Soft Comput
DOI 10.1007/s00500-017-2777-2

FOUNDATIONS

Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation

Ali Wagdy Mohamed[1] · **Ponnuthurai Nagaratnam Suganthan**[2]

Abstract This paper presents enhanced fitness-adaptive differential evolution algorithm with novel mutation (EFADE) for solving global numerical optimization problems over continuous space. A new triangular mutation operator is introduced. It is 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. Triangular mutation operator helps the search for better balance between the global exploration ability and the local exploitation tendency as well as enhancing the convergence rate of the algorithm through the optimization process. Besides, two novel, effective adaptation schemes are used to update the control parameters to appropriate values without either extra parameters or prior knowledge of the characteristics of the optimization problem. In order to verify and analyze the performance of EFADE, numerical experiments on a set of 28 test problems from the CEC2013 benchmark for 10, 30 and 50 dimensions, including a comparison with 12 recent DE-based algorithms and six recent evolutionary algorithms, are executed. Experimental results indicate that in terms of robustness, stability and quality of the solution obtained, EFADE is significantly better than, or at least comparable to state-of-the-art approaches with outstanding performance.

Keywords Evolutionary computation · Global optimization · Differential evolution · Novel triangular mutation · Adaptive parameter control

Communicated by A. Di Nola.

Electronic supplementary material The online version of this article (doi:10.1007/s00500-017-2777-2) contains supplementary material, which is available to authorized users.

✉ Ali Wagdy Mohamed
aliwagdy@gmail.com

Ponnuthurai Nagaratnam Suganthan
epsugan@ntu.edu.sg

1 Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza 12613, Egypt

2 School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore

1 Introduction

Differential evolution (DE), proposed by Storn and Price (1995, 1997), is a stochastic population-based search method. It exhibits excellent capability in solving a wide range of optimization problems with different characteristics from several fields and many real-world application problems (Price et al. 2005). Similar to all other evolutionary algorithms (EAs), the evolutionary process of DE uses mutations, crossover and selection operators at each generation to search for the global optimum. Besides, it is one of the most efficient evolutionary algorithms (EAs) currently in use. The performance of DE basically depends on the mutation strategy, the crossover operator. Besides, the intrinsic control parameters (population size NP, scaling factor F, the crossover rate CR) play a vital role in balancing the diversity of population and convergence speed of the algorithm. The advantages are simplicity of implementation, reliability, speed and robustness (Price et al. 2005). Thus, it has been widely applied in solving many real-world applications of science and engineering, such as classification (Zhai and Jiang 2015), image processing (Sarkar et al. 2015), feature selection (Paul and Das 2015), admission capacity planning higher education (El-Quliti et al. 2015; El-Qulity and Mohamed 2016), and solar energy (El-Quliti and Mohamed 2016). For more applications, interested readers can refer to Das et al. (2016) for more details. 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 (Noman and Iba 2008). Additionally, the parameters of DE are problem dependent and it is difficult to adjust them to different problems. Moreover, DE performance decreases as search space dimensionality increases (Das et al. 2009). Finally, the performance of DE deteriorates significantly when the problems of premature convergence and/or stagnation occur (Das et al. 2009; Lampinen and Zelinka 2000). Consequently, researchers have suggested many techniques to improve the basic DE. From the literature (Das and Suganthan 2011), these proposed modifications, improvements and developments on DE focus on adjusting control parameters in an adaptive or self-adaptive manner while there are a few attempts in proposing new mutations rule. In this paper, in order to overcome these drawbacks of DE, two evolutionary process components are proposed. In fact, one or both can be combined with the DE family of algorithms to enhance their search capabilities on difficult and complicated optimization problems.

(1) First, to enhance global and local search capabilities and simultaneously increase the convergence speed of DE, a new triangular mutation operator is introduced.
(2) Second, two simple adaptive schemes without extra parameters to update the values of scaling factors and crossover in each generation are proposed. They guided the search process by the previous and current knowledge of their successful ratios that produced better trial vectors in the last generation.

EFADE has been tested on 28 benchmark test functions developed for the 2013 IEEE Congress on Evolutionary Computation (IEEE CEC2013) (Liang et al. 2013). Extensive numerical experiments and comparisons indicate that the proposed (EFADE) algorithm is superior and competitive with the 12 DE-based algorithms and 6 recent evolutionary algorithms particularly in the case of high-dimensional complex and composition optimization problems. The rest of the paper is organized as follows: Sect. 2 gives a brief introduction to canonical DE algorithm, including its typical mutation operators, crossover and selection operators. Section 3 reviews the related work. Next, in Sect. 4, the proposed (EFADE) algorithm is introduced. In Sect. 5, computational results of testing benchmark functions and on the comparison with other techniques are reported. Section 6 discusses the effectiveness of the proposed modifications. Finally, conclusions and future works are drawn in Sect. 7.

2 Differential evolution

This section provides a brief summary of the basic differential evolution (DE) algorithm. In simple DE, generally known as DE/rand/1/bin (Storn and Price 1997; Fan and Lampinen 2003), an initial random population, denoted by P, consists of NP individual. Each individual is represented by the vector $x_i = (x_{1,i}, x_{2,i}, \ldots, x_{D,i})$, where D is the number of dimensions in solution space. Since the population will be varied with the running of evolutionary process, the generation times in DE are expressed by $G = 0, 1, \ldots, Gmax$, where $Gmax$ is the maximal times of generations. For the ith individual of P at the G generation, it is denoted by $x_i^G = (x_{1,i}^G, x_{2,i}^G, \ldots, x_{D,i}^G)$. The lower and upper bounds in each dimension of search space are, respectively, recorded by $x_{\mathrm{L}} = (x_{1,\mathrm{L}}, x_{2,\mathrm{L}}, \ldots, x_{D,\mathrm{L}})$ and $x_{\mathrm{U}} = (x_{1,\mathrm{U}}, x_{2,\mathrm{U}}, \ldots, x_{D,\mathrm{U}})$. The initial population P^0 is randomly generated according to a uniform distribution within the lower and upper boundaries $(x_{\mathrm{L}}, x_{\mathrm{U}})$. 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 (Das and Suganthan 2011; Mallipeddi et al. 2011). DE steps are discussed below:

2.1 Initialization

In order to establish a starting point for the optimization process, an initial population P^0 must be created. Typically, each jth component $(j = 1, 2, \ldots, D)$ of the ith individuals $(i = 1, 2, \ldots, \mathrm{NP})$ in the P^0 is obtained as follows:

$$x_{j,i}^0 = x_{j,L} + \mathrm{rand}(0,1) \cdot (x_{j,U} - x_{j,L}), \tag{1}$$

where rand (0,1) returns a uniformly distributed random number in [0, 1].

2.2 Mutation

At generation G, for each target vector x_i^G, a mutant vector v_i^G is generated according to the following:

$$v_i^G = x_{r_1}^G + F \cdot (x_{r_2}^G - x_{r_3}^G), r_1 \neq r_2 \neq r_3 \neq i. \tag{2}$$

With randomly chosen indices $r_1, r_2, r_3 \in \{1, 2, \ldots, \mathrm{NP}\}$.$F$ is a real number to control the amplification of the difference vector $(x_{r_2}^G - x_{r_3}^G)$. According to Storn and Price (1997), the range of F is in [0, 2]. In this work, If a component of a mutant vector violates search space, then the value of this component is generated a new using Eq. (1). The other most frequently used mutations strategies are

$$\text{“DE/best/1”[26]} : v_i^G = x_{\text{best}}^G + F \cdot (x_{r_1}^G - x_{r_2}^G), \tag{3}$$

$$\text{“DE/best/2”[26]} : v_i^G = x_{\text{best}}^G + F \cdot (x_{r_1}^G - x_{r_2}^G) + F \cdot (x_{r_3}^G - x_{r_4}^G), \tag{4}$$

$$\text{“DE/rand/2”[26]} : v_i^G = x_{r_1}^G + F \cdot (x_{r_2}^G - x_{r_3}^G) + F \cdot (x_{r_4}^G - x_{r_5}^G), \tag{5}$$

$$\text{“DE/current-to-best/1”[26]} : v_i^G = x_i^G + F \cdot (x_{\text{best}}^G - x_i^G) + F \cdot (x_{r_1}^G - x_{r_2}^G), \tag{6}$$

$$\text{“DE/current-to-rand/1”[26]} : v_i^G = x_i^G + F \cdot (x_{r_1}^G - x_i^G) + F \cdot (x_{r_2}^G - x_{r_3}^G). \tag{7}$$

The indices r_1, r_2, r_3, r_4, r_5 are mutual integers randomly generated within the range [1, NP], which are also different from the index i. These indices are randomly generated once for each mutant vector. The scale factor F is a positive control parameter for scaling the difference vector. x_{best}^G is the best individual vector with the best fitness value in the population at generation G.

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.

$$u_{j,i}^G = \begin{cases} v_{j,i}^G, & \text{if}\,(\text{rand}_{j,i} \leq \text{ CR or } j = j_{\text{rand}}) \\ x_{j,i}^G, & \text{otherwise,} \end{cases} \tag{8}$$

where $\text{rand}_{j,i}$ ($i \in [1, \text{NP}]$ and $j \in [1, D]$) is a uniformly distributed random number in [0,1], CR $\in$ [0, 1] called the crossover rate that controls how many components are inherited from the mutant vector, j_{rand} is a uniformly distributed random integer in $[1, D]$ that makes sure at least one component of trial vector is inherited from the mutant vector.

2.4 Selection

DE adapts a greedy selection strategy. If and only if the trial vector u_i^G yields as good as or a better fitness function value than x_i^G, then u_i^G 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):

$$x_i^{G+1} = \begin{cases} u_i^G, & f(u_i^G) \leq f(x_i^G) \\ x_i^G, & \text{otherwise.} \end{cases} \tag{9}$$

A detailed description of standard DE algorithm is shown in Fig. 1.

```
Begin
  G=0
  Create a random initial population $\vec{x}_i^G \ \forall i, i = 1,\ldots, NP$
  Evaluate $f(\vec{x}_i^G) \ \forall i, i = 1,\ldots, NP$
  For G=1 to Gmax Do
      For i=1 to NP Do
           Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$
           $j_{rand}$= randint(1,D)
         For j=1 to D Do
           If (rand$_{j,i}$[0,1]< CR or j= $j_{rand}$) Then
              $u_{i,j}^G = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$
           Else
              $u_{i,j}^G = x_{i,j}^G$
           End If
         End For
         If( $f(\vec{u}_i^G) \leq f(\vec{x}_i^G)$ )Then
              $\vec{x}_i^{G+1} = \vec{u}_i^G$
         Else
              $\vec{x}_i^{G+1} = \vec{x}_i^G$
         End If
      End For
    G=G+1
  End For
End
```

Fig. 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, Gmax, CR and F are user-defined parameters. D is the dimensionality of the problem

3 Related work

Virtually, the performance of the canonical DE algorithm mainly depends on the chosen mutation/crossover strategies and the associated control parameters. Moreover, due to DE limitations as aforementioned in Sect. 1, during the past 15 years, many researchers have been working on the improvement of DE. Thus, many researchers have proposed novel techniques to overcome its problems and improve its performance (Das and Suganthan 2011). A brief overview of these algorithms is proposed in this section. Firstly, many trial-and-error experiments have been conducted to adjust the control parameters. Storn and Price (1995) suggested that NP (population size) between 5D and 10D is 0.5, which is a good initial value of F (mutation scaling factor). The effective value of F usually lies in a range between 0.4 and 1. The CR (crossover rate) is an initial good choice of CR = 0.1; however, since a large CR often accelerates convergence, it is appropriate to first try CR as 0.9 or 1 in order to check if a quick solution is possible. Gämperle et al. (2002) recommended that a good choice for NP is between 3D and 8D, with F = 0.6, and CR lies in [0.3, 0.9]. However, Ronkkonen et al. (2005) concluded that $F = 0.9$ is a good compromise between the convergence speed and the convergence rate. Additionally, CR depends on the nature of the problem, so CR with a value between 0.9 and 1 is suitable for non-separable and multimodal objective functions, while a value of CR is between 0 and 0.2 when the objective function is separable. On the other hand, due to the apparent contradictions regarding the rules for determining the appropriate values of the control parameters from the literature, some techniques have been designed with a view of adjusting control parameters in adaptive or self-adaptive manner instead of manual tuning procedure. Wu et al. (2015) proposed a multipopulation-based framework to realize an adapted ensemble of three mutation strategies (i.e., “current-to-pbest/1” and “current-to-rand/1” and “rand/1”) into a novel DE variant, named MPEDE. The population of MPEDE is dynamically partitioned into several subpopulations including three indicator subpopulations and one reward subpopulation. Each indicator subpopulation with a relatively smaller size is assigned to a constituent mutation strategy, and the rewarded subpopulation with a relatively larger size is assigned to the currently best performed mutation strategy as an extra reward. This way, dynamically computational resource allocation among the mutation strategies is realized and the best mutation strategy is expected to obtain the most computational resources. To nonlinearly change both the values of F and CR along with their direction of changing with the progress of iterations, Draa et al. (2015) proposed six adaptation schemes based on sinusoidal functions. In these schemes, the F and CR values can be obtained from scaled and shifted sinusoidal terms. It is experimentally shown that both F and CR values are sampled from sinusoidal terms with increasing coefficients, which outperform other proposed schemes. Likewise, Brest et al. (2006) proposed an efficient technique, named jDE, for self-adapting control parameter settings. This technique encodes the parameters into each individual and adapts them by means of evolution. The results showed that jDE is better than, or at least comparable to, the standard DE algorithm, (FADE) algorithm and other state-of-the-art algorithms when considering the quality of the solutions obtained. In the same context, controlling the degree of randomization of F and CR and investigating the effect of the same on the performance of DE are important research directions in DE. A very interesting work in this direction was recently published by Zamuda and Brest (2015). Here, the authors introduced a randomness level parameter, which influences the dynamics of the control parameter values and their propagation through “suitable individuals’ improvement contributions during elitist selection” in an extended framework of the jDE algorithm. Qin et al. (2009) introduced a self-adaptive differential evolution (SaDE). The main idea of SaDE is to simultaneously implement two mutation schemes: “DE/rand/1/bin” and “DE/best/2/bin” as well as adapt mutation and crossover parameters. The performance of SaDE was evaluated on a suite of 26 several benchmark problems, and it was compared with the conventional DE and three adaptive DE variants. The experimental results demonstrated that SaDE yielded better quality solutions and had a higher success rate. In the same context, and as inspired by SaDE algorithm and motivated by the success of diverse self-adaptive DE approaches, Mallipeddi et al. (2011) developed a self-adaptive DE, called EPSDE, based on ensemble approach. In EPSDE, a pool of distinct mutation strategies along with a pool of values for each control parameter coexists throughout the evolution process and competes to produce offspring. The performance of EPSDE was evaluated on a set of bound constrained problems and was compared with conventional DE and other state-of-the-art parameter-adaptive DE variants. The resulting comparisons showed that EPSDE algorithm outperformed conventional DE and other state-of-the-art parameter-adaptive DE variants in terms of solution quality and robustness. Similarly, motivated by the important results obtained by other researchers during past years, Wang et al. (2011) proposed a novel method, called composite DE (CoDE). This method used three trial vector generation strategies and three control parameter settings. It randomly combines them to generate trial vectors. The performance of CoDE has been evaluated on 25 benchmark test functions developed for IEEE CEC2005, and it was very competitive to other compared algorithms. Recently, super-fit multicriteria adaptive DE (SMADE) is proposed by Caraffini et al. (2013b), which is a Memetic approach based on the hybridization of the covariance matrix adaptive evolutionary strategy (CMA-ES) with modified DE, namely multicriteria adaptive DE (MADE). At the beginning of the optimization process, CMA-ES is used to generate a solution with a high quality which is then injected into the population of MADE. The performance of SMADE has been evaluated on 28 benchmark test functions developed for IEEE CEC2013, and the experimental results are very promising. On the other hand, improving the trial vector generation strategy has attracted much research. In order to improve the convergence velocity of DE, Fan and Lampinen (2003) proposed a trigonometric mutation scheme, which is considered local search operator, and combined it with DE/rand/1 mutation operator to design TDE algorithm. Zhang and Sanderson (2009) 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. Along the same lines, to overcome the premature convergence and stagnation problems observed in the classical DE, Islam et al. (2012) proposed modified DE with p-best crossover, named MDE_pBX. The novel mutation strategy is similar to DE/current-to-best/1 scheme. It selects the best vector from a dynamic group of $q\%$ of the randomly selected population members. Moreover, their crossover strategy uses a vector that is randomly selected from the $p$top ranking vectors according to their objective values in the current population (instead of the target vector) to enhance the inclusion of generic information from the elite individuals in the current generation. The parameter p is linearly reduced with generations to favor the exploration at the beginning of the search and exploitation during the later stages by gradually downsizing the elitist portion of the population. Das et al. (2016) proposed two kinds of topological neighborhood models for DE in order to achieve better balance between its explorative and exploitative tendencies. In this method, called DEGL, two trial vectors are created by the global and local neighborhood-based mutation. These two trial vectors are combined to form the actual trial vector by using a weight factor. The performance of DEGL has been evaluated on 24 benchmark test functions and two real-world problems and showed very competitive results. In order to solve unconstrained and constrained optimization problems, Mohamed and Sabry (2012); Mohamed et al. (2011, 2012) propose a novel directed mutation based on the weighted difference vector between the best and the worst individuals at a particular generation. The new directed mutation rule is combined with the modified basic mutation strategy DE/rand/1/bin, where only one of the two mutation rules is applied with the probability of 0.5. The proposed mutation rule is shown to enhance the local search ability of the basic differential evolution (DE) and to get a better trade-off between convergence rate and robustness. Numerical experiments on well-known unconstrained and constrained benchmark test functions and five engineering design problems have shown that the new approach is efficient, effective and robust.

Practically, it can be observed that the main modifications, improvements and developments on DE focus on adjusting control parameters in an adaptive or self-adaptive manner. However, a few enhancements have been implemented to modify the structure and/or mechanism of basic DE algorithm or to propose new mutation rules so as to enhance the local and global search ability of DE and to overcome the problems of stagnation or premature convergence. It is worth noting that the proposed algorithm is different from an earlier DE. As reported in Mohamed (2015), there are significant differences as follows: (1) in previous work (Mohamed 2015), 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 of the selected vectors with corresponding three scaling factors, which are adaptively generated according to uniform distribution based on the fitness function values of selected vectors, are used in the mutation scheme. (2) In Mohamed (2015), the real weights of the convex combination vector are calculated using uniform distributions, but in this work, the real weights of the vectors are calculated according to a novel rule that is based on the position of the selected vector in the search space. (3) The crossover rate in Mohamed (2015) is given by a dynamic nonlinear increased probability scheme, but in this work, a novel adaptation scheme for CR is developed that is composed of two different uniform sets. These two sets are adaptively selected based on their experiences of generating promising solutions during the evolution process. (4) In previous work (Mohamed 2015), a restart mechanism based on random mutation and modified BGA mutation is used to avoid stagnation or premature convergence, whereas this work is not.

4 Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation (EFADE)

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

4.1 Triangular mutation scheme

DE/rand/1 is the fundamental mutation strategy developed by Storn and Price (1995, 1997) and is reported to be the most successful and widely used scheme in the literature (Das and Suganthan 2011). Obviously, in this strategy, the three vectors are chosen from the population at random for mutation and the base vector is then selected at random among the three. The other two vectors form the difference vector that is added to the base vector. Consequently, it is able to maintain population diversity and global search capability with no bias to any specific search direction, but it slows down the convergence speed of DE algorithms (Qin et al. 2009). DE/rand/2 strategy, like the former scheme with extra two vectors that forms another difference vector, might lead to better perturbation than one-difference-vector-based strategies (Qin et al. 2009). Furthermore, it can provide more various differential trial vectors than the DE/rand/1/bin strategy which increases its exploration ability of the search space. On the other hand, greedy strategies like DE/best/1, DE/best/2 and DE/current-to-best/1 incorporate the information of the best solution found so far in the evolutionary process to increase the local search tendency that leads to fast convergence speed of the algorithm. However, the diversity of the population and exploration capability of the algorithm may deteriorate or may be completely lost through a very small number of generations, i.e., at the beginning of the optimization process, that cause problems such stagnation and/or premature convergence. Consequently, in order to overcome the shortcomings of both types of mutation strategies, most of the recent successful algorithms utilize the strategy candidate pool that combines different trail vector generation strategies. These have diverse characteristics and distinct optimization capabilities, with different control parameter settings to be able to deal with a variety of problems with different features at different stages of evolution (Qin et al. 2009; Wang et al. 2011; Mallipeddi et al. 2011). Contrarily, taking into consideration the weakness of existing greedy strategies, Zhang and Sanderson (2009) introduced a new differential evolution (DE) algorithm, named JADE, to improve optimization performance by implementing a new mutation strategy "DE/current-to-*p*best" with optional external archive and updating control parameters in an adaptive manner. Consequently, proposing new mutations strategies that can considerably improve the search capability of DE algorithms and increase the possibility of achieving promising and successful results in complex and large-scale optimization problems is still an open challenge for the evolutionary computation research. Therefore, this research uses a new triangular mutation rule with a view of balancing the global exploration ability and the local exploitation tendency and enhancing 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 of the selected vectors. The proposed mutation vector is generated in the following manner:

$$v_i^{G+1} = \bar{x}_c^G + F1 \cdot \left(x_{\text{best}}^G - x_{\text{better}}^G\right) + F2 \cdot \left(x_{\text{best}}^G - x_{\text{worst}}^G\right) + F3 \cdot \left(x_{\text{better}}^G - x_{\text{worst}}^G\right), \tag{10}$$

where $\bar{x}_c^G$ is a convex combination vector of the triangle, $F1$, $F2$ and $F3$ are the mutation factors that are associated with x_i and are generated at each generation by the adaptation process introduced later on, and $x_{\text{best}}^G, x_{\text{better}}^G$ and x_{worst}^G are the tournament best, better and worst three randomly selected vectors, respectively. The convex combination vector $\bar{x}_c^G$ of the triangle is a linear combination of the three randomly selected vectors and is defined as follows:

$$\bar{x}_c^G = w_1^* \cdot x_{\text{best}} + w_2^* \cdot x_{\text{better}} + w_3^* \cdot x_{\text{worst}}, \tag{11}$$

where the real weights w_i^* satisfy $w_i^* \geq 0$ and $\sum_{i=1}^{3} w_i^* = 1$. The real weights w_i^* are given by $w_i^* = w_i / \sum_{i=1}^{3} w_i, i = 1, 2, 3$. Without loss of generality, we only consider minimization problems. The weights w_i are calculated according to the following equation:

$$w_i = 1 - \left| \frac{f(x_{\text{best}}) - f(x_i)}{f(x_{\text{best}}) - f(x_{\max})} \right|, \quad i = 1, 2, 3, \tag{12}$$

where $f(x_{\text{best}}) = f(x_{\min}) = \min\{f(x_i)\}, i = 1, 2, 3$, i.e., the best vector among the three randomly selected vectors, $f(x_{\max})$, is the maximum objective function values over the individuals of the population obtained in a generation. Generally, w_1, which is the weight of the best vector among the three selected vectors, equals 1, $w_2 \geq w_3$ and $0 < w_2 \leq 1, 0 \leq w_3 \leq 1$. Simply, it can be easily perceived that $w_3 = 0$ if $f(x_{\text{worst}}) = f(x_{\max})$ .Thus, to avoid zero weight in this case, $w_3 = \text{rand}(0, w_2)$. On the other hand, $w_2, w_3 = 1$ if $f(x_{\text{best}}) = f(x_{\text{better}})$ or $f(x_{\text{best}}) = f(x_{\text{worst}})$ with all values including zero, this is the case of convergence to global or suboptimal point or when it is stagnated in a specific point in the search domain . Thus, $w_1^* \geq w_2^*, w_1^* \geq w_3^*$ and $w_2^* \geq w_3^*$. From Eq. (12), it can be observed that the value of the weight of the better or worst vectors depends directly on its position in the search region. If it is relatively close to the best vector, its weight is near to 1, and if it is relatively close to the worst individual in the population, its weight relatively approaches to zero. Obviously, from mutation Eq. (10), 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 resemble the concept of gradient as the difference vectors are directed from the worst to the better to the best vectors (Feoktistov 2006). Thus, it is considerably used to explore the landscape of the objective function being optimized in different subregions around the best vectors within search space through optimization process. Secondly, the convex combination vector $\bar{x}_c^G$ is a weighted sum of the three randomly selected vectors where the best vector has the significant contribution. Therefore, $\bar{x}_c^G$ 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 and 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 $\bar{x}_c^G$ 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 global exploration capability and local exploitation tendency while the basic mutation favors exploration only, the probability of using the proposed mutation is much more than the probability of applying the basic rule. Therefore, the new mutation strategy is embedded into the DE algorithm and is combined with the basic mutation strategy DE/rand/1/bin through nonlinear probability rule as follows:

$$\begin{aligned}
&\text{If } \left(u(0,1) \geq (1 - G/GEN)^2\right) \text{ Then} \\
&v_i^{G+1} = \bar{x}_c^G + F1 \cdot (x_{\text{best}}^G - x_{\text{better}}^G) + F2 \cdot (x_{\text{best}}^G - x_{\text{worst}}^G) \\
&\quad + F3 \cdot (x_{\text{better}}^G - x_{\text{worst}}^G) \\
&\text{Else} \\
&v_{i,j}^{G+1} = x_{r_1,j}^G + F \cdot (x_{r_1,j}^G - x_{r_3,j}^G),
\end{aligned} \tag{13}$$

where $u(0, 1)$ returns a real number between 0 and 1 with uniform random probability distribution, G is the current generation number, GEN is the maximum number of generations, and F is set to 0.7 as recommended in Poikolainen and Neri (2013) and as the mean value, to balance exploration and exploitation capabilities, of the specified ranges suggested in Mallipeddi et al. (2011). From the above-mentioned 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 smaller than (1-G/GEN)^2, then the basic mutation is applied. Otherwise, the proposed one is performed. Certainly, it can be seen that, from Eq. (4), the probability of selecting one of these two new search strategies is a function of the generation number. The value of (1-G/GEN)^2 decreases rapidly from one to zero in order to balance exploration and exploitation. In fact, in the beginning, two different strategies are selected to produce the offspring. However, the probability of selecting the DE/rand/1/bin is greater than the probability of the proposed mutation. This process causes exploration. Then, as the generation increases, the two search strategies are still used. However, in the opposite, the probability of selecting the proposed mutation is greater than the probability of the DE/rand/1/bin. Finally, it enhances both the exploration and exploitation. Hence, based on the nonlinear decreasing probability rule and two mutations, the algorithm can balance the exploitation and exploration in the search space. The pseudo-code of EFADE is presented in Table 1. We next introduce the parameter adaptation of F and CR.

4.2 Parameter adaptation schemes in EFADE

The successful performance of DE algorithm is significantly dependent upon the choice of its two control parameters: the scaling factor F and crossover rate CR (Price et al. 2005; Das and Suganthan 2011). 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 general, F is an important parameter that controls the evolving rate of the population, i.e., it is closely related to the convergence speed (Qin et al. 2009). 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 (Feoktistov 2006). On the other hand, a large F value improves the exploration capability of the algorithm that can make the mutant vectors distribute widely in the search space and can increase the diversity of the population (Wang et al. 2011). However, it may slow down the search (Feoktistov 2006). Recently, the authors in Ronkkonen et al. (2005) claim that typically $0.4 < F < 0.95$ with $F = 0.9$ is a good first choice The constant crossover (Cr) reflects the probability with which the trial individual inherits the actual individual's genes, i.e., which and how many components are mutated in each element of the current population (Feoktistov 2006; Mallipeddi et al. 2011). The constant crossover (CR) practically controls the diversity of the population (Wang et al. 2011). 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 weak 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 non-separable 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 (Ronkkonen et al. 2005). 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 based on generating random values from uniform, normal or cauchy distributions or by generating different values from pre-defined parameter candidate pool use the previous experience (of generating better solutions) to guide the adaptation of these parameters (Noman and Iba 2008; Mallipeddi et al. 2011; Brest et al. 2006; Qin et al. 2009; Wang et al. 2011; Zhang and Sanderson 2009; Islam et al. 2012; Pan et al. 2011). However, there are few studies which make use of full information of the distribution conditions of each vector individual in successive populations dynamically through generations to generate suitable control parameters (Ghosh et al. 2011; Ali and Törn 2004; Zhang et al. 2010). The presented work differs com-

Table 1 Results of multiple-problem Wilcoxon's test between EFADE and other algorithms for $D = 10$

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
EFADE versus ADE	284.5	40.5	0.001	Yes	Yes
EFADE versus DE-APC	172	59	0.05	Yes	Yes
EFADE versus SaDE	162	163	0.989	No	No
EFADE versus TLBSaDE	104	172	0.301	No	No
EFADE versus b6e6rl	152	124	0.670	No	No
EFADE versus DE-IPS	325.5	80.5	0.005	Yes	Yes
EFADE versus SPSRDEMMS	286	32	0.001	Yes	Yes
EFADE versus DEcfbLS	153	172	0.798	No	No
EFADE versus SMADE	195	81	0.083	No	Yes
EFADE versus SHADE	94	137	0.455	No	No
EFADE versus jDEsoo	253	72	0.015	Yes	Yes
EFADE versus MDE-pBX	271	54	0.004	Yes	Yes
EFADE versus GA-TPC	273	27	0.000	Yes	Yes
EFADE versus f_k-PSO	256	69	0.012	Yes	Yes
EFADE versus CDASA	337.5	40.5	0.000	Yes	Yes
EFADE versus PLES	324	1	0.000	Yes	Yes
EFADE versus CMA-ES-RIS	184	116	0.331	No	No
EFADE versus CCPSO2	323	2	0.000	Yes	Yes

pletely from all other previous studies in the following major aspects.

(1) EFADE uses new adaptation schemes to update F and CR during generations without using extra parameters or determining specific learning periods like (Qin et al. 2009; Zhang and Sanderson 2009; Pan et al. 2011).
(2) Regarding CR, EFADE uses a pre-determined specific candidate pool for generating values for control parameter CR based on other research studies. Thus, EFADE benefits from the previous experience and results of other research studies by focusing on choosing the most appropriate values during the evolution process from narrow set of possible values of the control parameters. On the other hand, when it comes to the scaling factors in triangular mutation, EFADE benefits from the relative position of the randomly chosen vectors utilized to produce the difference vectors.

Therefore, taking into consideration all the above-mentioned guiding information about the major control parameters of DE, two novel fitness-based adaptation schemes are used for F and CR, respectively.

4.2.1 Adaptive scheme for scaling factors

Scale factors adaptation: at each generation G, the scale factors $F1$, $F2$ and $F3$ of each individual target vector are independently generated according to uniform distribution

$$F_i = \text{rand}(0, k_i), \quad i = 1, 2, 3, \tag{14}$$

$$k_1 = \begin{cases} \left|\frac{f(x_{\text{better}})}{f(x_{\text{best}})}\right| + \varepsilon, & \text{if } \left|\frac{f(x_{\text{better}})}{f(x_{\text{best}})}\right| < 1 \\ \left|\frac{f(x_{\text{best}})}{f(x_{\text{better}})}\right| + \varepsilon, & \text{otherwise,} \end{cases}$$
$$k_2 = \begin{cases} \left|\frac{f(x_{\text{worst}})}{f(x_{\text{best}})}\right| + \varepsilon, & \text{if } \left|\frac{f(x_{\text{worst}})}{f(x_{\text{best}})}\right| < 1 \\ \left|\frac{f(x_{\text{best}})}{f(x_{\text{worst}})}\right| + \varepsilon, & \text{otherwise,} \end{cases}$$
$$k_3 = \begin{cases} \left|\frac{f(x_{\text{worst}})}{f(x_{\text{better}})}\right| + \varepsilon, & \text{if } \left|\frac{f(x_{\text{worst}})}{f(x_{\text{better}})}\right| < 1 \\ \left|\frac{f(x_{\text{better}})}{f(x_{\text{worst}})}\right| + \varepsilon, & \text{otherwise,} \end{cases} \tag{15}$$

where $rand\,(a, b)$ is a function that returns a real number between a and b, in which a and b are not included. The small constant value $\varepsilon = 0.0001$ is used to avoid zero value of perturbation, but k_i are truncated to be 1 if $k_i > 1$. Thus, $k_1, k_2, k_3 = 1$ if $f(x_{\text{best}}) = f(x_{\text{better}})$, $f(x_{\text{best}}) = f(x_{\text{worst}})$ or $f(x_{\text{better}}) = f(x_{\text{worst}})$ with all values including zero. Practical experience through many developed evolutionary algorithms and experimental investigation prove that the success of population-based search algorithms is based on maintaining a balance dynamically between two contradictory aspects: global exploration and local exploitation (Das et al. 2009). It can be obviously seen from Eq. (1) that the scaling factors $F1$, $F2$ and $F3$ control significantly the amplification of the directed perturbations in the proposed mutation that considerably influence these two contradictory aspects through generations . Thus, the core ideas behind the fitness-based adaptation scheme for the scaling factors are based on the following two fundamental principles: First, at the initial phase of searching, the directed perturbations or the difference vectors in the proposed mutation are large due to the large variance of the distribution of the individual of the whole population in the solution space. It requires a relatively smaller scaling factor in order to avoid exceeding level of exploration and to keep on the search within the solution space that may result in slowing the search down. Then, through generation, the difference vectors continually decrease as the individuals of the population get much closer to the best vector found so far and become almost similar which significantly reduces the variance of the distribution of the vectors in the whole population. Thus, the scaling factors should be adopted with larger value to adjust the smaller difference vectors so that it promotes the local exploitation ability. Second, if the two randomly chosen vectors for producing the directed difference vector are nearby in the search space, the magnitude of the directed difference vector will be small. The scaling factor should be adopted with a larger value to modify the smaller difference to keep the mutation effect that increases local search. On the other hand, if the magnitude of the directed difference vector is larger, it means that the two randomly chosen vectors are separate in the search space. The scaling factor should be set a smaller value to adjust the larger difference to keep the mutation vector within search space, and this improves global search. According to the analysis and empirical results discussed in Weber et al. (2011), DE has limited amount of search moves. Additionally, the effectiveness of such moves explicitly depends on how the scale factor values generated over generations. Therefore, in order to balance global and local abilities of the algorithm as well as to enhance convergence speed, the scaling factors should be continually changed during the optimization process according to the relative position of the randomly chosen vectors utilized to produce the difference vector. Equation (4) shows that, if the randomly chosen vectors utilized to produce the difference vector are dispersed in the search space as the situation at the early stage of the search process, i.e., the magnitude of the difference vector is larger, it means that the fitness function value of $f(x_{\text{best}})$, $f(x_{\text{better}})$ or $f(x_{\text{worst}})$ has a small ratio k_i that is relatively close to zero. Correspondingly, the possible random value of scaling factor F_i uniformly generated by $\text{rand}(0, k_i)$ will be smaller, which limits the magnitude of the perturbation. Conversely, if the randomly chosen vectors utilized to produce the difference vector are nearby in the search space as the situation at the middle and later stages of the search process, i.e., the magnitude of the difference vector is smaller, the fitness function value of $f(x_{\text{best}})$, $f(x_{\text{better}})$ or $f(x_{\text{worst}})$ will have a large ratio k_i that is relatively close to 1. Correspondingly, the possible random value of scaling factor F_i uniformly generated by $rand(0, k_i)$ will be larger which prevents too small magnitude of the perturbation.

4.2.2 *Adaptive scheme for CR*

Crossover adaptation: At each generation G, the crossover probability CR of each individual target vector is independently generated according to one of the following two uniform distributions

$$(1)\ \text{CR}_1 \in [0.05, 0.15]; \quad (2)\ \text{CR}_2 \in [0.9, 1]. \tag{16}$$

In fact, the lower and upper limits of the ranges for these proposed sets are recommended by Ronkkonen et al. (2005). Obviously, at each generation G, these two sets are adaptively selected based on their experiences of generating promising solutions during the evolution process which are as follows:

If $G = 1$

$$\text{CR}_i^1 = \begin{cases} \text{CR}_1, & \text{if } u(0,1) \le 1/2 \\ \text{CR}_2, & \text{otherwise,} \end{cases}$$

Else

$$\text{CR}_i^G = \begin{cases} \text{CR}_1, & \text{if } u(0,1) \le p_1 \\ \text{CR}_2, & \text{if } p_1 < u(0,1) \le p_1 + p_2. \end{cases} \tag{17}$$

Denote $p_j, j = 1, 2, \ldots, m$ as the probability of selecting jth set, where m is the total number of sets and it is set to 2 and the sum of p_j is 1. p_j is initialized as $1/j$, i.e., 1/2. The roulette wheel selection method is used to select the appropriate set for each target vector based on the probability p_j. p_j is continuously updated during generations in the following manner:

$$p_j^{G+1} = \left((G-1) * p_j^{G-1} + ps_j^G\right)/G, \tag{18}$$

$$ps_j^G = \frac{s_j^G}{\sum_{j=1}^{m} s_j^G}, \tag{19}$$

where

$$s_j^G = \frac{ns_j^G}{\sum_{G=1}^{G} ns_j^G + \sum_{G=1}^{G} nf_j^G} + \varepsilon, \tag{20}$$

where G is the generation counter; ns_j^G and nf_j^G are the respective numbers of the offspring vectors generated by the jth set that survives or fails in the selection operation in the last G generations; s_j^G is the success ratio of the trial vector generated by the jth set and successfully entering the next generations; ps_j^G is the probability of selecting jth set in the current generation. ε is a small constant value to avoid the possible null success ratios. ε is set to 0.01 in our experiments to prevent s_j^G from becoming zero. It is clearly from the Eq. (6) that the probability p_j is the weighted mean of both the previous and current success ratios to consider the complete history of the experience of a specific set during generations. Moreover, in case of null success ratios of both sets the p_j is reinitialized as 1/2 to overcome the possibility of stagnation. The core idea of the fitness-based adaptation scheme for the crossover rate is based on the following two fundamental principles. First, 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 is clustered because each individual gets closer to the best vector found so far. Consequently, in this stage, in order to advance the diversity and increase the convergence speed, the value of CR must be a large value. Second, as previously mentioned, these two sets are designed as recommended in Ronkkonen et al. (2005) to deal effectively with problems with different characteristics such as unimodality and multimodality, separable and non-separable problems. The pseudo-code of EFADE is presented in Fig. 2.

It is noteworthy to mention that the proposed triangular mutation and the trigonometric mutation proposed by Fan and Lampinen (2003) 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 of the selected vectors. They are directed difference, i.e., resembles 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$(0, k_i)$, where k_i are the ratios between the objective function values of the best, better and worst vectors to enrich the search behavior by generating suitable random values to balance global and local abilities of the algorithm. 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, influences the new trial solution strongly and takes it 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 (Fan and Lampinen 2003). 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. 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 fitness-based adaptation scheme for gradual change of the values of the scaling factors and crossover rate can excellently benefit from the distribution of the individuals in the search space during the evolution process which in turn can considerably balance the common trade-off between the population diversity and convergence speed.

5 Numerical experiments and comparisons

In this section, the computational results of EFADE are discussed along with comparisons with other state-of-the-art algorithms

5.1 Experiments setup

The performance of the proposed DEAP algorithm was tested on 28 benchmark functions proposed in the CEC 2013 special session on real-parameter optimization. A detailed description of these test functions can be found in Liang et al. (2013). These 28 test functions can be divided into three classes:

(1) unimodal functions $f_1 - f_5$;
(2) multimodal functions $f_6 - f_{20}$;
(3) composition functions $f_{21} - f_{28}$;

5.2 Parameter settings and involved algorithms

To evaluate the performance of algorithms, experiments were conducted on the test suite. We adopt the solution error measure $(f(x) - f(x^*))$, where x is the best solution obtained by

```
01.  Begin
02.   G=0
03.   Create a random initial population $\vec{x}_i^G \ \forall i, i = 1,...,NP$, F=0.7.
04.   Evaluate $f(\vec{x}_i^G) \ \forall i, i = 1,...,NP$
05.   **For** G=1 to GEN **Do**
06.     **For** i=1 to NP **Do**
07.     Compute the scaling factors (K1,K2,K3) according to Eq. (14).
08.     Compute the (crossover rate) Cr$_i$ according to Eq. (17).
09.     **If** $\left(u(0,1) \geq (1 - G/GEN)^2\right)$ **Then** (Use New Triangular Mutation Scheme)
10.       Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$,      j$_{rand}$= randint(1,D)
11.       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 $\bar{x}_c^G$ the associated weights (p$_i$ ) according to eq (2).and eq(3).
12.       **For** j=1 to D **Do**
13.         **If** (rand$_j$[0,1]< CR **or** j= j$_{rand}$) **Then**
14.   ${}^{G+1}_{i,j} = \bar{x}_{c,j}^G + k_1(x_{best,j}^G - x_{better,j}^G) + k_2(x_{best,j}^G - x_{worst,j}^G) + k_3(x_{better,j}^G - x_{worst,j}^G)$
15.         **Else**
16.         $u_{i,j}^{G+1} = x_{i,j}^G$
17.         **End If**
18.       **End For**
19.     **Else** (Use Basic Mutation Scheme)
20.       Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$,      j$_{rand}$= randint(1,D)
21.       **For** j=1 to D **Do**
22.         **If** (rand$_j$[0,1]< CR **or** j= j$_{rand}$) **Then**
23.               $u_{i,j}^{G+1} = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$
24.         **Else**
25.         $u_{i,j}^{G+1} = x_{i,j}^G$
26.         **End If**
27.       **End For**
28.     **End**
29.           **If(** $f(\vec{u}_i^{G+1}) \leq f(\vec{x}_i^G)$ **)Then**
30.             $\vec{x}_i^{G+1} = \vec{u}_i^{G+1}$, ( $f(\vec{x}_i^{G+1}) = f(\vec{u}_i^{G+1})$ )
31.             **If(** $f(\vec{u}_i^{G+1}) \leq f(\vec{x}_{best}^G)$ **)Then**
32.             $\vec{x}_{best}^{G+1} = \vec{u}_i^{G+1}$, ( $f(\vec{x}_{best}^{G+1}) = f(\vec{u}_i^{G+1})$ )
33.             $ns_j^G = ns_j^G + 1$
34.             **End**
35.           **Else**
36.             $\vec{x}_i^{G+1} = \vec{x}_i^G$
37.             $nf_j^G = nf_j^G + 1$
38.           **End If**
39.       **End For**
40.         **Generate** $p_j^{G+1}$ according to Eq.(18) for the next generation
41.     G=G+1
42.   **End For**
43.  **End**
```

Fig. 2 Description of EFADE algorithm

algorithms in one run and x^* is well-known global optimum of each benchmark function. Error values and standard deviations smaller than 10^{-8} are taken as zero (Liang et al. 2013). The dimensions (D) of function are 10, 30 and 50, respectively. The maximum number of function evaluations (FEs), the terminal criteria, is set to $10{,}000 \times D$, all experiments for each function and each algorithm run 51 times independently. The population size NP in EFADE was set to 75 for $D = 10$ and 50 for $D = 30$ and 50. EFADE was compared to 18 population-based algorithms, 12 DE-based algorithms and 6 non-DE-based algorithms, which were all tested on CEC 2013 benchmark functions and gave very good results. These algorithms are:

- Population's variance-based adaptive differential evolution for real-parameter optimization (ADE) (Coelho et al. 2013).
- Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization (DE-APC) (Elsayed et al. 2013b).
- Investigation of self-adaptive differential evolution on the CEC-2013 real-parameter single-objective optimization (SaDE) (Qin et al. 2013).
- Teaching and learning best differential evolution with self-adaptation for real-parameter optimization (TLB-SaDE) (Biswas et al. 2013).
- Competitive differential evolution applied to CEC 2013 problems (b6e6rl) (Tvrdík and Poláková 2013).
- Differential evolution: performances and analyses (DE-IPS) (Padhye et al. 2013)
- Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real-parameter optimization (SPSRDEMMS) (Zamuda et al. 2013).
- Differential evolution with concurrent fitness-based local search (DEcfbLS) (Poikolainen and Neri 2013).
- Super-fit multicriteria adaptive differential evolution (SMADE) (Caraffini et al. 2013b).
- Evaluating the performance of SHADE on CEC 2013 benchmark problems (SHADE) (Tanabe and Fukunaga 2013).
- Real-parameter single-objective optimization using self-adaptive differential evolution with more strategies (jDEoo) (Brest et al. 2013)
- The self-adaptive differential evolution with $_P$BX cross over (MDE_$_P$BX) (Caraffini et al. 2013b).
- A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization (GA-TPC) (Elsayed et al. 2013a).
- A self-adaptive heterogeneous PSO for real-parameter optimization (f_k-PSO) (Nepomuceno and Engelbrecht 2013).
- The continuous differential ant-stigmergy algorithm applied on real-parameter single-objective optimization problems (CDASA) (Korošec and Šilc 2013).
- The parameter-less evolutionary search for real-parameter single-objective optimization (PLES) (Papa and Šilc 2013).
- A CMA-ES super-fit scheme for the re-sampled inheritance search (CMA-ES) (Caraffini et al. 2013a).
- Cooperatively coevolving particle swarms for large-scale optimization (CCPSO2) (Caraffini et al. 2013b).

To perform comprehensive evaluation, the presentation of the experimental results is divided into two subsections. First, an overall performance comparison between EFADE and other 18 state-of-the-art algorithms is provided. Second, the effectiveness of the proposed modifications is discussed. To compare and analyze the solution quality from a statistical angle of different algorithms and to check the behavior of the stochastic algorithms (Garcia et al. 2009), the results are compared using two nonparametric statistical hypothesis tests: (i) multiproblem Wilcoxon signed-rank test (to check the differences between all algorithms for all functions); at a 0.05 significance level (Caraffini et al. 2013b), 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. 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 $N = 28$ for $D = 10$, 30 and 50 dimensions and 5% significance level. Use the p value and compare it with the significance level. Reject the null hypothesis if the p-value is less than or equal the significance level (5%). All the p values in this paper were computed using SPSS (the version is 20.00); (ii) the Friedman test (to obtain the final rankings of different algorithms for all functions).

5.3 Experimental results and discussions

5.3.1 Results of the proposed approach (EFADE)

The statistical results of the EFADE on the benchmarks with 10, 30, and 50 dimensions are summarized in supplemental file (Tables S1–S3). It includes the known optimal solution for each test problem and the obtained best, median, mean, worst values and the standard deviations of error from optimum solution of the proposed EFADE over 51 runs for all 28 benchmark functions. Generally, for 10D problems, Table 1 clearly shows that EFADE has outstanding performance on

Table 2 Average ranking of EFADE and other algorithms according to Friedman test for $D = 10$

Rank	Algorithm	Mean ranking
1	SHADE	5.80
2	TLBSaDE	6.61
3	EFADE	6.89
4	DEcfbLS	7.05
5	SaDE	7.68
6	SMADE	7.70
7	b6e6rl	7.82
8	CMA-ES-RIS	9.14
9	DE-APC	9.91
10	SPSRDEMMS	10.04
11	jDEsoo	10.23
12	f_k-PSO	10.25
13	MDE-pBX	10.36
14	ADE	10.93
15	DE-IPS	11.36
16	CCPSO2	13.54
17	GA-TPC	13.86
18	CDASA	14.32
19	PLES	16.52

unimodal problems (f_1-f_5); EFADE is able to find the global optimal solution consistently in four problems over 51 runs with the exception of problems f_3. With respect to f_3, the smooth but narrow bridge present is a challenge to all algorithms that prevent EFADE from finding global solution consistently. For the multimodal functions (f_6-f_{20}), EFADE is able to find the global optimal solution, at least once, in five problems, while it is very close to the optimal solution for the rest problems with the exception of problems f_{15}. EFADE gets trapped in local optima only on f_{15}, where f_{15} has a huge number of local optima and the location of the second best local optimum being far away from the global one makes the problem challenging (Biswas et al. 2013). The composition functions ($f_{21}-f_{28}$) considered the most difficult problems in the benchmark suite as they are highly multimodal, non-separable and possess different properties around local optima (Liang et al. 2013). Thus, EFADE is often trapped in local optimum. From the results presented in Tables 2 and 3, for 30D and 50D problems, a similar trend as was observed in 10D is continued here. For unimodal functions, EFADE is able to find the global optimal solution consistently for problems (f_1 and f_5), while it is very close to the optimal solution in problems (f_3 and f_4). Besides, the performance of EFADE degrades in f_2 due to the high conditioning of the elliptic function (Liang et al. 2013). Concerning multimodal functions, EFADE is able to obtain good solutions for all test problems, except f_{15}. In regard to the composition function, the algorithm may easily get stuck in local optimum. Therefore, it can be clearly observed that, apart from f_2 (in 30D) and f_2, f_3 (in 50D), EFADE is able to reach the global optimum in case of the unimodal functions. Regarding the multimodal functions, EFADE reaches results which show small deviation from the global optimum and gets very close to the optimum in all functions in all dimensions with exception to problem $f_{15.}$ On the other hand, regarding com-

Table 3 Results of multiple-problem Wilcoxon's test between EFADE and other algorithms for $D = 30$

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
EFADE versus ADE	266	112	0.064	No	Yes
EFADE versus DE-APC	320	58	0.002	Yes	Yes
EFADE versus SaDE	272	106	0.046	Yes	Yes
EFADE versus TLBSaDE	215	163	0.532	No	No
EFADE versus b6e6rl	241	137	0.212	No	No
EFADE versus DE-IPS	375	31	0.000	Yes	Yes
EFADE versus SPSRDEMMS	248	103	0.066	No	Yes
EFADE versus DEcfbLS	256	95	0.041	Yes	Yes
EFADE versus SMADE	237	141	0.249	No	No
EFADE versus SHADE	34.5	343.5	0.000	Yes	Yes
EFADE versus jDEsoo	291	87	0.014	Yes	Yes
EFADE versus MDE-pBX	373	33	0.000	Yes	Yes
EFADE versus GA-TPC	311	40	0.001	Yes	Yes
EFADE versus f_k-PSO	305.5	100.5	0.020	Yes	Yes
EFADE versus CDASA	366	40	0.000	Yes	Yes
EFADE versus PLES	390	16	0.000	Yes	Yes
EFADE versus CMA-ES-RIS	260	146	0.194	No	No
EFADE versus CCPSO2	289	62	0.004	Yes	Yes

Table 4 Average ranking of EFADE and other algorithms according to Friedman test for $D = 30$

Rank	Algorithm	Mean ranking
1	SHADE	4.36
2	TLBSaDE	7.25
3	EFADE	7.79
4	SaDE	8.16
5	SMADE	8.20
6	SPSRDEMMS	8.55
7	b6e6rl	8.63
8	jDEsoo	8.64
9	CMA-ES-RIS	8.70
10	DEcfILS	9.29
11	f_KPSO	10.71
12	DE-APC	11.20
13	MDE-pBX	11.23
14	ADE	11.32
15	CCPSO2	11.57
16	GA-TPC	12.21
17	CDASA	12.79
18	DE-IPS	13.04
19	PLES	16.38

position functions, apart from f27 (in 50D) and f23 (in 30D and 50D), EFADE is often trapped in local optimum which is still not far away from the optimum in all functions for all the three dimensionalities. Additionally, in all functions for all the three dimensionalities, 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 dimensions. That implies that EFADE is a robust algorithm. Finally, due to insignificant difference between the results in three dimensions, it can be concluded that the performance of the EFADE algorithm slightly diminishes and it is still more stable and robust against the curse of dimensionality, i.e., it is overall steady as the dimensions of the problems increase.

5.3.2 Comparison against state-of-the-art algorithms

The statistical results of the comparisons on the benchmarks with 10, 30 and 50 dimensions are summarized in supplemental file (Tables S4–S6), (Tables S7–S9) and (Tables S10–S12), respectively. It includes the obtained best and the standard deviations of error from optimum solution of EFADE and other 18 state-of-the-art algorithms over 51 runs for all 28 benchmark functions. The results provided by these approaches were directly taken from their references. The best results are marked in bold for all problems. The multi-problem Wilcoxon signed-rank and Friedman tests between EFADE and others in 10D, 30D and 50D are summarized in Tables 1, 2, 3, 4, 5 and 6, respectively. Firstly, regarding DE-based algorithms in three dimensions, it can be observed that different DE algorithms can be good at different functions in some or all dimensions except that DE-IPS performs poorly on all the functions in all dimensions. Generally, EFADE, SHADE, SMADE and TLBSaDE do significantly better than the others on most functions in different dimensions. On the other hand, regarding non-DE algorithms, it can be obviously shown that CMA-ES-RIS is competitive with EFADE. However, EFADE is superior to the others in most functions in all dimensions. Secondly, the performance of EFADE and other competitive algorithms on the functions of different dimensions is discussed. Table 1 summarizes the statistical analysis results of applying multiple-problem Wilcoxon's test between EFADE and other compared algorithms for 10D problems. From Table 1, we can see that EFADE obtains higher R^+ values than R^- in most of cases, while slightly lower R^+ value than R^- value in comparison with SaDE. However, in the cases of EFADE versus SHADE, TLBSaDE and DEcfbLS, they get higher R^- than R^+ values. The reason is that EFADE gains the performance far away of what these three algorithms do on some functions, resulting in higher ranking values. According to the Wilcoxon's test at $\alpha = 0.05$ and 0.1, the significance difference can be observed in 11 and 12 cases, respectively, which means that EFADE is significantly better than 12 algorithms out of 18 algorithms on 28 test functions at $\alpha = 0.1$. Besides, Table 2 lists the average ranks EFADE and other algorithms according to Friedman test for $D = 10$. The p value computed through Friedman test is 0.00E+00. Thus, it can be concluded that there is a significant difference between the performances of the algorithms. Table 2 clearly shows that SHADE gets the first ranking among all algorithms in ten-dimensional functions, followed by TLBSaDE and EFADE. Regarding 30D and 50D problems, the results of multiproblem Wilcoxon's test in Tables 3 and 5 show that EFADE obtains higher R^+ values than R^- in all cases with exception of SHADE algorithm, where SHADE gets higher R^- values than R^+ values. The reason is that SHADE outperforms EFADE in most of the cases. According to the Wilcoxon's test at $\alpha = 0.05$ and 0.1, the significance difference can be observed in 11 and 13 cases, respectively, which means that EFADE is significantly better than 13 algorithms out of 18 algorithm on 28 test functions at $\alpha = 0.1$. Besides, Tables 4 and 6 list the average ranks EFADE and other algorithms according to Friedman test for D = 30 and 50. The p value computed through Friedman test is 0.00E+00. Thus, it can be concluded that there is a significant difference between the performances of the algorithms. Tables 4 and 6 clearly shows that SHADE gets the first ranking among all algorithms in 30-dimensional functions, followed by TLBSaDE and EFADE, while SHADE gets the first ranking among all algorithms in 50-dimensional functions, followed by EFADE and TLB-

Table 5 Results of multiple-problem Wilcoxon's test between EFADE and other algorithms for $D = 50$

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
EFADE versus ADE	320	5	0.000	Yes	Yes
EFADE versus DE-APC	289	36	0.001	Yes	Yes
EFADE versus SaDE	254	71	0.014	Yes	Yes
EFADE versus TLBSaDE	199	126	0.326	No	No
EFADE versus b6e6rl	213	63	0.023	Yes	Yes
EFADE versus DE-IPS	363	15	0.000	Yes	Yes
EFADE versus SPSRDEMMS	242	83	0.032	Yes	Yes
EFADE versus DEcfbLS	196	80	0.078	No	Yes
EFADE versus SMADE	209	116	0.211	No	No
EFADE versus SHADE	112	238	0.110	No	No
EFADE versus jDEsoo	274	77	0.012	Yes	Yes
EFADE versus MDE-pBX	322	29	0.000	Yes	Yes
EFADE versus GA-TPC	293	95	0.024	Yes	Yes
EFADE versus f_k-PSO	280	45	0.002	Yes	Yes
EFADE versus CDASA	282	69	0.007	Yes	Yes
EFADE versus PLES	324	1	0.000	Yes	Yes
EFADE versus CMA-ES-RIS	242	136	0.203	No	No
EFADE versus CCPSO2	232	93	0.061	No	Yes

SaDE. Furthermore, the performance of all algorithms is analyzed using all dimensions and different categories of functions. Therefore, the mean aggregated rank of all the 18 algorithms across all problems (28) and all dimensions (10D, 30D, 50D) is presented in Table 7. Additionally, the mean aggregated ranks of all the 18 algorithms across unimodal problems, multimodal problems and composition problems and all dimensions (10D, 30D, 50D) are presented in Tables 8, 9 and 10, respectively. From Table 7, it can be clearly concluded that SHADE is the best followed by EFADE as second best among all algorithms. From Tables 8, 9 and 10, it can be obviously shown that EFADE gets the fifth, fourth and first ranking in unimodal, multimodal and compositions functions, respectively. Overall, from the above results, comparisons and discussion, the proposed EFADE algorithm is of better searching quality, efficiency and robustness for solving unconstrained global optimization problems. It is clear that the proposed EFADE algorithm performs well and it has shown its outstanding superiority with separable, non-separable, unimodal and multimodal functions with shifts in dimensionality, rotation, multiplicative noise in fitness and composition of functions. Consequently, its performance is not influenced by all these obstacles. Contrarily, it greatly keeps the balance between the local optimization speed and the global optimization diversity in challenging optimization environment with invariant performance. Besides, its performance is superior and competitive with the performance of the state-of-the-art well-known algorithms. Moreover, compared to the complicated structures and number of methods and number of control parameters used in other algorithms such as SMADE that uses four mutation Strategies combined with well-known CMA-ES optimizer with adaptive crossover and scaling factors, we can see that EFADE is very simple and easy to implement and program in many programming languages. It only uses one mutation rule and very simple adaptive crossover rate and scaling factors without extra parameters. Thus, it increases neither the complexity of the original DE algorithm nor the number of control parameters. Finally, it can be obviously concluded that the power of the EFADE algorithm is especially observed for composition functions, known to be harder to optimize. The superior and outstanding performance of EFADE stems from two aspects: (1) the proposed triangular mutation scheme that enhances the global search capability and local search tendency of the algorithm by motivating the population toward promising directions and (2) the two simple adaptive schemes of crossover and scaling factors that balance the diversity and convergence speed by introducing the most appropriate values during optimization process.

6 EFADE parametric study

In this section, in order to investigate the impact of the proposed modifications, some experiments are conducted. Another four different versions of EFADE have been tested and compared against the proposed one. These four versions of EFADE have been compared with the proposed one on 28 benchmark functions benchmark on 50 dimensions.

Table 6 Average ranking of EFADE and other algorithms according to Friedman test for $D = 50$

Rank	Algorithm	Mean ranking
1	SHADE	4.55
2	EFADE	6.79
3	SPSRDEMMS	7.95
4	SMADE	8.29
5	CMA-ES-RIS	8.38
6	TLBSaDE	8.46
7	DEcfbLS	8.8
8	b6e6rl	8.82
9	SaDE	8.88
10	jDEsoo	9.73
11	f_k-PSO	10.8
12	MDE-pBX	10.84
13	DE-APC	11.25
14	CCPSO2	11.27
15	GA-TPC	11.46
16	CDASA	11.86
17	ADE	12.39
18	DE-IPS	13.3
19	PLES	16.18

Table 7 Average ranks for all algorithms across all problems and all dimensions

Rank	Algorithm	10D	30D	50D	Mean ranking
1	SHADE	5.80	4.36	4.55	4.90
2	EFADE	6.89	7.79	6.79	7.15
3	TLBSaDE	6.61	7.25	8.46	7.44
4	SMADE	7.70	8.20	8.29	8.06
5	SaDE	7.68	8.16	8.88	8.24
6	DEcfbLS	7.05	9.29	8.80	8.38
7	b6e6rl	7.82	8.63	8.82	8.42
8	CMA-ES-RIS	9.14	8.70	8.38	8.74
9	SPSRDEMMS	10.04	8.55	7.95	8.85
10	jDEsoo	10.23	8.64	9.73	9.54
11	f_k-PSO	10.25	10.71	10.80	10.59
12	DE-APC	9.91	11.20	11.25	10.79
13	MDE-pBX	10.36	11.23	10.84	10.81
14	ADE	10.93	11.32	12.39	11.55
15	CCPSO2	13.54	11.57	11.27	12.13
16	GA-TPC	13.86	12.21	11.46	12.51
17	DE-IPS	11.36	13.04	13.30	12.57
18	CDASA	14.32	12.79	11.86	12.99
19	PLES	16.52	16.38	16.18	16.36

Table 8 Average ranks for all algorithms on unimodal functions and all dimensions

Rank	Algorithm	10D	30D	50D	Mean ranking
1	SMADE	5.80	5.30	4.10	5.07
2	SHADE	6.30	5.30	5.10	5.57
3	CMA-ES-RIS	7.10	4.90	5.20	5.73
4	TLBSaDE	7.50	6.10	7.50	7.03
5	EFADE	7.30	8.10	6.90	7.43
6	b6e6rl	8.60	6.90	7.50	7.67
7	DE-APC	5.80	9.50	9.80	8.37
8	GA-TPC	5.80	10.50	12.20	9.50
9	SaDE	12.30	9.50	8.90	10.23
10	MDE-pBX	10.30	10.50	9.90	10.23
11	SPSRDEMMS	11.60	10.10	9.50	10.40
12	ADE	10.40	12.70	10.20	11.10
13	DEcfbLS	11.20	10.70	12.50	11.47
14	DE-IPS	12.80	11.80	12.80	12.47
15	CDASA	13.00	12.90	12.00	12.63
16	f_k-PSO	13.20	13.30	12.90	13.13
17	jDEsoo	11.80	13.10	15.20	13.37
18	CCPSO2	14.20	14.10	13.70	14.00
19	PLES	15.00	14.70	14.10	14.60

Table 9 Average ranks for all algorithms on multimodal functions and all dimensions

Ranks	Algorithms	10D	30D	50D	Mean ranking
1	SHADE	5.37	4.10	4.07	4.51
2	TLBSaDE	6.50	6.47	7.70	6.89
3	DEcfbLS	6.23	8.20	8.47	7.63
4	EFADE	7.93	7.10	8.07	7.70
5	SPSRDEMMS	8.77	8.13	7.33	8.08
6	SaDE	6.93	8.40	9.57	8.30
7	SMADE	7.93	8.73	8.40	8.36
8	jDEsoo	9.00	7.80	8.53	8.44
9	b6e6rl	7.43	8.87	9.27	8.52
10	DE-APC	9.00	10.90	10.07	9.99
11	f_k-PSO	8.90	10.57	10.57	10.01
12	CMA-ES-RIS	11.53	9.97	9.33	10.28
13	MDE-pBX	10.47	11.63	11.40	11.17
14	CCPSO2	14.50	11.40	10.67	12.19
15	ADE	10.90	12.37	13.73	12.33
16	CDASA	13.87	12.10	11.13	12.37
17	DE-IPS	11.97	13.47	12.90	12.78
18	GA-TPC	15.60	12.97	12.17	13.58
19	PLES	17.17	16.83	16.63	16.88

Table 10 Average ranks for all algorithms on composition functions and all dimensions

Rank	Algorithms	10D	30D	50D	Mean ranking
1	EFADE	4.69	6.56	4.31	5.19
2	SHADE	6.31	4.44	5.13	5.29
3	SaDE	6.19	7.00	7.56	6.92
4	CMA-ES-RIS	5.94	8.81	8.56	7.77
5	DEcfbLS	6.00	10.56	7.13	7.90
6	b6e6rl	8.06	9.44	8.81	8.77
7	TLBSaDE	6.25	9.63	10.50	8.79
8	jDEsoo	11.56	7.56	8.56	9.23
9	SPSRDEMMS	11.44	8.56	8.13	9.38
10	SMADE	8.44	9.13	10.69	9.42
11	f_k-PSO	10.94	9.50	9.94	10.13
12	ADE	11.31	8.56	11.25	10.38
13	MDE-pBX	10.19	11.06	10.38	10.54
14	CCPSO2	11.31	10.50	10.88	10.90
15	DE-IPS	9.31	13.00	14.38	12.23
16	GA-TPC	15.63	12.00	9.69	12.44
17	DE-APC	14.19	12.94	14.38	13.83
18	CDASA	16.00	14.06	13.13	14.40
19	PLES	16.25	16.69	16.63	16.52

(1) *Version 1* To study the effectiveness of the proposed new mutation scheme alone, the proposed adaptive schemes for crossover rate and scaling factors are combined with the proposed triangular mutation rule only. This version denotes the EFADE-1.

(2) *Version 2* To study the effectiveness of the basic mutation scheme alone, the proposed adaptive scheme for crossover rate is combined with basic mutation rule only. This version denotes the EFADE-2.

(3) *Version 3* To investigate the impact of the proposed self-adaptive scheme for the scaling factors, the scale factors $F1$, $F2$ and $F3$ of each individual target vector are independently generated according to uniform distribution $F_i = rand(0, 1), i = 1, 2, 3$. This version denotes the EFADE-3.

(4) *Version 4* To investigate the impact of the proposed self-adaptive scheme for crossover rate, the recommended constant crossover rate 0.9 is used. This version denotes the EFADE-4.

The overall comparison results of the EFADE algorithm against its versions (EFADE-1, EFADE-2, EFADE-3 and EFADE-4) on the benchmarks with 50 dimensions are summarized in supplemental file in Table S13. It includes the known optimal solution for each test problem and the obtained best and the standard deviations of error from optimum solution of EFADE, EFADE-1, EFADE-2, EFADE-3 and EFADE-4 algorithms over 51 runs for all 28 benchmark functions. The best results are marked in bold for all problems. It can be observed from Table S13 that EFADE and its versions exhibit equal performance on two unimodal functions f_1 and f_2 and one multimodal function f_8. Moreover, EFADE, EFADE-2, EFADE-3 and EFADE-4 do better than EFADE-1 on one multimodal function f_6. EFADE and EFADE-2 do better than EFADE-1, EFADE-3 and EFADE-4 on multimodal function f_{16}.Additionally, EFADE, EFADE-2 and EFADE-3 do better than EFADE-1 and EFADE-4 on one composition function f_{28}. However, EFADE-2 outperforms EFADE, EFADE-1, EFADE-3 and EFADE-4 on one composition function f_{26}. Besides, EFADE-2 performs better than EFADE-1 on f_{11}, f_{20} and f_{21}. Therefore, EFADE-2 can obtain higher accuracy solution than EFADE-1 on seven out of 28 problems. On the other hand, EFADE-1 outperforms EFADE and EFADE-2 in two unimodal functions f_2 and f_4, four multimodal functions f_{14}, f_{15}, f_{17} and f_{19} and two composition functions f_{22} and f_{23}. Furthermore, EFADE-1 significantly outperforms EFADE-2 on one unimodal function f_3, six multimodal functions f_7, f_9, f_{10}, f_{12}, f_{13} and f_{18} and three composition functions f_{24}, f_{25} and f_{27}. It is worthy mentioning that f_2, f_3, f_{14}, f_{15}, f_{22} and f_{23} are very hard problems to be optimized especially in high dimensions as mentioned previously. Overall, it can be concluded that EFADE-1 performs better than EFADE-2 on 18 out of 28 functions which proves the main benefits of the proposed triangular mutation in balancing exploration and exploitation capabilities that leads to outstanding and superior performance against the basic mutation. On the other hand, EFADE-3 is slightly better than EFADE on one unimodal function f_3, three multimodal functions f_7, f_9 and f_{10} and three composition functions f_{24}, f_{25} and f_{27}. However, EFADE significantly outperforms EFADE-3 on two unimodal functions f_2, f_4, eight multimodal functions f_{12}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19} and f_{20} and three composition functions f_{21}, f_{22} and f_{23} which proves the vital role of proposed adaptive scaling factors scheme in improving the quality of final solution by expediting the convergence speed. Furthermore, EFADE-4 is slightly better than EFADE on five multimodal functions f_7, f_{10}, f_{12}, f_{13} and f_{19} and three composition functions f_{24}, f_{25} and f_{27}. On the contrary, EFADE significantly outperforms EFADE-4 on three unimodal functions f_2, f_3 and f_4, eight multimodal functions f_9, f_{11}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18} and f_{20} and five composition functions f_{21}, f_{22}, f_{23}, f_{26} and f_{28} which explains the significant impact of using proposed adaptive scheme of crossover rate in controlling diversity. Regarding combined effect of both mutations, EFADE outperforms both EFAD-1 and EFADE-2 on one unimodal function f_3, eight multimodal functions f_7, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{18}, f_{20} and four composition functions f_{21}, f_{24}, f_{25}, f_{27} which proves the benefits of using both mutations. Secondly, the performance

Table 11 Results of multiple-problem Wilcoxon's test between EFADE, EFADE-1, EFADE-2, EFADE-3 and EFADE-4 algorithms for $D = 50$

Algorithm	R^+	R^-	p value	$\alpha = 0.05$	$\alpha = 0.1$
EFADE versus EFADE-1	202	123	0.288	No	No
EFADE versus EFADE-2	247	6	0.000	Yes	Yes
EFADE versus EFADE-3	200	76	0.059	No	Yes
EFADE versus EFADE-4	292	85	0.013	Yes	Yes
EFADE-1 versus EFADE-2	261	64	0.008	Yes	Yes
EFADE-1 versus EFADE-3	167	211	0.597	No	No
EFADE-1 versus EFADE-4	185	193	0.923	No	No
EFADE-2 versus EFADE-3	42	258	0.002	Yes	Yes
EFADE-2 versus EFADE-4	135	243	0.195	No	No
EFADE-3 versus EFADE-4	283	85	0.024	Yes	Yes

of EFADE and its versions on the functions of 50D are statistically validated. Table 11 summarizes the statistical analysis results of applying multiple-problem Wilcoxon's test between EFADE and other compared four versions for 50D problems. From Table 11, we can see that EFADE obtains higher R^+ values than R^- values in comparison with its four versions and EFADE-1 obtains higher R^+ values than R^- values in comparison with EFADE-2 which supports the previous analysis. Obviously, the reason is that EFADE-2 gains the performance far away of what these two algorithms do on some functions, resulting in higher ranking values. Furthermore, EFADE-3 and EFADE-4 obtain higher R^+ values than R^- values in comparison with EFADE-1 and EFADE-2. However, EFADE-3 obtains higher R^+ values than R^- values in comparison with EFADE-4. According to the Wilcoxon's test at $\alpha = 0.05$ and 0.1, the significance difference can be observed in five cases only (i.e., EFADE vs EFADE-2, EFADE vs EFADE-4, EFADE-3 vs EFADE-2, EFADE-3 vs EFADE-4 and EFADE-1 vs EFADE-2), which means that EFADE and EFADE-3 are significantly better than EFADE-2 and EFADE-4, respectively, and EFADE-1 is significantly better than EFADE-2. However, the insignificant difference can be observed in four cases (i.e., EFADE vs EFADE-1, EFADE-1 vs EFADE-3, EFADE-1 vs EFADE-4 and EFADE-2 vs EFADE-4), which means are competitive algorithms. When $\alpha = 0.1$, the significance difference can be observed in one case only (EFADE vs EFADE-3), which means that EFADE is significantly better than EFADE-3. Besides, Table 12 lists the average ranks of EFADE, EDAFE-1 and EFADE-2 algorithms according to Friedman test for $D = 50$. The p value computed through Friedman test is 0.00E+00. Thus, it can be concluded that there is a significant difference between the performances of the algorithms. Table 12 clearly shows that EFADE get the first ranking among all algorithms in 50-dimensional functions, followed by EFADE-3, EFADE-1, EFADE-4 and EFADE-2. Furthermore, in order to analyze the convergence behavior of each compared algorithm, the convergence characteristics in terms

Table 12 Average ranking of EFADE, EFADE-1, EFADE-2, EFADE-3 and EFADE-2 algorithms according to Friedman test for $D = 50$

Rank	Algorithm	Mean ranking
1	EFADE	2.13
2	EFADE-3	2.70
3	EFADE-1	3.11
4	EFADE-4	3.29
5	EFADE-2	3.77

of the best fitness value of the median run of each algorithm for 28 functions with dimension 50 are illustrated in supplemental file (Fig. S1), and it shows that EFADE and EFADE-1 algorithms converge to better or global solution faster than EFADE-2 in all cases. EFADE-3 converges slower than EFADE in most cases, while EFADE-4 converges faster than EFADE on two multimodal functions f_7, f_{12} and three composition functions f_{24}, f_{25} and f_{27}. On the other hand, in early stage, EFADE-1 algorithm converges faster than EFADE on all functions with exception to four multimodal functions f_{11}, f_{15}, f_{16} and f_{20} and one composition function f_{26} where EFADE-1 is slightly slower than EFADE. However, in later stage, EFADE-1 may be stagnated or it prematurely converges to local optima, while EFADE reaches better solutions due to the embedded DE/rand/1/bin strategy that enhances the exploration ability of EFADE as expected. It is clear that the proposed modifications play a vital role and has a significant impact on improving the convergence speed of EFADE algorithm for most problems. The EFADE algorithm has a considerable ability to maintain its convergence rate, improve its diversity as well as advance its local tendency through a search process. Thus, after the above analysis and discussion, the two proposed algorithms EFADE and EFADE-1 show competitive performance in terms of quality of solution, efficiency, convergence rate and robustness. They are superior to EFADE-2 algorithm that is basically based on DE/rand/1/ bin strategy. Accordingly, the main benefits of the proposed modifications are the remarkable balance between the exploration capability and exploitation tendency through the optimization process. This balance leads to superior performance with fast convergence speed and the extreme robustness over the entire range of benchmark functions which are the weak points of all evolutionary algorithms.

7 Conclusion

In order to enhance the overall performance of basic DE algorithm, introducing new mutation rules combined with advanced adaptive and/or self-adaptive schemes for crossover rate and scaling factor is a must although it is a challenging task. In this paper, a new DE algorithm, named EFADE, is proposed for solving global numerical optimization problems over continuous space. In the proposed algorithm, a new triangular mutation operator is introduced. It is 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. Besides, two novel and effective adaptation schemes are used to update the control parameters to appropriate values without either extra parameters or prior knowledge of the characteristics of the optimization problem. The proposed novel approach to mutation operator is shown to enhance the global and local search capabilities and to increase the convergence speed of the new algorithm compared with classical scheme. In order to test the effectiveness of EFADE, it is applied to solve the CEC-2013 real-parameter benchmark optimization problems. Experimental results are compared with 18 state-of-the-art algorithms which were all tested on CEC 2013 benchmark functions and gave very good results and two of its variants. In order to statistically analyze the performance of EFADE, two nonparametric tests (the Friedman test and Wilcoxon's test) are used with the significance level of 0.05. As a summary of results, the performance of the EFADE algorithm was statistically superior to and competitive with other recent and well-known state-of-the-art algorithms in the majority of functions and for different dimensions especially for composition functions. Virtually, it is easily implemented and has been proven to be a reliable approach for real-parameter optimization. In addition to its very promising performance, the effectiveness and benefits of the proposed modifications used in DEAP were experimentally investigated and compared. It was found that the proposed triangular mutation scheme has outperformed standard DE/rand/1/ bin mutation rules in the majority of functions on 50 dimensions. Moreover, experimental results indicate that the proposed adaptive schemes for crossover rate and scaling factor play a vital role in promoting the effectiveness of EFADE. Several current and future works can be developed from this study. Firstly, current research efforts focus on how to modify the EFADE algorithm for handling constrained and multiobjective optimization problems as well as solving practical engineering optimization problems and real-world applications. Secondly, concerning the improvement of EFADE, it would be very interesting to propose another adaptive EFADE version by combining other mutation strategies. Future research studies may focus on applying the algorithm to solve high dimensions or large-scale global optimization problems. Another possible direction is integrating the proposed mutation scheme with other self-adaptive DE variants like SHADE and TLBSaDE and many others. Finally, hybridizing the EFADE algorithm with other powerful metaheuristics, CMA-ES-RIS and CCPSO2 are thought to be promising direction. Furthermore, it will be greatly beneficial as a future direction to investigate a completely parameter tune-free adaptive EFADE by combining novel population reduction and increment method. The Matlab source code of EFADE can be downloaded from https://sites.google.com/view/optimization-project/files

Compliance with ethical standards

Conflict of interest Authors Ali Wagdy Mohamed and Ponnuthurai Nagaratnam Suganthan do not have conflict of interest

Ethical standard This article does not contain any studies with human participants performed by any of the authors

References

Ali MM, Törn A (2004) Population set based global optimization algorithms: some modifications and numerical studies. Comput Oper Res 31:1703–1725

Biswas S, Kundu S, Das S, Vasilakos AV (2013) Teaching and learning best differential evolution with self adaptation for real parameter optimization. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1115–1122

Brest J, Greiner S, Bošković B, Mernik M, žumer V, (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evolut Comput 10(6):646–657

Brest J, Boškovič B, Zamuda A, Fister I, Mezura-Montes E (2013) Real parameter single objective optimization using self-adaptive differential evolution with more strategies. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 377–383

Caraffini F, Iacca G, Neri F, Picinali L, Mininno E (2013a) A CMA-ES super-fit scheme for the re-sampled inheritance search. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1123–1130

Caraffini F, Neri F, Cheng J, Zhang G, Picinali L, Iacca G, Mininno E (2013b) Super-fit multicriteria adaptive differential evolution. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1678–1685

Coelho LS, Ayala HVH, Freire RZ (2013) Population's variance-based adaptive differential evolution for real parameter optimization. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1672–1677

Das SS, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evolut Comput 15(1):4–31

Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood based mutation operator. IEEE Trans Evolut Comput 13(3):526–53

Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution-an updated survey. Swarm Evolut Comput 27:1–30

Draa A, Bouzoubia S, Boukhalfa I (2015) A sinusoidal differential evolution algorithm for numerical optimization. Appl Soft Comput 27:99–126

El-Quliti SA, Ragab AH, Abdelaal R et al (2015) A nonlinear goal programming model for university admission capacity planning with modified differential evolution algorithm. Math Probl Eng 2015:13

El-Qulity SA, Mohamed AW (2016) A generalized national planning approach for admission capacity in higher education: a nonlinear integer goal programming model with a novel differential evolution algorithm. Comput Intell Neurosci 2016:14

El-Quliti SA, Mohamed AW (2016) A large-scale nonlinear mixed binary goal programming model to assess candidate locations for solar energy stations: an improved real-binary differential evolution algorithm with a case study. J Comput Theor Nanosci 13(11):7909–7921

Elsayed SM, Sarker RA, Ray T (2013a) A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 356–360

Elsayed SM, Sarker RA, Ray T (2013b) Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1932–1937

Fan HY, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Glob Optim 27(1):105–129

Feoktistov V (2006) Differential evolution: in search of solutions. Springer, Berlin

Gämperle R, Müller SD, Koumoutsakos P (2002) A parameter study for differential evolution. In: Grmela A, Mastorakis NE (eds) Advances in intelligent systems, fuzzy systems, evolutionary computation. WSEAS Press, Interlaken, Switzerland, pp 293–298

Garcia G, Molina SD, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behavior: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15:617–644

Ghosh A, Das S, Chowdhury A, Giri R (2011) An improved differential evolution algorithm with fitness-based adaptation of the control parameters. Inf Sci 181:3749–65

Islam S, Das S, Ghosh S, Roy S, Suganthan PN (2012) An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans Syst Man Cybern B Cybern 42(2):482–500

Korošec P, Šilc J (2013) The continuous differential ant-stigmergy algorithm applied on real-parameter single objective optimization problems. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1658–1663

Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. In: Sixth international mendel conference on soft computing, pp 76–83

Liang JJ, Qin BY, Suganthan PN, Hernndez-Diaz AG (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Zhengzhou University/Nanyang Technological University, Zhengzhou, China/Singapore, Technical Report, p 201212

Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696

Mohamed AW (2015) An improved differential evolution algorithm with triangular mutation for global numerical optimization. Comput Ind Eng 85:359–375

Mohamed AW, Sabry HZ (2012) Constrained optimization based on modified differential evolution algorithm. Inf Sci 194:171–208

Mohamed AW, Sabry HZ, Farhat A (2011) Advanced differential evolution algorithm for global numerical optimization. In: Proceedings of the IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE'11), Penang, Malaysia, pp 156–161

Mohamed AW, Sabry HZ, Khorshid M (2012) An alternative differential evolution algorithm for global optimization. J Adv Res 3(2):149–165

Nepomuceno FV, Engelbrecht AP (2013) A self-adaptive heterogeneous PSO for real-parameter optimization. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 361–368

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 200812(1):107–25

Padhye N, Mittal P, Deb K (2013) Differential evolution: performances and analyses. In: Proceedings of the IEEE congress on evolutionary computation, pp 1960–1967

Pan QK, Suganthan PN, Wang L, Gao L, Mallipeddi R (2011) A differential evolution algorithm with self-adapting strategy and control parameters. Comput Oper Res 38:394–408

Papa G, Šilc J (2013) The parameter-less evolutionary search for real-parameter single objective optimization. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1131–1137

Paul S, Das S (2015) Simultaneous feature selection and weighting–an evolutionary multi-objective optimization approach. Pattern Recognit Lett 65:51–59

Poikolainen I, Neri F (2013) Differential evolution with concurrent fitness based local search. In: IEEE congress on evolutionary computation, pp 384–391

Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolut Comput 13(2):398–417

Qin AK, Li X, Pan H, Xia S (2013) Investigation of self-adaptive differential evolution on the CEC-2013 real-parameter single-objective optimization Testbed. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1107–1114

Ronkkonen J, Kukkonen S, Price KV (2005) Real parameter optimization with differential evolution. In: Proceedings of the IEEE congress on evolutionary computation (CEC-2005), vol 1. IEEE Press, Piscataway, pp 506–513

Sarkar S, Das S, Chaudhuri SS (2015) A multilevel color image thresholding scheme based on minimum cross entropy and differential evolution. Pattern Recognit Lett 54:27–35

Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI. http://icsi.berkeley.edu/storn/litera.html

Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–59

Tanabe R, Fukunaga A (2013) Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1952–1959

Tvrdík J, Poláková R (2013) Competitive differential evolution applied to CEC 2013 problems. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1651–1657

Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evolut Comput 15(1):55–66

Weber M, Neri F, Tirronen V (2011) A study on scale factor in distributed differential evolution. Inf Sci 181:2488–2511

Wu GH, Mallipeddi R, Suganthan PN, Wang R, Chen H (2015) Differential evolution with multi population based ensemble of mutation strategies. Inf Sci 329:329–345

Zamuda A, Brest J (2015) Self-adaptive control parameters: randomization frequency and propagations in differential evolution. Swarm Evolut Comput 25:72–99

Zamuda A, Brest J, Mezura-Montes E (2013) Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization. In: Proceedings of the IEEE congress on evolutionary computation, México, pp 1925–1931

Zhai S, Jiang T (2015) A new sense-through-foliage target recognition method based on hybrid differential evolution and self-adaptive particle swarm optimization-based support vector machine. Neurocomputing 149:573–584

Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evolut Comput 13(5):945–958

Zhang X, Chen W, Dai C, Cai W (2010) Dynamic multi-group self-adaptive differential evolution algorithm for reactive power optimization. Int J Electr Power 32:351–357