2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Differential Evolution with Local Neighborhood

Uday K. Chakraborty
Dept. of Math & Comp. Sc.
University of Missouri,
St. Louis, MO 63121, USA
uday@cs.umsl.edu

Swagatam Das
Dept. of Electronics & Telecomm. Eng.
Jadavpur University
Kolkata 700032, India

Amit Konar
Dept. of Electronics & Telecomm. Eng.
Jadavpur University
Kolkata 700032, India

Abstract- Differential evolution (DE) is well known as a simple and efficient scheme for global optimization over continuous spaces. It is, however, not free from the problem of slow and premature convergence. In this paper we present an improved variant of the classical DE2 scheme, by utilizing the concept of the local neighborhood of each vector. This scheme attempts to balance the exploration and exploitation abilities of DE without requiring additional function evaluations. The new scheme is shown to be statistically significantly better than three other popular DE variants on a six-function test-bed and also on two real-world optimization problems with respect to the following performance measures: solution quality, time to find the solution, frequency of finding the solution, and scalability.

I. Introduction

Differential evolution (DE) [1] seeks to replace the classical crossover and mutation schemes of the genetic algorithm (GA) by alternative differential operators. The DE algorithm has recently become quite popular in the machine intelligence and cybernetics community. In many cases, it has outperformed the GA or the particle swarm optimization (PSO) [2]. As in other evolutionary algorithms, two fundamental processes drive the evolution of a DE population: the *variation* process, which enables exploring the different regions of the search space, and the *selection* process, which ensures exploitation of the acquired knowledge about the fitness landscape.

DE does suffer from the problem of premature convergence to some suboptimal region of the search space. In addition, like other stochastic optimization techniques, the performance of classical DE deteriorates with the increase of dimensionality of the search space. A large body of work exists on the application of DE to practical optimization problems [e.g., 3-5]. However, little has been done to modify the basic mutation strategies of DE in order to avoid the problem of slow or false convergence [6, 7].

In the present paper, we propose a modified version of Storn and Price's "scheme DE2" (also known as DE/best/1). We define a neighborhood for each vector. Based on the neighborhood, two types of mutation are defined: local and global. Finally, the local and global schemes are combined, using time-varying weights. During the earlier stages of search the local model is emphasized in order to facilitate better exploration of the search space. During the later stages global mutation is emphasized. In addition, the use of random scale factors is seen to improve performance.

The remainder of this paper is organized as follows. In Section II we provide a brief outline of DE. Section III introduces the proposed modifications. Experimental settings for the benchmarks and simulation strategies are explained in Section IV. Results are presented in Section V, and conclusions drawn in Section VI.

II. The DE Family

DE starts with a population of N_p, D-dimensional parameter vectors. The i-th vector of the population at iteration (time) t is

$$\vec{X}_i(t) = [x_{i,1}(t), x_{i,2}(t), \ldots x_{i,D}(t)] \quad (1)$$

In each iteration of the algorithm, for each population member $\vec{X}_i(t)$, a 'donor' vector $\vec{V}_i(t+1)$ is created. It is the method of creating this donor vector that distiguishes various DE schemes from one another. In "scheme DE1" (DE/rand/1), to create $\vec{V}_i(t+1)$, three vectors (say r_1, r_2, and r_3) are randomly chosen from the current population. Next the difference of any two of these three vectors is scaled by a scalar F and the scaled difference is added to the third vector to produce the donor vector:

$$v_{i,j}(t+1) = x_{r1,j}(t) + F.(x_{r2,j}(t) - x_{r3,j}(t)) \quad (2)$$

The DE family of algorithms use two kinds of crossover, namely 'exponential' and 'binomial'. In 'exponential' crossover, we first choose an integer n randomly from [0, D-1]. We choose another integer L from the interval [1, D]. An offspring vector:

$$\vec{U}_i(t+1) = [u_{i,1}(t+1), u_{i,2}(t+1), u_{i,3}(t+1), \ldots, u_{i,D}(t+1)] \quad (3)$$

is then formed:

$$\left.\begin{array}{ll} u_{i,j}(t+1) = V_{i,j}(t+1) & \text{for } j = <n>_D, <n+1>_D, \ldots <n-L+1>_D \\ = X_{i,j}(t+1) & \text{for all other } j \in [0, D-1] \end{array}\right\} \quad (4)$$

where the angular brackets $< >_D$ denote a modulo function with modulus D. The integer L is drawn from [1, D] according to the following policy:

L = 0;
Do L=L+1;
while (rand (0, 1) < Cr) AND (L<D));

Thus in effect, Probability (L>m) = $(Cr)^{m-1}$ for any m>0. Cr, the crossover constant, is a control parameter of DE. For each donor vector $\vec{V}_i$, a new set of n and L must be chosen randomly as shown above.

'Binomial' crossover is implemented as follows:

$$u_{i,j}(t+1) = v_{i,j}(t+1) \quad \text{if rand}(0, 1) < Cr$$
$$= x_{i,j}(t) \quad \text{otherwise} \qquad (5)$$

Selection favors the better between the parent and the child:

$$\left.\begin{aligned} \vec{X}_i(t+1) &= \vec{U}_i(t+1) \text{ if } f(\vec{U}_i(t+1)) \le f(\vec{X}_i(t)) \\ &= \vec{X}_i(t) \quad \text{if } f(\vec{U}_i(t+1)) > f(\vec{X}_i(t)) \end{aligned}\right\} \qquad (6)$$

where f( ) is the objective function to be minimized.

DE/best/1 follows the same procedure except that the donor vector is created using two randomly selected members of the population as well as the best vector of the current time step:

$$\vec{V}_i(t+1) = \vec{X}_i(t) + \lambda.(\vec{X}_{best}(t) - \vec{X}_i(t)) + F.(\vec{X}_{r_2}(t) - \vec{X}_{r3}(t)) \qquad (7)$$

where λ, another control parameter of DE, is in [0, 2]. To reduce the number of control parameters a usual choice is to put λ = F.

Storn and Price [8] suggested a total of ten different working strategies for DE. In this paper we have used DE/rand/1/bin, DE/best/1/bin, and DE/best/2/bin for comparison with the proposed method.

III. The Modified DE Algorithm

Most of the population-based search algorithms try to balance two contradictory aspects: exploration and exploitation. The former means the ability of the algorithm to explore or search different regions of the search-space while the latter denotes the tendency to stay near the already-found good solutions. Several versions of the classical DE (e.g., DE/best/1, DE/best/2, etc.) use the best vector of the population to mutate other vectors. By 'best' we mean the vector with the best fitness value in the entire population at a certain iteration. These schemes promote exploitation since all the population vectors at any iteration are attracted towards the same best position (pointed by the 'best' vector) on the fitness landscape. In such cases, the population may lose its global exploration abilities within a relatively small number of iterations, getting trapped in some local optimal point. Again, DE employs a greedy selection strategy (the better between the parent and the child is selected) and uses a fixed scale factor F (typically 0.8 or 0.9). Thus if the difference vector $\vec{X}_{r_2} - \vec{X}_{r_3}$, used for perturbation in (7) is small (this is usually the case when the vectors come very close to each other and the population converges to a small domain), the vectors may not be able to explore better regions of the search space, thereby finding it difficult to escape large plateaus or suboptimal peaks/valleys.

To (partially) overcome these difficulties with the classical DE/best strategies, we propose a neighborhood-based local mutation operator that draws inspiration from particle swarm optimization [9, 10]. Suppose we have a DE population $P = [\vec{X}_1, \vec{X}_2, ..., \vec{X}_{N_p}]$ where each $\vec{X}_i$ (i = 1, 2, ..., N_p) is a D-dimensional vector. Now for every vector $\vec{X}_i$ we define a neighborhood of radius k, consisting of vectors $\vec{X}_{i-k}, ...\vec{X}_i, ...\vec{X}_{i+k}$. We assume the vectors to be organized in a circular fashion such that the two immediate neighbors of vector $\vec{X}_1$ are $\vec{X}_{N_p}$ and $\vec{X}_2$. For each member of the population a local mutation is created by employing the fittest vector in the neighborhood of that member and two other vectors chosen from the same neighborhood. The model may be expressed as:

$$\vec{L}_i(t) = \vec{X}_i(t) + \lambda'.(\vec{X}_{nbest}(t) - \vec{X}_i(t)) + F'.(\vec{X}_p(t) - \vec{X}_q(t)) \qquad (8)$$

where the subscript *nbest* indicates the best vector in the neighborhood of $\vec{X}_i$ and $p, q \in (i-k, i+k)$. Apart from this, we also use a global mutation expressed as:

$$\vec{G}_i(t) = \vec{X}_i(t) + \lambda.(\vec{X}_{best}(t) - \vec{X}_i(t)) + F.(\vec{X}_r(t) - \vec{X}_s(t)) \qquad (9)$$

where the subscript *best* indicates the best vector in the entire population, and $r, s \in (1, N_p)$. Note that equation (9) is the same as equation (7). Global mutation encourages exploitation, since all members (vectors) of a population are biased by the same individual (the population best); local mutation, in contrast, favors exploration, since in general different members of the population are likely to be biased by different individuals. Now we combine these two models using a time-varying scalar weight $w \in (0,1)$ to form the actual mutation of the new DE as a weighted mean of the local and the global components:

$$\vec{V}_i(t) = w.\vec{G}_i(t) + (1-w).\vec{L}_i(t) \qquad (10)$$

The weight factor varies linearly with time as folows:

$$w = w_{\min} + (w_{\max} - w_{m\min}).(\frac{iter}{MAXIT}) \qquad (11)$$

where *iter* is the current iteration number, MAXIT is the maximum number of iterations allowed and w_{max}, w_{min} denote, respectively, the maximum and minimum value of the weight, with $w_{\max}, w_{\min} \in (0,1)$. Thus the algorithm starts at iter = 0 with $w = w_{min}$ but as *iter* increases towards MAXIT, w increases gradually and ultimately when iter = MAXIT w reaches w_{max}. Therefore at the beginning, emphasis is laid on the local mutation scheme, but with time, contribution from the global model increases. In the local model attraction towards a single point of the search space is reduced, helping DE avoid local optima. This feature is essential at the beginning of the search process when the candidate vectors are expected to explore the search space vigorously. Clearly, a judicious choice of w_{max} and w_{min} is necessary to strike a balance between the exploration and exploitation abilities of the algorithm. After some

experimenting, we found that $w_{max} = 0.8$ and $w_{min} = 0.4$ seem to improve the performance of the algorithm over a number of benchmark functions.

Although the classical versions of DE use fixed values for scaling factors, we choose the four scaling factors $\lambda, \beta, \alpha', F'$ (used in relations (8) and (9)) as uniformly distributed random numbers between 0.5 and 1.5 (with a mean value of 1). This makes the mutation scheme of the classical DE more time-dependent, a feature we found beneficial. In what follows we will refer to the modified version as DELG as it incorporates both the local and global models synergistically.

Table 1. Benchmark Functions Used

Function	Mathematical Representation
Sphere function	$f_1(x) = \sum_{i=1}^{n} x_i^2$
Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
Rastrigin	$f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
Griewank	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Ackley	$f_5(x) = -20\exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} x_i^2)} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20$
Shekel's Foxholes	$f_6(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}]^{-1}$

IV. Experiments

A. Benchmark Functions Used

We have used six well-known benchmarks (Table 1) [11] to evaluate the performance of the proposed DELG algorithm. In Table 1, *n* represents the number of dimensions (we used *n* = 25, 50, 75 and 100). The first two test functions are unimodal, having only one minimum each. The others are multimodal, with a considerable number of local minima in the region of interest. All benchmark functions except f_6 have the global minimum at the origin or very near to the origin [11]. For Shekel's foxholes (f_6), the global minimum is at (-31.95, -31.95) and f_6 (-31.95, -31.95) ≈ 0.998, and the function has only two dimensions. Table 2 summarizes the initialization and search ranges used for these functions. We have used an asymmetrical initialization procedure following the work reported in [12].

B. Other Problems Tested

B.1 The Spread Spectrum Radar Poly-phase Code Design Problem

A famous NP-hard problem of optimal design arises in the field of spread spectrum radar poly-phase code design [13]. Such a problem serves as a useful test of the applicability of global optimization algorithms like DE.

Table 2. Search Ranges of Benchmark Functions

Function	Range of search	Range of Initialization
f_1	$(-100, 100)^n$	$(50, 100)^n$
f_2	$(-100, 100)^n$	$(15, 30)^n$
f_3	$(-10, 10)^n$	$(2.56, 5.12)^n$
f_4	$(-600, 600)^n$	$(300, 600)^n$
f_5	$(-32, 32)^n$	$(15, 32)^n$
f_6	$(-65.536, 65.536)^2$	$(0, 65.536)^2$

The problem can be formally stated as:

$$\text{global min } f(x) = \max\{\varphi_1(x),....,\varphi_{2m}(x)\}$$
$$x \epsilon X$$
$$X = \{(x_1,....,x_n) \in R^n \mid 0 \le x_j \le 2\pi, j = 1,...,n\} \quad (12)$$

where m = 2n – 1 and

$$\varphi_{2i-1}(x) = \sum_{j=i}^{n} \cos(\sum_{k=|2i-j-1|-1}^{j} x_k), \quad i = 1, ..., n$$
$$\varphi_{2i}(x) = 0.5 + \sum_{j=i+1}^{n} \cos(\sum_{k=|2i-j-1|-1}^{j} x_k), \quad i=1,.....,n-1$$
$$\varphi_{m+i}(x) = -\varphi_i(x), \quad i=1,.....,m \quad (13)$$

B.2 Optimization of the Lennard Jones Atomic Cluster

One of the simplest to describe yet most difficult to solve problems in computational chemistry is the determination of molecular conformation. A molecular conformation problem can be described as finding the global minimum of a suitable potential energy function, which depends on relative atom positions. The Lennard Jones potential function [14] models the pair-wise interaction between non-bonded atoms in a substance. A system containing more than one atom, whose Van der Waals interaction can be described by LJ potential, is called an LJ cluster. In this work we use a scaled LJ potential which for the i-th and j-th atom of the cluster may be expressed as

$$V = \sum_{i<j} (\frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}}) \quad (14)$$

The LJ potential for a single pair of two neutral atoms is a simple unimodal function with overall minima at 1 and energy -1. However, in a complex system, many atoms interact and we need to sum up the LJ potential functions for each pair of the atoms. This results in a complex energy function riddled with multiple local minima.

The Lennard Jones atomic cluster problem can be formulated as follows. Let $z^i = (z_1^i, z_2^i, z_3^i)^T$ represent the coordinates of the atoms in 3-D Euclidean space. Let

$Z = \{(z^1)^T, ..., (z^N)^T\}^T$, where N is the number of atoms in the cluster. The LJ potential of a pair of atoms (i, j) is given by eqn. (12) where $r_{ij} = \|z^i - z^j\|$. The LJ cluster problem then reduces to finding the coordinates of the atoms in the cluster in such a fashion that the energy function V(Z), shown below, may attain its global minima:

$$\min_{Z \in R^{3N}} V(Z) = \sum_{i<j} v(\|z^i - z^j\|) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(\frac{1}{\|z^i - z^j\|^{12}} - \frac{2}{\|z^i - z^j\|^{6}}\right) \quad (15)$$

B.3 Simulation Strategy

Simulations were carried out to obtain a comparative performance analysis of the proposed method with respect to: (a) DE/rand/1/bin (b) DE/best/1/bin and (c) DE/best/2/bin. For all the four algorithms we used the same population size, which is 10 times the dimension of the problem. To make the comparison fair, the populations for all the competitor algorithms (for all problems tested) were initialized using the same random seeds.

We chose the number of fitness evaluations (FE) as a measure of the computational cost. The number of fitness evaluations roughly equals the product of the population size and the number of generations.

Twenty independent runs of each of the four algorithms were carried out and the average and the standard deviation of the best-of-run values were recorded. Different maximum number of FEs were used according to the complexity of the problem. For all benchmarks (excluding Shekel's Foxholes function f_6) the stopping criterion was set as reaching a fitness of 10^{-5}. However, for Shekel's Foxholes function (f_6) it was fixed at 0.998. The spread spectrum radar polyphase code design problem was tested vaying n from 2 to 20, and for the Lennard Jones potential minimization problem the number of atoms was varied from 2 to 30. However we report results of just two of the most difficult problem instances in each case (n = 19, 20 for radar polyphase and number of atoms = 29, 30 for Lennard Jones potential function) owing to the space limitations.

For all the versions of DE (including the newly developed DELG), we chose the cross-over rate Cr = 0.9. For the classical DE versions, scale factor F = λ = 0.8 was used. All the algorithms were developed from scratch in Visual C++ on a Pentium IV, 2.2 GHz PC, with 512 KB cache and 2 GB of main memory in Windows Server 2003 environment.

V. Results

The following performance measures are used for our comparative study: (a) quality of the final solution, (b) speed of convergence towards the optimal solution, (c) success rate (frequency of hitting the optimum), and (d) scalability of the algorithms against the growth of problem dimensions. Table 3 compares the algorithms on the quality of the optimum solution. The mean and the standard deviation (within parentheses) of the best-of-run values for 20 independent runs of each of the four algorithms are presented in Table 3. Missing values of standard deviation in this table indicate a zero standard deviation. The best solution in each case has been shown in bold. Table 4 shows results of unpaired *t*-tests between DELG and the best of the three competing algorithms in each case (standard error of difference of the two means, 95% confidence interval of this difference, the *t* value, and the two-tailed *P* value). For all cases in Table 4, sample size = 20 and degrees of freedom = 38. It is interesting to see from Tables 3 and 4 that the proposed method meets or beats the nearest competitor in a statistically significant way.

Table 5 shows, for all test functions and all algorithms, the number of runs (out of 20) that managed to find the optimum solution (within the given tolerance). In Table 6 we report the mean number of function evaluations (FEs) and standard deviations (within parenthesis) required by the algorithms to converge within the prescribed cut-off value. The entries marked NA (Not Applicable) in Table 6 imply that no run of the corresponding algorithms converged to the cut-off within the predetermined maximum number of FE. Missing values of standard deviation in this table also indicate a zero standard deviation. In Figure 1 we have graphically presented the rate of convergence of all the methods for all the functions (in 100 dimensions). Figure 2 shows the scalability of the four methods on six test functions - how the average computational cost (measured in number of function evaluations) to find the solution varies with an increase in the dimensionality of the search space. These results show that the proposed method leads to significant improvements in most cases.

Tables 7, 8, and 9 report results of similar experiments (mean best-of-the-run solution for 50,000 FEs, unpaired t-test results and the mean number of FEs required for each algorithm to reach a predetermined cut-off fitness value, respectively) done with the spread spectrum radar polyphase code design problem. Tables 10, 11 and 12 summarize the results for Lennard Jones potential minimization problem (number of atoms N = 29 and 30). In Tables 9 and 12, we have chosen the cut-off value to be somewhat larger than the minimum objective function value found by each algorithm in Tables 7 and 10, respectively. Figures 3 and 4 graphically present the rate of convergence of the algorithms for these two problems (graphs have been shown for 20 dimensions for the radar code design problem and 30 molecules for the molecular potential function problem).

Table 3. Average and the standard deviation of the best-of-run solution for 20 runs tested on six benchmark functions (for all cases except f_6, the cut-off value used is 1.00e-05, while for f_6 it is 0.998).

Function	Dim	Max^m FE	Mean Best Value (Standard Deviation)			
			DE/rand/1/bin	DE/best/1/bin	DE/best/2/bin	DELG
f_1	25	50,000	**1.00e-05**	**1.00e-05**	**1.00e-05**	**1.00e-05**
	50	1×10^5	**1.00e-05**	**1.00e-05**	**1.00e-05**	**1.00e-05**
	75	5×10^5	**1.00e-05**	**1.00e-05**	**1.00e-05**	**1.00e-05**
	100	1×10^6	**1.00e-05**	**1.00e-05**	**1.00e-05**	**1.00e-05**
f_2	25	50,000	1.6342e-01 (4.253e-05)	3.7691e-01 (5.209e-04)	3.4116e-01 (9.465e-05)	**9.4821e-04 (2.592e-05)**
	50	1×10^5	8.8639e-01 (3.237e-05)	2.7862e-01 (9.282e-05)	3.7821e-01 (3.283e-04)	**8.6752e-03 (5.746e-05)**
	75	5×10^5	5.2318e+00 (3.911e-02)	1.8232e+01 (5.243e-03)	2.8734e+00 (3.309e-02)	**2.4687e-02 (3.766e-05)**
	100	1×10^6	1.8704e+01 (6.282e-02)	3.5624e+00 (3.366e-03)	5.3675e+00 (4.568e-03)	**6.8669e-01 (4.346e-04)**
f_3	25	50,000	**1.00e-05**	**1.00e-05**	**1.00e-05**	**1.00e-05**
	50	1×10^5	**1.00e-05**	**1.00e-05**	2.5826e-04 (3.271e-07)	**1.00e-05**
	75	5×10^5	6.8734e-04 (5.329e-08)	1.2038e-04 (4.917e-08)	3.4039e-04 (2.112e-08)	**1.00e-05**
	100	1×10^6	2.1935e-03 (1.186e-08)	8.7325e-03 (5.521e-09)	4.5925e-02 (6.211e-09)	**1.00e-05**
f_4	25	50,000	**1.00e-05**	**1.00e-05**	**1.00e-05**	**1.00e-05**
	50	1×10^5	5.3281e-04 (3.291e-08)	4.5043e-04 (4.895e-09)	6.8528e-03 (5.271e-08)	**1.00e-05**
	75	5×10^5	6.4543e-03 (3.434e-06)	2.1309e-03 (7.281e-06)	9.233e-02 (4.816e-06)	**3.8971e-04 (3.832e-07)**
	100	1×10^6	8.1232e-03 (1.613e-06)	6.3298e-01 (1.217e-05)	1.3483e-01 (6.312e-06)	**8.8729e-04 (1.536e-07)**
f_5	25	50,000	9.4543e-05 (2.454e-09)	5.0291e-05 (4.883e-09)	1.3460e-04 (6.393e-08)	**1.00e-05**
	50	1×10^5	3.4872e-04 (8.413e-07)	8.0289e-04 (1.629e-07)	6.0322e-04 (9.324e-09)	**1.00e-05**
	75	5×10^5	1.0084e-03 (2.923e-06)	4.8271e-03 (4.294e-06)	9.0892e-02 (5.265e-05)	1.817e-04 (1.958e-07)
	100	1×10^6	3.6298e-02 (9.127e-04)	5.7309e-02 (1.682e-06)	3.2175e-01 (5.636e-05)	**2.2627e-04 (6.347e-05)**
f_6	2	50,000	**9.9980e-01**	9.998162e -01 (2.783-13)	**9.998613e -01 (4.264e-08)**	**9.9980e-01**

Table 4. Results of unpaired t-tests on the data of Table 3

Fn, Dim	Std. Err	*t*	95% Conf. Intvl	Two-tailed *P*	Significance
f_2, 25	0.000	14596.0003	0.1624to 0.1625	< 0.0001	**Extremely significant**
f_2, 50	0.000	11058.6669	0.26989 to 0.2699	< 0.0001	**Extremely significant**
f_2, 75	0.007	2.4139	2.8337 to 2.8637	< 0.0001	**Extremely significant**
f_2, 100	0.001	3789.2725	2.8742 to 2.8775	< 0.0001	**Extremely significant**
f_4, 75	0.000	1067.9956	0.0018 to 0.0017	<0.0001	**Extremely Significant**
f_4, 100	0.000	19971.6322	0.0072 to 0.007234	<0.0001	**Extremely Significant**
f_5, 75	0.000	1262.0075	0.00082 to 0.00083	< 0.0001	**Extremely significant**
f_5, 100	0.000	176.3219	0.035 to 0.036	< 0.0001	**Extremely significant**

Table 5. Number of runs converging to the cut off fitness value for six benchmark functions.

Function	Dim	No. of runs converging to the cut off			
		DE/rand /1/bin	DE/best/ 1/ bin	DE/best/ 2/ bin	DELG
f_1	25	**20**	**20**	**20**	**20**
	50	**20**	**20**	**20**	**20**
	75	**20**	**20**	**20**	**20**
	100	**20**	**20**	**20**	**20**
f_2	25	8	3	2	**16**
	50	0	0	0	**14**
	75	0	0	0	**9**
	100	0	0	0	**5**
f_3	25	**20**	**20**	**20**	**20**
	50	**20**	**20**	12	**20**
	75	14	11	10	**20**
	100	6	5	2	**20**
f_4	25	**20**	**20**	**20**	**20**
	50	15	12	10	**20**
	75	6	8	9	**18**
	100	8	4	6	**13**
f_5	25	13	15	8	**20**
	50	12	17	6	**20**
	75	5	8	3	**14**
	100	2	4	1	**12**
f_6	2	**20**	12	14	**20**

Table 6. Mean no. of FE and standard deviation required to converge to the cut-off fitness over the successful runs.

Function	Dim	Mean and Std Dev of FE required to reach the cut off			
		DE/rand/1/bin	DE/best/1/bin	DE/best/2/bin	DELG
f_1	25	8682.05 (11.725)	9306.65 (15.873)	9261.85 (5.968)	**4430.75 (28.855)**
	50	22193. 00 (3.961)	24019. 65 (45.911)	25271. 50 (69.313)	**17192. 50 (10.149)**
	75	45092.45 (291.49)	44915.05 (20.982)	37289.65 (12.583)	**35784.35 (15.722)**
	100	989173.65 (23.88)	900024.60 (32.814)	934452.45 (43.821)	**381921.50 (73.818)**
f_2	25	7621.125 (24.821)	8915.35 (4.122)	9372.50 (3.228)	**7093.25 (1.955)**
	50	NA	NA	NA	**13628.83 (3.92)**
	75	NA	NA	NA	**32055.67 (23.86)**
	100	NA	NA	NA	**601291.20 (37.69)**
f_3	25	34884.45 (4.841)	46828.05 (5.583)	22529.45 (5.902)	**15775.75 (4.858)**
	50	67124.05 (34.991)	95614.95 (45. 526)	44169.50 (10.697)	**34752.15 (5.949)**
	75	416454.67 (7.806)	349772.57 (56.92)	213555.00 (20.95)	**175148.35 (34.95)**
	100	892452.50 (15.00)	752754.40 (4.975)	781231.50 (12.828)	**702321.55 (6.23)**
f_4	25	8725.55 (4.806)	12365.75 (7.039)	5481.05 (3.207)	**4075.15 (5.656)**
	50	17934.85 (18.910)	27568.33 (12.876)	12783.70 (12.843)	**9382.25 (8.355)**
	75	409281.33 (30.663)	447328.15 (35.723)	381732.67 (18.587)	**302615.13 (22.637)**
	100	726121.375 (12.655)	837261.25 (36.786)	562311.35 (31.747)	**509182.00 (18.753)**
f_5	25	37261.67 (9.57)	45930.57 (30.56)	12905.125 (11.555)	**9720.50 (19.547)**
	50	62918.57 (22.546)	90382 (32.822)	27483.46 (26.465)	**20483.05 (22.849)**
	75	267211.25 (7.558)	389222.40 (29.804)	169023.66 (37.649)	**57833.05 (32.593)**
	100	890332.50 (13.540)	990321.25 (21.525)	680491	**109321.05 (47.465)**
f_6	2	16472.85 (24.242)	27502.40 (14.803)	18291.86 (13.447)	**10209.25 (7.81)**

Table 7. Average and the standard deviation of the best-of-run solution for 20 runs for spread spectrum radar poly-phase code design problem (number of dimensions n = 19 and n = 20). For all cases each algorithm was run for 50,000 FEs.

n	Mean best-of-run solution (Std Dev)			
	DE/rand/1/bin	DE/best/1/ bin	DE/best/2/ Bin	DELG
19	7.4849e-01 (8.93e-03)	7.5245e-01 (6.83e-03)	7.583e-01 (9.56e-04)	**7.4439e-01 (5.84e-04)**
20	8.5746e-01 (4.83e-03)	9.0932e-01 (4.63e-03)	8.3982e-01 (3.98e-03)	**8.0304e-01 (2.73e-03)**

Table 8. Results of unpaired t-tests on the data of Table 7

n	Std. Err	*t*	95% Conf. Intvl	Two-tailed *P*	Significance
19	0.002	2.0489	0.000049 to 0.008150	0.0474	significant
20	0.001	59.621	0.035531 to 0.038029	< 0.0001	**Extremely significant**

Table 9. Average no. of function evaluations (for 20 runs) and standard deviations for spread spectrum radar poly-phase code design problem to converge to a given cut-off value of the fitness function

N	Cut off Value of the Objective function	Mean No. of FE required to reach the cut-off value			
		DE/rand/1/ bin	DE/best/1/ bin	DE/best/2/ bin	DELG
19	-121.00e +00	49562.50 (3.29)	47229.10 (9.45)	48472.95 (21.54)	36993.85 (14.82)
20	-125.00e +00	49908.35 (12.54)	47632.65 (18.36)	46432.55 (5.39)	40229.05 (2.43)

Table 10. Average and the standard deviation of the best-of-run solution for 20 runs for Lennard Jones potential minimization problem (number of atoms N = 29 and N = 30). For all cases, each algorithm was run for 50,000 FEs.

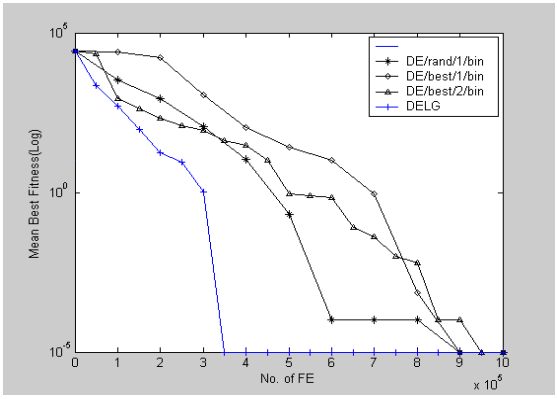
N	Potential Energy of the configuration			
	DE/rand/1/bin	DE/best/1/ bin	DE/best/2/ Bin	DELG
29	-122.027e+00 (1.2128e-02)	-121.819e+00 (2.3811e-03)	-121.726e+00 (3.2891e-04)	**-123.432e+00 (3.2819e-03)**
30	-126.473e+00 (1.2239e-05)	-125.381e+00 (9.3931e-03)	-125.792e+00 (5.9729e-04)	**-128.717e+00 (8.3782e-05)**

Table 11. Results of unpaired t-tests on the data of Table 10

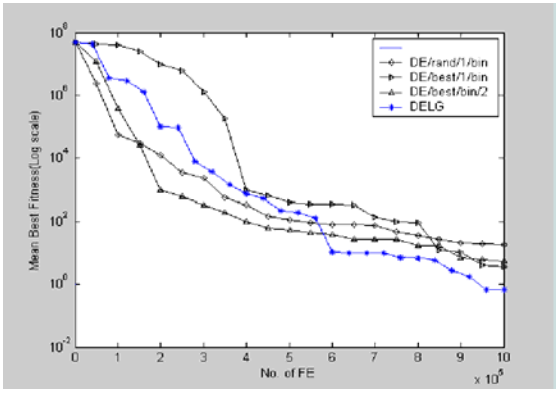
N	Std. Err	*t*	95% Conf. Intvl	Two-tailed *P*	Significance
29	0.003	500.099	1.399312 to 1.410687	< 0.0001	**Extremely significant**
30	0.000	118522.8260	2.243961 to 2.244038	< 0.0001	**Extremely significant**

Table 12. Mean no. of FEs (for 20 runs) and standard deviations for Lennard Jones potential minimization.

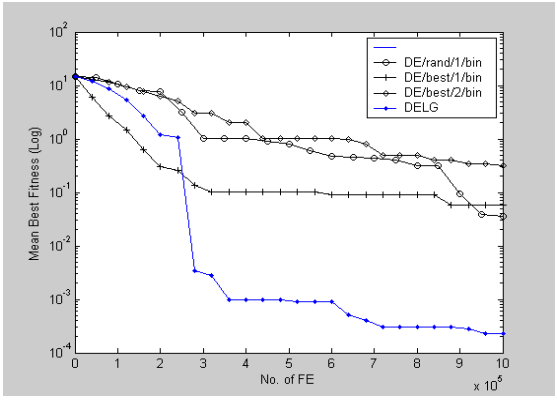
N	Cut off Value of the Objective function	Mean No. of FE required to reach the cut-off value			
		DE/rand/1/ bin	DE/best/1/ bin	DE/best/2/ bin	DELG
29	-121.00e +00	49562.50 (3.29)	47229.10 (9.45)	48472.95 (21.54)	**36993.85 (14.82)**
30	-125.00e +00	49908.35 (12.54)	47632.65 (18.36)	46432.55 (5.39)	**40229.05 (2.43)**



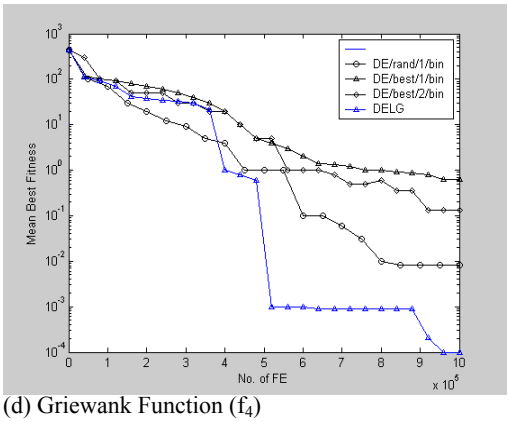
(a) Sphere Function (f_1)



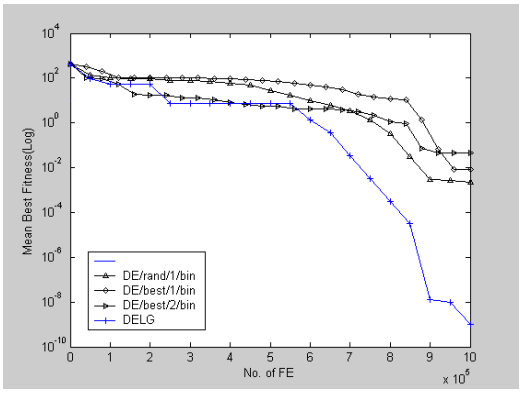
(b) Rosenbrock Function (f_2)



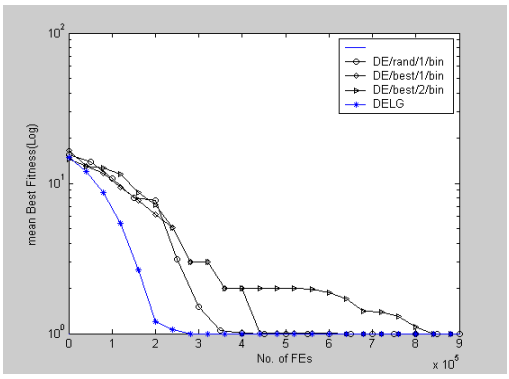
(c) Rastrigin Function (f_3)



(d) Griewank Function (f_4)

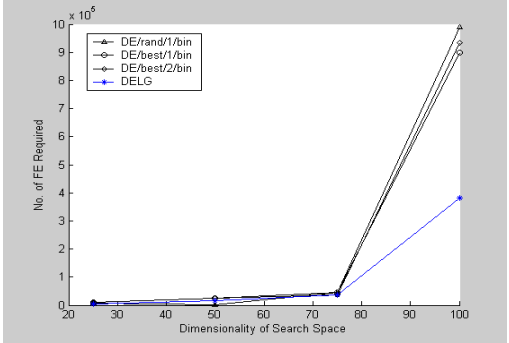


(e) Ackley Function (f_5)

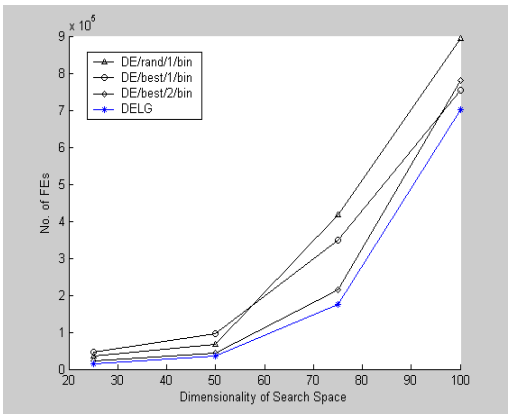


(f) Shekel's Foxhole Function (f_6)

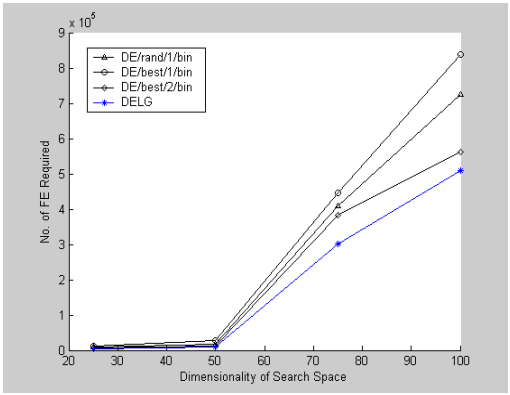
Fig.1 Progress to the optimum solution (all plots are for dimension = 100, except Shekel's foxhole which is 2D)



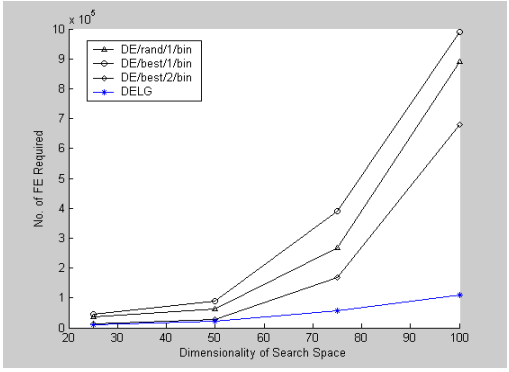
(a) Sphere Function (f_1)



(b) Rastrigin Function (f_3)



(c) Griewank Function (f_4)



(d) Ackley Function (f_5)

Fig 2. Variation of mean number of FE required for convergence with increase in dimensionality of the search space.

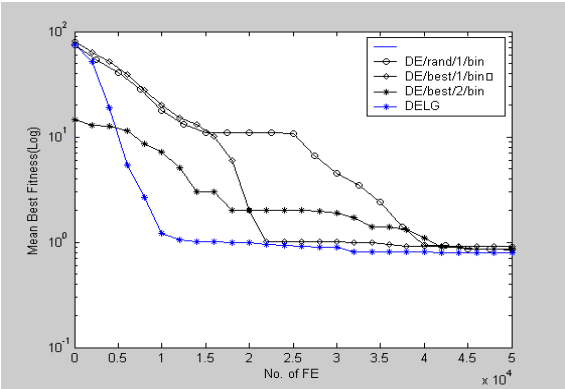


Fig.3 Progress to the optimum solution for Spread Spectrum Radar Polyphase Code Design Problem (with n = 20)

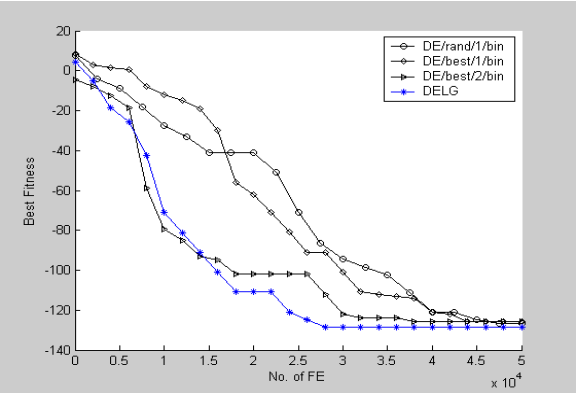


Fig.4 Progress to the optimum solution for Lennard Jones Potential Function Minimization Problem (with N = 30)

VI. Conclusion

A new, improved DE has been presented, based on the use of the local neighborhood of each vector. The new scheme attempts to make a judicious use of the exploration and exploitation abilities of the search mechanism and is therefore more likely to avoid false or premature convergence. It has also been shown to improve upon the convergence speed and scalability of the classical DE. The new DE-variant has been compared against three other popular variants of DE (DE/rand/1/bin, DE/best/2/bin and DE/best/1/bin) using a six-function test suite and two additional real-world problems. The following performance metrics have been used: (a) solution quality, (b) speed of convergence, (c) frequency of hitting the optimum, and (d) scalability. The proposed variant has been shown to meet or beat the nearest competitor in a statistically meaningful way for most of the tested problems.

Bibliography

[1]Storn, R., Price, K.: Differential evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 11(4) (1997) 341–359.

[2]Vesterstrøm, J., Thomson, R.: A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems, Proc. Sixth Congress on Evolutionary Computation (CEC-2004). IEEE Press.

[3]Stumberger, G., Dolinar, D. ,Pahner, U. and Hameyer, K., Optimization of radial active magnetic bearings using the finite element technique and differential evolution algorithm, *IEEE Transactions on Magnetics*, vol. 36, no. 4, pp.1009–1013, 2000.

[4] Lampinen, J., (2002). A Constraint Handling Approach for the Differential Evolution Algorithm. Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, Hawaii, May 12-17, 2002. Vol.2, pp.1468–1473. ISBN 0-7803-7282-4.

[5]Omran, M., Engelbrecht, A, P., Salman, A., Differential Evolution Methods for Unsupervised Image Classification, Proc. Seventh Congress on Evolutionary Computation (CEC-2005). IEEE Press.

[6]Fan H.-Y., Lampinen, J.A Trigonometric Mutation Approach to Differential Evolution. In Proc. of the EUROGEN2001 Conference, Athens, Greece, September 19-21, 2001, pp. 65–70. CIMNE, Barcelona (Spain).

[7]Das, S., Konar, A., Chakraborty, U.K., Two improved differential evolution schemes for faster global search, ACM-SIGEVO Proceedings of GECCO, Washington D.C., June 2005, pp. 991-998.

[8] Price, K., Storn, R., and Lampinen, J., Differential Evolution - A Practical Approach to Global Optimization, Springer, ISBN: 3-540-20950-6.

[9]Kennedy, J, Eberhart R.: Particle swarm optimization. Proc. IEEE Int. conf. Neural Networks. (1995) 1942-1948

[10]Parsopoulos, K.E., Vrahatis, M.N, UPSO: A unified particle swarm optimization scheme. In: Lecture Series on Computer and Computational Sciences, Vol. 1, Proc. Int. Conf. Comput. Meth. Sci. Eng. (ICCMSE 2004), VSP International Science Publishers, Zeist, the Netherlands (2004) 868–873

[11]Yao, X., Liu, Y., Lin, G. Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation, vol 3, No 2, (1999) 82-102.

[12]Angeline, P. J. Evolutionary optimization versus particle swarm optimization: Philosophy and the performance difference, *Lecture Notes in Computer Science (vol. 1447), Proceedings of 7th International Conference on. Evolutionary Programming* – Evolutionary Programming VII (1998) 84-89.

[13]Mladenovic, Petrovic, Kovacevic-Vuijcic, Cangalovic, Solving spread-spectrum radar polyphase code design problem by tabu search and variable neighbourhood search, European Journal of Operational Research, 153(2003) 389-399.

[14] Deaven, D.M., Tit, N., Morris, J.R., and Ho, K.M. Structural optimization of Lennard-Jones clusters by a genetic algorithm, Chemical Physics Letters, (256) 1996.