Differential Evolution with Controlled Annihilation and Regeneration of Individuals and A Novel Mutation Scheme

Sudipto Mukherjee[1], Sarthak Chatterjee[1],
Debdipta Goswami[1], and Swagatam Das[2]

[1] Department of Electronics and Telecommunication Engineering
Jadavpur University, Kolkata, India
[2] Electronics and Communication Sciences Unit
Indian Statistical Institute, Kolkata, India

Abstract. Differential Evolution is a stochastic, population-based optimization algorithm, which grew out of the need to optimize real-parameter, real-valued functions. The Differential Evolution variant that we propose to describe in this paper modifies the mutation scheme of the variant DE/best/1. We propose a three tier mutation scheme, to be suitably carried out on selected sections of the population in question. Also, the proposed variant tries to lessen the myriad troubles posed by stagnation, which is a problem faced by all Differential Evolution algorithms. Our comparative studies indicate that the proposed variant is able to compete in a direction parallel to the state-of-the-art Differential Evolution variants like JADE and jDE.

Keywords: optimization, Differential Evolution(DE), novel mutation scheme, stagnation.

1 Introduction

Differential Evolution [6], [8]-[10] emerged in the late 1990s to serve the immediate need of mathematicians, engineers and technicians to solve real-world optimization problems. Over the years, cumulative research on differential evolution and its sundry varieties has reached an impressive state. Modifications have been proposed by introducing innovative mutation schemes, schemes that better implement crossover, and tuning parameters like the scale factor, F and the crossover ratio, Cr.

In this paper, we propose two new algorithmic components, which can be used to improve results. They are given as follows:

1. The mutation scheme of DE/best/1 has been retained but a provision has been made to change the mutation schemes for three different sections of the population.
2. A sincere effort has been made to remove the problem of stagnation by introducing the control parameter *stagnate* when there is no improvement of

B.K. Panigrahi et al. (Eds.): SEMCCO 2013, Part I, LNCS 8297, pp. 286–297, 2013.

global best fitness value after mutation, crossover and selection for certain successive generations. If this happens, we annihilate and regenerate the stagnation-causing portion of the population.

The proposed DE variant is compared with DE/rand/1/bin, DE/current-to-best/1/bin, SaDE, JADE, jDE and DEGL over 25 standard numerical benchmark functions taken from the CEC 2005 competition and special session on real-parameter optimization.

The paper is organized as follows: Section II gives a general overview of the DE family of algorithms. Section III explains the essential features of the proposed DE variant. The experimental settings for the benchmarks and simulation strategies are presented in section IV along with results that outline the performance of the algorithm. Finally section V, presents a short discussion about the probable applications of the algorithm in prospective areas.

2 Differential Evolution : A General Discussion

Differential Evolution is one method which is a member of a class of methods called metaheuristics. It is an iterative scheme developed by Storn and Price in 1997 [8] which seeks to minimize an objective function by iteratively seeking a parameter vector $\boldsymbol{X}^*$ which minimizes the objective function $f(\boldsymbol{X}^*)$ ($f : \Omega \subseteq \mathbb{R}^D \longrightarrow \mathbb{R}$), that is $f(\boldsymbol{X}^*) < f(\boldsymbol{X})$ where $\boldsymbol{X} = [x_1, x_2, x_3, \ldots, x_D]^T$, the parameter vector which characterizes the performance of a system, for all $\boldsymbol{X} \in \Omega$, where Ω is a non-empty, large, finite set serving the purpose for the domain of the search.

2.1 Initialization of the Vectors Controlling the Performance of the System

Differential Evolution searches for a global optimum in a search space $\mathbb{R}^D$ comprising of D dimensions. The first step is to initialize a population of NP, D dimensional, real-valued parameter vectors, where NP is the population size for the optimization problem at hand. We call each such vector a genome or chromosome. In analogy with biological processes, the genomes will modify their values over generations which may be denoted by $G = 0, 1, 2, \ldots, G_{max}$. The ith vector of the population for the current generation may be represented by

$$\boldsymbol{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}].$$

The initialization of the population at $G = 0$ must be done taking care of the bounds which restrict the parameter vector $\boldsymbol{X}$. As such, the minimum and maximum values for $\boldsymbol{X}$ are given by

$$\boldsymbol{X}_{min} = [x_{1,min}, x_{2,min}, x_{3,min}, \ldots, x_{D,min}],$$

$$\boldsymbol{X}_{max} = [x_{1,max}, x_{2,max}, x_{3,max}, \ldots, x_{D,max}].$$

The initialization of the jth component of the ith vector is done as follows

$$x_{j,i,0} = x_{j,min} + rand_{i,j}[0,1]\ (x_{j,max} - x_{j,min}),$$

where $rand_{i,j}[0,1]$ is a uniformly distributed random number lying between 0 and 1 ($0 \leq rand_{i,j}[0,1] \leq 1$) and is instantiated independently for each component of the ith vector. This is done so as to ensure that the population initialized at $G = 0$ covers, as far as possible, the range of values that can possibly be taken by $\boldsymbol{X}$.

2.2 Mutation with Difference Vectors

Once the initialization has been done, the objective of the algorithm is to create a donor vector $\boldsymbol{V}_{i,G}$ corresponding to each member of the population (rechristened as the target vector, $\boldsymbol{X}_{i,G}$) in the current generation through a process called mutation. The five most commonly used mutation schemes are as follows:

1. DE/rand/1

   $$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{r_1^i,G} + F(\boldsymbol{X}_{r_2^i,G} - \boldsymbol{X}_{r_3^i,G})$$

2. DE/best/1

   $$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{best,G} + F(\boldsymbol{X}_{r_1^i,G} - \boldsymbol{X}_{r_2^i,G})$$

3. DE/current-to-best/1

   $$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{i,G} + F(\boldsymbol{X}_{best,G} - \boldsymbol{X}_{i,G}) + F(\boldsymbol{X}_{r_1^i,G} - \boldsymbol{X}_{r_2^i,G})$$

4. DE/best/2

   $$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{i,G} + F(\boldsymbol{X}_{r_1^i,G} - \boldsymbol{X}_{r_2^i,G}) + F(\boldsymbol{X}_{r_3^i,G} - \boldsymbol{X}_{r_4^i,G})$$

5. DE/rand/2

   $$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{r_1^i,G} + F(\boldsymbol{X}_{r_2^i,G} - \boldsymbol{X}_{r_3^i,G}) + F(\boldsymbol{X}_{r_4^i,G} - \boldsymbol{X}_{r_5^i,G})$$

The indices r_1^i, r_2^i, r_3^i, r_4^i and r_5^i are mutually exclusive random integers chosen from the closed interval $[1, NP]$ and all of them should be different from i. These indices are randomly generated anew for each donor vector. The scale factor F is a positive control parameter for scaling the difference vectors. $\boldsymbol{X}_{best,G}$ is the best individual vector with the best fitness (i.e. corresponding to a particular minimization problem, $\boldsymbol{X}_{best,G}$ has the lowest objective function value).

2.3 The Crossover Scheme

Crossover involves the mixing of components between the donor vector and the target vector $\boldsymbol{X}_{i,G}$ to form the trial vector $\boldsymbol{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, \ldots, u_{D,i,G}]$. The DE family of algorithms essentially uses two kinds of crossover: exponential (or two-point-modulo) and binomial (or uniform). We discuss the latter as the proposed DE variant uses it. The binomial crossover scheme may defined as follows:

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, \; rand_{i,j}[0,1] \leq Cr, or j = j_{rand}, \\ x_{j,i,G}, \; otherwise. \end{cases}$$

where, as explained in the previous section $rand_{i,j}[0,1]$ is a uniformly distributed random number created afresh for each jth component of the ith parameter vector. j_{rand} is a randomly chosen integer lying in the interval $[1, D]$ which ensures that $\boldsymbol{U}_{i,G}$ gets at least one component from $\boldsymbol{V}_{i,G}$. It is instantiated once for each vector for each generation. See [4] for a DE algorithm containing novel mutation and crossover strategies.

2.4 The Selection Scheme

Selection is the process that ascertains whether the target or the trial vector survives to the next generation which is denoted by $G = G + 1$. The selection operation may be described as follows:

$$\boldsymbol{X}_{i,G+1} = \begin{cases} \boldsymbol{U}_{i,G}, \; f(\boldsymbol{U}_{i,G}) \leq f(\boldsymbol{X}_{i,G}), \\ \boldsymbol{X}_{i,G}, \; f(\boldsymbol{U}_{i,G}) > f(\boldsymbol{X}_{i,G}). \end{cases}$$

where $f(\boldsymbol{X})$ is the given objective function to be minimized.

3 Algorithm of the Proposed Differential Evolution Variant

In this section, we slowly begin to develop the main aspects of the proposed DE variant. We start with defining the main parameters used. The scale factor, F has been given a value of 0.8, and the crossover rate, Cr has been given the value of 0.9, the latter value having been found suitable for a wide variety of optimization problems that can be successfully tackled by Differential Evolution. Parameter-selection in DE is addressed in [5]. The algorithm has been tested on a population NP of 100 individuals with the number of dimensions D being variable and set at 30, 50 and 100 respectively.

3.1 Development of a Novel Mutation Scheme that Closely Mimics Behavior Seen in the Natural World

Mutation Type I : Mutation between the Best and the Worst. The proposed DE variant uses a mutation scheme that is a slight departure from

the mutation scheme defined earlier by DE/best/1. The scheme in question is defined by

$$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{best,G} + F(\boldsymbol{X}_{r_1^i,G} - \boldsymbol{X}_{r_2^i,G})$$

where r_1^i and r_2^i are mutually exclusive random integers chosen in such a way that the maximum improvement in the present generation is given utmost importance. This is done by ensuring that r_1^i is chosen from the best $p\%$ and r_2^i from the worst $p\%$ of the population. As explained earlier, the population consists of 100 individuals, and here we have chosen $p = 20$. The novelty of this approach is clearly seen if we consider the equation defined above. A close look at the term in parenthesis suggests the formation of the vector $(\boldsymbol{X}_{r_1^i,G} - \boldsymbol{X}_{r_2^i,G})$, which indicates that this vector is nothing but one that provides the direction that is approximately the direction of maximum improvement, since r_1^i and r_2^i are chosen from the best and worst $p\%$ of the population respectively.

Mutation Type II : Mutation between Two Individuals of the Best p% of the Population Apart from the above, the proposed DE variant also considers the possibility of a mutation between two randomly selected members of the best $p\%$ of the population. The proposed advantage of this type of mutation is to overcome the overtly exploratory nature of the mutation type I and to make it a bit more exploitative. Since, in the present mutation type, the mutation is being performed taking into consideration $\boldsymbol{X}_{r_1^i,G}$ and $\boldsymbol{X}_{r_2^i,G}$, which are randomly selected vectors taken from the best $p\%$ of the population, it can be expected that the two aforementioned vectors do not differ by a great deal with respect to their fitness values. The mutation type II has the cumulative effect of directing the donor vector towards the historically best vector which has the best fitness value $\boldsymbol{X}_{best,G}$. Hence, we find that this type of mutation maintains the best members of the present generation and progressively directs them more and more towards $\boldsymbol{X}_{best,G}$. Therefore, this mutation type localizes the search region of the proposed DE variant around the region of the vector $\boldsymbol{X}_{best,G}$.

Mutation Type III : Average Case Mutation The average case mutation type is included to maintain diversity and also the general degree of randomness that characterizes a metaheuristic. This third mutation type seeks to mutate any two randomly selected vectors of the population excluding the best $p\%$ of the population, since this case has already been considered in the first mutation type. At this juncture, it becomes an immediate necessity to differentiate between the mutation type III and the earlier described mutation types I and II.

As compared to the exploratory nature of the mutation type I and the exploitative nature of the mutation type II, the mutation type III is neither exploratory nor exploitative. Mutating between two random individuals of the best $p\%$ of the population ensures that the search is confined to contours around $\boldsymbol{X}_{best,G}$, and mutating between two random individuals of the best and worst $p\%$ of the population ensures that the search space is generously explored.

The overall mutation scheme, which is a combination of the types, I, II and III is carried out in the following manner: For a number of individuals equal to

$2p\%$ of the population we carry out the mutation type I, that is, a mutation between the best and worst $p\%$ of the population. For a number equal to $p\%$ of the population, the mutation type II is chosen, namely a mutation between any two members of the best $p\%$ of the population. For the remaining percentage of the population, we carry out the mutation type III, the average case mutation type elucidated earlier.

The mutation scheme is successfully executed by taking into consideration the term *Sorted_Population* which contains the individuals of the present generation but in a sorted order. The sorting is done in such a manner that the individual with the best fitness value is placed first and the individual with the worst fitness value is placed last. In other words, *Sorted_Population* contains all the individuals of the present generation, sorted according to their fitness values, the fittest being given priority.

3.2 Crossover

The proposed DE variant uses a novel crossover scheme. This elite crossover operation incorporates in its working a greedy parent-selection strategy. For each donor vector $\boldsymbol{V}_{i,G}$, a vector is randomly selected from the best $p\%$ vectors of the present population and then binomial crossover is performed between the donor vector and the randomly chosen vector in order to generate the trial vector $\boldsymbol{U}_{i,G}$.

3.3 Stagnation, Its Connotations and an Attempted Removal

Stagnation refers to the trapping of the population near local extrema that causes major problems by preventing the population from progressing towards the much coveted global extrema.

The stagnation in the proposed DE variant is detected by the control parameter *stagnate*. When the fitness of the population does not improve even after mutation and crossover for several generations, there arises the need for annihilation and regeneration of the so-called "bad" part of the population. The rule that has been followed for annihilation and regeneration is as follows: Here the worst $\frac{p}{2}\%$ of the population is annihilated and the same number of chromosomes is regenerated using a regeneration rule. The parameter *stagnate* is chosen such that:

1. The stagnation is detected properly.
2. The population with regenerated chromosomes gets sufficient opportunity to overcome the stagnation by improving their fitness values.

The annihilation and regeneration rule can be presented as follows: To generate a new population, each chromosome is generated according to the following rule: For every dimension j, where $j = 1, 2, 3, \ldots, n$, a random number rnd_j uniformly distributed within $(0, 1)$ is generated and compared with a parameter $pro \in (0, 1)$. If $rnd_j \leq pro$, then $New\_Chromosome^j$ is set to a randomly generated real number uniformly distributed in the legal range $[Lj, Uj]$, where Lj and

Uj are the lower and upper bounds respectively of the dimension j. Otherwise, $New\_Chromosome^j$ is inherited from any member selected randomly from the best $\frac{p}{2}\%$ of the population, that is to say,

$$New\_Chromosome^j = \begin{cases} rand(Lj, Uj), & rnd_j \leq pro, \\ Sorted\_Population^j_{ran}, & otherwise. \end{cases}$$

where *Sorted_Population* is the sorted array containing the present population and *New_Chromosome* has each member of the newly generated sub-population (which is of the same size as the $\frac{p}{2}\%$ of the population to be annihilated). Here, *ran* is a random positive integer less than the ceiling of $\frac{p}{2}\%$ of the population. Now, after generation of the desired number of chromosomes, this sub-population is injected into the present population replacing the worst $\frac{p}{2}\%$ of its members. The parameter *pro* is kept equal to $\frac{D}{NP}$ where D is the dimensionality of the optimization problem at hand and NP is the number of individuals in the population. This newly generated sub-population possesses some desirable features of the present population as some of its components are chosen from the best vectors of the present population.

For unimodal functions, stagnation rarely occurs; so the stagnate parameter would be ideally quite high for such functions. However, for multimodal and hybrid functions with noise, stagnation occurs pretty frequently and choosing *stagnate* to be at a lesser value will lead to better results. Experiments with a wide range of problems suggest that $stagnate = \frac{D}{2}$ seems to be a suitable choice for black-box optimization problems.

4 Experiments and Results

4.1 Numerical Benchmarks

The proposed algorithm (abbreviated as CAR_DE from now on) is tested using a set of standard benchmark functions from the special session and competition on real parameter optimization held under the IEEE CEC 2005. These functions include a diverse set of features like multimodality, ruggedness, noise in fitness, ill-conditioning, rotation etc. and based on the classical benchmarks like Rosenbrock's, Rastrigin's, Griewank's, Schwefel's and Ackley's functions. A detailed description of these functions appears in [11] and is not repeated here. In summary, functions 1 to 5 are unimodal, functions 6 to 14 are multimodal and functions 15 to 25 are hybrid functions.

4.2 Algorithms Compared and Parametric Setup

The performance of CAR_DE is compared with the following algorithms that include two classical DE variants and four state-of-the-art adaptive DE variants:

1. DE/current-to-best/1/bin with $F = 0.8$ and $Cr = 0.9$;
2. DE/rand/1/bin with $F = 0.8$ and $Cr = 0.9$;

Table 1. Mean and Standard Deviation of the Error Values for F_1 to $F_{10}(50D)$. The Best Entries are Marked in Boldface.

Functions →	f_1	f_2	f_3	f_4	f_5
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$9.6015e-05$	$3.960e+13$	$5.469e+7$	$1.180e+4$	$8.709e+03$
	$(1.2837e-05)$	$(9.307e+02)$	$(1.328e+07)$	$(3.332e+03)$	$(6.938e+02)$
DE/current-to-best/bin	$2.1443e-06$	$2.136e+03$	$1.030e+07$	$8.5675e+03$	$7.462e+03$
	$(6.1254e-06)$	$(1.128e+03)$	$(6.375e+06)$	$(2.8624e+03)$	$(1.326e+03)$
JADE	$7.4615e-14$	$5.6310e-04$	**8.7156e+04**	$3.160e+03$	$3.055e+03$
	$(2.4190e-04)$	$(7.8233e-06)$	**(3.6847e+04)**	$(4.134e-01)$	$(5.485e+02)$
jDE	$3.1544e-09$	$5.202e+03$	$2.977e+07$	$1.0194e+04$	$4.206e+03$
	$(4.9946e-09)$	$(1.486e+03)$	$(5.744e+06)$	$(2.1828e-01)$	$(5.088e+02)$
SaDE	$1.4872e-11$	$2.280e-03$	$7.179e+05$	$9.778e+04$	$5.992e+03$
	$(2.8335e-12)$	$(8.545e-03)$	$(1.007e+06)$	$(9.835e+01)$	$(4.464e+02)$
DEGL	$2.3462e-20$	$1.1757e-07$	$2.3114e+05$	$1.5746e+03$	$5.0692e+02$
	$(5.6234e-20)$	$(6.5592e-08)$	$(1.032e+05)$	$(9.501e+00)$	$(5.803e+02)$
CAR_DE	**5.8927e-36**	**5.1189e-13**	$8.5215e+04$	**1.5795e-02**	**4.0927e+02**
	(0.0000e+00)	**(1.4956e-14)**	$(1.6874e+04)$	**(2.2227e-02)**	**(1.6839e+02)**

Functions →	f_6	f_7	f_8	f_9	f_{10}
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$4.9162e+01$	$6.1953e+03$	$2.1142e+01$	$3.468e+02$	$3.763e+02$
	$(1.182e+01)$	$(4.594e-12)$	$(3.330e-02)$	$(1.199e+01)$	$(1.578e+01)$
DE/current-to-best/bin	$1.0782e+07$	$6.6691e+03$	$2.1133e+01$	$2.406e+02$	$2.5467e+02$
	$(1.237e+07)$	$(1.795e+02)$	$(3.251e-02)$	$(2.939e+01)$	$(7.6634e+01)$
JADE	$1.5413e+01$	$6.1932e+03$	$2.1136e+01$	**1.352e+02**	$1.935e+02$
	$(1.0642e+01)$	$(1.840e+00)$	$(3.251e-02)$	**(2.591e+00)**	$(2.060e+01)$
jDE	$4.1758e+01$	$6.3114e+03$	$2.1132e+01$	$1.716e+02$	$1.9597e+02$
	$(8.910e+00)$	$(1.596e+01)$	$(3.807e-02)$	$(1.409e+01)$	$(5.6236e+01)$
SaDE	$1.1337e+01$	$6.1951e+03$	$2.1132e+01$	$1.148e+02$	**6.342e+01**
	$(1.044e+01)$	$(4.594e-12)$	$(3.458e-02)$	$(1.266e+01)$	**(1.287e+01)**
DEGL	$1.3452e+01$	$6.1953e+03$	$2.1131e+01$	$1.620e+02$	$1.0217e+02$
	$(1.108e+01)$	$(4.594e-12)$	$(3.917e-02)$	$(1.743e+01)$	$(3.5590e+01)$
CAR_DE	**1.0653e+01**	**1.1084e-12**	**2.1131e+01**	$1.5422e+02$	$1.7213e+02$
	(3.9319e+00)	**(4.0164e-14)**	**(2.0287e-02)**	$(2.1849e+01)$	$(4.3521e+01)$

3. JADE with $c = 0.1$, $p = 0.05$, and optional external archive [12];
4. jDE with $F_l = 0.1$, $F_u = 0.9$, and $\tau_1 = \tau_2 = 0.1$ [1];
5. SaDE [7];
6. DEGL/SAW [3] with $\alpha = \beta = F = 0.8$, $Cr = 0.9$, and neighborhood size=$0.1 * NP$.

The population size NP of all the DE variants is kept at 100 irrespective of the dimension D.

4.3 Simulation Strategies

Functions f_1 to f_{25} are tested for 50 and 100 dimensions. The maximum number of FEs are set to 500000 for 50D problems and 1000000 for 100D problems as per the guidelines of the CEC 2005 special session [11]. All the simulations have been done on an Intel Core-i5 CPU machine with 4 GB memory and 2.5 GHz speed.

4.4 Results on the Numerical Benchmarks

Tables 1, 2, 3 and 4 show the mean and standard deviation of the 50 best-of-the-run-errors for 50 independent runs of each of the seven algorithms on

Table 2. Mean and Standard Deviation of the Error Values for F_{11} to $F_{25}(50D)$. The Best Entries are Marked in Boldface.

Functions →	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$7.264e+01$	$2.049e+06$	$3.596e+01$	$2.339e+01$	$5.090e+02$
	$(1.212e+00)$	$(5.925e+05)$	$(1.446e+00)$	$(1.486e-01)$	$(7.981e+01)$
DE/current-to-best/bin	$4.950e+01$	$2.505e+05$	$3.412e+01$	$2.286e+01$	$4.989e+02$
	$(4.4551e+00)$	$(1.137e+05)$	$(8.9042e+00)$	$(4.091e-01)$	$(5.001e+01)$
JADE	$6.208e+01$	$1.768e+05$	$2.3112e+01$	$2.284e+01$	$3.769e+02$
	$(1.744e+00)$	$(7.105e+04)$	$(4.784e-01)$	$(2.5486e-01)$	$(8.764e+01)$
jDE	$7.330e+01$	$1.473e+05$	$2.5603e+01$	$2.309e+01$	$4.000e+02$
	$(1.008e+00)$	$(1.928e+05)$	$(1.322e+00)$	$(2.8437e-01)$	$(0.000e+00)$
SaDE	$6.634e+01$	**8.871e+03**	$2.771e+01$	$2.284e+01$	$3.8827e+01$
	$(1.485e+00)$	**(7.092e+03)**	$(4.112e+00)$	$(2.0634e-01)$	$(1.0755e+02)$
DEGL	$6.290e+01$	$5.781e+04$	$3.063e+01$	$2.262e+01$	$3.8982e+02$
	$(1.1360e+01)$	$(4.566e+04)$	$(4.361e+00)$	$(3.3750e-01)$	$(4.9284e+01)$
CAR_DE	**4.1575e+01**	$1.4989e+06$	**1.4648e+01**	**2.1956e+01**	**3.6446e+02**
	(1.4847e+00)	$(2.8857e+05)$	**(4.0440e+00)**	**(1.0006e+00)**	**(1.2395e+01)**

Functions →	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$2.7343e+02$	$3.7286e+02$	$9.9043e+02$	$9.4100e+02$	$9.8536e+02$
	$(1.0498e+01)$	$(3.1287e+01)$	$(4.8709e+01)$	$(3.8003e+01)$	$(4.4956e+01)$
DE/current-to-best/bin	$2.5387e+02$	$2.5234e+02$	$9.2089e+02$	$9.2667e+02$	$9.3586e+02$
	$(1.4757e+01)$	$(5.7296e+01)$	$(5.6632e+01)$	$(5.9865e+01)$	$(5.3925e+01)$
JADE	$1.437e+02$	$1.896e+02$	$9.206e+02$	$9.6031e+02$	$9.8672e+02$
	$(5.2267e+01)$	$(3.8745e+01)$	$(1.893e+00)$	$(2.5236e+01)$	$(1.8675e+02)$
jDE	$2.716e+02$	$3.059e+02$	$9.145e+02$	$9.2090e+02$	$9.9121e+02$
	$(4.7190e+00)$	$(1.163e+01)$	$(3.163e+01)$	$(1.0406e+01)$	$(1.5365e+01)$
SaDE	$1.5420e+01$	$1.934e+02$	$9.041e+02$	$9.3493e+02$	$9.3167e+02$
	$(6.1686e+01)$	$(2.9679e+00)$	$(5.208e+01)$	$(1.9639e+01)$	$(2.0137e+01)$
DEGL	$1.3153e+02$	$1.7659e+02$	$9.6067e+02$	$9.1430e+02$	$9.2196e+02$
	$(1.9986e+01)$	$(2.3653e+01)$	$(2.8458e+01)$	$(2.0105e+01)$	$(4.5874e+01)$
CAR_DE	**1.2552e+02**	**1.2393e+02**	**8.4017e+02**	**8.4094e+02**	**8.3885e+02**
	(1.2970e+00)	**(1.3762e+00)**	**(1.0514e+00)**	**(3.1157e+00)**	**(1.8098e+00)**

Functions →	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$9.1108e+02$	$9.9463e+02$	$9.1185e+02$	$7.9849e+02$	$1.7586e+03$
	$(5.7474e+02)$	$(1.3465e+01)$	$(2.5986e+01)$	$(2.4586e+01)$	$(4.5365e+00)$
DE/current-to-best/bin	$8.9465e+02$	$9.3443e+02$	$9.2645e+02$	$7.9456e+02$	$1.7846e+03$
	$(1.7465e+02)$	$(1.1026e+01)$	$(2.5986e+02)$	$(1.3642e+01)$	$(6.7654e+00)$
JADE	$8.523e+02$	$9.1370e+02$	$8.103e+02$	$2.000e+02$	$1.6632e+03$
	$(3.5175e+02)$	$(2.4356e+01)$	$(2.4572e+02)$	(0)	$(5.5842e+00)$
jDE	$8.0619e+02$	$9.796e+02$	$8.3044e+02$	$2.000e+02$	$1.728e+03$
	$(1.0896e+02)$	$(1.4851e+01)$	$(1.0787e+02)$	(0)	$(6.2562e+00)$
SaDE	$8.6400e+02$	$9.7245e+02$	$8.6405e+02$	$2.000e+02$	$1.7586e+03$
	$(1.5799e+02)$	$(3.3383e+01)$	$(1.5266e+02)$	(0)	$(3.1453e+00)$
DEGL	$8.3600e+02$	$9.4242e+02$	$8.3934e+02$	$7.2465e+02$	$1.571e+03$
	$(2.1772e+02)$	$(3.5647e+01)$	$(1.6620e+02)$	$(8.3066e+01)$	$(6.5096e+00)$
CAR_DE	**7.3459e+02**	**5.001e+02**	**7.0374e+02**	**2.000e+02**	**2.3126e+02**
	(8.9992e+00)	**(8.2260e+00)**	**(7.8156e+01)**	**(0)**	**(6.4983e+00)**

25 numerical benchmarks for 50D and on the first 14 benchmarks for 100D respectively. Note that the best-of-the-run error corresponds to the absolute difference between the best-of-the-run value $f(\boldsymbol{X}_{best})$ and the actual optimum f^* of a particular objective function, i.e., $|f(\boldsymbol{X}_{best}) - f^*|$. Table 1 and 2 reveal that CAR_DE outperformed the other DE variants in 21 out of 25 functions for 50D problems and performs well specially for multimodal and hybrid functions. Tables 3 and 4 show that this algorithm defeats the other DE variants in 11 out of 14 functions for 100D problems. Thus the increasing dimensionality does not worsen the performance of this algorithm. Note that CAR_DE retains its

Table 3. Mean and Standard Deviation of the Error Values For F_1 To $F_{10}(100D)$. The Best Entries are Marked in Boldface.

Functions →	f_1	f_2	f_3	f_4	f_5
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$2.9467e-05$	$8.9371e+04$	$9.7635e+07$	$3.0937e+05$	$1.0675e+06$
	$(3.0947e-04)$	$(7.4925e+04)$	$(8.4625e+03)$	$(4.9875e+03)$	$(4.8437e+03)$
DE/current-to-best/bin	$8.4735e-06$	$9.0927e+03$	$9.8525e+06$	$7.5821e+04$	$5.0967e+05$
	$(9.4927e-05)$	$(8.4736e-05)$	$(5.0948e+03)$	$(8.6745e+03)$	$(5.8453e+03)$
JADE	$6.4825e-10$	$3.4923e+03$	**2.9371e+06**	**5.0342e+04**	$7.5251e+05$
	$(4.0249e-09)$	$(8.4725e-06)$	**(7.4728e+04)**	**(6.9785e-03)**	$(3.9464e+03)$
jDE	$7.4627e-07$	$5.9371e+03$	$8.9372e+06$	$7.9261e+04$	$2.9361e+05$
	$(3.9371e-09)$	$(8.4625e-07)$	$(3.0927e+04)$	$(2.7456e+03)$	$(6.4536e+03)$
SaDE	$8.2615e-08$	$2.9471e+04$	$7.9171e+06$	$6.0283e+04$	$9.7364e+05$
	$(5.0445e-09)$	$(7.7352e-03)$	$(8.0936e+02)$	$(6.9573e+03)$	$(2.0936e+03)$
DEGL	$9.6844e-07$	$8.9261e+04$	$4.8326e+07$	$1.9372e+05$	$9.0936e+05$
	$(4.0937e-08)$	$(8.4563e-06)$	$(5.0945e+04)$	$(4.9463e+04)$	$(6.9382e+03)$
CAR_DE	**9.099e-13**	**1.5207e+01**	$4.5832e+06$	$9.7403e+04$	**1.7284e+04**
	(7.1901e-14)	**(2.8769e+01)**	$(2.9612e+04)$	$(2.6157e+04)$	**(2.7346e+03)**

Functions →	f_6	f_7	f_8	f_9	f_{10}
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$1.8946e+05$	$1.8574e+05$	$2.2497e+01$	$9.3752e+02$	$9.7631e+02$
	$(4.9372e+01)$	$(4.0832e+02)$	$(3.0423e+01)$	$(6.7322e+01)$	$(1.5787e+01)$
DE/current-to-best/bin	$9.3967e+04$	$1.6738e+05$	$2.2308e+01$	$9.4065e+02$	$7.5467e+02$
	$(7.4925e+01)$	$(8.4735e+02)$	$(8.4725e+00)$	$(7.6695e+01)$	$(5.64334e+01)$
JADE	$7.5329e+04$	$9.0417e+04$	$2.1964e+01$	$8.9272e+02$	$5.935e+02$
	$(7.3725e+01)$	$(8.4735e+02)$	$(9.4673e+01)$	$(4.5043e+01)$	$(8.8760e+01)$
jDE	$8.1876e+04$	$1.1625e+05$	$2.2197e+01$	$9.3264e+02$	$6.9597e+02$
	$(8.4752e+00)$	$(7.4637e+02)$	$(7.4627e+00)$	$(7.3645e+01)$	$(7.7653e+01)$
SaDE	$2.0172e+04$	$9.8463e+04$	$2.2075e+01$	$9.2737e+02$	$8.3426e+02$
	$(8.2514e+01)$	$(8.4623e+02)$	$(9.3736e+00)$	$(5.1369e+01)$	$(1.6378e+01)$
DEGL	$4.2684e+04$	$1.2383e+05$	$2.2210e+01$	$9.5591e+03$	$8.2017e+02$
	$(5.8352e+01)$	$(9.4573e+02)$	$(4.8252e+00)$	$(3.9274e+01)$	$(3.5590e+01)$
CAR_DE	**1.8396e+02**	**1.8914e+04**	**2.1674e+01**	**5.9232e+02**	**4.9012e+02**
	(9.8705e+01)	**(1.5657e+03)**	**(3.0845e+01)**	**(2.2333e+01)**	**(8.4560e+01)**

Table 4. Mean and Standard Deviation of the Error Values for F_{11} to $F_{14}(100D)$. The Best Entries are Marked in Boldface.

Functions →	f_{11}	f_{12}	f_{13}	f_{14}
Algo ↓	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	$9.2648e+01$	$9.049e+06$	$5.2926e+01$	$4.4339e+01$
	$(1.9212e+00)$	$(7.925e+05)$	$(4.346e+00)$	$(1.4836e-01)$
DE/current-to-best/bin	$6.9150e+01$	$6.505e+05$	$4.4132e+01$	$4.5186e+01$
	$(6.6451e+00)$	$(4.137e+05)$	$(3.5042e+00)$	$(4.4091e-01)$
JADE	$9.1208e+01$	$6.768e+05$	$3.4141e+01$	$4.2684e+01$
	$(4.7744e+00)$	$(2.105e+04)$	$(4.0834e+00)$	$(3.8163e-01)$
jDE	$7.7630e+01$	$7.473e+05$	$3.8650e+01$	$4.3309e+01$
	$(9.4008e+00)$	$(7.928e+05)$	$(4.322e+01)$	$(2.7163e-01)$
SaDE	$8.4684e+01$	**9.0781e+04**	$3.5713e+00$	$4.2874e+01$
	$(3.485e+00)$	**(5.9092e+03)**	$(5.112e+00)$	$(2.5343e-01)$
DEGL	$8.1290e+01$	$8.8781e+05$	$4.063e+01$	$4.2662e+01$
	$(1.360e+01)$	$(7.1566e+04)$	$(2.361e+01)$	$(6.9834e-01)$
CAR_DE	**6.2910e+01**	$4.4677e+05$	**3.3179e+01**	**4.0225e+01**
	(4.1313e+00)	$(1.8670e+04)$	**(4.9355e+00)**	**(2.4122e-01)**

superiority in case of rotated, shifted functions and functions with noise in fitness. Therefore, function rotation and incorporation of multiplicative noise does not hamper the performance of the algorithm significantly.

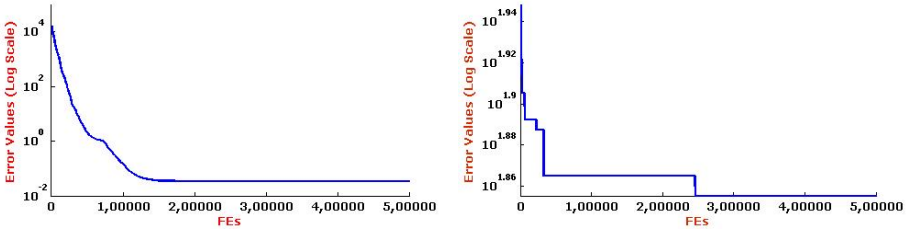

Fig. 1. Progress toward the optimum solution for median run of eight algorithms over two numerical benchmarks (in 50D).
Left: Shifted rotated Griewank's function f_7; Right: Shifted Rotated Weierstrass function f_{11}.

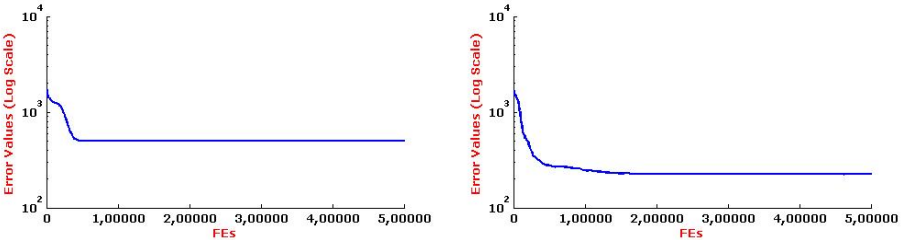

Fig. 2. Progress toward the optimum solution for median run of eight algorithms over two numerical benchmarks (in 50D).
Left: Rotated Hybrid Composition Function f_{21}; Right: Composition function f_{25}.

For further illustration, in Fig. 1 and 2, we show the convergence graphs for the median run of (where the runs were sorted according to the final best error values achieved in each) the CAR_DE algorithm on four benchmarks in 50D.

5 Conclusion

The results for 50 dimensions reveal the performance of CAR_DE to be almost comparable to that of JADE (and SADE in some instances) for all the unimodal functions, however the performance of CAR_DE is slightly better for multimodal functions, and significantly better for hybrid functions. In all the hybrid functions, CAR_DE performs better than all the state-of-the-art DE variants. The results obtained for higher dimensions support the superiority of CAR_DE over the other DE variants. Specially, the hybrid functions of 50 dimensions and multimodal functions of 100 dimensions lucidly demonstrate how the novel mutation scheme coupled with annihilation and regeneration can improve the efficiency of DE.

The real challenge to an evolutionary algorithm comes in the form of optimization of highly complex, noisy and higher dimensional functions (such as those working in 50 or 100 dimensions). As the complexity of the function increases, there arrives the need to efficiently explore the search space.

The present algorithm tries to optimize such more complex functions, attaining that purpose without losing much of its explorative nature for functions

of smaller dimensions. Further improvements to the CAR_DE algorithm may be made by tuning the algorithmic parameters and maintaining an adaptive external archive.

References

1. 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. 10(6), 646–657 (2006)
2. Das, S., Suganthan, P.N.: Differential Evolution A survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15(1), 4–31 (2011)
3. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a neighbourhood based mutation operator. IEEE Trans. Evol. Comput. 13(3), 526–553 (2009)
4. Islam, S.M., Das, S., Ghosh, S., Roy, S., Suganthan, P.N.: An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 42(2), 482–500 (2012)
5. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing 11(2), 1679–1696 (2011)
6. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution A practical Approach to Global Optimization. Springer, Berlin (2005)
7. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential Evolution Algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13(2), 398–417 (2009)
8. Storn, R., Price, K.V.: Differential Evolution A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11(4), 341–359 (1997)
9. Storn, R., Price, K.V.: Differential Evolution A simple and efficient adaptive scheme for global optimization over continuous spaces, ICSI, Berkeley, CA. Tech. Rep. TR-95-012
10. Storn, R., Price, K.V.: Minimizing the real functions of the ICEC 1996 contest by differential evolution. In: Proc. IEEE Int. Conf. Evol. Comput., Nagoya, Japan, pp. 842–844 (1996)
11. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technol. Univ., Singapore (2005)
12. Zhang, J., Sanderson, A.C.: JADE: Adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13(5), 945–958 (2009)