A New Differential Evolution with Improved Mutation Strategy

Pavel Bhowmik, Sauvik Das, Amit Konar, Swagatam Das and Atulya K. Nagar

Abstract—**The paper employs Lagrange's mean value theorem of differential Calculus to design a new strategy for the selection of parameter vectors in the Differential Evolution (DE) algorithm. Classical differential evolution selects parameter vectors randomly to obtain the donor vectors. These donor vectors thus cannot be directly used as trial solution to the optimization problem. The recombination step indeed is very useful to generate potential trial solutions. The proposed algorithm eliminates the recombination step as the trial solutions can be directly generated by the extended mutation step only. Performance analysis of the proposed algorithm with respect to standard benchmark functions reveals that both in expected convergence time and accuracy in solutions, the proposed algorithm outperforms classical DE/rand/1. Besides extension in mutation strategy, an adaptive selection strategy in the scaling factor *F* also improves the performance of the proposed algorithm. In addition, the proposed algorithm outperforms classical DE in noisy optimization problem. Further, the number of function evaluation with scaled up dimensions of the optimization problem adds insignificantly small complexity in comparison to that in classical differential evolution to meet up a prescribed level of accuracy in solution quality.**

I. INTRODUCTION

Pioneered by Storn and Price, Differential Evolution (DE) [12] is a derivative-free evolutionary algorithm often employed to determine global optima in a multimodal rough and nonlinear search landscape of multiple variables. The central idea behind DE is a scheme for generating offspring vectors by exploiting the difference vector based perturbation instead of relying on some pre-defined probability density functions.

Like any evolutionary/swarm optimization algorithm, in DE too we need to balance between exploration and exploitation by judicious selection of parameters of the algorithm. A fast pre-mature convergence is not expected. At the same time a quality solution at the expense of very large computational cost is not desired. In practice, we need to have a DE that ensures quality solution within a feasible number of function evaluations. There exist several approaches to balance the trade-off between exploration of the search space and expected convergence time to handle the above problem. However, nevertheless very few approaches truly could provide an acceptable solution. In this paper, we propose a new version of DE, which is expected to eliminate the impasse by improving the mutation strategy using one fundamental theorem of differential calculus. Philosophically, the proposed mutation strategy identifies two n-dimensional points corresponding to one parameter vector over the n-dimensional search landscape in a manner, such that a sense of (virtual) gradient exists between the parameter vector and each selected point. Consequently, out of three points (the parameter vector and pair of selected points) the point with highest fitness measure gives a chance to examine the local optima on the search landscape. This idea motivated us to realize a new version of DE, which fortunately gives superb performance when tested with 7-test suites. Both the quality of solution and number of function evaluation give better results in comparison to DE/rand/1 with respect to the selected 7-benchmark functions. An adaptive selection strategy of the scale factor *F*, undertaken here, also controls the over-exploration of the population.

Pavel Bhowmik, Sauvik Das, Amit Konar, Swagatam Das are with Jadavpur Universty, Kolkata-700032, India (e-mail: bpavel88@gmail.com, sauvik.095@gmail.com, konaramit@yahoo.co.in, swagatamdas19@yahoo.co.in).

Atulya K. Nagar is with the dept. of Computer Science, Liverpool Hope University, Liverpool, UK. (e-mail: nagara@hope.ac.uk).

The paper is classified into nine sections. In section II, we present the classical DE algorithm for convenience of a novice reader. In section III, we discuss about various research works performed to modify the classical algorithm of DE. In section IV, we provide the mathematical foundation of our work that leads us to develop the extended-DE algorithm in section V. Computer simulation and performance analysis with standard benchmark functions is undertaken in section VI. The scope of handling noisy optimization is taken care of in section VII. The performance of the proposed DE with respect to number of function computations does not increase significantly. The issue of the curse of dimensionality is undertaken in section VIII. Conclusions are listed in section IX.

II. THE CLASSICAL DE

An iteration of the classical DE algorithm consists of the four basic steps – initialization of a population of vectors, mutation, crossover or recombination and finally selection. The main steps of classical DE are given below.

I. Set the generation number $t = 0$ and randomly initialize a population of NP individuals $\vec{P}_t = \{\vec{X}_1(t), \vec{X}_2(t), \dots, \vec{X}_{NP}(t)\}$ with $\vec{X_i}(t) = \{x_{i,1}(t), x_{i,2}(t), \dots, x_{i,D}(t)\}$ and each individual uniformly distributed in the range $[\vec{X}_{min}, \vec{X}_{max}]$, where $\vec{X}_{min} = \{x_{min,1}, x_{min,2}, \dots, x_{min,D}\}$ and $\vec{X}_{max} = \{x_{max,1}, x_{max,2}, \dots, x_{max,D}\}$ with $i = [1, 2, \dots, NP]$

```
II.    while stopping criterion is not reached, do
         for $i = 1\ to\ NP$
```

```
        II. a   Mutation:
            Generate a donor vector
            $\vec{V_i}(t) = \{v_{i,1}(t), v_{i,2}(t), \dots, v_{i,D}(t)\}$
            corresponding to the $i^{th}$ target vector $\vec{X}_1(t)$
            via one of the mutation schemes of DE.
        II. b   Crossover:
            Generate trial vector
            $\vec{U}_1(t) = \{u_{i,1}(t), u_{i,2}(t), \dots, u_{i,D}(t)\}$ for the
            $i^{th}$ target vector $\vec{X}_1(t)$ through binomial
            crossover or exponential crossover.
        II. c   Selection:
            Evaluate the trial vector $\vec{U}_1(t)$
            if $f(\vec{U}_1(t)) \le f(\vec{X}_1(t))$, then $\vec{X}_1(t+1) =$
                $\vec{U}_1(t), f(\vec{X}_1(t+1)) = f(\vec{U}_1(t))$
                if $f(\vec{U}_1(t)) \le f(\vec{X}_{best}(t))$, then
                    $\vec{X}_{best}(t) = \vec{U}_1(t)$,
                    $f(\vec{X}_{best}(t)) = f(\vec{U}_1(t))$
                end if
                else $\vec{X}_1(t+1) = \vec{X}_1(t)$
                and $f(\vec{X}_1(t+1)) = f(\vec{X}_1(t))$
            end if
    end for
    II. d   Increase the counter value $t = t + 1$.
end while
```

The parameters used in the algorithm namely scaling factor '*F*' and crossover rate '*CR*' should be initialized before calling the 'while' loop. The terminate condition can be defined in many ways, a few of which include

i) fixing the number of iterations *N*.

ii) when best fitness of population does not change appreciably over successive iterations.

iii) Either of (i) and (ii), whichever occurs earlier.

There are two kinds of crossover/ recombination schemes in DE family of algorithms, namely exponential and binomial recombination. [14], [15]. In exponential crossover, initially, an integer n is chosen randomly among the numbers [0, D-1]. This 'n' denotes the starting point in the target vector, from where the crossover starts. Another integer L is chosen from the interval [1, D]. L denotes the number of components the donor vector actually contributes to the target. The trial vectors are chosen as mentioned below:

$$u_{i,j}(t) = v_{i,j}(t) \quad \text{for} \quad j = \langle n\rangle_D, \langle n+1\rangle_D, \dots, \langle n+L-1\rangle_D. \tag{1.a}$$

$$= x_{i,j}(t) \text{ for } j \in [0, D-1]. \tag{1.b}$$

where the angular bracket $\langle\,\rangle_D$ denotes a modulo function with modulus D. The algorithm to choose L is given as:

```
L = 0;
do
{
    L=L+1;
} while (rand (0, 1) < CR) AND (L<D));
```

Hence in effect Probability that (L ≥ υ) is equal to $(CR)^{υ-1}$ for any υ > 0. For each donor vector, a new set of n and L must be chosen randomly as shown above.

On the other hand, binomial crossover [2] is performed on each of the D variables whenever a randomly picked number between 0 and 1 is within the *CR* value. In this case the number of parameters inherited from the mutant has a (nearly) binomial distribution. The scheme may be outlined as,

$$u_{i,j,G} = v_{i,j,G} \text{ if } rand_{i,j}(0,1) \le CR \text{ or } j = j_{rand}$$
$$= x_{i,j,G} \text{ otherwise}$$

where $rand_{i,j}(0,1) \in [0,1]$ is a uniformly distributed random number. $j_{rand} \in [1,2,\dots,D]$ is a randomly chosen index, which ensures that $\overrightarrow{U_{i,G}}$ gets at least one component from $\overrightarrow{V_{i,G}}$. It is instantiated once for each vector in one generation. We note that for this additional demand, *CR* is only approximating the true probability that a component of the trial vector will be inherited from the donor.

There are 5 basic schemes of DE based on different mutation strategies. They are:

DE/rand/1: $$v_i = x_{1i} + F(x_{2i} - x_{3i}) \tag{2}$$

DE/best/1: $$v_i = x_{BESTi} + F(x_{1i} - x_{2i}) \tag{3}$$

DE/current-to-best/1:

$$v_i = x_i + F(x_{BEST} - x_i) + F(x_{1i} - x_{2i}) \tag{4}$$

DE/best/2:

$$v_i = x_{BEST} + F(x_{1i} - x_{2i}) + F((x_{3i} - x_{4i})) \tag{5}$$

DE/rand/2: $$v_i = x_{1i} + F(x_{2i} - x_{3i}) + F(x_{4i} - x_{5i}) \tag{6}$$

Here v_i is the donor vector, x_{BEST} represents the best vector in the current generation, x_{BESTi} is the i^{th} component of x_{BEST}, x_{1i} represents i^{th} component of x_1, x_{2i} represents i^{th} component of x_2, and so on, where x_1, x_2 etc are randomly selected vector from the population.

Schemes DE/rand/1 and DE/current-to-best/1, are most often used in practice due to their good performance. [13], [11] In this paper, we have considered the scheme DE/rand/1.

III. Related Works on Modification of Classical DE algorithm

Current research on DE is targeted at: i) control parameter selection of the algorithm, ii) improving mutation and/or selection strategy, so as to enhance its performance in both quality of solution and speed of convergence. Among the well-known works on parameter selection, [9], [4], [10], [3] need special mention. Some of the interesting works on selection strategy includes [6], [1], [5].

Gamperle et al. [7] estimated different parameter settings for DE on the Sphere, Rosenbrock's, and Rastrigin's functions. Their results revealed that the global optimum searching capability and the convergence speed are very sensitive to the choice of control parameters *NP*, *F* and *CR*. Furthermore, a reasonable choice of the population size NP is between 3D and 8D, where D is the dimensionality of the problem, the scaling factor *F* = 0.6 and the crossover rate *CR* is between [0.3, 0.9].

Yang et al. [16] proposed a self-adaptive differential evolution algorithm with neighborhood search (SaNSDE). SaNSDE includes self-adaptive choice of the mutation strategy between two alternatives, self-adaptation of the scale factor *F*, and self-adaptation of the crossover rate *CR*. They applied their proposed method on various benchmark functions and proved their scheme is better than NSDE.

Das et al. [4] introduced two schemes for adapting the scale factor *F* in DE. First they varied *F* randomly between 0.5 and 1.0 in successive iterations. Next, they decreased *F* linearly from 1.0 to 0.5. Thus, sampling of diverse zones of the search space during the early stages of exploration can be done. Later, a dwindling scaling factor facilitates to adapt to the movements of trial solutions finely, so that they can explore the interior of a relatively small space in which the suspected global optimum lies.

IV. Mathematical foundations of the Proposed DE

The extended-DE presented here is based on one fundamental theorem of differential calculus, well known as Lagrange's Mean Value Theorem (MVT). The theorem states as follows.

THEOREM: *If $f(x)$ is continuous in the (closed) interval [a, b] and differentiable in the (open) interval (a, b), and b>a then there exists at least one point c where:*

$$\left.\frac{df}{dx}\right|_{x=c} = \frac{f(b)-f(a)}{b-a} \quad . \tag{7}$$

A mathematical simplification of expression (1) yields

$$b = a + \frac{f(b)-f(a)}{\left.\frac{df}{dx}\right|_{x=c}}$$

$$b = a + \frac{f(b)-f(a)}{f(b')-f(a')}(b'-a')$$

where $b'$and a' are close enough to $x = c$ such that $c > a'$ and $b' > c$, and consequently

$$\left.\frac{df}{dx}\right|_{x=c} = \frac{f(b\prime)-f(a\prime)}{b\prime-a\prime}$$

It is needless to say that in the present context,

$a < a'$, and $b' < b$.

An extension of the MVT from one dimension to multi-dimensional function is given in expression (8).

$$b_i = a_i + (b_i' - a_i')\left[\frac{f(b)-f(a)}{f(b')-f(a')}\right] \tag{8}$$

where b_i, and a_i are the i^{th} components of vector $\vec{b}$ and $\vec{a}$ respectively. We here presumed that $f\,(x_1, x_2, \dots, x_n)$ is an n-dimensional function of variables: $x_1, x_2, \dots, x_n$.

Expression (8) has some form of similarity with the basic expression for the donor vector obtained after mutation. The term within the square bracket in (8) is similar to scaling factor *F*. However, expression (8) gives us an interesting insight to selection of three parameter vector to obtain the donor vector b. This was missing from the random selection procedure of the parameter vectors in classical DE. The new knowledge that we gained in selection of parameter vector is given below as new strategy.

Strategy of parameter Vector Selection: Select any random parameter vector $\vec{a}$, and two other parameter vector $\vec{b'}$ and $\vec{a'}$such that $b_i > b_i'$ and $a_i > a_i'$ for all i, to evaluate a new donor vector $\vec{b}$. Since the square bracketed term in (8) can be either positive or negative, we keep our option open for the sign of F. However, in the traditional DE, F is always positive. To keep similarity with classical DE, we consider the sign of F in the present context explicitly, and thus have two alternative forms of (2). They are: $a + (b' - a')F$ and $a - (b' - a')F$.

In the selection step of extended-DE, we thus would identify the best fit vector from the list of following vectors:

i) a,
ii) $a + (b' - a')F$,
iii) $a - (b' - a')F$.

V. The Extended DE

Classical DE algorithms select parameter vectors randomly to obtain the donor vectors. They (donor vectors) thus cannot be directly used as trial solution to the optimization problem. Recombination is hence useful to generate potential trial solutions. However, in our method, which we name *Mean Value Based Differential Evolution with Adaptive Scaling Factor (MVMAFDE)*, as scaling factor is varied according to the information obtained from previous generations, mutation alone is sufficient for our scheme to generate trial solutions. Experimentally, we note that recombination cannot always improve the quality of solution further. So, the whole operation of recombination can be done away with, giving less complexity and computational cost. In order to establish the significance of MVMAFDE, we have also performed statistical tests.

Proposed scheme:

The proposed scheme includes three fundamental changes with respect to classical DE. First, the mutation step employed here is slightly different. Second, an adaptive algorithm is used to adapt *F* over the iterations. Third, the recombination step of classical DE is dropped.

A. *The Proposed Mutation*

There is a basic difference between the mutation strategies of classical DE and that of MVMAFDE. In classical DE, vectors selected for mutation are picked up randomly from the population only. But in our proposed algorithm, vectors selected for mutation is considered from the whole search landscape.

TABLE I: BENCHMARK FUNCTIONS USED.

Function names	Function expressions	Position of minima	Search Range
Sphere function	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$x_i = 0$	[-100, 100]
Ackley function	$f_2(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)$	$x_i = 0$	[-32, 32]
Rosenbrock function	$f_3(x) = \sum_{i=1}^{D-1}[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$	$x_i = 1$	[-100, 100]
Griewank function	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$x_i = 0$	[-100, 100]
Quadric function with noise	$f_5(x) = \sum_{i=1}^{D} x_i^4 + rand(0,1)$	$x_i = 0$	[-1.28, 1.28]
De Jong's f_4 with noise	$f_6(x) = \sum_{i=1}^{D} i.x_i^4 + rand(0,1)$	$x_i = 0$	[-1.28, 1.28]
Baele function	$f_7(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$	$x_1 = 3$ $x_2 = 0.5$	[-10,10]

B. Adaptive Selection of F

In order to improve the performance of DE, we propose an adaptive selection of *F*. We initialize a population x of *NP* vectors $P_t = \{\overrightarrow{X_1}(t), \overrightarrow{X_2}(t), \ldots, \overrightarrow{X_{NP}}(t)\}$ with $\overrightarrow{X_i}(t) = \{x_{i,1}(t), x_{i,2}(t), \ldots, x_{i,D}(t)\}$ uniformly random over a search range and initialize a variable named F_{mean}. Within a range of ±10% of F_{mean}, we initialize a set of *NP* variables $[F_1, F_2, \ldots, F_{NP}]$, from which, we select the scaling factor F_k, for each $\overrightarrow{x_k}$.

For each vector $\overrightarrow{x_k}$, we search for $\overrightarrow{X_k^l}$, $\overrightarrow{X_k^m}$ such that $\overrightarrow{X_k^l} \prec \overrightarrow{X_k} \prec \overrightarrow{X_k^m}$, where $\vec{A} \prec \vec{B}$ represents that A_i is less than B_i, $\forall i$. For each such $\overrightarrow{X_k}$, we find two vectors $\overrightarrow{U_k^1}, \overrightarrow{U_k^2}$, given by (9) and (10). Based on the Elitist Principle, we select the best NP vectors based on fitness from P_t, U^1, U^2 to construct P_t for the next generation.

Next, we select D number of vectors from U^1, U^2, where D=number of dimension. We calculate F_{sum}= the sum of corresponding D numbers of scaling factors F_k . Dividing F_{sum} with the number of dimensions D, we obtain the modified value of F_{mean} which we then use as the new F_{mean}. If F_{mean} goes below a particular threshold value, we reinitialize F_{mean}. The algorithm for this new process is given below.

```
Begin
    Initialize NP=dimension×10
    Initialize population of NP vectors
    $\overrightarrow{P_t} = \{\overrightarrow{X_1}(t), \overrightarrow{X_2}(t), \ldots, \overrightarrow{X_{NP}}(t)\}$ uniformly in the search range.
    Initialize $F_{mean}$
    While (Termination condition is not satisfied) Do
        Initialize $[F_1, F_2, \ldots, F_k]$ randomly in the interval [$0.9\times F_{mean}$, $1.1\times F_{mean}$]
        For each $\overrightarrow{x_k}$
            Compute $\overrightarrow{X_k^l}$, $\overrightarrow{X_k^m}$ such that $\overrightarrow{X_k^l} \prec \overrightarrow{X_k} \prec \overrightarrow{X_k^m}$
        End For
        For each $\overrightarrow{x_k}$ compute
            $\overrightarrow{U_k^1} = \overrightarrow{X_k} + F_k \times (\overrightarrow{X_k^m} - \overrightarrow{X_k^l})$        (9)
        and $\overrightarrow{U_k^2} = \overrightarrow{X_k} - F_k \times (\overrightarrow{X_k^m} - \overrightarrow{X_k^l})$        (10)
        End For
        Select the best NP vectors based on fitness from $P_t$, $U^1$, $U^2$ to construct $P_t$ for the next generation.
        Select the best dimension number of vectors based on fitness from $U^1$, $U^2$ and calculate
        $F_{sum}$= the sum of corresponding $F_k$.
        Calculate $F_{mean} = F_{sum}/dimension$
        If $F_{mean} < F_{Threshold}$
            Reinitialize $F_{mean}$
        End
    End While
End
```

VI. EXPERIMENTS AND PERFORMANCE EVALUATION

To test our proposed scheme and compare it with DE/rand/1, we select 7 benchmark functions [17]. They are enlisted in Table I.

A. Comparison of the proposed scheme with DE/rand/1

To check the performance of MVMAFDE, we test the above mentioned benchmark functions on our new scheme, as well as on conventional DE/rand/1. The dimension D is set to 30 in this section (Except for f_7, for which, dimensionality is 2). The average values of fitness function for 50 runs in case of both the methods are calculated and the comparison of these values with change of generations in case of these functions is shown in Fig. 1(a)-(g). The broken lines denote the plots for DE/rand/1 and the continuous ones denote the plots for benchmark suites, when tested with MVMAFDE.

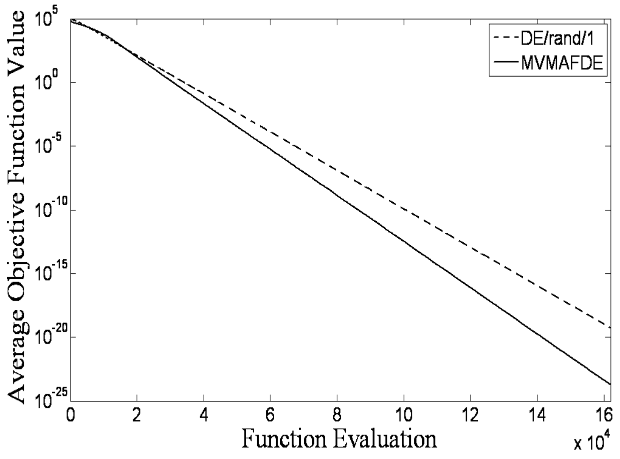

Fig. 1.a: Mean convergence graph on Sphere function.

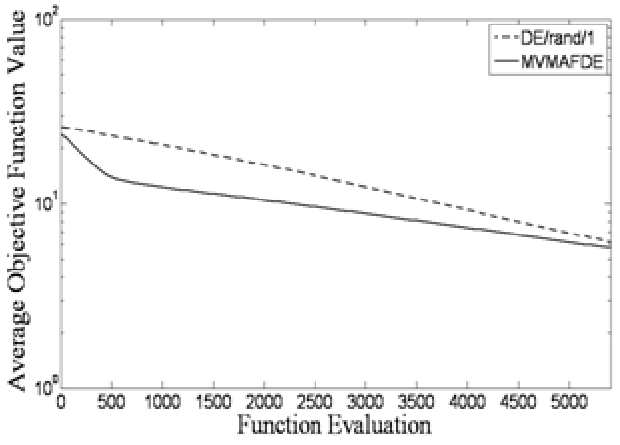

Fig. 1.d: Mean convergence graph on Griewank's function.

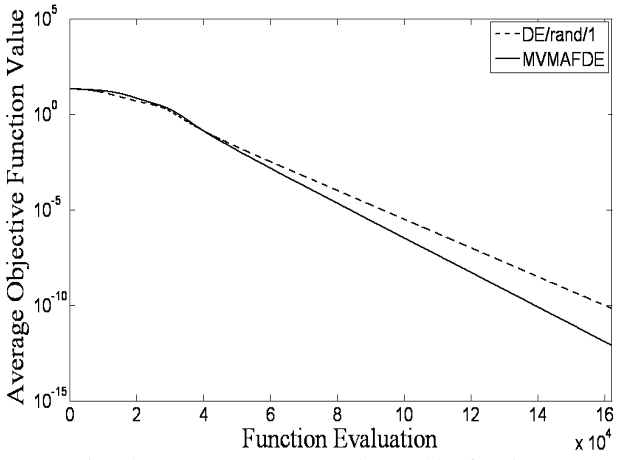

Fig. 1.b: Mean convergence graph on Ackley function.

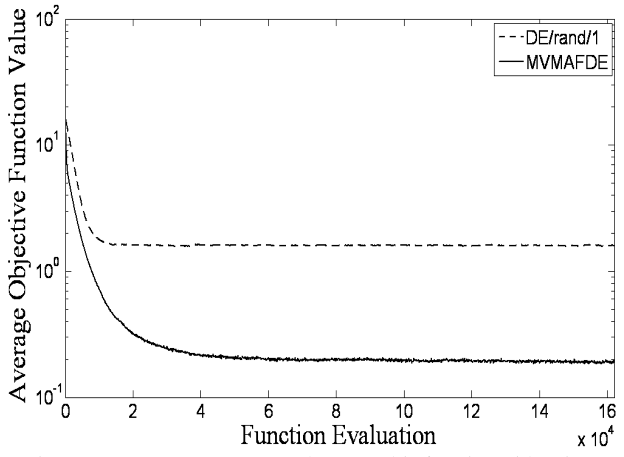

Fig. 1.e: Mean convergence graph on Quadric function with noise.

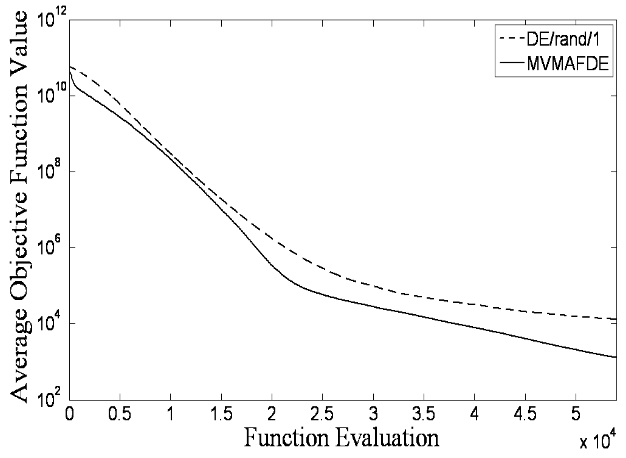

Fig. 1.c: Mean convergence graph on Rosenbrock function.

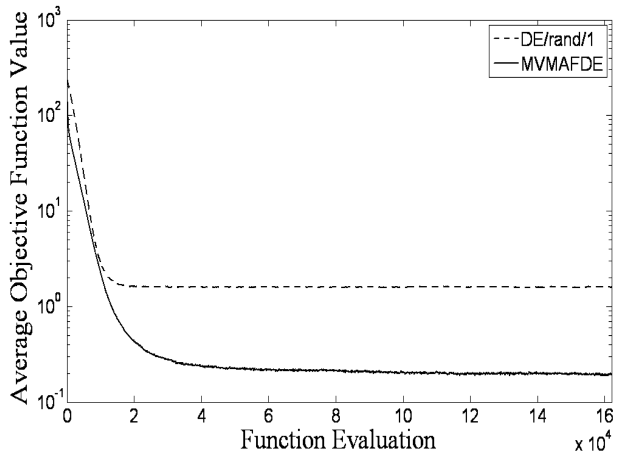

Fig. 1.f: Mean convergence graph on De Jong's f_4 with noise.

TABLE II: RESULTS OF UNPAIRED T-TEST

Function	t	Standard Error	95% Confidence interval	Two-tailed P	Significance
f_1	29.2325	1.3819×10-14	(118.9×10-15, 136.6×10-15)	1.1807×10-27	Extremely Significant
f_2	46.9072	6.7460×10-9	(95.7×10-9, 104.4×10-9)	3.0215×10-35	Extremely Significant
f_3	12.8652	2660.5	(6469.2, 8837.9)	5.6644×10-21	Extremely Significant
f_4	17.1909	3.2346×10^{-12}	$(1.897\times10^{-11}, 2.41\times10^{-11})$	2.0617×10^{-16}	Extremely Significant
f_5	174.7163	0.0255	(1.3914, 1.4240)	8.0613×10-57	Extremely Significant
f_6	184.2270	0.0241	(1.3908, 1.4217)	1.0781×10-57	Extremely Significant

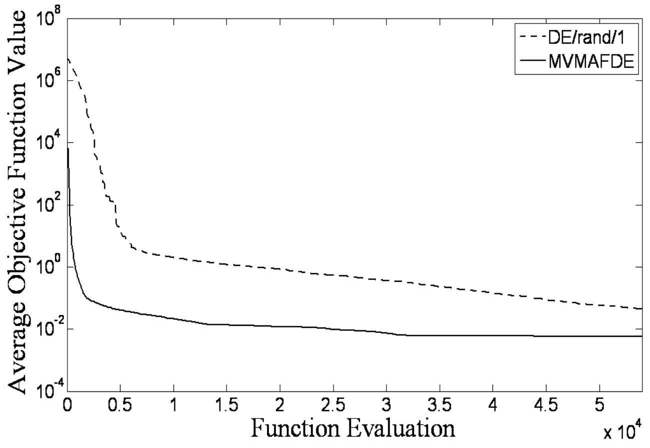

Fig. 1.g: Mean convergence graph of Beale function.

B. *Unpaired t-test*

Table II shows the results of unpaired t-test run between DE/rand/1 and MVMAFDE on the benchmark functions given in Table I. The table shows the values of standard error of difference of the two means, 95% confidence interval of this difference, the t value, and the two-tailed P values are presented. For all the cases, sample size=50 and degrees of freedom=98.

The t statistic can be calculated as follows:

$$t = (\overline{X_1} - \overline{X_2})/S_{\overline{X_1}-\overline{X_2}} \qquad (11)$$

where

$$S_{\overline{X_1}-\overline{X_2}} = \sqrt{\left(\frac{S_1^2}{n_1}\right) + \left(\frac{S_2^2}{n_2}\right)} \qquad (12)$$

S_1^2 and S_2^2 are the unbiased estimators of the variance of the two samples, n_1 and n_2 are the numbers of participants of group 1 and group 2 respectively. In our case, $n_1 = n_2 = n$, say. For use in significance testing, the distribution of the test statistic is approximated as being an ordinary Student's t distribution with the degrees of freedom calculated using

$$DF = (n-1)\left[\left(\frac{S_1^2}{n}\right) + \left(\frac{S_2^2}{n}\right)\right]^2 / \left[\left(\frac{S_1^2}{n}\right)^2 + \left(\frac{S_2^2}{n}\right)^2\right] \qquad (13)$$

The last column signifies whether the null hypothesis that the means of the two data are equal is accepted or rejected. "Extremely Significant" signifies that the hypothesis is false and the two data differ significantly. It can also be concluded that the proposed method beats the nearest competitor in a statistically meaningful way.

VII. EXPERIMENTS TO HANDLE NOISY OPTIMIZATION PROBLEM

To study the robustness of our proposed scheme of MVMAFDE to noise, we perform similar operations as described before, but this time with noisy benchmark functions. The noisy versions of the benchmark functions are defined as:

$$f_{noisy}(\vec{x}) = f(\vec{x}) + N(0, \sigma^2) \qquad (14)$$

where $N(0,\sigma^2)$ = Normal distribution with mean = 0 and variance=σ^2. We generate N with the help of the function "normrnd" of MATLAB, version 7.7 for various values of σ.

We have experimented by increasing noise standard deviation values from 0 to 1, in steps of 0.2, up to 2000 generations. Table III presents the mean and standard deviation of the best fitness values for the 6 benchmark functions (to) for noise standard deviations of 0, 0.2, 0.4, 0.6, 0.8 and 1.0. All the readings given are the averages over 50 independent runs per function per noise variance value. The better result among the two methods for any standard deviation for any function is shown in **bold**. The values of standard deviation are shown in parentheses '( )'.

TABLE III: MEAN AND STANDARD DEVIATION OF THE FITNESS AFTER 2000 GENERATIONS. (MEAN OF 50 RUNS FOR EACH DIFFERENT VALUE OF NOISE VARIANCE)

	Value of Standard Deviation (σ)											
	0		0.2		0.4		0.6		0.8		1.0	
	DE /rand/1	New Method	DE /rand/1	New Method	DE /rand/1	New Method	DE /rand/1	New Method	DE /rand/1	New Method	DE /rand/1	New Method
f_1	1.2×10^{-13} (1.9×10^{-14})	**7.1×10^{-28} (3.8×10^{-28})**	1.386 (0.0250)	**-0.1632 (0.0141)**	2.7688 (0.0503)	**-0.3111 (0.0500)**	4.1825 (0.0839)	**-0.476 (0.0444)**	5.5536 (0.1169)	**-0.65 (0.0661)**	6.9645 (0.1409)	**-0.7736 (0.0621)**
f_2	10^{-7} (9.3×10^{-9})	**2.32×10^{-14} (1.4×10^{-14})**	21.5790 (0.0170)	**-0.1097 (0.0176)**	21.651 (0.0233)	**-0.2232 (0.0416)**	21.6733 (0.0353)	**-0.3226 (0.0673)**	21.69 (0.0504)	**-0.4546 (0.0894)**	21.6936 (0.0641)	**-0.5086 (0.1210)**
f_3	9.1296×10^{3} (2.61×10^{3})	**1.476×10^{3} (2.64×10^{3})**	9.877×10^{3} (2.19×10^{3})	**1.538×10^{3} (2.93×10^{3})**	9.403×10^{3} (2.87×10^{3})	**1.205×10^{3} (2.57×10^{3})**	1.066×10^{4} (2.4×10^{3})	**798.8409 (1.81×10^{3})**	9.604×10^{3} (3.15×10^{3})	**847.4190 (2.06×10^{3})**	9.224×10^{3} (2.99×10^{3})	**1.237×10^{3} (2.36×10^{3})**
f_4	**8.38×10^{-14} (4×10^{-14})**	2.16×10^{-11} (5.7×10^{-12})	2.3828 (0.0214)	**0.3778 (0.4642)**	3.7737 (0.0576)	**0.6773 (0.0334)**	5.1756 (0.0865)	**0.5014 (0.0466)**	6.5425 (0.0979)	**0.3193 (0.0913)**	7.9161 (0.1310)	**0.1507 (0.0877)**
f_5	1.5980 (0.0323)	**0.1903 (0.0138)**	1.8203 (0.0376)	**0.1390 (0.0122)**	2.3170 (0.0640)	**0.0033 (0.0249)**	2.9277 (0.0686)	**-0.1758 (0.0330)**	3.6446 (0.0817)	**-0.3762 (0.0437)**	4.2833 (0.1071)	**-0.5497 (0.0493)**
f_6	1.6017 (0.0302)	**0.1955 (0.0140)**	1.8203 (0.0521)	**0.1512 (0.0214)**	2.3193 (0.0663)	**0.0154 (0.0316)**	2.9430 (0.0664)	**-0.1463 (0.0340)**	3.6477 (0.1248)	**-0.3400 (0.0510)**	4.3563 (0.1221)	**-0.5280 (0.0523)**

VIII. CURSE OF DIMENSIONALITY

The prime objective of this section is to study the curse of dimensionality. The experiment was undertaken for DE/rand/1 and MVMAFDE and the relative comparison in performance is estimated by number of function evaluations over increasing dimension of search landscape. Figs. 2.a, 2.b provide comparative estimates of the results obtained by the two algorithms to reach a desired error-limit in fitness value. Our proposed algorithm gives promising results for many benchmark functions, including Sphere function (f_1), Griewank function (f_4), and De Jong's f_4 (f_6).

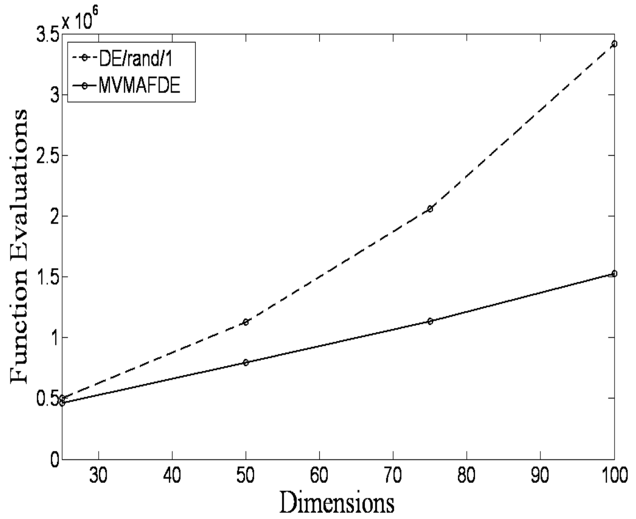

Fig. 2.a: Variation of mean number of Generation required for convergence with increase in dimensionality of the search space for Sphere function.

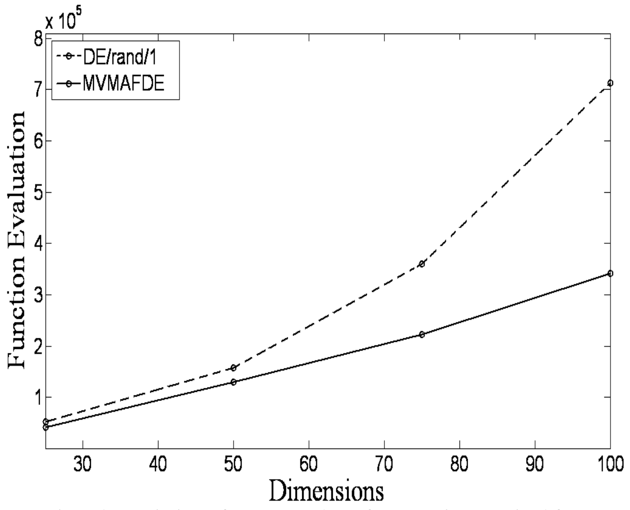

Fig. 2.b: Variation of mean number of Generation required for convergence with increase in dimensionality of the search space for De Jong's f_4.

IX. CONCLUSIONS

The paper extends classical DE by the following counts. First, it proposes a modified mutation scheme, which enhances the performance of the algorithm both in computational time and accuracy. Second, it introduces an adaptive selection of *F* over iterations, which ensures better trial solutions over the iterations. Indirectly, such adaptation in *F* helps early solution. The most important merit of the proposed algorithm of MVMAFDE is that it saves significant computation time due to dropping off of the recombination step. Experiments also confirm that MVMAFDE yields quality solution with statistically significant results in comparison to classical DE. The limitations of DE when applied to noisy fitness functions were pointed out in [8]. We are not aware of any published paper that solved this problem. As a remedy to this problem, we modified the basic algorithm with an adaptively changing scaling factor. Empirical evidence has been provided to justify the usefulness of the proposed approach to handle noisy fitness functions. Our proposed algorithm has also been shown to improve upon the scalability of DE with respect to the classical DE.

REFERENCES

[1] D. Ashlock, Evolutionary Computation for Modeling and Optimization, Springer, 2006.

[2] T. Back, D. B. Fogel, and Z. Michalewicz, Handbook of Evolutionary Computation, IOP and Oxford University Press, Bristol, UK, 1997.

[3] J. Brest, S. Greiner, B. Boˇskoi´c, M. Mernik, and V. ˇ Zumer, “Selfadaptive control parameters in differential evolution: A comparative study on numerical benchmark problems,” in IEEE Trans. Evol. Comput., 2006.

[4] Das, S., Konar, A. and Chakraborty, U. K., “Two improved differential evolution schemes for faster global search”, ACMSIGEVO Proceedings of GECCO, Washington D.C., 2005, pp. 991- 998.

[5] A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing, Springer, 2003.

[6] H.-Y. Fan and J. Lampinen, “A trigonometric mutation operation to differential evolution”, International Journal of Global Optimization, 27(1), 105-129, 2003.

[7] R. Gamperle, S. D. Muller, and A. Koumoutsakos, “Parameter study for differential evolution”, In WSEAS NNA-FSFS-EC 2002, Interlaken, Switzerland, Feb. 11-15 (2002).

[8] Krink, T, Bodgan, F, Fogel, G.B., Thomson, R.: Noisy Optimization Problems – “A Particular Challenge for Differential Evolution”, Proc. Sixth Congress on Evolutionary Computation (CEC-2004), IEEE Press.

[9] Ya-Liang Li, Fei Ding, Yu-Xuan, “Wang Iterated function system based adaptive Differential Evolution algorithm”, IEEE Congress on Evolutionary Computation, CEC 2008, pp. 1290-1294

[10] J. Liu and J. Lampinen, “A fuzzy adaptive differential evolution algorithm,” in Soft Computing - A Fusion of Foundations, Methodologies and Applications, 2005, pp. 448–462.

[11] K. Price, R. Storn, and J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag, ISBN: 3-540-20950-6, 2005.

[12] K. Price, and Storn, R., “Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces”, Technical Report, International Computer Science Institute, Berkley, 1995.

[13] A. K. Qin and P. N. Suganthan, “Self-adaptive differential evolution algorithm for numerical optimization,” Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791, 2005.

[14] 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, http://http.icsi.berkeley.edu/~storn/litera.html, 1995.

[15] R. Storn and K. Price, “Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces”, Journal of Global Optimization, 11(4) 341–359, 1997.

[16] Yang, Z., Tang, K. and Yao, X., “Self-adaptive Differential Evolution with Neighborhood Search”, In Proc. IEEE Congress on Evolutionary Computation, Hong Kong, 2008, pp. 1110-1116.

[17] X.Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster.” in IEEE Trans. Evol. Comput., 1999.