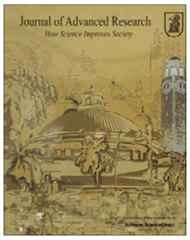
Cairo University

Journal of Advanced Research



ORIGINAL ARTICLE

An alternative differential evolution algorithm for global optimization

Ali W. Mohamed [a,*], Hegazy Z. Sabry [b], Motaz Khorshid [c]

[a] *Department of Operations Research, Institute of Statistical Studies and Research, Cairo University, Giza, Egypt*
[b] *Department of Mathematical Statistics, Institute of Statistical Studies and Research, Cairo University, Giza, Egypt*
[c] *Department of Decision Support, Faculty of Computers and Information, Cairo University, Giza, Egypt*

Received 20 November 2010; revised 12 June 2011; accepted 21 June 2011
Available online 23 July 2011

KEYWORDS

Differential evolution;
Directed mutation;
Global optimization;
Modified BGA mutation;
Dynamic non-linear crossover

Abstract The purpose of this paper is to present a new and an alternative differential evolution (ADE) algorithm for solving unconstrained global optimization problems. In the new algorithm, a new directed mutation rule is introduced based on the weighted difference vector between the best and the worst individuals of a particular generation. The mutation rule is combined with the basic mutation strategy through a linear decreasing probability rule. This modification is shown to enhance the local search ability of the basic DE and to increase the convergence rate. Two new scaling factors are introduced as uniform random variables to improve the diversity of the population and to bias the search direction. Additionally, a dynamic non-linear increased crossover probability scheme is utilized to balance the global exploration and local exploitation. Furthermore, a random mutation scheme and a modified Breeder Genetic Algorithm (BGA) mutation scheme are merged to avoid stagnation and/or premature convergence. Numerical experiments and comparisons on a set of well-known high dimensional benchmark functions indicate that the improved algorithm outperforms and is superior to other existing algorithms in terms of final solution quality, success rate, convergence rate, and robustness.

Introduction

For several decades, global optimization has received wide attention from researchers, mathematicians as well as professionals in the field of Operations Research (OR) and Computer Science (CS). Nevertheless, global optimization problems, in almost fields of research and real-world applications, have many different challenging features such as high nonlinearity, non-convexity, non-continuity, non-differentiability, and/or multimodality. Therefore, classical nonlinear optimization techniques have difficulties or have always failed in dealing with complex high dimensional global optimization problems. As a

* Corresponding author. Tel.: +20 105157657.
E-mail address: aliwagdy@gmail.com (A.W. Mohamed).

Peer review under responsibility of Cairo University.
doi:10.1016/j.jare.2011.06.004

result, the challenges mentioned above have motivated researchers to design and improve many kinds of efficient, effective and robust algorithms that can reach a high quality solution with low computational cost and high convergence performance. In the past few years, the interaction between computer science and operations research has become very important in order to develop intelligent optimization techniques that can deal with such complex problems. Consequently, Evolutionary Algorithms (EAs) represent the common area where the two fields of OR and CS interact. EAs have been proposed to meet the global optimization challenges [1]. The structure of (EA) has been inspired from the mechanisms of natural evolution. Generally, the process of (EAs) is based on the exploration and the exploitation of the search space through selection and reproduction operators [2]. Differential Evolution (DE) is a stochastic population-based search method, proposed by Storn and Price [3]. DE is considered the most recent EAs for solving real-parameter optimization problems [4]. DE has many advantages including simplicity of implementation, is reliable, robust, and in general is considered an effective global optimization algorithm [5]. Therefore, it has been used in many real-world applications [6], such as in the chemical engineering field [7], machine intelligence applications [8], pattern recognition studies [9], signal processing implementations [10], and in the area of mechanical engineering design [11]. In a recent study [12], DE was evaluated and compared with the Particle Swarm Optimization (PSO) technique and other EAs in order to test its capability as a global search technique. The comparison was based on 34 benchmark problems and DE outperformed other recent algorithms. DE, nevertheless, also has the shortcomings of all other intelligent techniques. Firstly, while the global exploration ability of DE is considered adequate, its local exploitation ability is regarded weak and its convergence velocity is too low [13]. Secondly, DE suffers from the problem of premature convergence, where the search process may be trapped in local optima in multimodal objective function and losing its diversity [6]. Additionally, it also suffers from the stagnation problem, where the search process may occasionally stop proceeding toward the global optimum even though the population has not converged to a local optimum or any other point [14]. Moreover, like other evolutionary algorithms, DE performance decreases as search space dimensionality increases [6]. Finally, DE is sensitive to the choice of the control parameters and it is difficult to adjust them for different problems [15]. Therefore, in order to improve the global performance of basic DE, this research uses a new directed mutation rule to enhance the local exploitation ability and to improve the convergence rate of the algorithm. Two scaling factors are also introduced as uniform random variables for each trial vector instead of keeping them as a constant to cover the whole search space. This will advance the exploration ability as well as bias the search in the direction of the best vector through generations. Furthermore, a dynamic non-linear increased crossover probability scheme is proposed to balance exploration and exploitation abilities. In order to avoid the stagnation and the premature convergence issues through generations, modified BGA mutation and random mutation are embedded into the proposed ADE algorithm. Numerical experiments and comparisons conducted in this research effort on a set of well-known high dimensional benchmark functions indicate that the proposed alternative differential evolution (ADE) algorithm is superior and competitive to other existing recent memetic, hybrid, self-adaptive and basic DE algorithms particularly in the case of high dimensional complex optimization problems. The remainder of this paper is organized as follows. The next section reviews the related work. Then, the standard DE algorithm and the proposed ADE algorithm are introduced. Next, the experimental results are discussed and the Final section concludes the paper.

Related work

Indeed, due to the above drawbacks, many researchers have done several attempts to overcome these problems and to improve the overall performance of the DE algorithm. The choice of DE's control variables has been discussed by Storn and Price [3] who suggested a reasonable choice for NP (population size) between 5D and 10D (D being the dimensionality of the problem), and 0.5 as a good initial value of F (mutation scaling factor). The effective value of F usually lies in the range between 0.4 and 1. As for the CR (crossover rate), an initial good choice of CR = 0.1; however, since a large CR often speeds convergence, it is appropriate to first try CR as 0.9 or 1 in order to check if a quick solution is possible. After many experimental analysis, Gämperle et al. [16] recommended that a good choice for NP is between 3D and 8D, with $F = 0.6$ and CR lies in [0.3,0.9]. On the contrary, Rönkkönen et al. [17] concluded that $F = 0.9$ is a good compromise between convergence speed and convergence probability. 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 between 0 and 0.2 when the objective function is separable. Due to the contradiction claims that can be seen from the literature, some techniques have been designed to adjust control parameters in a self-adaptive or adaptive manner instead of using manual tuning. A Fuzzy Adaptive Differential Evolution (FADE) algorithm was proposed by Liu and Lampinen [18]. They introduced fuzzy logic controllers to adjust crossover and mutation rates. Numerical experiments and comparisons on a set of well known benchmark functions showed that the FADE Algorithm outperformed basic DE algorithm. Likewise, Brest et al. [19] described an efficient technique for self-adapting control parameter settings. The results showed that their algorithm is better than, or at least comparable to, the standard DE algorithm, (FADE) algorithm and other evolutionary algorithms from the literature when considering the quality of the solutions obtained. In the same context, Salman et al. [20] proposed a Self-adaptive Differential Evolution (SDE) algorithm. The experiments conducted showed that SDE generally outperformed DE algorithms and other evolutionary algorithms. On the other hand, hybridization with other heuristics or local different algorithms is considered as the new direction of development and improvement. Noman and Iba [13] recently proposed a new memetic algorithm (DEahcSPX), a hybrid of crossover-based adaptive local search procedure and the standard DE algorithm. They also investigated the effect of the control parameter settings in the proposed memetic algorithm and realized that the optimal values for control parameters are $F = 0.9$, CR = 0.9 and NP = D. The presented experimental results demonstrated that (DEahcSPX) performs better, or at least comparable to classical DE algorithm, local search heuristics and other well-known evolutionary algorithms. Similarly, Xu et al. [21] suggested the NM-DE algorithm, a hybrid of Nelder–Mead simplex search method and basic DE algorithm. The comparative results showed that the proposed new hybrid algorithm outperforms some existing algorithms including hybrid DE and hybrid NM algorithms in terms of solution quality, convergence rate and robustness. Additionally, the stochastic properties of chaotic systems are used to spread the individuals in the search spaces as much as possible [22]. Moreover, the pattern search is employed to speed up the local exploitation. Numerical experiments on benchmark problems demonstrate that this new method achieved an improved success rate and a final solution with less computational effort. Practically, from the literature, it can be observed that the main modifications, improvements and developments on DE focus on adjusting control parameters in self-adaptive manner and/or hybridization with other local search techniques. However, a few enhancements have been implemented to modify the standard mutation strategies or to propose new mutation rules so as to enhance the local search ability of DE or to overcome the problems of stagnation or premature convergence [6,23,24]. As a result, proposing new mutations and adjusting control parameters are still an open challenge direction of research.

Methodology

The differential evolution (DE) algorithm

A bound constrained global optimization problem can be defined as follows [21]:

$$\min f(X),\quad X=[x_1,\ldots,x_n];\quad \text{S.t. } x_j\in[a_j,b_j];\quad j=1,2,\ldots n, \tag{1}$$

where f is the objective function, X is the decision vector consisting of n variables, and a_j and b_j are the lower and upper bounds for each decision variable, respectively. Virtually, there are several variants of DE [3]. In this paper, we use the scheme which can be classified using the notation as DE/rand/1/bin strategy [3,19]. This strategy is most often used in practice. A set of D optimization parameters is called an individual, which is represented by a D-dimensional parameter vector. A population consists of NP parameter vectors x_i^G, $i = 1, 2, \ldots, NP$. G denotes one generation. NP is the number of members in a population. It is not changed during the evolution process. The initial population is chosen randomly with uniform distribution in the search space. DE has three operators: mutation, crossover and selection. The crucial idea behind DE is a scheme for generating trial vectors. Mutation and crossover operators are used to generate trial vectors, and the selection operator then determines which of the vectors will survive into the next generation [19].

Initialization

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

$$x_{ij}^0 = a_j + \text{rand}_j\cdot(b_j-a_j) \tag{2}$$

where rand_j denotes a uniformly distributed number between [0,1], generating a new value for each decision parameter. a_j and b_j are the lower and upper bounds for the jth decision parameter, respectively.

Mutation

For each target vector x_i^G, a mutant vector v_i^{G+1} is generated according to the following:

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

with randomly chosen indices and $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$.

Note that these indices must be different from each other and from the running index i so that NP must be at least four. 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 [4], the range of F is in [0,2]. If a component of a mutant vector goes off the search space, then the value of this component is generated anew using (2).

Crossover

The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector u_i^{G+1}.

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, \text{rand}(j)\leqslant CR \text{ or } j=\text{rand}\,n(i),\\ x_{ij}^G, \text{rand}(j) > CR \text{ and } j\neq\text{rand}\,n(i), \end{cases} \tag{4}$$

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

Selection

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

$$x_i^{G+1} = \begin{cases} u_i^{G+1}, & f(u_i^{G+1}) < f(x_i^G),\\ x_i^G, & f(u_i^{G+1}) \geqslant f(x_i^G). \end{cases} \tag{5}$$

An alternative differential evolution (ADE) algorithm

All evolutionary algorithms, including DE, are stochastic population-based search methods. Accordingly, there is no guarantee to reach the global optimal solution all the times. Nonetheless, adjusting control parameters such as the scaling factor, the crossover rate and the population size, alongside developing an appropriate mutation scheme, 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. Therefore, in this paper, four modifications are introduced in order to significantly enhance the overall performance of the standard DE algorithm.

Modification of mutations

A success of the population-based search algorithms is based on balancing two contradictory aspects: global exploration and local exploitation [6]. Moreover, the mutation scheme plays a vital role in the DE search capability and the convergence rate. However, even though the DE algorithm has good global exploration ability, it suffers from weak local exploitation ability as well as its convergence velocity is still too low as the region of the optimal solution is reached [23]. Obviously, from the mutation equation (3), it can be observed that three vectors are chosen at random for mutation and the base vector is then selected at random among the three. Consequently, the basic mutation strategy DE/rand/1/bin is able to maintain population diversity and global search capability, but it slows down the convergence of DE algorithms. Hence, in order to enhance the local search ability and to accelerate the convergence of DE techniques, a new directed mutation scheme is proposed based on the weighted difference vector between the best and the worst individual at a particular generation. The modified mutation scheme is as follows:

$$v_i^{G+1} = x_r^G + F_l \cdot (x_b^G - x_w^G) \tag{6}$$

where x_r^G is a random chosen vector and x_b^G and x_w^G are the best and worst vectors in the entire population, respectively. This modification is intended to keep the random base vector $x_{r_1}^G$ in the mutation equation (3) as it is and the remaining two vectors are replaced by the best and worst vectors in the entire population to yield the difference vector. In fact, the global solution can be easily reached if all vectors follow the same direction of the best vector besides they also follow the opposite direction of the worst vector. Thus, the proposed directed mutation favors exploitation since all vectors of population are biased by the same direction but are perturbed by the different weights as discussed later on. As a result, the new mutation rule has better local search ability and faster convergence rate. It is worth mentioning that the proposed mutation is inspired from nature and human behavior. Briefly, although all the people in a society are different in many ways such as aims, cultures, thoughts and so on, all of them try to significantly improve themselves by following the same direction of the other successful and superior people and similarly they tend to avoid the direction of failure in whatever field by competition and/or co-operation with others. The new mutation strategy is embedded into the DE algorithm and it is combined with the basic mutation strategy DE/rand/1/bin through a linear decreasing probability rule as follows:

If

$$\left(u(0,1) \geqslant \left(1 - \frac{G}{GEN}\right)\right) \tag{7}$$

Then

$$v_i^{G+1} = x_r^G + F_l \cdot (x_b^G - x_w^G) \tag{8}$$

Else

$$v_i^{G+1} = x_{r1}^G + F_g \cdot (x_{r_2}^G - x_{r_3}^G) \tag{9}$$

where F_l and F_g are two uniform random variables, $u(0, 1)$ returns a real number between 0 and 1 with uniform random probability distribution and G is the current generation number, and GEN is the maximum number of generations. From the above 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 $(1 - \frac{G}{GEN})$, then the basic mutation is applied. Otherwise, the proposed one is performed. Of course, it can be seen that, from Eq. (7), the probability of using one of the two mutations is a function of the generation number, so $(1 - \frac{G}{GEN})$ can be gradually changed form 1 to 0 in order to favor, balance, and combine the global search capability with local search tendency.

The strength and efficiency of the above scheme is based on the fact that, at the beginning of the search, two mutation rules are applied but the probability of the basic mutation rule to be used is greater than the probability of the new strategy. So, it favors exploration. Then, in the middle of the search, through generations, the two rules are approximately used with the same probability. Accordingly, it balances the search direction. Later, two mutation rules are still applied but the probability of the proposed mutation to be performed is greater than the probability of using the basic one. Finally, it enhances exploitation. Therefore, at any particular generation, both exploration and exploitation aspects are done in parallel. On the other hand, although merging a local mutation scheme into a DE algorithm can enhance the local search ability and speed up the convergence velocity of the algorithm, it may lead to a premature convergence and/or to get stagnant at any point of the search space especially with high dimensional problems [6,24]. For this reason, random mutation and a modified BGA mutation are merged and incorporated into the DE algorithm to avoid both cases at early or late stages of the search process. Generally, in order to perform random mutation on a chosen vector x_i at a particular generation, a uniform random integer number j_{rand} between $[1, D]$ is first generated and than a real number between $(b_j - a_j)$ is calculated. Then, the j_{rand} value from the chosen vector is replaced by the new real number to form a new vector x'. The random mutation can be described as follows.

$$x'_j = \begin{cases} a_j + \text{rand}_j(b_j - a_j) & j = j_{\text{rand}}, \\ x_j & \text{otherwise}, \end{cases} \quad j = 1, \ldots, D \tag{10}$$

Therefore, it can be deduced from the above equation that random mutation increases the diversity of the DE algorithm as well decreases the risk of plunging into local point or any other point in the search space. In order to perform BGA mutation, as discussed Mühlenbein and Schlierkamp Voosen [25], on a chosen vector x_i at a particular generation, a uniform random integer number j_{rand} between $[1, D]$ is first generated and then a real number between $0.1 \cdot (b_j - a_j) \cdot \alpha$ is calculated. Then, the j_{rand} value from the chosen vector is replaced by the new real number to form a new vector x'_i. The BGA mutation can be described as follows.

$$x'_j = \begin{cases} x_j + 0.1 \cdot (b_j - a_j) \cdot \alpha & j = j_{\text{rand}}, \\ x_j & \text{otherwise}, \end{cases} \quad j = 1, \ldots, D \tag{11}$$

The + or − sign is chosen with probability 0.5. α is computed from a distribution which prefers small values. This is realized as follows:

$$\alpha = \sum_{k=0}^{15} \alpha_k \cdot 2^{-k}, \quad \alpha_k \in \{0, 1\} \tag{12}$$

Before mutation, we set $\alpha_i = 0$. Afterward, each α_i is mutated to 1 with probability $p_\alpha = 1/16$. Only α_k contributes to the sum as in Eq. (12). On average, there will be just one α_k with value 1, say α_m, then α is given by $\alpha = 2^{-m}$. In this paper, the modified BGA mutation is given as follows:

$$x'_j = \begin{cases} x_j \pm \text{rand}_j \cdot (b_j - a_j) \cdot \alpha & j = j_{\text{rand}}, \\ x_j & \text{otherwise}, \end{cases} \quad j = 1, \ldots, D \tag{13}$$

where the factor of 0.1 in Eq. (11) is replaced by a uniform random number in (0, 1], because the constant setting of $0.1 \cdot (b_j - a_j)$ is not suitable. However, the probabilistic setting of $\text{rand}_j \cdot (b_j - a_j)$ enhances the local search capability with small random numbers besides it still has an ability to jump to another point in the search space with large random numbers so as to increase the diversity of the population. Practically, no vector is subject to both mutations in the same generation, and only one of the above two mutations can be applied with the probability of 0.5. However, both mutations can be performed in the same generation with two different vectors. Therefore, at any particular generation, the proposed algorithm has the chance to improve the exploration and exploitation abilities. Furthermore, in order to avoid stagnation as well as premature convergence and to maintain the convergence rate, a new mechanism for each solution vector is proposed that satisfies the following condition: if the difference between two successive objective function values for any vector except the best one at any generation is less than or equal a predetermined level δ for predetermined allowable number of generations K, then one of the two mutations is applied with equal probability of (0.5). This procedure can be expressed as follows:

$$\text{If } |f_c - f_p| \leqslant \delta \text{ for } K \text{ generations, then} \tag{14}$$

If $(u(0,1) \geqslant 0.5)$, then

$$x'_j = \begin{cases} a_j + \text{rand}_j \cdot (b_j - a_j) & j = j_{\text{rand}}, \\ x_j & \text{otherwise}, \end{cases}$$

$j = 1, \ldots, D$ (Random mutation)

Else

$$x'_j = \begin{cases} x_j \pm \text{rand}_j \cdot (b_j - a_j) \cdot \alpha & j = j_{\text{rand}}, \\ x_j & \text{otherwise}, \end{cases}$$

$j = 1, \ldots, D$(Modified BGA mutation)

where f_c and f_p indicate current and previous objective function values, respectively.After many experiments, in order to make a comparison with other algorithms with 30 dimensions, we observed that δ = E−07 and K = 75 generations are the best settings for these two parameters over all benchmark problems and these values seem to maintain the convergence rate as well as avoid stagnation and/or premature convergence in case they occur. Indeed, these parameters were set to their mean values as we observed that if δ and K are approximately less than or equal to E0−5 and 50, respectively then the convergence rate deteriorated for some functions. On the other hand, if δ and K are nearly greater than or equal E−10 and 100, respectively, then it could be stagnated. For this reason, the mean values of E−07 for δ and 75 for K were selected for all dimensions as default values. In this paper, these settings were fixed for all dimensions without tuning them to their optimal values that may attain good solutions better than the current results and improve the performance of the algorithm over all the benchmark problems.

Modification of scaling factor

In the mutation Eq. (3), the constant of differentiation F is a scaling factor of the difference vector. It is an important parameter that controls the evolving rate of the population. In the original DE algorithm [4], the constant of differentiation F was chosen to be a value in [0, 2]. The value of F has a considerable influence on exploration: small values of F lead to premature convergence, and high values slow down the search [26]. However, to the best of our knowledge, there is no optimal value of F that has been derived based on theoretical and/or systematic study using all complex benchmark problems. In this paper, two scaling factors F_l and F_g are proposed for the two different mutation rules, where F_l and F_g indicate scaling factor for the local mutation scheme and the scaling factor for global mutation scheme, respectively. For the difference vector in the mutation equation (8), we can see that it is a directed difference vector from the worst to the best vectors in the entire population. Hence, F_l must be a positive value in order to bias the search direction for all trial vectors in the same direction. Therefore, F_l is introduced as a uniform random variable in (0, 1). Instead of keeping F constant during the search process, F_l is set as a random variable for each trial vector so as to perturb the random base vector by different directed weights. Therefore, the new directed mutation resembles the concept of gradient as the difference vector is oriented from the worst to the best vectors [26]. On the other hand, for the difference vector in the mutation equation (9), we can see that it is a pure random difference as the objective function values are not used. Accordingly, the best direction that can lead to good exploration is unknown. Therefore, in order to advance the exploration and to cover the whole search space F_g is introduced as a uniform random variable in the interval $(-1, 0) \cup (0, 1)$, unlike keeping it as a constant in the range [0, 2] as recommended by Feoktistov [26]. Therefore, the new enlarger random variable can perturb the random base vector by different random weights with opposite directions. Hence, F_g is set to be random for each trial vector. As a result, the proposed evolutionary algorithm is still a random search that can enhance the global exploration performance as well as ensure the local search ability. The illustration of the process of the basic mutation rule, the new directed mutation rule and modified basic mutation rule with the constant scaling factor and the two new scaling factors are illustrated in Fig. 1(a)–(c). From this figure it can be clearly noticed that $_i$ is the mutation vector generated for individual x_i using the associated mutation constant scaling factor F in (a). However, $_i$ is the new scaled directed mutation vector generated for individual x_i using the associated mutation factor F_l in (b). Moreover, $_i$ is the mutation vector generated for individual x_i using the associated mutation factor F_g.

Modification of the crossover rate

The crossover operator, as in Eq. (4), shows that the constant crossover (CR) reflects the probability with which the trial individual inherits the actual individual's genes [26]. The constant crossover (CR) practically controls the diversity of the population. If the CR value is relatively high, this will increase the population diversity and improve the convergence speed. Nevertheless, the convergence rate may decrease and/or the population may prematurely converge. On the other hand,

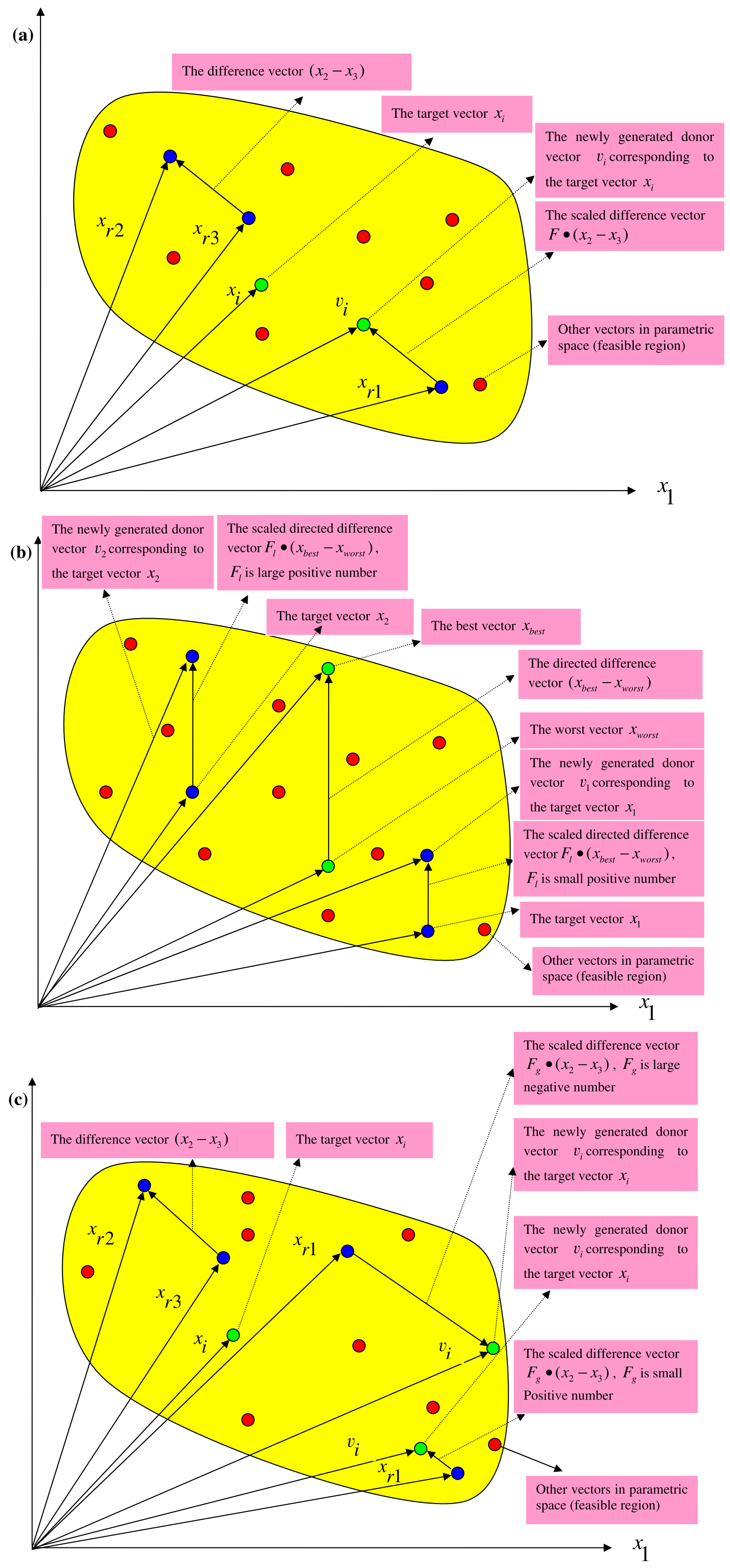

(a)
The difference vector $(x_2 - x_3)$
The target vector x_i
The newly generated donor vector v_i corresponding to the target vector x_i
The scaled difference vector $F \bullet (x_2 - x_3)$
Other vectors in parametric space (feasible region)
x_{r2}
x_{r3}
x_i
v_i
x_{r1}
x_1
(b)
The newly generated donor vector v_2 corresponding to the target vector x_2
The scaled directed difference vector $F_l \bullet (x_{best} - x_{worst})$, F_l is large positive number
The target vector x_2
The best vector x_{best}
The directed difference vector $(x_{best} - x_{worst})$
The worst vector x_{worst}
The newly generated donor vector v_1 corresponding to the target vector x_1
The scaled directed difference vector $F_l \bullet (x_{best} - x_{worst})$, F_l is small positive number
The target vector x_1
Other vectors in parametric space (feasible region)
x_1
(c)
The difference vector $(x_2 - x_3)$
The target vector x_i
The scaled difference vector $F_g \bullet (x_2 - x_3)$, F_g is large negative number
The newly generated donor vector v_i corresponding to the target vector x_i
The newly generated donor vector v_i corresponding to the target vector x_i
The scaled difference vector $F_g \bullet (x_2 - x_3)$, F_g is small Positive number
Other vectors in parametric space (feasible region)
x_{r2}
x_{r3}
x_{r1}
x_i
v_i
v_i
x_{r1}
x_1

Fig. 1 (a) An illustration of the DE/rand/1/bin a basic DE mutation scheme in two-dimensional parametric space. (b) An illustration of the new directed mutation scheme in two-dimensional parametric space (local exploitation). (c) An illustration of the modified DE/rand/1/bin basic DE mutation scheme in two-dimensional parametric space (global exploration).

small values of CR increase the possibility of stagnation and slow down the search process. Additionally, at the early stage of the search, the diversity of the population is large because the vectors in the population are completely different from each other and the variance of the whole population is large. Therefore, the CR must take a small value in order to avoid the exceeding level of diversity that may result in premature convergence and slow convergence rate. Then, through generations, the variance of the population will decrease as the vectors in the population become similar. Thus, in order to advance diversity and increase the convergence speed, the CR must be a large value. Based on the above analysis and discussion, and in order to balance between the diversity and the convergence rate or between global exploration ability and local exploitation tendency, a dynamic non-linear increased crossover probability scheme is proposed as follows:

$$CR = CR_{\max} + (CR_{\min} - CR_{\max}) \cdot (1 - G/GEN)^k \tag{16}$$

where G is the current generation number, GEN is the maximum number of generations, $CR_{\min}$ and $CR_{\max}$ denote the minimum and maximum value of the CR, respectively, and k is a positive number. The optimal settings for these parameters are $CR_{\min} = 0.1$, $CR_{\max} = 0.8$ and $k = 4$. The algorithm starts at $G = 0$ with $CR_{\min} = 0.1$ but as G increases toward GEN, the CR increases to reach $CR_{\max} = 0.8$. As can be seen from Eq. (16), $CR_{\min} = 0.1$ is considered as a good initial rate in order to avoid high level of diversity in the early stage as discussed earlier and in Storn and Price [4]. Additionally, $CR_{\max} = 0.8$ is the maximum value of crossover that can balance between exploration and exploitation. However, beyond this value, mutation vector $_i^{G+1}$ has more contribution to the trial vector u_i^{G+1}. Consequently, the target vector x_i^G is destroyed greatly and the individual structure with better function values is destroyed rapidly. On the other hand, k balances the cross over rate which results in changing the CR from a small value to a large value in a dramatic curve. k was set to its mean value as it was observed that if it is approximately less than or equal to 1 or 2 then the diversity of the population deteriorated for some functions and it might have caused stagnation. On the other hand, if it is nearly greater than 6 or 7 it could cause premature convergence as the diversity sharply increases. The mean value of 4 was thus selected for dimensions 30 with all benchmark problems and is also fixed for all dimensions as the default value.

Results and discussions

In order to evaluate the performance and show the efficiency and superiority of the proposed algorithm, 10 well-known benchmark problems are used. The definition, the range of the search space, and the global minimum of each function are presented in Appendix 1 [13]. Furthermore, to evaluate and compare the proposed ADE algorithm with the recent differential evolution algorithms, the proposed ADE was compared with Basic DE and memetic DEahcSPX algorithm proposed by Noman and Iba [13], and the recent hybrid NM-DE algorithm proposed by Xu et al. [21]. Secondly, the proposed ADE was tested and compared with the recent memetic DEahcSPX algorithm and Basic DE against the growth of dimensionality. Thirdly, the performance of the proposed ADE algorithm was studied by comparing it with other memetic algorithms proposed by Noman and Iba [13]. Finally, the proposed ADE algorithm was compared with two well-known self-adaptive evolutionary algorithms, namely CEP and FEP proposed by Yao et al. [27] and with the recent self-adaptive jDE and SDE1 algorithms proposed by Brest et al. [19] and Salman et al. [20], respectively, as well as with another hybrid CPDE1 algorithm proposed by Wang and Zhang [22]. The best results are marked in bold for all problems. The experiments were carried out on an Intel Pentium core 2 due processor 2200 MHz and 2 GB-RAM. The algorithms were coded and realized in Matlab language using Matlab version 8. The description of the ADE algorithm is demonstrated in Fig. 2. These various algorithms are listed in Table 1.

Comparison of ADE with DEahcSPX, basic DE and NM-DE algorithms

In order to make a fair comparison for evaluating the performance of the algorithms, the performance measures and experimental setup [13,21] were used. The comparison was performed on the benchmark problems, listed in Appendix 1, at dimension $D = 30$, where D is the dimension of the problem. The maximum number of function evaluations was $10000 \times D$. For each problem, all of the above algorithms are independently run 50 times. The population size NP was set to D ($NP = 30$). Moreover, an accuracy level ε is set as 1.0E−06. That is, a test is considered as a successful run if the deviation between the obtained function value by the algorithm and the theoretical optimal value is less than the accuracy level [21]. For all benchmark problems at dimension $D = 30$, the resulted average function values and the standard deviation values of ADE, basic DE, DEahcSPX and NM-DE algorithms are listed in Table 2(a). Furthermore, the average function evaluation times and the time of successful run (data within parenthesis) of these algorithms are presented in Table 2(b). Finally, Fig. 3 presents the convergence characteristics of ADE in terms of the average fitness values of the best vector found during generations for selected benchmark problems. From Table 2(a), it is clear that the proposed ADE algorithm is superior to all other competitor algorithms in terms of average values and standard deviation. Furthermore, the results showed that ADE algorithm outperformed the basic DE algorithm in all functions. Moreover, it also outperformed DEahcSPX algorithm in all functions except for Ackley and Salomon functions (they are approximately the same). Additionally, the ADE algorithm outperformed the NM-DE algorithm in all functions except for the Sphere function. It is worth mentioning that the ADE algorithm considerably improves the final solution quality and it is extremely robust since it has a small standard deviation on all functions. From Table 2(b), it can be observed that the ADE algorithm costs much less computational effort than the basic DE and DEahcSPX algorithms while the ADE implementation requires more computational effort than NM-DE algorithm. Therefore, as a lower number of function evaluations corresponds to a faster convergence [6], the NM-DE algorithm is the fastest one among all competitor algorithms. However, it clearly suffered from premature convergence, since it absolutely did not achieve the accuracy level in all runs with Rastrigin, Schwefel, Salomon and Whitley functions. Additionally, the time of successful runs of the NM-DE and DEahcSPX algorithms was

```
01. Begin
02.   G=0
03.   Create a random initial population $\overset{n_G}{x_i}$ $\forall i, i=1,\dots,NP$
04.   Evaluate $f(\overset{n_G}{x_i})$ $\forall i, i=1,\dots,NP$
05.   Determine $x_{best}^{G}$ and $x_{worst}^{G}$ based on $f(\overset{n_G}{x_i})$, $i=1,\dots,NP$
06.   For G=1 to GEN Do
07.     CR=0.8+(0.1- 0.8)·(1-G/GEN)^4
08.       For  i=1 to NP Do
09.         If ( rand[0,1] >= ( 1-G/GEN)) Then (Use New Directed Mutation Scheme)
10.               Select randomly $r1 \neq best \neq worst \neq i \in [1, NP]$
11.               $F_l$ =rand(0,1)
12.               $j_{rand}$= randint(1,D)
13.           For j=1 to D Do
14.               If (rand_j[0,1]< CR or j= $j_{rand}$) Then
15.                 $u_{ij}^{G+1} = x_{r1}^{G} + F_l \cdot (x_{best}^{G} - x_{worst}^{G})$
16.               Else
17.                 $u_{ij}^{G+1} = x_{ij}^{G}$
18.               End If
19.           End For
20.         Else (Use Modified Basic Mutation Scheme)
21.               Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$
22.               $F_g$ =rand((-1,0) $\hbar$ (0,1))
23.               $j_{rand}$= randint(1,D)
24.             For j=1 to D Do
25.                 If (rand_j[0,1]< CR or j= $j_{rand}$) Then
26.                   $u_{ij}^{G+1} = x_{r1}^{G} + F_g \cdot (x_{r2}^{G} - x_{r3}^{G})$
27.                 Else
28.                   $u_{ij}^{G+1} = x_{ij}^{G}$
29.                 End If
30.           End For
31.         End If
32.           If $|f_{current} - f_{previous}| \le \delta$ for $K$ generations Then
33.               If $(rand(0,1) \ge 0.5)$ Then
34.                 $\overset{\hbar}{x}{}_{ij}^{G+1} = \begin{cases} a_j + rand_{ij} * (b_j - a_j) & j = j_{rand} \\ x_{ij} & \text{otherwise} \end{cases}$, $j = 1, \dots, D$   (Random Mutation)
35.               Else
                  $\overset{\hbar}{x}{}_{ij}^{G+1} = \begin{cases} x_{ij} + rand_{ij} * (b_j - a_j) * \alpha & j = j_{rand} \\ x_{ij} & \text{otherwise} \end{cases}$, $j = 1, \dots, D$   (Modified BGA mutation)
36.               End If
37.                 Determine $x_{best}^{G+1}$ and $x_{worst}^{G+1}$ based on $f(\overset{n_{G+1}}{x_i})$, $i=1,\dots,NP$
38.           End If
39.       End For
40.     G=G+1
41.   End For
42. End
```

Fig. 2 Description of ADE algorithm.

Table 1 The list of various algorithms in this paper.

Algorithm	Reference
An alternative differential evolution algorithm for global optimization (ADE)	This paper
Standard differential evolution (DE)	[13]
Accelerating differential evolution using an adaptive local search (DEahcSPX)	[13]
Enhancing differential evolution performance with local search for high dimensional function optimization (DEfirSPX)	[13]
Accelerating differential evolution using an adaptive local search (DExhcSPX)	[13]
An effective hybrid algorithm based on simplex search and differential evolution for global optimization(NM-DE)	[21]
Evolutionary programming made faster (FEP,CEP)	[27]
Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems (jDE)	[19]
Empirical analysis of self-adaptive differential evolution (SDE1)	[20]
Global optimization by an improved differential evolutionary algorithm (CPDE1)	[22]

Table 2 (a) Comparison of the ADE, basic DE, DEahcSPX and NM-DE Algorithms $D = 30$ and population size = 30. (b) Comparison of the ADE, basic DE, DEahcSPX and NM-DE Algorithms in terms of average evaluation times and time of successful runs $D = 30$ and population size = 30.

Function	DE [13]	DEahcSPX [13]	NM-DE [21]	ADE
(a)				
Sphere	5.73E−17 ± 2.03E−16	1.75E−31 ± 4.99E−31	**4.05E−299 ± 0.00E+00**	2.31E−149 ± 1.25E−148
Rosenbrock	5.20E+01 ± 8.56E+01	4.52E+00 ± 1.55E+01	9.34E+00 ± 9.44E+00	**4.27E−11 ± 2.26E−10**
Ackley	1.37E−09 ± 1.32E−09	**2.66E−15 ± 0.00E+00**	8.47E−15 ± 2.45E−15	**2.66E−15 ± 0.00E+00**
Griewank	2.66E−03 ± 5.73E−03	2.07E−03 ± 5.89E−03	8.87E−04 ± 6.73E−03	**0.00E+00 ± 0.00E+00**
Rastrigin	2.55E+01 ± 8.14E+00	2.14E+01 ± 1.23E+01	1.41E+01 ± 5.58E+00	**0.00E+00 ± 0.00E+00**
Schwefel	4.90E+02 ± 2.34E+02	4.70E+02 ± 2.96E+02	3.65E+03 ± 7.74E+02	**0.00E+00 ± 0.00E+00**
Salomon	2.52E−01 ± 4.78E−02	1.80E−01 ± 4.08E−02	1.11E+00 ± 1.91E−01	1.93E−01 ± 2.39E−02
Whitely	3.10E+02 ± 1.07E+02	3.06E+02 ± 1.10E+02	4.18E+02 ± 7.06E+01	**2.65E+01 ± 2.97E+01**
Penalized 1	4.56E−02 ± 1.31E−01	2.07E−02 ± 8.46E−02	8.29E−03 ± 2.84E−02	**1.58E−32 ± 7.30E−34**
Penalized 2	1.44E−01 ± 7.19E−01	1.71E−31 ± 5.35E−31	2.19E−04 ± 1.55E−03	**1.77E−32 ± 2.69E−32**
(b)				
Sphere	148650.8 **(50)**	87027.4 **(50)**	**8539.4 (50)**	15928.8 **(50)**
Rosenbrock	–	299913.0 (2)	**74124.9** (40)	189913.8 **(50)**
Ackley	215456.1 **(50)**	129211.6 **(50)**	**13574.7** (29)	22589.4 **(50)**
Griewank	190292.5 (38)	121579.2 (43)	**9270.2** (36)	16887.446809 **(50)**
Rastrigin	–	–	–	**62427 (50)**
Schwefel	–	–	–	**41545.6 (50)**
Salomon	–	–	–	–
Whitely	–	–	–	**82181.538462 (13)**
Penalized 1	160955.2 (43)	96149.0 (46)	**7634.3** (44)	14685.6 **(50)**
Penalized 2	156016.9 (48)	156016.9 **(50)**	**7996.1** (42)	16002 **(50)**

–: None of the algorithms achieved the desired accuracy level $\varepsilon < 10^{-6}$.

very close in other functions and they exhibited unstable performance with the predefined level of accuracy. Contrarily, The ADE algorithm achieved the accuracy level in all 50 runs with all functions except for Salomon and was the only algorithm that reached the accuracy level in all runs with Rastrigin and Schwefel problems as well as in many runs with Whitley function. Moreover, the number of successful runs was also greatest for the ADE algorithm over all functions. Thus, this indicates the higher robustness of the proposed algorithm as compared to other algorithms and also proves the capability in maintaining higher diversity with an improved convergence rate. Similarly, consider the convergence characteristics of selected functions presented in Fig. 3, it is clear that the convergence speed of the ADE algorithm is fast at the early stage of the optimization process for all functions with different shapes, complexity and dimensions. Furthermore, the convergence speed is dramatically decreased and its improvement is found to be significant at the middle and later stages of the optimization process especially with Sphere and Rosenbrock functions. Additionally, the convergent figure suggests that the ADE algorithm can reach the true global solution in all problems in a fewer number of generations less than the maximum predetermined number of generations. Therefore, the proposed ADE algorithm is proven to be an effective, powerful approach for solving unconstrained global optimization problems. In general, the mean fitness values obtained by the ADE algorithm show that it has the most significant and efficient exploration and exploitation capabilities. Therefore, it is concluded that the new CR rule besides the proposed two new scaling factors greatly balance the two processes. The ADE algorithm was able to also reach the global optimum and escape from local ones in all runs in almost all functions. This indicates the importance of the new directed mutation scheme as well as the random and modified BGA mutations in improving the searching process quality and their significance in advancing exploitation process. On the other hand, in order to investigate the sensitivity of all algorithms to population size, the effect of population size on the performance of algorithms is studied with the fixed total evaluation times (3.0E+05) [21]. The results were reported in Table 3. From this table, it can be concluded that as the population size increases, the performance of the basic DE and DEahcSPX algorithms rapidly deteriorates whereas the performance of NM-DE algorithm slightly decreases. Additionally, the results show that the proposed ADE algorithm outperformed the basic DE and DEahcSPX techniques in all functions by remarkable difference while it outperformed the NM-DE algorithm in most test functions, for various population sizes. The performance of the ADE algorithm shows relative deterioration with the growth of population size, which suggests that the ADE algorithm is more stable and robust on population size.

Scalability comparison of ADE with DEahcSPX and basic DE algorithms

The performance of most of the evolutionary algorithms deteriorates with the growth of dimensionality of the search space [6]. As a result, in order to test the performance of the ADE, DEahcSPX and basic DE algorithms, the scalability study was conducted. The benchmark functions were studied at $D = 10, 50, 100, 200$ dimensions. The population size was chosen as $NP = 30$ for $D = 10$ dimensions and for all other

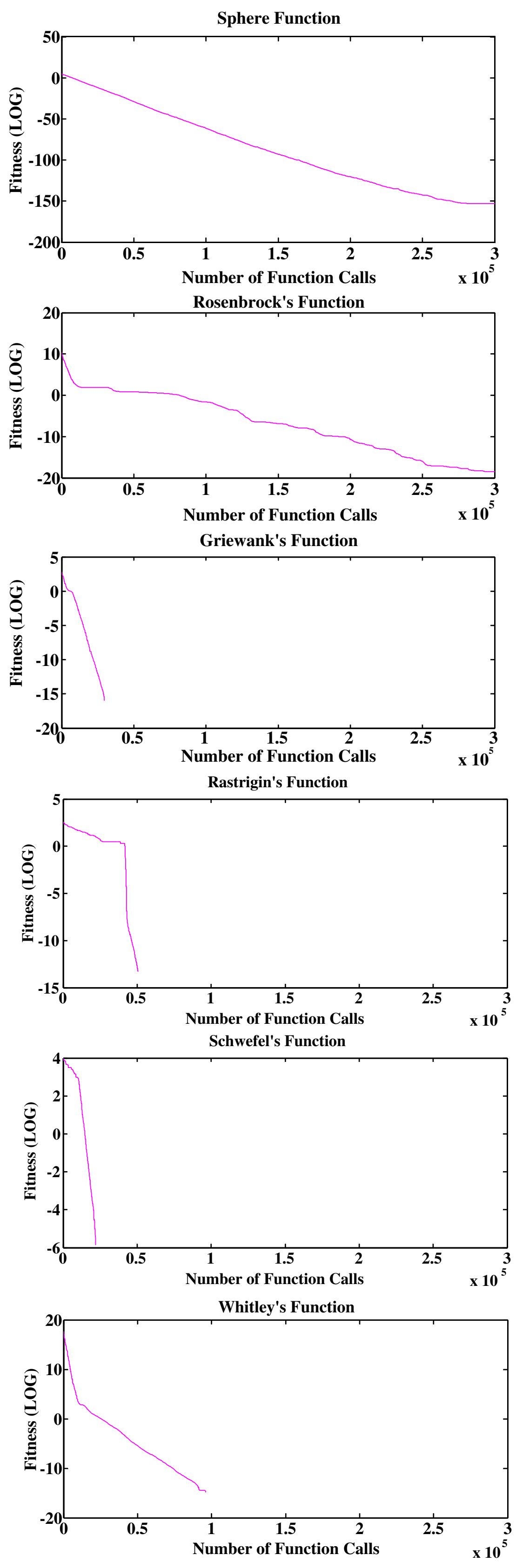


Fig. 3 Average best fitness curves of the ADE algorithm for selected benchmark functions for $D = 30$ and population size = 30.

dimensions, it was selected as $NP = D$ [13]. The resulted average function values and standard deviation using $10000 \times D$ are listed in Table 4(a). Convergence Figs. 4–7 for $D = 10$, 50, 100, 200 dimensions, respectively, present the convergence characteristics of the proposed ADE algorithm in terms of the average fitness values of the best vector found during generations for selected benchmark problems. For $D = 10$ dimensions, the average function evaluation times and the time of successful run (data within parenthesis) of these algorithms are presented in Table 4(b). Similarly, to the previous subsection, the performance of the basic DE and DEahcSPX algorithms shows completely deterioration with the growth of the dimensionality. From Table 4(a), it can be clearly concluded that the ADE algorithm outperformed the basic DE and DEahcSPX algorithms by a significant difference especially with 50, 100, and 200 dimensions and in all functions. Moreover, with these high dimensions, the ADE algorithm still could reach the global solution in most functions. As discussed earlier, the performance of the ADE algorithm slightly diminishes with the growth of the dimensionality, while still more stable and robust for solving problems with high dimensionality. Moreover, consider the convergence characteristics of selected functions presented in Figs. 4–7; it is clear that the proposed modifications play a vital role in improving the convergence speed for most problems in all dimensions. The ADE algorithm has still the ability to maintain its convergence rate, improve its diversity as well as advance its local tendency through a search process. Accordingly, it can be deduced that the superiority and efficiency of the ADE algorithm is due to the proposed modifications introduced in the previous sections. From Table 4(b), for $D = 10$ dimensions, it can be observed that the ADE algorithm reached the global solution in all runs in all functions except with the Salomon function and the time of successful runs was also greatest for the ADE algorithm over all functions. Moreover, the ADE implementation costs much less computational efforts than the basic DE and DEahcSPX algorithms, so ADE algorithm is the fastest one among all competitor algorithms.

Comparison of the ADE with DEfirSPX and DExhcSPX algorithms

The performance of the proposed ADE algorithm was also compared with two other memetic versions of the DE algorithm, as discussed in Noman and Iba [13]. The comparison was performed on the same benchmark problems at dimensions $D = 30$ and population size $NP = 30$. The average results of 50 independent runs are reported in Table 5(a). The average function evaluation times and the time of successful run (data within parenthesis) of these algorithms are presented in Table 5(b). The comparison shows the superiority of the ADE algorithm in terms of average values and standard deviation in all functions. Therefore, the minimum average and standard deviation values indicate that the proposed ADE algorithm is of better searching quality and robustness. Additionally, from Table 5(b), it can be observed that the ADE algorithm requires less computational effort than the other two algorithms, so it remained the fastest one besides it still has the greatest time of successful runs over all functions.

Table 3 Comparison of the ADE, basic DE, DEahcSPX and NM-DE algorithms $D = 30$ with different population size, after 3.0E + 05 function evaluation.

Function	DE [13]	DEahcSPX [13]	NM-DE [21]	ADE
Population size = 50				
Sphere	2.31E−02 ± 1.92E−02	6.03E−09 ± 6.86E−09	**8.46E−307 ± 0.00E+00**	1.45E−92 ± 6.11E−92
Rosenbrock	3.07E+02 ± 4.81E+02	4.98E+01 ± 6.22E+01	2.34E+00 ± 1.06E+01	**1.76E−09 ± 4.17E−09**
Ackley	3.60E−02 ± 1.82E−02	1.89E−05 ± 1.19E−05	8.26E−15 ± 2.03E−15	**2.66E−15 ± 0.00E+00**
Griewank	5.00E−02 ± 6.40E−02	1.68E−03 ± 4.25E−03	2.12E−03 ± 5.05E−03	**0.00E+00 ± 0.00E+00**
Rastrigin	5.91E+01 ± 2.65E+01	2.77E+01 ± 1.31E+01	1.54E+01 ± 4.46E+00	**0.00E+00 ± 0.00E+00**
Schwefel	7.68E+02 ± 8.94E+02	2.51E+02 ± 1.79E+02	3.43E+03 ± 6.65E+02	**0.00E+00 ± 0.00E+00**
Salomon	8.72E−01 ± 1.59E−01	2.44E−01 ± 5.06E−02	1.16E+00 ± 2.36E−01	**1.95E−01 ± 1.97E−02**
Whitely	8.65E+02 ± 1.96E+02	4.58E+02 ± 7.56E+01	3.86E+02 ± 8.39E+01	**4.93E+01 ± 4.15E+01**
Penalized 1	2.95E−04 ± 1.82E−04	1.12E−09 ± 2.98E−09	4.48E−28 ± 1.64E−31	**1.59E−32 ± 1.02E−33**
Penalized 2	9.03E−03 ± 2.03E−02	4.39E−04 ± 2.20E−03	6.59E−04 ± 2.64E−03	**1.50E−32 ± 2.35E−33**
Population size = 100				
Sphere	3.75E+03 ± 1.14E+03	3.11E+01 ± 1.88E+01	**1.58E−213 ± 0.00E+00**	1.12E−38 ± 3.16E−38
Rosenbrock	4.03E+08 ± 2.59E+08	1.89E+05 ± 1.47E+05	2.06E+01 ± 1.47E+01	**3.57E−5 ± 8.90E−5**
Ackley	1.36E+01 ± 1.48E+00	3.23E+00 ± 5.41E−01	8.12E−15 ± 1.50E−15	**2.66E−15 ± 0.00E+00**
Griewank	3.75E+01 ± 1.26E+01	1.29E+00 ± 1.74E−01	3.45E−04 ± 1.73E−03	**0.00E+00 ± 0.00E+00**
Rastrigin	2.63E+02 ± 2.79E+01	1.64E+02 ± 2.16E+01	1.24E+01 ± 5.80E+00	**0.00E+00 ± 0.00E+00**
Schwefel	6.56E+03 ± 4.25E+02	6.30E+03 ± 4.80E+02	3.43E+03 ± 6.65E+02	**0.00E+00 ± 0.00E+00**
Salomon	5.97E+00 ± 6.54E−01	1.20E+00 ± 2.12E−01	8.30E−01 ± 1.27E−01	**1.93E−01 ± 2.39E−2**
Whitely	1.29E+14 ± 1.60E+14	3.16E+08 ± 4.48E+08	4.34E+02 ± 5.72E+01	1.72E+02 ± 9.62E+01
Penalized 1	6.94E+04 ± 1.58E+05	2.62E+00 ± 1.31E+00	6.22E−03 ± 2.49E−02	**1.57E−32 ± 5.52E−48**
Penalized 2	6.60E+05 ± 7.66E+05	4.85E+00 ± 1.59E+00	6.60E−04 ± 2.64E−03	**1.35E−32 ± 2.43E−34**
Population size = 200				
Sphere	4.01E+04 ± 6.26E+03	1.10E+03 ± 2.98E+02	**5.05E−121 ± 2.44E−120**	1.08E−16 ± 1.19E−16
Rosenbrock	1.53E+10 ± 4.32E+09	1.49E+07 ± 7.82E+06	2.04E+01 ± 8.49E+00	**8.70E+00 ± 1.09E+00**
Ackley	2.02E+01 ± 2.20E−01	9.11E+00 ± 7.81E−01	**7.83E−15 ± 1.41E−15**	5.29E−10 ± 2.53E−10
Griewank	3.73E+02 ± 6.03E+01	1.08E+01 ± 2.02E+00	3.45E−04 ± 1.73E−03	**1.07E−15 ± 1.78E−15**
Rastrigin	3.62E+02 ± 2.12E+01	2.05E+02 ± 1.85E+01	1.23E+01 ± 6.05E+00	**2.93E−01 ± 5.11E−01**
Schwefel	6.88E+03 ± 2.55E+02	6.72E+03 ± 3.24E+02	4.61E+03 ± 6.73E+02	**0.00E+00 ± 0.00E+00**
Salomon	1.34E+01 ± 8.41E−01	3.25E+00 ± 4.55E−01	6.36E−01 ± 9.85E−02	**1.94E−01 ± 2.14E−02**
Whitely	2.29E+16 ± 1.16E+16	5.47E+10 ± 6.17E+10	4.16E+02 ± 5.40E+01	**3.20E+02 ± 4.61E+01**
Penalized 1	2.44E+07 ± 7.58E+06	9.10E+00 ± 2.42E+00	**4.48E−28 ± 1.55E−31**	5.68E−17 ± 1.36E−16
Penalized 2	8.19E+07 ± 1.99E+07	6.18E+01 ± 6.30E+01	**4.29E−28 ± 2.59E−31**	2.19E−16 ± 3.65E−16
Population size = 300				
Sphere	1.96E+04 ± 2.00E+03	6.93E+02 ± 1.34E+02	**5.55E−86 ± 7.59E−86**	3.51E−11 ± 5.21E−11
Rosenbrock	3.97E+09 ± 8.92E+08	5.35E+06 ± 2.82E+06	2.25E+01 ± 1.16E+01	**1.73E+01 ± 6.91E−01**
Ackley	1.79E+01 ± 3.51E−09	7.23E+00 ± 4.50E−01	**7.19E−15 ± 1.48E−15**	9.81E−08 ± 2.65E−08
Griewank	1.79E+02 ± 1.60E+01	7.26E+00 ± 1.74E+00	6.40E−04 ± 3.18E−03	**1.76E−10 ± 1.67E−10**
Rastrigin	2.75E+02 ± 1.27E+01	2.03E+02 ± 1.49E+01	1.30E+01 ± 7.48E+00	**1.00E+01 ± 5.65E+00**
Schwefel	6.87E+03 ± 2.72E+02	6.80E+03 ± 3.37E+02	4.41E+03 ± 6.41E+02	**2.30E−05 ± 6.30E−05**
Salomon	1.52E+01 ± 5.43E−01	3.59E+00 ± 4.54E−01	5.32E−01 ± 8.19E−02	**2.00E−01 ± 5.92E−03**
Whitely	2.96E+16 ± 1.09E+16	1.83E+11 ± 1.72E+11	4.28E+02 ± 5.47E+01	**3.72E+02 ± 1.80E+01**
Penalized 1	3.71E+07 ± 1.29E+07	1.09E+01 ± 3.76E+00	**4.48E−28 ± 1.64E−31**	1.44E−11 ± 1.08E−11
Penalized 2	1.03E+08 ± 1.87E+07	3.42E+02 ± 4.11E+02	**4.29E−28 ± 5.44E−43**	6.34E−11 ± 5.07E−11

Comparison of the ADE algorithm with the CEP, FEP, CPDE1, jDE and SDE1 algorithms

In order to demonstrate the efficiency and superiority of the proposed ADE algorithm, the CEP and FEF [27], CPDE1 [22], SDE1 [19] and jDE [20] algorithms are used for comparison. All algorithms tested on the common benchmark functions set listed in Table 6 with dimensionality of $D = 30$ and population size was set to $NP = 100$. The maximum numbers of generations used are presented in Table 7 [19]. From Table 7(a), it can be seen that the ADE algorithm is superior to the CEP and FEP algorithms in all functions in terms of average values and standard deviation values but the ADE and FEP algorithms attained the same result in step function $f_6(x)$. Furthermore, the results showed that the ADE algorithm outperformed the CPDE1 algorithm in all multimodal functions by significant difference, except for two unimodal functions $f_1(x)$ and $f_2(x)$ where it achieved competitive results. On the other hand, the results in Table 7(b) show that the ADE algorithm outperformed the SDE1 algorithm in $f_5(x)$, $f_8(x)$ and $f_9(x)$ functions which are complex and multimodal functions. Finally, it can be observed that the performance of the ADE and jDE algorithms are almost the same and they approximately achieved the same results in all functions. Last but

Table 4 (a) Scalability comparison of the ADE, basic DE and DEahcSPX algorithms. (b) Comparison of the ADE, basic DE, DEahcSPX and NM-DE in terms of average evaluation times and time of successful runs D = 10 and population size = 30.

Function	DE [13]	DEahcSPX [13]	ADE
(a)			
D = 10 and population size = 30			
Sphere	3.26E−28 ± 5.83E−28	1.81E−38 ± 4.94E−38	**0.00E+00 ± 0.00E+00**
Rosenbrock	4.78E−01 ± 1.32E+00	3.19E−01 ± 1.10E+00	**1.59E−29 ± 2.61E−29**
Ackley	8.35E−15 ± 8.52E−15	2.66E−15 ± 0.00E+00	5.32E−16 ± 1.77E−15
Griewank	5.75E−02 ± 3.35E−02	4.77E−02 ± 2.55E−02	**4.43E−4 ± 1.77E−03**
Rastrigin	1.85E+00 ± 1.68E+00	1.60E+00 ± 1.61E+00	**0.00E+00 ± 0.00E+00**
Schwefel	14.21272743 ± 39.28155167	4.73766066 ± 23.68766692	**0.00E+00 ± 0.00E+00**
Salomon	0.107873375 ± 0.027688791	0.099873361 ± 3.47E−08	**0.09987335 ± 7.60E−12**
Whitely	18.11229734 ± 15.85783313	18.00697444 ± 13.11270338	**0.00E+00 ± 0.00E+00**
Penalized 1	3.85E−29 ± 7.28E−29	4.71E−32 ± 1.12E−47	**4.711634E−32 ± 1.11E−47**
Penalized 2	1.49E−28 ± 2.20E−28	1.35E−32 ± 5.59E−48	1.34E−32 ± 1.10E−47
D = 50 and population size = 50			
Sphere	5.91E−02 ± 9.75E−02	8.80E−09 ± 2.80E−08	**6.40E−94 ± 2.94E−93**
Rosenbrock	1.13E+10 ± 2.34E+10	1.63E+02 ± 3.02E+02	**9.27E−06 ± 2.00E−05**
Ackley	2.39E−02 ± 8.90E−03	1.69E−05 ± 8.86E−06	**5.15E−15 ± 1.64E−15**
Griewank	7.55E−02 ± 1.14E−01	2.96E−03 ± 5.64E−03	**0.00E+00 ± 0.00E+00**
Rastrigin	6.68E+01 ± 2.36E+01	3.47E+01 ± 9.23E+00	**0.00E+00 ± 0.00E+00**
Schwefel	1.07E+03 ± 5.15E+02	9.56E+02 ± 2.88E+02	**0.00E+00 ± 0.00E+00**
Salomon	1.15E+00 ± 1.49E−01	4.00E−01 ± 1.00E−01	**2.27E−01 ± 4.53E−02**
Whitely	1.43E+05 ± 4.10E+05	1.41E+03 ± 2.90E+02	**3.01E+02 ± 2.12E+02**
Penalized 1	3.07E−02 ± 7.93E−02	2.49E−03 ± 1.24E−02	**1.42 E−32 ± 1.35E−32**
Penalized 2	2.24E−01 ± 3.35E−01	2.64E−03 ± 4.79E−03	**4.85E−32 ± 5.57E−32**
D = 100 and population size = 100			
Sphere	4.28E+03 ± 1.27E+03	5.01E+01 ± 8.94E+01	**6.37E−45 ± 1.12E−44**
Rosenbrock	3.33E+08 ± 1.67E+08	1.45E+05 ± 1.11E+05	**8.90E+01 ± 3.46E+01**
Ackley	8.81E+00 ± 8.07E−01	1.91E+00 ± 3.44E−01	**6.21E−015 ± 0.00E+00**
Griewank	3.94E+01 ± 8.01E+00	1.23E+00 ± 2.14E−01	**0.00E+00 ± 0.00E+00**
Rastrigin	8.30E+02 ± 6.51E+01	4.75E+02 ± 6.55E+01	**0.00E+00 ± 0.00E+00**
Schwefel	2.54E+04 ± 2.15E+03	2.48E+04 ± 2.14E+03	**0.00E+00 ± 0.00E+00**
Salomon	1.02E+01 ± 7.91E−01	3.11E+00 ± 5.79E−01	**3.03E−01 ± 1.97E−02**
Whitely	5.44E+15 ± 5.07E+15	4.06E+10 ± 6.57E+10	**7.70E+02 ± 8.69E+02**
Penalized 1	6.20E+05 ± 7.38E+05	4.34E+00 ± 1.75E+00	**9.18E−33 ± 8.09E−33**
Penalized 2	4.34E+06 ± 2.30E+06	7.25E+01 ± 2.44E+01	**6.40E−32 ± 5.87E−32**
D = 200 and population size = 200			
Sphere	1.26E+05 ± 1.06E+04	7.01E+03 ± 1.07E+03	**4.28E−22 ± 4.50E−22**
Rosenbrock	2.97E+10 ± 3.81E+09	1.11E+08 ± 2.63E+07	**2.33E+02 ± 2.52E+01**
Ackley	1.81E+01 ± 2.26E−01	8.45E+00 ± 4.13E−01	**7.12E−13 ± 3.44E−13**
Griewank	1.15E+03 ± 9.22E+01	6.08E+01 ± 9.30E+00	**2.37E−16 ± 2.03E−16**
Rastrigin	2.37E+03 ± 7.24E+01	1.53E+03 ± 8.31E+01	**1.03E+01 ± 3.59E+00**
Schwefel	6.66E+04 ± 1.32E+03	6.61E+04 ± 1.44E+03	**0.00E+00 ± 0.00E+00**
Salomon	3.69E+01 ± 1.80E+00	1.10E+01 ± 4.38E−01	**4.33E−01 ± 4.78E−02**
Whitely	3.13E+18 ± 9.48E+17	4.21E+13 ± 1.74E+13	**1.26E+03 ± 8.07E+02**
Penalized 1	3.49E+08 ± 7.60E+07	2.27E+01 ± 5.73E+00	**1.31E−20 ± 2.83E−20**
Penalized 2	8.08E+08 ± 1.86E+08	6.24E+04 ± 4.77E+04	**1.31E−20 ± 1.36E−20**
(b)			
Sphere	31639.7 **(50)**	22926.4 **(50)**	**6061.8 (50)**
Rosenbrock	73803.8 **(43)**	59275.7 (46)	**54590.4 (50)**
Ackley	48898.2 **(50)**	36389 **(50)**	**9033.6 (50)**
Griewank	–	–	**13891.836735 (49)**
Rastrigin	94089 (13)	84309 (18)	**9582 (50)**
Schwefel	–	–	**7921.2 (50)**
Salomon	–	–	–
Whitely	–	–	**16525.714286 (50)**
Penalized 1	28885.8 **(50)**	20543.5 **(50)**	**5321.4 (50)**
Penalized 2	30812.6 **(50)**	21633.5 **(50)**	**5603.4 (50)**

not least, it is clear that the proposed ADE algorithm performs well with both unimodal and multimodal functions so it greatly balances the local optimization speed and the global optimization diversity.

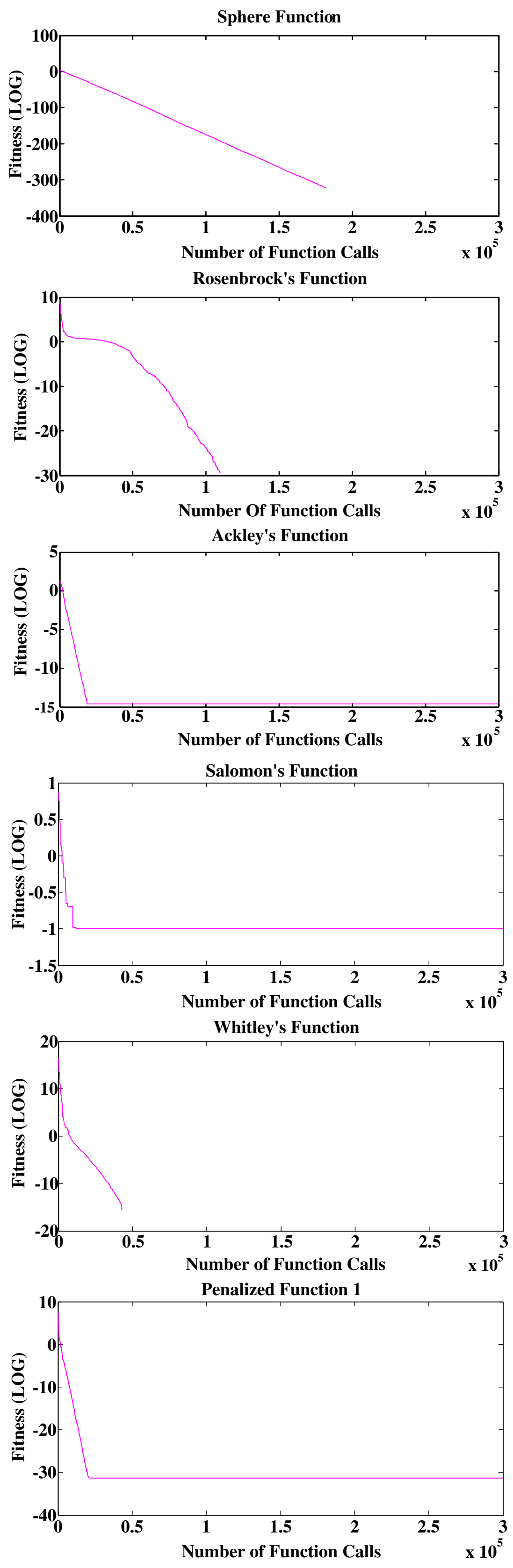


Fig. 4 Average best fitness curves of the ADE algorithm for selected benchmark functions for $D = 10$ and population size = 30.

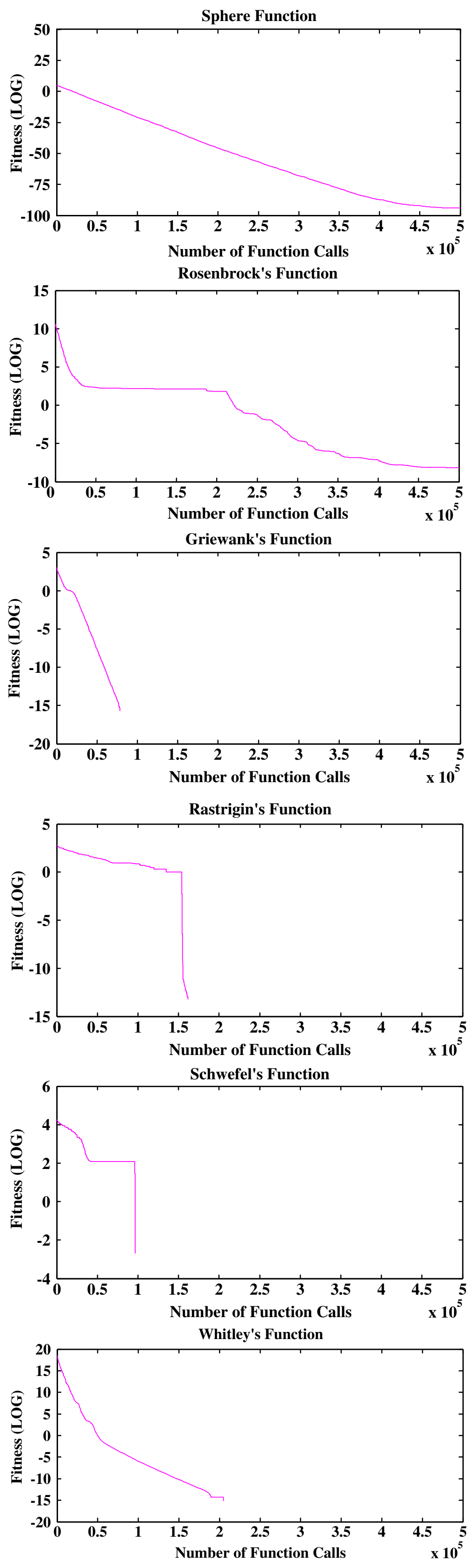


Fig. 5 Average best fitness curves of the ADE algorithm for selected benchmark functions for $D = 50$ and population size = 50.

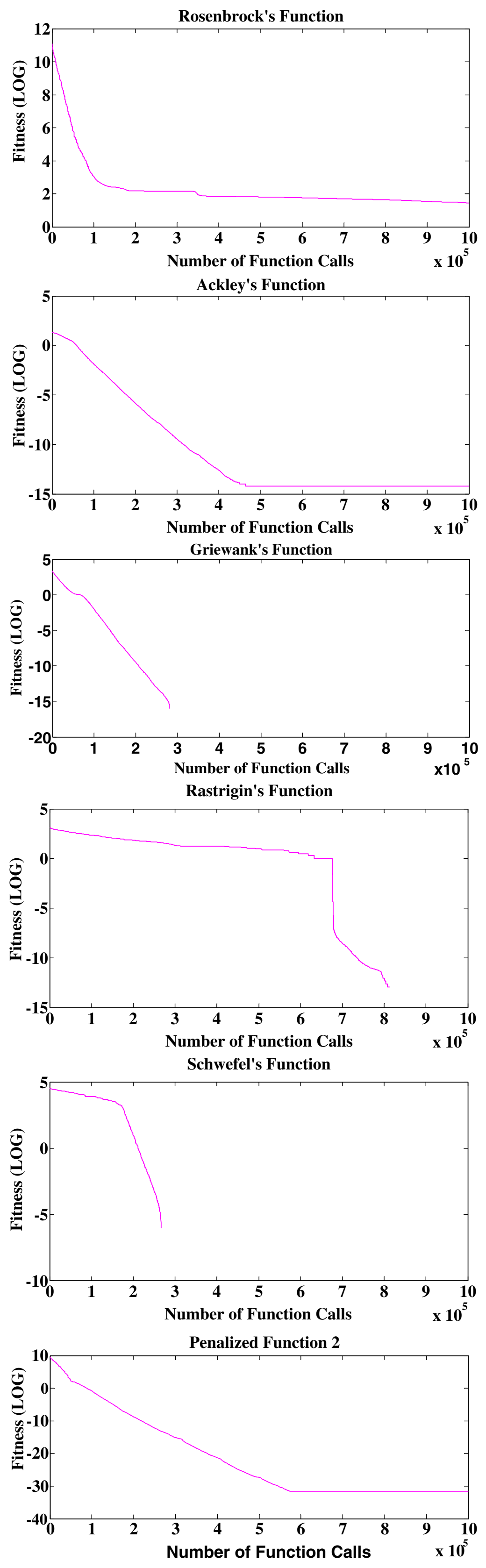


Fig. 6 Average best fitness curves of the ADE algorithm for selected benchmark functions for $D = 100$ and population size = 100.

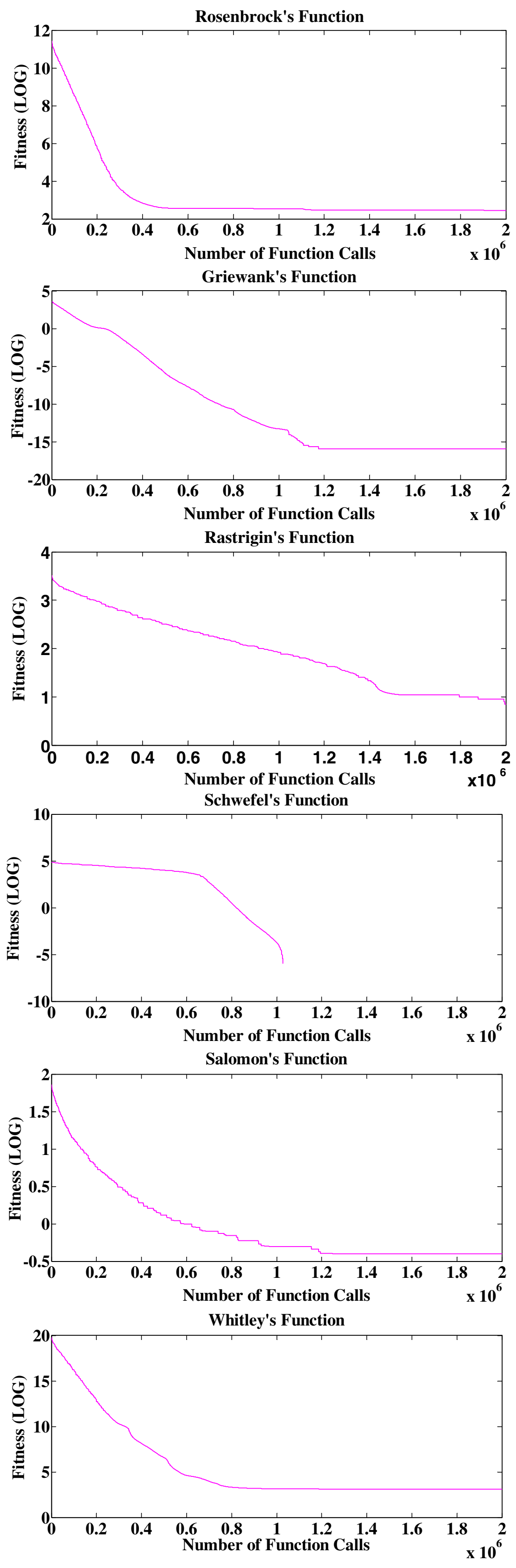


Fig. 7 Average best fitness curves of the ADE algorithm for selected benchmark functions for $D = 200$ and population size = 200.

Table 5 (a) Comparison of the ADE, DEfirSPX and DExhcSPX algorithms $D = 30$ and population size = 30. (b) Comparison of the ADE, DEfirSPX and DExhcSPX algorithms in terms of average evaluation times and time of successful runs $D = 30$ and population size = 30.

Function	DEfirSPX [25]	DExhcSPX [13]	ADE
(a)			
Sphere	1.22E−27 ± 2.95E−27	7.66E−29 ± 1.97E−28	**2.31E−149 ± 1.25E−148**
Rosenbrock	4.84E+00 ± 3.37E+00	5.81E+00 ± 4.73E+00	**4.27E−11 ± 2.26E−10**
Ackley	8.35E−15 ± 1.03E−14	5.22E−15 ± 2.62E−15	**2.66E−15 ± 0.00E+00**
Griewank	3.54E−03 ± 7.55E−03	3.45E−03 ± 7.52E−03	**0.00E+00 ± 0.00E+00**
Rastrigin	2.27E+01 ± 7.39E+00	1.86E+01 ± 7.05E+00	**0.00E+00 ± 0.00E+00**
Schwefel	5.23E+02 ± 3.73E+02	4.91E+02 ± 4.60E+02	**0.00E+00 ± 0.00E+00**
Salomon	1.84E−01 ± 7.46E−02	1.92E−01 ± 4.93E−02	1.93E−01 ± 2.39E−02
Whitely	3.11E+02 ± 9.38E+01	2.84E+02 ± 1.10E+02	**2.65E+01 ± 2.97E+01**
Penalized 1	3.24E−02 ± 3.44E−02	2.49E−02 ± 8.61E−02	**1.58E−32 ± 7.30E−34**
Penalized 2	1.76E−03 ± 4.11E−03	4.39E−04 ± 2.20E−03	**1.77E−32 ± 2.69E−32**
(b)			
Sphere	96588.2 **(50)**	92111.4 **(50)**	**15928.8 (50)**
Rosenbrock	–	–	**189913.8 (50)**
Ackley	142169.88 **(50)**	139982.1 **(50)**	**22589.4 (50)**
Griewank	146999.76 (38)	153119.1 (37)	**16887.446809 (50)**
Rastrigin	–	–	**62427 (50)**
Schwefel	–	–	**41545.6 (50)**
Salomon	–	–	–
Whitely	–	–	**82181.538462 (13)**
Penalized 1	126486.56 (44)	122129.1 (44)	**14685.6 (50)**
Penalized 2	135395.48 (43)	106820.1 (48)	**16002 (50)**

–: None of the algorithms achieved the desired accuracy level $\varepsilon < 10^{-6}$.

Table 6 Benchmark functions.

Gen. no	Test function	D	S	$f_{\min}$
1500	$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100,100]^D$	0
2000	$f_2(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	30	$[-10,10]^D$	0
20000	$f_5(x) = [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30,30]^D$	0
1500	$f_6(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5\rfloor)^2$	30	$[-100,100]^D$	0
9000	$f_8(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i\rvert})$	30	$[-500,500]^D$	−12569.486
5000	$f_9(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12,5.12]^D$	0
1500	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e$	30	$[-32,32]^D$	0
2000	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_{i=1}^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]^D$	0

Conclusions and future work

In this paper, a new and an Alternative Differential Evolution algorithm (ADE) is proposed for solving unconstrained global optimization problems. In order to enhance the local search ability and advance the convergence rate, a new directed mutation rule was presented and it is combined with the basic mutation strategy through a linear decreasing probability rule. Also, two new global and local scaling factors are introduced as two new uniform random variables instead of keeping them constant through generations so as to globally cover the whole search space as well as to bias the search direction to follow the best vector direction. Additionally, a dynamic non-linear increased crossover probability scheme is formulated to balance the global exploration and the local exploitation. Furthermore, a modified BGA mutation and a random mutation scheme are successfully merged to avoid stagnation and/or premature convergence. The proposed ADE algorithm has been compared with the basic DE and other recent two hybrids, three memetic and four self-adaptive DE algorithms that are designed for solving unconstrained global optimization problems on a set of difficult unconstrained continuous optimization benchmark problems. The experimental results and comparisons have shown that the ADE algorithm performs better in global optimization especially with complex and high dimensional problems; it performs better with regard to the search process efficiency, the final solution quality, the convergence rate, and success rate, when compared with other algorithms. Moreover, the ADE algorithm shows robustness and stability for large population size and high dimensionality. Finally yet importantly, the performance of the ADE algorithm is superior and competitive to other recent well-known memetic, self-adaptive and hybrid DE algorithms. Current research efforts focus on how to modify the ADE algorithm to solve constrained and engineering optimization problems. Additionally, future research will investigate the performance

Table 7 (a) Comparison of the ADE, CEP, FEP and CPDE1 algorithms $D = 30$ and population size = 100. (b) Comparison of the ADE, jDE and SDE1 algorithms $D = 30$ and population size = 100.

Gen. no.	Function	CEP [22]	FEP [22]	CPDE1 [22]	ADE
(a)					
1500	$f_1(x)$	0.00022 ± 0.00059	0.00057 ± 0.00013	**0.00E+00 ± 0.00E+00**	1.61E−20 ± 1.70E−20
2000	$f_2(x)$	2.6E−03 ± 1.7E−04	8.1E−03 ± 7.7E−04	**0.00E+00 ± 0.00E+00**	3.38E−21 ± 1.43E−21
20000	$f_5(x)$	6.17 ± 13.6	5.06 ± 5.87	1.5E−06 ± 2.2E−06	**2.08E−29 ± 2.51E−29**
1500	$f_6(x)$	577.76 ± 1125.76	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**
9000	$f_8(x)$	−7917.1 ± 634.5	−12554.5 ± 52.6	−12505.5 ± 97	**−12569.5 ± 1.85E−12**
5000	$f_9(x)$	89 ± 23.1	0.046 ± 0.012	4.5 ± 24.5	**0.00E+00 ± 0.00E+00**
1500	$f_{10}(x)$	9.2 ± 2.8	0.018 ± 0.0021	5.3E−01 ± 6.6E−02	**6.93E−11 ± 3.10E−11**
2000	$f_{11}(x)$	0.086 ± 0.12	0.016 ± 0.022	1.7E−04 ± 2.4E−02	**0.00E+00 ± 0.00E+00**

Gen. no.	Function	jDE [19]	SDE1 [20]	ADE
(b)				
1500	$f_1(x)$	1.1E−28 ± 1.0E−28	**0.00E+00 ± 0.00E+00**	1.61E−20 ± 1.70E−20
2000	$f_2(x)$	1.0E−23 ± 9.7E−24	**0.00E+00 ± 0.00E+00**	3.38E−21 ± 1.43E−21
20000	$f_5(x)$	**0.00E+00 ± 0.00E+00**	2.641954 ± 1.298528	2.08E−29 ± 2.51E−29
1500	$f_6(x)$	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**
9000	$f_8(x)$	−12569.5 ± 7.0E−12	−12360.245 ± 157.628	**−12569.5 ± 1.85E−12**
5000	$f_9(x)$	**0.00E+00 ± 0.00E+00**	1.0358020 ± 0.911946	**0.00E+00 ± 0.00E+00**
1500	$f_{10}(x)$	7.7E−15 ± 1.4E−15	**0.00E+00 ± 0.00E+00**	6.93E−11 ± 3.10E−11
2000	$f_{11}(x)$	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**

of the ADE algorithm in solving multi-objective optimization problems and real world applications.

Appendix 1

Definitions of the benchmark problems are as follows:

Sphere function:

$$f(x) = \sum_{i=1}^{D} x_i^2; \quad -100 \leqslant x_i \leqslant 100; \; f^* = f(0,\ldots,0) = 0$$

Rosenbrock's function:

$$f(x) = [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]; \quad -100 \leqslant x_i \leqslant 100;$$
$$f^* = f(1,\ldots,1) = 0$$

Ackley's function:

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e; \quad -32 \leqslant x_i \leqslant 32; \; f^* = f(0,\ldots,0) = 0.$$

Griewank's function:

$$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_{i=1}^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1; \quad -600 \leqslant x_i \leqslant 600; \; f^* = f(0,\ldots,0) = 0.$$

Rastrigin's function:

$$f(x) = \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i) + 10]; \quad -5.12 \leqslant x_i \leqslant 5.12;$$
$$f^* = f(0,\ldots,0) = 0.$$

Schwefel's function:

$$f(x) = 418{,}9829D - \sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|}); \quad -500 \leqslant x_i \leqslant 500;$$
$$f^* = f(420.9687,\ldots,420.9687) = 0.$$

Salomon's function:

$$f(x) = -\cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2} + 1; \quad -100 \leqslant x_i \leqslant 100, \; f^* = f(0,\ldots,0) = 0$$

Whitley's function:

$$f(x) = \sum_{i=1}^{D}\sum_{i=1}^{D}\left(\frac{y_{i,j}^2}{4000} - \cos(y_{i,j}) + 1\right), \quad -100 \leqslant x_i \leqslant 100,$$
$$f^* = f(1,\ldots,1) = 0$$

Penalized function 1:

$$f(x) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4);$$

where

$$y_i = 1 + \frac{1}{4}(x_i + 1) \text{ and } u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leqslant x_i \leqslant a, \\ k(-x_i - a)^m, & x_i < a. \end{cases}$$
$$-50 \leqslant x_i \leqslant 50, \; f^* = f(-1,\ldots,-1) = 0$$

Penalized function 2:

$$f(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + 3\sin^2(\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4);$$

where

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leqslant x_i \leqslant a, \\ k(-x_i - a)^m, & x_i < a. \end{cases}$$

$$-50 \leqslant x_i \leqslant 50, \; f^* = f(1, \ldots, 1) = 0$$

References

[1] Jie J, Zeng J, Han C. An extended mind evolutionary computation model for optimizations. Appl.Math.Comput. 2007;185(2):1038–49.

[2] Engelbrecht AP. Computational intelligence: an introduction. Wiley-Blackwell; 2002.

[3] Storn R, Price K. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, 1995; Technical Report TR-95-012. ICSI.

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

[5] Price K, Storn R, Lampinen J. Differential evolution – a practical approach to global optimization. Berlin: Springer; 2005.

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

[7] Wang FS, Jang HJ. Parameter estimation of a bio-reaction model by hybrid differential evolution. 2005 IEEE Congr Evol Comput 2000;1:410–7.

[8] Omran MGH, Engelbrecht AP, Salman A. Differential evolution methods for unsupervised image classification. The 2005 IEEE Congress on Evolutionary Computation, vol. 2, Sep 2–5; 2005. p. 966–73.

[9] Das S, Abraham A, Konar A. Automatic clustering using an improved differential evolution algorithm. IEEE Trans Syst Man Cybern A Syst Hum 2008;38(1):218–37.

[10] Das S, Konar A. Design of two dimensional IIR filters with modern search heuristics: A comparative study. Int J Comput Intell Appl 2006;6(3):329–55.

[11] Joshi R, Sanderson AC. Minimal representation multisensor fusion using differential evolution. IEEE Trans Syst Man Cybern A Syst Hum 1999;29(1):63–76.

[12] Vesterstrøm J, Thomson R. A comparative study of differential evolution, particle swarm optimization and evolutionary algorithms on numerical benchmark problems. In: Proceedings of Sixth Congress on Evolutionary Computation: IEEE Press; 2004.

[13] Noman N, Iba H. Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 2008;12(1):107–25.

[14] Lampinen J, Zelinka I. On stagnation of the differential evolution algorithm. In: Ošmera P, editor. Proceedings of 6th International Mendel Conference on Soft Computing; 2000. p. 76–83.

[15] Liu J, Lampinen J. On setting the control parameter of the differential evolution algorithm. In: Matousek R, Osmera P, editors. Proceedings of the 8th International Mendel Conference on Soft Computing, 2002. p. 11–8.

[16] Gämperle R, Müller SD, Koumoutsakos P. A parameter study for differential evolution. In: Grmela A, Mastorakis N, editors. Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation: WSEAS Press; 2002. p. 293–8.

[17] Rönkkönen J, Kukkonen S, Price K. Real-parameter optimization with differential evolution. IEEE Congr Evol Comput 2005:506–13.

[18] Liu J, Lampinen J. A fuzzy adaptive differential evolution algorithm. Soft Comput 2005;9(6):448–62.

[19] Brest J, Greiner S, Boskovic B, Mernik M, Zumer V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 2006;10(6):646–57.

[20] Salman A, Engelbrecht AP, Omran MGH. Empirical analysis of self-adaptive differential evolution. Eur J Oper Res 2007;183(2):785–804.

[21] Xu Y, Wang L, Li L. An effective hybrid algorithm based on simplex search and differential evolution for global optimization. International Conference on Intelligent Computing, 2009. p. 341–350.

[22] Wang YJ, Zhang JS. Global optimization by an improved differential evolutionary algorithm. Appl Math Comput 2007;188(1):669–80.

[23] Fan HY, Lampinen J. A trigonometric mutation operation to differential evolution. J Global Optim 2003;27(1):105–29.

[24] Das S, Konar A, Chakraborty UK. Two improved differential evolution schemes for faster global search. In: GECCO '05 Proceedings of the 2005 conference on Genetic and evolutionary computation; 2005. p. 991–8.

[25] Mühlenbein H, Schlierkamp Voosen D. Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. Evol Comput 1993;1(1):25–49.

[26] Feoktistov V. Differential evolution: In search of solutions. 1st ed. Springer; 2006.

[27] Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans Evol Comput 1999;3(2):82–102.