DE GRUYTER OPEN

JAISCR, 2018, Vol. 8, No. 3, pp. 211 – 235
10.1515/jaiscr-2018-0014

EFFECT OF STRATEGY ADAPTATION ON DIFFERENTIAL EVOLUTION IN PRESENCE AND ABSENCE OF PARAMETER ADAPTATION: AN INVESTIGATION

Deepak Dawar[1] and Simone A. Ludwig[2]

[1]*Department of Computer and Information Technology, Miami University Hamilton, Ohio, USA*

[2]*Department of Computer Science, North Dakota State University, Fargo, ND, USA*

Submitted: 13th September 2017; accepted: 3rd September 2017

Abstract

Differential Evolution (DE) is a simple, yet highly competitive real parameter optimizer in the family of evolutionary algorithms. A significant contribution of its robust performance is attributed to its control parameters, and mutation strategy employed, proper settings of which, generally lead to good solutions. Finding the best parameters for a given problem through the trial and error method is time consuming, and sometimes impractical. This calls for the development of adaptive parameter control mechanisms. In this work, we investigate the impact and efficacy of adapting mutation strategies with or without adapting the control parameters, and report the plausibility of this scheme. Backed with empirical evidence from this and previous works, we first build a case for strategy adaptation in the presence as well as in the absence of parameter adaptation. Afterwards, we propose a new mutation strategy, and an adaptive variant SA-SHADE which is based on a recently proposed self-adaptive memory based variant of Differential evolution, SHADE. We report the performance of SA-SHADE on 28 benchmark functions of varying complexity, and compare it with the classic DE algorithm (DE/Rand/1/bin), and other state-of-the-art adaptive DE variants including CoDE, EPSDE, JADE, and SHADE itself. Our results show that adaptation of mutation strategy improves the performance of DE in both presence, and absence of control parameter adaptation, and should thus be employed frequently.

Keywords: Evolutionary algorithms, Differential evolution, mutation strategy, adaptive control

1 Introduction

Challenging real world optimization problems are ubiquitous in scientific and engineering domains. Complexity of the problem notwithstanding, its objective function may also be non-continuous, and non-differentiable adding to the overall difficulty, and negotiability of the search space. Researchers have been looking towards Darwinian inspired evolutionary theories like social group behavior, and foraging strategies, to name a few, for tackling hard, and complex optimization problems. Nature inspired algorithms are the outcomes of such research activity. These algorithms can be broadly

classified into two categories: evolutionary computing methods, and swarm intelligence algorithms, both of which employ their own set of control parameters.

The underlying idea behind evolutionary algorithms is the iterative fitness improvement of a population of individuals (solutions), through natural selection. An iteration generally involves, producing new individuals through a series of mutations and recombinations, gradually removing lesser fit individuals from the population, and replacing them with newly generated individuals if their fitness proves to be better than the individuals they were generated to replace [1]. The operation of swarm intelligence algorithms may be behaviorally characterized as a decentralized swarm searching for optimal food sources (solutions) [2]. The direction of individual search is influenced by the current location of the individual, its best location ever, and the location of the best individual in the whole swarm. The performance of both these classes of algorithms is quite sensitive to their respective control parameter settings, good values of which are problem dependent. Unless the user has quite an experience in parameter tuning, finding the best parameter settings for a given problem through trial and error may prove, at best, an arduous, and sometimes an infeasible task. A way out of this conundrum lies in an arrangement that may alter or adapt these parameters during the course of the algorithm. Much attention has been paid to this problem and many adaptive schemes have been proposed in the past [3]-[7].

Lately, Differential Evolution (DE) [8], an evolutionary algorithm, has established itself as a robust real parameter optimizer. Intensive research activity on the subject in the past decade speaks volumes of its power and popularity. DE has been rigorously evaluated on a broad range of benchmark problems, and has been extensively applied to real life scientific, and engineering problems [9]. It also secured first position in the First International Contest on Evolutionary Optimization in May 1996 [10].

DE is simple and operates with only a few control parameters namely scale factor (F), crossover rate (Cr), and population size (NP). The performance of DE, as with any evolutionary algorithm, is quite sensitive to the appropriate settings of these parameters as reported in [1], [11], [12]. A good setting can improve both the convergence speed, and the quality of the solution. Conversely, a poorly chosen setting of these parameters can seriously deteriorate the algorithm's efficacy. Given the importance the parameter setting carries, choosing effective control parameter values, at the same time, can be quite a tedious task.

Generally, an effective combination of these parameters depends upon the problem being tackled, and necessitates a good amount of user experience. It would not be inappropriate to remark that the more informed values of these control parameters are, the better the results. While the role of good parameter settings in DE's performance may be unequivocal, there is no single accepted scheme to ascertain their universally applicable or effective values.

As a result, a good deal of research effort has been spent to devise and further improve the alter/adapt schemes to automatically find good, and acceptable values of these control parameters. These methods were categorized in [1] and [13], into three major classes:

- **Deterministic -** the parameters are altered based on some user defined rules [8], [14].
- **Adaptive -** the parameters are allowed to adapt based on some feedback from the algorithm [15].
- **Self-Adaptive -** the parameters are encoded into the solution itself and they evolve as a part of the general population [16], [17], [18].

During the search process a particular combination of control parameters and mutation strategy may prove more favorable than the others [19]. As a result, many partially [20]-[22] adaptive schemes that adapt one or more control parameters, and fully adaptive schemes [23] that adapt mutation strategy and control parameters, have been proposed in the past.

Our contributions in this paper are as follows: (1) We investigate the efficacy of employing an adaptive mutation strategy module both in presence and absence of a control parameter adaptation scheme. Based on empirical results obtained

through experimentation, we create a pool of successful mutation strategies. (2) We propose a memory based fully adaptive version of Differential Evolution, SA-SHADE, that adapts the control parameters to their appropriate values and chooses the best suited mutation strategy from the pool. This variant of DE is quite different from previous work. (3) Empirical results are obtained using 28 benchmark functions in order to show the efficacy of our approach.

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 presents the empirical results for building a case for strategy adaptation irrespective of parameter adaptation. In Section V, SA-SHADE is described with all its features and then compared with state-of-the-art adaptive DE variants. Section VI concludes this paper.

2 Differential Evolution

Like any evolutionary algorithm, DE works with a population of solutions which is expressed as a set of NP D dimensional real parameter vectors, X=(x_1,....,x_{NP}) where x_i ($1 \leq i \leq NP$) is a D dimensional vector. In broadly accepted DE terminology, each solution is called a vector and we use the terms solution, individual, and vector interchangeably. Essential steps of DE are explained in Sections 2.1 to 2.4.

2.1 Initialization

Being a global optimizer, DE searches for the optimum in a D dimensional real parameter space R^D. As a first step, a NP number of D dimensional vectors are randomly initialized to form a population. The idea behind this random initialization is to allow the population the possibility to cover the complete landscape of the objective function.

The i^{th} vector ($1 \leq i \leq NP$), X^i, in the current generation G can be represented as

$$X^i_{j,G} = [x^i_{1,G}, x^i_{2,G}, x^i_{3,G}, ..., x^i_{D,G}] \qquad (1)$$

Every parameter x^i_j in a given vector has a specific range denoted by x^{min}_j and x^{max}_j, within which it has to be restricted, where $1 \leq j \leq D$.

2.2 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], and are also different from the target vector index i. The donor is then created as

$$D^i_G = X^{r1}_G + F \times (X^{r2}_G - X^{r3}_G) \qquad (2)$$

where F is known as the scale factor. Equation 2 represents the classic DE mutation strategy *DE/rand/1* [24] 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 [8], [24], but *DE/rand/1* is the most widely used [11], [12], [28].

2.3 Crossover

To improve the potential diversity of the population, after the mutation step, a crossover operation is performed on every pair of target vector, X^i_G, and its corresponding donor vector, D^i_G. In this step, parameters either from the target vector, X^i_G, or the donor vector, D^i_G, are selected based on some probability distribution to form a trial (child) vector, T_G. There are two commonly employed crossover operations in DE literature [14]. One of them, known as *binomial* or *uniform* crossover may be elucidated as

$$T^i_{j,G} = \begin{cases} D^i_{j,G} & \text{if } rand^i_j[0,1] \leq Cr \text{ or } j = j_{rand} \\ X^i_{j,G} & \text{otherwise} \end{cases} \qquad (3)$$

where T^i_G is the trial vector generated for the i_{th} target vector, X^i_G, for generation G, $rand^i_j$ [0, 1] is a randomly generated real number ranging between the interval [0, 1], and is generated newly 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.

The other crossover method frequently used is the exponential or the two-point modulo crossover in which a random integer n_1 is chosen from the interval [1, D] that represents the starting point in the target vector X^i_G from where the crossover operation would start. After that, another integer n_2 is chosen from the same interval that specifies the number of

parameters that the donor vector D^i_G contributes to the formation of trial vector T^i_G. This scheme may be described as

$$T^i_{j,G} = \begin{cases} D^i_{j,G} & \text{for j = } (n_1) \bmod D \\ X^i_{j,G} & \text{otherwise} \end{cases} \quad (4)$$

A new set of n_1 and n_2 is chosen for every donor vector.

2.4 Selection

The objective function is evaluated for all trial vectors in this step. The fitness value of each target vector is compared to its corresponding trial vector. If the fitness (considering a minimization problem) of the trial vector is better or at least equal to the target vector, it moves to the next generation. Otherwise the target vector is promoted and trial vector is discarded.

$$X^i_{G+1} = \begin{cases} T^i_G & \text{if } f(T^i_G) \leq f(X^i_G) \\ X^i_G & \text{otherwise.} \end{cases} \quad (5)$$

The mutation, crossover, and selection steps are iteratively performed generation after generation until a stopping criterion specified by the user is met.

Algorithm 1 Pseudo-code for DE

1: Set the values of control parameters scale factor (F), crossover rate (Cr), and population size (NP)
2: Set generation number, G=0
3: 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^j_i=[x^1_i, x^2_i ... x^D_i] where $1 \leq j \leq D$; restrict x^j_i to its minimum and maximum bounds as $x^j_{i,min}$ and $x^j_{i,max}$.
4: **while** stopping criteria is not met **do**
5: **for** every target vector X_{ta} in P **do**
6: 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 target index
7: **//Mutation**
8: Produce a donor vector through mutation as:
9: $X_{do} = X_{r_1} + F \times (X_{r_2} - X_{r_3})$
10: **//Crossover**
11: Produce a trial vector as:

$$X^j_{tr} = \begin{cases} X^j_{do} & \text{if } rand_j(0,1] \leq Cr \text{ or } j = j_{rand} \\ X^j_{ta} & \text{otherwise} \end{cases}$$

12: **//Selection**
13: Select either the target vector or the trial vector based on their fitness values as:

$$X^{G+1}_{survivor} = \begin{cases} X^G_{tr} & \text{if } F(X^G_{tr}) \leq F(X^G_{ta}) \\ X^G_{ta} & \text{otherwise} \end{cases}$$

14: **end for**
15: **end while**

3 Related Work

It is an established notion that the performance of DE depends greatly on the mutation strategy employed, and the corresponding control parameters [11], [12], [30], [32]. As the complexity of the problem increases, this dependence becomes even more profound [12]. A good choice of mutation strategy and control parameters can lead to better results, and at the same time, an unfavorable choice may seriously degrade DE's performance [14], [21], [25]. Choosing a good mutation strategy and associated control parameters is not an easy task and requires quite a bit of user experience. A good amount of research activity has happened in the area of determining good values of the control parameters. Authors in [26] suggested that good values of F lie between 0.4 and 0.95. For Cr, they ascribed the range (0, 0.2) for separable functions and (0.9, 1) for non-separable functions. On the other side of the spectrum, the authors in [12] suggested good value of F to be 0.6 and Cr ranging between [0.3, 0.9]. As can be seen, these suggestions differ, and sometimes, are conflicting at best. This situation naturally calls for adaptive mechanisms that would require little or no user intervention in setting up the control parameters while optimizing with DE.

Much work has been reported on this problem of automating mutation strategy and control parameters [19], [27], [28], [29]. A fuzzy adaptive differential evolution with fuzzy logic controllers was presented in [27] where F and Cr are adapted based on the relative fitness values and individuals of subsequent generations. Authors developed linguistic fuzzy sets to encode knowledge by taking into consideration the noise and non-linearity of the objective function. Qin *et al*. [30] proposed a memory based self adaptive differential evolution (SaDE) algorithm. They adapted the mutation strategy depending upon its success history. The mutation

strategy is chosen from a pool and successful strategies and Cr values are recorded. The subsequent mutation strategy is selected probabilistically based on its ability to produce successful trials. The scale factor, F, is not adapted and instead randomly sampled from the normal distribution (0.5, 0.3). The idea was to employ exploration (large F values) and exploitation (small F values) throughout the search process. The successful values of the crossover rate, Cr, on the other hand, are stored in a memory bank, and new values of Cr are generated from the normal distribution $N(Cr_m, 0.1)$, where Cr_m is the median Cr value in the memory bank m. A small value of standard deviation 0.1 was chosen to guarantee that most of the Cr values generated by $N(Cr_m, 0.1)$ are between [0,1], even when Cr_m is close to 0 or 1.

In the self-adaptive scheme, jDE, proposed by Brest *et al*. [17], F and Cr are encoded directly into the individuals so that individuals with better values of these parameters are more likely to survive, thus automatically retaining good parameter values, increasing the length of the vector. Two new parameters τ_1 and τ_2 are introduced to control the values of F and Cr as

$$F^i_{G+1} = \begin{cases} F_l + rand_l \times F_u & \text{with probability } \tau_1 \\ F^i_G & \text{otherwise} \end{cases} \quad (6)$$

$$Cr^i_{G+1} = \begin{cases} rand_2 & \text{with probability } \tau_2 \\ Cr^i_G & \text{otherwise} \end{cases} \quad (7)$$

where F_l and F_u are the lower and upper limits of F restricted to the range [0,1]. The authors used τ_1=τ_2=0.1 with F_l=0.1 and F_u=0.9. Thus, essentially F (0.1, 0.9) and CR (0, 1) are restricted to their respective ranges. It should be noted that this scheme has four extra parameters to be set namely F_l, F_h, τ_1, and τ_2, which might pose a problem in itself. The authors, in this case, opined to have used only a single setting for them and kept them constant throughout the search. Recently a large scale study of τ_1, and τ_2 was conducted in [18].

A fitness based adaptation of F was proposed in [31]. Cr was fixed at 0.5. The mechanism comprised of two evolving populations. After every generation, F was updated as

$$F = \begin{cases} \max(l_{min}, 1 - \frac{f_{max}}{f_{min}}) & \text{if } \frac{f_{max}}{f_{min}} < 1 \\ \max(l_{min}, 1 - \frac{f_{min}}{f_{max}}) & \text{otherwise} \end{cases} \quad (8)$$

where f_{min} and f_{max} are the generational minimum and maximum objective function values obtained by the individuals over the populations and l_{min} is the lower bound on F.

In [32], authors proposed a scheme wherein F was reduced linearly with an increase in the number of function evaluations. The idea was to use high values of F during the exploration stage and small values during the exploitation state in the later part of the search. Dawar *et al*. in [33] proposed a similar technique with the difference that they used random perturbation of F in the initial stages of the search, and reduced F non-linearly afterwards. Both of the above approaches demonstrated favorable performance over conventional DE.

Authors in [20] proposed another adaptive version of DE named SDE, in which F and population size NP were adapted but Cr was sampled from a normal distribution N(0.5, 0.15). SDE was reported to have outperformed other basic versions of DE described in [8]. On similar lines, DESAP (Differential evolution with self adaptive population size) was proposed by Teo [34] in which the population size, NP was adapted alongside F and Cr. Population size reduction has also been reported to have a favorable effect on the performance of DE as argued in [35]. Authors of the same work reported an improvement in both efficiency and robustness of DE when NP is gradually reduced.

Another novel adaptive mechanism proposed in [23] uses three different pools of values, one for each mutation strategy, F, and Cr, respectively. The F pool contained the values in the range [0.4, 0.9] with an increment of 0.1, and the CR pool had values in the range [0.1, 0.9] with 0.1 increments. The mutation strategy pool contained three strategies namely rand/1/bin, best/2/bin, and current-to-rand/1/bin. Initially every individual is randomly assigned a set of [F, Cr, Ms] and during the search successful sets are carried forward to the next generation while unsuccessful sets are re-initialized. The parameter Ms in the set denotes a mutation strategy.

In [36], the authors presented an adaptive scheme, JADE, and also proposed a new mutation strategy current-to-$p$best/1. The scheme also included a diversity maintenance mechanism by keeping an optional external archive of unsuccessful parents that were unable to move to the next generation owing to their worse fitness. The muta-

tion strategy current-to-*p*best/1 that the authors employed is different from the basic current-to-*p*best/1 strategy in the sense that the individual *p*best can be selected from a user controlled set of top individuals instead of representing just the top individual. The search mechanism of current-to-*p*best/1 is quite greedy in nature and experimentally, it has been shown that this greediness often leads to poor performance on multimodal functions [37]. In other words, the greediness of this basic mutation strategy can be controlled to some extent in the new version. The donor from current-to-*p*best/1 is obtained as

$$D_{i,G} = X_{i,G} + F_i \times (X_{pbest,G} - X_{i,G}) + \\ + F_i \times (X_{r1,G} - X_{r2,G}) \quad (9)$$

where the individual $x_{pbest,G}$ is randomly selected from the top $NP \times n$ ($n \in [0,1]$) members in the G-th generation. Here, n may be regarded as the greediness control operator. The authors adopted a memory based control parameter adaptation scheme. F and Cr were drawn from a normal N (μ_F, 0.1) and a cauchy C (μ_{Cr}, 0.1) distribution respectively where μ_F and μ_{Cr} are the respective mean values of the distributions. At the beginning of the search, μ_F and μ_{Cr} are initialized to 0.5 and adapted thereafter as

$$\mu_{Cr} = (1-c)\mu_{Cr} + c.A(S_{Cr}) \quad (10)$$

$$\mu_F = (1-c)\mu_F + c.L(S_F) \quad (11)$$

where c is the learning rate which was suggested to be set to 0.1. S_F and S_{Cr} are the successful values stored in the memory during the generation. A and L are the arithmetic and Lehmer means, respectively.

Several extensions to JADE have been proposed. Authors in [38] propose a restart strategy for JADE and also suggest replacing the arithmetic mean in Equation 10 by a weighted mean, where higher weights are assigned to Cr values that achieve a higher fitness difference. A co-evolutionary extension to JADE was proposed in [39]. In [40], authors adaptively select the mutation strategy to be applied among current-to-*pbest*/1 with/without the external archive, and rand-to-*pbest*/1 with/without the external archive. JADE has also been successfully applied to combinatorial and multi-objective optimization problems [41], [42].

In another memory based parameter adaptation scheme called success history based adaptive DE (SHADE) [43], authors improve upon the robustness of JADE. They argue that the continuous mean update mechanism used in JADE may allow unfavorable values of F and CR to impact their mean value thereby allowing the possibility of a degraded search performance. They maintain a historical memory of means as M_F, and M_{Cr} which are the successful values of means calculated from S_F and S_{Cr}. In essence, SHADE maintains a pool of successful pairwise means in contrast with JADE, which works with a single pair of means. In case an unfavorable set of μ_F and μ_{Cr} are recorded, its impact would be far less profound as there may be other successful and favorable means in the pool to offset this disadvantage. SHADE was shown to have outperformed JADE in [44].

Apart from F and Cr, adaptation of population size NP has also received much attention. The population size significantly impacts the convergence rate of any evolutionary algorithm, with DE being no exception. A smaller value of population size, NP, tends to favor exploitation and the solution converges faster while always breaming with the possibility of getting stuck in a local minima. Large population sizes favor exploration of the landscape thereby slowing down the convergence rate. Many population resizing methods have been proposed that have shown to be effective in improving the performance of evolutionary algorithms [35], [45]-[48]. These methods of population size reduction are essentially deterministic instead of adaptive, as they increase or decrease the population size based on some predefined rules.

One of such techniques named Linear Population Size Reduction (LPSR) was incorporated by authors in [49] to enhance the performance of SHADE. Authors in [50] proposed jDE*dyn*NP, a self adaptive version of DE, in which F and Cr are self adapted and a population size reduction technique is used. This algorithm is further extended to work with multiple mutation strategies in [51] but these mutation strategies are not adapted. Population size adaptation has not been investigated in this work.

4 Experimentation And Results

4.1 Benchmark Functions

To gauge the impact of strategy adaptation with and without adapting F and Cr, we first evaluated five basic mutation strategies on 28 benchmark functions listed in [52] at problem dimensionality 10, 30, and 50 results of which are tabulated in Tables 1, 2, and 3, respectively, and are discussed in the next section. The parameter values used in this experiment were NP=100, F=0.5 and Cr=0.9 as suggested in [43], [53].

4.2 Relative performance of basic strategies

We initially experimented with five basic mutation strategies described below. Uniform crossover and population size of 100 was used with every mutation strategy.

Rand/1:

$$X_i = X_{r1} + F \times (X_{r2} - X_{r3}) \tag{12}$$

Rand/2:

$$X_i = X_{r1} + F \times (X_{r2} - X_{r3}) + F \times (X_{r4} - X_{r5}) \tag{13}$$

Best/2:

$$X_i = X_{best} + F \times (X_{r1} - X_{r2}) + F \times (X_{r3} - X_{r4}) \tag{14}$$

CurrToBest:

$$X_i = X_{target} + F \times (X_{best} - X_{target}) + \\ + F \times (X_{r1} - X_{r2}) \tag{15}$$

RandToBest:

$$X_i = X_{r1} + F \times (X_{best} - X_{r2}) + F \times (X_{r3} - X_{r4}) \tag{16}$$

Our first experiment led to a generally accepted result that different mutation strategies perform differently on different problems. Table 4 summarizes the relative performance of the basic mutation strategies as the ranks obtained by applying Friedman test [54] at problem dimensionality 10, 30, and 50.

The Friedman test [54] is a multiple comparisons procedure that aims to detect significant performance differences between the k compared algorithms where $k \geq 2$. 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 [58]. The Friedman statistic is computed as

$$F_s = \frac{12 n_p}{n_a(n_a+1)} [\Sigma R_i^2 - \frac{n_a(n_a+1)^2}{4}] \tag{17}$$

where n_p is the number of test problems, n_a is the number of algorithms being compared, and R_i is the relative rank of the i^{th} algorithm.

The statistic F_s is distributed according to the χ^2 distribution with $n_a - 1$ degrees of freedom, when $n_p > 10$ and $n_a > 5$. If the number of algorithms and test problems are small, then critical values have been computed and presented, see [55], [56] for more details.

It is clear that Rand/1 is relatively the most consistent strategy across problem dimensionality. The next question one might ask - "Is the performance of Rand/1 statistically significant?". The experimental results pertaining to this question are shown in Table 5 which contains the p-values (for α = 0.05) obtained by applying the Hochberg post hoc method [57] over the results of Table 1, 2, 3, respectively.

Table 4: Relative ranks obtained by Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 10D, 30D, and 50D respectively. The best rank is highlighted in bold.

Strategy	Rank-10D	Rank-30D	Rank-50D
Rand/1	**2.46**	**2.01**	**2.00**
Rand/2	3.97	4.14	4.23
Best/2	3.5	3.08	3.08
RandToBest	2.5	2.58	2.41
CurrToBest	2.57	3.16	3.26

Given a control algorithm, the Hochberg method tries to identify the algorithms that are better or worse by calculating p multiple values. The Hochberg method adjusts the value of α in a step up way. It works by comparing the largest p-value with α, the next largest with $\alpha/2$, the next with $\alpha/3$, and so forth until it finds a hypothesis that definitely reject it. All hypotheses with smaller p-values are then rejected as well [58].

The results in Table 5 show that Rand/1 significantly outperforms Rand/2 and Best/2 at every problem dimension. At 10 dimensions there is not much of a performance difference between

Table 1: Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 10. Reported values are the averages of 51 independent runs for each function. Error values reaching 10^{-8} of the global optimum of the function are reported as 0.00+E00. The best result is highlighted in bold.

Function	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	**0.00E+00**	3.54E+00	3.40E-06	3.36E+04	9.03E+02
3	5.21E-01	5.23E+03	**1.38E-01**	1.30E+06	2.76E+05
4	**0.00E+00**	1.87E-02	2.26E-08	2.33E+02	3.04E+01
5	0.00E+00	0.00E+00	0.00E+00	6.33E-01	3.25E-04
6	4.50E-03	1.64E-06	**0.00E+00**	6.37E+00	7.66E+00
7	4.80E-04	1.29E+00	7.71E-02	5.07E-02	**7.39E-05**
8	2.04E+01	2.04E+01	2.04E+01	2.04E+01	2.04E+01
9	**1.93E-01**	6.75E+00	4.88E+00	8.65E-01	5.57E-01
10	3.60E-01	5.39E-01	5.47E-01	**5.16E-03**	1.07E-02
11	1.73E+01	2.49E+01	2.22E+01	**2.03E+00**	8.24E+00
12	2.59E+01	3.23E+01	3.21E+01	**1.02E+01**	1.54E+01
13	2.59E+01	3.14E+01	2.96E+01	**1.01E+01**	1.70E+01
14	1.02E+03	1.28E+03	1.24E+03	**7.88E+02**	1.01E+03
15	1.25E+03	1.36E+03	1.37E+03	**1.04E+03**	1.16E+03
16	9.99E-01	1.14E+00	1.15E+00	**9.48E-01**	9.69E-01
17	3.07E+01	3.99E+01	3.54E+01	**1.76E+01**	1.80E+01
18	3.58E+01	4.60E+01	4.19E+01	2.56E+01	**2.46E+01**
19	1.84E+00	2.62E+00	2.45E+00	**1.41E+00**	1.64E+00
20	2.54E+00	2.95E+00	2.65E+00	**1.95E+00**	2.09E+00
21	3.72E+02	**3.05E+02**	3.81E+02	4.00E+02	4.00E+02
22	9.47E+02	1.44E+03	1.29E+03	**3.39E+02**	9.11E+02
23	1.16E+03	1.39E+03	1.33E+03	**5.83E+02**	9.39E+02
24	**1.98E+02**	2.05E+02	2.05E+02	2.01E+02	2.01E+02
25	2.00E+02	2.00E+02	2.02E+02	2.00E+02	2.00E+02
26	1.26E+02	1.39E+02	1.46E+02	1.23E+02	**1.18E+02**
27	3.00E+02	3.07E+02	3.00E+02	3.05E+02	3.00E+02
28	**2.52E+02**	2.90E+02	2.52E+02	3.00E+02	3.00E+02

Table 2: Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 30D. 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. The best result is highlighted in bold.

Function	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest
1	0.00E+00	2.40E-01	0.00E+00	1.18E+02	5.95E+02
2	**5.09E+05**	3.72E+07	1.41E+06	4.21E+06	3.16E+06
3	**2.29E-02**	1.71E+09	5.26E+04	1.97E+09	1.91E+09
4	**9.98E+02**	3.54E+04	7.30E+03	4.56E+03	2.63E+03
5	**0.00E+00**	5.06E-01	4.83E-10	2.43E+02	4.21E+02
6	9.21E+00	1.80E+01	**6.26E+00**	1.04E+02	1.08E+02
7	**1.00E-01**	5.82E+01	9.91E+00	1.79E+01	1.15E+01
8	2.09E+01	2.09E+01	2.10E+01	2.09E+01	2.09E+01
9	2.25E+01	3.89E+01	3.84E+01	**8.68E+00**	1.04E+01
10	5.40E-03	2.55E+01	**1.39E-04**	5.80E+01	9.98E+01
11	1.23E+02	2.10E+02	1.87E+02	**2.06E+01**	8.89E+01
12	1.77E+02	2.26E+02	1.98E+02	**7.71E+01**	1.60E+02
13	1.72E+02	2.28E+02	1.98E+02	**1.37E+02**	1.66E+02
14	6.25E+03	6.81E+03	6.85E+03	**6.22E+03**	6.43E+03
15	7.12E+03	7.31E+03	7.22E+03	**6.57E+03**	6.84E+03
16	2.49E+00	**2.45E+00**	2.57E+00	2.48E+00	2.51E+00
17	1.83E+02	2.69E+02	2.23E+02	**1.59E+02**	1.67E+02
18	2.12E+02	2.83E+02	2.31E+02	**1.78E+02**	1.86E+02
19	**1.50E+01**	2.04E+01	1.73E+01	3.44E+01	3.43E+01
20	**1.21E+01**	1.27E+01	1.26E+01	1.21E+01	1.24E+01
21	**2.91E+02**	3.12E+02	3.10E+02	6.69E+02	6.67E+02
22	6.45E+03	6.95E+03	6.90E+03	**4.52E+03**	5.89E+03
23	7.14E+03	7.23E+03	7.08E+03	**6.10E+03**	6.68E+03
24	**2.00E+02**	2.70E+02	2.06E+02	2.17E+02	2.21E+02
25	**2.39E+02**	2.82E+02	2.51E+02	2.47E+02	2.45E+02
26	2.00E+02	2.03E+02	2.00E+02	2.00E+02	2.11E+02
27	**3.37E+02**	1.17E+03	5.93E+02	5.13E+02	4.64E+02
28	3.00E+02	3.25E+02	3.00E+02	3.25E+02	9.93E+02

Table 3: Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest at 50D. 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. The best result is highlighted in bold.

Function	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest
1	0.00E+00	7.13E+01	0.00E+00	1.44E+03	3.55E+03
2	**2.70E+06**	2.30E+08	2.23E+07	1.76E+07	1.07E+07
3	**3.36E+05**	3.15E+10	3.90E+06	8.89E+09	1.07E+10
4	2.10E+04	7.38E+04	4.32E+04	4.50E+03	**3.47E+03**
5	**0.00E+00**	1.84E+01	8.11E-09	5.56E+02	9.32E+02
6	**4.34E+01**	6.04E+01	4.34E+01	2.17E+02	2.54E+02
7	**1.03E+00**	1.24E+02	2.03E+01	2.86E+01	2.98E+01
8	**2.11E+01**	2.11E+01	2.11E+01	2.11E+01	2.11E+01
9	7.04E+01	7.23E+01	7.23E+01	**2.36E+01**	2.39E+01
10	4.06E-02	4.63E+02	**1.42E-02**	3.08E+02	4.17E+02
11	2.16E+02	4.32E+02	3.65E+02	**6.69E+01**	8.27E+01
12	3.61E+02	4.77E+02	3.86E+02	**7.87E+01**	2.34E+02
13	3.51E+02	4.79E+02	3.83E+02	**3.14E+02**	3.68E+02
14	**1.13E+04**	1.30E+04	1.30E+04	1.21E+04	1.26E+04
15	1.39E+04	1.39E+04	1.39E+04	**1.31E+04**	1.35E+04
16	3.33E+00	**3.17E+00**	3.32E+00	3.31E+00	3.36E+00
17	**3.30E+02**	5.45E+02	4.19E+02	3.48E+02	3.81E+02
18	4.01E+02	5.60E+02	4.43E+02	**3.75E+02**	3.97E+02
19	**2.97E+01**	4.93E+01	3.36E+01	4.54E+02	1.34E+03
20	2.21E+01	2.27E+01	2.24E+01	**2.06E+01**	2.07E+01
21	4.06E+02	4.31E+02	**2.74E+02**	2.06E+03	2.30E+03
22	1.08E+04	1.34E+04	1.32E+04	**3.88E+03**	1.19E+04
23	1.37E+04	1.39E+04	1.39E+04	**1.23E+04**	1.30E+04
24	**2.07E+02**	3.61E+02	2.14E+02	2.60E+02	2.68E+02
25	**2.78E+02**	3.81E+02	3.13E+02	3.31E+02	3.31E+02
26	**2.45E+02**	3.45E+02	3.76E+02	3.15E+02	2.90E+02
27	**5.71E+02**	2.04E+03	1.22E+03	8.96E+02	9.57E+02
28	4.00E+02	4.59E+02	4.00E+02	1.26E+03	1.53E+03

Rand/1, RandToBest, and CurrToBest. At 30 dimensions though, Rand/1 significantly outperforms CurrToBest, and RandToBest remains the only competitive strategy against Rand/1, and this observation is repeated at 50 dimensions.

Table 5: p values obtained using Hochberg procedure by mutation strategies Rand/2, Best/2, RandToBest and CurrToBest when compared to Rand/1 at 10D, 30D, and 50D respectively at α level 0.05.

Strategy	p_{Hoc}-10D	p_{Hoc}-30D	p_{Hoc}-50D
Rand/2	**0.001**	**0.000**	**0.000**
Best/2	**0.042**	**0.020**	**0.019**
RandToBest	0.932	0.176	0.331
CurrToBest	0.932	**0.022**	**0.008**

The first inference that can be drawn from these results is that for the given number of function evaluations and in the absence of control parameter adaptation, Rand/1 remains the most competitive strategy across problem dimensionality, and the relative competitiveness of Rand/1 against Rand/2, Best/2, and CurrToBest marginally improves with an increase in problem dimensionality. RandToBest stands at second rank according to the Friedman ranking procedure but statistically inseparable when compared to Rand/1.

The performance of Best2 relative to CurrToBest improves as problem dimensionality increases as it is ranked 3 at 50 dimensions as compared to 4 at 30 dimensions. Rand2 turns out to be the worst performing strategy at every problem dimensionality.

To ascertain the relative competitiveness of RandToBest and CurrToBest at 30D and 50D, we performed the Wilcoxon test, and the results are presented in Table 6.

Table 6: Results obtained by the Wilcoxon test for strategy RandToBest against CurrToBest

Problem Dimensionality	Asymptotic P-value
30	0.070897
50	0.001324

Table 6 shows that RandToBest turns out to be a better performing strategy, significantly outperforming CurrToBest at 50 (α=0.05) and 30 (α=0.1) dimensions.

4.3 A case for strategy adaptation irrespective of parameter adaptation

Looking at the above results, one may surmise that Rand1 is relatively the most robust strategy, and can be used with confidence while optimizing with DE. This observation, however, requires greater scrutiny. Table 7 shows the number of wins scored by all the mutation strategies at 10, 30, and 50 dimensions. Functions on which multiple strategies score equally are not counted as wins.

Cumulative results presented in Tables 5 and 7 coupled with works reported in [19], [30] are indicative that no single strategy has the ability to perform relatively better on all the problems. This is evidence that calls for automating the selection of mutation strategies during the course of the search.

It must be noted that several works in the past [37], [59], [60] have evaluated multiple variants of DE on various real life and benchmark functions, and have arrived at seemingly contrasting results, possibly due to varying test subjects and problems. For example, authors in [59] reported that DE/Best/* variants perform much better than DE/Rand/* variants on the problem of optimal design of shell-and-heat tube exchangers. On similar lines as ours, authors in [60] report superior performance of DE/Rand/1/bin.

As is clear from the literature, the different strategies seem to work well on different problems, and different authors report possibly contrasting claims, a possible reason of which might be different problem sets they worked upon. All in all, in light of these multi-faceted issues, it would be prudent to let the mutation strategy adapt during the search operation. Strategy adaptive variants proposed in the past [19], [30], [61], [62], [63] have reported favorable results.

4.4 Impact of mutation strategy on adaptive control parameter models

To ascertain the importance of mutation strategy employed while using adaptive control parameters (F and Cr) models, we performed experiments on the same test suite, but this time we used a control parameter adaptive model, SHADE, proposed

Table 7: Number of wins scored, out of 28, by all mutation strategies at 10, 30, and 50 dimensions, respectively.

Dimensions	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest
10	5	1	2	**12**	3
30	**11**	1	2	10	0
50	**13**	1	2	9	1

in [43]. The choice of this model was based on the superior performance this demonstrated over other adaptive models as is evident from the work in [43]. After making this choice, we asked the following questions.

1. What impact does a mutation strategy have on the assessed performance of the algorithm with adapted control parameters?
2. Is the impact profound enough to necessitate automated strategy selection?

To answer the first question we tested the adaptive model with multiple mutation strategies at problem dimensionality 10, 30, and 50, the results of which are presented in Tables 8, 8, and 10, respectively. Table 11 and 12 show the ranks and p values obtained using the Friedman and Hochberg test respectively, for the tested mutation strategies at problem dimensionality 10, 30, and 50, respectively.

It is worth noting that SHADE used CurrentTopBest as the mutation strategy that used an external archive of inferior solutions, and draws the $p$best vector from the top x% individuals in the population. Results from Tables 11 and 12 show that when a control parameter adaptation model is used, the strategy used in SHADE is ranked at the top of the strategies tested, and proves vastly superior to Rand1, Rand2, and Best2 at every problem dimensionality. It, however, is not significant when compared to CurrentToBest and RandToBest as is clear from the p values shown in Table 12.

There are two important inferences that can be drawn from these results.

– With or without a given parameter adaptation model, the choice of mutation strategy plays an important role in determining the quality of solutions.

– Given the control parameter adaptation scheme that we have used, RandToBest and CurrentToBest tend to perform relatively better that Rand1, which according to the results tabulated earlier, was the most robust strategy in absence of parameter adaptation.

Table 11: Relative ranks obtained by Rand/1, Rand/2, Best/2, RandToBest, CurrToBest, and SHADE at 10D, 30D, and 50D. AD is prefixed with basic strategies to denote their usage with an adaptive control parameter model. The best rank is highlighted in bold.

Strategy	Rank-10D	Rank-30D	Rank-50D
AD-Rand/1	4.17	4.16	4.30
AD-Rand/2	4.50	5.19	5.00
AD-Best/2	4.05	4.03	4.00
AD-RandToBest	3.16	2.82	2.71
AD-CurrToBest	2.57	2.46	2.71
SHADE	**2.53**	**2.32**	**2.26**

Table 13 shows the number of wins scored by each strategy and provides some insights into investigating the plausibility of automated strategy selection vis-a-vis control parameter adaptation.

Table 12: p values obtained using Hochberg procedure by Rand/1, Rand/2, Best/2, RandToBest, and CurrToBest when compared with SHADE at 10D, 30D, and 50D at α level 0.05. AD is prefixed with basic strategies to denote their usage with an adaptive control parameter model.

Strategy	p_{Hoc}-10D	p_{Hoc}-30D	p_{Hoc}-50D
AD-Rand/1	0.004	0.000	0.000
AD-Rand/2	0.000	0.000	0.000
AD-Best/2	0.000	0.000	0.019
AD-RandToBest	**0.422**	**0.317**	**0.371**
AD-CurrToBest	**0.943**	**0.775**	**0.371**

As is clear from the results in Table 12 and 13, even though the strategy used in SHADE is highest ranked, there is a possibility of further improving SHADE by automating the selection of mutation strategy as the cumulative wins scored by all strategies at every problem dimensionality is greater than the wins recorded by SHADE.

Table 8: Performance of Rand/1, Rand/2, Best/2, RandToBest, CurrentToBest, and SHADE at 10D when employed with adaptive control parameter model used in SHADE. 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. The best result is highlighted in bold.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3	2.30E+01	1.39E+04	**2.08E-05**	6.11E-01	3.13E-01	1.27E-01
4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	8.88E+00	8.41E+00	9.81E+00	9.35E+00	8.41E+00	**7.89E+00**
7	3.18E-01	9.09E-01	1.16E-04	**2.60E-05**	4.26E-02	3.26E-03
8	2.04E+01	2.04E+01	2.04E+01	2.03E+01	2.03E+01	2.04E+01
9	4.16E+00	4.81E+00	4.01E+00	3.64E+00	3.48E+00	**3.39E+00**
10	4.74E-02	6.42E-02	6.42E-02	1.70E-02	**1.02E-02**	1.20E-02
11	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	8.48E+00	1.05E+01	9.00E+00	5.27E+00	3.55E+00	**3.14E+00**
13	1.20E+01	1.31E+01	1.50E+01	7.48E+00	4.98E+00	**3.77E+00**
14	0.00E+00	2.73E-02	2.83E-04	5.95E-03	0.00E+00	4.90E-03
15	7.21E+02	7.19E+02	7.01E+02	6.27E+02	4.57E+02	**4.21E+02**
16	1.01E+00	1.16E+00	1.05E+00	8.92E-01	**4.99E-01**	7.08E-01
17	1.01E+01	**1.00E+01**	1.01E+01	1.01E+01	1.01E+01	1.01E+01
18	2.29E+01	2.17E+01	1.88E+01	1.78E+01	1.77E+01	**1.69E+01**
19	4.11E-01	4.66E-01	4.30E-01	3.65E-01	**3.32E-01**	3.44E-01
20	2.54E+00	2.80E+00	2.32E+00	2.24E+00	2.38E+00	**2.16E+00**
21	4.00E+02	**3.53E+02**	3.81E+02	4.00E+02	4.00E+02	4.00E+02
22	1.70E+01	5.86E+01	3.80E+01	5.58E+00	1.06E+01	**4.84E+00**
23	7.71E+02	8.13E+02	6.94E+02	5.98E+02	5.42E+02	**4.61E+02**
24	2.06E+02	1.98E+02	2.06E+02	2.00E+02	1.93E+02	**1.93E+02**
25	2.01E+02	2.00E+02	2.03E+02	2.00E+02	2.00E+02	2.00E+02
26	1.12E+02	1.22E+02	1.24E+02	1.07E+02	**1.05E+02**	1.33E+02
27	3.00E+02	3.30E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02
28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 9: Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrentToBest at 30D when employed with adaptive control parameter model used in SHADE. 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. The best result is highlighted in bold.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	6.89E+06	1.72E+07	1.11E+07	3.78E+04	1.77E+04	**9.00E+03**
3	4.36E+06	7.06E+07	5.89E+05	1.37E+05	1.31E+05	**4.02E+01**
4	2.55E+04	2.25E+04	1.61E+04	2.45E+03	1.64E-01	**1.92E-04**
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	1.42E+01	1.49E+01	1.23E+01	1.47E+01	8.42E+00	**5.96E-01**
7	2.43E+01	3.59E+01	5.09E+00	**1.97E+00**	3.18E+00	4.60E+00
8	2.09E+01	2.10E+01	2.09E+01	2.06E+01	2.06E+01	2.07E+01
9	2.79E+01	2.81E+01	2.80E+01	**2.74E+01**	2.74E+01	2.75E+01
10	1.23E+00	1.00E+01	8.69E-02	1.72E-01	1.07E-01	**7.69E-02**
11	0.00E+00	0.00E+00	3.79E-01	0.00E+00	0.00E+00	0.00E+00
12	6.28E+01	6.13E+01	4.54E+01	2.52E+01	2.44E+01	**2.30E+01**
13	9.42E+01	9.67E+01	8.57E+01	5.47E+01	5.20E+01	**5.03E+01**
14	1.09E-02	3.60E+00	1.55E+00	2.08E-02	**8.92E-03**	3.18E-02
15	4.72E+03	4.96E+03	4.57E+03	4.38E+03	3.22E+03	3.22E+03
16	1.43E+00	1.96E+00	1.61E+00	**6.09E-01**	9.22E-01	9.13E-01
17	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01
18	1.11E+02	1.42E+02	8.99E+01	**6.89E+01**	7.36E+01	7.25E+01
19	1.79E+00	1.92E+00	1.80E+00	1.52E+00	1.38E+00	**1.36E+00**
20	1.17E+01	1.19E+01	1.10E+01	**1.04E+01**	1.07E+01	1.05E+01
21	**2.48E+02**	2.74E+02	2.81E+02	2.95E+02	2.97E+02	3.09E+02
22	**8.19E+01**	2.07E+02	1.52E+02	1.08E+02	9.79E+01	9.81E+01
23	4.83E+03	5.16E+03	4.44E+03	4.57E+03	3.74E+03	**3.51E+03**
24	2.38E+02	2.66E+02	2.28E+02	**2.02E+02**	2.05E+02	2.05E+02
25	2.84E+02	2.87E+02	2.85E+02	2.69E+02	2.82E+02	**2.59E+02**
26	2.06E+02	2.01E+02	2.01E+02	2.15E+02	2.02E+02	2.02E+02
27	9.85E+02	1.01E+03	9.62E+02	**3.26E+02**	4.60E+02	3.88E+02
28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 10: Performance of Rand/1, Rand/2, Best/2, RandToBest, and CurrentToBest at 50D when employed with adaptive control parameter model used in SHADE. 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. The best result is highlighted in bold.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	3.13E+07	4.93E+07	2.71E+07	1.79E+05	8.70E+04	**2.66E+04**
3	5.14E+08	2.87E+09	2.85E+06	2.67E+06	**6.84E+05**	8.80E+05
4	6.14E+04	6.13E+04	3.02E+04	1.24E+04	5.21E-01	**1.61E-03**
5	0.00E+00	0.00E+00	0.00E+00	6.92E-09	0.00E+00	0.00E+00
6	4.35E+01	4.35E+01	4.34E+01	4.38E+01	4.36E+01	**4.28E+01**
7	6.15E+01	8.33E+01	2.43E+01	**1.62E+01**	2.58E+01	2.33E+01
8	2.11E+01	2.11E+01	2.11E+01	2.09E+01	**2.08E+01**	2.09E+01
9	5.57E+01	5.63E+01	5.52E+01	**5.47E+01**	5.56E+01	5.54E+01
10	1.27E+01	5.63E+01	1.71E-01	2.50E-01	1.28E-01	**7.37E-02**
11	0.00E+00	4.74E-02	4.03E+00	0.00E+00	0.00E+00	0.00E+00
12	1.24E+02	1.48E+02	8.25E+01	6.46E+01	6.09E+01	**5.86E+01**
13	2.03E+02	2.07E+02	1.53E+02	**1.40E+02**	1.50E+02	1.45E+02
14	**8.33E-03**	1.76E+01	1.25E+01	3.69E-02	1.61E-02	3.45E-02
15	9.22E+03	9.03E+03	9.22E+03	8.79E+03	7.04E+03	**6.82E+03**
16	2.04E+00	2.12E+00	1.81E+00	1.42E+00	**1.22E+00**	1.28E+00
17	5.08E+01	5.08E+01	5.08E+01	5.08E+01	5.08E+01	5.08E+01
18	2.33E+02	2.64E+02	1.49E+02	**1.16E+02**	1.33E+02	1.37E+02
19	3.26E+00	3.59E+00	3.24E+00	2.77E+00	2.74E+00	**2.64E+00**
20	2.11E+01	2.14E+01	2.07E+01	**1.92E+01**	1.97E+01	1.93E+01
21	4.94E+02	**3.57E+02**	5.54E+02	8.15E+02	9.66E+02	8.45E+02
22	3.28E+01	2.35E+02	9.92E+01	**1.24E+01**	1.50E+01	1.33E+01
23	9.79E+03	1.04E+04	9.28E+03	8.58E+03	8.07E+03	**7.63E+03**
24	3.22E+02	3.40E+02	2.93E+02	**2.21E+02**	2.30E+02	2.34E+02
25	3.72E+02	3.74E+02	3.71E+02	3.54E+02	3.74E+02	**3.40E+02**
26	2.70E+02	2.28E+02	3.07E+02	3.08E+02	**2.06E+02**	2.58E+02
27	1.67E+03	1.71E+03	1.65E+03	**6.99E+02**	9.57E+02	9.36E+02
28	4.00E+02	4.00E+02	5.42E+02	4.00E+02	4.00E+02	4.58E+02

Table 13: Number of wins scored, out of 28, by all mutation strategies and SHADE at 10, 30, and 50 dimensions, respectively.

D	AD-Rand/1	AD-Rand/2	AD-Best/2	AD-RandToBest	AD-CurrToBest	SHADE
10	0	2	1	1	4	**10**
30	2	0	0	7	1	**10**
50	1	1	0	8	4	**9**

5 SA-SHADE Algorithm: Improvements with strategy adaptation

Motivated by the results presented in the previous section, we investigated the benefits of plugging a strategy adaptation module in the original SHADE algorithm.

We propose SA-SHADE, a memory based adaptive version of Differential Evolution wherein F, Cr, and mutation strategy are adapted during the search process. The basis of our paper is the SHADE algorithm [43], which adapts F and Cr but uses a fixed mutation strategy which is current-to-$p$best/1 with an optional external archive that was originally used in [36].

Algorithm 2 Pseudo-code for SA-SHADE

Set max number of generations G_{max}=0
Set current generation number, G=0
Set memory size M=100, reset rate R=0.1, counter k=1, and initialize external archive $A = \emptyset$
Initialize the mutation strategy pool P_{Ms}
Initialize a population of NP individuals $Pop = [X_1, X_2, ...X_{NP}]$ where every i^{th} individual is a D dimensional vector represented as X_i^j=[x_i^1, x_i^2 ... x_i^D] where $1 \leq j \leq D$. Restrict x_i^j to its minimum and maximum bounds as $x_{i,min}^j$ and $x_{i,max}^j$
Set all values in memory M_{Cr}, M_F to 0.5 and randomly initialize M_{Ms} with mutation strategies from the pool P_{Ms}
while $G \leq G_{max}$ or required error precision is not achieved **do**
    $S_{Cr} = \emptyset$ $S_F = \emptyset$; $S_{Ms} = \emptyset$
    for every target vector X_i in Pop **do**
        Select a random integer r from [1,M]
        Draw F from a cauchy distribution as $C(M_{F,r}, 0.1)$
        Draw Cr from a normal distribution as $N(M_{Cr,r}, 0.1)$
        Choose a mutation strategy M_i from $M_{Ms,r}$ indexed at r
        Produce a trial vector V_i^G, using the control parameters generated and the strategy selected above
        Select either the target vector or the trial vector based on their fitness values as

$$X_i^{G+1} = \begin{cases} V_i^G & \text{if } f(V_i^G) \leq f(X_i^G) \\ X_i^G & \text{otherwise} \end{cases}$$

        //update external archive A
        if $(f(V_i^G) < f(X_i^G))$ **then**
            Add X_i^G to external archive A
            Add Cr to the set S_{Cr}
            Add F to the set S_F
            Add M_i to the set S_{Ms}
        end if
    end for
    //update memories based on performance
    if $S_{Cr} \neq \emptyset$ and $S_F \neq \emptyset$ and $S_{Ms} \neq \emptyset$ **then**
        Add the most successful strategy, given by mode of successful strategies in S_{Ms} to M_{Ms}
        Update M_{Cr} and M_F based on S_{Cr} and S_F respectively
    end if
    //check for mutation memory reset
    if $G = (k \times R) \times G_{max}$ **then**
        Randomly initialize M_{Ms} with mutation strategies from the pool P_{Ms}
        Increase the counter k to $k+1$
    end if
    Increase the generation count G to $G+1$
end while

SA-SHADE and SHADE differ on three aspects.

- SA-SHADE adapts the mutation strategy during the search process, while SHADE uses a single mutation strategy throughout the search process.
- SA-SHADE uses the mode of successful mutation strategies to update its memory, which is slightly different from the way F and Cr are adapted in SHADE.
- The learned memory of successful strategies in SA-SHADE is wiped out after a certain number of function evaluations which is determined by the reset rate R. In SHADE, on the other hand, such reset is not performed for updating the memory of successful F and Cr values.

The operation of SA-SHADE is described as follows. First, we select the mutation strategies to be included in the pool P, which are, of course, selected on the basis of their relative strengths as described in Section 5.1. We then initialize an integer vector M_{Ms} of size 100, the memory containing successful mutation strategies after every generation, length of which is user controlled, with randomly selected mutation strategies from the pool, P. All of the mutation strategies in P are included in M_{Ms} at least once. Then, every individual solution is allowed to randomly choose a mutation strategy from the memory M_{Ms}. During the course of a generation, for every successful individual, just like F and Cr, we record, in another integer vector named suc-

cessful mutation strategies vector, S_{Ms}, the successful mutation strategies over a generation. Then, we store the most successful strategy of the generation, given by the mode of S_{Ms}, in the memory M_{Ms}.

The distinction between M_{Ms} and S_{Ms} is as follows. S_{Ms} maintains the record of successful mutation strategies within a generation. The most successful strategy in S_{Ms} is then recorded in M_{Ms}. This way the most successful of the mutation strategies are retained in the memory M_{Ms}. Successful F and Cr are recorded in their respective memories according to the mechanism proposed in the original SHADE algorithm [43]. This operation is repeated until a number of function evaluations are reached after which the memory M_{Ms} is re-initialized. This resetting of the memory is done to disallow any probabilistic bias created by the system towards a particular mutation strategy, and we show that this indeed proves useful. At the same time, it can be argued that the same reset scheme can be applied to F and Cr but our experiments show that this proves counterproductive in most of the cases. Hence, we have steered clear of using this reset method on F and Cr. SA-SHADE is summarized in Algorithm 2.

5.1 Choice of mutation strategies used in SA-SHADE

As proven in Section III-B, the choice of mutation strategy has a significant impact on the solution quality. A good mutation strategy is problem dependent, i.e., for the one which is successful on one landscape may prove adverse on others. There are some characteristics associated with every mutation strategy that may justify its use or otherwise. For example, double difference vector strategies like DE/rand/2/bin and DE/best/2/bin exhibit better diversity than DE/rand/1/bin and DE/best/1/bin [12], [15], [25], [24], making them more suitable on landscapes riddled with local minima. Strategies that use the best individual to generate mutant like DE/best/1/bin and DE/rand-to-best/1/bin tend to be greedy and score well on unimodal problems but their performance worsens on difficult and highly multimodal problems. A rotationally invariant strategy, DE/current-to-rand/1/, tends to do better on rotated problems [64]. The scheme DE/target-to-best/1/bin with neighborhood search proposed in [65], provides a good balance between exploration and exploitation.

Our pool P_{Ms} was obviously designed to contain the mutation strategies with diverse capabilities. We choose the following strategies for the listed reasons.

1. DE/rand/1/: Most widely used, less greedy but robust.
2. DE/rand/2/: Even though it has a poor record of achieving good solutions, it has the ability to improve the diversity of population as it is capable of generating more trial vectors due to presence of two difference vectors [8], [12].
3. DE/best/2/: Greedy but also has the ability of diversity improvement as it utilizes two different vectors [8], [12].
4. DE/current-to-*pbest*WithArchive/: Proposed in [36] and used in [43] which is the basis of our paper.
5. DE/current-rand-to-*pbest*/: A new mutation strategy that we experimented with, that uses the target vector as the base vector, a difference of one of the top 20% of best vectors and a randomly chosen vector, and another difference vector of two randomly chosen vectors. It has proven to be unstable sometimes but has the capability to negotiate local minima.

$$X_i = X_{target} + F \times (X_{pbest} - X_{r1}) + \\ + F \times (X_{r2} - X_{r3}) \tag{18}$$

We have incorporated a memory based adaptation mechanism into SA-SHADE on similar lines as the memory based adaptation of F and Cr. SHADE does not adapt the mutation strategy but only F and Cr.

5.2 Results

The comparative results of SA-SHADE with other variants at 30 dimensions are shown in Table 14. Table 15 lists the rank and p values obtained by SA-SHADE. It is clear that SA-SHADE is the top ranked algorithm among all the algorithms compared. SA-SHADE displays an improved performance compared to AD-Rand/1 (α = 0.05), AD-Rand/2 (α = 0.05), AD-Best2 (α = 0.05), AD-RandToBest (α = 0.05), and AD-CurrToBest (α = 0.1) while being highly competitive against

SHADE. The prefix AD denotes that the algorithm uses the adaptive control parameter mechanism. While in previous results, SHADE was not found to be better than Ad-RandToBest and AD-CurrToBest at any significance level, SA-SHADE improves upon SHADE and shows superior results.

Table 14: Relative ranks obtained by Rand/1, Rand/2, Best/2, RandToBest, CurrToBest, SHADE, and SA-SHADE at 30D. AD is prefixed with basic strategies to denote their usage with an adaptive control parameter model. The best rank and outperformed algorithms at α value = 0.05 are highlighted in bold.

Strategy	Rank	p_{Hoc}
AD-Rand/1	4.17	**0.00**
AD-Rand/2	4.50	**0.00**
AD-Best/2	4.05	**0.00**
AD-RandToBest	3.16	**0.03**
AD-CurrToBest	2.57	0.06
SHADE	2.53	0.26
SA-SHADE	**2.28**	-

Table 17: Relative ranks and p values obtained by SHADE against CoDE, EPSDE, JADE, and dynNP-jDE at 30D. The best rank and outperformed algorithms at α value = 0.05 are highlighted in bold.

Algorithm	Rank-10D	p value
SHADE	**2.30**	–
CoDE	2.91	0.15
JADE	2.50	0.64
dynNP-jDE	2.76	0.27
EPSDE	4.51	**0.00**

Table 17 shows the relative performance of SHADE with recently proposed state-of-the-art adaptive DE mechanisms. It is clear that SHADE, apart from EPSDE, is not statistically superior when compared to other algorithms at α = 0.05 or 0.1. SA-SHADE, with the results listed in Table 18, improves upon SHADE and shows statistically significant performance against EPSDE, CoDE, and dynNP-jDE while being highly competitive against JADE and SHADE. The best rank and outperformed algorithms at α value = 0.05 are highlighted in bold.

Table 18: Relative ranks and p values obtained by SA-SHADE against SHADE, JADE, dynNP-jDE, CoDE, and EPSDE at 30D.

Algorithm	Rank	p value
SA-SHADE	**2.46**	–
SHADE	2.92	0.35
JADE	3.14	0.15
dynNP-jDE	3.41	**0.05**
CoDE	3.60	**0.02**
EPSDE	5.44	**0.00**

While SA-SHADE (9 wins) is not statistically superior to SHADE (5 wins), it does improve upon SHADE on number of wins scored. The superior performance of SA-SHADE can be attributed to the strategy adaptation module as the underlying control parameter adaptation mechanism remains the same as SHADE. This also drives home the point that strategy adaptation is indeed a useful mechanism and should be used to improve the performance of DE variants.

5.3 Parameter study of the reset rate parameter used in SA-SHADE

Table 20: Average rankings of SA-SHADE at different memory reset rates (Friedman)

Reset Rate	Ranking
R=0.05	3
R=0.1	**2.0536**
R=0.2	3.0893
R=0.3	3.5179
R=0.4	3.4107

We performed a parameter analysis of the memory reset rate R to determine the most useful interval of clearing up the learned successful strategies in the system. The results obtained using five different reset rates are shown in Table 19 and the respective Friedman ranks obtained are listed in Table 20. These results show that the reset rate does have a crucial impact on the performance of SA-SHADE, and lower reset rates (0.05, 0.1) tend to be generally more useful than the higher ones (0.2, 0.3, 0.4). This may well be due to the bias created by the successful mutation strategies when they are retained in memory for longer times (higher reset rates).

Table 15: Performance of parameter adaptive Rand/1, Rand/2, Best/2, RandToBest, and CurrentToBest against SHADE, and SA-SHADE at 30D. 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. The best result is highlighted in bold.

F	Rand/1	Rand/2	Best/2	RandToBest	CurrToBest	SHADE	SA-SHADE
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	6.89E+06	1.72E+07	1.11E+07	3.78E+04	1.77E+04	9.00E+03	**8.15E+03**
3	4.36E+06	7.06E+07	5.89E+05	1.37E+05	1.31E+05	**4.02E+01**	1.19E+05
4	2.55E+04	2.25E+04	1.61E+04	2.45E+03	1.64E-01	**1.92E-04**	3.10E-02
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	1.42E+01	1.49E+01	1.23E+01	1.47E+01	8.42E+00	5.96E-01	**0.00E+00**
7	2.43E+01	3.59E+01	5.09E+00	**1.97E+00**	3.18E+00	4.60E+00	3.06E+00
8	2.09E+01	2.10E+01	2.09E+01	2.06E+01	2.06E+01	2.07E+01	2.07E+01
9	2.79E+01	2.81E+01	2.80E+01	2.74E+01	2.74E+01	2.75E+01	**2.69E+01**
10	1.23E+00	1.00E+01	8.69E-02	1.72E-01	1.07E-01	7.69E-02	**6.13E-02**
11	0.00E+00	0.00E+00	3.79E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	6.28E+01	6.13E+01	4.54E+01	2.52E+01	2.44E+01	2.30E+01	**1.82E+01**
13	9.42E+01	9.67E+01	8.57E+01	5.47E+01	5.20E+01	5.03E+01	**3.84E+01**
14	1.09E-02	3.60E+00	1.55E+00	2.08E-02	**8.92E-03**	3.18E-02	6.34E-01
15	4.72E+03	4.96E+03	4.57E+03	4.38E+03	3.22E+03	**3.22E+03**	3.24E+03
16	1.43E+00	1.96E+00	1.61E+00	**6.09E-01**	9.22E-01	9.13E-01	1.01E+00
17	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01
18	1.11E+02	1.42E+02	8.99E+01	**6.89E+01**	7.36E+01	7.25E+01	7.17E+01
19	1.79E+00	1.92E+00	1.80E+00	1.52E+00	1.38E+00	1.36E+00	**1.33E+00**
20	1.17E+01	1.19E+01	1.10E+01	1.04E+01	1.07E+01	1.05E+01	**1.03E+01**
21	**2.48E+02**	2.74E+02	2.81E+02	2.95E+02	2.97E+02	3.09E+02	2.81E+02
22	**8.19E+01**	2.07E+02	1.52E+02	1.08E+02	9.79E+01	9.81E+01	9.75E+01
23	4.83E+03	5.16E+03	4.44E+03	4.57E+03	3.74E+03	**3.51E+03**	3.58E+03
24	2.38E+02	2.66E+02	2.28E+02	2.02E+02	2.05E+02	2.05E+02	**2.01E+02**
25	2.84E+02	2.87E+02	2.85E+02	2.69E+02	2.82E+02	**2.59E+02**	2.80E+02
26	2.06E+02	2.01E+02	2.01E+02	2.15E+02	2.00E+02	2.02E+02	2.00E+02
27	9.85E+02	1.01E+03	9.62E+02	**3.26E+02**	4.60E+02	3.88E+02	4.11E+02
28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 16: Relative performance of SA-SHADE against state-of-the-art adaptive variants of DE at 30D. 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. The best result is highlighted in bold.

F	SA-SHADE	SHADE	CoDE	EPSDE	JADE	dynNP-jDE
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	8.15E+03	9.00E+03	9.78E+04	1.37E+06	**7.67E+03**	9.52E+04
F3	1.19E+05	**4.02E+01**	1.08E+06	1.75E+08	4.71E+05	1.71E+06
F4	3.10E-02	**1.92E-04**	8.18E-02	8.08E+03	6.09E+03	4.76E+01
F5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F6	**0.00E+00**	5.96E-01	4.16E+00	9.27E+00	2.07E+00	1.19E+01
F7	3.06E+00	4.60E+00	9.32E+00	5.88E+01	3.16E+00	**2.62E+00**
F8	2.07E+01	2.07E+01	2.08E+01	2.09E+01	2.09E+01	2.10E+01
F9	2.69E+01	2.75E+01	**1.45E+01**	3.50E+01	2.65E+01	2.20E+01
F10	6.13E-02	7.69E-02	**2.71E-02**	1.02E-01	4.04E-02	3.63E-02
F11	0.00E+00	0.00E+00	0.00E+00	1.95E-02	0.00E+00	0.00E+00
F12	**1.82E+01**	2.30E+01	3.98E+01	4.94E+01	2.29E+01	4.07E+01
F13	**3.84E+01**	5.03E+01	8.04E+01	7.68E+01	4.67E+01	7.10E+01
F14	6.34E-02	3.18E-02	3.60E+00	3.99E-01	2.86E-02	**9.39E-03**
F15	3.24E+03	**3.22E+03**	3.36E+03	6.75E+03	3.24E+03	4.39E+03
F16	1.01E+00	9.13E-01	**3.38E-01**	2.48E+00	1.84E+00	2.32E+00
F17	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01
F18	7.17E+01	7.25E+01	**6.69E+01**	1.37E+02	7.76E+01	1.35E+02
F19	1.33E+00	1.36E+00	1.61E+00	1.84E+00	1.44E+00	**1.27E+00**
F20	**1.03E+01**	1.05E+01	1.06E+01	1.30E+01	1.04E+01	1.13E+01
F21	**2.81E+02**	3.09E+02	3.02E+02	3.05E+02	3.04E+02	2.94E+02
F22	9.75E+01	9.81E+01	1.17E+02	3.09E+02	**9.39E+01**	1.03E+02
F23	3.58E+03	3.51E+03	3.56E+03	6.74E+03	**3.36E+03**	4.36E+03
F24	**2.01E+02**	2.05E+02	2.21E+02	2.91E+02	2.17E+02	2.04E+02
F25	2.80E+02	2.59E+02	2.57E+02	2.99E+02	2.74E+02	**2.55E+02**
F26	2.00E+02	2.02E+02	2.18E+02	3.56E+02	2.15E+02	2.00E+02
F27	4.11E+02	**3.88E+02**	6.20E+02	1.21E+03	6.70E+02	3.90E+02
F28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Table 19: Relative performance of SA-SHADE at different memory reset rates at 30D. 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. The best result is highlighted in bold.

F	R=0.05	R=0.1	R=0.2	R=0.3	R=0.4
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	1.29E+04	**8.15E+03**	1.06E+04	1.16E+04	6.18E+05
3	**5.37E+01**	1.19E+05	1.11E+08	4.48E+07	9.65E+07
4	**2.52E-03**	3.10E-02	3.29E+02	3.40E-01	6.78E+00
5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
6	8.80E-01	**0.00E+00**	5.43E-01	1.58E+00	8.80E-01
7	**2.78E+00**	3.06E+00	1.31E+01	9.38E+00	6.64E+00
8	2.08E+01	**2.05E+01**	2.07E+01	2.08E+01	2.08E+01
9	2.69E+01	2.69E+01	2.69E+01	2.72E+01	2.72E+01
10	**5.81E-02**	6.13E-02	7.49E-02	7.01E-02	7.69E-02
11	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	2.00E+01	**1.82E+01**	2.00E+01	2.10E+01	1.98E+01
13	5.40E+01	**3.84E+01**	4.16E+01	4.54E+01	4.49E+01
14	**1.32E-02**	6.34E-02	2.46E-02	2.68E-02	2.57E-01
15	3.24E+03	3.24E+03	3.26E+03	3.31E+03	3.35E+03
16	1.00E+00	1.01E+00	**8.74E-01**	1.00E+00	1.00E+00
17	3.04E+01	3.04E+01	3.04E+01	3.04E+01	3.04E+01
18	7.67E+01	7.17E+01	7.09E+01	7.14E+01	**7.06E+01**
19	1.45E+00	**1.33E+00**	1.44E+00	1.43E+00	1.42E+00
20	1.07E+01	**1.03E+01**	1.07E+01	1.06E+01	1.06E+01
21	2.88E+02	2.81E+02	2.83E+02	2.96E+02	**2.80E+02**
22	1.09E+02	**9.75E+01**	1.13E+02	1.02E+02	1.24E+02
23	3.72E+03	**3.58E+03**	3.67E+03	3.74E+03	3.84E+03
24	2.09E+02	**2.01E+02**	2.13E+02	2.16E+02	2.08E+02
25	2.83E+02	**2.80E+02**	2.86E+02	2.85E+02	2.85E+02
26	2.00E+02	**2.00E+02**	2.00E+02	2.00E+02	2.00E+02
27	8.57E+02	**4.11E+02**	6.95E+02	8.10E+02	7.64E+02
28	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02

Another interesting observation is that some of the functions respond well to low reset rates and some to higher ones, keeping in line with the no free lunch theorem [66]. A detailed analysis may be performed in the future to determine the generalized reset rate intervals depending upon the characteristics of a function.

6 Conclusions

In this paper, we presented the plausibility of integrating a strategy adaptation mechanism with a parameter adaptation mechanism. Given fixed control parameter settings, we first demonstrate, on a test suite of 28 benchmark functions, that no single mutation strategy performs significantly better than all other mutation strategies. Then, we show that similar results are observed in the presence of control parameter adaption. This built the case of automating the mutation strategies with or without control parameter adaptation. We then incorporated a strategy adaptation mechanism into a well known history based parameter adaptation mechanism, SHADE. We compare the enhanced version SA-SHADE with other well known adaptive mechanisms and show the competitive results obtained by SA-SHADE. SA-SHADE performs significantly better than well known adaptive variants, i.e., CoDE, EPSDE, and dynNP-jDE and is highly competitive compared to SHADE, and JADE. SHADE, though being higher ranked, was not found to be statistically significantly different when compared with CoDE and dynNP-jDE. Thus, SA-SHADE improves upon SHADE in this regard. The memory reset rate R of SA-SHADE is found to have crucial impact on its performance wherein lower reset rates are found to be relatively more conducive than higher ones. Another important conclusion that can be drawn from the results is that strategy adaptation is a useful mechanism both in presence and absence of control parameter adaptation, and we propose that it should always be used while optimizing with DE.

Future work includes investigating the impact of different adaptive strategies on multiple classes of benchmark functions and classifying strengths and weaknesses of each mechanism accordingly. Further to that, the utility of a population size adaptation mechanism into SA-SHADE is also proposed as future work.

References

[1] A. E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation, 3 (2), 124–141, 1999.

[2] G. Beni, J. Wang, Swarm Intelligence in Cellular Robotic Systems, in: Proceedings of the NATO Advanced Workshop on Robots and Biological Systems. Tuscany, Italy, 1989.

[3] P.J. Angeline, Adaptive and self-adaptive evolutionary computation, in: M. Palaniswami, Y. Attikiouzel, R.J. Marks, D.B. Fogel, T. Fukuda (Eds.), Computational Intelligence: A Dynamic System Perspective, IEEE Press, pp. 152–161, 1995.

[4] J. Gomez, D. Dasgupta, F. Gonazalez, Using adaptive operators in genetic search, in: Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO03), Chicago, Illinois, USA, pp. 1580–1581, 2003.

[5] B. R. Julstrom, What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm, in: Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, PA, USA, pp. 81-87, 1995.

[6] J. E. Smith, T.C. Fogarty, Operator and parameter adaptation in genetic algorithms, Soft Computing 1, pp. 81-87, 1997.

[7] A. Tuson, P. Ross, Adapting operator settings in genetic algorithms, Evolutionary Computation 6, pp. 161-184, 1998.

[8] R. M. Storn, 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, 1995.

[9] S. Das, P. N. Suganthan, Differential evolution - A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation, 15 (1), pp. 4–31, 2011.

[10] R. M. Storn, K. V. Price, Minimizing the real functions of the ICEC 1996 contest by differential evolution, in: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 842–844, 1996.

[11] J. Liu, J. Lampinen, On setting the control parameter of the differential evolution method, in: Proceedings of 8th Int. Conference Soft Computing (MENDEL), pp. 11–18, 2002.

[12] R. Gamperle, S. D. Muller, P. Koumoutsakos, A parameter study for differential evolution, NNA-FSFS-EC 2002, Interlaken, Switzerland, WSEAS, pp. 11–15, 2002.

[13] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, Natural Computing. Berlin, Germany: Springer-Verlag, 2003.

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

[15] S. Das, A. Konar, U. K. Chakraborty, Two improved Differential Evolution schemes for faster global search, in Proceedings of ACM-SIGEVO GECCO, pp. 991–998, 2005.

[16] H. A. Abbass, The self-adaptive pareto differential evolution algorithm, in Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, USA, 1, pp. 831–836, 2002.

[17] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation, 10 (6), pp. 646–657, 2006.

[18] A. Zamuda, J. Brest, Self-adaptive control parameters randomization frequency and propagations in differential evolution. Swarm and Evolutionary Computation, 25(1), pp. 72–99, 2015.

[19] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13, pp. 398–417, 2009.

[20] M. G. H. Omran, A. Salman, A. P. Engelbrecht, Self-adaptive differential evolution, in: Computational Intelligence and Security, PT 1, Proceedings Lecture Notes in Artificial Intelligence, pp. 192-199, 2005.

[21] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: Proceedings of the 9th International Conference on Soft Computing, Brno, pp. 41-46, 2003.

[22] J. Tvrdik, Adaptation in differential evolution: a numerical comparison, Applied Soft Computing 9, pp. 1149–1155, 2009.

[23] R. Mallipeddi, P. N. Suganthana, Q. K. Pan, M. F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied Soft Computing, 11 (2), pp. 1679–1696, 2011.

[24] R. Storn, K. Price, Differential evolution A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11, pp. 341–359, 1997.

[25] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing, pp. 76–83, 2000.

[26] J. Ronkkonen, S. Kukkonen, K. V. Price, Real parameter optimization with differential evolution, in Proceedings of IEEE Congress on Evolutionary Computation, 1, pp. 506–513, 2005.

[27] J. Liu, J. Lampinen, A Fuzzy Adaptive Differential Evolution Algorithm, in: Soft Computing, A Fusion of Foundations, Methodologies and Applications, 9 (6), pp. 448–462, 2005.

[28] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis Artificial Intelligence Review, 33 (1-2), pp. 61–106, 2010.

[29] J. Ronkkonen, J. Lampinen, On using normally distributed mutation step length for the differential evolution algorithm, in: Proceedings of the 9th Int. Conf. on Soft Computuing MENDEL, Brno, Czech Republic, pp. 11–18, 2003.

[30] A. K. Qin, P. N. Suganthan, Self-adaptive Differential Evolution Algorithm for Numerical Optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2005.

[31] M. M. Ali, A. Trn, Population set based global optimization algorithms: Some modifications and numerical studies, Journal of Computers and Operations Research, 31 (10), pp. 1703–1725, 2004.

[32] U. K. Chakraborty, Advances in Differential Evolution, in: Differential Evolution Research-Trends and Open Questions, Springer, pp. 11–12, 2008.

[33] D. Dawar, S. A. Ludwig, Differential evolution with dither and annealed scale factor, in: Proceedings of the IEEE Symposium Series on Computational Intelligence, Orlando, Florida, U.S.A., pp. 1–8, 2014.

[34] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, Soft Computing - A Fusion of Foundations, Methodologies and Applications, 10 (8), pp. 673–686, 2006.

[35] J. Brest, M. S. Mauec, Population size reduction for the differential evolution algorithm, Applied Intelligence, 29 (3), pp. 228–247, 2008.

[36] J. Zhang, A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Transaction on Evolutionary Computation, 13 (5), pp. 945-958, 2009.

[37] E. Mezura-Montes, J. Velazquez-Reyes, C. A. Coello Coello, A comparative study of differential evolution variants for global optimization, in GECCO, pp. 485–492, 2006.

[38] F. Peng, K. Tang, G. Chen, X. Yao, Multi-start JADE with knowledge transfer for numerical optimization, in: Proceedings of the IEEE CEC, pp. 1889–1895, 2009.

[39] Z. Yang, J. Zhang, K. Tang, X. Yao, A. C. Sanderson, An adaptive coevolutionary differential evolution algorithm for large-scale optimization, in: Proceedings of the IEEE CEC, pp. 102–109, 2009.

[40] W. Gong, Z. Cai, C. X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics, PartB, 41 (2), pp. 397–413, 2011.

[41] J. Zhang, V. Avasarala, A. C. Sanderson, T. Mullen, Differential evolution for discrete optimization: An experimental study on combinatorial auction problems, in: Proceedings of the IEEE CEC, pp. 2794–2800, 2008.

[42] J. Zhang, A. C. Sanderson, Self-adaptive multiobjective differential evolution with direction information provided by archived inferior solutions, in: Proceedings of the IEEE CEC, pp. 2801–2810, 2008.

[43] R. Tanabe, A. Fukunaga, Success-History Based Parameter Adaptation for Differential Evolution, in: Proceedings of the IEEE CEC, pp. 71–78, 2013.

[44] R. Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: Proceedings of the IEEE CEC, pp. 1952–1959, 2013.

[45] A. Auger, N. Hansen, A Restart CMA Evolution Strategy With Increasing Population Size, in: Proceedings of the IEEE CEC, pp. 1769–1776, 2005.

[46] C. Garca-Martnez, M. Lozano, F. Herrera, D. Molina, A. M. Sanchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, European Journal of Operations Research, 185 (3), pp. 1088–1113, 2008.

[47] M. A. M. de Oca, T. St utzle, K. V. den Enden, M. Dorigo, Incremental Social Learning in Particle Swarms, IEEE Transactions on Systems, Man, and Cybernetics, PartB, 41 (2), pp. 368–384, 2011.

[48] J. L. J. Laredo, C. Fernandes, J. J. M. Guervos, C. Gagne, Improving Genetic Algorithms Performance via Deterministic Population Shrinkage, in: Proceedings of the GECCO, pp. 819–826, 2009.

[49] R. Tanabe, A. Fukunaga, Improving the Search Performance of SHADE Using Linear Population Size Reduction, in: Proceedings of the IEEE CEC, pp. 1658–1665, 2014.

[50] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, V. Zumer, Highdimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction, in: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2032–2039, 2008.

[51] A. Zamuda, J. Brest. Population Reduction Differential Evolution with Multiple Mutation Strategies in Real World Industry Challenges. Artificial Intelligence and Soft Computing – ICAISC 2012, 7269, pp. 154–161, 2012.

[52] J. J. Liang, B.Y. Qu, P. N. Suganthan, 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.

[53] 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.

[54] 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, pp. 674–701, 1937.

[55] D. J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed., Chapman and Hall/CRC, 2006.

[56] J. H. Zar, Biostatistical Analysis, Prentice Hall, 2009.

[57] Y. Hochberg, A sharper Bonferroni procedure for multiple tests of significance, Biometrika, pp. 800–803, 1988.

[58] J. Derrac, S. Garca, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation, vol 1, pp. 3–18, 2011.

[59] B. V. Babu, S. A. Munawar, Optimal design of shell-and-tube heat exchangers bu different strategies of Differential Evolution, Technical Report PILANI -333 031, Department of chemical engineering, BITS, Rajasthan, India, 2001.

[60] J. Vesterstrom, R. A. Thomson, Comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, 1980–1987, 2004.

[61] X. F. Xie, W. J. Zhang. SWAF: Swarm algorithm framework for numerical optimization, in: Proceedings of the Genetic Evolutionary Computation Conference, Part I, pp. 238–250, 2004.

[62] A. Zamuda, J. Brest, B. Bokovic, V. umer. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution, in: Proceedings of the 2008 IEEE World Congress on Computational Intelligence, pp. 3719–3726, 2008.

[63] Z. Yang, K. Tang, X. Yao. Self-adaptive differential evolution with neighborhood search. In Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1110–1116, 2008.

[64] A. Iorio, X. Li, Solving rotated multi-objective optimization problems using differential evolution, in: Australian Conference on Artificial Intelligence, Cairns, Australia, pp. 861–872, 2004.

[65] S. Das, A. Abraham, U.K. Chakraborthy, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13, pp. 526–553, 2009.

[66] D. H. Wolpert, W. G. Macready, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67–82, 1997.

Deepak Dawar received his B.Tech degree in Electronics and Communication Engineering from Kurukshetra University in 2007, and has worked in software development industry for a little above four years as a software engineer, and later as a program analyst. He received his M.S. and Ph.D. in computer science from North Dakota State University in 2013, and 2016 respectively.
He joined Miami in 2016 as an assistant professor, and has experience in teaching an ensemble of computing courses including Fundamental Problem Solving, Operating Systems, Computer Networking, Web Programming, Java, Visual Basic, Micro-Computer Packages etc. His current research interests include evolutionary computing, swarm intelligence, pattern recognition, and video based vehicle classification.

Simone A. Ludwig is an Associate Professor of Computer Science at North Dakota State University, USA. She received her PhD degree and MSc degree with distinction from Brunel University, UK, in 2004 and 2000, respectively. Before starting her academic career she worked several years in the software industry. Her research interests lie in the area of computational intelligence including swarm intelligence, evolutionary computation, neural networks, and fuzzy reasoning.