Egyptian Informatics Journal (2014) **15**, 175–188

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

RDEL: Restart Differential Evolution algorithm with Local Search Mutation for global numerical optimization

Ali Wagdy Mohamed *

Statistics Department, Faculty of Sciences, King Abdulaziz University, P.O. Box 80203, Jeddah 21589, Saudi Arabia
Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza, Egypt

Received 8 October 2013; revised 24 June 2014; accepted 21 July 2014
Available online 24 August 2014

KEYWORDS

Evolutionary computation;
Differential evolution;
Local search mutation;
Restart mechanism;
Global numerical optimization

Abstract In this paper, a novel version of Differential Evolution (DE) algorithm based on a couple of local search mutation and a restart mechanism for solving global numerical optimization problems over continuous space is presented. The proposed algorithm is named as Restart Differential Evolution algorithm with Local Search Mutation (RDEL). In RDEL, inspired by Particle Swarm Optimization (PSO), a novel local mutation rule based on the position of the best and the worst individuals among the entire population of a particular generation is introduced. The novel local mutation scheme is joined with the basic mutation rule through a linear decreasing function. The proposed local mutation scheme is proven to enhance local search tendency of the basic DE and speed up the convergence. Furthermore, a restart mechanism based on random mutation scheme and a modified Breeder Genetic Algorithm (BGA) mutation scheme is combined to avoid stagnation and/or premature convergence. Additionally, an exponent increased crossover probability rule and a uniform scaling factors of DE are introduced to promote the diversity of the population and to improve the search process, respectively. The performance of RDEL is investigated and compared with basic differential evolution, and state-of-the-art parameter adaptive differential evolution variants. It is discovered that the proposed modifications significantly improve the performance of DE in terms of quality of solution, efficiency and robustness.

1. Introduction

Differential Evolution (DE), proposed by Storn and Price [1,2], is a stochastic population-based search method. It shows excellent capability in solving a wide range of optimization problems with different characteristics from several fields and many real-world application problems [3]. Similar to all other Evolutionary algorithms (EAs), the evolutionary process of

* Address: Statistics Department, Faculty of Sciences, King Abdulaziz University, P.O. Box 80203, Jeddah 21589, Saudi Arabia. Tel.: +966 556269723.
E-mail address: aliwagdy@gmail.com

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

1110-8665 © 2014 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University.
http://dx.doi.org/10.1016/j.eij.2014.07.001

DE uses mutations, crossover and selection operators at each generation to reach the global optimum. 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, reliable, speed and robustness [3]. 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 [4]. Additionally, the parameters of DE are problem dependent and it is difficult to adjust them for different problems. Moreover, DE performance decreases as search space dimensionality increases [5]. Finally, the performance of DE deteriorates significantly when the problems of premature convergence and/or stagnation occur [5,6]. Consequently, researchers have suggested many techniques to improve the basic DE. From the literature [7], 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 developing new mutations rule. In this paper, In order to overcome these drawbacks of DE, a restart DE with novel local search mutation is proposed, referred to as RDEL, for global numerical optimization. Therefore, this study develops a new local mutation strategy inspired by Particle swarm optimization (PSO) in DE to enhance the local exploitation tendency and to improve the convergence rate of the algorithm. In fact, in the global PSO version, each particle learns from the personal best position and the best position achieved so far by the whole population. Similarly, the main idea of the proposed novel mutation is based on that each vector learns from the position of the best and the worst individuals among the entire population of a particular generation. Additionally, an exponent increased crossover probability rule and a uniform scaling factors of DE are introduced to promote the diversity of the population and to improve the search process, respectively. Furthermore, a restart mechanism based on random mutation scheme and a modified Breeder Genetic Algorithm (BGA) mutation scheme is combined to avoid stagnation and/or premature convergence. Extensive numerical experiments and comparisons have been conducted in this paper on a set of 14 well-known high dimensional benchmark functions indicate that the proposed (RDEL) algorithm is superior and competitive with conventional DE and several state-of-the-art parameter adaptive DE variants particularly in the case of high dimensional complex optimization problems. The rest of the paper is organized as follows. In Section 2, the standard DE algorithm is introduced with a review of its operators and parameters. Next, in Section 3, the proposed algorithm is introduced. Section 4 computational results of testing benchmark functions and on the comparison with other techniques are reported and discusses the effectiveness of the proposed modifications. Finally, conclusions and future works are drawn in Section 5.

2. Differential evolution

To start with, a bound constrained global optimization problem can be defined as follows [8]:

$$\min f(\vec{x}), \vec{x} = [x_1, x_2, \ldots, x_D]; \quad S.t. x_j \in [a_j, b_j], \forall j = 1, 2, \ldots, D \quad (1)$$

where f is the objective function, $\vec{x}$ is the decision vector consisting of variables, and a_j and b_j are the lower and upper bounds for each decision variable, respectively.

In simple DE, generally known as DE/rand/1/bin [1,9], an initial random population consists of NP vectors $\vec{X}_i$, $\forall i = 1, 2, \ldots, NP$, is generated within the boundaries. These individuals are evolved by DE operators (mutation and crossover) to generate a trial vector. A comparison between the parent and its trial vector is then done to select the vector which should survive to the next generation [9]. DE steps are discussed below:

2.1. Initialization

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

$$x_{ij}^0 = a_j + rand_j \cdot (b_j - a_j) \quad (2)$$

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

2.2. Mutation

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

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

with randomly chosen indices $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$. 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 [2], the range of F is in [0, 2]. If a component of a mutant vector violates search space, then the value of this component is generated a new using (2) or new other repair method.

2.3. Crossover

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

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

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

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

2.4. Selection

DE adapts a greedy selection strategy. If and only if the trial vector u_i^{G+1} yields as good as or 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):

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

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

2.5. DE literature review

Due to DE cons many researchers have proposed novel techniques to overcome these problems as well as improve its performance. Storn and Price [2] suggested that NP (population size) between 5D and 10D is preferred, while 0.5 is as a good initial value of F (mutation scaling factor). The effective value of F usually lies in a 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 accelerates 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. [10] recommended that a good choice for NP is between 3D and 8D, with $F = 0.6$ and CR lies in [0.3, 0.9]. Contrarily, Ronkkonen et al. [11] concluded that $F = 0.9$ is a good compromise between convergence speed and 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 between 0 and 0.2 when the objective function is separable.

Table 1 Description of standard **DE** algorithm.

```
01. Begin
02.   G = 0
      Create a random initial population x_i^G, Evaluate
03.   f(x_i^G) ∀i, i = 1,...,NP
04.   For G = 1 to GEN Do
05.     For i = 1 to NP Do
06.   Select randomly r1 ≠ r2 ≠ r3 ≠ i ∈ [1, NP], j_rand = randint(1,D)
07.       For j = 1 to D Do
08.         If (rand_j[0,1] < CR or j = j_rand) Then
09.           u_{i,j}^{G+1} = x_{r1,j}^G + F·(x_{r2,j}^G − x_{r3,j}^G)
10.         Else
11.           u_{i,j}^{G+1} = x_{i,j}^G
12.         End If
13.       End For
14.       If(f(u_i^{G+1}) ⩽ f(x_i^G))Then
15.         x_i^{G+1} = u_i^{G+1}
16.       Else
17.         x_i^{G+1} = x_i^G
18.       End If
19.     End For
20.     G = G + 1
21.   End For
22. End
```

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

To avoid the seeming contradictions from the literature, some techniques have been designed to adjust control parameters in adaptive or self-adaptive manner instead of trial-and-error procedure. A Fuzzy Adaptive Differential Evolution (FADE) algorithm was proposed by Liu and Lampinen [12]. 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. [9] proposed an efficient technique, named jDE, for self-adapting control parameter settings by encoding the parameters into each individual and adapting 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. Along the same line, Omran et al. [13] proposed a Self-adaptive Differential Evolution (SDE) algorithm. In it, F was self-adapted using a mutation rule similar to the mutation operator in the basic DE. The experiments conducted showed that SDE generally outperformed DE algorithms and other evolutionary algorithms.

Zaharie [14] introduced an adaptive DE (ADE) algorithm based on the idea of controlling the population diversity and implemented a multi-population approach. Qin et al. [15] 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 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 was able to obtain better quality solutions and had higher success rate.

Similarly, Zhang and Sanderson [16] 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. Simulation results show that JADE was better than, or at least competitive to, other classic or adaptive DE algorithms, Particle swarm and other evolutionary algorithms from the literature in terms of convergence performance.

Recently, inspired by SaDE algorithm and motivated by the recent success of diverse self-adaptive DE approaches, Mallipeddi et al. [17] 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 compared with conventional DE and other state-of-the-art parameter adaptive DE variants. The comparative results showed that EPSDE algorithm outperformed conventional DE and other state-of-the-art parameter adaptive DE variants in terms of solution quality and robustness.

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 [5,18–32].

3. RDEL: Restart Differential Evolution algorithm with Local Search mechanism

3.1. Local search mechanism

In order to enhance the local search tendency, and to accelerate the convergence of DE technique, a new local mutation rule is proposed based on the position of the best and the worst individuals the entire population, respectively. It is worth mentioning that the proposed mutation is inspired by the structure of PSO algorithm that mimics social-psychological principles such as flocks of birds, schools of fish and ant colonies. Briefly, in the global PSO version, each particle learns from the personal best position and the best position achieved so far by the whole population. Similarly, the main idea of the proposed novel mutation is based on that each vector learns from the position of the best and the worst individuals among the entire population of a particular generation. Simply, the new position of each mutant vector depends on the position of the best and worst vectors achieved so far by the whole population of a particular generation by following the same direction of the best and similarly by avoiding the direction of the worst. The modified mutation scheme is as follows:

$$v_i^{G+1} = x_{r_1}^G + F1 \cdot (x_{best}^G - x_{r_1}^G) + F2 \cdot (x_{r_1}^G - x_{worst}^G) \tag{6}$$

where $x_{r_1}^G$ is a random chosen vector and x_{best}^G and x_{worst}^G are the best and worst vectors in the entire population, respectively. The novel local mutation scheme is joined with the basic mutation rule through a linear decreasing function as follows:

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

$$v_i^{G+1} = x_{r_1}^G + F1 \cdot (x_{best}^G - x_{r_1}^G) + F2 \cdot (x_{r_1}^G - x_{worst}^G) \tag{8}$$

Else

$$v_i^{G+1} = x_{r1}^G + F3 \cdot (x_{r2}^G - x_{r3}^G) \tag{9}$$

where $F1$, $F2$, $F3$ are three 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. Obviously, from mutation Eq. (6), it can be observed that the movement of the mutant vector is carried out in the direction of the best vector and in the opposite direction to the worst one. Thus, the directed perturbation in the proposed mutation resembles the concept of gradient as the difference vector is oriented from the worst to the best vectors [34]. Consequently, the proposed directed mutation favors exploitation since all vectors of population are biased by the best direction but are perturbed by the different weights. As a result, the new mutation rule has better local search ability and faster convergence rate. Thus, Therefore, this process can easily reach the location of the global and/or local optimum in the search space when solve unimodal and multimodal problems, respectively. In the proposed RDEL algorithm, both local mutation operator and basic mutation operator are selected based on a linear decreasing probability rule Eq. (7). In fact, it can be seen that, from Eq. (7), it favor global exploration at the beginning of the search process since the probability of using the basic mutation strategy is greater than the probability of using new local mutation. However, at latter period, it biases to exploitation tendency because the probability of using the local mutation scheme is greater than the probability of using the basic mutation rule. As a result, through generations, both global exploration and local exploitation capabilities are executed. On the contrary, enhancing local search ability of DE may lead to premature convergence or stagnation situations. Hence, a restart mechanism based on random mutation scheme and a modified Breeder Genetic Algorithm (BGA) mutation scheme [33] is combined to avoid both situations. In this mechanism, one of two mutation operators is performed based on a predefined probability.

3.2. Restart mechanism

A restart mechanism is applied for each solution vector 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 to a predetermined level δ for predetermined allowable number of generations K, then one of the above mentioned mutations is applied with equal probability of (0.5). This restart mechanism can be expressed as follows:

$$\begin{aligned} &\text{If} \quad |f_c - f_p| \leqslant \delta \quad \text{for } K \text{ generations, then} \\ &\text{If} \quad (u(0,1) \geqslant 0.5), \text{then Apply Random Mutation} \\ &\text{Else} \quad \text{Apply Modified BGA mutation} \end{aligned} \tag{10}$$

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 all dimensions, it has been observed that $\delta = \text{E}{-}06$ and $K = 25$ 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. 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.

Generally, in the random mutation, for a chosen vector x_i at a particular generation, a uniform random integer number j_{rand} between $[1, D]$ is firstly generated and then 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.

$$x_j' = \begin{cases} a_j + rand_j(b_j - a_j) & j = j_{rand} \\ x_j & \text{otherwise} \end{cases} \forall j = 1, \ldots, D \tag{11}$$

Therefore, it can be deduced from the above equation that random mutation increases the diversity of the DE algorithm and decreases the risk of plunging into local point or any other point in the search space. In order to perform BGA mutation, as discussed by Mühlenbein and Voosen [33], 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 $0.1 \bullet (b_j - a_j) \bullet \alpha$ 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.

$$x'_j = \begin{cases} x_j \pm 0.1 \cdot (b_j - a_j) \cdot \alpha & j = j_{rand} \\ x_j & \text{otherwise} \end{cases}, \forall j = 1, \ldots, D \quad (12)$$

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

$$\alpha = \sum_{k=0}^{15} \alpha_k \cdot 2^{-k}, \alpha_k \in \{0, 1\} \quad (13)$$

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:

$$x'_j = \begin{cases} x_j \pm rand_j(b_j - a_j) \cdot \alpha & j = j_{rand} \\ x_j & \text{otherwise} \end{cases}, \forall j = 1, \ldots, D \quad (14)$$

where the factor of 0.1 in Eq. (12) is replaced by a uniform random number in (0, 1], because the constant setting of $0.1 \bullet (b_j - a_j)$ is not suitable. However, the probabilistic setting of $rand_j \bullet (b_j - a_j)$ 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.

3.3. Crossover scheme

Crossover (CR) reflects the probability with which the trial individual inherits the actual individual's genes [34]. As a matter of fact, if Cr is high, this will increase the population diversity. Nevertheless, the convergence rate may decrease and/or the population may prematurely converge. On the other hand, 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.

Table 2 The description of RDEL.

```
Begin
G = 0
Create a random initial population x_i^G, Evaluate f(x_i^G) ∀i, i = 1, ..., NP
For G = 1 to GEN Do
    CR = 0.8 + (0.1−0.8)·(1-G/GEN)^4 (exponent increased Crossover)
  For i = 1 to NP Do
  F1 = rand(0,1), F2 = rand(0,1), F3 = rand(0,1), (Scaling Factors)
    Select randomly r1 ≠ r2 ≠ r3 ≠ i ∈ [1, NP], j_rand = randint(1,D)
   For j = 1 to D Do
    If (rand_j[0,1] < CR or j = j_rand) Then
    If (rand[0,1] >= (1 − G/GEN)) Then (Use New local Mutation operator)
    Determine the tournament x_best^G, and x_worst^G based on f(x_i^G), i = 1, 2, 3 ... NP
    u_i^{G+1} = x_{r1}^G + F1 · (x_best^G − x_{r1}^G) + F2 · (x_{r1}^G − x_worst^G),
    Else (use basic mutation operator)
    u_{i,j}^{G+1} = x_{r1,j}^G + F3 · (x_{r2,j}^G − x_{r3,j}^G)
    End If
    Else
   u_{i,j}^{G+1} = x_{i,j}^G
   End If
   End For
   If (f(u_i^{G+1}) ⩽ f(x_i^G)) Then
   x_i^{G+1} = u_i^{G+1}
   Else
   x_i^{G+1} = x_i^G
   End If
   If |f(i)_current − f(i)_previous| ⩽ δ = 10^{-6} for K = 25 generations, ∀i, i = 1, ..., NP
   If (rand(0, 1) ⩾ 0.5) Then (Use restart mechanism)
   x_ij^{G+1} = { a_j + rand_ij * (b_j − a_j)   j = j_rand
                { x_ij                          otherwise , j = 1, ..., D (Random Mutation)
   Else
   x_ij^{G+1} = { x_ij + rand_ij * (b_j − a_j) * α   j = j_rand
                { x_ij                               otherwise , j = 1, ..., D (Modified BGA
   End If
   End If
  End For
   G = G + 1
 End For
End
```

Therefore, Cr should 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, Cr should 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:

$$Cr = Cr_{\max} + (Cr_{\min} - Cr_{\max}) \cdot (1 - G/GEN)^k \quad (15)$$

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. (15), $Cr_{\min} = 0.1$ is considered as a good initial rate in order to avoid high level of diversity in the early stage as discussed earlier and by Storn and Price [2]. Additionally, $Cr_{\max} = 0.8$ is the maximum value of crossover that can balance between exploration and exploitation. k is set to its mean value as it is observed, if it is approximately less than or equal to 1 or 2 then the diversity of the population deteriorates for some functions and it could cause 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 all dimensions as the default value with all benchmark problems. A detailed description of RDEL is given in Table 2.

4. Experiments and discussion

4.1. Benchmark functions

In order to evaluate the performance of the proposed algorithm (RDEL), 14 well-known benchmark test functions mentioned in [5,35] are used. All these functions are minimization problems. Among the functions, f_1–f_4 are unimodal and functions f_5–f_{14} are multimodal. However, the generalized Rosenbrock's function f_3 is a multimodal function when $D > 3$ [36]. These 14 test functions are dimension wise scalable. Definitions of the Benchmark Problems are presented in Appendix A. The initialization ranges, the range of the search space, and the position of the global minimum for these 14 benchmark functions are presented in Table 3.

4.2. Algorithms for comparisons

In order to evaluate the benefits of the proposed modifications, a comparison of RDEL with six state-of-the-art self-adaptive differential evolution algorithms is made. These approaches are SaDE [15], jDE [9], ADE [14], JADE [16], SDE [13] and EPSDE [17]. The above benchmark functions f_1 to f_{14} are tested in 10-dimensions (10-D), 30-dimensions (30-D) and 50-dimensions (50-D).The maximum number of function evaluations is set to 100,00 for 10D problems, 300,000 for 30D problems and 500,000 for 50D problems. The population size is set to 50 for all dimensions with all functions. For each problem, 30 independent runs are performed and statistical results are provided including the mean and the standard deviation values. The performance of different algorithms is statistically compared with RDEL by statistical t-test with significance level of 0.05. Numerical values -1, 0, 1 (h values) represent that the RDEL is inferior to, equal to and superior to the algorithm with which it is compared, respectively. The experiments were carried out on an Intel (R) core i7 processor 1.6 GHz and 4 GB-RAM. RDEL algorithm is coded and realized in MATLAB.

4.3. Experimental results and discussions

The results (mean, standard deviation of the best-of-run errors and t-test results) of the comparisons are provided in Table 4 for 10-dimensions problems, Table 5 for 30-dimensions problems and Table 6 for 50-dimensions problems. Note that the best-of-the-run error corresponds to absolute difference between the best-of-the-run value $f(\vec{x}_{best})$ and the actual optimum f_* of a particular objective function i.e. $|f(\vec{x}_{best}) - f_*|$. The results provided by these approaches were directly taken from Ref. [17]. In Tables 4–6, results are compared in terms of mean error and standard deviation. From the results, it

Table 3 Global optimum, search ranges and initialization ranges of the test functions.

Functions	Dimension	Global optimum x^*	$f(x^*)$	Search range	Initialization range
f_1	10,30 and 50	$\mathbf{o}$	0	$[-100,100]^D$	$[-100,100]^D$
f_2		$\mathbf{o}$	0	$[-100,100]^D$	$[-100,100]^D$
f_3		$(1,1,\ldots,1)$	0	$[-100,100]^D$	$[-100,100]^D$
f_4		$\mathbf{o}$	0	$[-100,100]^D$	$[-100,100]^D$
f_5		$\mathbf{o}$	0	$[-32,32]^D$	$[-32,32]^D$
f_6		$\mathbf{o}$	0	$[-32,32]^D$	$[-32,32]^D$
f_7		$\mathbf{o}$	0	R	$[0,600]^D$
f_8		$\mathbf{o}$	0	R	$[0,600]^D$
f_9		$\mathbf{o}$	0	$[-5,5]^D$	$[-5,5]^D$
f_{10}		$\mathbf{o}$	0	$[-5,5]^D$	$[-5,5]^D$
f_{11}		$(420.96,\ldots,420.96)$	0	$[-500,500]^D$	$[-500,500]^D$
f_{12}		$(420.96,\ldots,420.96)$	0	$[-500,500]^D$	$[-500,500]^D$
f_{13}		$\mathbf{o_1}$	0	$[-5,5]^D$	$[-5,5]^D$
f_{14}		$\mathbf{o_1}$	0	$[-5,5]^D$	$[-5,5]^D$

$\mathbf{o}$ is the shifted vector. $\mathbf{o_1}$ is the shifted vector for the first basic function in the composition function.

Table 4 Comparison between RDEL and various state-of-the-art methods on 10D problems.

Fcn	SaDE		jDE		ADE		SDE		JADE		EPSDE		RDEL	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
f_1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	0	0	0	0	0	0	0	0	0	0	0	0	0	–
f_2	**0**	**0**	**0**	**0**	7.50E−03	6.60E−03	**0**	**0**	**0**	**0**	**0**	**0**	**1.37E−20**	**2.30E−20**
	0	0	0	0	1	0	0	0	0	0	0	0	0	–
f_3	**0**	**0**	2.66E−01	1.01E+00	7.59E−01	7.38E−01	2.21E+00	1.77E+00	5.30E−01	1.40E+00	7.13E−10	3.90E−09	**1.06E−12**	**3.27e−12**
	0	0	1	0	1	0	1	0	1	0	1	0	0	–
f_4	**0**	**0**	**0**	**0**	1.02E+00	5.93E−01	1.83E−09	1.00E−08	**0**	**0**	**0**	**0**	**8.13E−18**	**2.09E−17**
	0	0	0	0	1	0	1	0	0	0	0	0	0	–
f_5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	0	0	0	0	0	0	0	0	0	0	0	0	0	–
f_6	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2.04E+01	8.48E−01	**0**	**0**	**0**	**0**
	0	0	0	0	0	0	0	0	1	0	0	0	0	–
f_7	**0**	**0**	**0**	**0**	**0**	**0**	4.20E−03	7.90E−03	9.68E−12	1.77E−11	**0**	**0**	**0**	**0**
	0	0	0	0	0	0	1	0	0	0	0	0	0	–
f_8	**1.37E−02**	**1.18E−02**	**2.26E−02**	**1.77E−02**	7.93E−02	4.24E−02	3.81E−02	3.06E−02	**2.19E−02**	**8.70E−3**	8.91E−02	4.27E−02	**8.21E−03**	**6.73E−03**
	0	0	0	0	1	0	1	0	0	0	1	0	0	–
f_9	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1.33E−02	1.26E−02	**0**	**0**	**0**	**0**	**0**
	0	0	0	0	0	0	1	0	0	0	0	0	0	–
f_{10}	**2.82E+00**	**1.28E+00**	4.41E+00	1.14E+00	5.46E+00	1.37E+00	3.98E+00	2.06E+00	4.22E+00	1.30E+00	7.33E+00	1.39E+00	5.50E+00	2.06E+00
	−1	0	0	0	0	0	0	0	0	0	1	0	0	–
f_{11}	**0**	**0**	**0**	**0**	**0**	**0**	1.40E+00	1.19E+00	**0**	**0**	**0**	**0**	**0**	**0**
	0	0	0	0	0	0	1	0	0	0	0	0	0	–
f_{12}	**0**	**0**	1.18E+01	3.61E+01	**0**	**0**	7.90E+00	3.01E+01	**0**	**0**	**0**	**0**	**0**	**0**
	0	0	1	0	0	0	1	0	0	0	0	0	0	–
f_{13}	**0**	**0**	6.67E+00	2.54E+01	1.20E−03	3.00E−03	1.67E+01	3.79E+01	2.33E+01	4.30E+01	**0**	**0**	**0**	**0**
	0	0	1	0	1	0	1	0	1	0	0	0	0	–
f_{14}	2.54E−01	5.21E−01	1.27E+00	3.20E+00	5.87E+00	4.85E+00	8.26E+00	1.28E+01	3.33E+00	1.83E+01	**0**	**0**	6.52E−01	9.94E−01
	−1	0	1	0	1	0	1	0	1	0	0	0	0	–

Table 5 Comparison between RDEL and various state-of-the-art methods on 30D problems.

Fcn	SaDE		jDE		ADE		SDE		JADE		EPSDE		RDEL	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
f_1	**0**	**0**	**0**	**0**	**0**	**0**	3.14E−01	1.62E+00	**0**	**0**	**0**	**0**	**0**	**0**
	0	0	0	1	0	0	–							
f_2	**0**	**0**	**0**	**0**	9.29E+02	2.55E+02	3.77E−01	7.09E−01	**0**	**0**	2.78E−12	1.26E−11	8.83E−06	1.21E−05
	−1		−1		1		1		−1		−1		–	
f_3	1.33E−01	7.28E−01	5.98E−01	1.44E+00	3.18E+01	1.57E+01	5.31E+02	1.59E+03	4.02E+00	1.60E+01	**2.69E−22**	**9.9E−22**	5.98E−09	1.81E−08
	1		1		1		1		1		−1		–	
f_4	1.01E+01	3.01E+01	2.24E+00	1.01E+01	7.51E+03	1.54E+03	3.33E+02	3.22E+02	3.60E−03	6.70E−03	**1.32E−05**	**5.41E−05**	6.83E−02	7.54E−02
	1		1		1		1		1		−1		–	
f_5	**0**	**0**	**0**	**0**	4.03E−15	1.23E−15	1.67E−01	3.37E−01	5.32E−15	1.81E−15	**0**	**0**	3.55E−15	0
	−1		−1		1		1		1		−1		–	
f_6	**0**	**0**	**0**	**0**	3.91E−15	1.08E−15	2.11E−01	5.16E−01	1.65E−01	5.21E−01	**0**	**0**	3.55E−15	0
	−1		−1		0		1		1		−1		–	
f_7	4.10E−03	8.80E−03	**0**	**0**	**0**	**0**	1.54E+00	4.58E+00	**0**	**0**	**0**	**0**	**0**	**0**
	1		0		0		1		0		0		–	
f_8	2.20E−03	4.70E−03	**0**	**0**	1.36E−07	6.27E−07	1.08E+00	1.87E+00	1.49E−02	1.34E−02	**0**	**0**	**0**	**0**
	1		0		1		1		1		0		–	
f_9	1.33E−01	4.32E−01	3.32E−02	1.82E−01	**0**	**0**	1.00E+01	3.49E+00	**0**	**0**	**0**	**0**	**0**	**0**
	1		1		0		1		0		0		–	
f_{10}	**2.38E+00**	**4.97E+00**	2.36E+01	3.23E+00	4.22E+01	4.10E+00	2.32E+01	6.98E+00	2.57E+01	4.47E+00	7.81E+01	1.33E+01	2.37E+01	6.05E+00
	−1		0		1		0		0		1		–	
f_{11}	2.00E−01	4.07E−01	3.33E−02	1.83E−01	**0**	**0**	1.46E+01	4.23E+00	**0**	**0**	**0**	**0**	**0**	**0**
	1		1		0		1		0		0		–	
f_{12}	**0**	**0**	3.55E+01	6.34E+01	**0**	**0**	3.76E+02	1.89E+02	7.90E+00	3.00E+01	**0**	**0**	**0**	**0**
	0		1		0		1		1		0		–	
f_{13}	**0**	**0**	1.00E+01	4.03E+01	1.67E−03	9.13E−03	6.76E+00	3.65E+01	1.33E+01	5.07E+01	**0**	**0**	**5.25E−31**	**4.53E−30**
	0		1		1		1		1		0		–	
f_{14}	3.51E+01	9.38E−01	9.22E+00	2.56E+01	2.56E+01	4.18E+00	1.16E+01	1.86E+01	8.84E+01	2.55E+01	4.94E+00	7.19E−01	**1.61E+00**	**7.79E−01**
	1		1		1		1		1		1		–	

Table 6 Comparison between RDEL and various state-of-the-art methods on 50D problems.

Fcn	SaDE		jDE		ADE		SDE		JADE		EPSDE		RDEL	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
f_1	**0**	**0**	**2.57E−29**	**6.61E−29**	**0**	**0**	2.17E+01	3.85E+01	**0**	**0**	**0**	**0**	7.79E−29	1.12E−28
	0		0		0		1		0		0		–	
f_2	**1.47E−09**	**5.93E−09**	2.11E−04	2.72E−04	1.70E+04	2.72E+03	3.26E+02	3.57E+02	**5.20E−23**	**1.13E−22**	**4.47E−09**	**1.75E−08**	4.58E−01	5.12E−01
	−1		1		1		1		−1		−1		–	
f_3	1.42E+00	2.44E+00	7.25E+00	1.77E+01	4.55E+01	1.08E+00	5.54E+06	1.36E+07	1.20E+00	1.93E+00	**0**	**0**	7.93E−06	2.82E−05
	1		1		1		1		1		−1		–	
f_4	3.05E+03	1.98E+03	1.26E+03	1.39E+03	4.15E+04	4.98E+03	9.46E+03	3.38E+03	1.42E+03	1.55E+03	5.47E+02	9.47E+02	**2.66E+02**	**2.34E+02**
	1		1		1		1		1		1		–	
f_5	6.45E−01	7.09E−01	**4.62E−15**	**1.63E−15**	7.11E−11	8.45E−12	1.17E+00	7.35E−01	7.11E−15	1.52E−16	**8.05E−15**	**2.46E−15**	**3.90E−15**	**1.08E−15**
	1		0		1		1		0		0		–	
f_6	1.05E+00	6.58E−01	**4.38E−15**	**1.50E−15**	**6.87E−15**	**8.86E−16**	9.90E−01	7.95E−01	1.08E+00	7.78E−01	**7.11E−15**	**2.46E−15**	**3.67E−15**	**6.48E−16**
	1		0		0		1		1		0		–	
f_7	6.40E−03	1.15E−02	2.00E−04	1.30E−03	**0**	**0**	1.40E+01	1.20E+01	5.70E−03	1.09E−01	**0**	**0**	**0**	**0**
	1		1		0		1		1		0		–	
f_8	5.00E−03	1.26E−02	7.00E−04	2.90E−03	**6.49E−12**	**2.56E−11**	1.14E+01	1.40E+01	3.20E−03	5.30E−03	**0**	**0**	**0**	**0**
	1		1		0		1		1		0		–	
f_9	2.09E+00	1.34E+00	1.99E−01	3.99E−01	**0**	**0**	2.39E+01	5.54E+00	**0**	**0**	**0**	**0**	**0**	**0**
	1		1		0		1		0		0		–	
f_{10}	7.33E+01	1.65E+01	4.09E+01	5.98E+00	1.27E+02	9.79E+01	5.94E+01	1.23E+01	7.75E+01	1.96E+01	2.00E+02	2.90E+01	**4.40E+01**	**9.03E+00**
	1		1		1		1		1		1		–	
f_{11}	1.57E+00	1.28E+00	3.33E−01	4.71E−01	2.00E−01	4.00E−01	3.46E+01	5.71E+00	**0**	**0**	**0**	**0**	**0**	**0**
	1		1		1		1		0		0		–	
f_{12}	3.95E+00	2.16E+01	1.10E+02	1.36E+02	**1.28E−11**	**3.45E−12**	1.07E+03	3.29E+02	5.92E+01	1.15E+02	**0**	**0**	1.82E−11	0
	1		1		−1		1		1		−1		–	
f_{13}	1.67E+01	4.61E+01	2.33E+01	4.96E+01	1.19E−08	6.52E−08	2.24+01	5.44E+01	1.00E+01	3.16E+01	**0**	**0**	**1.72E−30**	**3.55E−30**
	1		1		1		1		1		0		–	
f_{14}	5.01E+01	1.75E+02	2.41E+01	4.02E+01	1.26E+01	2.16E+00	2.14E+01	2.55E+01	1.52E+01	3.11E+01	3.15E+01	4.13E+01	**1.15E+00**	**3.71E−01**
	1		1		1		1		1		1		0	

can be clearly concluded that RDEL can perform better than other compared algorithms in most of the cases. From Table 4, it can be obviously seen that RDEL, SaDE and EPSDE exhibit better high quality results for all benchmark problems than jDE, ADE, SDE and JADE. From the *t*-test results, it can be observed that RDEL is inferior to, equal to, superior to compared algorithms in 2, 53 and 19 cases, respectively out of the total 84 cases. Thus, the RDEL is always either better or equal. Table 5 indicates that the RDEL algorithm is superior to the SDE algorithm in all functions in terms of average and Standard deviation values. However, RDEL is surpassed by the SDE algorithm on function f_{10}. Generally, the performance of RDEL algorithm is superior in most of the cases to SaDE, jDE, ADE and JADE algorithms, respectively. Finally, it can be observed that the performance of the RDEL and EPSDE algorithms are almost the same and they approximately achieved the same results in most of the functions. All in all, from the t-test results, it can be observed that RDEL is inferior to, equal to, superior to compared algorithms in 3, 44 and 41 cases, respectively out of the total 84 cases. Thus, the RDEL is almost either better or equal. According to Table 6, as the dimensionality of the problems increases from 10D to 50D, we can conclude that the performance of all other compared algorithms except RDEL and EPSDE deteriorates significantly. Therefore, it can be deduced that RDEL is superior to all algorithms and is competitive with EPSDE with high quality final solution with lower mean and standard deviation values. Moreover, the results show that the proposed RDEL algorithm outperforms EPSDE algorithm on the most difficult functions f_4, f_{10} and f_{14} by remarkable difference. From the *t*-test results, it is obvious that the RDEL are inferior to, equal to, superior to compared algorithms in 6, 22 and 56 cases, respectively out of the total 84 cases. Thus, the RDEL is almost either better or equal. In summary, our proposed RDEL approach can achieve better performance than all other competitive algorithms in terms of both the quality of the final solutions and robustness.

4.4. A parametric study on RDEL

In this section, in order to investigate the impact of the proposed modifications, some experiments are conducted. Two different versions of RDEL and conventional DE algorithm have been tested and compared against the proposed one.

1. Version 1: local mutation strategy is used without both the basic mutation scheme and the random and modified (BGA) mutations. (Denoted as RDEL-1.)
2. Version 2: local mutation strategy is combined with random and modified (BGA) mutations without using the basic mutation strategy. (Denoted as RDEL-2.)
3. Conventional DE: To investigate the gained advancement over the standard DE algorithm. DE/rand/1/bin strategy with ($F = 0.5$, $CR = 0.9$) is considered for comparison. These parameter settings are extensively used in the literature [1,7,9]. (Denoted as DE.)

In order to evaluate the final solution quality, efficiency, convergence rate, and robustness produced by all algorithms, the performance of the two different versions of RDEL and

Table 7 Comparison between RDEL and RDEL with different versions and Basic DE on 50D problems.

Fcn	DE		RDEL-1		RDEL-2		RDEL	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
f_1	**1.57E−27**	**4.44E−28**	**2.05E−28**	**3.98E−28**	**1.51E−23**	**1.88E−23**	7.79E−29	1.12E−28
	0	0	1	–				
f_2	**1.12E−05**	**3.13E−05**	1.55E+03	4.28E+03	8.90E−01	8.09E−01	4.58E−01	5.12E−01
	−1	1	1	–				
f_3	4.88E+01	4.11E+01	6.62E−01	1.55E+00	7.86E−04	5.13E−04	7.93E−06	2.82E−05
	1	1	1	–				
f_4	**8.35E+01**	**8.44E+01**	9.67E+03	6.13E+03	2.68E+02	1.22E+02	2.66E+02	2.34E+02
	−1	1	−1	–				
f_5	4.16E−01	5.35E−01	**6.53E−15**	**1.67E−15**	**1.57E−13**	**2.36E−13**	**3.90E−15**	**1.08E−15**
	1	0	1	–				
f_6	3.61E−01	6.60E−01	**3.55E−15**	**9.83E−16**	**2.94E−10**	**1.51E−10**	**3.67E−15**	**6.48E−16**
	1	0	1	–				
f_7	8.04E−03	1.06E−02	**0**	**0**	**0**	**0**	**0**	**0**
	1	0	0	–				
f_8	8.53E−03	9.15E−03	**4.88E−17**	**1.50E−16**	**1.11E−10**	**1.07E−10**	**0**	**0**
	1	0	1	–				
f_9	4.03E+01	8.39E+00	2.22E+02	1.60E+00	0	0	**0**	**0**
	1	1	0	–				
f_{10}	2.98E+02	1.15E+02	3.04E+02	3.02E+01	6.24E+01	1.54E+02	**4.40E+01**	**9.03E+00**
	1	1	1	–				
f_{11}	8.09E+01	4.56E+01	1.07E+01	2.99E+00	3.33E−02	1.82E−01	**0**	**0**
	1	1	1	–				
f_{12}	2.33E+03	5.50E+02	2.76E+01	5.09E+01	1.81E−11	0	**1.82E−11**	**0**
	1	1	0	–				
f_{13}	**6.88E−30**	**8.04E−30**	**6.67E−30**	**4.10E−30**	**2.19E−16**	**7.63E−16**	**1.72E−30**	**3.55E−30**
	0	0	0	–				
f_{14}	1.12E+01	2.53E+01	5.92E+00	4.63E+00	1.81E+00	7.81E−01	**1.15E+00**	**3.71E−01**
	1	1	0	–				

conventional DE algorithm are investigated based on the 50-dimensional functions. The parameters used are fixed as same as those in Section 4.2.

The overall comparison results of the RDEL algorithm against its versions and conventional DE algorithm are summarized in Table 7. Furthermore, in order to analyze the convergence behavior of each algorithm compared, the convergence characteristics in terms of the best fitness value of the median run of each algorithm for selected functions $f_3, f_4, f_6, f_8, f_{10}, f_{12}$ and f_{13}–f_{14} with dimension 50 is illustrated in Fig. 1. Indeed, the presented results in Table 7 explain that the conventional DE only obtains better results on unimodal functions f_1, f_2, f_4 and composition function f_{13} while it completely fails on all multi-modal problems. Obviously, from the t-test results, it can be detected that RDEL is inferior to, equal to, superior to DE/rand/1/bin ($F = 0.5$, $Cr = 0.9$) in 2, 2 and 10 problems, respectively. Thus, the RDEL is much more efficient, robust and effective than DE/rand/1/bin ($F = 0.5$, $Cr = 0.9$) in finding high quality solution for real parameter optimization problems with dimensions 50. Accordingly, it can be deduced that the superiority, efficiency and the remarkable performance of the RDEL algorithm is due to the proposed modifications. On the other hand, concretely, it can be seen that the RDEL-1 algorithm significantly outperforms the conventional DE algorithm on functions (f_3, f_5–f_8, f_{11}, f_{12} and f_{14}). Thus, it is worth mentioning that the RDEL-1

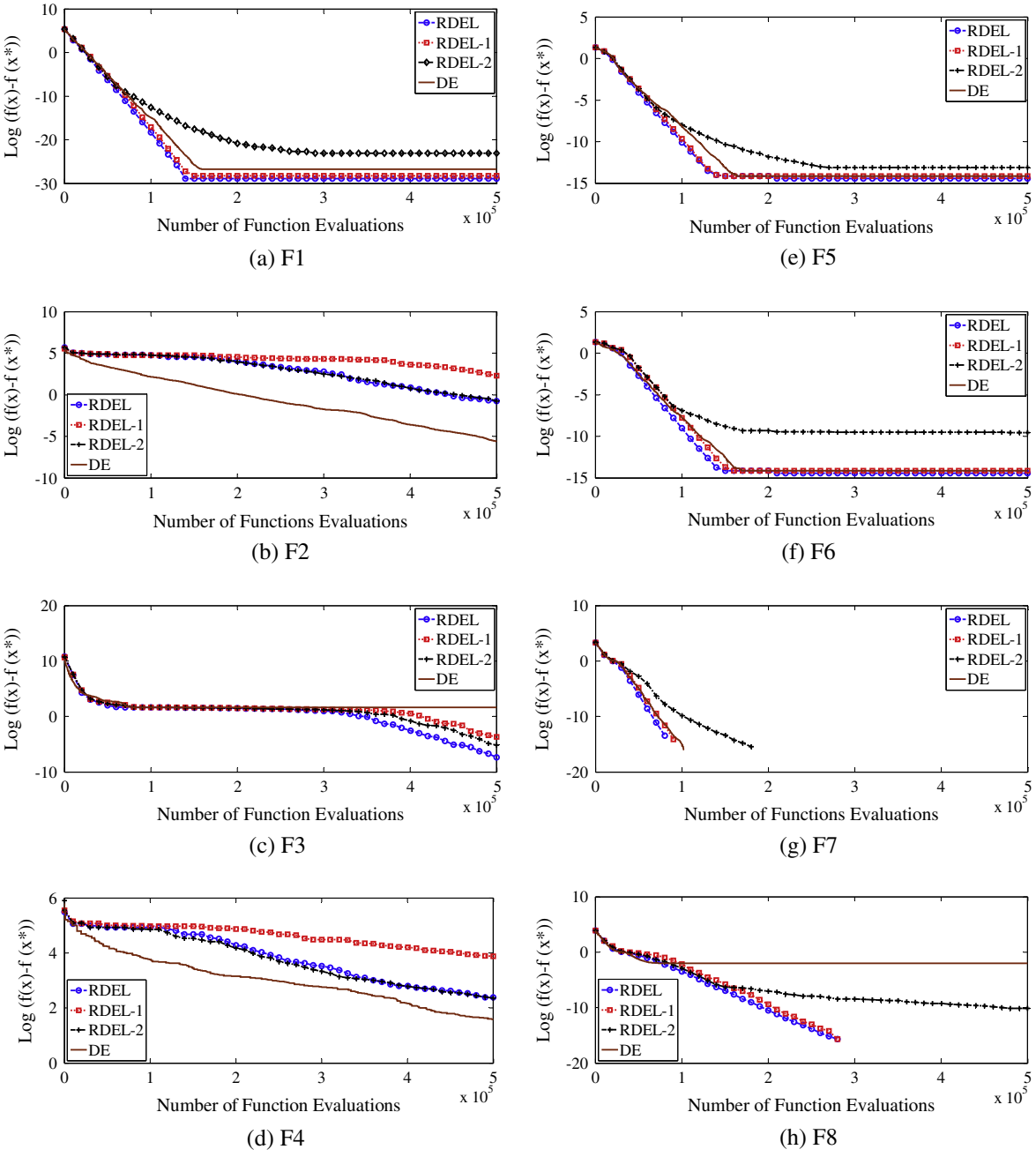

Figure 1 Convergence graph (median curves) of RDEL, RDEL-1, RDEL-2 and DE on 50-dimensional test functions f_1–f_{14}.

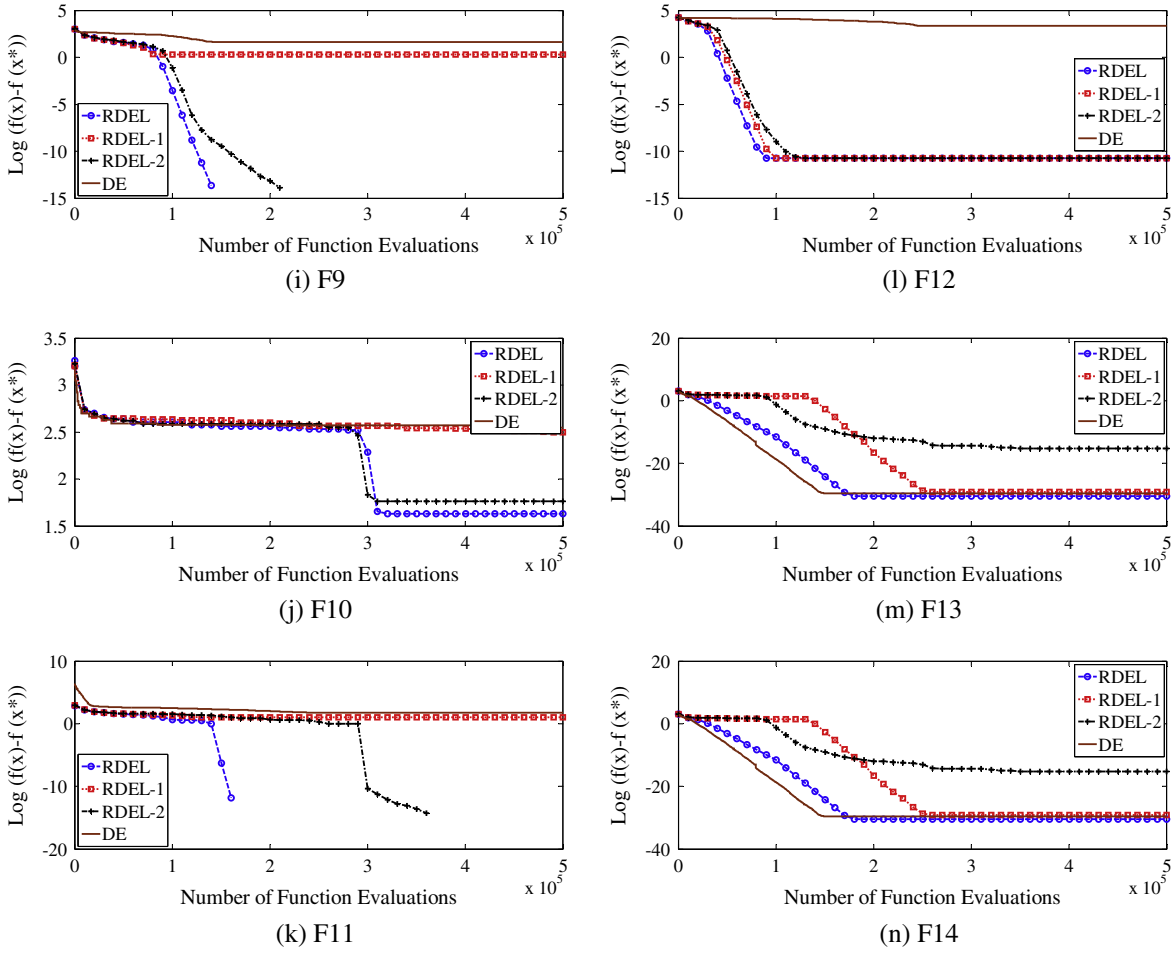

Fig 1. *(continued)*

algorithm considerably improves the final solution quality. On the contrary, conventional DE algorithm has performed better than RDEL-1 on problem (f_2, f_4, f_9 and f_{10}). Besides, they exhibit equal performance on 2 functions (f_1 and f_{13}). Therefore, it is clearly observed that the proposed local mutation scheme with the proposed increased exponent crossover rule considerably balances the global exploration ability and local exploitation tendency for the majority of functions with different characteristics much more than standard DE algorithm, namely, DE/rand/1/bin strategy with ($F = 0.5$, $Cr = 0.9$). Indeed, from the t-test results, RDEL is inferior to, equal to, superior to RDEL-1 algorithm in 0, 6 and 8 problems, respectively. On the other hand, it can be seen that by embedding the random mutation and modified (BGA) mutation in RDEL-1 algorithm, extreme and ultimate improvement in the performance of RDEL-2 has been detected and achieved on functions (f_2–f_4, f_9–f_{12} and f_{14}). However, the joining of the random mutation and modified (BGA) mutation in RDEL-1 algorithm has a slight negative influence on the final solution quality and the convergence speed of RDEL-2 algorithm on functions (f_1, f_5–f_8). Thus, from the t-test results, it can be observed that RDEL is inferior to, equal to, superior to RDEl-2 algorithm in 0, 9 and 5 problems, respectively. Consequently, RDEL algorithm is always either better or equal. Overall, it can be concluded that the excellent performance of the RDEL depends on the integration between its components and it is superior to conventional DE, RDEL-1, RDEL-2. Additionally, the convergence graph in Fig. 1, illustrates that RDEL, RDEL-1 and RDEL-2 algorithms converge to better or global solution faster than conventional DE, in presented cases with exception to function f_4 where basic DE converges faster than all compared algorithms. Accordingly, the main benefits of the proposed modifications are the remarkable balance between the exploration capability and exploitation tendency through the optimization process that 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.

5. Conclusion and future work

In order to promote the exploitation capability and speed up the convergence during the evolutionary process of the conventional DE algorithm, a restart Differential Evolution algorithm with novel local search mutation strategy for solving global numerical optimization problems over continuous space is proposed in this paper. Inspired by PSO algorithm, the proposed algorithm introduces a novel local mutation rule based on the position of the best and the worst individuals among the entire population of a particular generation. The mutation rule is combined with the basic mutation strategy through a

linear decreasing probability rule. The proposed mutation rule is shown to enhance the local search capabilities of the basic DE and to increase the convergence speed. Additionally, an exponent increased crossover probability rule 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. The proposed RDEL algorithm has been compared with 6 recent state-of-the-art parameter adaptive differential evolution variants over a suite of 14 bound constrained numerical optimization problems. The experimental results and comparisons have shown that the RDEL algorithm performs better in unconstrained optimization problems with different types, complexity and dimensionality; it performs better with regard to the search process efficiency, the final solution quality, the convergence rate, and robustness, when compared with other algorithms. Current research efforts focus on applying the algorithm to solve high dimensions or large scale global optimization problems as well as solving practical engineering optimization problems and real world applications. Finally, it would be very interesting to propose a self-adaptive RDEL version.

Appendix A

(1) Shifted sphere function

$$f_1(x) = \sum_{i=1}^{D} z_i^2, z = \mathbf{x} - \mathbf{o}, \mathbf{o} = [o_1, o_2, \ldots, o_D]$$
: the shifted global optimum

(2) Shifted Schwefel's Problem 1.2

$$f_2(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j \right)^2, \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{o} = [o_1, o_2, \ldots, o_D]$$
: the shifted global optimum

(3) Rosenbrock's function

$$f_3(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

(4) Shifted Schwefel's Problem 1.2 with noise in fitness

$$f_4(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j \right)^2 (1 + 0.4|N(0,1)|), \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{o}$$
$$= [o_1, o_2, \ldots, o_D] : \text{the shifted global optimum}$$

(5) Shifted Ackley's function

$$f_5(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} z_i^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi z_i) \right) + 20 + e, \mathbf{z}$$
$$= \mathbf{x} - \mathbf{o}, \mathbf{o} = [o_1, o_2, \ldots, o_D] : \text{the shifted global optimum}$$

(6) Shifted rotated Ackley's function

$$f_6(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} z_i^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi z_i) \right) + 20 + e, \mathbf{z} = \mathbf{M}(\mathbf{x} - \mathbf{o}), \text{cond}(\mathbf{M}) = 1, \mathbf{o}$$
$$= [o_1, o_2, \ldots, o_D] : \text{the shifted global optimum}$$

(7) Shifted Griewank's function

$$f_7(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}} \right) + 1, \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{o}$$
$$= [o_1, o_2, \ldots, o_D] : \text{the shifted global optimum}$$

(8) Shifted rotated Griewank's function

$$f_8(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}} \right) + 1, \mathbf{z} = \mathbf{M}(\mathbf{x} - \mathbf{o}), \ \text{cond}(\mathbf{M})$$
$$= 3, \mathbf{o} = [o_1, o_2, \ldots, o_D] : \text{the shifted global optimum}$$

(9) Shifted Rastrigin's function

$$f_9(x) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10), \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{o}$$
$$= [o_1, o_2, \ldots, o_D] : \text{the shifted global optimum}$$

(10) Shifted rotated Rastrigin's function

$$f_{10}(x) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10), \mathbf{z} = \mathbf{M}(\mathbf{x} - \mathbf{o}), \ \text{cond}(\mathbf{M})$$
$$= 2, \mathbf{o} = [o_1, o_2, \ldots, o_D] : \text{the shifted global optimum}$$

(11) Shifted non-continuous Rastrigin's function

$$f_{11}(x) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10), y_i$$
$$= \begin{cases} z_i & |z_i| < 1/2 \\ round(2z_i)/2 & |z_i| \geqslant 1/2 \end{cases}$$
for $i = 1, 2, \ldots, D$, $\mathbf{z} = (\mathbf{x} - \mathbf{o})$,
$\mathbf{o} = [o_1, o_2, \ldots, o_D]$: the shifted global optimum

(12) Schwefel's function

$$f_{12}(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(|x_i|^{1/2})$$

(13) Composition function 1 (CF1) in [35]
The function $f_{13}(x)$ (CF1) is composed by using 10 sphere functions. The global optimum is easy to find once the global basin is found.

(14) Composition function 6 (CF6) in [35]
The function $f_{14}(x)$ (CF6) is composed by using 10 different benchmark functions, i.e. 2 rotated Rastrigin's functions, 2 rotated Weierstrass functions, 2 rotated Griewank's functions, 2 rotated Ackley's functions and 2 rotated Sphere functions.
Where $\vec{o}$ indicates the position of the shifted optima, M is a rotation matrix, and cond (M) is the condition number of the matrix.

References

[1] Storn R, Price K. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI; 1995. <http://.icsi.berkeley.edu/~storn/litera.html>.

[2] 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.

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

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

[5] 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.

[6] Lampinen J, Zelinka I. On stagnation of the differential evolution algorithm. In: Matoušek R, Ošmera P, editors. Proceedings of Mendel 2000, 6th international conference on soft computing; 2000. p. 76–83.

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

[8] Xu Y, Wang L, Li L. An effective hybrid algorithm based on simplex search and differential evolution for global optimization. In: International conference on intelligent computing; 2009. p. 341–50.

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

[10] Gämperle R, Müller SD, Koumoutsakos P. A parameter study for differential evolution. In: Grmela A, Mastorakis NE, editors. Advances in intelligent systems, fuzzy systems, evolutionary computation. Interlaken, Switzerland: WSEAS Press; 2002. p. 293–8.

[11] Rönkkönen J, Kukkonen S, Price KV. Real-parameter optimization with differential evolution. Proc IEEE cong evol comput, vol. 1. Washington, DC: IEEE Computer Society; 2005. p. 506–13.

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

[13] Omran MGH, Salman A, Engelbrecht AP. Self-adaptive differential evolution, in: Computational intelligence and security, PT 1, proceedings lecture notes in artificial intelligence; 2005. p. 192–99.

[14] Zaharie D. Control of population diversity and adaptation in differential evolution algorithms. In: Matousek R, Osmera P, editors. Proceedings of Mendel 2003, 9th international conference on soft computing; 2003, p. 41–6.

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

[16] Zhang JQ, Sanderson AC. JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 2009;13(5):945–58.

[17] Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF. Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 2011;11(2):1679–96.

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

[19] 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.

[20] Kaelo P, Ali MM. A numerical study of some modified differential evolution. Eur J Oper Res 2006;196(3):1176–84.

[21] Price KV. An introduction to differential evolution. In: Corne D, Dorgio M, Glover F, editors. New ideas in optimization. London, UK: McGraw-Hill; 1999. p. 79–108.

[22] Mohamed AW, Sabry HZ. Constrained optimization based on modified differential evolution algorithm. Inf Sci 2012;194:171–208.

[23] Liu G, Li Y, Nie X, Zheng H. A novel clustering-based differential evolution with 2 multi-parent crossovers for global optimization. Appl Soft Comput 2012;12(2):663–81.

[24] Omran MGH, Engelbrecht AP, Salman A. Bare bones differential evolution. Eur J Oper Res 2009;196:128–39.

[25] Piotrowski AP, Napiorkowski JJ, Kiczko A. Differential evolution algorithm with separated groups for multi-dimensional optimization problems. Eur J Oper Res 2012;216:33–46.

[26] Mohamed AW, Sabry HZ, Abd-Elaziz T. Real parameter optimization by an effective differential evolution algorithm. Egypt Inform J 2013;14:37–53.

[27] Ali M, Siarry P, Pant M. An efficient differential evolution based algorithm for solving multi-objective optimization problems. Eur J Oper Res 2012;2012(217):404–16.

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

[29] Elsayed S, Sarker R, Essam D. Self-adaptive differential evolution incorporating a heuristic mixing of operators. Comput Optim Appl 2012:1–20.

[30] Elsayed S, Sarker R, Ray T. Parameters adaptation in differential evolution. In: IEEE congress on evolutionary computation, Brisbane; 2012b. p. 1–8.

[31] Triguero I, Derrac J, Garci S, Herrera F. Integrating a differential evolution feature weighting scheme into prototype generation. Neurocomputing 2012;97:332–43.

[32] Ali M, Pant M, Abraham A. Improving differential evolution algorithm by synergizing different improvement mechanisms. ACM Trans Auton Adapt Syst 2012;7(2):1–32.

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

[34] Feoktistov V. Differential evolution: in search of solutions. Berlin, Germany: Springer-verlag; 2006.

[35] Liang JJ, Suganthan PN, Deb K. Novel composition test functions for numerical global optimization. In: Proceedings of IEEE swarm intelligence symposium, Pasadena, CA; 2005. p. 68–75.

[36] Shang YW, Qiu YH. A note on the extended Rosenbrock function. Evol Comput 2006;14(1):119–26.