

An adaptive differential evolution algorithm with restart for solving continuous optimization problems

JEERAYUT WETWEERAPONG PIKUL PUPHASUK

Department of Mathematics, Faculty of Science

Khon Kaen University, Khon Kaen, 40002

THAILAND

wjeera@kku.ac.th ppikul@kku.ac.th

Abstract: A new adaptive differential evolution algorithm with restart (ADE-R) is proposed as a general-purpose method for solving continuous optimization problems. Its design aims at simplicity of use, efficiency and robustness. ADE-R simulates a population evolution of real vectors using vector mixing operations with an adaptive parameter control based on the switching of two selected intervals of values for each scaling factor and crossover rate of the basic differential evolution algorithm. It also incorporates a restart technique to supply new contents to the population to prevent premature convergence and stagnation. The method is tested on several benchmark functions covering various types of functions and compared with some well-known and state-of-art methods. The experimental results show that ADE-R is effective and outperforms the compared methods.

Keywords: Continuous optimization, optimization method, adaptive differential evolution algorithm, adaptive parameter control, restart technique

Received: January 21, 2020. Revised: June 2, 2020. Accepted: June 14, 2020. Published: June 25, 2020.

1 Introduction

Solving continuous optimization problems is an important task in engineering, economics and applied sciences. Difficult optimization problems often occur in computational systems involving several decision variables. For example, clustering data vectors in data science requires optimized conditions of many representative clusters [1,2], and training artificial neural networks needs optimized weights to classify the input data in supervised learning [3,4]. Such continuous optimization problems usually consist of high dimensional objective functions which are nonlinear and may contain large numbers of local optima. Thus, the efficient optimization methods become indispensable tools to handle the problems. These solution methods can be divided into two groups: local methods and global methods [5]. The local methods use the derivatives (or some analytical approximations of directions) and require the initial approximate solutions, which makes them sensitive to the initial guesses and limits their solving ability for general applications. To address this issue, many researchers have proposed the global methods or the stochastic direct search methods as the alternative approach. The available global methods include population-based methods, swarm-based methods, and most of nature-inspired methods [6].

In this study, we focus on the differential evolution algorithm (DE) which is a popular population-based method [7]. DE has been shown to be an efficient method but its performance depends on the control parameters and the problems to be solved [8, 9]. The aim of this work is to improve the performance of the basic DE by incorporating a suitable, adaptive parameter control and a restart technique. The obtained adaptive differential evolution algorithm is called ADE-R. It combines two main features of the adaptive switching of two selected intervals of values for each scaling factor and crossover rate of the basic DE, and a simple restart to prevent premature convergence and stagnation. The enhanced performance of the proposed ADE-R is empirically shown through extensive comparisons with several well-known methods on various benchmark functions.

2. Literature review

2.1 The basic differential evolution algorithm

Differential evolution algorithm is proposed by Storn and Price in the years 1995-1997 [7,10]. Due to its simple structure and efficiency, it has attracted many practitioners and researchers during the past two decades. A large number of modifications, improvements and variants have been proposed and

tested [8,11-13]. Like genetic and evolutionary algorithms that have been known many years before [14], DE consists of three basic population operations: mutation, crossover and selection. Its main distinguishing features are the differential mutation and the combined binomial crossover to each target vector to obtain a trial vector for comparing in the greedy selection. First, a population of NP real vectors are initialized by uniform random distribution in the search ranges. For each generation and each target vector x_i , three different random population vectors $x_{r_1}, x_{r_2}, x_{r_3}$, which are also different from the target vector, are used to generate a mutant vector v by adding the scaled difference of two vectors to another one: $v = x_{r_1} + F(x_{r_2} - x_{r_3})$ where F is the scaling factor. Then, some components of the target vector are exchanged with those of the mutant vector according to the crossover rate C to produce the trial vector. The target vector will be replaced by the trial vector if it produces a better solution. This description shows the three important control parameters of the basic DE: the population size NP , the scaling factor F and the crossover rate C . These control parameters have been found to affect the DE's performance greatly and in order to successfully solve a specific problem, a user needs to supply the suitable values [15-19]. Moreover, different parameter settings may be required for different stages of optimization. To overcome the problems, various mechanisms for setting or adjusting the control parameters and the adaptive versions of the differential evolution algorithm have been designed and proposed [20, 21].

2.2 The adaptive differential evolution algorithms

The review of some well-known adaptive differential evolution variants are given. Some of them are considered state-of-art methods and will be used to compare their performances with that of our proposed ADE-R. The concepts of parameter control have been already widely studied for the evolutionary algorithm [22]. They can be classified into three groups: deterministic parameter control, adaptive parameter control and self-adaptive parameter control. Deterministic parameter control alters the strategy parameters by some deterministic rule without using any feedback from the search while the adaptive parameter control monitors and utilizes the feedback from the search. Self-adaptive parameter control is a higher level of an adaptive control which encodes some information into some components of the individual vectors and utilizes the evolution process to alter and promote the strategy

parameters. In 2005, Liu and Lampinen proposed a fuzzy adaptive differential algorithm (FADE) by using fuzzy logic controllers as the parameter control for DE [23]. FADE uses the authors' designed fuzzy sets and fuzzy rules to dynamically control the parameters F and C . Compared with a static DE with $F=0.9$ and $C=0.9$, FADE shows a better convergence speed, particularly for high-dimensional test functions. In 2006, Brest et al. presented a DE version with self-adaptive control parameter settings, which is called jDE [24]. The control parameters F and C are adjusted by means of evolution and are applied at the individual level. The values $F_l=0.1$ and $F_u=0.9$ are set and a new value F takes values $F_l = rand() \cdot F_u$ in the range of $[0.1, 1]$ in a random manner with the probability $t_1=0.1$. Similarly, C takes new values in $[0, 1]$ in a random manner with the probability $t_2=0.1$. The new values of F and C are obtained before the mutation and crossover are performed and the better parameter values are propagated by the selection operations. They tested jDE on 25 benchmark functions and showed that it outperformed overall the basic DE with static values $F = 0.5$ and $C = 0.9$. It was also shown to outperform FADE and other two variants of evolutionary programming algorithm.

Qin and Suganthan in 2005 [25], and Qin et al. in 2009 [26] proposed an adaptive DE called SaDE. It is a self-adaptive DE that gradually self-adapts both the trial vector generation strategies and their associate control parameters. Four well-known mutant vector generation strategies are used and the probabilities to choose each strategy are initialized to equal probability. The F and C values for each individual population vector are initialized by normal distributions $N(0.5, 0.3)$ and $N(0.5, 0.1)$, respectively. A learning period (LP) is set to update the center of the probability distribution of each strategy according to the records from the successful selection operations. Through the learning and evolution process, SaDE aims to produce and promote the good control parameters. On several test functions, they have shown that SaDE outperformed overall the basic DE algorithms with various static values of F and C . It was also shown to outperform FADE and slightly outperform overall jDE.

At about the same time, Zhang and Sanderson introduced an adaptive differential evolution with an optional external archive called JADE in 2009 [27]. JADE implements a new mutation strategy that utilizes some top best individuals and the optional archive operation that utilizes historical data to provide information of progress direction. These two operations aim to diversify the population and improve the convergence performance. The trial

vectors that fail in the selection process are added to the archive set of inferior solutions and used in the mutation to diversify and balance the use of best individuals, which also helps prevent getting trapped to a local minimum. For each generation and for each individual, the values F and C are randomly initialized by using the Cauchy distribution and normal distribution, respectively. Then at the end of each generation, the centers of distributions are updated according to the extracted information obtained from the set of successful values. JADE has two new parameters: p for the proportion of top best individuals used in the mutation (the greediness of the mutation strategy) and c for controlling the rate of parameter adaptation. Note that for JADE, the authors used larger sizes of populations for test functions at higher dimensions ($NP=30$ for $D \leq 10$, $NP=100$ for $D=30$ and $NP=400$ for $D=100$). Their simulation results show that JADE performs better than the classic DE with $F=0.5$ and $C=0.9$, the adaptive DE algorithms jDE and SaDE, the canonical particle swarm optimization, and other evolutionary algorithms from the literature in terms of convergence performance for a set of 20 benchmark problems. In addition, JADE with an external archive shows promising results for relatively high-dimensional problems.

In 2016, Leon and Xiong presented the greedy adapting differential evolution algorithm called GADE which adds the greedy adjustment of the control parameters F and C during the running of DE [28]. The greedy search is performed for better parameter assignments in successive learning periods in the whole evolutionary process. For each learning period, the current parameter assignment and its neighboring assignments are tested, used and propagated to the next learning period. The initial center values of F and C are set to 0.5. Then, the greedy search creates two neighborhoods $F-d_1$, $F+d_1$ and $C-d_2$, $C+d_2$ where $d_1=d_2=0.01$. The best of them is identified using the metric of progress rate and the learning period $LP=20$ (generations) is used to update the new center values. They tested GADE (with $NP=60$) on 25 benchmark functions in comparison with five other DE variants including the basic DE with $F=0.9$ and $C=0.9$, SaDE and JADE. It gives overall best performance in terms of the summation of relative errors.

Recently in 2019, Opara and Arabas [29] have presented a useful survey on theoretical results obtained so far for DE. The survey gives a comprehensive view on the understanding of the underlying mechanisms of DE and suggests some promising research directions. For the topic concerning the convergence proofs of DE, they

pointed out several important works. Hu et al. proved that the classical DE cannot guarantee global convergence on a class of multimodal functions [30]. When the whole population is within a sufficiently large attraction basin of a single local optimum, the population cannot leave this basin because of elitist selection. However, the convergence can be obtained by softening the selection in DE and adding a mutation strategy that samples from the whole feasible set [31]. There is also another way to introduce the global optimization property to DE by re-initializing the population, or its part, for every some fixed iterations [32]. This fact is utilized in the design of our proposed ADE-R method in which a restart technique is incorporated to enhance the convergence, and at the same time to prevent the premature convergence or the stagnation of the basic DE.

From the review of the selected adaptive DE variants, we can observe the structural concepts and the implementation techniques in designing an adaptive DE. Our proposed ADE-R aims at simplicity of use (both in the structure and implementation), efficiency and robustness. Its mutation and crossover strategies manage the allowed values from the two selected intervals for the control parameters F and C , respectively. The probabilities for choosing these parameters are controlled by a simple adaptive mechanism of counter updating, adjusting and resetting.

3 The design of the proposed ADE-R method

As a stochastic population-based method, the basic DE improves the population of the individuals by the mutation, crossover and selection operations with the three main control parameters NP , F and C that are kept fixed during the optimization process [7]. For our proposed ADE-R, a relatively small population size NP is used and also kept fixed. Using a small population size is aimed for smaller number of function evaluations and a faster convergence speed. However, evolving a population of small size will lead to premature convergence or stagnation easily due to limited population diversity [15-19]. To encounter these convergence problems, ADE-R incorporates a simple restart technique to periodically replace some of the worst individuals with the new generated ones to supply new contents to the population. The restart technique works together with the adaptive mechanism of the algorithm.

For each of the control parameters F and C , ADE-R implements a probability-based switching control to learn and bias toward the use of the suitable values

from the two selected intervals. The two intervals of values for F are $[0.5, 0.7]$ and $[0.7, 0.9]$, which are aimed to provide short and long step sizes F in the mutation. And the two intervals of values for C are $[0.0, 0.1]$ and $[0.9, 1.0]$, which are aimed to provide better crossover vectors for the cases of multimodal functions and nonseparable functions, respectively.

Without loss of generality, we consider the minimization of a real-valued objective function $f : [L, U]^D \rightarrow R$, where L and U are the bounds for each component of a vector in the domain of f . The ADE-R algorithm can be described as follows.

Step 1 Set $NP=20$; $NR=300$ and $PR=20$ where NP is the population size or the number of individual vectors of dimension D , NR is for the restart operation which restarts PR percent of the vectors (excluding the best vector solution) at every NR generations.

Step 2 Initialization: Initialize the population matrix $P=[x_i]$ where $x_i=[x_{i,j}]$ for $i=1,2,\dots,NP$ and $j=1,2,\dots,D$ and each component of the vector x_i is uniformly randomized in $[L,U]$. Evaluate all vectors x_i and record the current best vector $xbest$ and its best value $fbest$.

Step 3 Setting control parameters: Set the initial probabilities $pf_1=pf_2=0.5$ and the corresponding counters $nf_1=nf_2=0$ for mutation. And $pc_1=pc_2=0.5$ and the corresponding counters $nc_1=nc_2=0$ for crossover.

Step 4 For each generation, generate two uniform random numbers a_1 and a_2 between 0 and 1.

- If $a_1 < pf_1$, random F_1, F_2 in the range of $[0.5, 0.7]$. Otherwise, random F_1, F_2 in the range of $[0.7, 0.9]$.
- If $a_2 < pc_1$, random C in the range of $[0.0, 0.1]$. Otherwise, random C in the range of $[0.9, 1.0]$.

Step 5 Mutation: For each target vector x_i , choose five random integer indices r_1, r_2, r_3, r_4, r_5 (r_1 is also different from i , the rest can be equal) from 1 to NP . Generate a mutant vector v by the equation

$$v = x_{r_1} + F_1(x_{r_2} - x_{r_3}) + F_2(x_{r_4} - x_{r_5})$$

This equation samples the directions and step sizes F_1 and F_2 , and also indirectly corrects the contents of vector components by confirming the similar contents and reducing the differences. The values of F_1 and F_2 are generated, used and adapted for each generation.

Step 6 Crossover: Construct the trial vector u (for the target vector x_i) by replacing some components of x_i with the corresponding components of mutant vector v as follows:

$$u_j = \begin{cases} v_j & ; \text{rand}() \leq C \text{ or } j = IC, \\ x_{ij} & ; \text{otherwise} \end{cases}$$

where C is the crossover rate in the range of $[0, 1]$ and IC is a randomly fixed index from 1 to D (for this current crossover), which guarantees a change of at least one component. The function $\text{rand}()$ gives a uniform random number between 0 and 1 and generates a new random number for each $j=1,\dots,D$. For each generation, the value of C is also adaptively controlled in the same manner as those of F_1 and F_2 .

Step 7 Selection: Apply the greedy selection. The trial vector u will replace the associated target vector x_i if u is fitter ($f(u) < f(x_i)$). It also updates the $xbest$ and $fbest$ if $f(u) < fbest$. The dynamic updating of the target vector is used for a fast evolution process.

Step 8 Updating control parameters: Update pf_1, pf_2, pc_1 and pc_2 as follows. If a better solution found in the selection is generated with $a_1 < pf_1$ then increase $nf_1 := nf_1 + 1$; otherwise, increase $nf_2 := nf_2 + 1$. Similarly, if it is generated with $a_2 < pc_1$ then increase $nc_1 := nc_1 + 1$; otherwise, increase $nc_2 := nc_2 + 1$. If $(nf_1 + nf_2) \geq 100$ then adjust $nf_1 := nf_1 + 5$ and $nf_2 := nf_2 + 5$ (to prevent both of them from 0). Update $pf_1 = nf_1 / (nf_1 + nf_2)$ and $pf_2 = nf_2 / (nf_1 + nf_2)$. Similarly, if $(nc_1 + nc_2) \geq 100$ then adjust $nc_1 := nc_1 + 5$ and $nc_2 := nc_2 + 5$ (to prevent both of them from 0). Update $pc_1 = nc_1 / (nc_1 + nc_2)$ and $pc_2 = nc_2 / (nc_1 + nc_2)$. Reset the associate counters to 0 when the probabilities are updated.

Step 9 Restart: Apply the restart for every NR generations i.e., when $\text{modulo}(g, NR) = 0$ where g is the current generation. Randomly choose PR percent of population vectors (excluding the $xbest$) to be replaced with the new generated ones.

Step 10 Repeat steps (4) - (9) until reaching the stopping condition (the maximum number of function evaluations $nfmax$ or the value to reach VTR for $fbest$). Then, report the obtained best solution as an approximate solution of the problem.

The flowchart of the proposed ADE-R algorithm is illustrated in Fig. 1.

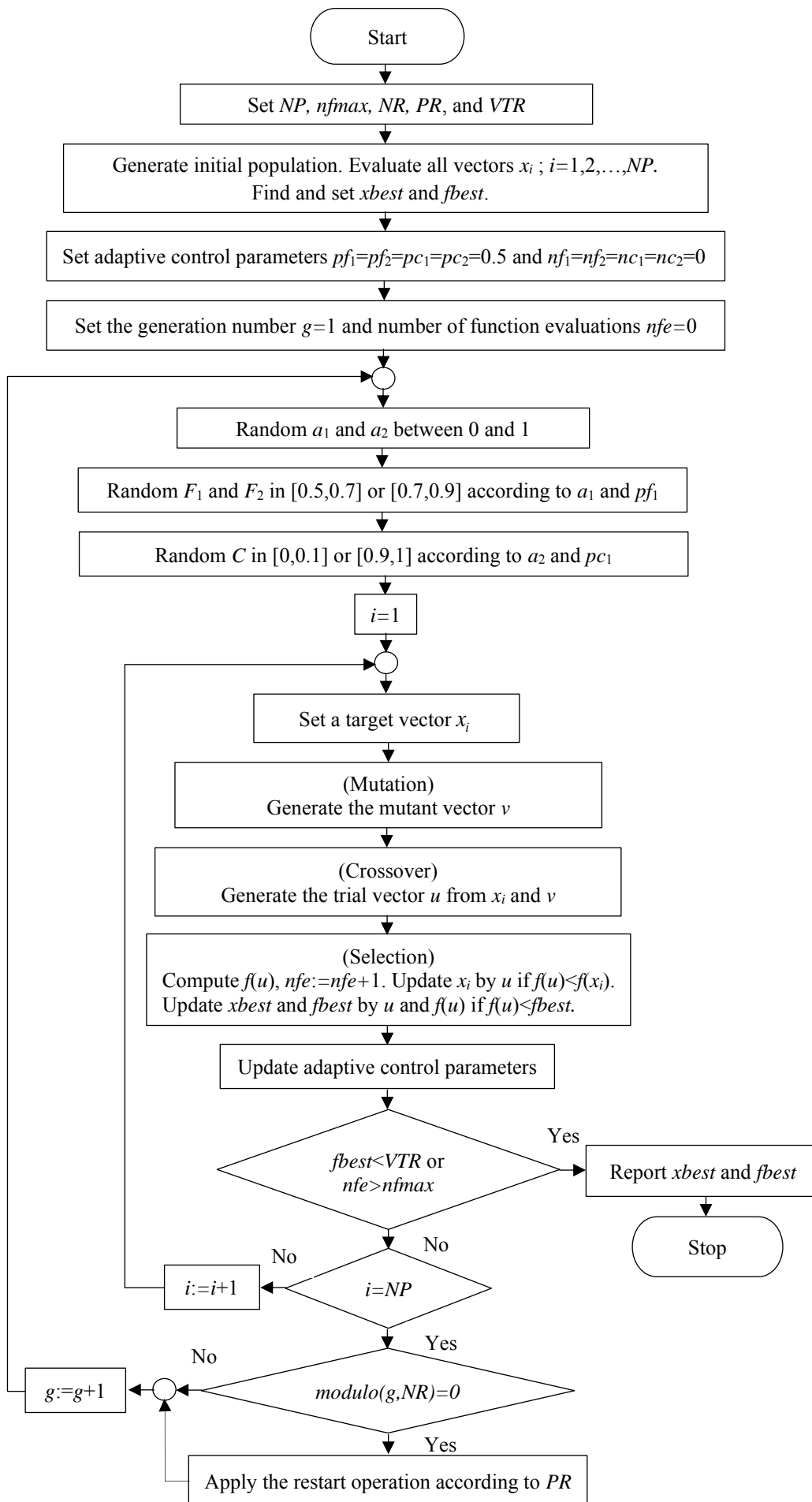


Fig. 1. Flowchart of the proposed ADE-R method.

Table 1. Test functions [25,30,31].

Function	Formulation	Type	Global optimum	Search range
Sphere	$f_1(x) = \sum_{i=1}^D x_i^2$	US	$\bar{0}$	$[-100,100]^D$
Schwefel 1.2	$f_2(x) = \sum_{i=1}^D (\sum_{j=1}^i x_j)^2$	US	$\bar{0}$	$[-100,100]^D$
Rosenbrock	$f_3(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	UN	$\bar{1}$	$[-100,100]^D$
Schwefel 2.22	$f_4(x) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	UN	$\bar{0}$	$[-100,100]^D$
Rastrigin	$f_5(x) = 10D + \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i))$	MS	$\bar{0}$	$[-5.2,5.2]^D$
Schwefel	$f_6(x) = 418.98288727243369D - \sum_{i=1}^D x_i \sin(\sqrt{ x_i })$	MS	$\overline{420.96}$	$[-500,500]^D$
Ackley	$f_7(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + \exp(1)$	MN	$\bar{0}$	$[-32,32]^D$
Griewank	$f_8(x) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	MN	$\bar{0}$	$[-600,600]^D$
Shifted sphere	$f_9(x) = \sum_{i=1}^D z_i^2; z = x - o$	US	o	$[-100,100]^D$
Shifted Schwefel 1.2	$f_{10}(x) = \sum_{i=1}^D (\sum_{j=1}^i z_j)^2; z = x - o$	US	o	$[-100,100]^D$
Shifted Rastrigin	$f_{11}(x) = 10D + \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i)); z = x - o$	MS	o	$[-5,5]^D$
Shifted Ackley	$f_{12}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)) + 20 + \exp(1); z = x - o$	MN	o	$[-32,32]^D$
Shifted Griewank	$f_{13}(x) = \frac{1}{4000} \sum_{i=1}^D z_i^2 - \prod_{i=1}^D \cos(\frac{z_i}{\sqrt{i}}) + 1; z = x - o$	MN	o	$[0,600]^D$

From the above description, the proposed ADE-R extends the basic DE by modifying and incorporating the following important mechanisms: the adaptation of the control parameters F and C based on the switching of two selected intervals of values for each of them, the mutation using five random population vectors which adds two scaled difference pairs to another vector, and the restart technique to periodically supply small amount of new contents to the evolving population.

4 Preliminary experiments and results

This section presents the experiments to find suitable parameters for the proposed ADE-R algorithm and show its computational complexity compared with that of the basic DE algorithm. The test functions used for the preliminary experiments and the comparison experiments in the next section are listed in Table 1. They cover all 4 important types of functions: unimodal and separable (US), unimodal and nonseparable (UN), multimodal and separable (MS), and multimodal and nonseparable (MN). Their formulations, types, global minima, and search

ranges are given. Note that the functions f_9 to f_{13} are the shifted versions of the preceding functions.

The experiments are carried out on an Intel® core i5 processor 2.0 GHz and 4 GB RAM. The ADE-R algorithm is coded in Scilab version 6.0.2, an open source software available at <http://www.scilab.org/>.

4.1 Finding suitable parameters for ADE-R

The first experiment tests the performances of ADE-R using various settings of NP , D , and NR on two representative functions: highly nonseparable Rosenbrock function and highly multimodal Griewank function. Aiming for a small population, the population sizes are varied as $NP = 10, 20, 30$. The dimensions and the fixed periods (generations) for applying a restart are varied as $D = 5, 10, 20$, and $NR = 200, 300, 400$. The $VTR = 10^{-10}$ and $nfmax = 20000D$ are used, and each configuration is performed 30 independent runs. The number of successful runs (NS), the mean of number of function evaluations (Mean nfe) and the percentage of standard derivation of the function evaluations (%SD) are reported in Table 2.

Table 2. Performances of ADE-R with different settings of NP , D , and NR at $VTR = 10^{-10}$ averaged over 30 independent runs for Rosenbrock and Griewank functions.

NP	D	NR	Rosenbrock		Griewank	
			NS	Mean nfe (%SD)	NS	Mean nfe (%SD)
10	5	200	28	33436.93(44.24)	12	9738.67(40.89)
		300	28	25174.86(56.41)	11	8905.73(25.52)
		400	28	24778.96(74.99)	10	7009.90(30.62)
	10	200	28	130456.75(24.66)	10	16860.00(26.51)
		300	29	121838.21(26.93)	14	14827.21(25.97)
		400	25	122064.32(22.45)	7	15096.43(18.29)
	20	200	22	310317.68(22.11)	21	14870.19(32.58)
		300	24	277871.96(24.79)	22	14178.10(21.95)
		400	23	274012.74(23.43)	25	13296.08(17.65)
20	5	200	30	16027.77(19.99)	28	30873.43(32.10)
		300	30	15820.83(17.55)	30	28703.83(18.91)
		400	30	16460.20(24.70)	30	27880.47(18.82)
	10	200	30	39617.13(11.02)	27	58748.82(29.83)
		300	30	40717.57(11.84)	30	47170.70(19.09)
		400	30	41832.17(15.81)	28	43502.14(19.28)
	20	200	30	131814.13(13.80)	27	57326.26(43.75)
		300	30	132702.83(12.79)	30	41908.13(33.05)
		400	30	131188.33(14.45)	30	37865.90(26.27)
30	5	200	30	28248.43(18.26)	30	52923.63(18.11)
		300	30	28742.17(16.26)	30	49291.07(21.73)
		400	30	27335.23(18.60)	30	46322.10(25.19)
	10	200	30	79231.57(10.26)	28	120920.86(19.13)
		300	30	77262.27(12.97)	30	78176.60(22.10)
		400	30	81026.80(20.23)	30	70493.87(18.01)
	20	200	30	241753.90(8.11)	30	126546.50(40.98)
		300	30	232716.00(10.12)	30	68580.10(25.95)
		400	30	231391.33(11.52)	30	60496.87(27.63)

From this table, the convergence results are considered first. The population size $NP=10$ is too small since it gives the successful runs NS less than 30 for all combinations of D and NR . Both $NP=20$ and 30 give good convergence results but using $NP=30$ requires more computations (nfe). For $NP=20$, we can observe that only $NR=300$ gives $NS=30$ for all cases of D . Thus, this setting ($NP=20$ and $NR=300$) is employed in the proposed ADE-R as described in section 3.

population-based methods including both the basic methods and the more advanced methods with the adaptive parameter controls. ADE-R and other methods are tested on the 13 selected test functions [25, 27, 28, 33, 34] and the performance comparisons are divided into four experiments.

The results of all comparison experiments are shown in Tables 3-7. In each table, the best values are indicated in bold. If the best function values or numbers of function evaluations of the compared

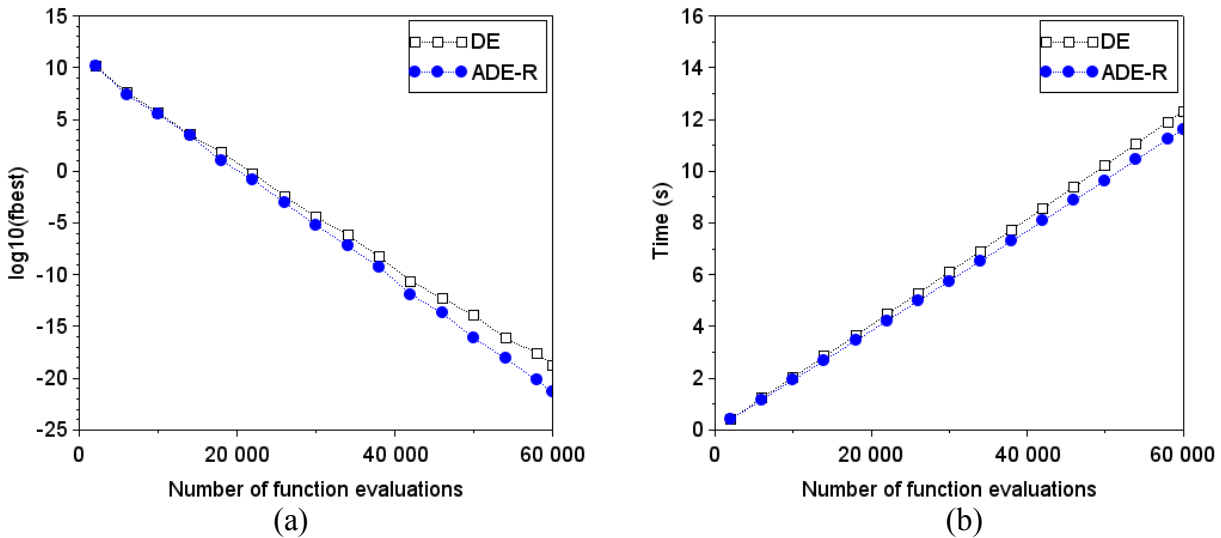


Fig. 2. (a) Convergence graphs and (b) Runtimes (second) of ADE-R and basic DE for the 50-dimensional Sphere function.

4.2 Computational complexity of ADE-R

This experiment compares the runtimes of the ADE-R and basic DE algorithms for optimizing the Sphere function at $D=50$. The ADE-R algorithm uses the setting $NP=20$ and $NR=300$ from the first experiment while the basic DE uses the setting $NP=50$, $F=0.5$ and $C=0.9$ as recommended in [7]. The $n_{fmax}=60000$ is set and at every 4000 function evaluations the current best function values and the runtimes for each algorithm are recorded. The comparisons of their convergence and runtime graphs are shown in Fig. 2. We can observe that ADE-R gives slightly faster convergence speed and also slightly less runtime. Thus, the proposed algorithm does not incur more computational complexity.

5 Comparison experiments and results

To assess the performance of the proposed ADE-R method, we compare it with several well-known

methods are not reported in the references, the notation “n/a” is used. If a method cannot succeed for some runs out of the total runs, the notation “-” is used. For the case that a method fails for all runs, the notation “--” is used.

In section 5.1, we implement the compared methods by using the recommended settings from the literature and use our stopping criterion $VTR=10^{-10}$ while the different stopping criteria are set according to the original papers for sections 5.2, 5.3, and 5.4. Except for the experiment in section 5.1, the results of all other methods are taken from the original papers.

5.1 Performance comparison of ADE-R with basic DE, PSO, and ABC algorithms

We compare the performances of ADE-R, basic differential evolution algorithm (DE), particle swarm optimization (PSO) and artificial bee colony algorithm (ABC). The experiment is conducted by setting $NP=50$ for all classic algorithms whereas $NP=20$ is used for ADE-R. The dimensions are varied

as $D=5, 10, 30, 50$ and the maximum number of function evaluations $n_{fmax} = 50000D$ is set for all test functions except the Rosenbrock function for which $n_{fmax} = 150000D$ is used. The value to reach $VTR = 10^{-10}$ is set and 50 independent runs are performed for each algorithm. The control parameters of basic DE are $F=0.5$ and $C=0.9$ as recommended in [7]. For PSO, the inertia weight w is started with 0.9 and decreased linearly to 0.4, and $C_1=C_2=2$ is used as recommended in [35]. The parameter for ABC is $limit = NP \times D$ as in [36]. For comparison, the number of successful runs (NS), the mean of number of function evaluations (Mean nfe), and the percentage of standard deviation of the function evaluations (% SD) are reported.

The results of comparing the performances of DE, PSO, ABC and ADE-R for minimizing the test function f_1-f_8 are shown in Table 3 and Table 4 and the convergence graphs of all methods for $D=30$ are illustrated in Fig. 3. For the ability and stability of solving each problem, the number of successful runs

NS out of the total 50 runs is considered first. It is evident that ADE-R outperforms all other methods. ADE-R solves all test functions at all dimensions $D=5, 10, 30, 50$ successfully for all 50 runs and gives the smallest numbers of function evaluations. The convergence graphs for $D=30$ in Fig. 3 clearly show its fast convergence speeds.

The basic DE can solve most of these test functions except for one highly nonseparable Rosenbrock function and two multimodal Rastrigin and Schwefel functions at high dimensions $D=30, 50$. Excluding these test functions, DE gives the convergence graphs that are very close to those of ADE-R.

PSO cannot solve most of the multimodal functions: Schwefel 1.2, Rastrigin, Schwefel, and Griewank, at high dimensions $D=10, 30, 50$. It can solve all the other test functions including the multimodal Ackley function. However, the convergence graph comparison shows that it has relatively slow convergence speeds.

Table 3. Performance comparison of DE, PSO, ABC and ADE-R at $VTR = 10^{-10}$ averaged over 50 independent runs for f_1-f_4 .

Functions	D	Statistics	DE	PSO	ABC	ADE-R	Significance
Sphere	5	NS Mean nfe(%SD)	50 6128.28(3.28)	50 46902.14(3.34)	50 11902.52(4.08)	50 4630.68(4.44)	+,+,+
	10	NS Mean nfe(%SD)	50 13090.36(3.27)	50 100998.12(2.11)	50 26500.72(2.84)	50 10259.34(3.40)	+,+,+
	30	NS Mean nfe(%SD)	50 38969.54(2.51)	50 358375.98(1.47)	50 88664.60(2.22)	50 34442.76(3.14)	+,+,+
	50	NS Mean nfe(%SD)	50 69677.70(5.71)	50 648755.48(1.21)	50 153280.84(2.06)	50 60437.66(2.38)	+,+,+
Schwefel 1.2	5	NS Mean nfe(%SD)	50 7477.16(3.35)	50 53117.42(2.93)	50 292925.40(2.92)	50 6717.48(12.55)	+,+,+
	10	NS Mean nfe(%SD)	50 21477.88(5.12)	50 130061.72(2.39)	0 --	50 19934.66(10.45)	+,+,+
	30	NS Mean nfe(%SD)	50 227246.24(6.55)	50 609435.24(1.90)	0 --	50 193841.64(6.48)	+,+,+
	50	NS Mean nfe(%SD)	50 673550.84(4.81)	50 1289121.40(1.49)	0 --	50 569999.36(5.08)	+,+,+
Rosenbrock	5	NS Mean nfe(%SD)	36 19146.72(79.19)	44 217331.33(6.16)	0 --	50 16641.94(27.34)	+,+,+
	10	NS Mean nfe(%SD)	48 179004.50(54.91)	30 419833.73(5.80)	0 --	50 41992.46(16.08)	+,+,+
	30	NS Mean nfe(%SD)	36 3270826.30(17.15)	33 1253743.60(7.23)	0 --	50 244203.76(16.99)	+,+,+
	50	NS Mean nfe(%SD)	0 --	25 1952210.60(7.52)	0 --	50 531859.28(7.97)	+,+,+
Schwefel 2.22	5	NS Mean nfe(%SD)	50 10161.42(3.23)	50 59720.16(2.03)	50 19896.14(3.21)	50 7245.86(3.77)	+,+,+
	10	NS Mean nfe(%SD)	50 21974.00(2.28)	50 116789.46(1.62)	50 42847.30(1.76)	50 15661.94(3.06)	+,+,+
	30	NS Mean nfe(%SD)	50 62175.04(2.06)	50 385935.66(1.47)	50 139942.50(1.27)	50 51409.22(1.77)	+,+,+
	50	NS Mean nfe(%SD)	50 98524.98(4.44)	50 688973.98(1.24)	50 240754.64(1.01)	50 89421.32(2.25)	+,+,+

ABC can solve the Sphere, Schwefel 2.22, and all of the multimodal functions. For these test functions, it has better convergence speeds than PSO. However, it cannot solve the Rosenbrock function for all dimensions and cannot solve the Schwefel 1.2 function at $D=10, 30, 50$.

For the performance comparison of DE and ADE-R, the results clearly show the benefits of the parameter adaptation of ADE-R. The basic DE using the fixed parameters $F=0.5$ and $C=0.9$, as widely recommended and accepted, can provide generally good performances for easy-to-moderate problems with low dimensions ($D \leq 10$), but not for the highly nonseparable and multimodal functions, especially at high dimensions.

The Welch t-test at a 0.05 level of significance shown in the last column is also used to compare the performances of ADE-R with those of DE, PSO, and ABC in this order. The values “+”, “0”, and “-” denote that ADE-R performs significantly better than, similarly to, and worse than a compared method. The value “++” denotes that the solutions of

ADE-R can reach $VTR=10^{-10}$ while those of the compared methods cannot.

5.2 Performance comparison of ADE-R with jDE, JADE and IABC algorithms

In this experiment, the performances of ADE-R are compared with those of jDE, JADE [27] and IABC [34] on the test functions f_1 - f_8 at dimension $D=30$. The n_{fmax} values are set between 5×10^4 to 2×10^6 depending on the original settings for the considered functions. For each algorithm, 50 independent runs are performed, and the mean of best function values (Mean fb) and the standard deviation (SD) are reported in Table 5. The results of JADE are from its original authors while the results of jDE are from the experimental runs of the JADE's authors. The results of both methods are reported in [27]. The results of IABC are compared with those of jDE and JADE in [34]. For difficult test functions, this experiment uses two settings of the maximum number of function evaluations (n_{fmax}) as in [27],

Table 4. Performance comparison of DE, PSO, ABC and ADE-R at $VTR = 10^{-10}$ averaged over 50 independent runs for f_5 - f_8 .

Functions	D	Statistics	DE	PSO	ABC	ADE-R	Significance
Rastrigin	5	NS	50	50	50	50	+,+,+
		Mean nfe(%SD)	21221.80(13.61)	54629.10(7.12)	17803.10(6.05)	6170.54(7.36)	
	10	NS	18	16	50	50	+,+,+
		Mean nfe(%SD)	80390.78(24.20)	128731.12(10.78)	39803.90(4.09)	13432.66(4.34)	
	30	NS	0	0	50	50	++,+,+
		Mean nfe(%SD)	--	--	140665.04(6.26)	54003.82(5.02)	
50	NS	0	0	50	50	++,+,+	
	Mean nfe(%SD)	--	--	259768.62(5.66)	119735.84(5.21)		
Schwefel	5	NS	50	42	50	50	+,+,+
		Mean nfe(%SD)	11020.82(11.79)	204021.26(5.68)	17854.04(5.82)	5657.38(6.07)	
	10	NS	49	0	50	50	+,+,+
		Mean nfe(%SD)	42320.12(16.83)	--	39273.44(4.36)	12211.36(4.90)	
	30	NS	0	0	50	50	++,+,+
		Mean nfe(%SD)	--	--	132172.08(3.85)	43238.80(2.45)	
50	NS	0	0	50	50	++,+,+	
	Mean nfe(%SD)	--	--	234691.00(4.20)	80506.92(3.83)		
Ackley	5	NS	50	50	50	50	+,+,+
		Mean nfe(%SD)	10602.18(2.73)	61491.54(2.45)	22892.26(2.42)	7985.04(4.10)	
	10	NS	50	50	50	50	+,+,+
		Mean nfe(%SD)	22202.78(2.25)	122731.42(1.48)	48612.60(2.06)	17211.06(3.23)	
	30	NS	49	50	50	50	+,+,+
		Mean nfe(%SD)	63378.69(3.05)	416518.56(1.49)	154410.00(1.02)	55635.70(2.32)	
50	NS	37	50	50	50	+,+,+	
	Mean nfe(%SD)	112384.08(4.43)	751469.76(1.24)	262882.40(1.32)	95548.08(1.77)		
Griewank	5	NS	43	6	49	50	+,+,+
		Mean nfe(%SD)	38394.93(14.61)	87352.50(42.61)	41340.61(27.10)	25422.72(19.98)	
	10	NS	13	0	43	50	+,+,+
		Mean nfe(%SD)	38748.77(25.21)	--	59967.16(29.67)	44236.26(21.61)	
	30	NS	36	0	50	50	+,+,+
		Mean nfe(%SD)	40416.31(3.55)	--	108302.52(6.49)	42939.32(15.61)	
50	NS	37	0	50	50	+,+,+	
	Mean nfe(%SD)	69884.35(5.83)	--	172363.84(4.77)	64940.34(6.51)		

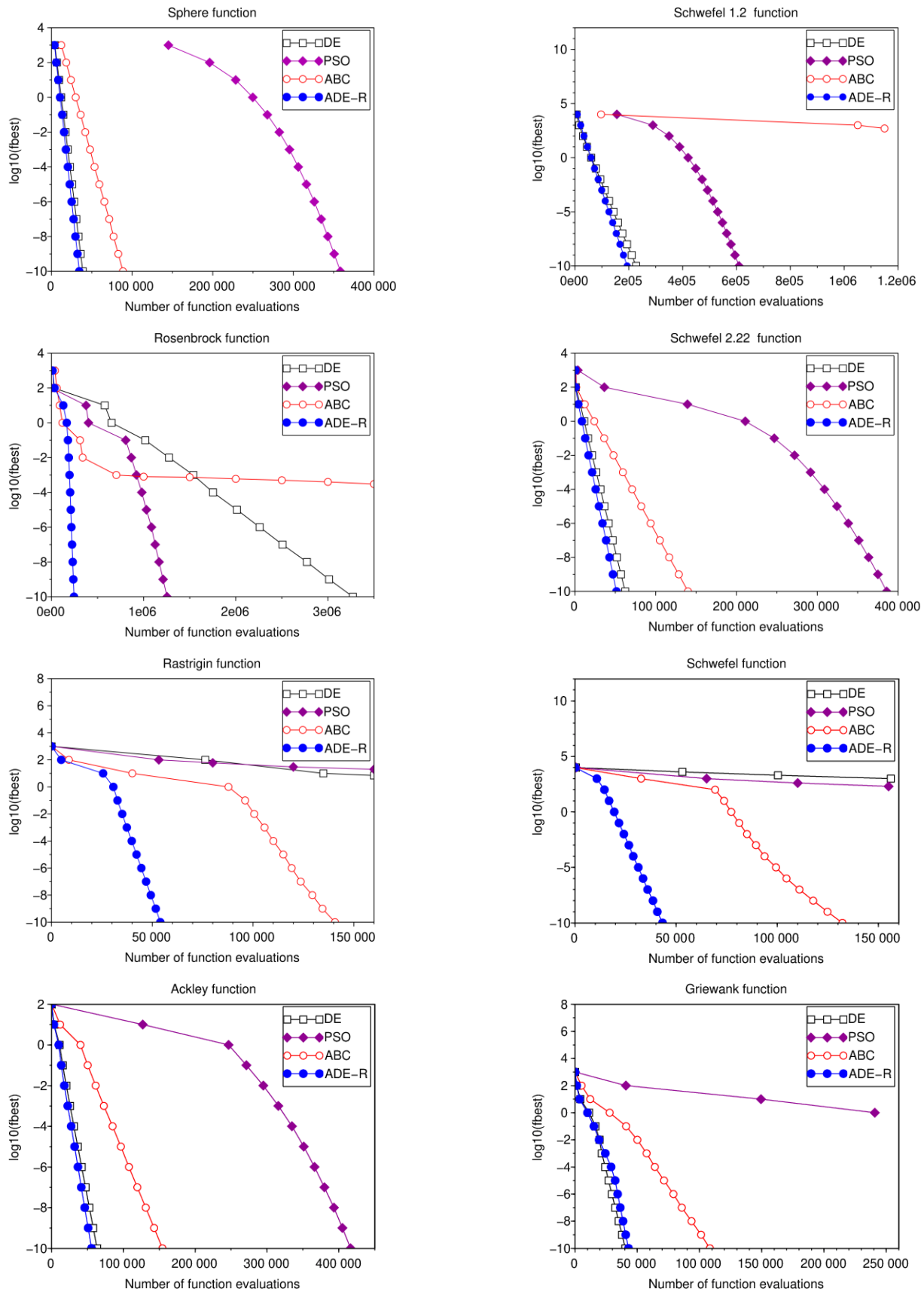


Fig. 3. Convergence graphs of DE, PSO, ABC and ADE-R for 30-dimensional functions.

Table 5. Performance comparison of jDE, JADE, IABC and ADE-R for 30-dimensional functions averaged over 50 independent runs.

Functions	nfmax	Statistics	jDE[27]	JADE[27]	IABC[34]	ADE-R	Significance
Sphere	150,000	Mean fb (SD)	0 (0)	0 (0)	0 (0)	0 (0)	+,0,0
Schwefel 1.2	500,000	Mean fb (SD)	5.20E-14 (1.10E-13)	0 (0)	0 (0)	0 (0)	+,0,0
Rosenbrock	300,000	Mean fb (SD)	1.30E+01 (1.40E+01)	8.00E-02 (5.60E-01)	n/a (n/a)	8.26E-13 (5.81E-12)	+,+,*
	2,000,000	Mean fb (SD)	8.00E-02 (5.60E-01)	8.00E-02 (5.60E-01)	4.75E-03 (4.22E-02)	0 (0)	+,+,+
Schwefel 2.22	200,000	Mean fb (SD)	0 (0)	0 (0)	0 (0)	0 (0)	0,0,0
Rastrigin	100,000	Mean fb (SD)	1.50E-04 (2.00E-04)	1.00E-04 (6.00E-05)	0 (0)	0 (0)	+,+,0
	500,000	Mean fb (SD)	0 (0)	0 (0)	n/a (n/a)	0 (0)	+,+,*
Schwefel	100,000	Mean fb (SD)	7.90E-11 (1.30E-10)	3.30E-05 (2.30E-05)	0 (0)	0 (0)	+,+,0
	900,000	Mean fb (SD)	0 (0)	0 (0)	n/a (n/a)	0 (0)	0,0,*
Ackley	50,000	Mean fb (SD)	3.50E-04 (1.00E-04)	8.20E-10 (6.90E-10)	3.87E-14 (8.52E-15)	1.98E-09 (1.07E-09)	+,,-
	200,000	Mean fb (SD)	4.70E-15 (9.60E-16)	4.40E-15 (0)	n/a (n/a)	7.55E-15 (0)	-,*,*
Griewank	50,000	Mean fb (SD)	1.90E-05 (5.80E-05)	9.90E-08 (6.00E-07)	0 (0)	1.24E-11 (5.89E-11)	+,+,-
	300,000	Mean fb (SD)	0 (0)	0 (0)	n/a (n/a)	0 (0)	0,0,*

and compares the final function values. We consider the obtained function value less than 10^{-20} as 0. Also note that the author of IABC selected only either one of the settings for those test functions. The last column of the table shows the performances of ADE-R compared with those of jDE, JADE, and IABC in this order using the Welch t-test at a 0.05 level of significance. The notation “*” denotes that the values of compared methods are not reported.

For the Sphere, Schwefel 1.2 and Schwefel 2.22 functions, all methods can successfully solve the problems with the means equal to 0 except for jDE on Schwefel 1.2. The ADE-R clearly outperforms on Rosenbrock function for both *nfmax* settings while all other methods perform quite poorly on this function. For Rastrigin and Schwefel functions ADE-R performs better than other methods when considering both *nfmax* settings.

For the other two multimodal Ackley and Griewank functions, all methods give comparable results. jDE performs quite poorly for the low *nfmax* settings while IABC performs slightly better for these settings on both functions. JADE performs slightly better on Ackley function for the high *nfmax* setting.

5.3 Performance comparison of ADE-R with SaDE algorithm

This experiment compares the performances of ADE-R and SaDE [26] on 8 test functions including 5 shifted ones at dimensions $D=10, 30$. Each method performs 30 independent runs at $VTR = 10^{-5}$. The NS and Mean nfe of both algorithms are presented in Table 6. The results of SaDE are not provided in [26] for particular test functions at some dimensions. ADE-R successfully solves all functions at both dimensions and clearly outperforms on Rosenbrock, Schwefel, Shifted Rastrigin and Shifted Griewank functions. Moreover, ADE-R performs much better than SaDE for Schwefel function at both dimensions and for Shifted Rastrigin at $D=10$ with each nfe value of ADE-R being roughly half of that of SaDE. For other functions, both SaDE and ADE-R give comparable results. SaDE performs slightly better on Schwefel 2.22, Shifted Sphere, Shifted Schwefel 1.2 and Shifted Ackley functions at high dimension $D=30$. However, it succeeds only 24 out of 30 runs for Shifted Griewank at $D=30$. From this, we may conclude that for overall cases, ADE-R performs better than SaDE.

Table 6. Performance comparison of SaDE and ADE-R for 10 and 30-dimensional functions at $VTR = 10^{-5}$ averaged over 30 independent runs.

Function	D	Statistics	SaDE [26]	ADE-R
Rosenbrock	10	NS Mean nfe	30 42446	30 34053
	30	NS Mean nfe	n/a n/a	30 221587
Schwefel 2.22	10	NS Mean nfe	n/a n/a	30 8698
	30	NS Mean nfe	30 25137	30 29739
Schwefel	10	NS Mean nfe	30 16663	30 8429
	30	NS Mean nfe	30 77920	30 30944
Shifted sphere	10	NS Mean nfe	30 8375	30 6537
	30	NS Mean nfe	30 20184	30 22426
Shifted Schwefel 1.2	10	NS Mean nfe	30 14867	30 13409
	30	NS Mean nfe	30 118743	30 125536
Shifted Rastrigin	10	NS Mean nfe	30 23799	30 9574
	30	NS Mean nfe	30 58723	30 42367
Shifted Ackley	10	NS Mean nfe	30 12123	30 9761
	30	NS Mean nfe	30 26953	30 32201
Shifted Griewank	10	NS Mean nfe	30 35393	30 25427
	30	NS Mean nfe	24 -	30 32643

5.4 Performance comparison of ADE-R with GADE algorithm

The performances of ADE-R and GADE [28] are compared on test functions with $D=30$. For each method, 30 independent runs are performed at $VTR = 10^{-8}$. The Mean fb and SD for both algorithms are presented in Table 7. The Welch t-test at a 0.05 level of significance is conducted and the results are shown in the last column. The table shows that ADE-R can successfully solve all 10 test functions whereas GADE can successfully solve only 7 functions. The function values obtained by GADE for Schwefel 1.2, Rosenbrock, and Shifted Schwefel 1.2 functions are quite poor. This indicates that ADE-R is more effective than GADE.

Table 7. Performance comparison of GADE and ADE-R for 30-dimensional functions at $VTR = 10^{-8}$ averaged over 30 independent runs.

Function	Statistics	GADE [28]	ADE-R	Significance
Sphere	Mean fb (SD)	0 (0)	0 (0)	0
Schwefel 1.2	Mean fb (SD)	3.09E-01 (7.00E+00)	0 (0)	+
Rosenbrock	Mean fb (SD)	2.54E+01 (5.26E+01)	0 (0)	+
Schwefel 2.22	Mean fb (SD)	0 (0)	0 (0)	0
Rastrigin	Mean fb (SD)	0 (0)	0 (0)	0
Ackley	Mean fb (SD)	0 (0)	0 (0)	0
Griewank	Mean fb (SD)	0 (0)	0 (0)	0
Shifted sphere	Mean fb (SD)	0 (0)	0 (0)	0
Shifted Schwefel 1.2	Mean fb (SD)	6.63E+00 (2.60E+01)	0 (0)	+
Shifted Rastrigin	Mean fb (SD)	0 (0)	0 (0)	0

6 An application of ADE-R for solving an engineering design problem

The ADE-R algorithm is applied to solve the cantilever beam design problem [37] which is related to the weight optimization of a cantilever beam with square cross section (see Fig. 4). The beam is rigidly supported at node 1, and there is a given vertical force acting at node 6. The design variables are the heights (or widths) of the different beam elements. The bound constraints are set as $0.01 \leq x_j \leq 100$. This problem can be written as follows:

$$\text{Min } f(\mathbf{x}) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5)$$

subject to

$$g(\mathbf{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0.$$

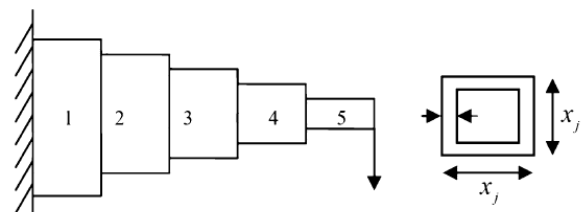


Fig. 4. Cantilever beam design [37].

Since the problem is constrained, the penalty technique is adopted to adjust the objective function. The penalty function is defined by

$$p(x) = 100[\min(0, g(x))].$$

Then the unconstrained objective function

$$F(x) = f(x) + p(x)$$

is used for ADE-R. Note that only the feasible solutions with $p(x)=0$ are considered at the end of the optimization process. The maximum number of generations $maxgen=500$ is set and 30 independent runs are performed.

The ADE-R algorithm gives the convergence results for all 30 runs with the max, mean, min, and SD of f_{min} values equal to 1.3412507, 1.340127, 1.3399566, and 0.0002793, respectively. The best solutions of the problem obtained by ADE-R, Cuckoo Search algorithm (CS) [37], and analytic method [38] are compared and presented in Table 8.

7 Discussion

In section 4.1, the effect of using different settings of population size (NP), dimension (D) and the fixed period of generations for a restart (NR) on the performance of ADE-R algorithm is verified. It has been shown that ADE-R using $NP = 20$ and $NR = 300$ can achieve good convergence results. It also shows that the adaptive control of the scaling factor and crossover rate values, and a restart technique allow the use of the relatively small population size. In section 4.2, the ADE-R with this setting has been shown to have the same computational complexity as that of the basic DE.

The performance comparisons of ADE-R and several other methods are conducted in section 5. In section 5.1, ADE-R has been shown to significantly outperform the basic DE, PSO, and ABC algorithms. In section 5.2, 5.3 and 5.4, ADE-R has also been shown to overall outperform four well-known adaptive DE variants.

In section 6, the ADE-R algorithm is applied to solve a constrained engineering design problem. By using the penalty technique to transform the problem into the unconstrained optimization problems, ADE-R can solve it very well and also gives high quality solutions.

The setting of $NP=20$ and $NR=300$ for the proposed ADE-R is only a general suggestion obtained in this study. Although it gives the satisfied performances on the comparison tests and a real-world application, applying the algorithm to other specific optimization problems may require a minor adjustment of these control parameters.

8 Conclusions

In this research, an efficient adaptive differential evolution algorithm named ADE-R is presented for solving a wide range of continuous optimization problems. It is aimed as a general-purpose optimization method which has a simple structure and is easy to implement. The parameter adaptation based on the switching of two selected interval values for each of the scaling factor and crossover rate of the basic DE, the associated mutation operation, and a restart technique are designed to work together to balance both intensifying and diversifying searches. The restart technique is particularly helpful in preventing premature convergence and stagnation. Extensive experiments show that ADE-R outperforms several well-known and state-of-art methods.

Acknowledgment

The authors would like to thank Department of Mathematics, Faculty of Science, Khon Kaen University for simulation equipment support.

References:

- [1] S. J. Nanda and G. Panda, A survey on nature inspired metaheuristic algorithms for partitional clustering, *Swarm and Evolutionary Computation*, 16, 2014, pp. 1-18.
- [2] A. José-García and W. Gómez-Flores, Automatic clustering using nature-inspired metaheuristics: A survey, *Applied Soft Computing*, 41, 2016, pp. 192-213.
- [3] L. Hamm, B. W. Broesen and M. T. Hagan, Comparison of stochastic global optimization methods to estimate neural network weights, *Neural Process Lett*, 26, 2007, pp. 145-158.

Table 8. The best solutions for the cantilever beam design problem obtained by ADE-R, CS algorithms and the analytic method.

Methods	x_1	x_2	x_3	x_4	x_5	f_{min}
CS [37]	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999
ADE-R	6.0160544	5.3102067	4.4949328	3.5003258	2.1521444	1.3399566
Analytic method [38]	6.0160159	5.3091739	4.4943296	3.5014750	2.15266533	1.339956367

- [4] A. P. Piotrowski, Differential evolution algorithms applied to neural network training suffer from stagnation, *Applied Soft Computing*, 21, 2014, pp. 382-406.
- [5] G. Venter, *Review of optimization techniques*, in: R. Blockley, S. Wei (eds.), *Encyclopedia of aerospace engineering*, Wiley and Sons, 2010.
- [6] I. Boussaïd, J. Lepagnot and P. Siarry, A survey on optimization metaheuristics, *Information sciences*, 237, 2013, pp. 82-117.
- [7] R. Storn and K. Price, Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces, *J Glob Optim*, 11(4), 1997, pp. 341-359.
- [8] S. Das and P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Trans Evol Comput*, 15(1), 2011, pp. 4-31.
- [9] S. Das, S.S. Mullick and P. Suganthan, Recent advances in differential evolution - An updated survey, *Swarm Evol. Comput*, 27, 2016, pp. 1-30.
- [10] R. Storn and K. Price, *Differential evolution a simple and efficient adaptive scheme for global optimization over continuous spaces*, Technical Report TR-95-012, ICSI, 1995.
- [11] R. Storn, Differential evolution research-trends and open questions, in: U. K. Chakraborty (ed.), *Advances in Differential Evolution*, Springer, 2008, pp. 1-31.
- [12] F. Neri and V. Tirronen, Recent advances in differential evolution: A survey and experimental analysis, *Artif Intell Rev*, 33, 2010, pp. 61-106.
- [13] T. Eltaieb and A. Mahmood, Differential Evolution: A Survey and Analysis, *Applied Sciences*, 8(10), 2018, pp. 1945.
- [14] T. Bäck and H. P. Schwefel, An overview of evolutionary algorithms for parameter optimization, *Evol. Comput*, 1(1), 1993, pp. 1-23.
- [15] J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm, in: R. Matouek, P. Omera (eds.), *Proceedings of Mendel 2000, 6th international conference on soft computing*, 2000, pp. 76-83.
- [16] R. Gamperle, S. D. Muller and P. Koumoutsakos, A parameter study for differential evolution, in: A. Gremla, N. E. Mastorakis (eds.), *Advances in intelligent systems, fuzzy systems, evolutionary computation*, WSEAS Press, 2002, pp. 293-298.
- [17] D. Zaharie, Critical values for control parameters of differential evolution algorithm, *Proceedings of the 8th international Mendel conference on soft computing*, 2002, pp. 62-67.
- [18] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, *Proceedings of the 9th international Mendel conference on soft computing*, 2003, pp. 41-46.
- [19] A. P. Piotrowski, Review of differential evolution population size, *Swarm Evol. Comput.*, 32, 2017, pp. 1-24.
- [20] T. C. Chiang, C. N. Chen and Y. C. Lin, Parameter control mechanisms in differential evolution: a tutorial review and taxonomy, *2013 IEEE symposium on differential evolution (SDE)*, 2013, pp. 1-8.
- [21] R. D. Al-Dabbagh, F. Neri, N. Idris, and M. S. Baba, Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy, *Swarm Evol. Comput.*, 43, 2018, pp. 284-311.
- [22] A. E. Eiben, R. Hinterding, and Z. Michalewicz, Parameter control in evolutionary algorithms, *IEEE Trans. Evol. Comput.*, 3(2), 1999, pp. 124-141.
- [23] J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm, *Soft Comput*, 9(6), 2005, pp. 448-462.
- [24] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *Evol. Comput. IEEE Trans.*, 10(6), 2006, pp. 646-657.
- [25] A. K. Qin and P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, *Proceedings of the 2005 IEEE congress on evolutionary computation*, 2, 2005, pp. 1785-1791.
- [26] A. K. Qin, V. L. Huang, and P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Trans Evol Comput*, 13(2), 2009, pp. 398-417.
- [27] J. Q. Zhang and A. C. Sanderson, JADE: adaptive differential evolution with optional external archive, *IEEE Trans. Evol. Comput.*, 13(5), 2009, pp. 945-958.
- [28] M. Leon and N. Xiong, Adapting differential evolution algorithms for continuous optimization via greedy adjustment of control parameters, *Journal of artificial intelligence and soft computing research*, 6(2), 2016, pp. 103-118.

- [29] K. Opara and J. Arabas, Differential Evolution: A survey of theoretical analyses, *Swarm Evol. Comput.*, 44, 2019, pp. 546-558.
- [30] Z. Hu, Q. Su, X. Yang, and Z. Xiong, Not guaranteeing convergence of differential evolution on a class of multimodal functions, *Appl. Soft Comput.*, 41, 2016, pp. 479-487.
- [31] Z. Hu, S. Xiong, Q. Su, and Z. Fang, Finite Markov chain analysis of classical differential evolution algorithm, *J. Comput. Appl. Math.*, 268, 2014, pp. 121-134.
- [32] Y. Wang and J. Zhang, Global optimization by an improved differential evolutionary algorithm, *Appl. Math. Comput.*, 188(1), 2007, pp. 669-680.
- [33] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*, Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, IIT Kanpur, India, 2005.
- [34] W. Gao and S. Liu, Improved artificial bee colony algorithm for global optimization, *Information Processing Letters*, 111, 2011, pp. 871-882.
- [35] Y. Shi and R. C. Eberhart, Empirical study of particle swarm optimization, *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, 1999, pp. 1945-1950.
- [36] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing*, 8, 2008, pp. 687-697.
- [37] A. H. Gandomi, X. S. Yang, and A. H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, *Engineering with Computers*, 29, 2013, pp. 17-35.
- [38] X. S. Yang, C. Huyck, M. Karamanoglu, and N. Khan, True global optimality of the pressure vessel design problem: A benchmark for bio-inspired optimization algorithms, *International Journal of Bio-Inspired Computation (IJBIC)*, 5(6), 2013, pp. 329-335.

**Creative Commons Attribution License 4.0.
(Attribution 4.0 International, CC BY 4.0).**

This article is published under the terms of the Creative Commons Attribution License 4.0 https://creativecommons.org/licenses/by/4.0/deed.en_US