Differential Evolution with Dither and Annealed Scale Factor

Deepak Dawar and Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, ND, USA
deepak.dawar@ndsu.edu, simone.ludwig@ndsu.edu

***Abstract*—Differential Evolution (DE) is a highly competitive and powerful real parameter optimizer in the diverse community of evolutionary algorithms. The performance of DE depends largely upon its control parameters and is quite sensitive to their appropriate settings. One of those parameters commonly known as scale factor or F, controls the step size of the vector differentials during the search. During the exploration stage of the search, large step sizes may prove more conducive while during the exploitation stage, smaller step sizes might become favorable. This work proposes a simple and effective technique that alters F in stages, first through random perturbations and then through the application of an annealing schedule. We report the performance of the new variant on 20 benchmark functions of varying complexity and compare it with the classic DE algorithm (DE/Rand/1/bin), two other scale factor altering variants, and state of the art, SaDE.**

I. INTRODUCTION

Differential Evolution (DE) [1], proposed by Storn and Price in 1995, has emerged as a robust real parameter optimizer in the field of evolutionary computation. The power and popularity of DE can be gauged from the heightened research activity on the subject in the past decade. Numerous studies have been conducted to ascertain DE's efficacy on a broad range of problems ranging from benchmark to real life scientific and engineering problems [2]. After a year of its introduction, the power of DE was on display at the First International Contest on Evolutionary Optimization in May 1996 [3], where it secured first place among evolutionary algorithms. Since then, DE and its variants have performed exceedingly well in the optimization contests such as IEEE Congress on Evolutionary Computation and alike.

DE is simple and employs few control parameters namely scale factor (F), crossover rate (Cr), and population size (NP). The performance of DE is very sensitive to the proper settings of these parameters [4], [5], [6]. An unfavorable combination of these parameters can seriously degrade the algorithm's efficacy. At the same time, choosing effective control parameter settings can be quite cumbersome. A good combination of these parameters depends upon the problem at hand and requires a good amount of experience of the user. The bottom-line is, "better and more informed values of control parameters yield better results for a given problem." As crucial as the role of good control parameters may sound to be, there seems to be no accepted methodology to determine their universally effective values. Since it may be difficult and time consuming to generalize a set of control parameter values, there is always a motivation to alter or adapt them during the course of the algorithm as defined by certain rules. Intensive research activity has been reported in the area of finding good values of control parameters. In [4] and [7], authors grouped this change into three broad classes:

- **Deterministic -** the parameters are altered based on some user defined rules [1], [8].
- **Adaptive -** the parameters are allowed to adapt based on some feedback from the algorithm [9].
- **Self-Adaptive -** the parameters are encoded into the solution itself and they evolve as a part of the general population [10], [11].

In this paper, we primarily focus our attention towards deterministic parameter control methods and the control parameter, scale factor (F), in particular.

Dither is a deterministic parameter control technique wherein F is allowed to take on random values between a specific range represented by F_{low} and F_{high}. We propose combining dither with an annealing based cooling schedule to alter F and introduce a two stage technique, DEDASF, based on this concept. In the first stage, dither is applied and in the second stage, F is allowed to be reduced by a randomized factor. Every stage runs for a fixed number of function evaluations, a count of which has to be set beforehand. The idea is to scatter the population to diverse and favorable areas first and then reduce the step size to take advantage of the notion that during early stages of exploration of the search space by DE, large step sizes may prove beneficial for investigating the maximum area of the problem landscape, and when the exploitation stage kicks in, small step sizes may become more advantageous. Though it is possible to alternate between exploration-exploitation during the course of the search, which may be beneficial for many landscapes, we have limited our investigation to a single stage exploration-exploitation model.

The rest of this paper is structured as follows. Section II describes the basic DE algorithm. In Section III, related work is presented. Section IV describes and explains the new algorithm with all its features. In Section V, results and their analysis are presented, and Section VI concludes the paper.

II. Differential Evolution

Essential steps of DE are explained in Sections II-A to II-D.

A. Initialization

A NP number of D dimensional solutions (vectors) are initialized randomly to form a generation G. The vectors are changed over generations G_n, where $n = 1, 2, \ldots, G_{max}$. The i^{th} vector ($1 \leq i \leq NP$), X^i, in the current generation G can be represented as:

$$X_G^i = [x_{1,G}^i, x_{2,G}^i, x_{3,G}^i, \ldots.., x_{D,G}^i] \tag{1}$$

Every parameter x_j^i in a given vector has a specific range within which it has to be restricted.

B. Mutation

In the most basic arrangement of DE, for each i^{th} target vector from the current generation, three other distinct vectors, say X^{r1}, X^{r2} and X^{r3} are selected randomly. The indices r_1, r_2 and r_3 are mutually exclusive integers randomly chosen from the range [1,NP], which are also different from the target vector index i. The donor is then created as:

$$D_G^i = X_G^{r1} + F \times (X_G^{r2} - X_G^{r3}) \tag{2}$$

where F is the scale factor. Equation 2 represents the classic DE mutation strategy *DE/rand/1* [12] where *rand* means that parents are selected randomly, and 1 signifies the presence of only one differential perturbation. There are other strategies suggested by authors in [1], [12], but *DE/rand/1* is the most widely used [5], [6].

C. Crossover

In this operation, parameters either from the target vector, X_G, or the donor vector, D_G, are selected based on some probability distribution to form a trial (child) vector, T_G. This operation can be elucidated as:

$$T_{j,G}^i = \begin{cases} D_{j,G}^i & \text{if } rand_j^i[0,1] \leq Cr \text{ or } j = j_{rand} \\ X_{j,G}^i & \text{otherwise} \end{cases} \tag{3}$$

where T_G^i is the trial vector generated for the i_{th} target vector, X_G^i, for generation G, $rand_j^i$ [0,1] is a randomly generated real number ranging between the interval [0,1], and is generated afresh for every j^{th} parameter of the target vector. The number j_{rand}, is a randomly chosen index between [1,D] to ensure that the $trial$ vector gets at least one element from the $donor$ vector.

D. Selection

If the fitness (considering a minimization problem) of the trial vector is smaller or at least equal to the target vector, it moves to the next generation. Otherwise the target vector is promoted.

$$X_{G+1}^i = \begin{cases} T_G^i & \text{if } f(T_G^i) \leq f(X_G^i) \\ X_G^i & \text{otherwise.} \end{cases} \tag{4}$$

III. Related Work

There have been many attempts to improve the performance of DE by varying the scale factor during the search process and multiple methods have been suggested to achieve this goal. One of them is Dither [8]. It is a deterministic scheme of randomization of scale factor, F. Many different ways of randomizing F are possible. For example in [13], F was randomized on generational basis as:

$$F_{dither} = F_l + rand_G(0,1) \times (F_h - F_l) \tag{5}$$

where F_l and F_h are the predefined lowest and highest values of F, respectively, and $rand_G(0,1)$ is a uniformly distributed real number generated anew for every generation G. In [9], convergence of DE was reported to have been improved while using dither, though the authors applied dither on an individual basis rather than generational basis. Therefore, preliminary results make a good case for using this technique.

There is a similar technique proposed in [8] called *Jitter* wherein F is randomized for every difference of the parameters involved in the differential operation. The operation can be represented as:

$$F_{jitter} = F \times (1 + \gamma \times (rand_j[0,1] - 0.5)) \tag{6}$$

where it is imperative that γ be small. In [14], the author mixed both *dither* and *jitter*.

Apart from randomization schemes, another technique that appears to be useful while negotiating the search space is step-size reduction. Step size may be considered as the distance between position of the current vector and the newly generated vector, in a D dimensional space. In DE, the step size is controlled by F.

In [15], authors describe a step size reduction technique for their one point direct search algorithm. The algorithm starts with a point x_0 in a D dimensional space. The nearby space is explored and a new point x_1 is selected for evaluation. If x_1 is found to be worse than x_0, then the step size is supposed to have been large and is reduced by a certain factor. An obvious drawback of this scheme is that the technique only contracts the step size and never expands it thereby increasing the chances of the solution getting stuck in a local optimum.

In [9], authors proposed a step size reduction scheme, DETVSF, wherein they reduced the step size with every generation. Mathematically, this scheme is described as:

$$F_{curr} = (F_{max} - F_{min}) \times (G_{max} - G_{curr})/G_{max} \tag{7}$$

where F_{curr} is the current value of the scale factor, F_{max} and F_{min} are the predefined maximum and minimum values of the scale factor, respectively. G_{curr} and G_{max} are the current and maximum generation number, respectively. This technique was reported to have improved the performance of DE in a statistically meaningful way [9].

The step size reduction by a constant factor may well be juxtaposed with the concept of the metallurgical technique of annealing, which is the process of treating a metal by first heating it above its critical temperature and then cooling it at a certain rate.

The work in this paper is motivated by the encouraging results reported in [9] and [13], which clearly insinuate the use of dither and step size reduction techniques. Though the individual results of these schemes are promising, a closer look might reveal some shortcomings of the individual use of these techniques. Dither offers randomized step sizes throughout the search process. During the initial phase of the search, this may prove useful but during the later phase, when focus shifts to a particular area of landscape, arbitrary and occasional large step sizes may prove detrimental to convergence. Thus, dither may be avoided during the later part of the search. During the exploration stage of the search, large step sizes are advantageous, while during the exploitation stage, small values prove more effective. While step size reduction techniques make a lot of sense, solely contracting step size run the risk of getting stuck in local minimum if there is not enough diversity in the population.

With these points in mind we decided to hybridize these techniques to take advantage of their individual strengths. Our hybrid is stage based. We apply dither in the first and step size reduction in the second stage. This sequence of stages proves more conducive and effective, as we shall present in the results, than employing the dither or step size reduction technique alone.

IV. DEDASF Algorithm

We propose DEDASF, Differential Evolution with Dither and Annealed Scale Factor, an algorithm that applies dither and annealing to the scale factor, F, one after the other. DEDASF is summarized in Algorithm 1.

The algorithm makes changes to F in two stages. The duration of the first stage is a pre-specified number of generations or function evaluations (FEs). In the first stage, dither is applied to F within a pre-specified range. For our experiments we chose the range [0.1,0.9] to promote a multitude of step sizes that would help sample different and wide areas of the search landscape. Also, loss of diversity is a known problem in DE and dither may help improve it as argued in [16].

After the first stage, when the population has scattered sufficiently to seemingly favorable areas of the landscape, F is allowed to cool slowly and the cooling rate is controlled by the randomized factor α. During the exploration stage a high value of F is advantageous, while during the exploitation stage small values of F are desirable. Thus, while in general it is difficult to specify the step size for different stages of the search, this scheme may help the search make a smooth transition from the exploration to the exploitation stage. The two stages of this scheme can be outlined as:

$$F_c = \begin{cases} F_l + rand(0,1] \times (F_h - F_l) & \text{if } G_c < G_d \\ \alpha^{G_c - G_d} \times F_0 & \text{otherwise} \end{cases} \tag{8}$$

Algorithm 1 Pseudo-code for DEDASF

```
1)  Set values of $NP$, $Cr$
2)  Set Dither range $F_l$, $F_h$. Set no. of generations, $G_d$, for which dither would be applied
3)  Set Annealing constants $F_0$, $\alpha_l$, $\alpha_h$
4)  Initialize a population of $NP$ individuals $P = [X^1, X^2, ...X^{NP}]$ where every $i^{th}$ individual is a $D$ dimensional vector represented as $X^i_j$=[$x^i_1$, $x^i_2$ ... $x^i_D$].
5)  **While** stopping criteria is not met **do**
6)      **For** every target vector $X^{target}$ in $P$ **do**
7)          **Select** three vectors $X^{r_1}, X^{r_2}, X^{r_3}$ where $r_1, r_2$, and $r_3$ are three mutually exclusive indices and different from the index of target vector
8)          **Produce** a donor vector through mutation as
            $X^{donor} = X^{r_1} + F \times (X^{r_2} - X^{r_3})$ where $F$ is calculated as:
            $F = \begin{cases} F_l + rand(0,1] \times (F_h - F_l) & \text{if } G_c \le G_d \\ \alpha^{G_c - G_d} \times F_0 & \text{otherwise} \end{cases}$
9)          **Produce** a trial vector, $X^{trial} = (x^{trial}_1, \dots, x^{trial}_D)$, through crossover as:
            $x^{trial}_j = \begin{cases} x^{donor}_j & \text{if } rand_j(0,1] \le Cr \text{ or } j = j_{rand} \\ x^{target}_j & \text{otherwise.} \end{cases}$
10)         **Select** either the target vector or the trial vector based on their fitness values as:
            $X^{survivor}_{G+1} = \begin{cases} X^{trial}_G & \text{if } F(X^{trial}_G) \le F(X^{target}_G) \\ X^{target}_G & \text{otherwise.} \end{cases}$
11)     end **For**
12) end **While**
```

where F_c is the current value of scale factor, F_l and F_h, the predefined lowest and highest values of scale factor. G_c is the current generation, and G_d is the number of generations allotted to the dither stage.

F_0 is the value at which the reduction of scale factor starts or critical temperature in metallurgical terms. After an empirical study, we fixed F_0 at 0.7 as this value should not be too high or too low for the search would have progressed towards favorable regions by the time the annealing stage kicks in. Keeping F_0 high would slow down the convergence and a low value might result in a failure to explore prospective areas. In Equation 8, α represents the cooling rate, a value which is randomized between α_l and α_h. After an empirical study, a part of which is explained in the Results section, the values of α_l and α_h were fixed at 0.995 and 0.998, respectively, and α calculated as:

$$\alpha = rand(0,1] \times (\alpha_h - \alpha_l) \tag{9}$$

where $rand(0,1]$ is a uniformly distributed random number between 0 and 1.

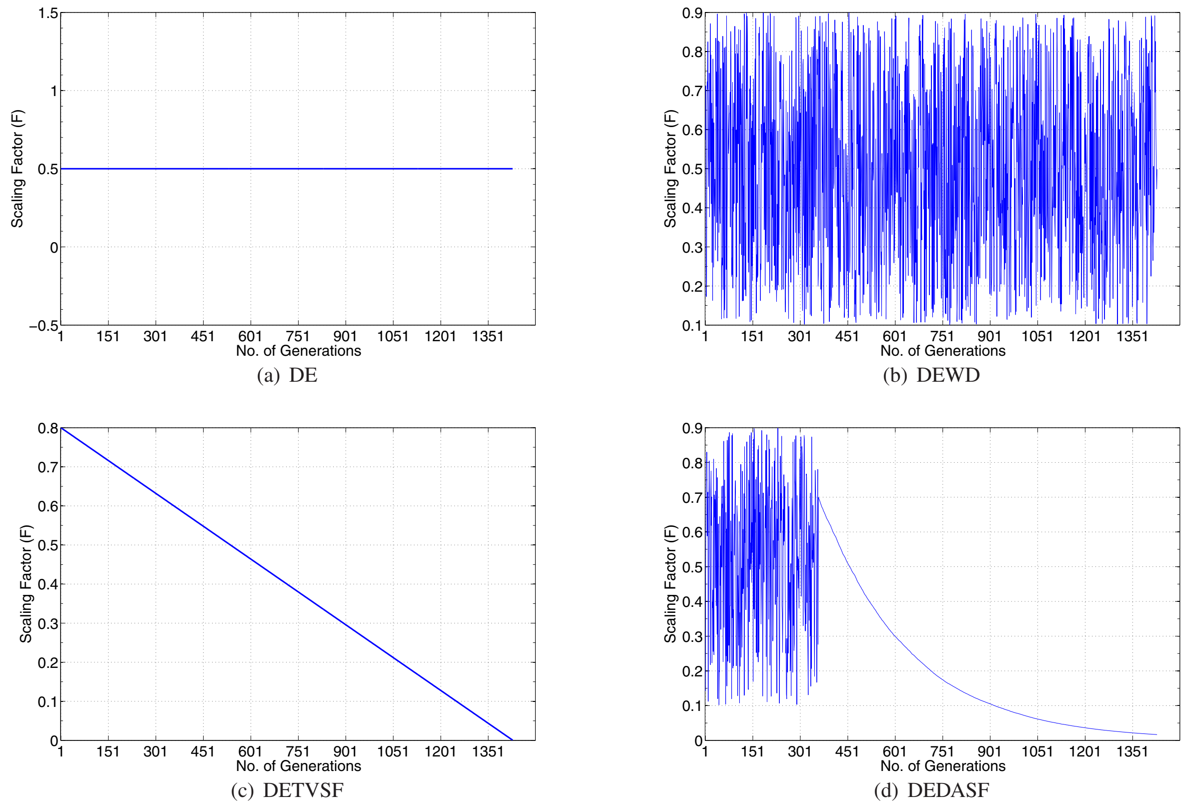


Fig. 1: Variation in scale factor with number of generations executed for DE, DEWD, DETVSF, and DEDASF.

V. Experimentation And Results

A. Benchmark Functions

We performed our experiments on 20 benchmark functions of varying properties and geometric orientations. The first five functions (f1 to f5) are unimodal, the next fifteen (f6 to f20) are multimodal. Due to paucity of space in this paper, we are unable to provide the details about the test functions but a detailed insight into the functions can be found in [17].

B. DEDASF vs other methods

We perform algorithm comparison in two ways. First we compare DEDASF with other deterministic parameter control techniques to determine its rank. We perform a comparative performance test of DEDASF with: (a) Classic DE, (b) DETVSF, as proposed in [9], and (c) DEWD, classic DE with dither as proposed in [8]. DETVSF and DEWD were described in Section III. Three of the compared algorithms namely DEDASF, DETVSF, and DEWD alter F as the search progresses, while in DE, F is kept constant. For these compared algorithms, the variation in F with the number of generations is presented in Fig 1.

After the initial comparative evaluations, DEDASF is then compared with Self Adaptive Differential Evolution algorithm, SaDE [21], to contrast the performance of with our deterministic parameter control method with the adaptive one.

C. Control Parameter Set Up

A large body of research on control parameter settings is available, which while not being able to provide a panacea for the control parameter setting problem, provides suitable guidelines for their use.

Authors in [6] suggest that the population size NP be between 3D to 8D, where D is the dimensionality of the problem. Using their guidelines coupled with our own experience, we fixed NP to be seven times the problem dimensionality. Storn and Price in [8] suggest that Cr be either between [0.0,0.2] or [0.9,1]. The reason for such division is that separable functions are solved quite well at low values of Cr, and non-separable at high values. However, to maintain uniformity, we fixed Cr at 0.9. Another good reason for this choice is to increase the diversity and minimize the orthogonal movements of vectors. A high value of Cr is also recommended in [18].

DEDASF, DTVSF, DEWD alter the scale factor with their own mechanisms. The scale factor for classic DE is fixed at 0.5 as suggested in [18].

According to the guidelines laid down in [17], the maximum number of function evaluations (MaxFEs) has been restricted to 10^4 times the dimensionality of the problem, every benchmark function is evaluated 51 times, and evaluation is terminated once MaxFEs is reached or the difference between the global optimum and current best reaches a value of 10^{-8}.

TABLE I: Performance of DEDASF, DETVSF, DEWD, and DE at 10D, 30D, and 50D, respectively. Reported values are the averages of 51 independent runs for each function. Error values reaching within 10^{-8} of the global optimum of the function are reported as 0.00+E00.

10D				
Function	**DEDASF**	**DETVSF**	**DEWD**	**DE**
1	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
2	2.63E+03±6.38E+03	**1.31E-02**±5.73E-02	7.81E+02±1.97E+03	2.01E+01±9.04E+01
3	4.21E+00±1.38E+01	1.95E-01±2.10E-01	1.74E+00±6.15E+00	**2.25E-02**±6.32E-02
4	4.03E+01±9.05E+01	**2.62E-04**±9.37E-04	1.96E+00±5.94E+00	3.56E-02±1.70E-01
5	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
6	**0.00E+00**±3.65E-08	2.00E-01±1.70E-01	2.48E+00±0.00E+00	2.97E-04±1.06E-03
7	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
8	2.05E+01±5.54E-02	2.05E+01±5.09E-02	2.05E+01±6.00E-02	2.05E+01±5.13E-02
9	3.00E-02±6.14E-01	1.80E-01±3.29E-01	9.72E-02±3.22E-01	**1.92E-05**±9.05E-05
10	4.63E-02±2.06E-01	8.77E-02±6.31E-02	**4.35E-02**±3.27E-02	7.13E-02±4.88E-02
11	8.95E-01±5.51E-01	9.12E-01±1.16E+00	**7.06E-01**±7.66E-01	6.46E+00±5.48E+00
12	6.15E+00±3.95E+00	7.03E+00±2.84E+00	**5.52E+00**±2.66E+00	1.67E+01±8.40E+00
13	9.32E+00±2.09E-01	**8.66E+00**±3.74E+00	1.16E+01±5.34E+00	1.46E+01±8.26E+00
14	5.65E+01±3.41E-01	**1.75E+01**±1.10E+01	1.92E+02±1.63E+02	9.21E+02±2.69E+02
15	2.42E+02±3.09E-02	**1.59E+02**±9.27E+01	1.05E+03±3.05E+02	1.38E+03±1.08E+02
16	1.11E+00±7.47E-01	**1.04E+00**±1.67E-01	1.08E+00±2.14E-01	1.04E+00±1.80E-01
17	**1.15E+01**±2.64E+00	1.16E+01±7.99E-01	2.38E+01±5.03E+00	2.52E+01±3.27E+00
18	2.21E+01±4.44E+00	**1.90E+01**±2.03E+00	3.27E+01±6.32E+00	3.46E+01±4.80E+00
19	5.69E-01±5.98E+01	**5.38E-01**±7.21E-02	5.57E-01±1.25E-01	5.38E-01±1.69E-01
20	2.27E+00±2.13E+02	1.65E+00±4.57E-01	**1.07E+00**±5.52E-01	2.61E+00±2.00E-01

30D				
Function	**DEDASF**	**DETVSF**	**DEWD**	**DE**
1	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
2	1.09E+06±4.17E+05	1.53E+06±5.52E+05	**9.38E+05**±4.61E+05	3.91E+06±1.11E+06
3	**2.41E+06**±2.97E+06	2.49E+06±2.80E+06	4.80E+06±3.14E+06	7.90E+06±5.01E+06
4	9.76E+03±2.23E+03	1.44E+04±2.96E+03	**4.64E+03**±1.44E+03	3.03E+04±5.69E+03
5	2.87E-06±4.72E-06	2.32E-06±9.25E-06	**0.00E+00**±0.00E+00	1.01E-05±2.84E-06
6	9.48E+01±2.01E+01	**2.93E+00**±2.29E-01	1.40E+01±2.86E+01	8.17E+01±9.88E+00
7	**2.87E-02**±1.42E-01	5.33E-01±6.29E-01	3.79E-01±8.72E-01	3.08E+00±1.57E+00
8	2.10E+01±4.87E-02	2.10E+01±4.80E-02	2.10E+01±3.38E-02	2.10E+01±4.32E-02
9	3.79E+00±2.48E+00	6.39E+00±2.00E+00	**1.42E+00**±1.33E+00	3.75E+01±1.19E+00
10	7.66E-02±9.02E-02	2.10E-01±2.88E-01	**2.86E-02**±1.36E-02	8.07E-01±2.03E-01
11	5.69E+00±2.17E+00	**3.84E+00**±1.56E+00	6.16E+01±1.61E+01	1.82E+02±1.09E+01
12	3.14E+01±6.82E+00	**2.84E+01**±7.25E+00	1.38E+02±3.63E+01	1.98E+02±1.15E+01
13	**5.50E+01**±1.85E+01	5.57E+01±1.57E+01	1.56E+02±2.37E+01	1.97E+02±8.63E+00
14	**1.90E+02**±1.09E+02	6.80E+02±1.76E+02	6.56E+03±3.11E+02	7.08E+03±2.50E+02
15	**3.91E+03**±1.02E+03	5.84E+03±3.79E+02	7.10E+03±2.14E+02	7.33E+03±2.33E+02
16	**2.43E+00**±2.97E-01	2.54E+00±3.10E-01	2.47E+00±2.45E-01	2.46E+00±2.56E-01
17	**4.06E+01**±3.20E+00	4.89E+01±4.05E+00	1.86E+02±8.03E+00	2.16E+02±8.35E+00
18	**1.43E+02**±2.17E+01	1.82E+02±8.98E+00	2.01E+02±1.07E+01	2.27E+02±1.07E+01
19	**1.26E+00**±2.16E-01	1.86E+00±1.16E-01	2.01E+00±3.35E-01	1.90E+00±4.75E-01
20	**1.09E+01**±8.08E-01	1.20E+01±2.81E-01	1.18E+01±2.99E-01	1.33E+01±1.41E-01

50D				
Function	**DEDASF**	**DETVSF**	**DEWD**	**DE**
1	8.72E-07±2.38E-06	1.18E-06±3.87E-06	**0.00E+00**±0.00E+00	1.64E+00±3.90E-01
2	**7.55E+06**±2.06E+06	1.08E+07±3.04E+06	1.56E+07±3.29E+06	1.11E+08±1.86E+07
3	**1.31E+07**±2.71E+07	2.94E+07±2.79E+07	2.76E+07±3.11E+07	8.97E+07±2.74E+07
4	**5.00E+04**±7.41E+03	7.02E+04±7.01E+03	9.82E+04±9.46E+03	1.25E+05±8.85E+03
5	3.88E-03±4.51E-03	1.29E-03±2.48E-03	**2.41E-06**±1.56E-06	7.96E-01±1.34E-01
6	**1.20E+02**±1.11E+01	1.68E+02±2.60E+01	1.31E+02±3.17E+01	1.55E+02±1.37E+01
7	7.46E+00±1.95E+00	**4.40E+00**±1.93E+00	8.62E+00±3.42E+00	8.53E+01±1.00E+01
8	2.11E+01±3.84E-02	2.11E+01±3.06E-02	2.11E+01±2.78E-02	2.11E+01±4.67E-02
9	**1.33E+01**±4.77E+00	2.16E+01±4.03E+00	1.52E+01±1.42E+01	7.02E+01±1.68E+00
10	4.13E+00±1.21E+00	6.06E+00±2.08E+00	**1.03E+00**±1.54E-01	6.88E+01±1.31E+01
11	8.13E+00±2.69E+00	**8.10E+00**±2.73E+00	2.12E+02±2.39E+01	3.91E+02±1.55E+01
12	**6.80E+01**±1.32E+01	1.18E+02±4.22E+01	3.38E+02±1.25E+01	4.12E+02±1.91E+01
13	**1.33E+02**±2.68E+01	1.89E+02±4.96E+01	3.49E+02±1.56E+01	4.18E+02±1.38E+01
14	**4.74E+02**±2.17E+02	3.01E+03±4.77E+02	1.31E+04±4.01E+02	1.33E+04±2.71E+02
15	**1.27E+04**±6.77E+02	1.35E+04±4.29E+02	1.36E+04±4.03E+02	1.34E+04±4.13E+02
16	3.31E+00±2.58E-01	3.42E+00±2.65E-01	**3.26E+00**±3.92E-01	3.29E+00±2.10E-01
17	**8.20E+01**±9.57E+00	1.07E+02±8.20E+00	3.71E+02±1.49E+01	4.44E+02±1.58E+01
18	**3.62E+02**±1.29E+01	3.81E+02±1.13E+01	3.97E+02±1.49E+01	4.58E+02±1.48E+01
19	**2.56E+00**±3.72E-01	2.70E+00±3.52E-01	2.65E+00±4.55E-01	1.47E+01±1.89E+00
20	2.25E+01±9.87E-01	**2.16E+01**±8.75E-01	2.18E+01±2.51E-01	2.50E+01±2.14E-01

D. Results

Table I reports the performance of the deterministic parameter control algorithms namely DEDASF, DETVSF, DEWD, and the classic DE, at problem dimensionality 10, 30, and 50, respectively.

With a mere glance at the Table I, it can be inferred that none of the algorithms perform significantly better than its competitors at problem dimensionality 10 where DEDASF and DE score two wins each, DETVSF eight, and DEWD four wins. Table I also revels that at problem dimensionality 30, DEDASF wins 10 times, DETVSF 3 times, and DEWD wins 4 times while classic DE does not record any win. At 50D, DEDASF scores 12 wins, DETVSF 3, and DEWD 4, while DE again scores no wins. To decipher the statistical difference between the algorithms, we compare them with the Friedman test and later on with the Hochberg post-hoc procedure.

The Friedman test [19], is a multiple comparisons procedure that aims to detect significant performance differences between the compared algorithms. It calculates the relative ranks of the algorithms through an average ranking procedure and computes the Friedman statistic, which is further used to calculate the p value.

Table II presents the relative ranks, and Table III reports the Friedman statistic and p values obtained by the algorithms at problem dimensionality 10, 30, and 50, respectively.

TABLE II: Relative Ranks obtained by DEDASF, DETVSF, DEWD, and DE at 10D, 30D, and 50D.

Algorithm	Rank-10D	Rank-30D	Rank-50D
DEDASF	2.475	**1.625**	**1.6**
DETVSF	**2.075**	2.325	2.375
DEWD	2.575	2.325	2.35
DE	2.875	3.725	3.675

It is observed that the Friedman test did not detect a significant difference between the algorithms at problem dimensionality 10. DETVSF emerges as the best ranked algorithm, but it is not statistically significant when compared to other algorithms either at 0.05 or 0.1 level of significance.

At problem dimensionality 30 and 50, however, the Friedman test reports a significant difference between the algorithms. The difference is significant at the 0.05 level of significance, as clearly shown in Table III, and DEDASF clearly emerges as the best ranked algorithm.

TABLE III: Friedman statistic (distributed according to chi-square with 3 degrees of freedom) and p value computed by Friedman Test at 10D, 30D, and 50D.

Dimension	Friedman Statistic	p Value
10	3.93	0.269123
30	27.93	**0.000004**
50	26.745	**0.000007**

The Friedman test is capable of detecting significant differences between algorithms, but is unable to perform comparisons between some of the algorithms, for example, when a particular control algorithm is to be compared with other algorithms. To do this, a family of hypothesis must be defined and then a post-hoc analysis should be conducted to find a p value indicating rejection or acceptance of the family of hypothesis. To compute this p value, we performed a post-hoc analysis using the Hochberg procedure [20].

Table V and VI present the unadjusted and adjusted p values obtained by the Hochberg post-hoc procedure at problem dimensionality 30 and 50, respectively. The Hochberg post-hoc procedure was not applied to the Friedman test results obtained at 10D as there was not any significant difference reported between the compared algorithms. The Hochberg post-hoc test suggests that DEDASF is significant at the 0.1 level of significance for problem dimensionality 30 and 50.

It can be inferred from Figure 2 that DEDASF is not relatively effective at lower dimensions but its performance improves as the dimensionality of the problem increases, though more research needs to be conducted to firmly confirm this observation.

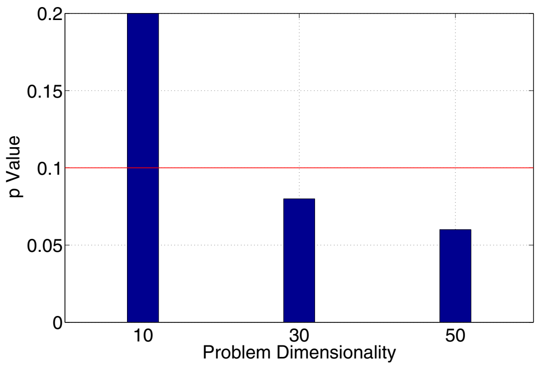
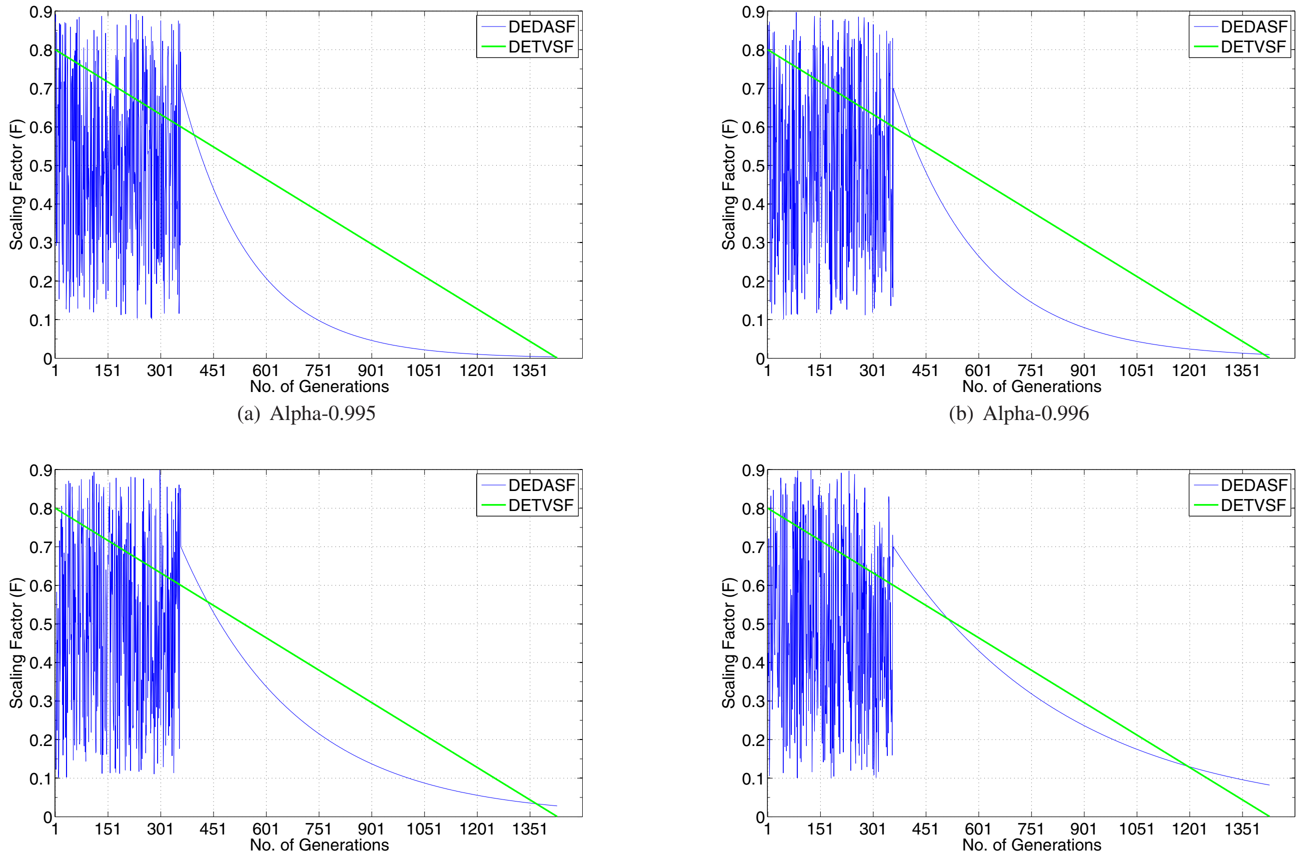


Fig. 2: Adjusted p values obtained by Hochberg procedure for DEDASF at problem dimentionality 10, 30, and 50. DEDASF is significant at the 0.1 level of significance at problem dimensionality 30 and 50.

Another aspect of the results that demands attention is the effect of the cooling rate of DEDASF on the search process. We found that some of the benchmark functions respond well to a slow cooling rate, while others lend themselves well to a quick rate of cooling. There are a few test functions that respond to cooling rates in an arbitrary manner. Hence, after extensive experimentation with several cooling rates, we concluded that a single cooling rate would not be suitable for all the test functions. Through this experimentation we found the approximate range between which the cooling rate is effective, which we denote as $\alpha_{low}(0.995)$ and $\alpha_{high}(0.998)$, the highest and lowest cooling rate, respectively. The higher the value of α, the lower the cooling rate.

A division of test functions based on their response to cooling rates is shown in Table IV, where Type I represents the functions that show good results with slow cooling rates (high α), Type II represents the functions that show good results with fast cooling rates (low α), and functions in Type III do not follow a specific pattern.



(a) Alpha-0.995

(b) Alpha-0.996

(c) Alpha-0.997

(d) Alpha-0.998

Fig. 3: Comparison of slope of DEDASF and DETVSF at different cooling rates.

TABLE IV: Division of benchmark functions based on their response to cooling rates.

Type I	Type II	Type III
2	11	1
3	12	8
4	13	16
5	14	19
6	15	-
7	17	-
9	18	-
10	20	-

Figure 3 shows the slope of DEDASF with different values of α as compared to DETVSF. Since a single cooling rate would not yield good results for all the test functions, we decided to randomize it between the range α_{high} and α_{low}. Randomization indeed proved useful and resulted in significant performance improvements at problem dimensionality 30 and 50 as is clear from the Tables V and VI.

TABLE V: Adjusted p-values-30D.

Algorithm	unadjusted p	$p_{Hochberg}$
DE	0	**0.000001**
DETVSF	0.086411	**0.086411**
DEWD	0.086411	**0.086411**

TABLE VI: Adjusted p-values-50D.

Algorithm	unadjusted p	$p_{Hochberg}$
DE	0	**0.000001**
DETVSF	0.057649	**0.066193**
DEWD	0.066193	**0.066193**

We also conducted a one-on-one comparison between the algorithms that reduce the scale factor during the course of execution, i.e., DEDASF and DETVSF since comparing multiple algorithms may run the risk of accumulating the Family Wise Error Rate (FWER), even when it is controlled. We used the well known Wilcoxon test for the comparison. The result is reported in Figure 4.

It is clear from Figure 4 that again there is no significant difference between DEDASF and DETVSF at problem dimensionality 10. But at problem dimensionality 30 and 50, DEDASF is statistically significant compared to DETVSF at the significance level 0.05. Also, the performance of DEDASF improves as the problem dimensionality increases, which was also the case during the comparison of multiple algorithms.

To contrast the performance of DEDASF with SaDE (results sourced from [22]), we first performed the sign test, and then the Wilcoxon test. The reason for choosing a double test measure is the fact that while sign test being very crude and insensitive, provides a general idea about the performance

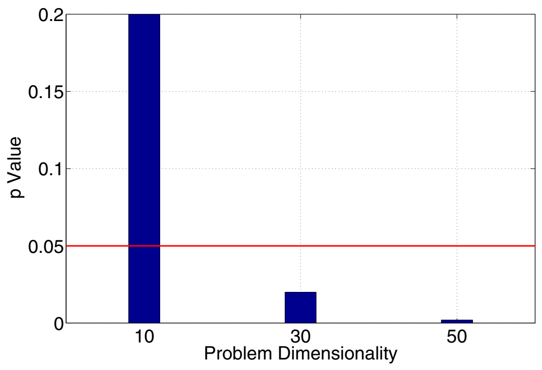


Fig. 4: Exact p values obtained by Wilcoxon test for DEDASF when compared to DETVSF, at problem dimensionality 10, 30, and 50. DEDASF is significant at the 0.05 level of significance at problem dimensionality 30 and 50.

difference, and Wilcoxon test provides a more sensitive overall performance report. Table VII indicates that DEDASF is competitive at problem dimensionality 10 and 30 but does not perform well at 50 dimensions. Part of the reason for less than convincing performance of DEDASF at 50 dimensions may be attributed to the lack of adaptive capabilities of DEFASF.

TABLE VII: Performance of DEDASF when compared to SaDE.

Dimensions	Wins	Loses	p Val (Sign Test)	p Val (Wilcoxon)
10	8	12	0.50	0.42
30	9	11	0.82	0.73
50	7	13	0.26	0.21

VI. Conclusions

This work presents DEDASF, a variation of the classic DE algorithm wherein the scale factor, F, is altered first with dither and then reduced at a certain rate. We report the performance of DEDASF at three different problem dimensionalities, 10, 30 and 50, on the twenty benchmark functions. We compare DEDASF with DETVSF (another algorithm that reduces F with time with a constant factor), DEWD (an algorithm that randomizes F), the classic DE, and SaDE.

We conduct a post-hoc analysis using the Hochberg procedure to determine the best performing deterministic parameter control algorithm. The results indicate that though there is no significant difference between the compared algorithms at problem dimensionality 10, DEDASF is significant at significance level 0.1 at problem dimensionality 30 and 50. Also, DEDASF is significant at significance level 0.05 when only scale factor reduction algorithms, namely DEDASF and DETVSF, are compared. DEDASF also fairs well at problem dimensionality 10 and 30 when compared with SaDE but is outperformed by SaDE at 50 dimensions, though the difference is not significant. Moreover, an interesting finding from this work is the observation that different functions respond differently to various cooling rates. Future work includes testing the scheme at higher dimensions and on a variety of test problems, and performing further analysis to contrast the behavior of the functions when subjected to different step sizes.

References

[1] R. M. Storn and K. V. Price, “Differential evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces,” International Computer Science Institute, Berkeley, CA, USA, ICSI Technical Report 95-012, Mar. 1995.

[2] S. Das and P. N. Suganthan, “Differential evolution - A survey of the state-of-the-art,” IEEE Transactions on Evolutionary Computation, vol. 15, no. 1, pp. 4-31, Feb. 2011.

[3] R. M. Storn and K. V. Price, “Minimizing the real functions of the ICEC 1996 contest by differential evolution,” in Proc. IEEE Int. Conf. Evol. Comput., 1996, pp. 842-844.

[4] A. E. Eiben, R. Hinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms,” IEEE Trans. Evol. Comput., vol. 3, no. 2, pp. 124-141, Jul. 1999.

[5] J. Liu and J. Lampinen, “On setting the control parameter of the differential evolution method” in Proc. 8th Int. Conf. Soft Computing (MENDEL), 2002, pp. 11-18 .

[6] R. Gamperle, S. D. Muller, and P. Koumoutsakos, “A parameter study for differential evolution” NNA-FSFS-EC 2002. Interlaken, Switzerland, WSEAS, Feb. 11-15, 2002.

[7] A. E. Eiben and J. E. Smith, “Introduction to Evolutionary Computing,” ser. Natural Computing. Berlin, Germany: Springer-Verlag, 2003.

[8] K. Price, R. Storn, and J. Lampinen, “Differential Evolution - A Practical Approach to Global Optimization” Berlin, Germany: Springer, 2005.

[9] S. Das, A. Konar, and U. K. Chakraborty, “Two improved Differential Evolution schemes for faster global search,” in Proc. ACM-SIGEVO GECCO, Jun. 2005, pp. 991-998.

[10] H. A. Abbass, “The self-adaptive pareto differential evolution algorithm,” in Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC02), Honolulu, Hawaii, USA, vol. 1, pp. 831-836, 2002.

[11] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” IEEE Transactions on Evolutionary Computation, vol. 10, no. 6, pp. 646-657, Dec. 2006.

[12] R. Storn and K. Price, “Differential evolutionA simple and efficient heuristic for global optimization over continuous spaces” Journal of Global Optimization, vol. 11, pp. 341-359, 1997.

[13] D. Karabogam and S. Okdem, “A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm,” in Turkish Journal of Electrical Engineers, vol.12(1), pp. 53-60, 2004.

[14] R. Storn, “Digital Filter Design Program” FIWIZ 2000 [Online]. Available: http://www.icsi.berkeley.edu/~storn/fiwiz.html.

[15] R. Hooke And T. A. Jeeves, “Direct Search Solution of Numerical and Statistical Problems,” Journal of the ACM (JACM), vol. 8, Issue 2, pp. 212-229, April 1961.

[16] U. K. Chakraborty, “Advances in Differential Evolution,” in Differential Evolution Research-Trends and Open Questions, Springer, 2008, pp. 11-12.

[17] J. J. Liang, B.Y. Qu, P. N. Suganthan, and A. G. Hernandez-Daz, “Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization,” Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang, 2013.

[18] A. K. Qin, Xiaodong Li, “Differential Evolution on the CEC-2013 Single-Objective Continuous Optimization Testbed,” IEEE Congress on Evolutionary Computation, Cancun, Mexico, June 20-23, 2013.

[19] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” Journal of the American Statistical Association 32, 1937, pp. 674-701.

[20] Y. Hochberg, “A sharper Bonferroni procedure for multiple tests of significance,” Biometrika, 1988, pp. 800-803.

[21] A. K. Qin and P. N. Suganthan, ”Self-adaptive Differential Evolution Algorithm for Numerical Optimization”, Proc. IEEE Congress on Evolutionary Computation, Sept. 2005.

[22] A. K. Qin, Xiaodong Li, Hong Pany, Siyu Xiay, ”Investigation of Self-adaptive Differential Evolution on the CEC-2013 Real-Parameter Single-Objective Optimization Testbed,” IEEE Congress on Evolutionary Computation, Cancun, Mexico, June 20-23, 2013.