure 1

convergence speed. The pseudocode of ANDE is presented in Algorithm 2.

It is worth noting that although this work is an extension and modification of our previous work in [47], there are significant differences as follows: (1) Previous work in [47] is proposed for solving small scale unconstrained problems (i.e., 10, 30, and 50 dimensions), whereas this work is proposed for solving large scale unconstrained problems with 1000 dimensions. (2) The crossover rate in [47] is given by a dynamic nonlinear increased probability scheme, but in this work a novel self-adaptation scheme for CR is developed that can benefit from the past experience through generations of evolutionary. (3) In [47], only one difference vector between the best and the worst individuals among the three randomly selected vectors with one scaling factor, uniformly random number in $(0, 1)$, is used in the mutation, but in this work three difference vectors between the tournament best, better, and worst selected vectors with corresponding three scaling factors, which are independently generated according to uniform distribution in $(0, 1)$, are used in the mutation scheme. (4) The triangular mutation rule is only used in this work, but in the previous work [47], the triangular mutation strategy is embedded into the DE algorithm and combined with the basic mutation strategy DE/rand/1/bin through a nonlinear decreasing probability rule. (5) In previous work [47] a restart mechanism based on Random Mutation and modified BGA mutation is used to avoid stagnation or premature convergence, whereas this work does not.

5. Experimental Study

5.1. Benchmark Functions. The performance of the proposed ANDE algorithm has been tasted on 20 scalable optimization functions for the CEC 2010 special session and competition on large scale in global optimization. A detailed description of these test functions can be found in [1]. These 20 test functions can be divided into four classes:

(1) Separable functions F_1–F_3

(2) Partially separable functions, in which a small number of variables are dependent while all the remaining ones are independent ($m = 50$) F_4–F_8

(3) Partially separable functions that consist of multiple independent subcomponents, each of which is m-nonseparable ($m = 50$) F_9–F_{18}

(4) Fully nonseparable functions F_{19}-F_{20},

where the sphere function, the rotated elliptic function, *Schwefels* Problem 1.2, Rosenbrock function, the rotated Rastrigin function, and the rotated Ackley function are used as the basic functions. The control parameter used to define the degree of separability of a given function in the given test suite is set as $m = 50$. The dimensions (D) of functions are 1000.

5.2. Parameter Settings and Involved Algorithms. To evaluate the performance of algorithm, experiments were conducted on the test suite. We adopt the solution error measure $f((x) - (x^*))$, where x is the best solution obtained by algorithms in one run and x^* is well-known global optimum of each benchmark function and is recorded after $1.2e + 05$, $6.0e + 05$, and $3.0e + 06$ function evaluations (FEs), all experiments for each function run 25 times independently, and statistical results are provided including the best, median, mean, worst results and the standard deviation. The population size in ANDE was set to 50. The learning period (LP) and the maximum

(01) Begin
(02) $G = 0$
(03) Create a random initial population $\vec{x}_i^G$ $\forall i, i = 1, \ldots, \text{NP}$,
set the Learning Period (LP) = 10% GEN, set the Max_failure_counter = 20.
For each $\vec{x}_i^G$, set the failure_counter_list$[i] = 0$, set the CR_Flag_List$[i] = 0$,
For each CR values in the list, set the CR_Ratio_List$[k] = 0$ $\forall k, k = 1, \ldots, 11$.
(04) Evaluate $f(\vec{x}_i^G)$ $\forall i, i = 1, \ldots, \text{NP}$
(05) **For** $G = 1$ to GEN **Do**
(06) **For** $i = 1$ to NP **Do**
(07) Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, \text{NP}]$
(08) $j_{\text{rand}} = \text{randint}(1, D)$
(11) Compute the (crossover rate) Cr_i according to Procedure 1.
(12) **For** $j = 1$ to D **Do**
(13) **If** $(\text{rand}_j[0, 1] < \text{CR}$ **or** $j = j_{\text{rand}})$ **Then**
(14) **If** $(\text{rand}[0, 1] <= (2/3))$ **Then** (Use New Triangular Mutation Scheme)
(15) Determine the tournament x_{best}^G, x_{better}^G and x_{worst}^G based on $f(\vec{x}_i^G)$, $i = 1, 2, 3$
(three randomly selected vectors) and compute $\overline{x}_c^G$ according to eq. (2).
(16) $u_{i,j}^{G+1} = \overline{x}_{c,j}^G + F_1(x_{\text{best},j}^G - x_{\text{better},j}^G) + F_2(x_{\text{best},j}^G - x_{\text{worst},j}^G) + F_3(x_{\text{better},j}^G - x_{\text{worst},j}^G)$
(17) **Else**
(18) $u_{i,j}^{G+1} = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$
End If
(15) **Else**
(16) $u_{i,j}^{G+1} = x_{i,j}^G$
(17) **End If**
(18) **End For**
(19) **If** $(f(\vec{u}_i^{G+1}) \leq f(\vec{x}_i^G))$ **Then**
(20) $\vec{x}_i^{G+1} = \vec{u}_i^{G+1}$, $(f(\vec{x}_i^{G+1}) = f(\vec{u}_i^{G+1}))$
If $(f(\vec{u}_i^{G+1}) \leq f(\vec{x}_{\text{best}}^G))$ **Then**
$\vec{x}_{\text{best}}^{G+1} = \vec{u}_i^{G+1}$, $(f(\vec{x}_{\text{best}}^{G+1}) = f(\vec{u}_i^{G+1}))$
End
CR_Flag_List$[i]$**= 1**
The relative change improvement ratio (RCIR) is updated
(21) $CR\_Ratio\_List[k] = CR\_Ratio\_List[k] + (1 - \min(|f(\vec{u}_i^{G+1})|, |f(\vec{x}_i^G)|)/\max(|f(\vec{u}_i^{G+1})|, |f(\vec{x}_i^G)|))$.
Else
(22) $\vec{x}_i^{G+1} = \vec{x}_i^G$
CR_Flag_List$[i] =$ **0**
(23) **End If**
(24) **End For**
(25) $G = G + 1$
(26) **End For**
(27) **End**

Algorithm 2: Description of ANDE algorithm.

failure counter (MFC) are set to 10% of total generations and 20 generations, respectively. For separable functions F_1–F_3, CR is chosen to be 0.05 as they are separable functions. ANDE was compared to population-based algorithms that were all tested on this test suite in this competition. These algorithms are

(i) DE Enhanced by Neighborhood Search for Large Scale Global Optimization (SDENS) [37],

(ii) Large Scale Global Optimization using Self-Adaptive Differential Evolution algorithm (jDElsgo) [36],

(iii) Cooperative Coevolution with Delta Grouping for Large Scale Nonseparable Function Optimization (DECC-DML) [27],

(iv) the Differential Ant-Stigmergy Algorithm for Large Scale Global Optimization (DASA) [35],

(v) two-stage based Ensemble Optimization for Large Scale Global Optimization (EOEA) [28],

(vi) Memetic Algorithm Based on Local Search Chains for Large Scale Continuous Global Optimization (MA-SW-CHAINS) [38],

(vii) Dynamic Multiswarm Particle Swarm Optimizer with Subregional Harmony Search (DMS-PSO-SHS) [39].

Note that since paper [48] was not accepted (based on private communication with the author) so it was excluded from this comparison. To compare the solution quality from a statistical angle of different algorithms and to check the behavior

of the stochastic algorithms [38, 49], the results are compared using multiproblem Wilcoxon signed-rank test at a significance level 0.05. Wilcoxon signed-rank Test is a nonparametric statistical test that allow us to judge the difference between paired scores when it cannot make the assumption required by the paired-samples t-test; for example, the population should be normally distributed, where R^+ denotes the sum of ranks for the test problems in which the first algorithm performs better than the second algorithm (in the first column), and R^- represents the sum of ranks for the test problems in which the first algorithm performs worse than the second algorithm (in the first column). Larger ranks indicate larger performance discrepancy. The numbers in better, equal, and worse columns denote the number of problems in which the first algorithm is better than, equal to, or worse than the second algorithm. As a null hypothesis, it is assumed that there is no significance difference between the mean results of the two samples. Whereas the alternative hypothesis is that there is significance in the mean results of the two samples, the number of test problems $N = 20$ for $1.25e + 05$, $6.00e + 05$, and $3.00e + 006$ function evaluations and 5% significance level. Use the smaller values of the sums as the test value and compare it with the critical value or use the p value and compare it with the significance level. Reject the null hypothesis if the test value is less than or equal to the critical value or if the p value is less than or equal to the significance level (5%). Based on the result of the test, one of three signs (+, −, and ≈) is assigned for the comparison of any two algorithms (shown in the last column), where (+) sign means the first algorithm is significantly better than the second, (−) sign means the first algorithm is significantly worse than the second, and (≈) sign means that there is no significant difference between the two algorithms. To perform comprehensive evaluation, the presentation of the experimental results is divided into three subsections. At first, the effectiveness of the proposed self-adaptive crossover rate scheme, modified basic differential evolution, and new triangular mutation scheme are evaluated. Second, overall performance comparisons between ANDE, ANDE-1, and ANDE-2 and other state-of-the-art DEs and non-DEs approaches are provided. Finally, the effectiveness of the proposed modifications and parameter settings on the performance of ANDE algorithm is discussed.

5.3. Experimental Results and Discussions. Firstly, some trials have been performed to evaluate the benefits and effectiveness of the proposed new triangular mutation and self-adaptive crossover rate on the performance of ANDE. Two different versions of ANDE have been tested and compared against the proposed one denoted as ANDE-1 and ANDE-2.

(1) ANDE-1, which is same as ANDE except that the new triangular mutation scheme is only used

(2) ANDE-2, which is same as ANDE except that the basic mutation scheme is only used

The solution error of ANDE and the two variants ANDE-1 and ANDE-2 algorithms are recorded in $1.2e + 05$, $6.0e + 05$, and $3.0e + 06$ function evaluations (FEs); all experiments for each function run 25 times independently and statistical results are provided including the best, median, mean, and worst results and the standard deviation in the supplemental file (Tables S1–S3 in Supplementary Material available online at https://doi.org/10.1155/2017/7974218).

In this section, we compare directly the mean results obtained by ANDE, ANDE-1, and ANDE-2.

Tables S1, S2, and S3 contain the results obtained by ANDE, ANDE-1, and ANDE-2 in $1.2e + 05$, $6.0e + 05$, and $3.0e+06$ function evaluations (FEs), respectively. For remarking the best algorithm, best median and mean for each function are highlighted in boldface. The following characteristics can be clearly observed:

(i) In the majority of the functions, the differences between mean and median are small even in the cases when the final results are far away from the optimum, regardless of the number of function evaluations. That implies the ANDE and its versions are robust algorithms.

(ii) In many functions, results with FEs = $3.0E + 06$ are significantly better than with FEs = $6.0E+05$ and also results with FEs = $6.0E + 05$ are significantly better than with FEs = $1.2E+05$. Therefore, ANDE, ANDE-1, and ANDE-2 benefit from desired FEs and there are continual improvements until the maximum FEs are achieved.

From Table S1, we can see that all algorithms perform well and there is no significant difference between them. However, it can be clearly seen from Table S2 that ANDE-2 performs significantly better than ANDE and ANDE-1 algorithms on separable functions (F_1, F_2). For F_3 and single-group m-nonseparable functions (F_4–F_8), the performance of the three algorithms are almost similar with the exception of test function F_6 where ANDE-2 performs better than HDE and ANDE-2 algorithms. Regarding $D/2m$-group m-nonseparable functions (F_9–F_{13}), the performance of the three algorithms is almost similar for F_9. ANDE-2 outperforms ANDE and ANDE-1 on F_{10}, F_{11} while for F_{12}, it loses to them. For F_{13}, ANDE outperforms other two algorithms. For D/m-group m-nonseparable functions (F_{14}–F_{18}), ANDE and ANDE-1 perform best but ANDE-2 only performs better than ANDE and ANDE-1 on F_{16}. But results of ANDE-1 on F_{13} and F_{17} are relatively better than results of ANDE. As for fully nonseparable functions (F_{19}-F_{20}), it is obvious that ANDE and ANDE-1 perform better than ANDE-2. As for F_{20}, ANDE performs significantly better than ANDE-1 and ANDE-2 algorithms. As can be seen from Table S3, ANDE, ANDE-1, and ANDE-2 algorithms were able to nearly reach the global optimal solution with high consistency of the separable F_1 (unimodal) and F_3 (multimodal) functions. However, ANDE-2 performs better than ANDE and ANDE-1 on separable multimodal function F_2. Besides, ANDE, ANDE-1, and ANDE-2 got close to the optimum single-group m-nonseparable functions multimodal function F_6 while only ANDE and ANDE-1 were also very close to the optimal solution of unimodal function F_7. As for F_4, F_5, and F_8, ANDE-1 outperforms ANDE and ANDE-2. Regarding the

Table 1: Results of multiple-problem Wilcoxon's test for ANDE, ANDE-1, and ANDE-2 over all functions at a significance level 0.05 (with $1.25E + 05$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus ANDE-1	114	76	0.445	12	1	7	≈
ANDE versus ANDE-2	107	83	0.629	10	1	9	≈
ANDE-1 versus ANDE-2	117	93	0.654	10	0	10	≈

Table 2: Results of the multiple-problem Wilcoxon's test for ANDE, ANDE-1, and ANDE-2 over all functions at a significance level 0.05 (with $6.00E + 05$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus ANDE-1	92	118	0.627	12	0	8	≈
ANDE versus ANDE-2	170	40	0.015	12	0	8	+
ANDE-1 versus ANDE-2	165	45	0.025	11	0	9	+

Table 3: Results of multiple-problem Wilcoxon's test for ANDE, ANDE-1, and ANDE-2 over all functions at a significance level 0.05 (with $3.00E + 06$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus ANDE-1	85	124	0.467	12	0	8	≈
ANDE versus ANDE-2	171	39	0.014	14	0	6	+
ANDE-1 versus ANDE-2	165	45	0.025	11	0	9	+

remaining problems, the performance of ANDE and ANDE-1 is almost similar and they outperform ANDE-2 while ANDE-2 performs some better that ANDE and ANDE-1 on F_{10}, F_{15}, F_{16}, and F_{20}. On the other hand, convergence behavior is another important factor that must be considered in comparison among all proposed algorithms. Therefore, in order to analyze the convergence behavior of each algorithm compared, the convergence graph of the median run has been plotted for each test problem. Figure S1 presents the convergence characteristics of all algorithms. From Figure S1, it can be observed that the convergence behavior supports the abovementioned analysis and discussions. Finally, it is clearly deduced that the remarkable performance of ANDE and ANDE-1 algorithms is due to the proposed mutation scheme that has both exploration ability and exploitation tendency. Additionally, it also visible that the proposed self-adaptive crossover procedure enhances the performance of the basic DE algorithm and it significantly promotes the performance of ANDE and ANDE-1 algorithms. In order to investigate and compare the performance of the proposed algorithms ANDE, ANDE-1, and ANDE-2 at statistical level, multiproblem Wilcoxon signed-rank test at a significance level 0.05 is performed on mean errors of all problems (with $1.25E + 05$ FEs, $6.00E + 05$ FEs, and $3.00E + 06$ FEs) and the results are presented in Tables 1, 2, and 3, respectively. From Table 1, it can be obviously seen that there is no significant difference between ANDE-1, ANDE-2, and ANDE. Besides, it can be obviously seen from Tables 2 and 3 that ANDE and ANDE-1 are significantly better than ANDE-2. Besides, there is no significant difference between ANDE-1 and ANDE. Although ANDE-2 achieved good performance in the initial phase of the search, it cannot keep the same performance during the rest of the optimization phases as expected because it depends only on the basic mutation which has a lack of exploitation capability. On the other hand, ANDE and ANDE-1 have almost the same excellent performance due to the new triangular mutation.

In this section, we compare directly the mean results obtained by ANDE, ANDE-1, and ANDE-2 with the ones obtained by SDENS [37], jDElsgo [36], DECC-DML [27], DASA [35], EOEA [28], MA-SW-CHAINS [38], and DMS-PSO-SHS [39]. Tables S4–S6 contain the results obtained by ANDE, ANDE-1, and ANDE-2 in $1.2e + 05$, $6.0e + 05$, and $3.0e + 06$ function evaluations (FEs), respectively. For remarking the best algorithm, best median and mean for each function are highlighted in boldface. From these tables we have highlighted the following direct comparisons and conclusions.

(i) For many test functions, the worst results obtained by the proposed algorithms are better than the best results obtained by other algorithms with all FEs.

(ii) For many test functions, there is continuous improvement in the results obtained by our proposed algorithms, especially ANDE and ANDE-1, with all FEs while the results with FEs = $6.0E + 05$ are very close to the results with FEs = $3.0E + 06$ obtained by some of the compared algorithms which indicate that our proposed approaches are scalable enough and can balance greatly the exploration and exploitation abilities for solving high-dimensional problems until the maximum FEs are reached.

(iii) For many functions, the remarkable performance of ANDE and its two versions with FEs = $1.20E + 05$ and FEs = $6.0E + 05$ compared to the performance of other algorithms shows its fast convergence behavior.

Table 4: Results of multiple-problem Wilcoxon's test for ANDE, ANDE-1, and ANDE-2 versus SDENS, jDElsgo, DECC-DML, DASA, EOEA, MA-SW-CHAINS, and DMS-PSO-SHS over all functions at a significance level 0.05 (with $1.25E + 05$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus SDENS	203	7	0.000	19	0	11	+
ANDE versus jDElsgo	210	0	0.000	20	0	0	+
ANDE versus DECC-DML	162	48	0.033	14	0	6	+
ANDE versus DASA	44	166	0.023	6	0	14	−
ANDE versus EOEA	26	184	0.003	2	0	18	−
ANDE versus MA-SW-CHAINS	30	180	0.005	2	0	18	−
ANDE versus DMS-PSO-SHS	104	106	0.970	12	0	8	≈
ANDE-1 versus SDENS	210	0	0.000	20	0	0	+
ANDE-1 versus jDElsgo	210	0	0.000	20	0	0	+
ANDE-1 versus DECC-DML	149	61	0.100	13	0	7	≈
ANDE-1 versus DASA	44	166	0.023	6	0	14	−
ANDE-1 versus EOEA	61	149	0.100	4	0	16	≈
ANDE-1 versus MA-SW-CHAINS	19	191	0.001	2	0	18	−
ANDE-1 versus DMS-PSO-SHS	110	100	0.852	12	0	8	≈
ANDE-2 versus SDENS	185	25	0.003	17	0	3	+
ANDE-2 versus jDElsgo	194	16	0.001	18	0	2	+
ANDE-2 versus DECC-DML	157	53	0.052	15	0	5	≈
ANDE-2 versus DASA	45	165	0.025	6	0	14	−
ANDE-2 versus EOEA	37	173	0.011	3	0	17	−
ANDE-2 versus MA-SW-CHAINS	31	179	0.006	4	0	16	−
ANDE-2 versus DMS-PSO-SHS	105	105	1	13	0	7	≈

Thus, our proposed algorithms can perform well and achieve good results within limited number of function evaluations which is very important issue when dealing with real-world problems.

(iv) ANDE and its two versions got very close to the optimum of single-group m-nonseparable multimodal functions F_6 in all statistical results with $1.20E + 05$ FEs.

(v) ANDE-1, among all other algorithms, got very close to the optimum of single-group m-nonseparable multimodal functions F_8 in the best and median results with $3.0E + 06$ FEs.

(vi) The performance of ANDE and ANDE-1 is well in all types of problems indicating that it is less affected than the most of other algorithms by the characteristics of the problems.

Furthermore, compared to the complicated structures and number of methods and number of control parameters used in other algorithms such as EOEA that uses three EAs optimizers within CC framework plus estimation of distribution algorithm (EDA), we can see that our proposed three ANDEs are very simple and easy to be implemented and programmed in many programming languages. They only use very simple self-adaptive crossover rate with two parameters and a free parameters novel triangular mutation rule and basic mutation. Thus, they neither increase the complexity of the original DE algorithm nor increase the number of control parameters. In order to investigate and compare the performance of the proposed algorithms ANDE, ANDE-1, and ANDE-2 against other algorithms in statistical sense, multiproblem Wilcoxon signed-rank test at a significance level 0.05 is performed on mean errors of all problems (with $1.25E + 05$ FEs, $6.00E + 05$ FEs, and $3.00E + 06$ FEs) and the results are presented in Tables 4, 5, and 6, respectively, where R^+ is the sum of ranks for the functions in which first algorithm outperforms the second algorithm in the row, and R^- is the sum of ranks for the opposite.

From Table 4, it can be obviously seen that ANDE is significantly better than SDENS, jDElsgo, and DECC-DML algorithms; ANDE is significantly worse than DASA, EOEA, and MA-SW-CHAINS algorithms. Besides, there is no significant difference between DMS-PSO-SHS and ANDE. Besides, it can be obviously seen that ANDE-1 is significantly better than SDENS and jDElsgo algorithms; ANDE-1 is significantly worse than DASA and MA-SW-CHAINS algorithms. However, there is no significant difference between DMS-PSO-HA, DECC-DML, EOEA, and ANDE-1. Regarding ANDE-2, it is significantly better than SDENS and jDElsgo algorithms; ANDE-2 is significantly worse than DASA, EOEA, and MA-SW-CHAINS algorithms. Besides, there is no significant difference between ANDE-2 and DMS-PSO-SHS and DECC-DML.

On the other hand, from Table 5, it can be obviously seen that ANDE is significantly better than SDENS, jDElsgo, and DECC-DML algorithms; ANDE is significantly worse than MA-SW-CHAINS algorithms. Besides, there is no significant difference between DMS-PSO-HA, DASA, EOEA, and ANDE. Besides, ANDE-1 is significantly better than

Table 5: Results of multiple-problem Wilcoxon's test for ANDE, ANDE-1, and ANDE-2 versus SDENS, jDElsgo, DECC-DML, DASA, EOEA, MA-SW-CHAINS, and DMS-PSO-SHS over all functions at a significance level 0.05 (with $6.00E + 05$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus SDENS	210	0	0.000	20	0	0	+
ANDE versus jDElsgo	192	18	0.001	18	0	2	+
ANDE versus DECC-DML	178	32	0.006	15	0	5	+
ANDE versus DASA	77	133	0.296	8	0	12	≈
ANDE versus EOEA	67	143	0.156	6	0	14	≈
ANDE versus MA-SW-CHAINS	38	172	0.012	1	0	15	+
ANDE versus DMS-PSO-SHS	78	132	0.313	8	0	12	≈
ANDE-1 versus SDENS	210	0	0.000	20	0	0	+
ANDE-1 versus jDElsgo	172	38	0.012	15	0	5	+
ANDE-1 versus DECC-DML	162	48	0.033	12	0	8	+
ANDE-1 versus DASA	87	123	0.502	8	0	12	≈
ANDE-1 versus EOEA	61	149	0.100	4	0	16	≈
ANDE-1 versus MA-SW-CHAINS	41	169	0.017	4	0	16	−
ANDE-1 versus DMS-PSO-SHS	93	117	0.654	9	0	11	≈
ANDE-2 versus SDENS	187	23	0.002	18	0	2	+
ANDE-2 versus jDElsgo	154	56	0.067	15	0	5	≈
ANDE-2 versus DECC-DML	154	56	0.067	14	0	6	≈
ANDE-2 versus DASA	60	150	0.093	7	0	13	≈
ANDE-2 versus EOEA	52	158	0.048	5	0	15	−
ANDE-2 versus MA-SW-CHAINS	43	167	0.021	6	0	14	−
ANDE-2 versus DMS-PSO-SHS	61	149	0.100	7	0	13	≈

Table 6: Results of multiple-problem Wilcoxon's test for ANDE, ANDE-1, and ANDE-2 versus SDENS, jDElsgo, DECC-DML, DASA, EOEA, MA-SW-CHAINS, and DMS-PSO-SHS over all functions at a significance level 0.05 (with $3.00E + 06$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus SDENS	187	23	0.002	17	0	3	+
ANDE versus jDElsgo	31	179	0.006	4	0	16	−
ANDE versus DECC-DML	185	25	0.003	16	0	4	+
ANDE versus DASA	103	107	0.940	12	0	8	≈
ANDE versus EOEA	88	122	0.526	9	0	11	≈
ANDE versus MA-SW-CHAINS	66	144	0.145	8	0	12	≈
ANDE versus DMS-PSO-SHS	54	156	0.057	7	0	13	≈
ANDE-1 versus SDENS	187	23	0.002	17	0	3	+
ANDE-1 versus jDElsgo	50	160	0.040	5	0	15	−
ANDE-1 versus DECC-DML	164	26	0.005	13	1	6	+
ANDE-1 versus DASA	122	88	0.526	12	0	8	≈
ANDE-1 versus EOEA	100	110	0.852	9	0	11	≈
ANDE-1 versus MA-SW-CHAINS	85	125	0.455	9	0	11	≈
ANDE-1 versus DMS-PSO-SHS	87	123	0.502	8	0	12	≈
ANDE-2 versus SDENS	155	55	0.062	15	0	5	≈
ANDE-2 versus jDElsgo	19	191	0.001	4	0	16	−
ANDE-2 versus DECC-DML	131	79	0.332	10	0	10	≈
ANDE-2 versus DASA	64	146	0.126	8	0	12	≈
ANDE-2 versus EOEA	41	169	0.017	5	0	15	−
ANDE-2 versus MA-SW-CHAINS	40	170	0.015	5	0	15	−
ANDE-2 versus DMS-PSO-SHS	21	189	0.002	16	0	4	−

SDENS, DECC-DML, and jDElsgo algorithms; ANDE-1 is significantly worse than MA-SW-CHAINS algorithms. Besides, there is no significant difference between DASA, EOEA, DMS-PSO-SHS, and ANDE-1. Besides, ANDE-2 is significantly better than SDENS algorithm; ANDE-2 is significantly worse than EOEA and MA-SW-CHAINS algorithms. Besides, there is no significant difference between jDElsgo, DECC-DML, DASA, DMS-PSO-SHS, and ANDE-2.

Finally, from Table 6, it can be obviously seen that HDE is significantly better than SDENS and DECC-DML algorithms; HDE is significantly worse than jDElsgo algorithm. Besides, there is no significant difference between DMS-PSO-SHS, DASA, EOEA, MA-SW-CHAINS, and HDE.

Additionally, it can be obviously seen that HDE-1 is significantly better than SDENS and DECC-DML algorithms; HDE-1 is significantly worse than jDElsgo algorithm. Besides, there is no significant difference between DMS-PSO-SHS, DASA, EOEA, MA-SW-CHAINS, and ANDE-1. However, it can be obviously seen that ANDE-2 is significantly worse than jDElsgo, EOEA, MA-SW-CHAINS, and DMS-PSO-SHS algorithms. Besides, there is no significant difference between SDENS, DECC-DML, DASA, and ANDE-2.

From Tables 4 and 5, it is noteworthy that ANDE-2 is better than all DEs algorithms (SDENS, jDElsgo, and DECC-DML) which indicate that the proposed self-adaptive crossover strategy improves the performance of the exploration capability of the basic DE in the earlier search stages. However, from Table 6, the poor performance of ANDE-2 algorithm in the remaining function evaluations is due to the fact that it has great global exploration ability with weak local exploitation tendency leading to inability of finding promising subregions that may deteriorate the overall performance as well as cause stagnation with almost benchmark problems. However, it is still statistically equal to SDENS, DECC-DML, and DASA algorithms. Therefore, taking into consideration the results obtained by a very simple version ANDE-2 plus remarkable performance of ANDE and ANDE-1 in the majority of the problems, it can be obviously concluded from direct and statistical results that ANDE and its versions are powerful algorithms and they are competitive with, and in some cases superior to, other existing algorithms in terms of the quality, efficiency, and robustness of the final solution.

6. A Parametric Study on ANDE

In this section, as the performance of ANDE and ANDE-1 is almost similar but ANDE converged to better solutions faster than ANDE-1 in many problems and ANDE explicitly includes both ANDE-1 and ANDE-2 versions, some experiments are carried out so as to identify the key parameters that significantly lead to the perfect performance of our proposed approach by studying the effects of the self-adaptive crossover rate scheme and the learning period as well as the maximum failure counter on the performance of ANDE algorithm. Note that in this study we do not try to find the optimal values for these parameters but to verify that the improved performance obtained after combining the proposed self-adaptive crossover rate into basic and proposed mutation schemes.

6.1. Efficacy of Parameter Adaptation and Setting. In the following three subsections, we investigate the effectiveness of the self-adaptive crossover scheme and we experimentally determine the appropriate values of the learning period and maximum failure counter parameters. Since they are not associated with the hybridization process and they have approximately the same effect with different number of function evaluations (FEs), the solution errors of all variants of ANDE are only recorded in $6.0E+05$ function evaluations (FEs) as a median value of number of function evaluations; all experiments for each function run 25 times independently and statistical results are provided including the best, median, mean, and worst results and the standard deviation. Besides, since the adaptation scheme of crossover is not used with problems F_1–F_3, they are excluded from comparison.

6.1.1. Effectiveness of the Crossover Adaptation Procedure. In this subsection, some trials have been performed to evaluate the benefits and effectiveness of the proposed crossover rate adaptation scheme on the performance on the ANDE algorithm. Since all problems with exception to F_1–F_3 are partially or fully separable problems in nature, it is preferred to evaluate HDE algorithm with three different values of CR such as 0.1, 0.5, and 0.9. The following versions were implemented for comparisons:

(1) $\text{ANDE}_{\text{CR}=0.1}$, which is the same as ANDE except that the crossover rate CR is set to a constant number 0.1

(2) $\text{ANDE}_{\text{CR}=0.5}$, which is the same as ANDE except that the crossover rate CR is set to a constant number 0.5

(3) $\text{ANDE}_{\text{CR}=0.9}$, which is the same as ANDE except that the crossover rate CR is set to a constant number 0.9

On the other hand, in order to show the efficacy of the construction of the learning period of the self-adaptive crossover rate, Procedure 1 has been modified such that it uses only one list that contains all possible values of CR from the beginning of the learning period (LP) and, hence, all the values have an equally likely chance of occurrence instead of using it in the second half of the learning period (LP). The procedure with the proposed modification is explained in Procedure 2.

The new version of ANDE algorithm is called ANDE with one fixed list of CR values during the LP and it is abbreviated (ANDE-OFL).

The statistical results including the best, median, mean, and worst results and the standard deviation over 25 independent runs of these algorithms are summarized in Table S7.

It can be obviously seen from Table 7 that ANDE is significantly better than $\text{ANDE}_{\text{CR}=0.5}$, $\text{ANDE}_{\text{CR}=0.9}$, and ANDE-OFL algorithms. Besides, there is no significant difference between $\text{ANDE}_{\text{CR}=0.1}$ and ANDE. However, although no significant difference exists between ANDE and $\text{ANDE}_{\text{CR}=0.1}$, we can still say that ANDE performs better than $\text{ANDE}_{\text{CR}=0.1}$ because the R^+ value of ANDE is much larger than the R^- one. Moreover, ANDE outperforms $\text{ANDE}_{\text{CR}=0.1}$ in 10 problems out of 17 problems while losing to $\text{ANDE}_{\text{CR}=0.1}$ in 7 problems which are F_6, F_8, F_{11}, F_{13}, F_{16}, F_{18}, and F_{20} (Ackley and

If ((CR_Flag_List[i] = 0) and (G <= LP)), If the target vector is better than the trial vector during the learning period, then:
Cr_i = Randomly select one value from A, $A = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95]$, $0 \le G \le$ LP
Else If ((CR_Flag_List[i] = 0) and (G > LP)), If the target vector is better than the trial vector after the learning period, then:
If the failure_counter_list[i] = Max_failure_counter,
Randomly select one value from list A, $A = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95]$.
else
failure_counter_list[i]= *failure_counter_list*[i] + 1;
End If
Else ((CR_Flag_List[i] = 1)), If the trial vector is better than the target vector through generations, then:
Select the value from A with maximum CR ratio from CR_Ratio_List[k].
End IF

Procedure 2

Table 7: Results of the multiple-problem Wilcoxon's test for ANDE, $ANDE_{CR=0.1}$, $ANDE_{CR=0.5}$, $ANDE_{CR=0.9}$, and ANDE-OFL over all functions at a significance level 0.05 (with $6.00E+05$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus $ANDE_{CR=0.1}$	117	36	0.055	10	0	7	≈
ANDE versus $ANDE_{CR=0.5}$	144	9	0.001	15	0	2	+
ANDE versus $ANDE_{CR=0.9}$	146	7	0.001	16	0	1	+
ANDE versus ANDE-OFL	145	8	0.001	16	0	1	+

Rosenbrock) with all types of modality and nonseparability. This proves that the proposed triangular mutation with small value of CR performs well on some unimodal, multimodal, and nonseparable problems. Regarding the remaining problems, ANDE can perform well with the unknown optimal CR value that is appropriate for these types of problems. Additionally, it also shows the effectiveness of our proposed novel idea about the gradual increase of the crossover rate during the initial stage of the search process. In fact, the main idea behind this comparison is to prove that the proposed ANDE algorithm can achieve the same results on some of the problems but when CR is tuned manually. Therefore, it proves that the proposed mutation scheme has well exploration and exploitation abilities with associated appropriate CR values for each problem. Besides, it also proves that the proposed self-adaptive scheme of CR plays a vital role in determining the optimal CR values for all problems during the optimization process. Therefore, the statistical analysis on the results in Table 7 confirms that ANDE with the proposed self-adaptive crossover scheme outperforms the other compared ANDE versions with constant CR value and with one fixed list of CR values.

6.1.2. Parametric Study on the Learning Period Parameter. The crossover (CR) practically controls the diversity of the population and it is directly affected by the value of the learning period parameter. In this subsection, some trials have been performed to determine the suitable values of learning period (LP). Thus, the performance of ANDE has been investigated under two different LP values (5% GEN and 20% GEN) and the results have been compared to those obtained by the value of 10% GEN. To analyze the sensitivity of this parameter, we further tested two extra configurations:

(1) $ANDE_{LP=5\%}$, which is the same as ANDE except that the leaning period was set to LP = 5% GEN

(2) $ANDE_{LP=20\%}$, which is the same as ANDE except that the leaning period was set to LP = 20% GEN

The statistical results including the best, median, mean, and worst results and the standard deviation over 25 independent runs of these algorithms are summarized in Table S8.

It can be obviously seen from Table 8 that ANDE is significantly better than $ANDE_{LP=20\%}$ algorithm. Besides, there is no significant difference between $ANDE_{LP=5\%}$ and ANDE. However, Table 8 shows that ANDE obtains higher R^+ value, which means that ANDE is overall better than $ANDE_{LP=5\%}$. Moreover, from Table 8, ANDE outperforms $ANDE_{LP=5\%}$ and $ANDE_{LP=20\%}$ in 13 problems out of 17 problems while losing to $ANDE_{LP=5\%}$ and $ANDE_{LP=20\%}$ in 4 problems which indicate that the performance of ANDE with the value of LP 10% is better than the performance of ANDE with the other two learning periods. However, this is not meaning that the LP of 10% is suitable for all problems with all function evaluations. Indeed, by a closer look at Tables 2 and 3, it can be seen that the solution of F_6 obtained by the three algorithms using $6.00E+5$ FEs is better than the solution achieved using $3.00E+06$. The main reason of this case can be obviously deduced from the convergence performance for F_6 presented in Figure S1. It shows very fast convergence of all algorithms in the first $2.0E+05$ evaluations which is considered to be premature convergence, meaning that, in the rest of the given time interval, the algorithms are slowly moving in the neighborhood of a local optimum. This is due to the delay in finding the suitable crossover value during the learning period which is 6000 generations. Practically, we applied the same learning period of $6.00E+05$ FEs which is

Table 8: Results of the multiple-problem Wilcoxon's test for ANDE, $HDE_{LP=5\%}$, and $HDE_{LP=20\%}$ over all functions at a significance level 0.05 (with $6.00E + 05$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus $HDE_{LP=5\%}$	103.5	49.5	0.201	13	0	4	≈
ANDE versus $HDE_{LP=20\%}$	126	27	0.019	13	0	4	+

Table 9: Results of the multiple-problem Wilcoxon's test for ANDE, $ANDE_{MFC=0}$, $ANDE_{MFC=10}$, and $ANDE_{MFC=30}$ over all functions at a significance level 0.05 (with $6.00E + 05$ FEs).

Algorithm	R^+	R^-	p value	Better	Equal	Worse	Dec.
ANDE versus $ANDE_{MFC=0}$	123	30	0.028	13	0	4	+
ANDE versus $ANDE_{MFC=10}$	81	55	0.501	9	1	7	≈
ANDE versus $ANDE_{MFC=30}$	127	26	0.017	13	0	4	+

1200 generations with $3.0E + 06$ FEs; the same results were achieved and another premature convergence or stagnation occurred again in another point as the $6.0E + 05$ case. The new best, median, worst, mean, standard deviation results obtained by ANDE were $2.44E + 00$, $3.07E + 00$, $7.27E + 00$, $4.19E + 00$, and $1.82E + 00$. The new convergence figure of the new median run compared to ANDE-1 and ANDE-2 is presented in Figure S2. Therefore, the learning period plays a vital role in optimization process and it is surely varied from one problem to another. Really, future research will focus on how to control the learning period. Overall, the learning period is suitable for all other problems as explained.

6.1.3. Parametric Study on the Maximum Failure Counter. In this subsection, some trials have been performed to determine the appropriate values of maximum failure counter (MFC). The crossover (CR) practically controls the diversity of the population and it is directly affected by the value of the maximum failure counter parameter. Thus, the performance of ANDE has been investigated under three different maximum failure counter values (0, 10, and 30) and the results have been compared to those obtained by the value of 10. To analyze the sensitivity of this parameter, we further tested three extra configurations:

(1) $ANDE_{MFC=0}$, which is the same as ANDE except that the maximum failure counter was set to MFC = 0

(2) $ANDE_{MFC=10}$, which is the same as ANDE except that the maximum failure counter was set to MFC = 10

(3) $ANDE_{MFC=30}$, which is the same as ANDE except that the maximum failure counter was set to MFC = 30

The statistical results including the best, median, mean, and worst results and the standard deviation over 25 independent runs of these algorithms are summarized in Table S9.

It can be obviously seen from Table 9 that ANDE is significantly better than $ANDE_{MFC=0}$ algorithm which shows the vital role and benefits of introducing (MFC) in the proposed self-adaptive CR scheme. Besides, there is no significant difference between $ANDE_{MFC=10}$ and ANDE. Moreover, ANDE outperforms $ANDE_{MFC=10}$ in 9 problems out of 17 problems while it loses to $ANDE_{MFC=10}$ in 7 problems and ties in 1 problem indicating that the performance of ANDE with the value of MFC = 10 is almost similar to the performance of ANDE with MFC = 20. However, ANDE is significantly better than $ANDE_{MFC=30}$ deducing that the increasing of MFC value has negative effect that could lead to significant deterioration in the performance of ANDE algorithm. From the above statistical analysis, it can be concluded that ANDE with self-adaptive crossover rate and with learning period LP of 5% or 10% and maximum failure counter (MFC) of 10–20 generations performs well in the majority of the problems.

7. Conclusion

In order to efficiently concentrate the exploitation tendency of some subregion of the search space and to significantly promote the exploration capability in whole search space during the evolutionary process of the conventional DE algorithm, a Differential Evolution (ANDE) algorithm for solving large scale global numerical optimization problems over continuous space was presented in this paper. The proposed algorithm introduced a new triangular mutation rule based on the convex combination vector of the triplet defined by the three randomly chosen vectors and the difference vectors between the best, better, and the worst individuals among the three randomly selected vectors. The mutation rule is combined with the basic mutation strategy DE/rand/1/bin, where only one of the two mutation rules is applied with the probability of 2/3 since it has both exploration ability and exploitation tendency. Furthermore, we propose a novel self-adaptive scheme for gradual change of the values of the crossover rate that can excellently benefit from the past experience of the individuals in the search space during evolution process which in turn can considerably balance the common trade-off between the population diversity and convergence speed. The proposed mutation rule was shown to enhance the global and local search capabilities of the basic DE and to increase the convergence speed. The algorithm has been evaluated on the standard high-dimensional benchmark problems. The comparison results between ANDE and its versions and the other seven state-of-the-art evolutionary algorithms that were all tested on this test suite on the IEEE congress on evolutionary competition in 2010 indicate that the proposed algorithm and its two versions are highly competitive algorithms for solving large scale global optimization problem. The experimental

results and comparisons showed that the ANDE algorithm performed better in large scale global optimization problems with different types and complexity; it performed better with regard to the search process efficiency, the final solution quality, the convergence rate, and robustness, when compared with other algorithms. Finally, the performance of the ANDE algorithm was statistically superior to and competitive with other recent and well-known DEs and non-DEs algorithms. The effectiveness and benefits of the proposed modifications used in ANDE were experimentally investigated and compared. It was found that the two proposed algorithms ANDE and ANDE-1 are competitive in terms of quality of solution, efficiency, convergence rate, and robustness. They were statistically superior to the state-of-the-art DEs and non-DEs algorithms. Besides, they perform better than ANDE-2 algorithm in many cases. Although the remarkable performance of ANDE-2 was competitive with some of the compared algorithms, several current and future works can be developed from this study. Firstly, current research effort focuses on how to control the scaling factors by self-adaptive mechanism and develop another self-adaptive mechanism for crossover rate. Additionally, the new version of ANDE combined with Cooperative Coevolution (CC) framework is being developed and will be experimentally investigated soon. Moreover, future research will investigate the performance of the ANDE algorithm in solving constrained and multiobjective optimization problems as well as real-world applications such as data mining and clustering problems. In addition, large scale combinatorial optimization problems will be taken into consideration. Another possible direction is integrating the proposed triangular mutation scheme with all compared and other self-adaptive DE variants plus combining the proposed self-adaptive crossover with other DE mutation schemes. Additionally, the promising research direction is joining the proposed triangular mutation with evolutionary algorithms, such as genetic algorithms, harmony search, and particle swarm optimization, as well as foraging algorithms such as artificial bee colony, bees algorithm, and Ant-Colony Optimization.

Competing Interests

The authors declare that they have no competing interests.

References

[1] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC 2010 special session and competition on large scale global optimization," Tech. Rep., Nature Inspired Computation and Applications Laboratory, USTC, Hefei, China, 2009.

[2] K. Tang, X. Yao, P. N. Suganthan et al., "Benchmark functions for the CEC2008 special session and competition on large scale global optimization," Technical Report for CEC 2008 special issue, 2007.

[3] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, Wiley-Blackwell, 2002.

[4] 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.

[5] S. A. El-Quliti, A. H. Ragab, R. Abdelaal et al., "A nonlinear goal programming model for university admission capacity planning with modified differential evolution algorithm," *Mathematical Problems in Engineering*, vol. 2015, Article ID 892937, 13 pages, 2015.

[6] S. A. El-Qulity and A. W. Mohamed, "A generalized national planning approach for admission capacity in higher education: a nonlinear integer goal programming model with a novel differential evolution algorithm," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 5207362, 14 pages, 2016.

[7] N. Hachicha, B. Jarboui, and P. Siarry, "A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics," *Information Sciences*, vol. 181, no. 1, pp. 79–91, 2011.

[8] S. A. El-Quliti and A. W. Mohamed, "A large-scale nonlinear mixed-binary goal programming model to assess candidate locations for solar energy stations: an improved binary differential evolution algorithm with a case study," *Journal of Computational and Theoretical Nanoscience*, vol. 13, no. 11, pp. 7909–7921, 2016.

[9] 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.

[10] 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.

[11] 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.

[12] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proceedings of the Mendel 6th International Conference on Soft Computing*, pp. 76–83, Brno, Czech Republic, June 2000.

[13] A. W. Mohamed and H. Z. Sabry, "Constrained optimization based on modified differential evolution algorithm," *Information Sciences*, vol. 194, pp. 171–208, 2012.

[14] A. W. Mohamed, H. Z. Sabry, and M. Khorshid, "An alternative differential evolution algorithm for global optimization," *Journal of Advanced Research*, vol. 3, no. 2, pp. 149–165, 2012.

[15] 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.

[16] A. W. Mohamed, "Solving stochastic programming problems using new approach to differential evolution algorithm," *Egyptian Informatics Journal*, 2016.

[17] 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*, vol. 11, no. 2, pp. 1679–1696, 2011.

[18] A. W. Mohamed, "An efficient modified differential evolution algorithm for solving constrained non-linear integer and mixed-integer global optimization problems," *International Journal of Machine Learning and Cybernetics*, 2015.

[19] A. W. Mohamed, "A new modified binary differential evolution algorithm and its applications," *Applied Mathematics & Information Sciences*, vol. 10, no. 5, pp. 1965–1969, 2016.

[20] A. Wagdy Mohamed, H. Z. Sabry, and A. Farhat, "Advanced differential evolution algorithm for global numerical optimization," in *Proceedings of the IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE '11)*, pp. 156–161, Penang, Malaysia, December 2011.

[21] A. W. Mohamed, H. Z. Sabry, and T. Abd-Elaziz, "Real parameter optimization by an effective differential evolution algorithm," *Egyptian Informatics Journal*, vol. 14, no. 1, pp. 37–53, 2013.

[22] A. W. Mohamed, "A novel differential evolution algorithm for solving constrained engineering optimization problems," *Journal of Intelligent Manufacturing*, pp. 1–34, 2017.

[23] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[24] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer, New York, NY, USA, 1st edition, 2005.

[25] A. M. Potter and K. A. D. Jong, "A cooperative co-evolutionary approach to function optimization," in *Proceedings of the 3rd International Conference on Parallel Problem Solving from the Nature*, pp. 249–257, Springer, October 1994.

[26] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative co-evolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[27] M. N. Omidvar, X. D. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '10)*, pp. 1762–1769, 2010.

[28] Y. Wang and B. Li, "Two-stage based ensemble optimization for large-scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '10)*, pp. 4488–4495, Barcelona, Spain, July 2010.

[29] Y. Wang and B. Li, "A self-adaptive mixed distribution based uni-variate estimation of distribution algorithm for large scale global optimization," in *Nature-Inspired Algorithms for Optimisation*, R. Chiong, Ed., vol. 193 of *Studies in Computational Intelligence*, pp. 171–198, Springer, Berlin, Germany, 2009.

[30] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary porgramming with cooperative co-evolution," in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 1101–1108, 2001.

[31] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 3523–3530, IEEE, Singapore, September 2007.

[32] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 1663–1670, June 2008.

[33] A. Zamuda, J. Brest, B. Bošović, and V. Žumer, "Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, June 2008.

[34] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[35] K. Korošec, K. Tashkova, and J. Šilc, "The differential ant-stigmergy algorithm for large-scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 4288–4295, Barcelona, Spain, 2010.

[36] J. Brest, A. Zamuda, I. Fister, and M. S. Maučec, "Large scale global optimization using self-adaptive differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '10)*, pp. 3097–3104, July 2010.

[37] H. Wang, Z. Wu, S. Rahnamayan, D. Jiang, and D. E. Sequential, "Enhanced by neighborhood search for large scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 4056–4062, Barcelona, Spain, July 2010.

[38] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization," in *Proceedings of the 6th IEEE World Congress on Computational Intelligence (WCCI '10)—IEEE Congress on Evolutionary Computation (CEC '10)*, esp, July 2010.

[39] S.-Z. Zhao, P. N. Suganthan, and S. Das, "Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search," in *Proceedings of the 6th IEEE World Congress on Computational Intelligence (WCCI '10)—IEEE Congress on Evolutionary Computation (CEC '10)*, July 2010.

[40] 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.

[41] V. Feoktistov, *Differential Evolution, in Search of Solutions*, vol. 5, Springer, New York, NY, USA, 2006.

[42] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real parameter optimization with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 506–513, Edinburgh, UK, 2005.

[43] 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.

[44] M. Weber, F. Neri, and V. Tirronen, "A study on scale factor in distributed differential evolution," *Information Sciences*, vol. 181, no. 12, pp. 2488–2511, 2011.

[45] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 689–707, 2014.

[46] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An improved self-adaptive differential evolution algorithm for optimization problems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 89–99, 2013.

[47] A. W. Mohamed, "An improved differential evolution algorithm with triangular mutation for global numerical optimization," *Computers and Industrial Engineering*, vol. 85, pp. 359–375, 2015.

[48] S. Chen, "Locust Swarms for Large Scale Global Optimization for Nonseparable Problems".

[49] 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.